MINISTRY OF EDUCATION

ROBOTICS
For Senior High Schools

TEACHER MANUAL

YEAR 1 - BOOK 2

NATIONAL COUNCIL FOR CURRICULUM & ASSESSMENT OF MINISTRY OF EDUCATION

MINISTRY OF EDUCATION

REPUBLIC OF GHANA

Robotics

For Senior High Schools

Teacher Manual

Year One - Book Two

NATIONAL COUNCIL FOR CURRICULUM & ASSESSMENT OF MINISTRY OF EDUCATION

ROBOTICS TEACHER MANUAL

Enquiries and comments on this manual should be addressed to:
The Director-General
National Council for Curriculum and Assessment (NaCCA)
Ministry of Education
P.O. Box CT PMB 77
Cantonments Accra

Telephone: 0302909071, 0302909862
Email: info@nacca.gov.gh
website: www.nacca.gov.gh

CONTENTS

INTRODUCTION 1

Learner-Centred Curriculum 1

Promoting Ghanaian Values 1

Integrating 21st Century Skills and Competencies 1

Balanced Approach to Assessment - not just Final External Examinations 1

An Inclusive and Responsive Curriculum 2

Social and Emotional Learning 2

Philosophy and vision for each subject 2

SUMMARY SCOPE AND SEQUENCE 3

SECTION 5: ROBOT DESIGN SOFTWARE 4

Strand: Robot Design Methodologies 4

Sub-Strand: Tools and Applications for Robot Design 4

Theme or Focal Area: Exploring Tools & Apps for Robot Design: Modelling, Programming and Simulation 6

Theme or Focal Area: Virtual Robot Design and Simulation: Exploring Mechanics and Testing 12

Theme or Focal Area: 3D Printing and CAD Modelling for Robotic Systems 18

Theme or Focal Area: 3D Printing with G-Codes: From CAD Modelling to Physical Prototypes 22

SECTION 6: ROBOT CONSTRUCTION AND PROGRAMMING 27

Strand: Robot Construction & Programming 27

Sub-Strand: Higher Order Design Thinking 27

Theme or Focal Area: Higher order Design Thinking: Flowchart Diagrams for Algorithm Implementation 29

Theme or Focal Area: Algorithmic Problem-Solving in Robotics: Pseudocodes and Flowchart Diagrams 32

SECTION 7: ROBOT CONSTRUCTION 35

Strand: Robot Construction & Programming 35

Sub-Strand: Robot Construction 35

Theme or Focal Area: Robot Construction - Designing Stable Structures and Understanding Mass and Centre of Gravity 37

Theme or Focal Area: Building and Testing Robot Structures: Ensuring Stability and Force Resistance 41

Theme or Focal Area: Creating Robots with Basic Mechanics - Exploring Gears, Vehicles and Moving Mechanisms 45

SECTION 8: PROGRAMMING ROBOTS 60

Strand: Robot Construction & Programming 60

Sub-Strand: Programming Robots 60

Theme or Focal Area: Introduction to Programming with Block-Based Coding 62

Theme or Focal Area: Creating Computer Programmes from Flowcharts 66

Theme or Focal Area: Fundamentals of Control Principles in Automation and Robotics - Feedback and Non-Feedback Loop Systems 79

ACKNOWLEDGEMENTS 87

INTRODUCTION

The National Council for Curriculum and Assessment (NaCCA) has developed a new Senior High School (SHS), Senior High Technical School (SHTS) and Science, Technology, Engineering and Mathematics (STEM) Curriculum. It aims to ensure that all learners achieve their potential by equipping them with 21st Century skills, competencies, character qualities and shared Ghanaian values. This will prepare learners to live a responsible adult life, further their education and enter the world of work.

This is the first time that Ghana has developed an SHS Curriculum which focuses on national values, attempting to educate a generation of Ghanaian youth who are proud of our country and can contribute effectively to its development.

This Book Two of the Teacher Manual for Robotics covers all aspects of the content, pedagogy, teaching and learning resources and assessment required to effectively teach Year One of the new curriculum. It contains information for the second 12 weeks of Year One. Teachers are therefore to use this Teacher Manual to develop their weekly Learning Plans as required by Ghana Education Service.

Some of the key features of the new curriculum are set out below.

Learner-Centred Curriculum

The SHS, SHTS, and STEM curriculum places the learner at the center of teaching and learning by building on their existing life experiences, knowledge and understanding. Learners are actively involved in the knowledge-creation process, with the teacher acting as a facilitator. This involves using interactive and practical teaching and learning methods, as well as the learner's environment to make learning exciting and relatable. As an example, the new curriculum focuses on Ghanaian culture, Ghanaian history, and Ghanaian geography so that learners first understand their home and surroundings before extending their knowledge globally.

Promoting Ghanaian Values

Shared Ghanaian values have been integrated into the curriculum to ensure that all young people understand what it means to be a responsible Ghanaian citizen. These values include truth, integrity, diversity, equity, self-directed learning, self-confidence, adaptability and resourcefulness, leadership and responsible citizenship.

Integrating 21st Century Skills and Competencies

The SHS, SHTS, and STEM curriculum integrates 21st Century skills and competencies. These are:

- **Foundational Knowledge:** Literacy, Numeracy, Scientific Literacy, Information Communication and Digital Literacy, Financial Literacy and Entrepreneurship, Cultural Identity, Civic Literacy and Global Citizenship
- **Competencies:** Critical Thinking and Problem Solving, Innovation and Creativity, Collaboration and Communication
- **Character Qualities:** Discipline and Integrity, Self-Directed Learning, Self-Confidence, Adaptability and Resourcefulness, Leadership and Responsible Citizenship

Balanced Approach to Assessment - not just Final External Examinations

The SHS, SHTS, and STEM curriculum promotes a balanced approach to assessment. It encourages varied and differentiated assessments such as project work, practical demonstration, performance assessment, skills-based assessment, class exercises, portfolios as well as end-of-term examinations and final external assessment examinations. Two levels of assessment are used. These are:

- Internal Assessment (30%) – Comprises formative (portfolios, performance and project work) and summative (end-of-term examinations) which will be recorded in a school-based transcript.
- External Assessment (70%) – Comprehensive summative assessment will be conducted by the West African Examinations Council (WAEC) through the WASSCE. The questions posed by WAEC will test critical thinking, communication and problem solving as well as knowledge, understanding and factual recall.

The split of external and internal assessment will remain at 70/30 as is currently the case. However, there will be far greater transparency and quality assurance of the 30% of marks which are school-based. This will be achieved through the introduction of a school-based transcript, setting out all marks which learners achieve from SHS 1 to SHS 3. This transcript will be presented to universities alongside the WASSCE certificate for tertiary admissions.

An Inclusive and Responsive Curriculum

The SHS, SHTS, and STEM curriculum ensures no learner is left behind, and this is achieved through the following:

- Addressing the needs of all learners, including those requiring additional support or with special needs. The SHS, SHTS, and STEM curriculum includes learners with disabilities by adapting teaching and learning materials into accessible formats through technology and other measures to meet the needs of learners with disabilities.
- Incorporating strategies and measures, such as differentiation and adaptative pedagogies ensuring equitable access to resources and opportunities for all learners.
- Challenging traditional gender, cultural, or social stereotypes and encouraging all learners to achieve their true potential.
- Making provision for the needs of gifted and talented learners in schools.

Social and Emotional Learning

Social and emotional learning skills have also been integrated into the curriculum to help learners to develop and acquire skills, attitudes, and knowledge essential for understanding and managing their emotions, building healthy relationships and making responsible decisions.

Philosophy and vision for each subject

Each subject now has its own philosophy and vision, which sets out why the subject is being taught and how it will contribute to national development. The Philosophy and Vision for Robotics is:

Philosophy: The next generation of creators and technology developers can be empowered through observation, curiosity and exposure to related robotic concepts and opportunities that leverage practical activities in a learner-centered environment leading to global and local ("glocal") relevance.

Vision: A skilled learner armed with 21st century skills and competencies in critical thinking, designing, and development of robot-based solutions for increasingly complex societal problems.

SUMMARY SCOPE AND SEQUENCE

S/N	STRAND	SUB-STRAND	YEAR 1			YEAR 2			YEAR 3		
			CS	**LO**	**LI**	**CS**	**LO**	**LI**	**CS**	**LO**	LI
1.	Principles of Robotic Systems	Robots and Society	2	2	3	2	2	4	2	2	4
		Robot Control Principles	2	2	4	2	2	4	3	3	5
		Sensors and Actuators	2	2	4	2	2	4	1	1	2
2.	Robot Design Methodologies	Digital and Analogue System Design	2	2	4	2	2	3	1	1	2
		Tools and Apps for Robot Design	1	1	2	1	1	1	-	-	-
3.	Robot Construction and Programming	Higher Order Design Thinking	1	1	2	1	1	1	-	-	-
		Robot Construction	2	2	3	2	2	2	1	1	1
		Programming Robot	-	-	-	2	2	4	-	-	-
Total			**12**	**12**	**22**	**14**	**14**	**23**	**8**	**8**	**14**

Overall Totals (SHS 1 – 3)

Content Standards	**34**
Learning Outcomes	**34**
Learning Indicators	**59**

SECTION 5: ROBOT DESIGN SOFTWARE

Strand: **Robot Design Methodologies**

Sub-Strand: Tools and Applications for Robot Design

Learning Outcome: *Effectively use virtual platforms and simulation tools to design and test robots.*

Content Standards:

1. Demonstrate abilities to use Integrated Development Environments.
2. Use of modelling and simulation tools needed for the design and testing of robots.
3. Demonstrate Skills in the use and management of 3D printers.

INTRODUCTION AND SECTION SUMMARY

This section will explore essential tools and applications that aid in the design and testing of robots. Integrated Development Environments (IDEs), modelling and simulation tools are crucial in developing robotics projects. This section focuses on virtual robot design and simulation to evaluate robot performance. It introduces learners to 3D printing and Computer-Aided Design (CAD) modelling for robotic systems. This section aims to learn how to create custom parts for robotic systems. This exciting session will explore the final steps of the 3D printing process, focusing on using relevant intermediate tools to prepare CAD-modelled files into G-codes and printing the designs using a 3D printer.

The weeks covered by the section are:

Week 13: Explore features of selected modelling, programming and simulation tools useful for the design of robots

Week 14: Design robots using virtual platforms and use simulation tools and programming IDEs to test the mechanics of the designed robots.

Week 15: Use a CAD tool to model parts of robotic systems.

Week 16: Use relevant intermediate tools to prepare modelled files into g-codes and print the designs using a 3D Printer.

SUMMARY OF PEDAGOGICAL EXEMPLARS

This section uses different teaching strategies to engage learners and equip them with a good understanding of robot simulation software. Week 13 utilises initiating talk for learning to help learners discuss their experiences with robot simulation software and their use. Talk for learning is also employed to help learners explore the features of robot simulation software and discuss and compare their observations.

In week 14, enquiry-based learning guides learners in exploring virtual platforms such as V-REP or ROS with a teacher-led demonstration. Learners are asked to create comparison charts to help improve their critical thinking and research skills. Problem-based learning is also employed, where learners are given a mini-project and given resources to help them complete the project.

A Guided design challenge is utilised in week 15, where the teacher guides learners in a step-by-step process of designing a simple gripper attachment that can be integrated with the robot arm model.

Learners are introduced to interactive tutorials which use gamification elements such as points and badges to motivate them as they learn the basic functionalities of the software.

ASSESSMENT SUMMARY

Following each thematic area in this section, assessments gauge learner learning. These come in two forms: learning tasks and key assessments. Learning tasks, primarily formative, focus on solidifying understanding and acquiring new knowledge or skills. Facilitators guide these activities to enhance the learning process. In contrast, key assessments, typically summative, evaluate learner mastery after instruction. These are often given as homework or quizzes outside of class. Instructors have the flexibility to choose the assessment types that best suit their learners and learning objectives. However, it is advisable that instructors, at least, guide learners to do one of the learning tasks.

Week 13

Learning Indicator: *Explore features of selected modelling, programming and simulation tools useful for the design of robots.*

Theme or Focal Area: Exploring Tools & Apps for Robot Design: Modelling, Programming and Simulation

Introduction

This lesson will explore essential tools and applications that aid in the design and testing of robots. Integrated Development Environments (IDEs), modelling and simulation tools are crucial in developing robotics projects. Understanding these tools will enable learners to design, programme and simulate robotic systems efficiently. It will explore the features and functionalities of selected tools, providing learners with valuable insights into their applications in robotics.

1. **Integrated Development Environments (IDEs) for Robotics**:

   Integrated Development Environments (IDEs) are software platforms that provide a comprehensive set of tools for programming and managing robotic projects. It is a single software programme that combines all the tools one will need to write, test and debug code. Imagine it as a one-stop shop for your robot programming needs.

   The following are some benefits of using IDEs:

   a. **Code editing:** Write your code with syntax highlighting (colours for different parts of the code) and auto-completion (suggestions as you type).

   b. **Compiling and debugging:** Easily compile your code (turn it into instructions the robot understands) and troubleshoot any errors that might arise.

   c. **Simulation and testing:** Some IDEs allow you to simulate your robot's behaviour in a virtual environment before running it on the real hardware.

   Some popular IDEs for robotics include:

   a. **Lego Education Spike App**: This tool is for writing programmes for the Lego Spike Robot. The app includes introductory material, lessons, building instructions and a series of coding experiences that progress from icon- and word-block coding based on Scratch to text-based coding based on Python.

   b. **EV3 Classroom**: This is a tool for writing programmes for LEGO EV3 kits. It uses an intuitive icon-based environment which connects to the EV3 robot for instructions.

   c. **Arduino IDE**: Arduino IDE is widely used in robotics to programme Arduino microcontrollers. It offers a user-friendly interface for writing, compiling and uploading code to Arduino microcontroller boards.

   d. **Visual Studio Code (VS Code):** VS Code provides an advanced IDE for embedded systems, including robotics projects. It supports multiple microcontroller platforms, offering rich code editing and debugging features.

   e. **MATLAB/Simulink**: MATLAB is a powerful tool for numerical computing and data analysis. Simulink, an extension of MATLAB, is widely used for modelling and simulating robotic systems.

2. **Modelling and Simulation Tools for Robotics**:

   Modelling and simulation tools are like virtual playgrounds for your robot! They allow you to create a digital representation (model) of your robot and its environment and simulate its movements and behaviour before deploying it in the real world. Modelling and Simulation tools are essential for visualising and testing robotic systems before implementation.

   The following are some benefits of using them:

   a. **Safer testing:** Experiment with different programming approaches without risking damage to your robot or its surroundings.

   b. **Faster development:** Identify and fix errors early in the development process, saving time and resources.

   c. **Optimise performance:** Test different configurations and settings to find the most efficient way for your robot to operate.

   d. **Visualise robot behaviour:** See how your robot will interact with its environment before building it.

   Some widely used tools for modelling and simulation in robotics include:

   a. **BrickLink Studio**: It is a powerful 3D modelling software specifically designed for creating virtual LEGO models. It provides an intuitive user interface that makes it easy for users to navigate and access various tools and features and includes an extensive library of LEGO parts, elements and colours, allowing users to easily find and select the components they need to build their models. The library is regularly updated to include the latest LEGO parts and sets.

   b. **ROS (Robot Operating System)**: ROS is an open-source framework that provides a collection of software libraries and tools for developing robotic applications. It supports simulation, visualisation and communication between robot components. (Fairchild & Harman 2016)

   c. **Gazebo**: Gazebo is a robust 3D robot simulation environment that works seamlessly with ROS. It allows users to create virtual environments to test and validate robotic algorithms and interactions. (Open Robotics, 2024)

   d. **Webots:** Webots is a professional robot simulation software used to model, simulate and optimise robotic systems. It offers a user-friendly interface and supports various robot models.

   e. **Tinkercad**: Tinkercad is a software that creates 3D models of robots and then exports them for 3D printing. It offers a lot of flexibility in robot modelling and simulation.

3. **Features of Selected Modelling, Programming and Simulation Tools**:

   To explore the selected tools, we will focus on ROS and Gazebo, Tinkercad and The Virtual Robotics Toolkit and Arduino IDE:

   a. **OS and Gazebo**: ROS provides a modular and distributed architecture that facilitates easy integration of sensors, actuators and other components. Gazebo offers a realistic simulation environment, allowing users to visualise robots and test their interactions in 3D virtual worlds.

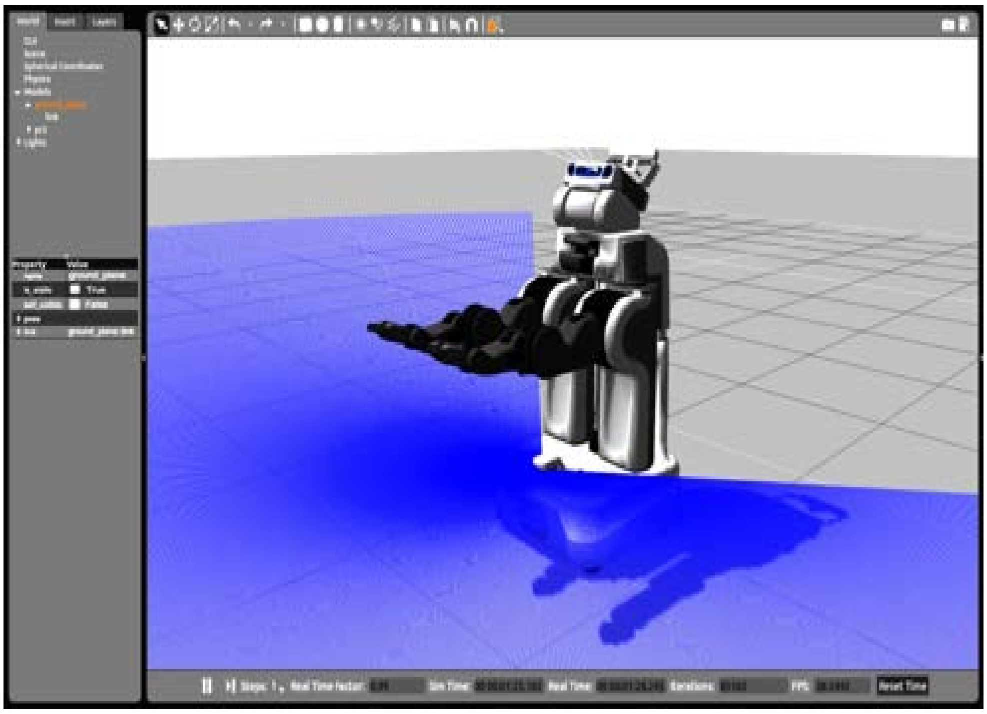


Fig. 13.1: *ROS Simulation*

b. **Tinkercad**: Tinkercad is a web-based computer-aided design (CAD) software developed by Autodesk. It is widely used for creating 3D models, circuits and simulations, making it a versatile tool for various design and engineering projects.

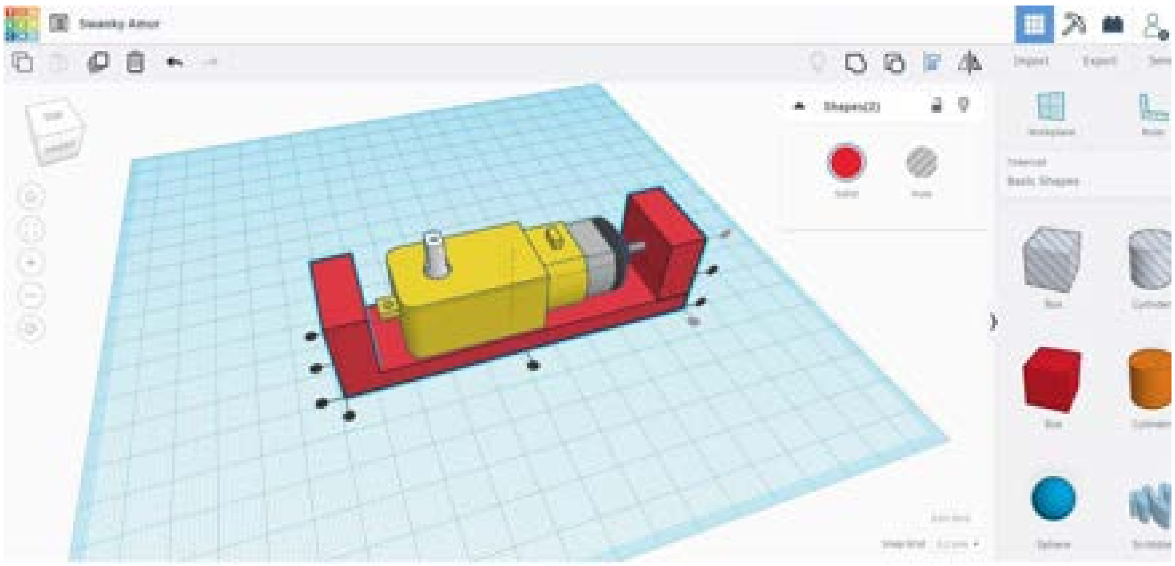


Fig. 13.2: *Tinkercad*

c. **The Virtual Robotics Toolkit:** The Virtual Robotics Toolkit is a software platform designed to provide a simulated environment for learning and practising robotics concepts without the need for physical hardware.

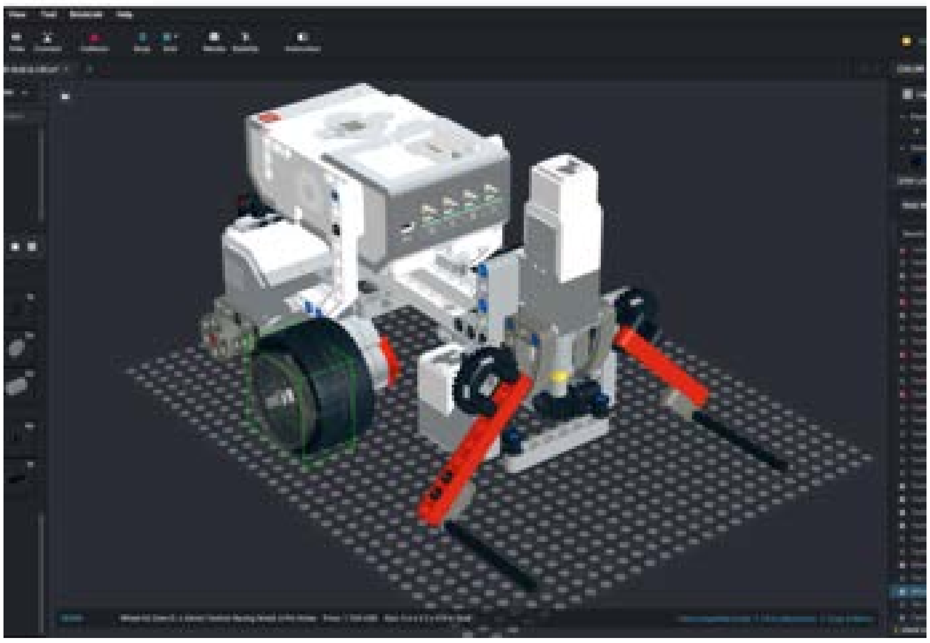
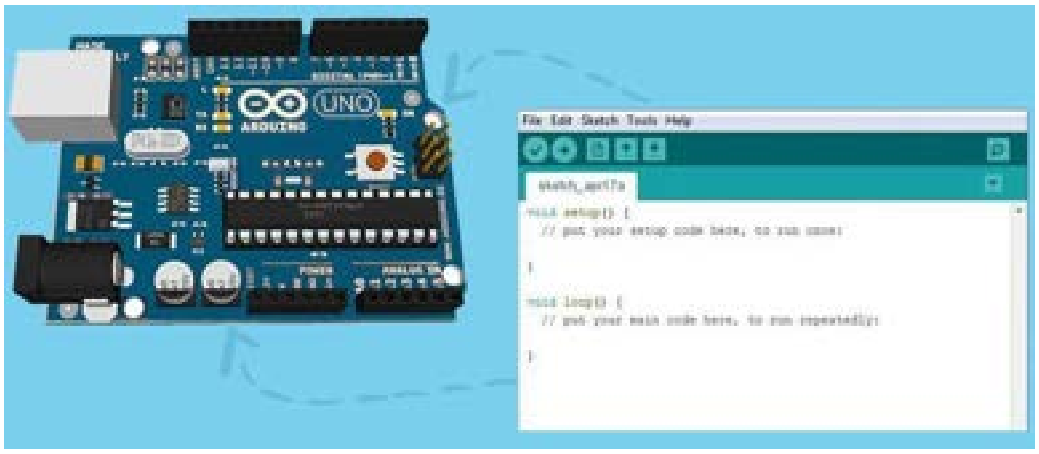


Fig. 13.3: *Virtual Robotics Toolkit*

d. **The Arduino IDE**: This IDE contains a text editor for writing code, a text console, a toolbar with buttons which have common functions and a series of menus. It connects to the Arduino hardware to upload programmes and communicate with them.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig 13.4: *Arduino IDE*

Learning Tasks

Depending on the available time or resources, administer one or more of the following learning tasks to help learners reinforce understanding and acquire new knowledge or skills.

1. Learners install robot design tools with teacher guidance.
2. Learners explore functionalities through online tutorials.
3. Learners delve deeper into specific tools (ROS, Tinkercad, Arduino IDE).
4. Learners complete a basic task using their chosen tool (simulation or programming).
5. Learners share their experiences and observations with the class.

Pedagogical Exemplars

This lesson aims to introduce learners to robot simulation software tools. Consider the following keynotes when administering the suggested pedagogical approaches in the curriculum:

1. **Initiating Talk for Learning:** In a tiered or scaffolded manner, address the learner groups' identified needs. You can use a good mixture of the following to address their needs:
   a. **K (Know):** Use simple prompts such as "Have you used a software before?" or "Can you name some software you have used before?" List learner responses on the board and categorise them broadly (e.g. If Yes, probe OS and software). Ask learners to share specific examples of design software they have encountered and their functionalities (e.g. Arduino, ROS). Discuss the level of difficulty they have in learning how to use these softwares. Have learners brainstorm and categorise different types of robot software based on their ease of use (e.g. ROS, Gazebo). Discuss the benefits and advantages of each software over the other.

   b. **(Want to Know):** Encourage them to ask questions about robotic software using prompts such as "How are robots designed?" or "What are simulations?" List these questions and address some of them briefly, piquing their curiosity for the lesson. Encourage them to ask more detailed questions about how to install and use the IDE and other software to design robots. Encourage them to ask questions about the future of robotic design and how the software can be made easier to understand and use.

   c. **L (Learn):** Briefly introduce the concept of robot design and how software is used to design robots. Introduce the key features of robot software (coding, drag-and-drop, etc.) and how these features contribute to their functionalities. Delve deeper into the technological advancements (AI, connectivity) in robot simulation. Briefly discuss the ethical considerations surrounding the design and simulation of robots.

2. **Talk for Learning:** Similarly, address the identified needs of learners with different readiness, interests and learning profiles. You can use a mixture of the suggested approaches below:
   a. Use very simple and clear examples with pictures or videos of robotic software. Examples could include a snippet of VS Code, a snippet of Gazebo and a video of Arduino IDE.

   b. Provide ample time for individual thinking before pairing up. Encourage learners to differentiate features of software that they notice and explain their reasoning in simple terms during the sharing phase. The facilitator can circulate and offer guidance where needed.

   c. Show learners the interfaces of IDEs and Simulation software. Give learners a moderate amount of time for individual thinking. Ask learners to differentiate in a table the differences between IDEs and Simulation software. The facilitator can ask clarifying questions to promote deeper discussion within pairs.

   d. Show learners the output files of IDEs and Simulation software. Provide sufficient time for individual thinking and analysis. Encourage learners to have a more in-depth discussion about the functionalities and limitations of each kind of software. The facilitator can ask probing questions to encourage critical thinking and analysis during the sharing phase.

Key Assessment

Assessment Level 1: What is the main purpose of a robot simulation tool?

Assessment Level 2: Explain why modelling and simulation tools are essential for robotic projects.

Assessment Level 3: Imagine you are designing a robot that sorts recycling materials. Which tool would be most useful at the beginning of the design process, and why?

Assessment Level 4: Design a flowchart that outlines the steps involved in using an IDE, modelling tool, and simulation tool together for a basic robot development process.

Conclusion

Integrated Development Environments, Modelling and Simulation tools are essential assets in the design and testing of robots. Learners have gained valuable insights into their applications and functionalities by exploring the features of selected tools such as ROS, Gazebo, Virtual Robotics Kit and Arduino IDE. These tools empower you to create, programme and simulate robotic systems efficiently, ensuring the successful development and deployment of robotic projects. As you progress in your robotics journey, mastering these tools will be instrumental in tackling more complex challenges and building innovative robots.

Week 14

Learning Indicator: *Design robots using virtual platforms and use simulation tools and programming IDEs to test the mechanics of the designed robots.*

Theme or Focal Area: Virtual Robot Design and Simulation: Exploring Mechanics and Testing

Introduction

This exciting session will delve into the world of virtual robot design and simulation. Virtual platforms and simulations provide a powerful and cost-effective way to design and test robots before physical implementation. By exploring virtual environments, we can examine the mechanics of designed robots and perform simulations to evaluate their performance. Let's discover how virtual tools open up new possibilities in robotics design and testing.

1. **Virtual Platforms for Robot Design**:
   Virtual platforms are computer-based environments that allow us to create and interact with virtual robots. These platforms offer various features, including 3D modelling, physics engines and simulation capabilities. Some popular virtual platforms for robot design include:
   a. **Bricklink**: Bricklink Studio is a Lego-based software and it provides a versatile environment for creating, visualising and simulating complex robotic systems.

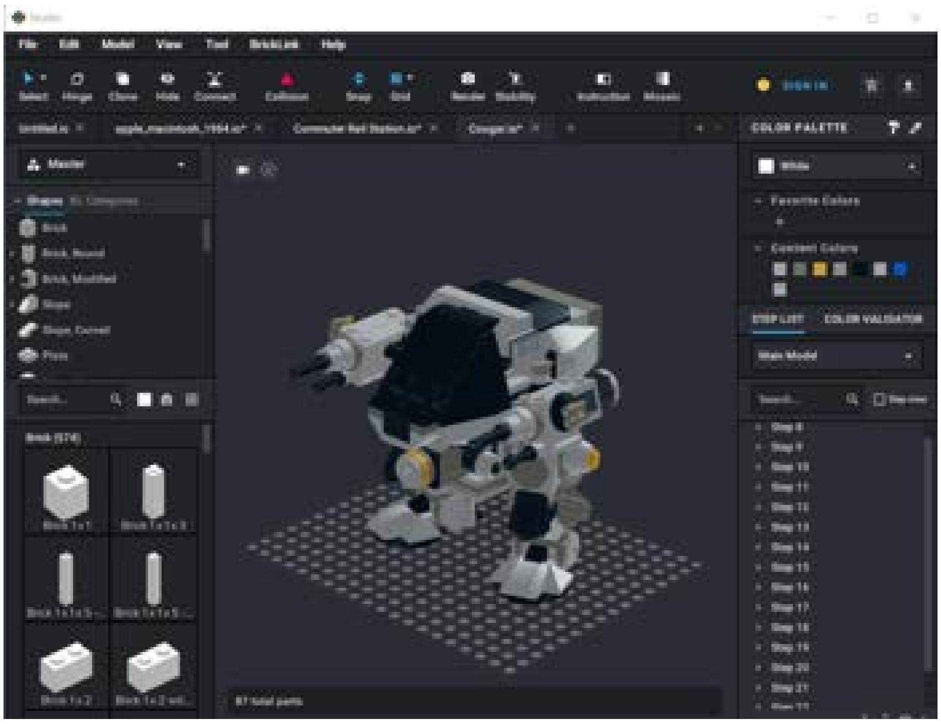


Fig. 14.1: *Bricklink Interface (Simulation). (Fileinfo, 2021)*

b. **Tinkercad**: Tinkercad by Autodesk offers a user-friendly interface for designing and simulating robotic systems. It offers 3D design of robotic parts, supports programming robots with block-based coding and uses Autodesk Fusion for the printing of 3D designs.

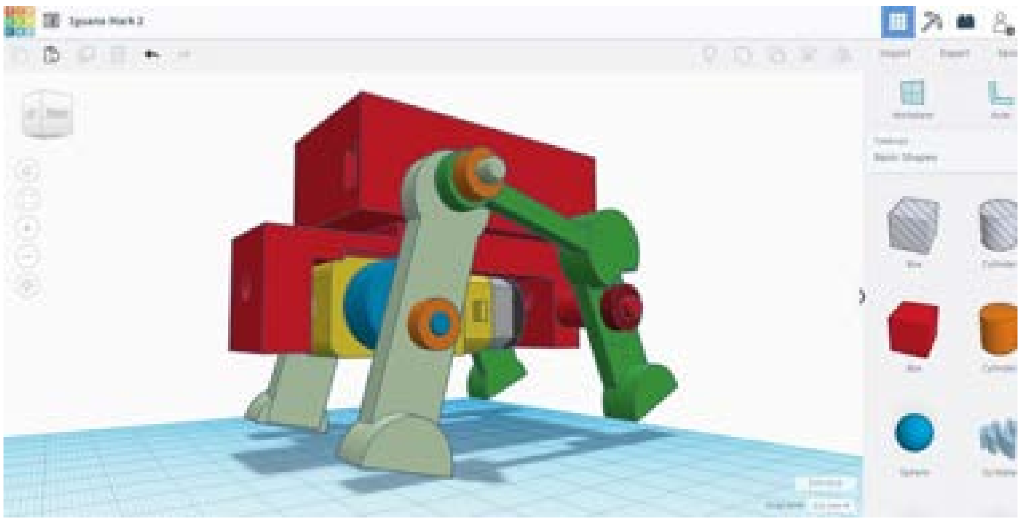


Fig. 14.2: *Tinkercad Interface (Tinkercad, 2020)*

c. **Virtual Robotics Toolkit**: Designed for use with the Lego Mindstorms EV3, the Virtual Robotics Toolkit is a physics-enabled robotics simulator. The simulator allows users to drive their own virtual robot but without the burden of ever needing space for testing or running out of physical bricks.

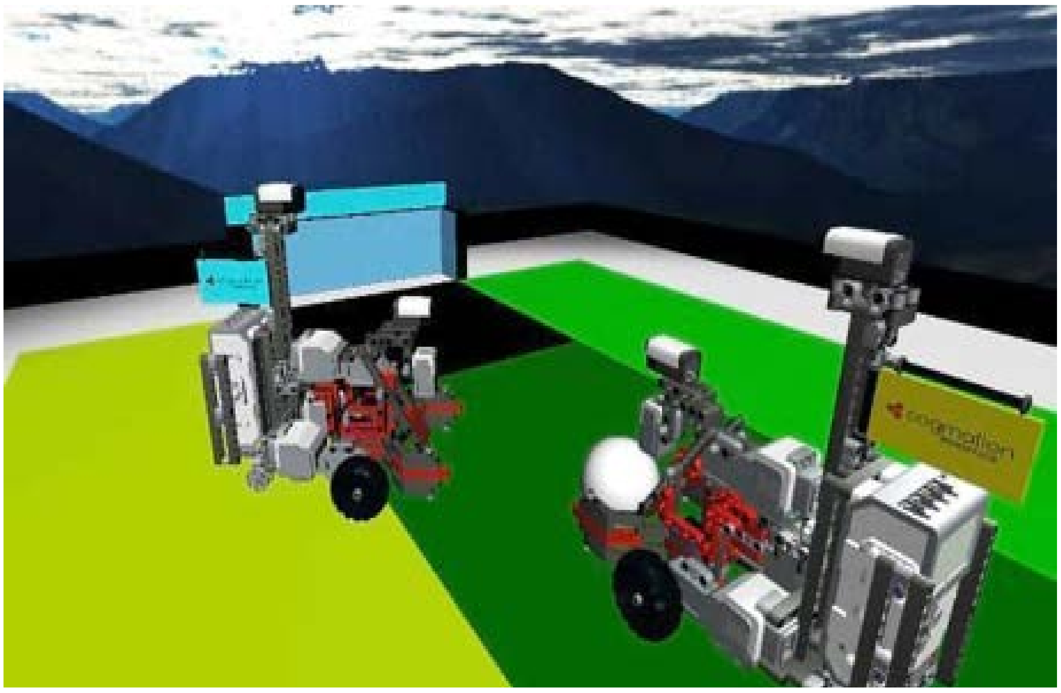


Fig. 14.3: *Virtual Robotics Toolkit*

2. **Exploring Robot Mechanics using LEGO Education SPIKE App (IDE)**:

As mentioned in the previous week, IDEs enable us to programme robots to accomplish given tasks. The LEGO Education SPIKE App provides us with an interface to programme robots in an intuitive and fun way. It allows learners of all skill levels to programme their built robots using either Icon-block, word-block or text-based coding.

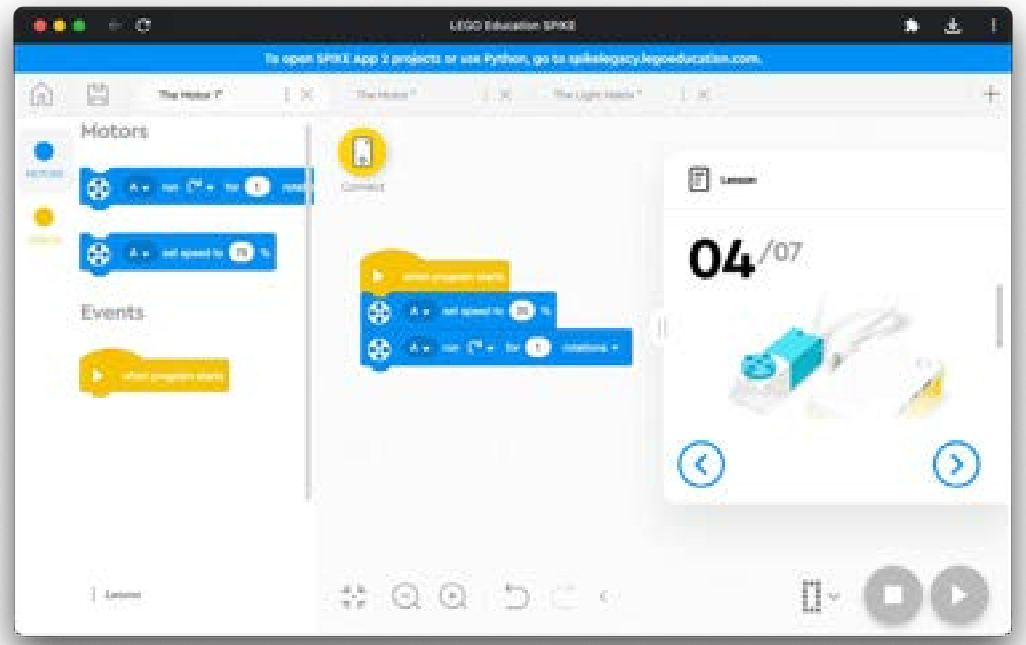

LEGO Education SPIKE
To open SPIKE App 2 projects or use Python, go to spikelegacy.legoeducation.com.
Motors
Events
Connect
04/07

Fig. 14.4: *LEGO Education SPIKE App (IDE)*

It also provides learners with tutorials to start building robot structures using the LEGO® Education SPIKE™ Essential or LEGO® Education SPIKE™ Prime Kits. Figure 14.5 shows a snapshot of the interface for doing this.

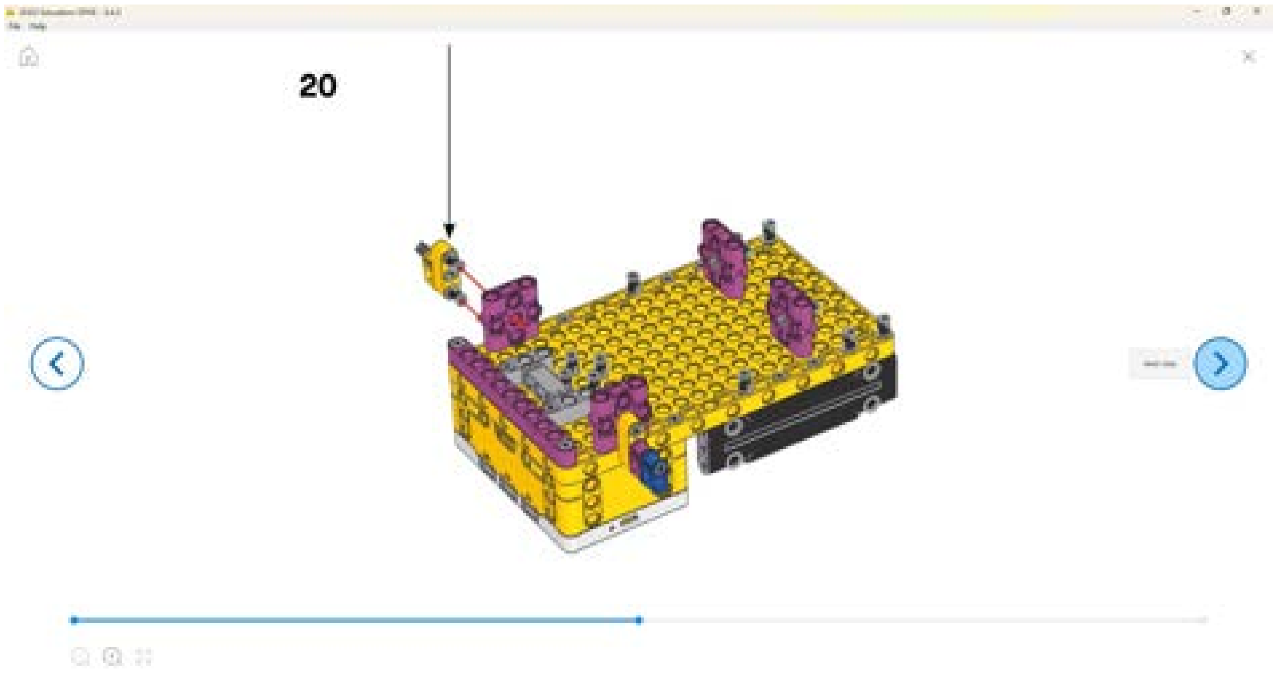


Fig. 14.5: *LEGO Education SPIKE App Build tutorials for constructing robots*

Also, to help learners with the basics of programming these kits, it provides some basic tutorial activities. These activities cover the major controllers, sensors and actuators.

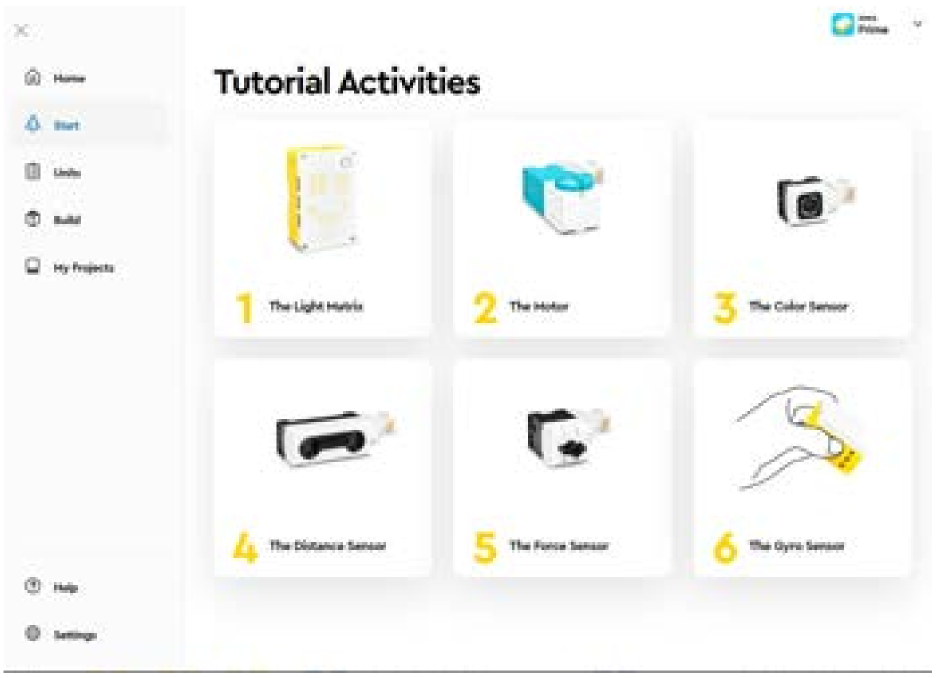


Fig 14.6: *Tutorial Activities in LEGO Education SPIKE App*

3. **Simulating Robot Performance**:

   Simulations provide a valuable tool for testing and evaluating robot performance without the need for physical prototyping. In virtual environments, we can:

   a. **Test Algorithms**: Implement and test control algorithms for navigation, path planning, and obstacle avoidance in a risk-free setting.

   b. **Validate Design Choices**: Evaluate the robot's stability, agility and efficiency based on its virtual performance, leading to improved design decisions.

   c. **Error and Risk Analysis**: Analyse potential errors or risks that the robot may encounter during real-world operation, aiding in pre-emptive problem-solving.

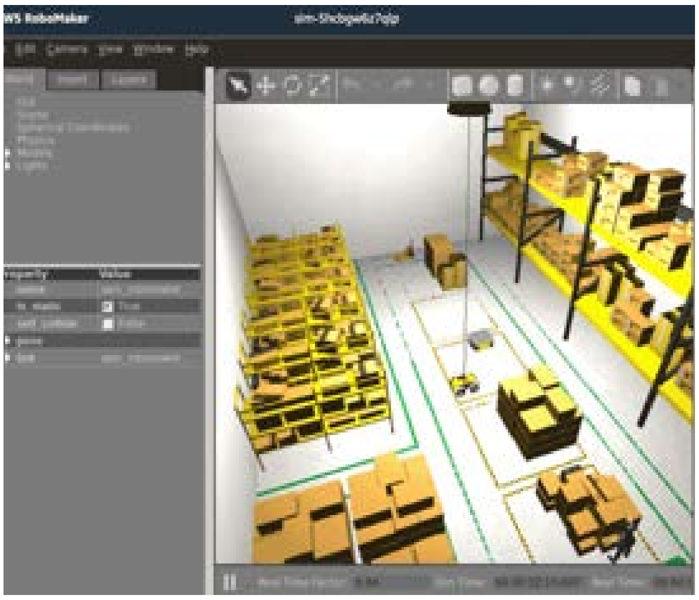


Fig. 14.7: *Virtual simulation of a mobile robot navigating through a warehouse environment (Vamshi Konduri, 2020).*

Learning Tasks

Learners shall:

1. Explore virtual platforms for robot design and simulation.
2. Design and build 3D models of robots in a virtual environment.
3. Understand robot mechanics through virtual exploration.
4. Run simulations to test robot performance and identify potential issues.
5. Refine their design based on simulation results.
6. Gain hands-on experience with virtual tools used in robotics development.

Pedagogical Exemplars

1. **Enquiry-Based Learning:**
   a. Interactive Platform Exploration: Learners explore a user-friendly virtual platform like Bricklink or Tinkercad with a teacher-led demonstration. Through guided navigation, they identify key interface elements (3D workspace, component library, simulation controls).
   b. The teacher poses a guiding question: "Which virtual platform would be more suitable for designing a complex robot with many sensors?" Learners work in pairs to research features of Bricklink or Tinkercad using the provided information and online resources.
   c. Comparison Chart Creation: Learners research and compare features of two virtual platforms (Bricklink or Tinkercad) using the provided information and online resources. They create a comparison chart highlighting factors such as user interface complexity, supported functionalities (e.g. sensor types), and suitability for beginners. This approach encourages critical thinking, research skills and collaboration among learners.
2. **Problem-Based Learning:** The teacher presents a project brief: "Design a robot capable of navigating a simulated warehouse environment."
   a. Learners are provided with a variety of robot components and building blocks in the virtual platform. Based on their skill level, they tackle the challenge with varying degrees of complexity in groups.
   b. Learners are put into mixed abilities based on their level of experience with robotic software.
   c. Provide access to diverse resources to cater to the varying preferences of learners. These resources may include videos, images, technical datasheets, podcasts and other multimedia formats.
   d. Ensure that all learners have opportunities to access the content in a way that best suits their learning preferences and abilities.

Key Assessment

Assessment Level 1: Identify three (3) robotic actions that can be simulated in the software package(s) you have used in this lesson.

Assessment Level 2: Write the command sequence for a simulation software package you have used which enables movement to take pace in all or part of a robotic arm.

Assessment Level 2: Use the auto-completion function on an IDE to input a code which simulates the movement of a robot arm.

Assessment Level 3: Using appropriate software, simulate a simple robot that can circumvent obstacles standing in its way.

Assessment Level 4: Design a robot to address one of the challenges of a recognised global robotics challenge (e.g. World Robotics Olympiad) using any design tool. Simulate the basic mechanics of the designed robot and present your solution in class.

Conclusion

Virtual robot design and simulation open up a world of possibilities for robotics enthusiasts. By using virtual platforms such as Bricklink or Tinkercad, we can design intricate robot mechanics, simulate their performance and optimise designs before physical implementation. The ability to explore mechanics and test robots in virtual environments empowers us to create more efficient and robust robotic systems.

Week 15

Learning Indicator: *Use a CAD tool to model parts of robotic systems.*

Theme or Focal Area: 3D Printing and CAD Modelling for Robotic Systems

Introduction

This week, the focus is on exploring the fascinating world of 3D printing and Computer-Aided Design (CAD) modelling for robotic systems. 3D printing has revolutionised the way we create physical objects, including parts for robots. CAD modelling enables us to design intricate and precise components that can be transformed into tangible objects through 3D printing. Learners will be introduced to the process of 3D printing and CAD modelling and learn how to create custom parts for robotic systems.

1. **Understanding 3D Printing**: 3D printing, also known as additive manufacturing, is a process of creating physical objects by depositing material layer by layer. It enables us to turn digital designs into tangible prototypes or end-use parts. 3D printing offers several advantages in robotics, such as rapid prototyping, custom part production and cost-effectiveness.
2. **Introduction to CAD Modelling**: Computer-Aided Design (CAD) is the process of using software to create detailed 2D or 3D models of objects. CAD modelling is essential for designing robotic components with precision and complexity. CAD software allows us to visualise, modify and simulate designs before they are 3D printed.
3. **CAD Tools for Robotics**: Several CAD software options are available for designing robotic parts:
   a. **Auto desk Fusion 360**: Fusion 360 is a powerful CAD tool that integrates 3D modelling, simulation and collaboration capabilities. It offers an intuitive interface and is widely used in robotics and engineering.

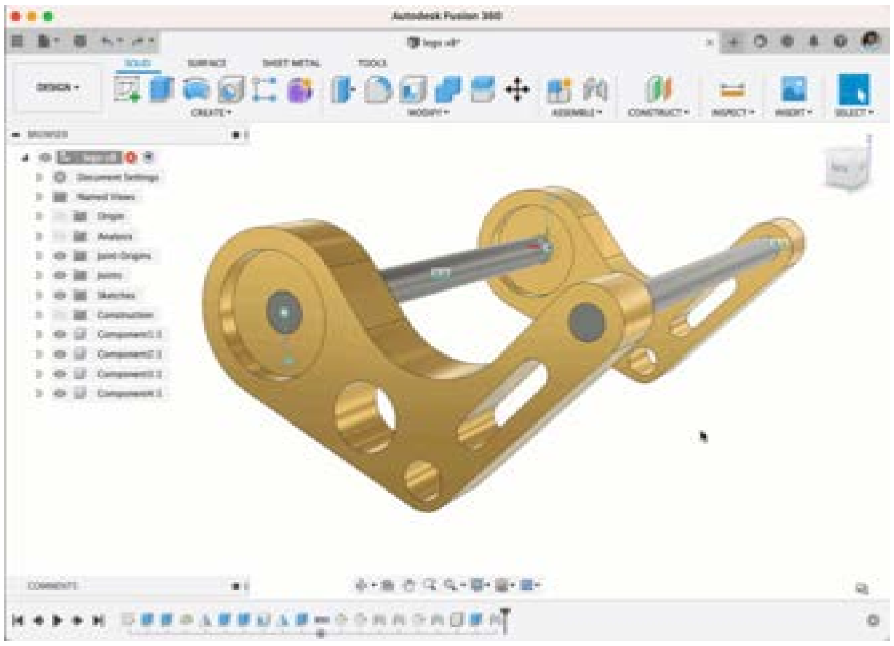


Fig. 15.1: *Autodesk Fusion 360 User Interface*

   b. **Solid Works**: Solid Works is a popular CAD software known for its advanced modelling features and industry-specific tools. It provides tools for designing complex robotic components.

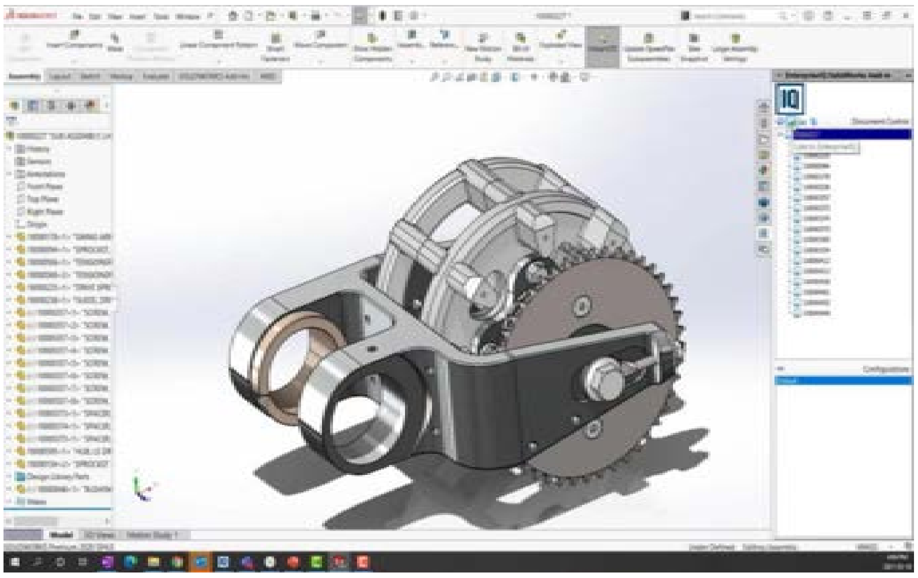


Fig. 15.2: *Solid Works User Interface (Rob Hauser, 2021)*

c. **Tinkercad**: Tinkercad is a user-friendly, web-based CAD tool suitable for beginners and learners. It is a great starting point for learning CAD modelling.

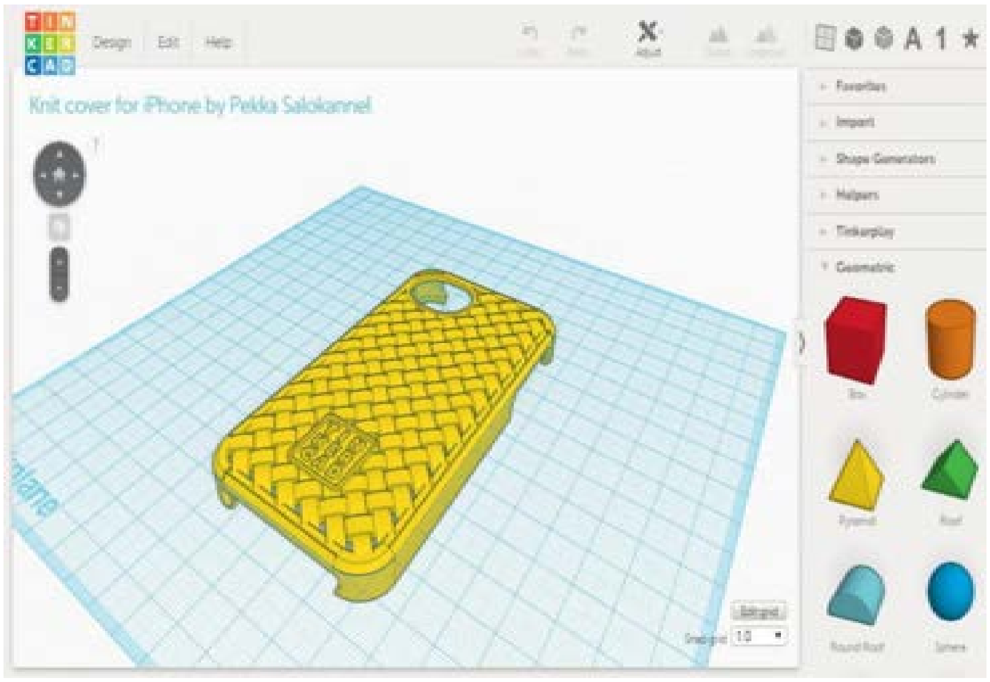


Fig. 15.3: *Tinkercad Interface*

4. **CAD Modelling for Robotic Components**: This session will focus on designing two common robotic components using CAD software:
   a. **Robot Chassis**: The robot chassis is the body or framework that holds all the components together. Design a customised robot chassis using CAD software, considering the robot size, shape and mounting points for sensors and actuators.
   b. **Wheel Assembly**: Design the wheel assembly, including motor mounts and wheels suitable for your robot's locomotion needs. Consider factors such as traction, size and compatibility with the chosen motors.
5. **Exporting for 3D Printing**: Once the CAD models are ready, they need to be prepared for 3D printing. Export the designs in a suitable file format (such as STL or OBJ) that is compatible with the 3D printer you will be using.

6. **3D Printing the Robotic Components**: Prepare the 3D printer by setting the appropriate print parameters, such as layer height and print speed. Load the exported CAD models into the 3D printer software and initiate the printing process. Watch as your designs come to life layer by layer.

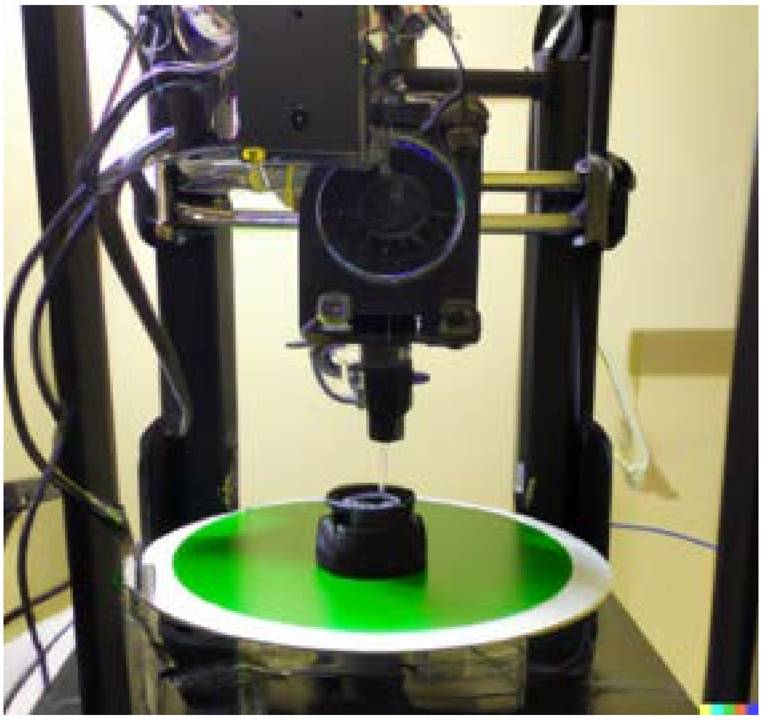


Fig. 15.4: *3D Printing Process*

Learning Tasks

To reinforce the concepts taught:

1. Learners will be introduced to 3D printing and CAD modelling through discussion.
2. Learners will work in mixed-ability groups to design robotic components using CAD software.
3. Learners will present their designs to the class and receive feedback.
4. Learners will work in smaller groups to prepare their designs for 3D printing.

Pedagogical Exemplars

1. **Guided Design Challenge**: Learners work in groups following a teacher-led presentation on gripper mechanisms.
    a. The teacher showcases different gripper designs (simple two-finger, claw-like) using physical models or visuals. Learners then receive a pre-made 3D model of a basic robot arm in the chosen CAD software (Tinkercad).
    b. The teacher guides them step-by-step through the process of designing a simple gripper attachment that can be integrated with the robot arm model. They experiment with different shapes and sizes to create a gripper that can grasp objects of two different sizes.
    c. Provide additional support or scaffolding for learners who may struggle with the task. Provide clarification to learners who may need it.
    d. Allow flexibility in how learners demonstrate their understanding such as through verbal explanations or written responses.

2. **Interactive Tutorials and Gamification:**
   a. Learners engage in a series of interactive tutorials within the chosen CAD software (e.g. Tinkercad). These tutorials use gamification elements such as points and badges to motivate them as they learn basic functionalities.
   b. The tutorials focus on creating fundamental 2D shapes, extruding them into 3D objects and making simple modifications such as adding holes using cutting tools.

3. **Problem-Based Learning:**
   a. Learners are put in mixed-ability groups and given tasks to print the basic robotic arm which they designed. Highly proficient learners lead the group to print the parts with the help of their teacher.
   b. Approaching proficiency learners should record the steps taken to get the part printed. Proficient learners should check for compliance with the design of the printed part and advise the group.
   c. Also, encourage active participation from all learners by ensuring each group member has a role in the activity.
   d. Provide feedback and reinforcement to reinforce learning and encourage continued engagement.

Key Assessment

Assessment Level 1: For a CAD design tool you have used, state three (3) features, describe their primary purpose and outline at what design stage each might be needed.

Assessment Level 2: Outline the stages of designing and quality assuring your robot gripper in preparation for printing. Assess your group’s performance by explaining what went well, what went not so well and areas which could be improved for future design projects.

Assessment Level 3: Explain why it is important to quality assure a robot component before printing.

Conclusion

By understanding 3D printing and CAD modelling, you have gained valuable skills in creating custom robotic components. Using CAD tools such as Autodesk Fusion 360, Solid Works or Tinkercad, you can design intricate and precise parts that enhance the functionality and aesthetics of your robotic systems. By 3D printing these components, you can quickly turn your digital designs into tangible reality, enabling you to build and customise robots efficiently. As you continue your robotics journey, remember the power of 3D printing and CAD modelling in advancing the field of robotics and engineering.

Week 16

Learning Indicator: *Use relevant intermediate tools to prepare modelled files into g-codes and print the designs using a 3D Printer.*

Theme or Focal Area: 3D Printing with G-Codes: From CAD Modelling to Physical Prototypes

Introduction

This lesson will explore the final steps of the 3D printing process, focusing on using relevant intermediate tools to prepare CAD (Computer Aided Design) files into G-codes and printing the designs using a 3D printer. Understanding G-codes and the intermediate tools involved in the printing process is essential for successfully bringing CAD designs to life.

1. **Preparing CAD Models for 3D Printing**: Before printing, CAD models need to be prepared and optimised for the 3D printing process. This involves:
   a. **Model Inspection**: Ensure that the CAD model is error-free, and there is no overlapping or intersecting parts.

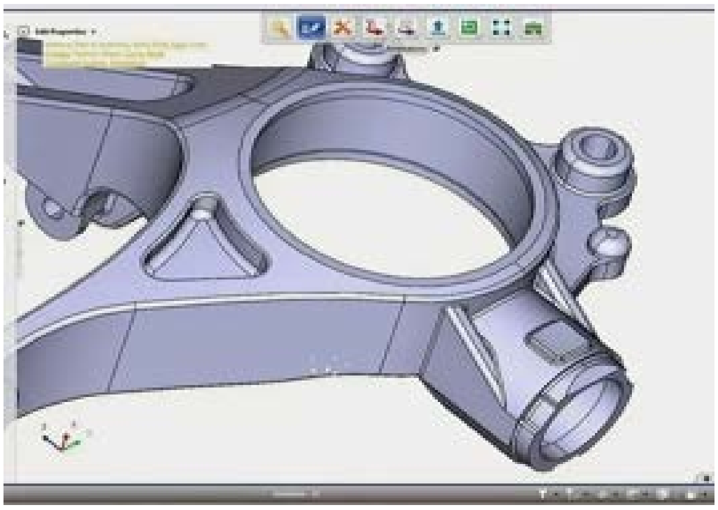


Fig. 16.1: *3D printing-model inspection*

   b. **Scale and Orientation**: Adjust the size and orientation of the model to fit the 3D printer's build volume and optimise printing speed and quality.

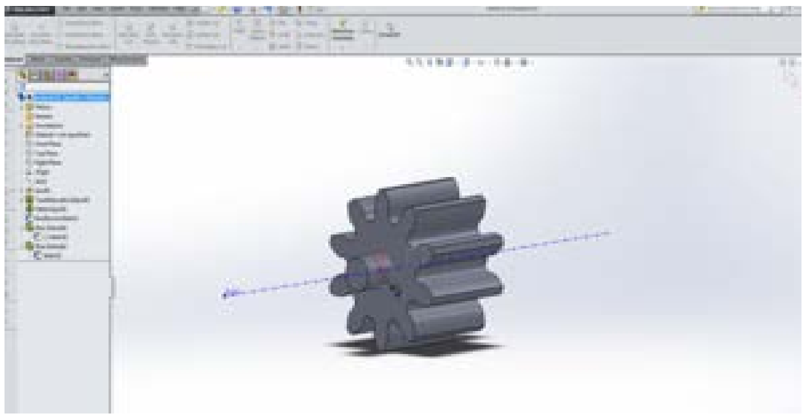


Fig. 16.2: *3D Printing- Scale and Orientation*

c. **Supports and Rafts**: Add support structures and rafts to stabilise overhangs and ensure better adhesion to the build plate during printing.

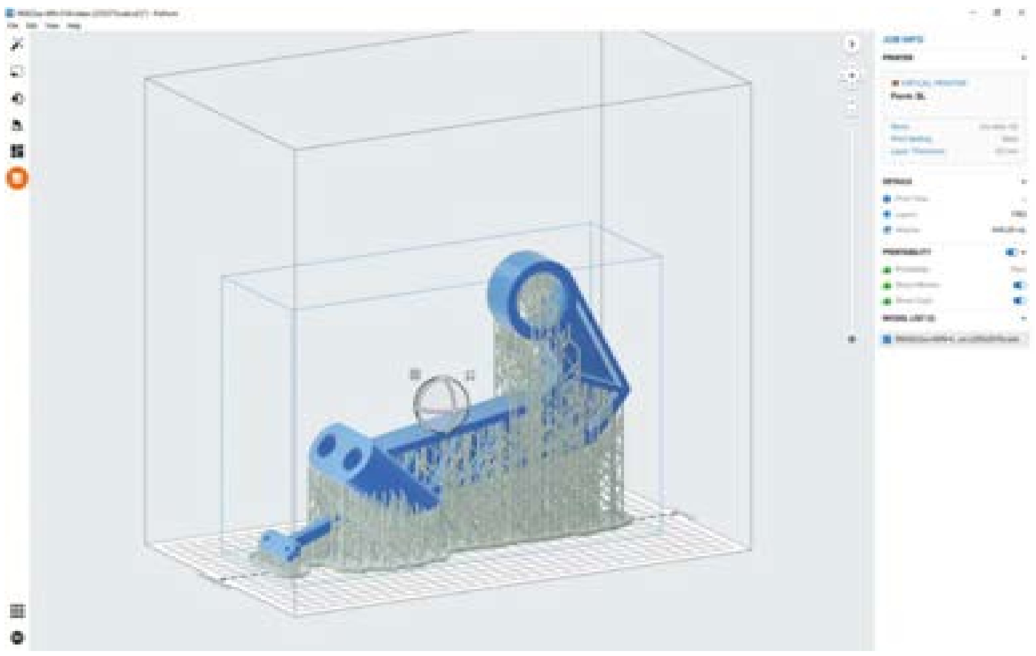


Fig. 16.3: *3D Printing-Support and Raft*

2. **Intermediate Tools: Slicers for G-Code Generation**: A slicer is an intermediate software tool used to convert the prepared CAD model into G-codes, which the 3D printer understands. Some popular slicer software include:

   a. **Ultimate Cura**: Cura is a widely used slicer known for its user-friendly interface and advanced features. It supports various 3D printer models and offers precise control over printing parameters.

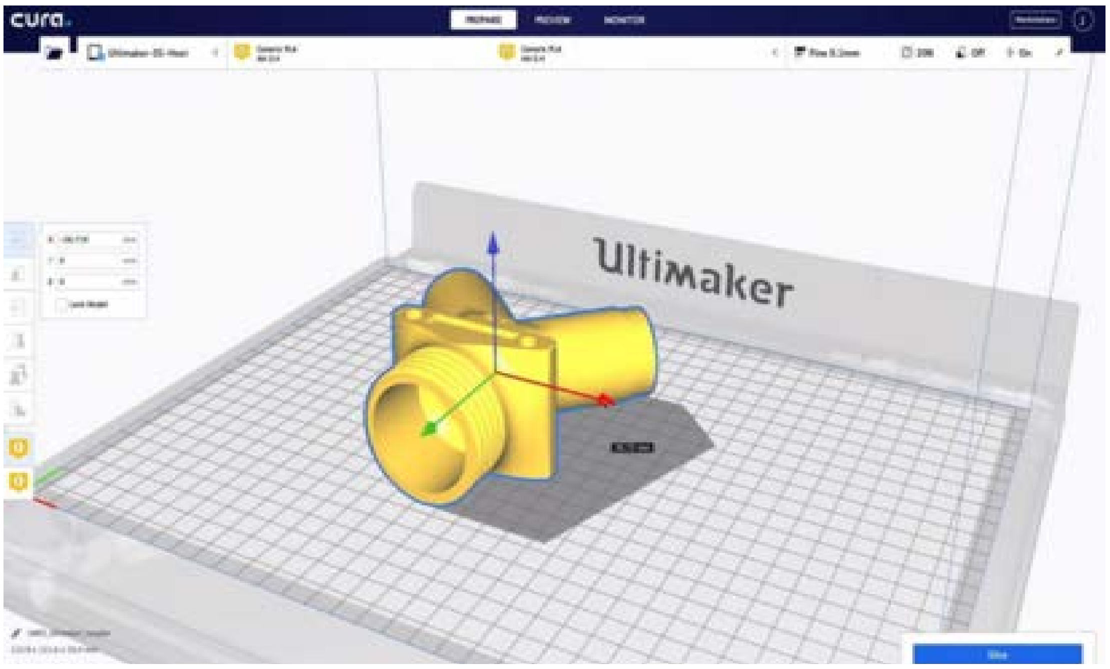


Fig. 16.4: *Slicing process using Ultimaker Cura*

   b. **PrusaSlicer**: PrusaSlicer is tailored for Prusa 3D printers and offers seamless integration with their hardware. It provides advanced customisation options for printing.

   c. **Simplify3D**: Simplify3D is a versatile slicer with extensive control over print settings. It supports a wide range of 3D printer models and is popular among experienced users.

3. **Exporting G-Codes and Printing**: Once the model is sliced and the G-codes are generated, it's time to initiate the 3D printing process. Follow these steps:
   a. **Export G-Codes**: Save the G-code file generated by the slicer onto an SD card or transfer it to the 3D printer using USB.
   b. **Preparing the 3D Printer**: Ensure the 3D printer is correctly set up, with the print bed levelled and the appropriate filament loaded.
   c. **Start Printing**: Insert the SD card with the G-code or initiate the print command through the 3D printer's interface. The 3D printer will start the printing process based on the G-code instructions.

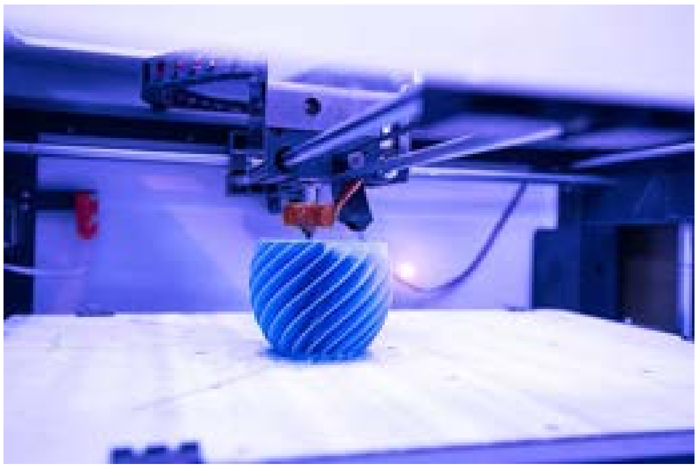


Fig. 16.5: *3D Printing Process in Action*

Learning Tasks

Depending on the available time or resources, administer one or more of the following learning tasks to help learners reinforce understanding and acquire new knowledge or skills.

Learners shall:

1. Inspect and prepare a CAD model for 3D printing.
2. Utilise slicing software to generate G-codes.
3. Initiate and monitor the 3D printing process.

Pedagogical Exemplars

1. **Experiential learning:** Learners use a beginner-friendly 3D modelling programme (Tinkercad, SketchUp Free) to design a simple name tag with their name and a fun graphic.
   a. Use visuals (simple diagrams) to show how a slicer cuts a 3D model into layers. Mention that slicers allow users to adjust settings for print quality and speed.
   b. Explain how slicers generate G-code instructions based on model geometry and user-defined settings.
   c. Introduce common slicer settings (layer height, infill density) and their impact on print quality and time.
   d. The teacher demonstrates the "slice" function and explains it as creating a recipe for the printer.
   e. Learners export their designs as G-code files with the teacher's guidance.
   f. The teacher bulk-prints the name tags on a pre-configured 3D printer, ensuring safety protocols are followed.

2. **Collaborative Learning**: Learners are made to sit in mixed-ability groups and choose from pre-designed 3D models of various keychains (animals, geometric shapes).
   a. The teacher introduces a slicing software interface (Cura, PrusaSlicer) and demonstrates adjusting basic settings such as layer height and print speed.
   b. Groups experiment with different settings within a designated range to see how they affect the estimated print time and quality (shown by the software).
   c. Groups export their G-code files with their chosen settings and print their keychains.

Key Assessment

Assessment Level 1: Name two 3D slicing software packages

Assessment Level 2: Explain the purpose of 3D slicing software.

Assessment Level 3: Revisit your design for the 3D robotic arm gripper (or part of it) from the last lesson and use software to slice it and generate G-codes, saving the resulting file to a medium large enough to accept it error-free.

Assessment Level 4: Print the 3D robotic arm with the 3D printer after slicing the design. Review your 3D printed component under the headings: Is it fit for purpose (size, functionality)? Does the 3D component need further finishing? Are any design improvements needed?

Conclusion

In this lesson, we have learned the final steps of the 3D printing process, using intermediate tools to convert CAD models into G-codes and bringing your designs to life using a 3D printer. Understanding how to prepare models, use slicers and manage 3D printing is crucial for successfully producing physical prototypes of your robotic components. As you continue to explore the world of robotics, these skills will empower you to prototype and optimise your designs efficiently, making your robotics projects a reality. Remember the incredible potential of 3D printing in revolutionising the field of robotics and engineering.

Section 5 Review

In this four-week section, we have seen how to effectively use virtual platforms and simulation tools to design, simulate and print robotic parts. Learners have been introduced to IDEs used in modelling and simulating robots. CAD modelling for robotic systems was introduced. Learners were able to create custom parts for robotic systems under supervision. Learners have been introduced to 3D printing software and hardware. This section covered the details of 3D printing and suggested activities to help learners make their first 3D printed parts. Learners were taken through generating G-codes and how to slice their designs.

Additional Reading

Link	QR Code
3D PRINTING 101: The ULTIMATE Beginner's Guide https://www.youtube.com/watch?v=2vFdwz4U1VQ	

Link	QR Code
Learn Fusion 360 in 30 Days for Complete Beginners! - 2023 EDITION https://www.youtube.com/playlist?list=PLrZ2zKOtC_-C4rWfapgngoe9o2-ng8ZBr	
Tinkercad Tutorial - Complete Guide https://www.youtube.com/playlist?list=PL90LC6zq_Lzf9tHyFPzX_9OA35BFTfEBs	
Introduction to Gazebo and .world file https://www.youtube.com/watch?v=QghfCqVSr6s	

References

1. Fairchild, C., & Harman, T. L. (2016). ROS robotics by example. Packt Publishing Ltd.
2. Open Robotics (March 1 2024). Getting Started with Gazebo? https://gazebosim.org/docs
3. Bricklink (October 29, 2021). Fileinfo https://fileinfo.com/extension/io
4. Tinkercad (August 31, 2020). Instructables.com https://www.tinkercad.com/projects/Tinkercad-Robotics-for-School-Create-TWO-Walking-M
5. Vamshi Konduri (February 15 2020). Fleet and multi-robot simulations in AWS RoboMaker. https://aws.amazon.com/blogs/robotics/fleet-and-multi-robot-simulations-in-aws-robomaker/#
6. Rob Hauser(March 24, 2021). Solidworks ERP Integration with DELMIAWorks. https://www.javelin-tech.com/blog/2021/03/solidworks-erp-integration-delmia-works/

SECTION 6: ROBOT CONSTRUCTION AND PROGRAMMING

Strand: **Robot Construction & Programming**

Sub-Strand: Higher Order Design Thinking

Learning Outcome: *Use flowchart diagrams to implement algorithms for solutions to basic problems.*

Content Standard: Demonstrate higher-order thinking (HOT – Analysis, Synthesis and Evaluation) in solving programming problems.

INTRODUCTION AND SECTION SUMMARY

Welcome to the exciting world of Higher Order Design Thinking (HOT) in robotics programming! This session will explore HOT's critical thinking skills, including analysis, synthesis and evaluation, to solve programming problems effectively.

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

The week covered by the section is:

Week 17:

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

SUMMARY OF PEDAGOGICAL EXEMPLARS

This section dives into Higher Order Design Thinking (HOT) in robotics programming! HOT will help learners develop their critical thinking, analysis, synthesis and evaluation skills to solve programming problems effectively. Teachers will utilise problem-based learning to cater to diverse learning needs.

Learners will be tasked with tackling real-world robotics problems by designing solutions using algorithms, pseudocode and flowcharts. Flowcharts are diagrams which use symbols and arrows to illustrate the sequence of steps in solving a problem. This way of simplifying problems into inputs, processes and outputs will boost the problem-solving skills of learners. Learners are taught how to simplify daily tasks into inputs, processes and outputs to increase their understanding of the HOT concepts. Through the use of pseudocode, learners will be able to develop algorithms to solve robotic challenges.

ASSESSMENT SUMMARY

Following each thematic area in this section, assessments gauge learner learning. These come in two forms: learning tasks and key assessments. Learning tasks, primarily formative, focus on solidifying understanding and acquiring new knowledge or skills. Facilitators guide these activities to enhance the learning process.

In contrast, key assessments, typically summative, evaluate learner mastery after instruction. These are often given as homework or quizzes outside of class. Instructors have the flexibility to choose the assessment types that best suit their learners and learning objectives. However, it is advisable that instructors at least guide learners to do one of the learning tasks.

Week 17

Learning Indicators:

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

Theme or Focal Area: Higher order Design Thinking: Flowchart Diagrams for Algorithm Implementation

Introduction

Welcome to the exciting world of Higher Order Design Thinking (HOT) in robotics programming! This session will explore HOT's critical thinking skills, including analysis, synthesis and evaluation, to solve programming problems effectively. Specifically, we will focus on using flowchart diagrams to represent algorithms for solving basic problems in robotics. Flowcharts provide visual representations of the problem-solving process, making it easier to determine inputs, processes and outputs required for specific challenges in robotics programming.

1. **Understanding Higher Order Design Thinking (HOT)**: Higher Order Design Thinking involves advanced cognitive skills that go beyond simple comprehension (Nigel Cross, 2023). By utilising HOT, you will analyse problems from multiple perspectives, synthesise information to create innovative solutions and evaluate the effectiveness of those solutions. These skills are instrumental in programming, leading to well-structured and efficient algorithms.

2. **Introduction to Flowchart Diagrams**: Flowchart diagrams are graphical representations of algorithms and processes. These diagrams use symbols and arrows to illustrate the sequence of steps in solving a problem. By using flowcharts, programmers can visualise the logic of the algorithm, making it easier to understand, debug and optimise the code. (Anderson et al 2002)

3. **Creating Flowcharts for Algorithm Implementation**: To solve a programming problem using flowcharts, follow these steps:
   a. **Define the Problem**: Clearly articulate the problem you want to solve, including the desired inputs and outputs.
   b. **Identify Inputs, Processes and Outputs**: Determine the data inputs, the operations or processes required, and the desired outputs for the solution.

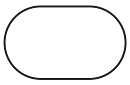
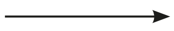
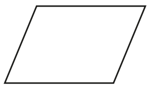
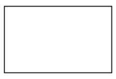
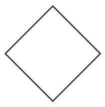
Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or output.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

Fig 17.1: *Basic Flowchart Symbols and their Meanings*

c. **Develop Flowchart**: Use the appropriate flowchart symbols to represent each step of the algorithm. Connect the symbols with arrows to show the flow of execution.

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.

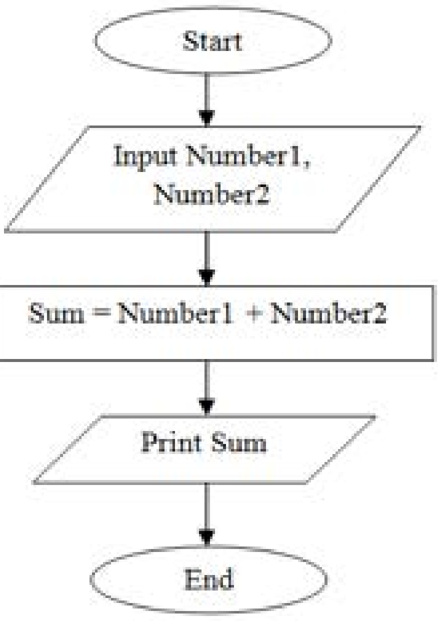


Fig. 17.2: *Flowchart Example for a basic algorithm for adding two numbers*

4. **Applying Flowchart Diagrams in Robotics Programming**: Flowchart diagrams find extensive application in robotics programming to:

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

   b. **Implement Control Loops**: Use flowcharts to design control loops that adjust robot actions based on feedback from sensors or other external factors.

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

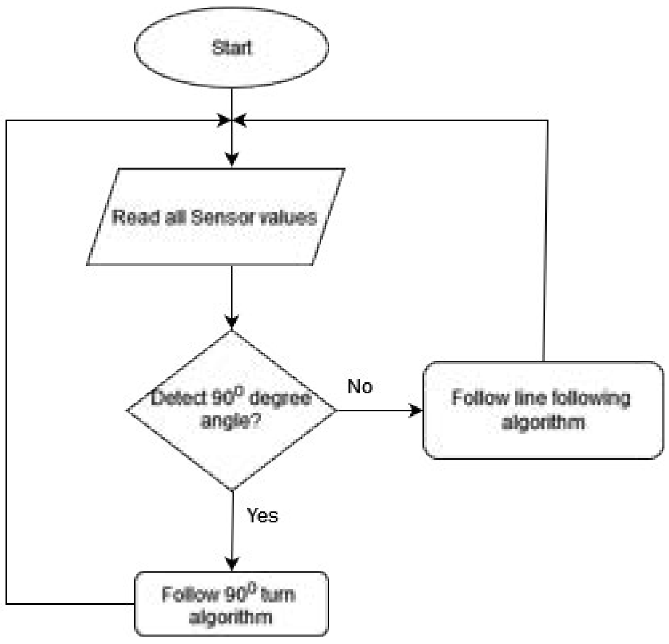


Fig. 17.3: *Flowchart Example for Robot Line Following algorithm*

Learning Tasks

Learners should:

1. Analyse a problem and identify key components (inputs, processes, outputs).
2. Translate a solution strategy for this problem into a visual flowchart using appropriate symbols.
3. Test and refine the flowchart to ensure it produces the desired results.c

Pedagogical Exemplars

Problem-Based Learning

1. In mixed-ability groups, let learners create a flowchart for making breakfast porridge.
   a. Provide symbols representing start/end, decision points, processes and inputs/outputs.
   b. Guide learners to break down the steps (get a bowl, add water, pour milk, etc.) and connect them with arrows.
2. In mixed-ability groups, learners select a real-world robotics challenge (e.g. line following, obstacle avoidance) and identify the necessary inputs, processes and outputs required to solve the problem. They then present identified parts to the class.
3. Learners should be made to provide alternative flow charts to the same problem.

Key Assessment

Assessment Level 1: Briefly describe the purpose of using flowchart diagrams in robotics programming.

Assessment Level 2: Describe three (3) ways in which flowcharts can be applied to the programming of robots.

Assessment Level 3: Create a flowchart for a 2-wheeled robot with two underside mounted light sensors to follow a white line in the shape of an oval. Consider sensor inputs, processes, flows, decisions, start.

Assessment Level 4: Research a real-world example of a robot that uses complex algorithms and decision-making processes. Explain how flowcharts likely played a role in the development of its programming.

Conclusion

In this lesson, we have explored Higher Order Design Thinking and its application in robotics programming. By using flowchart diagrams, you can implement algorithms for solving basic problems and plan the logic behind robotic tasks efficiently. As you continue your robotics journey, remember that HOT and flowchart diagrams are invaluable tools for developing well-structured and effective solutions to challenges in robotics programming. Embrace the power of design thinking to advance your robotic projects and optimise their performance. Through practice and application, you will enhance your programming skills and become a more proficient robotics engineer.

Theme or Focal Area: Algorithmic Problem-Solving in Robotics: Pseudocodes and Flowchart Diagrams

Introduction

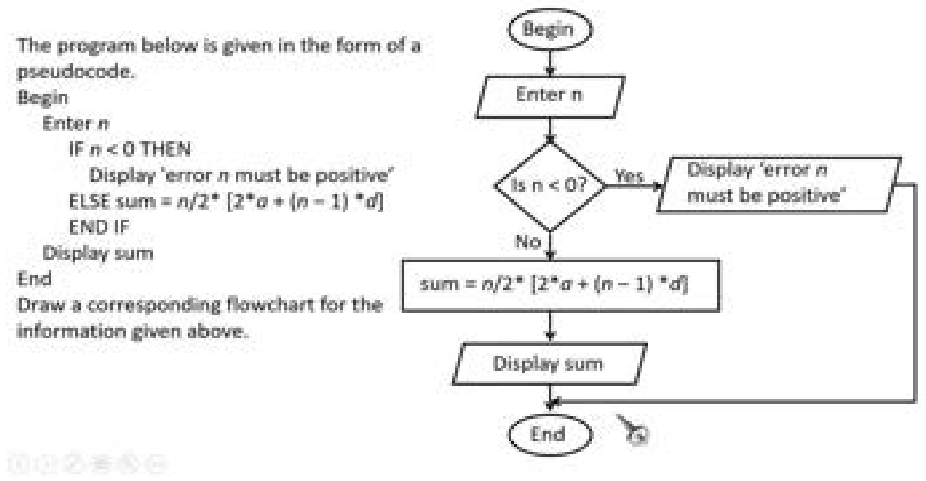
This session will explore the art of algorithmic problem-solving for robotics. To efficiently solve problems, we will use three powerful tools: algorithms, pseudocodes and flowchart diagrams. These tools will help us define solutions to basic automated and robotic challenges, making our robotic systems more capable and intelligent and provide the starting point for writing programmes which control their functionality.

Understanding Algorithms in Robotics

An algorithm is a step-by-step procedure for solving a problem or accomplishing a specific task. It is not so different from a recipe used in preparing a meal. Before implementing any functionality with a robot, we carefully think through the step-by-step approach to achieve this, that is, its algorithm for achieving such functionality. It is usually important to think about this in minutes or simple steps, not combining complex actions into one step. These steps are first written down in simple 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 have any syntax or rules that govern how it should be represented. It may even make use of symbols or abbreviations. It bridges the gap between natural language and programming language, allowing us to outline the logic of the solution before implementing it in a specific programming language. (Corment et al, 2022)

The Pseudocode is then translated to a flowchart diagram before implemented with a selected programming language. As stated earlier, flowchart diagrams provide a visual representation of the algorithm's logic. By using pseudocode and flowcharts, we can easily understand the sequence of steps and the decision-making process within the 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. 17.4: *Example of an algorithm written in pseudocode and translated into a flowchart*

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

Defining Solutions to Basic Robotic Problems:

This session will tackle basic robotic problems and define solutions using algorithms, pseudocodes and flowchart diagrams. Some examples of problems include:

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

b. **Line Following**: Develop solutions 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.

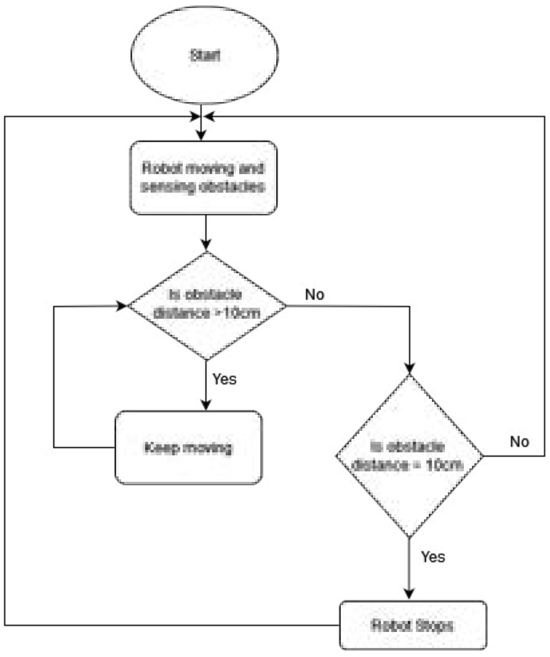

Start
Robot moving and sensing obstacles
Is obstacle distance >10cm
No
Yes
Keep moving
Is obstacle distance = 10cm
No
Yes
Robot Stops

Fig. 17.5: *Flowchart diagram illustrating the algorithm for obstacle avoidance*

Practice Problem:

Problem: Design a Robot Maze Solver

Task: Create an algorithm, pseudocode and flowchart diagram for a robot to solve a maze autonomously. The robot should explore the maze, avoid dead-ends and find the shortest path to the exit.

Learning Tasks

To reinforce understanding, learners should

1. Discuss the role of algorithms in robotics.
2. Analyse pseudocode examples.
3. Interpret flowchart diagrams.
4. Explore solutions for basic robotic problems.
4. Develop their own pseudocode and flowchart for a selected problem, etc.

Pedagogical Exemplars

Problem-Based Learning

1. In mixed-ability groups, Learners tackle real-world robotics problems (e.g. maze solving, sumo robot strategy) and design solutions using algorithms, pseudocode and flowcharts.

2. Learners can be given some pseudocodes to break down step by step by explaining what each line does.
3. Provide additional worked-through examples or mini-challenges for beginners. Offer opportunities for peer support and collaboration during problem-solving activities.

Key Assessment

Assessment Level 1: Draw a flowchart to represent automatic turning a light on and off based on a threshold value within a light sensor.

Assessment Level 2: Explain the difference between an algorithm and a flowchart diagram.

Assessment Level 3: Design an algorithm for a sumo robot that prioritises avoiding being pushed out of the ring. How would this differ from an algorithm focused on pushing opponents out?

Assessment Level 4: Research a real-world application of robotics that utilises more complex algorithms. Explain how the specific algorithms contribute to the robot's functionality.

Conclusion

In this lesson, we explored algorithmic problem-solving techniques in robotics, utilising pseudocodes and flowchart diagrams. By mastering these tools, you can efficiently define solutions to basic automated and robotic problems. Algorithms are the backbone of any robotic system, guiding their actions and behaviours. As you progress in your robotics journey, practise implementing algorithms, pseudocodes and flowcharts to enhance your programming skills and create smarter and more capable robotic systems. Remember, the power of algorithmic thinking is the key to unlocking the full potential of robotics in various real-world applications.

Section 6 Review

This section introduced higher-order thinking (HOT) to the learners as a critical component of robot design. Learners were taught how to use critical thinking skills, including analysis, synthesis and evaluation, to solve programming problems effectively. They were taught how to use flowchart diagrams to implement algorithms for solving basic problems in robotics. Flowcharts provide visual representations of the problem-solving process, making it easier to determine inputs, processes and outputs required for specific challenges in robotics programming. Learners were introduced to how algorithms are simplified using pseudocodes to depict the functions in the algorithm. These tools will help us define solutions to basic automated and robotic challenges, making our robotic systems more capable and intelligent.

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.

SECTION 7: ROBOT CONSTRUCTION

Strand: **Robot Construction & Programming**

Sub-Strand: Robot Construction

Learning Outcomes:

1. *Appraise the effects of mass and centre of gravity in designing structures that withstand forces*
2. *Create robots using fabricated robotic materials or local materials to implement basic mechanics*

Content Standards:

1. Demonstrate a general understanding of rigid bodies and design processes for stable structures.
2. Demonstrate the ability to create robots, vehicles and other contraptions with moving parts.

INTRODUCTION AND SECTION SUMMARY

This section of the robotics teaching manual focuses on Robot Construction. Learners will gain the knowledge and practical skills necessary to design and build their own robots. The key learning outcomes are twofold: understanding the impact of mass and centre of gravity on robot stability and constructing robots using both prefabricated kits and everyday materials.

Through hands-on activities, learners will explore the science behind stable structures and basic mechanics such as gears, levers and axles. This will culminate in the ability to create robots with moving parts, fostering creativity and problem-solving skills. By the end of the section, learners will be well equipped to design and build robots that can move, explore their environment, and potentially even complete exciting challenges.

The weeks covered by the section are:

Week 18: Robot Construction: Designing Stable Structures and Understanding Mass and Centre of Gravity

Weeks 19 and 20: Creating Robots with Basic Mechanics: Exploring Gears, Vehicles and Moving Mechanisms

SUMMARY OF PEDAGOGICAL EXEMPLARS

This section dives into how teachers can effectively guide learners in building robots. Several teaching methods are highlighted. Using Building on What Others Say, discussions tap into learners' existing knowledge of physics and real-life experiences to explore stability, forces and momentum in robot design. This activates prior knowledge and helps learners connect scientific principles to practical applications. Experiential Learning & Project-Based Learning approaches get learners actively building sub-structures, robots and gear trains. Through these activities, they gain practical skills in using gears, levers and axles while applying the learned concepts. Teachers guide these activities, while project-based learning fosters teamwork and problem-solving as learners collaborate on building robots that meet specific criteria.

These strategies can be adapted for different learning styles and can be further enriched for gifted learners by providing them with more complex challenges and encouraging the exploration of real-world applications in robotics engineering.

ASSESSMENT SUMMARY

Following each thematic area in this section, assessments gauge learner learning. These come in two forms: learning tasks and key assessments. Learning tasks, primarily formative, focus on solidifying understanding and acquiring new knowledge or skills. Facilitators guide these activities to enhance the learning process. In contrast, key assessments, typically summative, evaluate learner mastery after instruction. These are often given as homework, mid-semester exams and end-of-semester exams outside of class. Instructors have the flexibility to choose the assessment types that best suit their learners and learning objectives. However, it is advisable that instructors at least guide learners to do one of the learning tasks.

Week 18

Learning Indicators:

1. *Describe the effect of mass and centre of gravity on the stability of a structure or robot and strategies for designing systems that can withstand forces.*
2. *Build structures for a specified use case and test them for stability and ability to withstand forces.*

Theme or Focal Area: Robot Construction - Designing Stable Structures and Understanding Mass and Centre of Gravity

Introduction

Welcome to the world of Robot Construction! This section will explore the crucial aspects of designing stable structures for robots and understanding the effects of mass and centre of gravity. Building a robot that can withstand forces and maintain stability is essential for successful robotics projects. Let's delve into the design processes, principles of rigid bodies, and strategies for creating robots, vehicles and other contraptions with moving parts that can endure various forces.

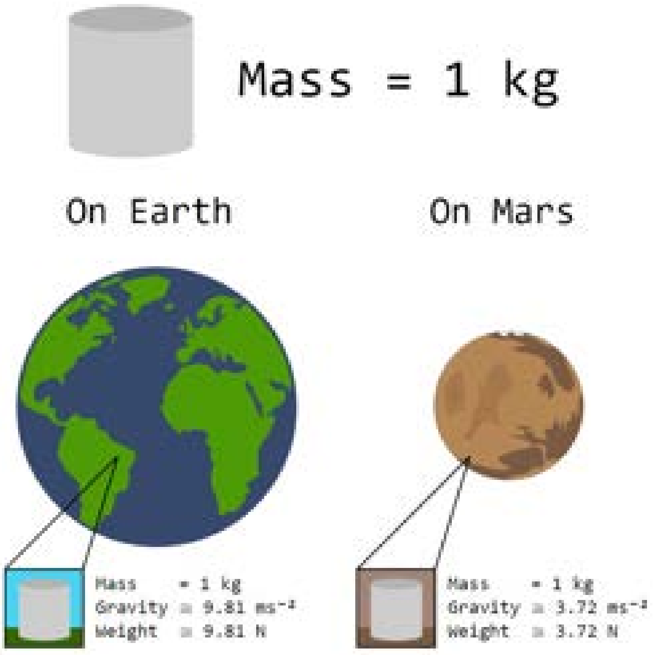


Fig. 18.1: *Mass on Earth vs. mass on Mars (Wikimedia Foundation. (2023, January 6). Mass versus weight.)*

Understanding Rigid Bodies and Stable Structures

In robotics, a rigid body refers to an object that does not deform under external forces. Designing stable structures involves ensuring that the robot's components are rigidly connected, preventing unnecessary movements that may compromise stability during operation.

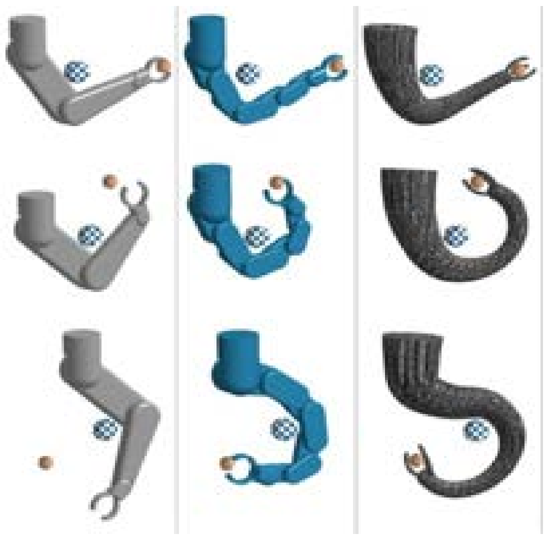


Fig. 18.2: *Robots with rigid structures and structures that lack rigidity (Charbel Dalely, Tawk Gursel Alici, Comparison of the proposed robotic arm system and related designs. | download scientific diagram 2021)*

Designing Stable Structures:

To create stable structures for robots and contraptions:

1. **Plan the Framework**: Carefully plan the robot's framework to ensure that it provides sufficient support and rigidity for all moving parts and components.
2. **Choose Sturdy Materials**: Select materials that are strong and rigid enough to withstand the anticipated forces and loads.
3. **Optimise Weight Distribution**: Distribute the robot's mass evenly to maintain a low centre of gravity, promoting stability during movement and operation.

Fig. 18.3: *Robotic arm showing optimised weight distribution (Standard Bots. (2023, October 4). https://standardbots.com/blog/how-much-does-a-robot-arm-cost)*

Understanding Mass and Centre of Gravity:

Mass refers to the amount of matter an object contains, while the centre of gravity is the point where the entire weight of an object is considered to be concentrated. Understanding these principles is critical for designing stable structures that can withstand forces and perform optimally.

Effects of Mass and Centre of Gravity on Stability

The mass and centre of gravity (CoG) of a robot play a crucial role in its stability. Here's how they impact a robot's ability to stay upright and function effectively:

1. **Mass**
   a. **Higher Mass:** A heavier robot generally has more momentum and inertia, making it more resistant to tipping over from small disturbances. However, a very heavy robot may also be cumbersome and require more powerful motors to move, impacting efficiency.
   b. **Lower Mass:** A lighter robot can be more agile and energy-efficient. However, it's also more susceptible to tipping over, especially if the weight is not distributed evenly.
2. **Centre of Gravity (CoG)**
   a. **Lower Centre of Gravity:** A lower centre of gravity provides better stability. Imagine a seesaw – the one with the lower centre of gravity is harder to tip. In robots, a lower Centre of Gravity means the weight is distributed closer to the base, making it more difficult to topple over.
   b. **Higher Centre of Gravity:** A higher centre of gravity increases the risk of tipping. Robots with high centres of gravity, such as some bipedal robots, may require complex control systems or additional balancing mechanisms to maintain stability.

Fig. 18.4: *Tightrope walking (Indian tightrope Walker)*

Think of a balancing act. A tightrope walker with a heavy backpack (high mass) might be more difficult to push off balance compared to someone carrying a light bag. However, the walker with the backpack also has a higher centre of gravity, making it easier to topple if they lean too far to one side.

Incorporating these concepts in robot design:

Robot designers consider both mass and Centre of Gravity to achieve optimal stability. This can involve:

1. **Strategic placement of components:** Placing heavier components lower in the robot's body lowers the Centre of Gravity.
2. **Counterweights:** Adding weights to the opposite side of a heavy component can help balance the Centre of Gravity.
3. **Wide and sturdy base:** A wider base increases the robot's footprint, making it harder to tip.

By understanding the effects of mass and Centre of Gravity, robot designers can create robots that are both stable and functional, allowing them to perform their tasks effectively.

Learning Tasks

Depending on the available time or resources, administer the following learning tasks to help learners reinforce understanding and acquire new knowledge or skills.

Learners:

1. design and construct a robot chassis using the provided materials. The challenge is to design a robot chassis that can withstand external forces, such as pushing or pulling, without tipping over or losing stability. They must consider the distribution of mass and the placement of the centre of gravity in their designs to maximise stability.
2. will test their designs by applying external forces to simulate real-world conditions.
3. will document their design process, including sketches, diagrams and explanations of how they applied knowledge of mass and centre of gravity to enhance stability.
4. will then evaluate the effectiveness of their designs and reflect on strategies for improving stability.

Pedagogical Exemplars

Building on What Others Say (Activating Prior Knowledge):

1. Begin by facilitating an interactive session where you ask learners to share their experiences of working with objects that are stable and unstable (e.g. leaning tower vs. pyramid).
2. Showcase real-world examples such as balancing bikes or tightrope walking. Discuss how weight distribution and centre of gravity affect stability in these scenarios.
3. For learners with prior physics knowledge, explore concepts such as linear momentum and forces acting on objects. Discuss how unbalanced forces can cause a robot to tip over.

Talk for Learning (Interactive Discussions):

1. Use simple demonstrations to illustrate the concepts. For example, tilt a ruler with a weight on one end and observe how the centre of gravity affects its stability. Learners can participate by predicting the outcome.
2. Divide learners into small mixed groups to discuss the relationship between mass, Centre of Gravity and stability. Encourage them to share examples and sketches of stable and unstable structures.
3. Pose open-ended questions such as “How can you modify a robot design to lower its centre of gravity?” or “What strategies can engineers use to improve the stability of a tall building?” to encourage deeper thinking.
4. Briefly review physics concepts (linear momentum) if needed, ensuring a foundation for all learners.
5. Offer manipulatives or alternative models for learners who benefit from hands-on exploration.
6. Use clear illustrations and diagrams throughout the lesson (e.g. rigid vs non-rigid structures, centre of gravity in robots). Encourage learners to sketch their designs and visualise the centre of gravity.
7. Offer 3D modelling tools or physical construction materials for learners who prefer kinaesthetic learning approaches, etc.

Key Assessment

Assessment Level 1: Explain why a pyramid is more stable than a narrow, tall tower.

Assessment Level 2: Explain how the mass and centre of gravity of a robot affects its stability.

Assessment Level 3: Collect images of robots with different designs. Explain how their design might affect their stability?

Assessment Level 4: Compare and contrast the stability of a robot with a high centre of gravity and a robot with a low centre of gravity. What measures can be taken to address centre of gravity issues when building robots?

Conclusion

In this lesson, we have explored the fundamentals of Robot Construction, focusing on designing stable structures and understanding the effects of mass and centre of gravity. Creating robots, vehicles and contraptions with moving parts that can withstand forces is essential for successful robotics projects. As you continue your journey in robotics, remember the significance of rigidity, weight distribution and centre of gravity in achieving stability and optimising robot performance. Through careful design and consideration of these principles, you will create robust and reliable robotic systems that excel in real-world applications.

Theme or Focal Area: Building and Testing Robot Structures: Ensuring Stability and Force Resistance

Introduction

This section will delve into the practical aspects of constructing robot chassis and frameworks for specified use cases. We will focus on ensuring stability and the ability to withstand forces during the design and fabrication process. By putting theory into practice, you will gain valuable insights into creating robust and reliable robotic systems that can excel in real-world applications.

Understanding Use Case Requirements

Before diving into construction, we must first analyse the specific use case and functional requirements of the robot. Consider factors such as:

1. **Task and environment**:
   Determine what the robot will be doing and the environment it will be operating in. Different tasks and environments may demand specific structural considerations.
2. **Load Capacity**:
   Assess the expected load that the robot may carry or interact with during its operation.
3. **Mobility and Manoeuvrability**:
   Define the required mobility and manoeuvrability to ensure the robot can navigate effectively.

Selecting Suitable Materials

Based on the use case requirements, choose materials that provide the necessary strength, rigidity and durability. Common materials used in robot construction include:

1. **Metals**: Aluminium and steel offer excellent strength-to-weight ratios and are ideal for heavy-duty applications.
2. **Plastics**: Lightweight and flexible plastics are suitable for smaller robots and rapid prototyping.

3. **Composites**: Composite materials, such as carbon fibre, offer high strength and low weight, making them suitable for advanced robotics projects.

Building Stable Structures:

To construct stable robot structures:

1. **Design Framework**: Create a detailed design of the robot's framework, considering the layout of components, attachment points and load distribution.
2. **Mechanical Connections**: Ensure strong and secure mechanical connections, such as bolts, nuts, screws or welding, to hold the structure together firmly.
3. **Reinforcements**: Use cross-bracing or additional support structures where needed to enhance stability.

Building a Robot Car Chassis Kit – Arduino

Follow the link: https://randomnerdtutorials.com/build-robot-car-chassis-kit-arduino/

Testing for Stability and Force Resistance:

Once the robot structure is built, it's time to put it to the test. The following are some tests that we can use to test the robot's structure.

1. **Stability Test**:Evaluate the robot's stability by simulating various movement scenarios, such as turning, accelerating or stopping abruptly.
2. **Load Test**: Test the robot's ability to withstand the expected load by gradually adding weight or applying force to critical points.
3. **Impact Test**: Assess the robot's resistance to impacts or external forces that it may encounter in its intended environment.

Reference Link	QR Code
Follow the link: Testing Robustness of Robot - https://www.youtube.com/watch?v=aFuA50H9uek	
Follow the link: Robot hand Performance Test - https://www.nist.gov/video/robotic-hand-performance-testing	

Learning Tasks

Depending on the available time or resources, administer the following learning tasks to help learners reinforce understanding and acquire new knowledge or skills.

Learners:

1. analyse robot use cases and identify requirements.
2. research and select suitable materials.
3. design a robot chassis and framework.
4. build the robot structure with strong connections.
5. conduct stability, load and impact tests.

Pedagogical Exemplars

Experiential learning:

1. Consider turning this activity into a project where learners work in teams throughout the week. This fosters collaboration, communication and problem-solving skills.
2. Provide clear instructions and demonstrations initially. Break down the task into smaller steps, especially for complex use cases. Offer guidance and support as learners progress.
3. Adjust the complexity of the use case and materials based on learner skill levels. Offer pre-cut materials or simpler design options for struggling learners.
4. Introduce concepts step by step (use case analysis, material selection, construction techniques). Provide clear instructions and visuals for each stage of the building process.
5. Offer peer support or group work for learners who benefit from collaborative learning. Break down complex steps into smaller, achievable tasks.
6. Offer a variety of use case categories and difficulty levels. Provide additional scaffolding (materials lists, design templates) for beginners while allowing advanced learners to explore more complex scenarios.
7. Offer 3D modelling software or simulations for learners who prefer a virtual approach. Provide alternative testing methods (e.g. using simulations) for those with limited physical capabilities.

Additional considerations:

1. Dedicate time for learners to explore and experiment with different materials (fabricated or local) to understand their properties and suitability for the robot design.
2. Encourage learners to adopt a design thinking approach. This involves sketching initial ideas, iterating on their designs based on testing, and refining their robots for optimal performance.
3. Guide learners on setting up appropriate testing procedures. This includes defining parameters for stability, load and impact testing, and establishing clear success criteria.
4. Ensure learners understand safety protocols when working with tools or building materials.
5. Encourage learners to document their design process, including sketches, material choices and test results. Facilitate a class discussion for reflection after testing, where learners share their experiences and learnings.

Key Assessment

Assessment Level 1: Identify two methods for creating a strong robot structure.

Assessment Level 2: Describe the steps involved in testing a robot structure for stability.

Assessment Level 3: A robot designed for line following in a competition needs to be lightweight and manoeuvrable. Explain how the choice of materials would differ from a robot designed for carrying heavy objects in a warehouse?

Assessment Level 4: A robot arm needs to be strong enough to lift objects but also be flexible enough to reach various positions. How can you design a structure that balances these seemingly opposing requirements?

Conclusion

In this practical lesson, you have learned how to design, build and test robot structures for specific use cases. Understanding use case requirements, selecting suitable materials and ensuring stability are essential steps in constructing reliable and efficient robotic systems. Through hands-on experience and testing, you have gained valuable insights into creating robust robots that can withstand forces and excel in various real-world applications. Keep experimenting, refining your designs, and pushing the boundaries of robotics to make innovative and impactful contributions to the field.

Weeks 19 & 20

Learning Indicator:

Sit *in groups and create robots using robotic kits and/or local materials to implement basic mechanics for actuations that make use of the following:*

- **GEARS (***Gear Ratios, Compound Gear Systems, Changing angle of rotation using gears, Using worm gears)*
- **VEHICLES (***Driving robots with single motors, Driving robots with two motors)*
- **MOVING WITHOUT TYRES (***Walking Machines)*
- **ARMS, WINGS & OTHERS (***Flapping Wings, Gripping Figures, Lifting Mechanisms)*

Theme or Focal Area: Creating Robots with Basic Mechanics - Exploring Gears, Vehicles and Moving Mechanisms

Introduction

This section will explore the world of basic mechanics in robotics by creating robots using fabricated robotic materials or locally available resources. Working in groups, you will get hands-on experience building robots with different actuation mechanisms, including gears, vehicles, walking machines, arms, wings, and more. By the end of this lesson, you will have created a variety of robots that demonstrate essential mechanical principles and actuation techniques.

Gears and Gear Ratios

Gears are mechanical components used to transmit motion and power between rotating shafts. They consist of toothed wheels or cylinders that mesh with each other to transmit torque, enabling speed and direction changes in machinery. Gears are essential components for transmitting motion and power in robots.

Understanding Gear Types: Different types of gears offer specific advantages and applications based on their designs and functionalities. Let's explore three common types of gears used in robotics:

1. **Spur Gears**: Spur gears are the most basic and commonly used type of gears. They consist of straight teeth mounted on parallel shafts, and the teeth mesh directly, resulting in a smooth and efficient transfer of motion. Spur gears are cost-effective to manufacture and provide precise speed and torque ratios. They are often used in applications where precise rotary motion is required, such as robot arms, robotic manipulators and transmission systems.

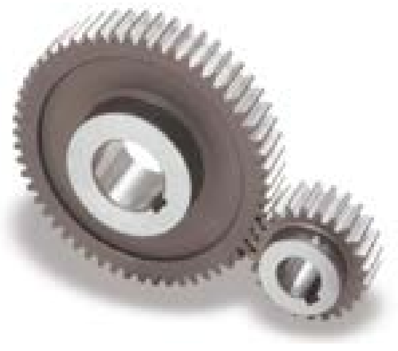


Fig. 19.1: *Spur Gears (KHK USA Metric Gears. KHK. (n.d.))*

***Applications in Robotics*:**

i. **Robotic Arm Joints**: Spur gears are frequently used in robotic arm joints to achieve precise rotational motion and control.

ii. **Gear Transmission**s: Spur gears are essential components in robot gearboxes for transmitting power from motors to various parts of the robot.

2. **Bevel Gears**: Bevel gears have teeth that are cut on conical surfaces and are used to transmit motion between intersecting shafts. They are ideal for changing the direction of motion by 90 degrees and are available in various configurations such as straight bevel gears and spiral bevel gears. Bevel gears offer smooth operation and are used in applications where space constraints require motion to change direction effectively.

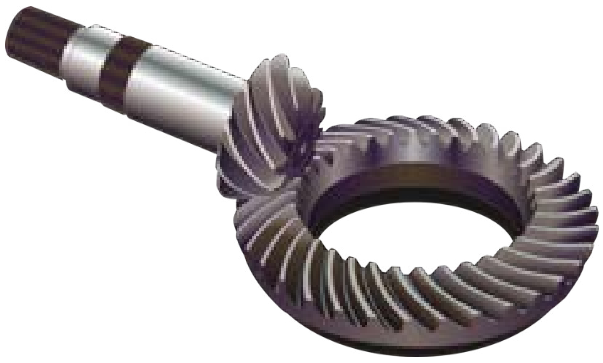


Fig. 19.2: *Bevel Gears (Thomas J.S. Cross. (n.d.). (PDF) generation of non-circular spiral bevel gears by face-milling method. A spiral bevel set)*

Applications in Robotics:

i. **Differential Mechanisms**: Bevel gears are often used in robot differentials to distribute power to the wheels and allow for smooth turning.

ii. **Gearboxes for Robotics with Non-parallel Shafts**: Bevel gears are used when the motor shaft and output shaft are not in the same plane.

3. **Worm Gears**: Worm gears consist of a cylindrical worm (screw) meshing with a toothed wheel (gear). They provide high gear ratios, making them ideal for applications requiring torque multiplication and reduced speed. Worm gears are highly efficient in one direction but have a lower efficiency in the reverse direction. Due to their self-locking nature, they are commonly used in applications where back-driving prevention is essential.

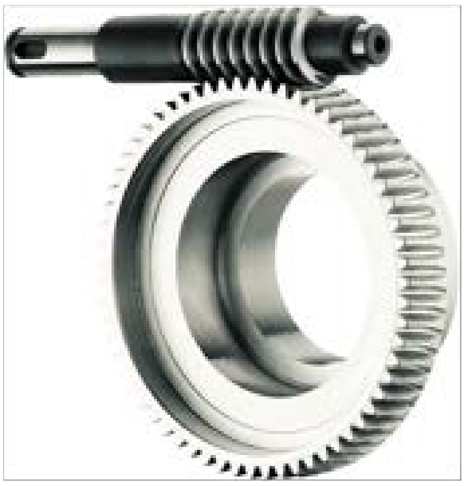


Fig. 19.3: *Worm Gears (Worm Gears. Stahl Gear & Machine Co. (2021, February 11))*

***Applications in Robotics*:**

i. **Robotic Arm and Gripper Mechanisms**: Worm gears are used in robotic arms and grippers to provide precise control and prevent back-driving.

ii. **Lifting Mechanisms**: Worm gears are employed in the lifting mechanisms of robots to lift heavy objects with reduced effort.

Overall, the selection of gears in robotics depends on the specific application requirements, including the need for precise motion control, direction changes, gear ratios and torque transmission. Each type of gear has its advantages and limitations, making them suitable for different robotic tasks. As robotics technology advances, new gear designs and applications will continue to contribute to the evolution of robotic systems.

Gear Ratios: Calculating Gear Ratios in Robotics: Gear ratios are crucial for controlling the speed and torque of robot movements. The gear ratio represents the ratio of the number of teeth on the driving gear (the gear connected to the motor) to the number of teeth on the driven gear (the gear connected to the robot's output shaft). Let's demonstrate how to calculate gear ratios using appropriate images and diagrams:

Step 1: Identify the Gears

Start by identifying the driving gear (Gear 1) and the driven gear (Gear 2). The driving gear is directly connected to the motor shaft, and the driven gear is connected to the robot's output shaft.

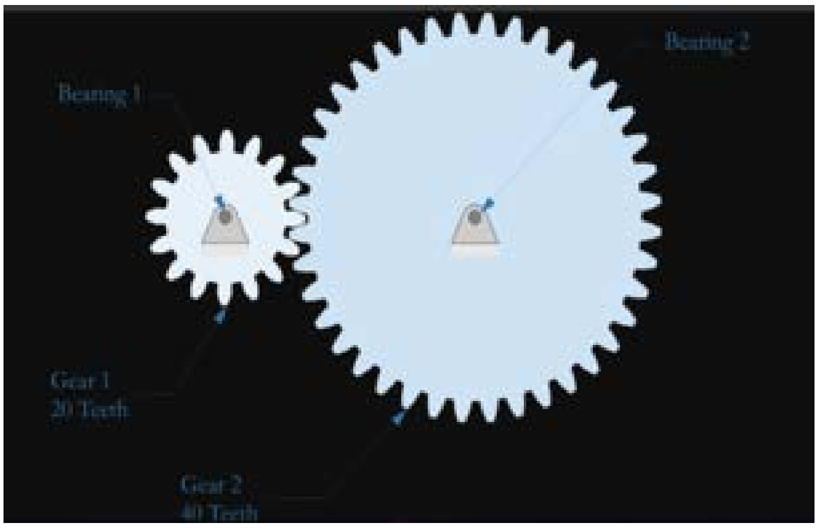


Fig. 19.4: *Diagram showing Gear 1 with 20 teeth and Gear 2 with 40 teeth*

Step 2: Count the Teeth

Count the number of teeth on each gear. Let's assume Gear 1 (the driving gear) has 20 teeth, and Gear 2 (the driven gear) has 40 teeth.

Step 3: Calculate the Gear Ratio

To calculate the gear ratio, divide the number of teeth on Gear 2 by the number of teeth on Gear 1.

Gear Ratio = Number of teeth on Gear 2 / Number of teeth on Gear 1

Gear Ratio = 40 / 20

This gives a figure of 2

The Gear Ratio = 2:1

This means that the driving gear has to rotate two full turns to move the driven gear ONCE.

Step 4: Interpret the Gear Ratio

The calculated gear ratio (2:1 in this case) represents how many times the driving gear (Gear 1) must rotate to make the driven gear (Gear 2) complete one full rotation. This ratio slows down the output movement to half the speed of the input.

Step 5: Speed and Torque Control

a. If the gear ratio is greater than 1 (e.g. 2:1), the output shaft will rotate at a slower speed than the motor shaft but with increased torque. This is known as a speed reduction gear, and it allows the robot to move slower but with more force.

b. If the gear ratio is less than 1 (e.g. 1:2), the output shaft will rotate at a faster speed than the motor shaft but with reduced torque. This is known as a speed-increasing gear that allows the robot to move faster but with less force.

Reference Link	QR Code
Everything about Gear Ratio - https://www.youtube.com/watch?v=40RX2HRKpwA	
Gear Ratio, Torque and Speed - https://www.youtube.com/watch?v=R1cxzDKBFuU	

By selecting appropriate gear ratios, robotics engineers can tailor the robot's movements to suit specific applications, optimising speed and torque based on the robot's tasks and requirements. The calculated gear ratio plays a significant role in achieving efficient and precise motion control for various robotic systems.

Compound Gear Systems: Compound gear systems combine multiple gears to achieve specific speed and torque requirements. Designing compound gear systems for different robot tasks requires careful consideration of the desired speed, torque and motion requirements. Follow these steps to design a compound gear system:

1. **Understand the Task Requirements**: Before designing the compound gear system, clearly define the robot task's speed and torque requirements. Determine the desired output speed and the amount of force or torque needed to perform the task effectively.
2. **Identify the Primary Motor and Output Shaft**: Identify the primary motor that will power the compound gear system. Determine the output shaft where the final motion or force will be delivered.
3. **Calculate the Gear Ratio**: Calculate the required gear ratio to achieve the desired output speed and torque. The gear ratio is the ratio of the number of teeth on the driven gear (output gear) to the number of teeth on the driving gear (input gear). The gear ratio determines how much the output shaft rotates concerning the input shaft.

   To calculate the gear ratio on systems that have more than two gears, work out the ratios between intermeshed gears, then multiply the ratios. For example:

   To find the overall gear ratio across three gears, multiply the individual gear ratios:

   Overall Gear Ratio=Gear Ratio (Gear 1 to Gear 2) ×Gear Ratio (Gear 2 to Gear 3)

For example, if Gear 1 to Gear 2 has a ratio of 2:1 and Gear 2 to Gear 3 has a ratio of 3:1, the overall gear ratio would be 6:1

4. **Choose Gear Types and Sizes**: Select the appropriate gear types based on the task requirements. Common gear types include spur gears, bevel gears and worm gears.
   a. ***Spur gears***: Provide straight-toothed gears that are ideal for transmitting motion between parallel shafts.
   b. ***Bevel gears***: Transmit motion between intersecting shafts and are suitable for changing the direction of rotation.
   c. ***Worm gears***: Use a screw-like gear to achieve a large gear reduction and are effective for driving heavy loads.
5. **Arrange Gear Components**: Arrange the gears in the compound gear system to achieve the desired gear ratio. The arrangement may involve multiple stages of gears, each providing a specific reduction or increase in speed and torque.
6. **Consider Efficiency and Backlash**: Keep in mind the efficiency of the gear system, as energy losses can occur during gear engagement. Additionally, account for backlash, which is the play or clearance between the gear teeth. Minimising backlash ensures smoother and more precise movements.
7. **Ensure Adequate Support and Alignment**: Properly support and align the gears within the system. Use high-quality bearings to reduce friction and ensure smooth operation. Misalignment can lead to premature wear and reduced efficiency.
8. **Test and Optimise**: Once the compound gear system is assembled, test it under different load conditions to verify its performance. Make adjustments if necessary to achieve the desired output speed and torque.
9. **Consider Safety and Material Selection**: Pay attention to safety considerations, especially when dealing with high torque applications. Choose materials that can withstand the forces and loads involved in the robot task.
10. **Iterate and Improve**: Robot design is an iterative process. Continuously gather feedback and data from the robot's performance to identify areas for improvement. Modify the compound gear system as needed to achieve better results.

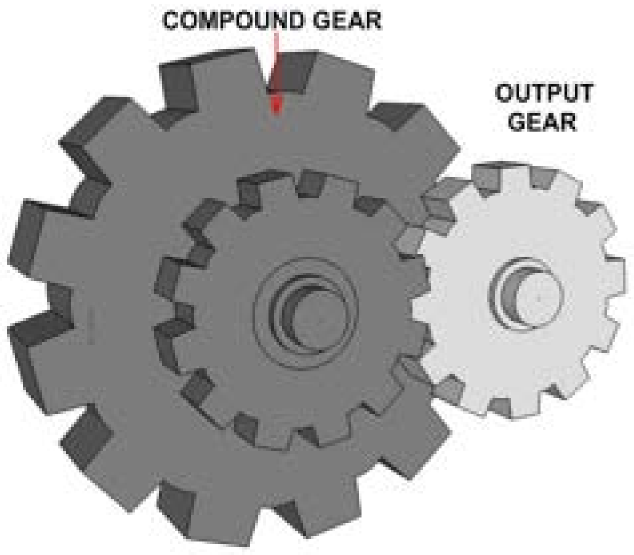


Fig. 19.5: *Compound Gear Systems (V. Ryan (© 2010-16), GEARS AND GEAR SYSTEMS)*

Determining Angle of Rotation using Gears

When two gears mesh, they transmit motion from one to the other. The number of teeth on each gear determines the gear ratio, which affects the angle of rotation. By choosing gears with different tooth counts, you can alter the output angle of rotation relative to the input angle.

For example,

If a small gear (with fewer teeth) drives a larger gear (with more teeth), the larger gear will rotate more slowly, producing a greater angle of rotation. In a robot gripper mechanism, a small driving gear and a larger gear will make the gripper open wider.

Conversely, if a larger gear drives a smaller gear, the smaller gear will rotate more quickly but with a smaller angle of rotation. In a robot gripper mechanism, a large driving gear and a smaller gear will make the gripper open a fraction.

This principle is often used in robotic arms and manipulators to control the precise movement of end-effectors or grippers.

Converting Rotational Motion into Linear Motion

Rack and Pinion Mechanism: One common way to convert rotational motion into linear motion is by using a rack and pinion mechanism. The pinion (a gear) meshes with a linear gear known as a rack. As the pinion rotates, it causes the rack to move linearly.

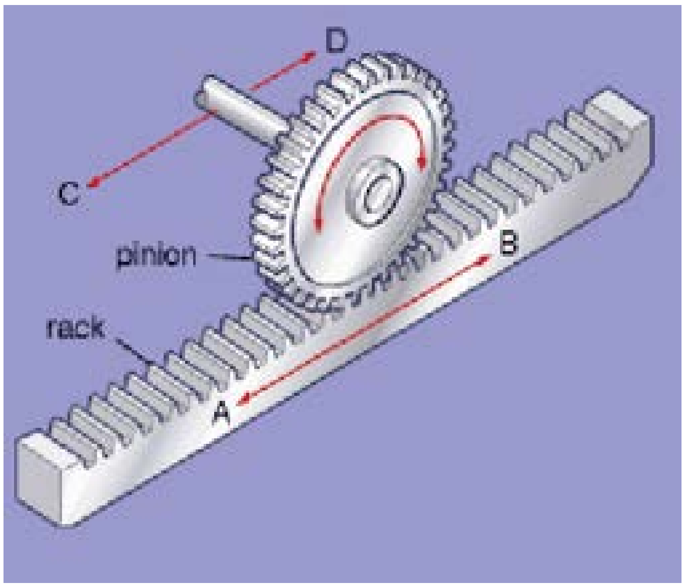


Fig. 19.6: *Rack and Pinion Mechanism*

This mechanism is commonly employed in various robotic systems, such as CNC machines, linear actuators and steering systems in vehicles.

Converting Linear Motion into Rotational Motion

Lead Screw Mechanism: Another way gears can transform motion is by converting linear motion into rotational motion using a lead screw mechanism. A lead screw is a threaded rod that rotates within a nut. As the lead screw turns, it pushes or pulls the nut along its length, converting the linear motion of the nut into rotational motion of the lead screw.

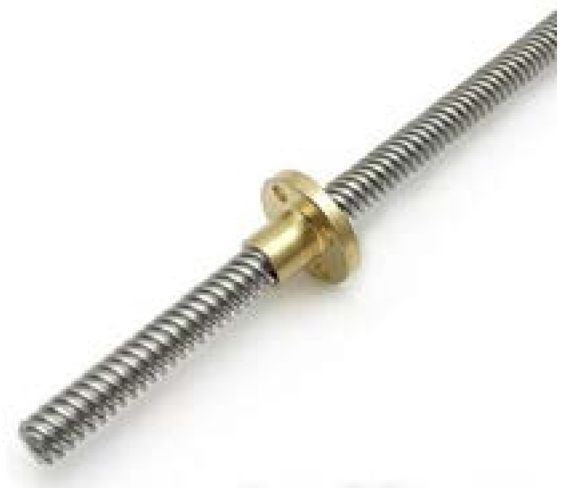


Fig. 19.7: *Lead Screw Mechanism*

Lead screws are frequently used in robotic systems for precise positioning, lifting mechanisms, and in combination with motors to achieve linear motion.

Combining Gears in Robotics: Achieving Functionality through Synergy

Robots often rely on a combination of different gear types to achieve specific functionalities. Here is an exploration of some common scenarios:

1. **Precise Gripper Control with Powerful Base Movement (Robot Arm):**
   - **Gears Used:**
     - **Worm Gear:** Offers a high reduction ratio for precise control of the gripper's opening and closing.
     - **Spur Gears:** Deliver efficient power transfer for robust arm movement.
   - **Why Combine?**
     - A worm gear provides fine rotational control over the gripper's delicate movements.
     - Spur gears efficiently transfer power from the motor for strong and stable arm movement.
   - **Example Activity:** A robot arm picks up a small object with its gripper (controlled by the worm gear) while the base rotates smoothly (powered by spur gears) to position itself for further actions.
2. **Articulated Steering with Power Delivery (Car Robot):**
   - **Gears Used:**
     - **Bevel Gears:** Change the direction of rotation by 90 degrees, allowing the steering wheel rotation to translate to perpendicular wheel movement.
     - **Spur Gears:** Efficiently transfer power from the motor to the wheels for driving.
   - **Why Combine?**
     - Bevel gears redirect the steering wheel's rotational input for perpendicular wheel control.
     - Spur gears ensure smooth power transfer from the motor to propel the car.
   - **Example Activity:** A car robot navigates a course, using the steering wheel to control direction while the motor and spur gears provide the driving force.

3. **Linear Actuator with Efficient Power Transfer (Gripper or Lifting Platform):**
   - **Gears Used:**
     - **Rack and Pinion Gears:** Convert the rotary motion of a motor into linear motion for extending and retracting the gripper or lifting platform.
     - **Spur Gears:** Can be used within the rack and pinion mechanism to connect the motor to the pinion gear for efficient power transfer.
   - **Why Combine?**
     - Rack and pinion gears translate rotary motion into the linear movement needed for the gripper or platform.
     - Spur gears (if used) ensure smooth power transfer from the motor to the rack and pinion mechanism.
   - **Example Activity:** A robot arm uses a rack and pinion mechanism in its gripper to extend and retract for grasping objects. Spur gears within the mechanism might ensure efficient power transfer from the motor.

Vehicles with Single and Dual Motors:

Build robots with wheels or tracks driven by single and dual motors

In this robotics project, we will build robots with either wheels or tracks for locomotion. We will explore how to control them using both single and dual motor configurations to achieve different driving capabilities. This hands-on activity will provide valuable insights into robot mechanics and motor control.

Materials Needed:

- Robot chassis with wheels or tracks
- Motors (both single and dual motor configurations)
- Motor driver board or H-bridge
- Micro controller (e.g. Arduino or Raspberry Pi)
- Battery pack or power supply
- Wheels (if using wheel-based locomotion)
- Tracks (if using track-based locomotion)
- Connecting wires and tools for assembly

Instructions:

- ***Step 1***: Assemble the Robot Chassis: Start by assembling the robot chassis, ensuring that it can accommodate either wheels or tracks, depending on your preference. Secure the motors in place and mount the wheels or tracks on the chassis.
- ***Step 2***: Single Motor Configuration: For single motor configuration, connect one motor to the motor driver board or H-bridge. Wire the motor driver board to the microcontroller, providing control over the motor's speed and direction.
- ***Step 3***: Dual Motor Configuration: For dual motor configuration, connect two motors to the motor driver board or H-bridge. Wiring the motors in parallel will allow synchronised control, enabling the robot to move forward or backwards smoothly.
- ***Step 4***: Motor Control Programming: Write a motor control programme using your chosen micro controller (Arduino or Raspberry Pi). Utilise the appropriate programming language (e.g. Arduino IDE for Arduino boards) to control the motor's speed and direction.

- ***Step 5***: Implementing Forward and Backward Movement: Test the robot's forward and backward movement by adjusting the motor speed and direction in the programme. Observe how different motor speeds impact the robot's movement capabilities.
- ***Step 6***: Turning Capabilities: Experiment with turning capabilities in both single and dual motor configurations. In a single motor configuration, use differential control of the wheels or tracks to achieve turning. In dual motor configuration, adjust the speed of the motors on each side to turn the robot smoothly.
- ***Step 7***: Testing Different Terrains: Place the robot on various terrains (e.g. smooth surface, carpet or rough terrain) to observe its performance. Analyse how the robot adapts to different surfaces and how its driving capabilities change.
- ***Step 8***: Fine-Tuning and Optimisation: Fine-tune the motor control programme to optimise the robot's performance. Adjust motor speeds and turning parameters to achieve smoother movements and enhanced manoeuvrability.
- ***Step 9***: Experimenting with Dual Motor Differential Steering: If using wheels, experiment with dual motor differential steering. By controlling each motor separately, you can enable the robot to rotate in place, providing precise turning capabilities.

Reference Link	QR Code
DIY Arduino Car with 1 DC Motor, Ultrasonic Sensor & VEX Robot Parts https://www.youtube.com/watch?v=Z3FJRHGcyqo	

Moving Without Tyres

Walking machines are a fascinating alternative method for robot locomotion, designed to navigate challenging terrains that may be difficult for wheeled or tracked robots. These robots imitate the movements of living creatures such as insects, animals or humans, to traverse uneven surfaces, climb obstacles and adapt to various environments. The concept of walking machines draws inspiration from nature's designs, allowing robots to achieve increased agility, versatility and adaptability. Let's explore some key features and advantages of walking machines:

1. Locomotion Principle: Walking machines utilise legged locomotion to move. The robot's legs replicate the walking or crawling motion of animals, giving them the ability to take steps, lift off the ground and make adjustments to maintain stability on uneven terrain.
2. Versatility in Terrain: One of the primary advantages of walking machines is their ability to navigate a wide range of terrains. They can move over rocky surfaces, climb stairs, traverse sandy or muddy landscapes and even cross gaps that would be challenging for wheeled robots.
3. Stability and Balance: Walking machines are designed to maintain stability and balance during locomotion. They can adapt their leg movements and body posture to keep themselves upright even on sloped or uneven surfaces.
4. Obstacle Negotiation: Walking machines can effectively negotiate obstacles by adjusting their leg movements to step over or climb onto objects in their path. This makes them valuable for search and rescue operations in disaster-stricken areas.
5. Energy Efficiency: In certain situations, walking machines can be more energy-efficient than wheeled or tracked robots. The legged locomotion requires less energy to traverse certain terrains compared to rolling or sliding on wheels or tracks.

6. Adaptability to Changing Environments: Walking machines' adaptability allows them to function well in environments that may change rapidly, such as disaster zones or rough outdoor terrains.
7. Biomimicry and Bio-inspired Designs: Many walking machines take inspiration from the locomotion of animals and insects, leading to bio-inspired designs. These robots can mimic the gait of animals such as insects, quadrupeds or even bipeds, depending on the application.
8. Research and Exploration: Walking machines have applications in scientific research and planetary exploration. Their ability to traverse rough landscapes on other planets such as Mars, can provide valuable data and insights for space missions.

While walking machines offer many advantages, they also present challenges such as control complexity and higher degrees of freedom in legged locomotion. Engineers and roboticists continue to work on refining the design and control of walking machines to enhance their performance and enable them to tackle even more challenging terrains and tasks.

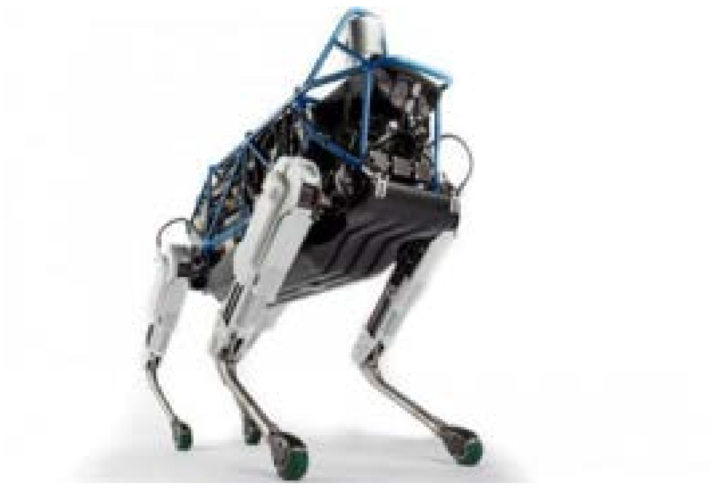


Fig. 19.8: *Walking Robots (Riccio, Boston Dynamics' 10 robots that will change the world: Near future 2022)*

Arms, Wings & Other Actuators

Flapping Wing Aerial Robot: Designing a robot with flapping wings mimics the flight of birds and insects. These robots are perfect for exploring challenging terrains and aerial surveillance.

Components Required:

- Flapping Wing Mechanism
- Fuselage or Frame
- Motors and Propellers
- Microcontroller or Flight Controller
- Sensors (optional for autonomous flight)

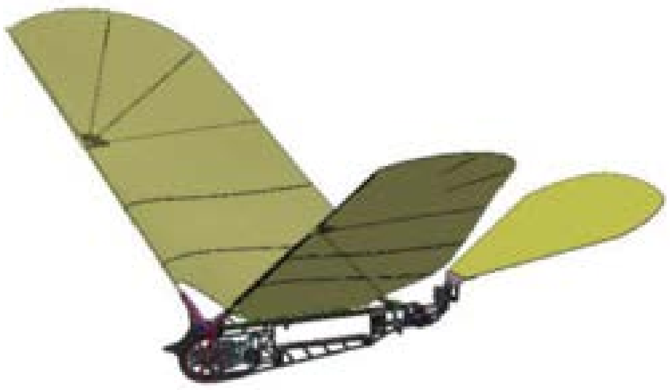


Fig. 19.9: *Flapping Wing Aerial Robot (Zhong & Xu, Power modelling and experiment study of large flapping-wing flying robot during forward flight 2022)*

Gripping Figure Pick-and-Place Robot: The gripping figure robot is designed to pick up objects from one location and place them in another. This robot is valuable in industrial automation and logistics applications.

Components Required:

- Robotic Arm with Gripper
- Base or Chassis
- Motors for Arm Movement
- Microcontroller or PLC (Programmable Logic Controller)
- Sensors (optional for object detection)

Fig. 19.10: *Gripping Figure Pick-and-Place Robot (Robotic ARM isolated Images-Adobe Stock)*

Lifting Mechanism Robot: The lifting mechanism robot is designed to carry and transport heavy objects with precision. This robot is useful in scenarios where human strength is insufficient or dangerous.

Components Required:

- Robotic Arm with Lifting Mechanism
- Base or Chassis
- Motors for Arm Movement
- Microcontroller or PLC
- Sensors (optional for object detection and safety)

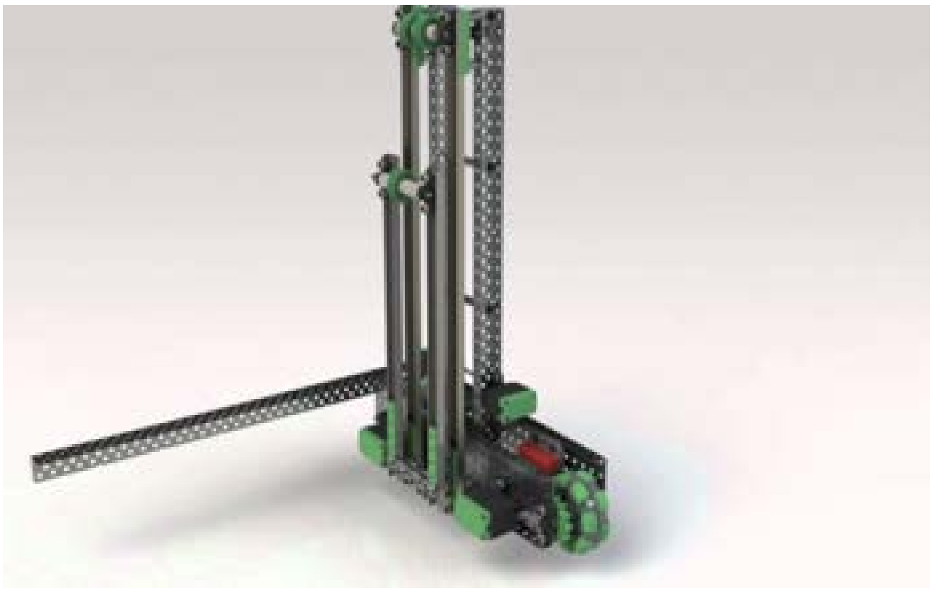


Fig. 19.11: *Lifting Mechanism Robot (254 VEX talk: Episode 3 - an overview of lifts 2012)*

Learning Tasks

Depending on the available time or resources, administer one or more of the following learning tasks to help learners reinforce understanding and acquire new knowledge or skills.

Design Your Custom Robot

1. Learners select one of the three robot types (flapping wing, gripping figure, or lifting mechanism) to build. Plan the robot's components and dimensions based on the intended use and task requirements.
2. Learners assemble the chosen robot using the appropriate materials and actuation mechanisms.
3. Learners test the robot's functionality, making necessary adjustments for optimal performance.

Pedagogical Exemplars

Project-Based Learning:

1. Organise learners into mixed groups to work collaboratively on determining gear ratios and output characteristics of gear trains.
2. Assign problem-solving tasks that require learners to calculate gear ratios and predict output speeds or torques based on given parameters.
3. Provide hands-on activities where learners build and test gear systems to reinforce their understanding of gear ratios and compound gear arrangements.

4. Encourage learners to document their project findings and present their results to the class, promoting communication and presentation skills.

Experiential Learning:

1. Provide real-world scenarios or use cases where gears are essential for controlling speed, torque, timing and direction of forces.
2. Allow learners to work in groups to design, build and test fully functional robotic subsystems that utilise gears to meet the given use case requirements.
3. Present challenges or constraints that require learners to apply their knowledge of gears creatively to solve problems and achieve desired outcomes.
4. Facilitate reflection sessions where learners evaluate their designs, identify areas for improvement and iterate on their solutions to optimise performance.

Key Notes for Teachers:

1. Create opportunities for learners to actively engage with the material through discussions, hands-on activities, and project work.
2. Offer guidance and support as learners navigate the learning tasks, ensuring they understand the concepts and objectives.
3. Foster collaboration among learners to promote teamwork and peer learning.
4. Encourage learners to think critically and apply their knowledge creatively to solve problems and design functional robotic subsystems.
5. Stress the importance of documenting project findings, calculations, and design iterations to reinforce learning and facilitate assessment.

Key Assessment

Assessment Level 1: Explain how gears can be used to speed up or slow down the actions of robots.

Assessment Level 2: Explain the difference between a single motor and a dual motor configuration in robots with wheels or tracks.

Assessment Level 3: You are designing a robot arm that needs to move slowly with high precision. Explain what type of gear and gear ratio you would choose?

Assessment Level 4: How can backlash (clearance between gear teeth) affect the performance of a compound gear system in a robot? How can it be minimised?

Conclusion

In this lesson, you have explored basic mechanics in robotics, creating robots with gears, vehicles, walking capabilities and various actuation mechanisms. By combining different mechanical principles, you have built versatile robots capable of performing various tasks. Robotics is a field that offers endless possibilities, and understanding these fundamental mechanics will set the foundation for more advanced robotic systems. Continue to innovate and experiment, using your creativity to design robots that can solve real-world challenges and push the boundaries of robotics.

Section 4 Review

In this section on Robot Construction, learners delved into fundamental concepts essential for designing and building stable and functional robots. They explored the effects of mass and centre of gravity on stability and learning strategies for creating structures that withstand forces. Through hands-on activities, they applied their understanding to build structures for specific use cases, testing them rigorously for stability and durability.

In the following weeks, learners worked collaboratively to create robots using both robotic kits and local materials. They applied basic mechanical principles such as gears, gear ratios and compound gear systems to implement various actuations. Additionally, they explored diverse locomotion methods, including driving robots with single and dual motors, walking machines, and mechanisms for gripping, lifting and flapping wings.

Throughout the section, learners demonstrated a general understanding of rigid bodies, design processes for stable structures, and the ability to create robots with moving parts. This comprehensive review encapsulates their journey in mastering the fundamentals of robot construction and programming.

Additional Reading

Link	QR Code
Gear Types, Design Basics, Applications and More - Basics of Gears https://www.youtube.com/watch?v=ZhDO16FDmxA&t=5s	
Everything about Gear Ratio - https://www.youtube.com/watch?v=40RX2HRKpwA	
Gear Ratio, Torque and Speed - https://www.youtube.com/watch?v=R1cxzDKBFuU	

References

1. Wikimedia Foundation. (2023, January 6). *Mass versus weight*. Wikipedia. https://simple.wikipedia.org/wiki/Mass_versus_weight
2. Charbel Dalely, Tawk Gursel Alici. (2021a, February). *Comparison of the proposed robotic arm system and related designs. | download scientific diagrams*. A Review of 3D-Printable Soft Pneumatic Actuators and Sensors: Research Challenges and Opportunities. https://www.researchgate.net/figure/Comparison-of-the-proposed-robotic-arm-system-and-related-designs_tbl1_372799179

3. *How much does a robot arm cost?* Standard Bots. (2023, October 4). https://standardbots.com/blog/how-much-does-a-robot-arm-cost

4. *Indian tightrope Walker Stock Photos - Free & royalty-free stock photos from Dreamstime*. Dreamstime. (n.d.). https://www.dreamstime.com/photos-images/indian-tightrope-walker.html

5. *KHK USA Metric Gears*. KHK. (n.d.). https://www.khkgears.us/products/spur-gears

6. Thomas J.S. Cross. (n.d.). *(PDF) generation of non-circular spiral bevel gears by face-milling method*. A spiral bevel set. https://www.researchgate.net/publication/298733810_Generation_of_Non-Circular_Spiral_Bevel_Gears_by_Face-Milling_Method

7. *Worm Gears*. Stahl Gear & Machine Co. (2021, February 11). https://stahlgear.com/project/worm-gears/

8. V. Ryan (© 2010-16) *GEARS AND GEAR SYSTEMS*. Spur Gears and simple gear trains. https://technologylearner.com/gears1/gears1.htm

9. Britannica, T. Editors of Encyclopaedia (2007, April 4). *rack and pinion*. *Encyclopedia Britannica*. https://www.britannica.com/technology/rack-and-pinion

10. Riccio, G. (2022, March 26). *Boston Dynamics' 10 robots that will change the world: Near future*. Futuro Prossimo. https://en.futuroprossimo.it/2022/03/10-robot-boston-dynamics/

11. Zhong, S., & Xu, W. (2022, March 21). *Power modelling and experiment study of large flapping-wing flying robots during forward flight*. MDPI. https://www.mdpi.com/2076-3417/12/6/3176

12. *Robotic ARM isolated images – browse 37,516 stock photos, vectors, and video*. Adobe Stock. (n.d.). https://stock.adobe.com/search?k=robotic%2Barm%2Bisolated

13. *254 VEX talk: Episode 3 - an overview of lifts*. An Overview of Lifts. (2012, January 10). https://challenges.robotevents.com/challenge/18/entry/498

SECTION 8: PROGRAMMING ROBOTS

Strand: **Robot Construction & Programming**

Sub-Strand: Programming Robots

Learning Outcomes:

1. *Create programmes that make use of decision structures and loop conditions to control robots.*
2. *Design and programme Finite State Machines*

Content Standards:

1. Establish the essence of programming and demonstrate skills in the use of programming constructs for robots.
2. Demonstrate understanding and programming skills in the implementation of Finite State Machines (FSM).

INTRODUCTION AND SECTION SUMMARY

This section will focus on programming robots. It will expose learners to the various programming categories and introduce them to block-based coding, a fun and beginner-friendly approach to programming. Learners will explore how to drag and drop these blocks to build instructions for their constructed robots without memorising complex codes. Through various practical activities, learners would familiarise themselves with the LEGO Education Spike App and the LEGO Spike Prime kit. Towards the end of the section, they will practise with robots that are categorised as either Finite State Machines or controlled feedback loop systems.

The weeks covered by the section are:

Weeks 21 - 23:

1. Create computer programmes from pre-designed flowcharts that have single-decision conditions.
2. Create computer programmes from pre-designed flowcharts that have nested decision conditions.

Week 24:

1. Create computer programmes from pre-designed flowcharts that have a controlled feedback loop with loop interrupts.
2. Formulate and programme FSMs for controlling different use cases.

SUMMARY OF PEDAGOGICAL EXEMPLARS

This section uses different teaching strategies to engage learners and equip them with a good understanding of programming robots. Weeks 21-23 introduce learners to block-based programming first using an inquiry-based learning approach where learners are made to brainstorm and discuss the concept of giving instructions to machines using either block-based or text-based programming through research and presentations. Then, using an experiential learning approach, working in mixed-ability groups, they are introduced to block-based programming using the LEG Education Spike App. They follow video tutorials, assemble robots and programme them, with the facilitator providing support throughout.

Later in weeks 21-23, using talk for learning approach, learners are made to review flowchart diagrams and their role in programme planning. An interactive presentation introduces common flowchart symbols and their corresponding block-based code equivalents. Then, using a problem-based learning approach, learners work in mixed-ability groups to translate flowchart diagrams into block-based code using the LEGO Education Spike App. They test their programmes on pre-built robots, debug any errors and receive feedback from the teacher.

Week 24 focuses on helping learners understand the differences between controlled feedback loop systems and finite state machines (FSMs). Utilising differentiated instruction, the facilitator tailors the explanations to cater for different learning needs, starting with a basic overview and progressing to more technical details for advanced learners. Using the managing talk for learning approach, a moderated discussion encourages learners to share their research findings and understanding of controlled feedback loops and FSMs. They use a Venn diagram or a chart to highlight the similarities and differences between these systems. The teacher facilitates the discussion, provides feedback and clarifies any questions. Finally, using a problem-based learning approach, working in mixed-ability groups, learners tackle a challenge: identifying the states needed for a robot to navigate a maze. They then draw an FSM diagram and test it using the LEGO Education Spike App. The teacher provides support and encourages learners to explain their reasoning behind the chosen states.

ASSESSMENT SUMMARY

Following each thematic area in this section, assessments gauge learner learning. These come in two forms: learning tasks and key assessments. Learning tasks, primarily formative, focus on solidifying understanding and acquiring new knowledge or skills. Facilitators guide these activities to enhance the learning process. In contrast, key assessments, typically summative, evaluate learner mastery after instruction. These are often given as homework or quizzes outside of class. Instructors have the flexibility to choose the assessment types that best suit their learners and learning objectives. However, it is advisable that instructors, at least, guide learners to do one of the learning tasks.

Week 21 - 23

Learning Indicators:

1. *Create computer programmes from pre-designed flowcharts that have single-decision conditions.*
2. *Create computer programmes from pre-designed flowcharts that have nested decision conditions.*

Theme or Focal Area: Introduction to Programming with Block-Based Coding

Introduction

In this lesson, we will explore the basics of how to give instructions to robots, just like giving a recipe to follow. We will see how robotics uses programming to control robots and make them perform specific tasks. We will learn about block-based coding, a fun and beginner-friendly approach that uses visual building blocks to represent programming concepts.

What is Programming?

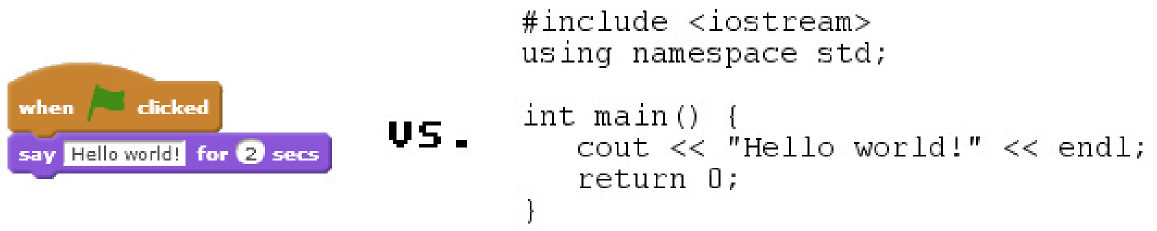
Programming, in essence, is giving a set of instructions to a computer to tell it what to do and how to do it. It is like creating a recipe for the computer to follow, but instead of ingredients and steps for cooking, you use programming languages to define actions and achieve specific results.

Relating programming to robotics, imagine you have a robot you want to control. Programming is like giving the robot instructions through a special language it understands. This language tells the robot exactly what to do, step by step, to complete a task. So simply put, programming can be described as the art of telling a computer, computer-based device or robot what to do through a set of instructions.

The language used for programming is known as a programming language or coding language. A programming language usually comprises a set of instructions (codes) and is often governed by syntax (rules that show how to use them correctly).

There are several programming languages in our world today. Some very common examples include C, C++, Python, Scratch, Snap, Java, Ruby, Javascript, and others. There are several ways in which programming languages can be categorised, but one simple way of categorising them is whether they are (visual) block-based or text-based.

1. **Block-Based Coding/Programming:** This is a beginner-friendly approach that uses visual building blocks to represent programming concepts. Just like building with LEGOs, you drag and drop these blocks together to create your programme. Examples include Scratch, Snap, and Blockly .
2. **Text-Based Coding/Programming:** This is a more traditional approach that uses written text commands to create programmes. While powerful, it can be more challenging to learn for beginners. Examples include C, C++, Python, and Java.



```
#include <iostream>
using namespace std;

int main() {
   cout << "Hello world!" << endl;
   return 0;
}
```

Fig. 21.1: *Block-based in Scratch vs Text-based Programming in C (REFERENCE)*

Block-Based Coding: A Stepping Stone to Programming

Block-based coding offers many advantages for beginners, especially young learners. Some of these advantages include:

1. **Easy to Learn:** The drag-and-drop interface makes it intuitive and requires no memorisation of complex syntax (like punctuation rules in written language)
2. **Visually Engaging:** The colourful blocks and animations make learning fun and engaging, keeping learners motivated.
3. **Reduces Errors:** Block-based coding often prevents syntax errors (common mistakes in text-based coding) since the blocks fit together logically. This allows you to focus on the core programming concepts.
4. **Experimentation Encouraged:** The visual nature of block-based coding makes it easy to experiment and try different things without worrying about breaking your programme.

Based on these advantages, in our current programming journey in robotics, we will start with Block-based coding.

A Brief History of Block-Based Coding

In 2003, MIT developed Scratch, one of the most popular block-based coding platforms. Scratch's success has led to the widespread adoption of block-based coding in schools and educational institutions, making programming accessible to a new generation of learners.

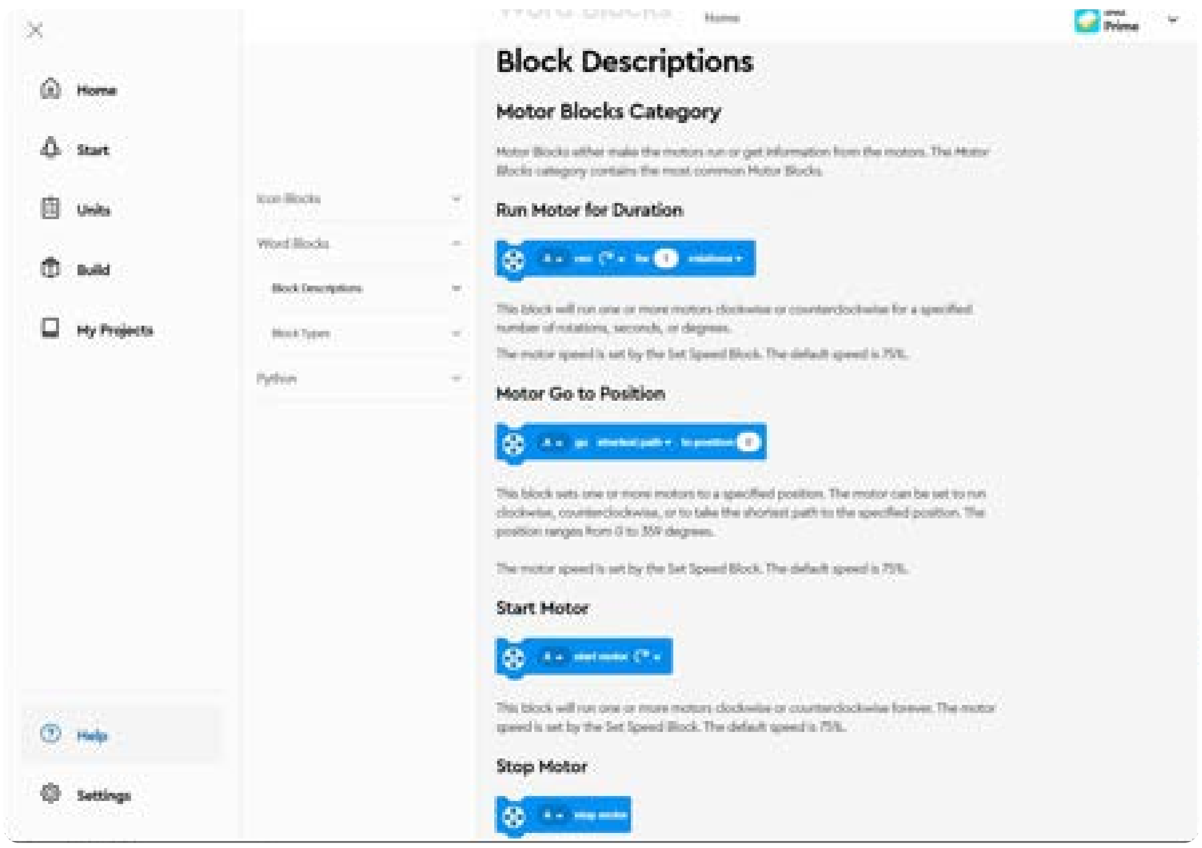
Block-based coding allows you to create a variety of interactive projects, including:

- **Games:** Design chase games, clicker games or even ping-pong games using blocks to control characters and objects.
- **Interactive Animations:** Bring your imagination to life by creating animated stories and characters.
- **Program Robots**: Create on-screen sprites that can be programmed to move and interact with their environment. This can simulate basic robot behaviour such as navigating a maze, avoiding obstacles, etc.

Block-based Programming with LEGO Education SPIKE™! Application/IDE

As mentioned in Week 14, LEGO® Education SPIKE™! App allows learners of all skill levels to programme their built robots using either Icon-block, word-block or text-based coding. The Icon-block option allows learners to programme using blocks which are similar to those used in LEGO MINDSTORM EV3 IDE. Word-Blocks, on the other hand, are similar to blocks used in Scratch and text-based coding using the Python programming language.

For now, our focus would be on using the word-block programming option. LEGO® Education SPIKE™! app uses this block-based programming in a fun and intuitive way to bring LEGO creations to life. Imagine building with colourful LEGO bricks, but instead of creating physical structures, you are building programmes. This block-based programming uses visual blocks that snap together, representing different programming concepts. Just like LEGO bricks, these blocks connect logically to create your programme. The App provides learners with a detailed tutorial on the use of each of the blocks. This can be found in the "Help" Section, as demonstrated in Fig. 21.2.



Block Descriptions

Motor Blocks Category

Motor Blocks either make the motors run or get information from the motors. The Motor Blocks category contains the most common Motor Blocks.

Run Motor for Duration

This block will run one or more motors clockwise or counterclockwise for a specified number of rotations, seconds, or degrees.

The motor speed is set by the Set Speed Block. The default speed is 75%.

Motor Go to Position

This block sets one or more motors to a specified position. The motor can be set to run clockwise, counterclockwise, or to take the shortest path to the specified position. The position ranges from 0 to 359 degrees.

The motor speed is set by the Set Speed Block. The default speed is 75%.

Start Motor

This block will run one or more motors clockwise or counterclockwise forever. The motor speed is set by the Set Speed Block. The default speed is 75%.

Stop Motor

Fig 21.2: *Block Descriptions of each block under the Help Section of LEGO Education SPIKE™! app*

To familiarise oneself with these building blocks, the following video resources have been provided. The first is a playlist of eight videos which provides a quick overview of using the LEGO® Education SPIKE™ Prime Kit. It provides tutorials for using both the word-blocks and Python Programming Language. However, learners can, for now, focus on using the word-blocks.

The second playlist of seventeen videos rehashes some of the points made in the first, and using some practical examples, it provides a detailed explanation of how to use the various blocks. Learners can focus on the first fifteen videos in this playlist.

The resources are as follows:

Reference Links	QR Code
SPIKE Prime Tutorials https://www.youtube.com/playlist?list=PL_zXBalpjbu33gw5CML3DtL7fN8640qku	
Lego Spike Prime https://www.youtube.com/playlist?list=PLS9qLR8VoFA62KcAzsUfAOQgLrEXCp78B	

Practising with these videos may take a while, so they should be spread judiciously over the ensuing weeks.

Learning Tasks

Based on the content covered under this thematic area, learners will

1. define programming and its role in robotics.
2. differentiate between block-based and text-based coding.
3. identify advantages of block-based coding for beginners.
4. explore the LEGO Education SPIKE app as a block-based programming platform.
5. get familiar with the various building blocks in the LEGO Education SPIKE app used in programming robots.

Pedagogical Exemplars

The goal of this lesson is for all learners to get familiar with block-based programming. Consider the following keynotes when administering the suggested pedagogical approaches in the curriculum:

1. Recognise and capitalise on the shared characteristics among learners while also addressing their individual differences, including interests, readiness levels and learning styles.
2. Offer multiple pathways for learners to engage with the content. This could involve providing varying levels of detail, from basic concepts to in-depth explorations, to accommodate different learning needs. The key thing is that the learning outcome set for the lesson is achieved among all learners.
3. **Inquiry-based learning:** Learners are made to sit in mixed-ability groups with guided questions to briefly differentiate between text-based and block-based programming.
   a. Using this approach, start with a brainstorming session. Ask learners: "What are some ways we can give instructions to a machine?" Write their answers on the board.
   b. Based on their submissions, explain the concept of programming as giving a set of instructions to a computer and differentiate it from using a computer.
   c. Briefly explain text-based programming and block-based programming. Though brief, your explanation should be enough to paint a good picture of these programming categories. This is to enable learners to carry out the next task.
   d. Form mixed-ability groups and give each group some guided questions to help them research further on text-based programming and block-based programming. The questions should seek to help learners describe these terms, differentiate them and state the advantages each has over the other.
   e. Allow learners to present the result of their research, showing a comparison chart (image, table, presentation, etc.) highlighting the key differences between block-based and text-based coding.
   f. Be ready to help learners who may need clarification.
4. **Experiential Learning:** Learners work in mixed-ability groups to familiarise themselves with block-based coding using the LEGO Education Spike App.
   a. Using this approach, start by showing a short video (2-3 minutes) demonstrating block-based coding in action (e.g. Scratch tutorial, LEGO Spike tutorial).
   b. Create groups with members having a mix of design, robot assembly, programming, communication, and presentation skills. This fosters collaboration and uses each learner's strengths.
   c. Provide each learner group with a watching guide as they watch the overview of the LEGO Spike Prime Tutorials (the first video playlist). All groups could watch the video from a single screen while putting down their observations and questions.

d. After the playlist, allow learner groups to discuss their observations and attempt to get answers from their peers. Circulate among groups, providing support, clarification and guidance as needed.

e. After the discussion, walk all learners on how to access the additional resources (video playlists, installation files, Spike Prime Kits, etc.). Where need be, provide multiple resources to allow learners of different interests.

f. In the ensuing weeks, allow each learner group to have access to a video playback device where they can engage with the videos in the second playlist. They will need to install the applications, assemble the robots and programme them as they follow the video tutorial.

g. Circulate among groups, providing support, clarification and guidance as needed. Ensure that no learner is left behind.

h. Provide additional support or scaffolding for learners who may struggle with the task. Provide clarification for learners who may need it.

i. Provide feedback and reinforcement to reinforce learning and encourage continued engagement.

Key Assessment

Assessment Level 1: Scratch is an example of a block-based coding platform. (True/False)

Assessment Level 2: Briefly explain the difference between block-based coding and text-based coding.

Assessment Level 2: Create a chart comparing and contrasting the advantages and disadvantages of block-based and text-based coding for beginners.

Assessment Level 3: In groups, use the LEGO Education SPIKE app to create a simple programme that controls a robot to complete a specific task (e.g. following a line, avoiding obstacles, etc.). Present your programme to the class and explain how it works.

Conclusion

This lesson has been an introduction to block-based coding and robotics. We have learned what programming is and how it is used to control robots. We have explored the differences between block-based and text-based coding and discovered the advantages of block-based coding for beginners. Learners have also gotten familiar with the LEGO Education SPIKE app, a block-based programming platform that allows them to experiment and create their own robot programmes.

Theme or Focal Area: Creating Computer Programmes from Flowcharts

Introduction

Up to this point, we have established that before programmes are developed to implement robotic solutions, they are well thought out and planned. In the planning process, after understanding the problem to be solved, an algorithm is developed using pseudocodes and later flowcharts. It is these flowcharts that are later implemented into code or programmes. Having been introduced to block-based programming and flowcharts. It is important that we learn to bridge the gap and see how to translate our flowcharts into actual programmes that control robots.

Translating Flowcharts to Blocks:

The flowchart simply represents a map to solve your defined problem and implement the desired task. Each flowchart symbol usually translates into a specific block(s) in your programme. Here is a breakdown of some basic ways to translate the common flowchart symbols when using the LEGO education SPIKE app.

- **Start/End:** These are your programme's "Start" and "Stop" blocks. It is an oval shape in a flowchart. In the LEGO Education SPIKE app, these blocks are found under the Events blocks category**.** Also note that the start blocks are of various types, some of which require additional parameters or information to make room for specific sensor input or data before they start. The start blocks are usually rounded at the top to show that they do not snap to any other block from the top. In other words, they are the first blocks usually used. The stop block, on the other hand, has its base flattened to show that it is usually the last of all blocks; no other blocks snap to its bottom. This is illustrated in Fig. 21.3.

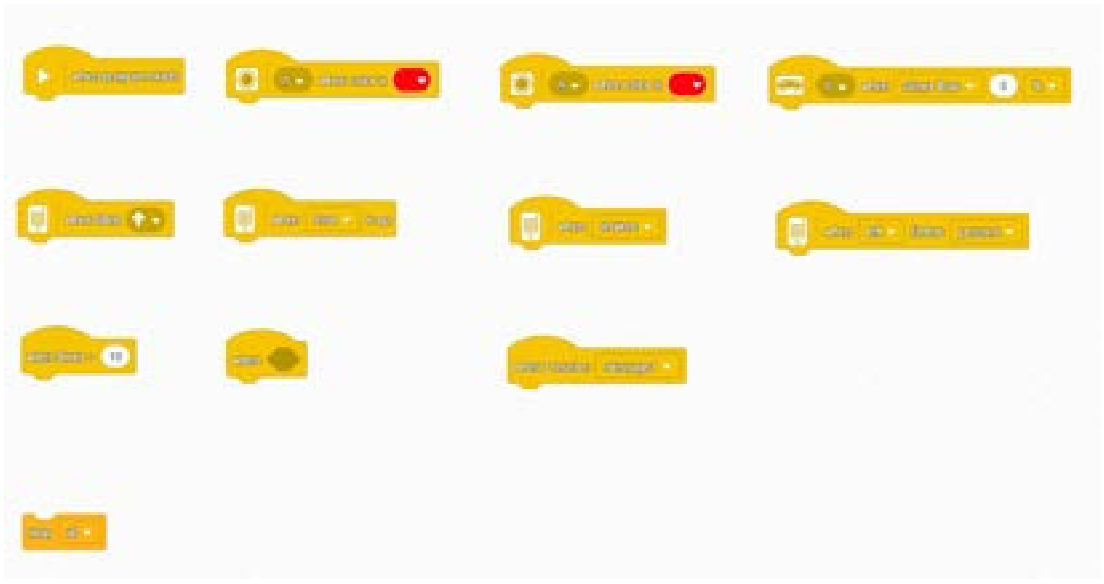


Fig. 21.3: *Start and Stop Blocks in LEGO Education SPIKE app*

- **Process:** These translate to various blocks depending on the action to be carried out. These processes may be found under the motors, movement, light and sound block categories in the LEGO Education SPIKE app. Some examples are presented in Fig. 21.4.

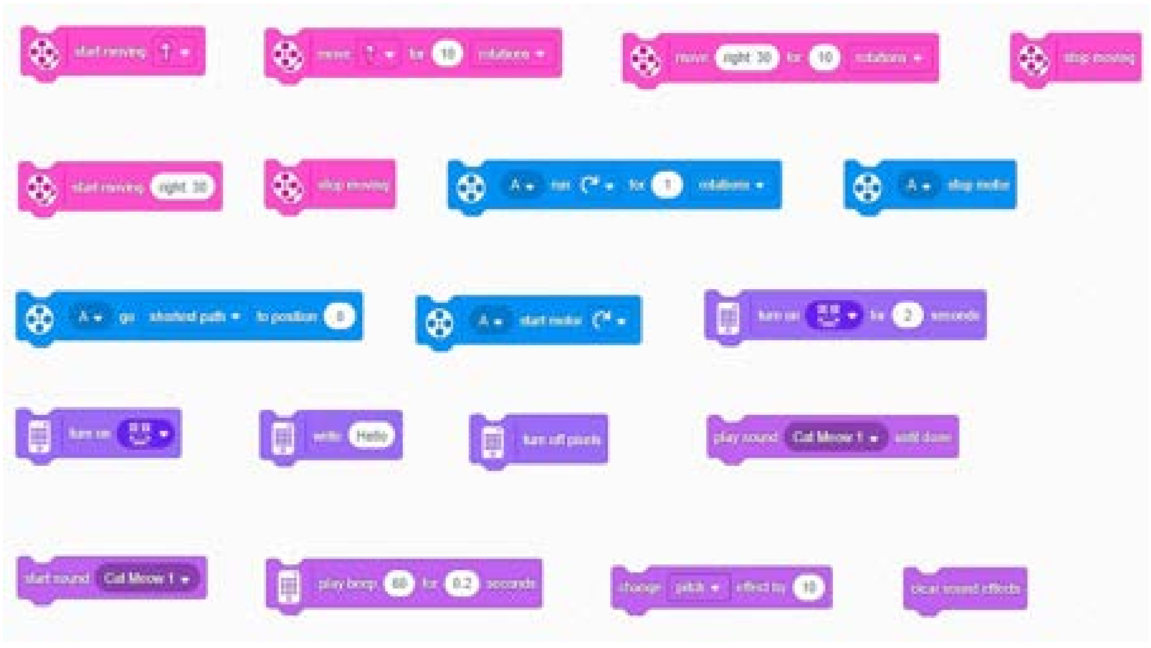


Fig. 21.3: *Examples of Blocks that carry out processes in the LEGO Education SPIKE app*

- **Decision:** Where decisions are to be taken based on prevailing conditions, various control blocks are used. The most typical blocks used to implement these conditions are the **if-then** block, **if-then-else** block, **wait until** block and **repeat until** block. These blocks usually have a hexagon shape, indicating an area where the condition to be sought for is defined.

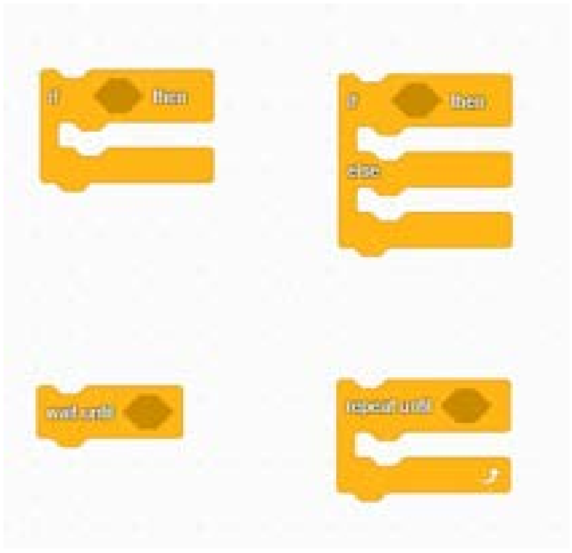


Fig. 21.4: *Examples of Control Blocks in the LEGO Education SPIKE app that are used to implement decisions*

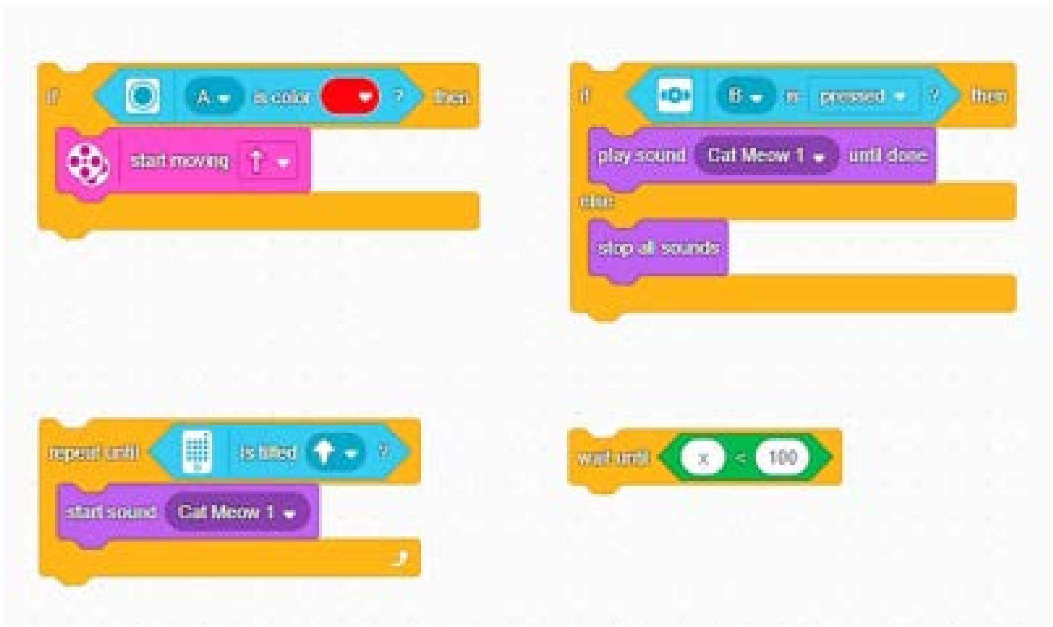


Fig. 21.5: *Examples of Control Blocks with specified conditions used to implement decisions*

- **Connector:** The arrows in a flowchart simply guide you in connecting the correct blocks in your programme. They tell of the logical flow of your programme and the order in which they are to appear.

Mastering Conditions: When to Use Which Block

As mentioned earlier, there are four main blocks that are used to implement conditions in the LEGO Education SPIKE app. Knowing the right block to select when checking for a specific condition or situation is important. The following are some general guidelines that can help in the selection process.

If Then Block: Use this block when the robot needs to perform an action only if a specific condition is true. In this case, we are not concerned about what to do when the condition is not true. Think of it as a "yes or no" question where we are only concerned with providing an action when the answer is yes. If the answer is "yes," the robot performs the action within the block.

Example:

- **Flowchart:** A robot which moves in the forward direction needs to stop only if it detects an object in front of it using an ultrasonic sensor. The object should not be more than 10cm close to it. This is depicted in the flowchart diagram in Fig. 21.6.

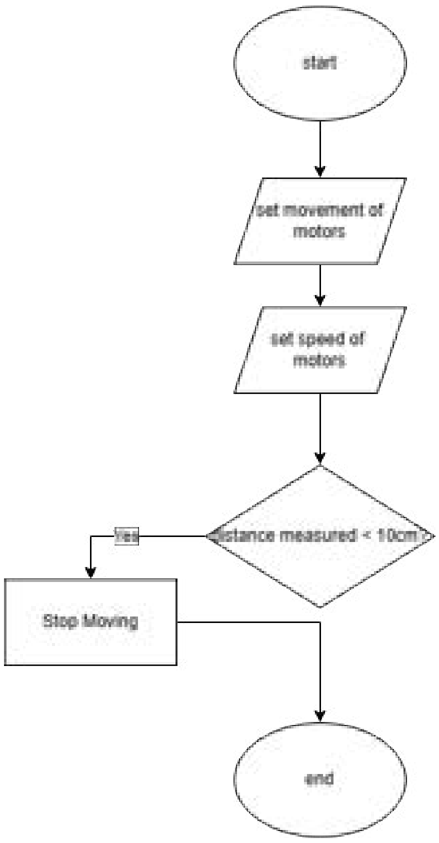
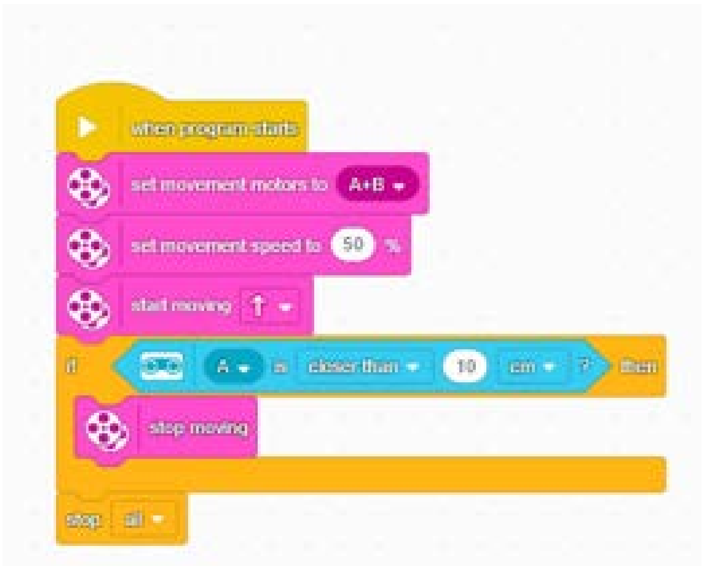


Fig. 21.6: *Flowchart depicting a situation with a condition*

- **Block Program:** An "If Then Block" checks the ultrasonic sensor. If the sensor detects an object in front of it not more than 10cm (true), the robot stops moving.



```
when program starts
set movement motors to A+B
set movement speed to 50 %
start moving ↑
if A is closer than 10 cm ? then
    stop moving
stop all
```

Fig. 21.7: *Block programme which implements the flowchart using an* **if then** *block.*

If Then Else Block: This block allows for two possible actions based on the condition. If the condition is true, the robot performs one set of actions within the "If" section. If the condition is false, the robot performs a different set of actions within the "Else" section.

Example:

- **Flowchart:** A robot installed with a colour sensor is to move forward only when it sees a black colour else it should stop moving.

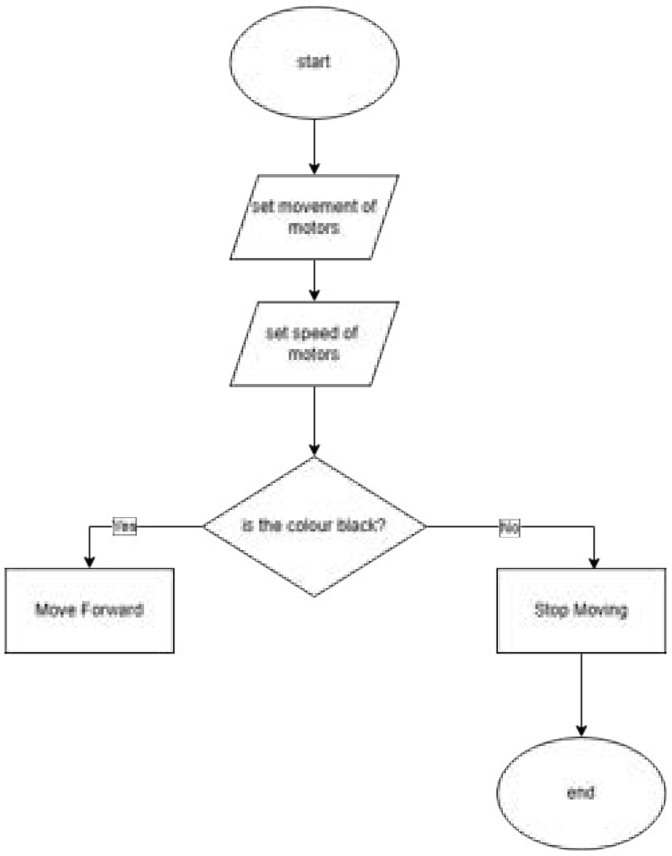


Fig. 21.8: *Flowchart depicting a situation with a condition*

- **Block Programme:** An "If Else Block" checks the line sensor. If the sensor detects black (true), the robot uses a "Turn Left" block. If it doesn't detect black (false), the robot uses a "Move Forward" block.

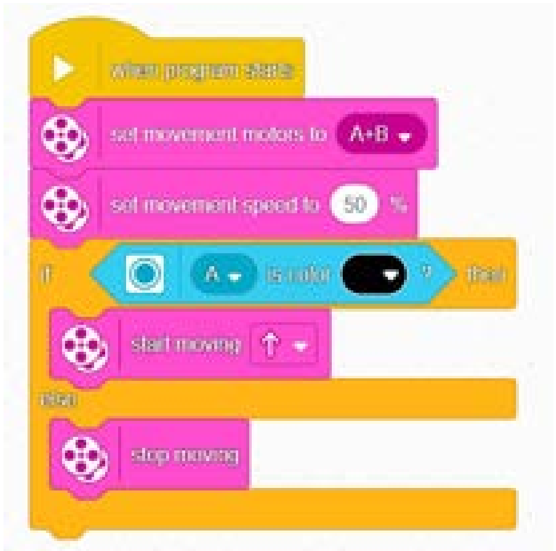

```
when program starts
set movement motors to A+B
set movement speed to 50 %
if A is color ? then
    start moving ↑
else
    stop moving
```

Fig. 21.9: *Block programme which implements the flowchart using an* **if then** *else block.*

Wait Until Block: This block pauses the programme until a specific condition becomes true. This is useful for situations where the robot needs to wait for something to happen before proceeding.

Example:

- **Flowchart:** A robot with a colour sensor must wait until it sees the colour green before moving forward.

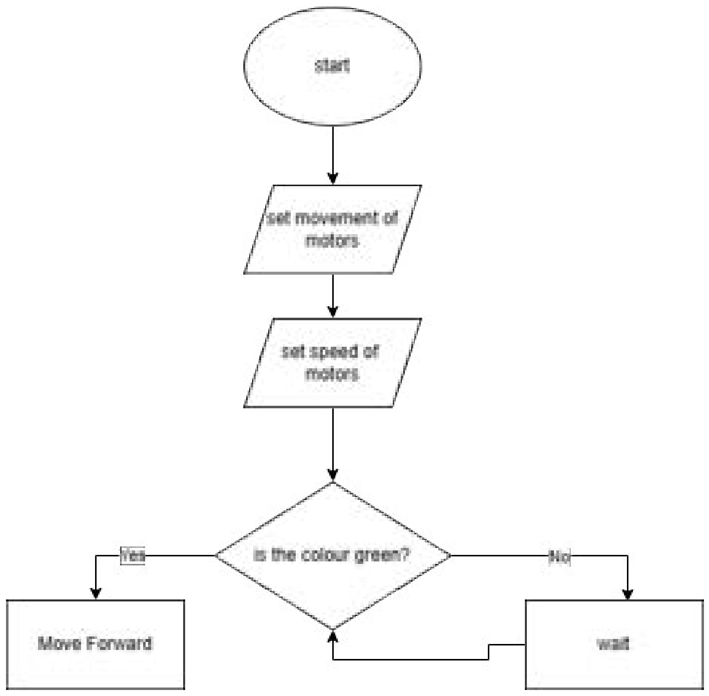


Fig. 21.10: *Flowchart depicting a situation with a condition*

- **Block Programme:** A "Wait Until" block checks the colour sensor. The programme pauses here until the colour sensor detects the colour green (true). Once the colour green is detected, the robot uses a "Move Forward" block.

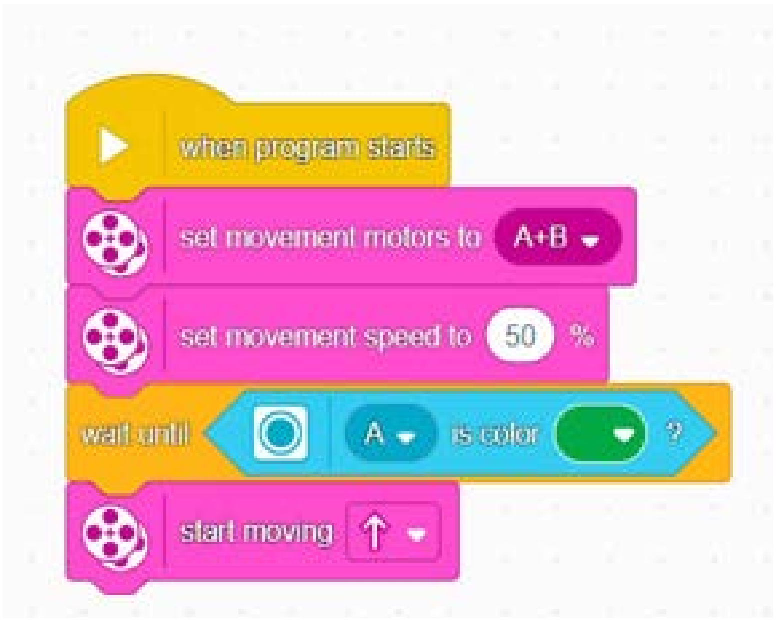


Fig. 21.11: *Block programme which implements the flowchart using a wait-until block.*

Note that the above situation could equally have been implemented using an **if then** block together with a **forever** block as shown below.

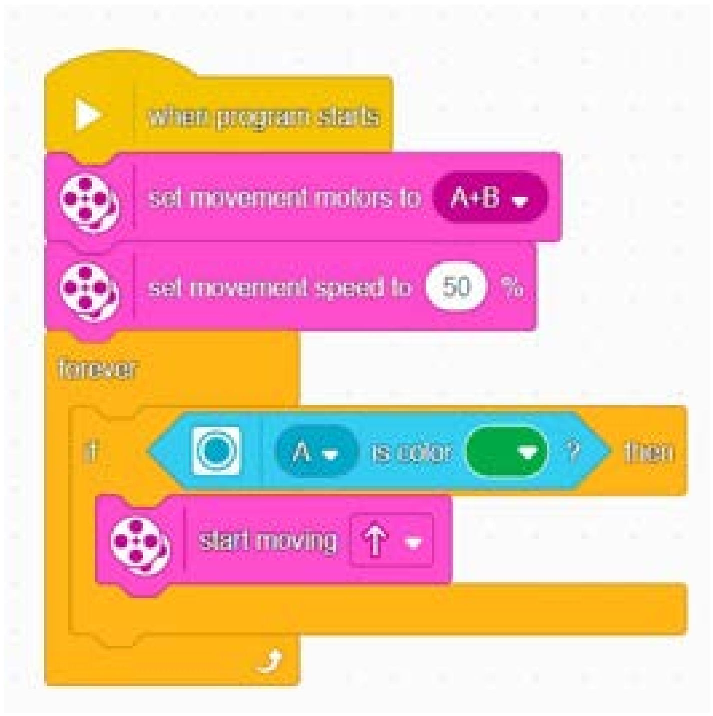


Fig. 21.12: *An equivalent of the* **wait-until** *block programme*

Repeat Until Block: This block repeats a series of actions until a specific condition becomes true. It is like a loop that keeps running until a certain requirement is met.

Example:

- **Flowchart:** A robot needs to keep playing a tune until it sees a green colour.

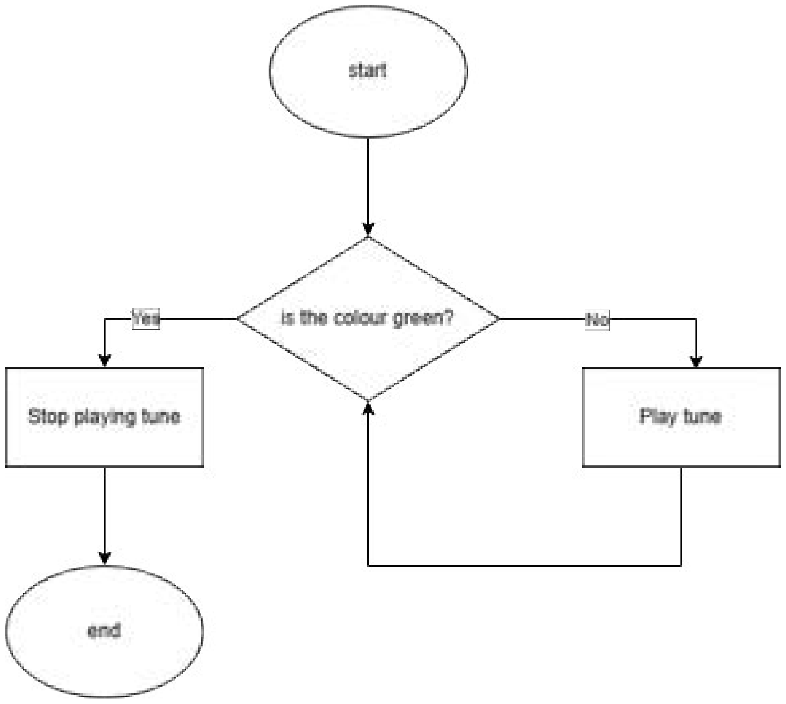


Fig. 21.13: *Flowchart depicting a situation with a condition*

- **Block Programme:** A **repeat until** block contains the actions for playing the tune (meow tune). The loop keeps repeating until the colour sensor detects the colour green (true), at which point the loop stops.

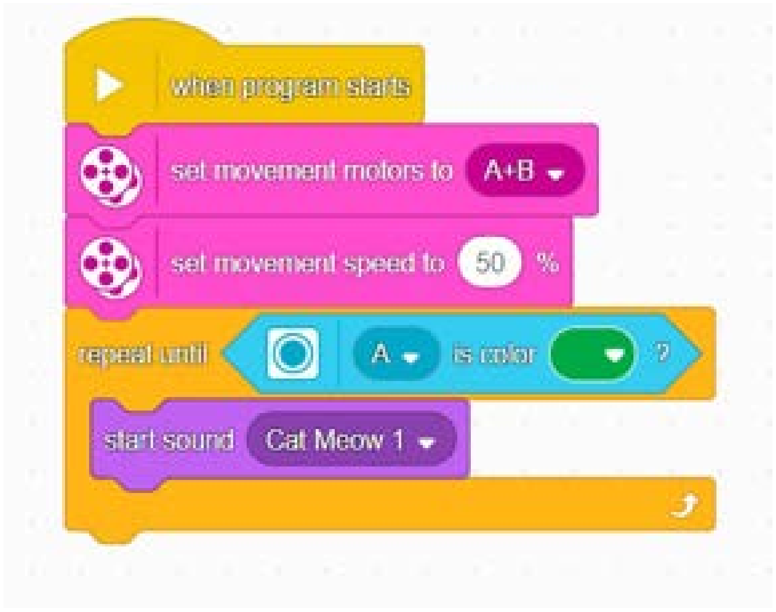

when program starts
set movement motors to A+B
set movement speed to 50 %
repeat until A is color ?
start sound Cat Meow 1

Fig. 21.14: *Block programme which implements the flowchart using a* **repeat-until** *block.*

Mastering Conditions: Nested Statements

You have mastered using "If then", "If then Else," "Wait Until," and "Repeat Until" blocks to create awesome robot programmes. But what if your robot needs to make even more complex decisions? That is where nested blocks come in.

Think of a set of Russian nesting dolls – the bigger doll holds a smaller doll inside. Similarly, nested blocks allow you to place one code block inside another. The outer block controls when the inner block's code executes.

Why Nest Blocks?

Sometimes, a simple "If then" or "If then Else" statement is not enough. Nesting allows for more intricate decision-making within your programme to deal with situations where there are either multi-step decisions or conditional actions with loops. For example, imagine a robot programmed to follow a line. It should normally move forward when it detects the line. But if it is anticipated that this is supposed to be done only for a period of time, say 10 seconds, then the first block (action) should be nested within the loop, which checks the time. So we first check the time before we move. As long as we are within time, the robot can keep following the line.

This example is depicted using the following flowchart and block programmes.

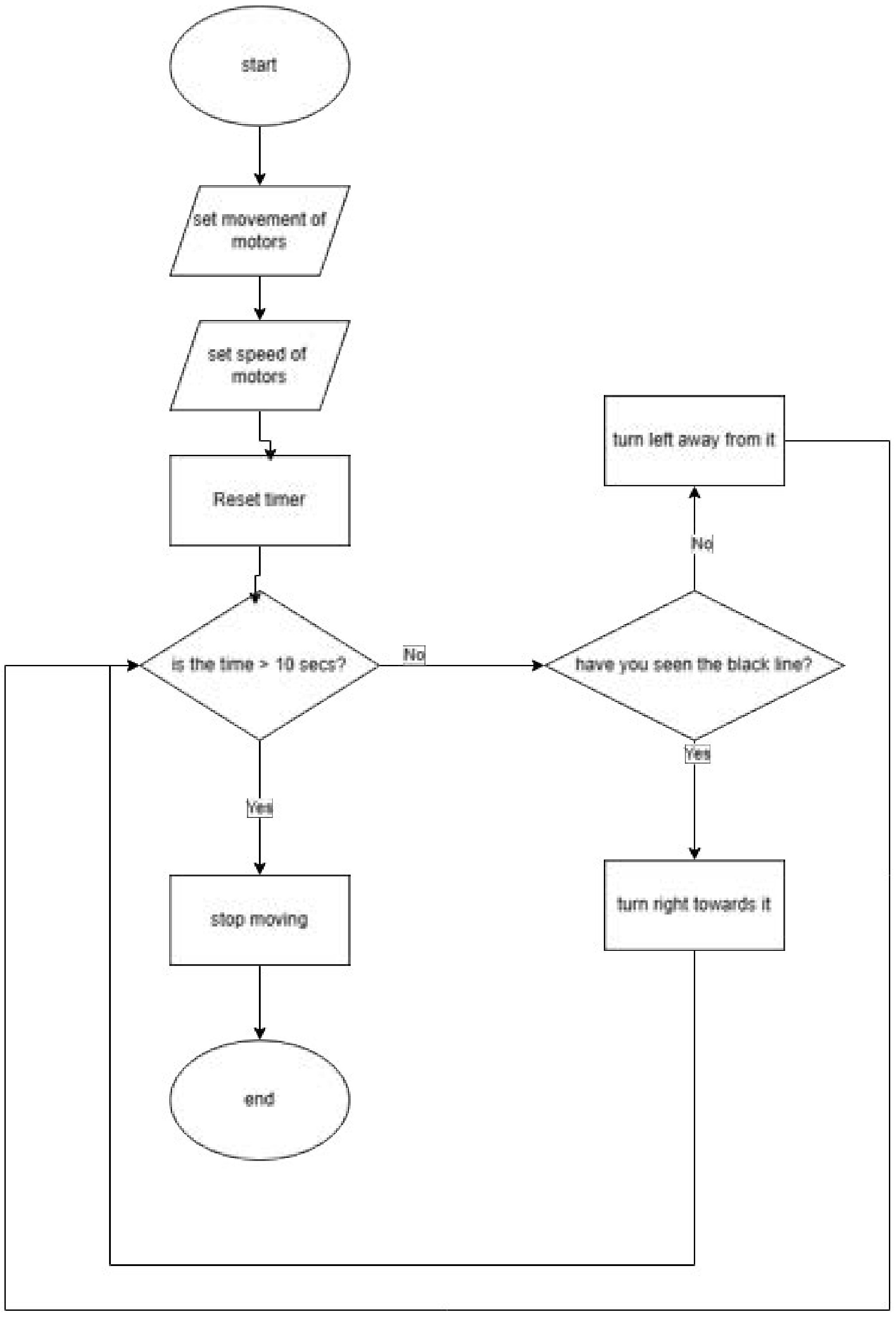
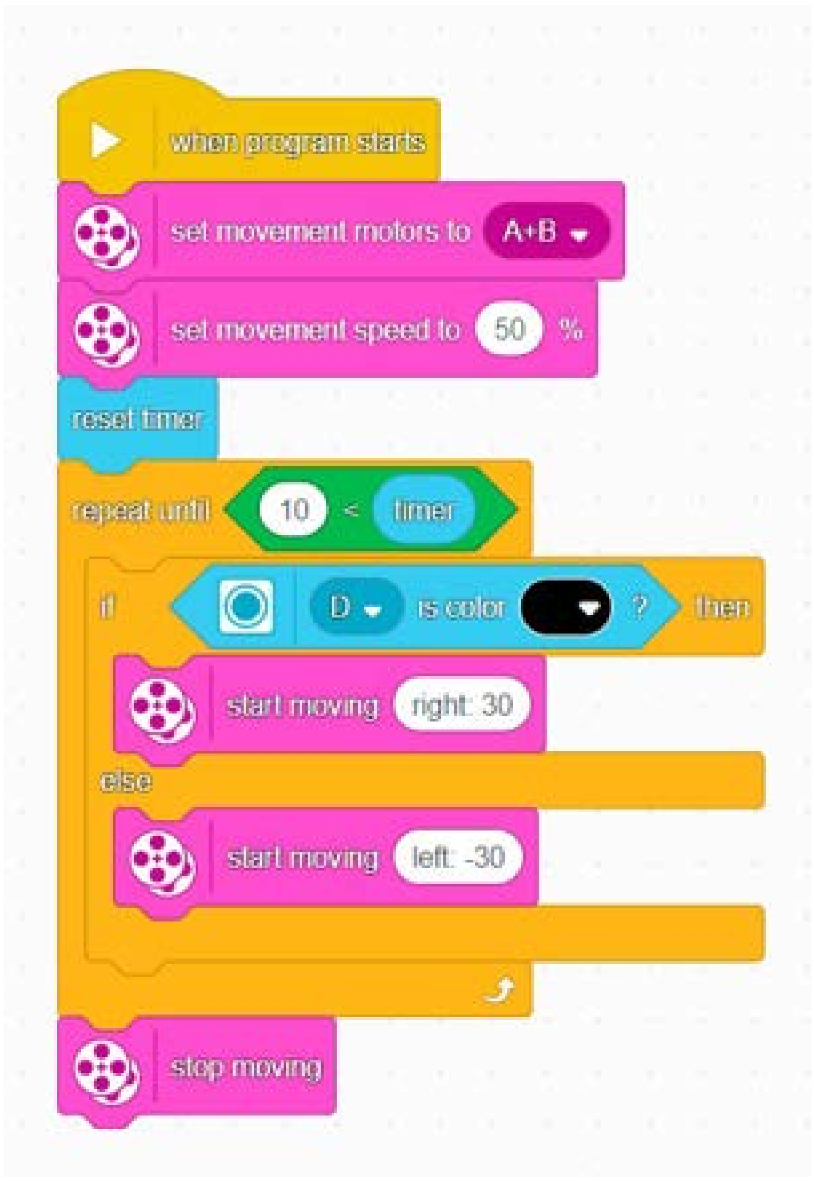


Fig. 21.15: *Flowchart showing a line following robot which uses nested blocks (multiple decisions)*



```
when program starts
set movement motors to A+B
set movement speed to 50 %
reset timer
repeat until 10 < timer
    if D is color ? then
        start moving right: 30
    else
        start moving left: -30
stop moving
```

Fig. 21.16: *Block programme showing a line following robot which uses nested blocks*

Always remember that the inner block's code only executes when the outer block's condition is met. Think of it as a special permission to run the code inside.

Learning Tasks

Based on the content covered under this thematic area, learners will

1. identify the appropriate block type (start/stop, process, decision) based on the flowchart symbol.
2. use various control blocks (if-then, if-then-else, wait until, repeat until) to implement decisions in their code.
3. understand the situations when to use each control block.
4. Apply nested blocks to create more complex decision-making logic in their programmes.

Pedagogical Exemplars

The goal of this lesson is for all learners to translate flowchart diagrams into block-based code. Consider the following keynotes when administering the suggested pedagogical approaches in the curriculum:

1. Recognise and capitalise on the shared characteristics among learners while also addressing their individual differences, including interests, readiness levels and learning styles.
2. Offer multiple pathways for learners to engage with the content. This could involve providing varying levels of detail, from basic concepts to in-depth explorations, to accommodate different learning needs. The key thing is that the learning outcomes set for the lesson are achieved among all learners.
3. **Talk for Learning:** Engage learners in a discussion where they are made to review what they learnt earlier on flowchart diagrams and their role in programme planning. Proceed further to showcase how flowchart diagrams are translated into block-based code.
   a. Using this approach as a starter, through questioning, remind learners by assessing their understanding of the basic symbols of flowchart diagrams, which were covered in previous lessons.
   b. Introduce the idea of translating flowcharts into block-based code.
   c. Use an interactive presentation tool (e.g. Mentimeter, Pear Deck) to introduce common flowchart symbols and their corresponding block-based code equivalents (Start/Stop, Process, Decision).
   d. Facilitate a class discussion where learners participate by matching symbols to blocks and explaining their functions.
4. **Problem-Based Learning:** Learners work in mixed-ability groups to translate flowchart diagrams into block-based code using the LEGO Education Spike App.
   a. Using this approach, consider creating groups with members having a mix of design, robot assembly, programming, communication and presentation skills. This fosters collaboration and uses each learner's strengths.
   b. Provide each group with a set of flowcharts with single-decision conditions (e.g. the robot moves forward until it detects an obstacle). Vary the difficulty of the flowcharts based on the group's composition and strengths.
   c. Learners work together to translate each flowchart into a block-based programme using LEGO® Education SPIKE™ App.
   d. Encourage learners to discuss their thought processes and reasoning behind their block choices.
   e. Each group tests their programme on an already assembled LEGO SPIKE robot. Advanced learners can be allowed to assemble their robots on their own as well as make changes to their robots if need be.
   f. Facilitate a group discussion to address any errors or unexpected behaviours encountered during testing.
   g. Guide learners through debugging their code by identifying logical errors and suggesting corrections.
   h. Provide additional support or scaffolding for learners who may struggle with the task. Provide clarification for learners who may need it. You could describe what the expected programmes could look like.
   i. Provide feedback and reinforcement to reinforce learning and encourage continued engagement.

5. **Project-Based Learning:** Learners work in mixed-ability groups on projects that make use of nested blocks using the LEGO Education Spike App.
   a. Using this approach, start by first reviewing the concepts of translating flowcharts with single-decision conditions into block-based code.
   b. Introduce the concept of nested-decision conditions in flowcharts and their applications.
   c. Show a short video demonstrating how to use nested block structures in LEGO® Education SPIKE™ App (e.g. following a line for a specific duration).
   d. Pause the video at key points to allow learners to ask questions and clarify any doubts.
   e. Divide learners into mixed-ability groups. Consider creating groups with members having a mix of design, robot assembly, programming, communication and presentation skills. This fosters collaboration and uses each learner's strengths.
   f. Provide each group with a flowchart with nested-decision conditions (e.g. a robot follows a line until it reaches a green object, then stops and plays a sound).
   g. Groups discuss and analyse the flowchart, identifying the main decision and any nested decisions within it.
   h. Encourage learners to explain the logic behind each branch of the flowchart.
   i. Learners work within their groups to create a block-based programme in the SPIKE™ App that corresponds to the assigned flowchart.
   j. Circulate among groups, providing support and guidance as needed.
   k. Each group shares their completed block-based programme with the class.
   l. A learner volunteer from each group explains their code structure, focusing on the use of nested blocks and their functionalities.
   m. The class provides constructive feedback and asks clarifying questions.

Key Assessment

Assessment Level 1: Learners are presented with a list of flowchart symbols (start, stop, process, decision) and a list of block types (start block, stop block, action block, control block). They need to match each symbol with its corresponding block type.

Assessment Level 1: Learners are given a flowchart with a single decision point and asked to identify the appropriate control block to implement that decision (if-then, if-then-else).

Assessment Level 2: Learners are provided with a simple flowchart with a single decision and asked to translate it into a block-based programme using a specific app (e.g. LEGO Education SPIKE). They need to use start/stop blocks, action blocks correctly, and the appropriate control block based on the decision point.

Assessment Level 3: Learners are provided with a block-based programme and asked to explain the logic behind the use of specific control blocks (if-then, if-then-else, wait until, repeat until) in the programme. They need to analyse the programme flow and explain how each block contributes to the overall decision-making process.

Assessment Level 4: Learners are challenged to design, build and programme a robot using LEGO Education SPIKE that accomplishes a specific task involving complex decision-making. They need to utilise nested blocks to create the robot's behaviour and present their project, explaining the programme logic and decision-making process.

Conclusion

This lesson taught learners how to translate flowcharts into block-based code for controlling robots. Learners matched flowchart symbols to block types (start/stop, process, decision) and used control blocks (if-then, etc.) to implement decisions in their programmes. The lesson also covered nested blocks for creating more complex decision-making logic.

Week 24

Learning Indicators:

1. *Create computer programmes from pre-designed flowcharts that have a controlled feedback loop with loop interrupts.*
2. *Formulate and programme FSMs to control different use cases.*

Theme or Focal Area: Fundamentals of Control Principles in Automation and Robotics - Feedback and Non-Feedback Loop Systems

Introduction

This lesson revisits and expands upon the concepts of finite state machines (FSMs) and controlled feedback loops, both of which are valuable tools for controlling robot behaviour. Learners will be reminded of the definitions and functionalities of each system before delving deeper into their key differences and practical applications in robotics programming.

Reviewing Key Concepts

A finite state machine (FSM) acts as a structured roadmap for a robot's actions. It defines a set of distinct states, such as "waiting," "moving forward," or "turning," along with the specific events that trigger transitions between these states. A simple example is a traffic light controller. It exists in three distinct states (red, yellow and green) and transitions between them based on a timer (the event). Remember that FSMs have the following key characteristics:

1. **Finite Set of States:** An FSM operates within a defined set of states. These states represent different conditions or stages the system can be in at any given time. Examples include "waiting," "moving forward," "turning left" or "off."
2. **Transitions:** Transitions are well-defined pathways between states. Specific events or conditions trigger them. For instance, a robot in a "waiting" state might transition to a "moving forward" state when it detects a button press (the event).
3. **State-Based Decisions:** The actions taken by the FSM depend on the current state and the triggering event. This means the system's behaviour is determined by its current state and the specific event that occurs.
4. **Deterministic or Non-deterministic:** FSMs can be either deterministic or non-deterministic. In a deterministic FSM, for every state and event, there is only one possible next state, ensuring predictable behaviour. In a non-deterministic FSM, there can be multiple possible next states for a given state-event combination, introducing an element of randomness or flexibility.
5. **Output Generation:** While the primary focus of FSMs is on state transitions, they can also generate outputs in each state. These outputs might involve controlling motors, activating sensors or displaying information.
6. **Simplicity and Readability:** FSMs offer a clear and concise way to represent complex systems. By breaking down the system into states and transitions, it is easier to understand the logic and control flow.
7. **Modular Design:** FSMs can be modular, allowing for the creation of larger systems by combining smaller FSMs. This promotes code reusability and simplifies complex system design.

In contrast, a controlled feedback loop focuses on continuous monitoring and adjustments. It constantly checks a specific value (such as temperature or distance) and adjusts the robot's actions based on that data. Imagine a line-following robot that uses a colour sensor to stay on the line. It continuously checks the sensor reading and makes adjustments if it goes off track. Remember that Controlled feedback loop systems have the following key characteristics:

1. **Continuous Monitoring:** The system constantly tracks its output value. This might involve using sensors to measure temperature, distance, colour or any relevant parameter.
2. **Error Detection:** The monitored output is compared to a predetermined reference value or desired outcome. This comparison helps identify any deviations or errors between the actual and desired states.
3. **Adjustment and Error Correction:** Based on the detected error, the system initiates corrective actions or adjustments. This ensures the output is brought closer to the desired value. Systems that can dynamically adjust their behaviour based on feedback are called self-correcting systems. They continuously learn and adapt to maintain optimal performance.

Key Similarities and Differences:

While both FSMs and controlled feedback loops play a crucial role in robot control, they operate in slightly different ways:

Feature	Finite State Machine (FSM)	Controlled Feedback Loop
Structure	Defined states and transitions	Continuous loop with checks inside
Decision-making	Based on the current state and triggering event	Based on the most recent sensor reading
Examples	Traffic light controller, robot performing a sequence of tasks	Line follower, temperature control system

These differences can be summed up as follows:

- **FSMs** operate within a defined set of states and rely on specific events to trigger transitions between them. The decision on which action to take is based on the current state and the triggering event.
- **Controlled feedback loops** involve a continuous loop where the robot constantly monitors a specific value and adjusts its behaviour based on the most recent reading.

Implementation of a Controlled Feedback Loop: Line Following Robot

A line follower equipped with a colour sensor implements a controlled feedback loop. It takes continuous feedback to determine its motion and direction. The following flowchart diagram and block-based code depict how it works.

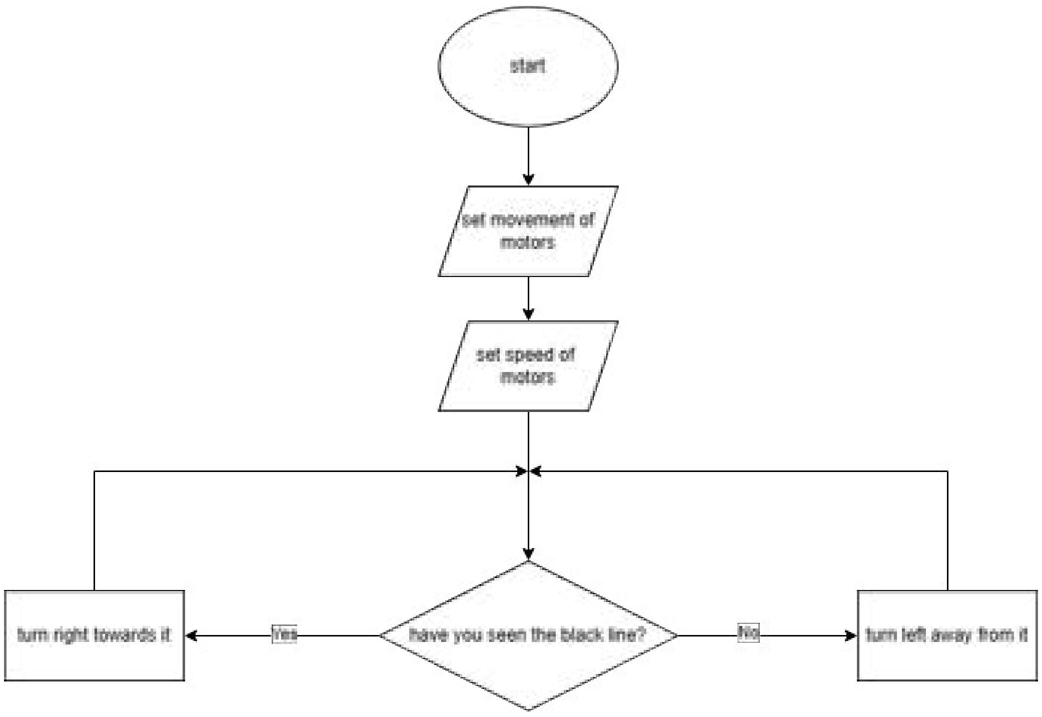
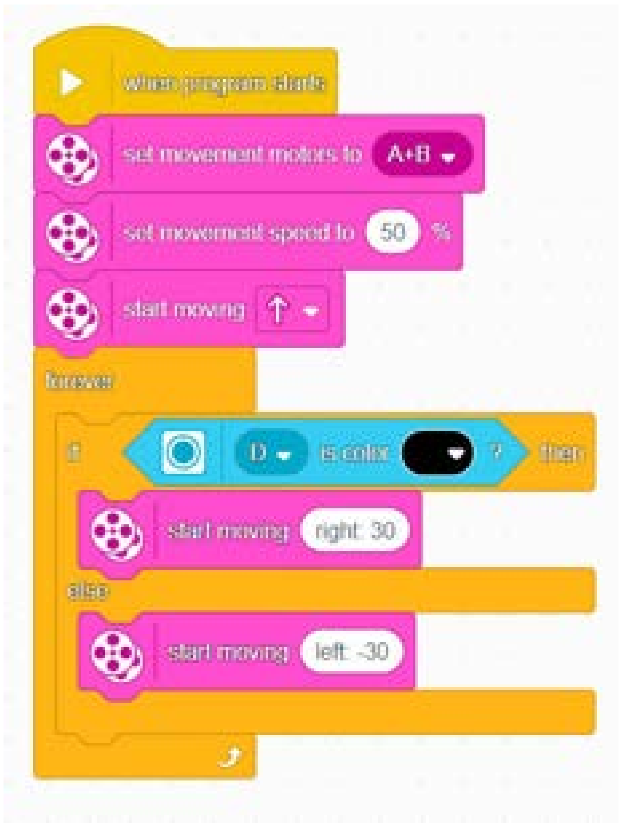


Fig. 24.1: *Flowchart showing a line following robot using continuous feedback*



```
when program starts
set movement motors to A+B
set movement speed to 50 %
start moving ↑
forever
    if D is color ● ? then
        start moving right: 30
    else
        start moving left: -30
```

Fig. 24.2: *Block program showing a line following robot which uses nested blocks*

Implementation of a Finite State Machine: Sorting Robot

Now, let us build a robot tasked with sorting coloured balls into different bins. This robot can be controlled as a finite state machine (FSM) with several states and transitions.

States:

1. **Waiting:** The robot is in its initial position, ready to receive a ball.
2. **Sense Colour:** The robot uses a colour sensor to identify the ball's colour (e.g. red, blue, yellow).

3. **Move to Bin:** Based on the sensed colour, the robot transitions to the appropriate bin state (e.g. Move to Red Bin, Move to Blue Bin, Move to Yellow Bin).
4. **Deposit Ball:** The robot positions itself over the designated bin and releases the ball.

Transitions:

- From "Waiting" to "Sense Colour": This transition occurs when a ball is detected by the robot.
- From "Sense Colour" to one of the "Move to Bin" states: This transition depends on the colour identified by the sensor.
- From "Move to Bin" to "Deposit Ball": This occurs when the robot reaches the designated bin.
- From "Deposit Ball" to "Waiting": This transition ensures the robot is ready for the next ball.

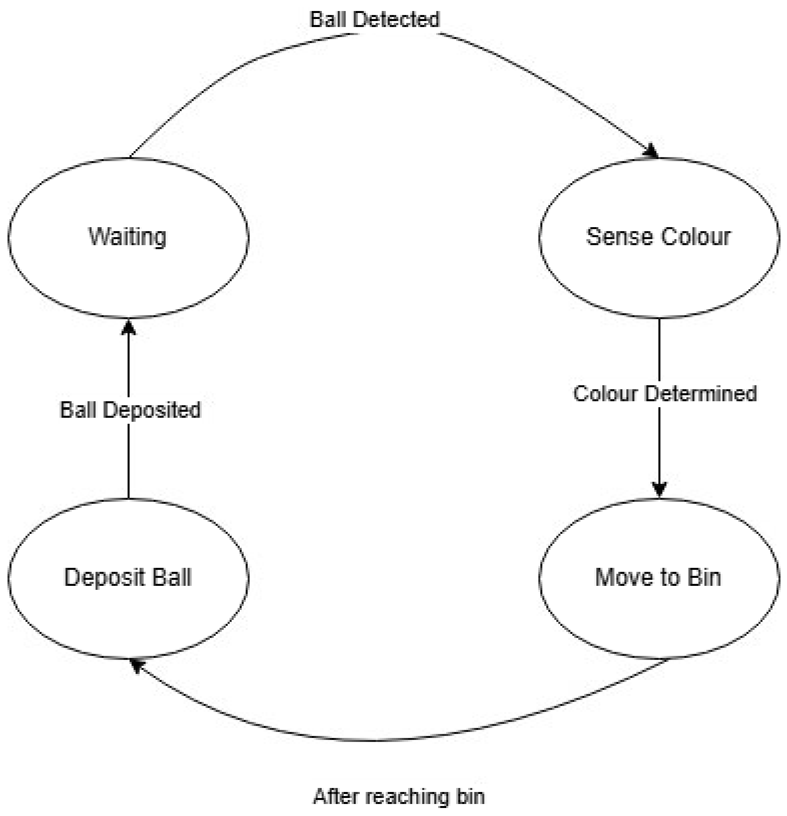
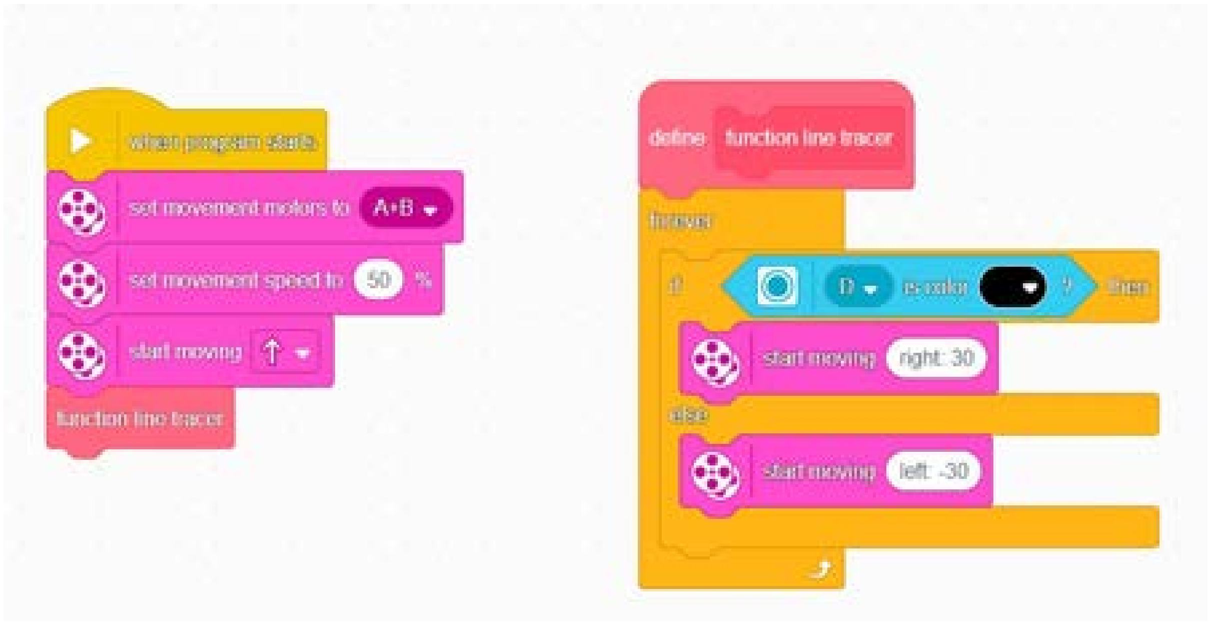


Fig. 24.3: *FSM for ball sorting robot*

This example demonstrates how a finite state machine can effectively control a robot that performs a sequence of tasks based on sensor data and state transitions.

Usually in programming these FSMs, it is advisable to represent these states as functions or routines. A function or routine in block-based programming is simply a number of blocks put together and given a customised name of the user's choice. These blocks are put together because they implement a specific task of functionality (function). So whenever that task is to be performed, that function block is referenced or called by its name using the function reference block. A lot more on functions will be discussed later. A simple demonstration of a function is shown in Fig. 24.4.



```
when program starts
set movement motors to A+B
set movement speed to 50 %
start moving ↑
function line tracer

define function line tracer
forever
    if D is color ? then
        start moving right: 30
    else
        start moving left: -30
```

Fig. 24.4 Using functions in block-based programming

Learning Tasks

Based on the content covered under this thematic area, learners will
1. distinguish between controlled feedback loop systems and finite state machines.
2. create flowchart diagrams for a controlled feedback loop robot
3. develop a block-based programme simulating a robot using a controlled feedback loop
4. create a FSM diagram for a robot which has to navigate its way out of a rectangular maze
5. develop function **blocks to represent states for** the FSM diagram they created

Pedagogical Exemplars

The goal of this lesson is for all learners to distinguish between controlled feedback loop systems and finite state machines and represent them with appropriate diagrams and block code. Consider the following keynotes when administering the suggested pedagogical approaches in the curriculum:

1. Stagger the instruction level of difficulty, starting first from a basic level, through intermediate to an advanced level to cater for the varying needs of learners. You could adopt the following approach:
    a. **Basic**: Provide a simplified explanation of controlled feedback loop and finite state machines, focusing on core concepts and real-world examples with minimal technical jargon.
    b. **Intermediate**: Expand on the basic explanation by introducing the terminology associated with each system. Include additional real-world examples relevant to the learners' context (e.g. automatic temperature control systems, grain sorting machines, etc).
    c. **Advanced**: Delve deeper into the technical aspects of feedback and non-feedback loops. Discuss the detailed characteristics of each system and introduce concepts such as self-correction in feedback loops and conditions for transitioning in FSMs.
2. Ensure that the examples provided at each level are relatable to learners and easy to understand. This will enable them to readily comprehend why these examples are classified as FSMs or controlled feedback loop systems.

3. **Managing Talk for Learning:** In a moderated discussion, guide learners to draw out contrasting differences on these systems, either from personal research findings as well as what they remember from the previous semester and share them with the class for feedback.
    a. Using this approach, focus the discussion on drawing out contrasting differences between the two systems. You may use a Venn diagram and/or a chart to visually represent the similarities and differences in researched differences between the two systems.
    b. Encourage all learner groups to share their thoughts based on their carried-out research and receive constructive feedback. Provide a framework for feedback using phrases such as "I liked how you explained..." or "One way you could improve your presentation is..."
    c. Try to find amicable ways of resolving disagreements in opinions among learner groups.
    d. Summarise all the similarities and differences of these systems and clearly answer any questions that learners may ask.
4. **Problem-Based Learning:** Learners work in mixed-ability groups to identify the necessary states for a robot trying to navigate its way out of a rectangular-shaped maze. They then draw an FSM diagram for this robot, showing the various transitions.
    a. Using this approach, consider creating groups with members having a mix of programming, communication and presentation skills. This fosters collaboration and uses each learner's strengths.
    b. Provide each group with a diagram of the maze, showing the entry and exit points and the labyrinth of paths the robot may have to traverse. Vary the difficulty of the maze based on the group's composition and strengths. Some mazes may be simpler than others.
    c. Explain clearly what the task is and provide them with one or two states which may be necessary (e.g. Start, turn left, turn right, etc.)
    d. Learners work together to identify the other necessary states.
    e. Encourage learners to discuss their thought processes and reasoning behind the states they identify.
    f. Each group draws out their FSM, showing clearly the transitions and tests it with the maze they have been presented with. Encourage learners to implement their states using function blocks in the LEGO Education Spike App.
    g. Facilitate a group discussion to address any challenges encountered during testing.
    h. Provide additional support or scaffolding for learners who may struggle with the task. Provide clarification to learners who may need it. You could describe what the expected FSM could look like.
    i. Provide feedback and reinforcement to reinforce learning and encourage continued engagement.
5. Provide access to diverse resources to cater for the varying preferences of learners. These resources may include videos, images, articles, podcasts, infographics and other multimedia formats.
6. Ensure that all learners have opportunities to access the content in a way that best suits their learning preferences and abilities.

Key Assessment

Assessment Level 1: State any difference between Finite State Machines and Controlled Feedback Systems.

Assessment Level 1: State any similarity between Finite State Machines and Controlled Feedback Systems.

Assessment Level 2: Categorise the following robot descriptions as either Finite State Machines or Controlled Feedback Systems

1. A robot that delivers packages to different drop points.
2. A traffic light controller that cycles through red, yellow and green states.
3. A line-following robot that adjusts its direction based on sensor readings.

Assessment Level 3: Using a simplified rectangular maze with clear entry and exit points, create an FSM diagram for a robot to navigate its way out.

Assessment Level 4: Research and analyse a specific real-world application that utilises a combination of controlled feedback loops and FSMs.

Conclusion

We have explored two powerful tools for robot control: FSMs and controlled feedback loops. We learned about their functionalities, key characteristics and how they differ. By understanding these concepts, learners can effectively design robots that perform complex tasks using sensors and state-based control.

Section 8 Review

In this four-week section, we introduced learners to programming concepts and differentiated between text-based and block-based coding. After being introduced to block-based programming using video tutorials, learners have familiarised themselves with using the LEGO Education Spike App to programme their assembled robots. Learners also learnt how to translate flowcharts into block-based programmes. Finally, learners were exposed to the differences between FSMs and controlled feedback loop systems and how to represent them using state diagrams, flowcharts and block-based programmes.

Additional Reading

Reference Link	QR Code
Block Coding – An A To Z Guide https://www.codingal.com/coding-for-kids/coding-guides/block-coding-guide/	
Block-Based Vs Text-Based Coding For Kids https://moonpreneur.com/blog/block-based-vs-text-based-coding/	
Getting started with LEGO® Education SPIKE™ Prime https://education.lego.com/en-us/start/spike-prime/#Introduction	

Reference Link	QR Code
Spike Prime - Tips and Tricks https://www.youtube.com/playlist?list=PLxFeBwC-gJMSV_EILSS1K6k1dXSYudln9	
Finite State Machines https://www.youtube.com/watch?v=4XEK7OU2gIw&pp=y-gUdZHJhd2luZyBmaW5pdGUgc3RhdGUgZGlhZ3JhbXM%3D	

References

Reference Link	QR Code
SPIKE Prime Tutorials https://www.youtube.com/playlist?list=PL_zXBalpjbu33gw5CML3DtL7fN8640qku	
Lego Spike Prime https://www.youtube.com/playlist?list=PLS9qLR8VoFA62KcAzsUfAOQgLrEXCp78B	

ACKNOWLEDGEMENTS

Special thanks to Professor Edward Appiah, Director-General of the National Council for Curriculum and Assessment (NaCCA) and all who contributed to the successful writing of the Teacher Manuals for the new Senior High School (SHS), Senior High Technical School (SHTS) and Science Technology, Engineering and Mathematics (STEM) curriculum.

The writing team was made up of the following members:

NaCCA Team	
Name of Staff	**Designation**
Matthew Owusu	Deputy Director-General, Technical Services
Reginald Quartey	Ag. Director, Curriculum Development Directorate
Anita Cordei Collison	Ag. Director, Standards, Assessment and Quality Assurance Directorate
Rebecca Abu Gariba	Ag. Director, Corporate Affairs
Anthony Sarpong	Director, Standards, Assessment and Quality Assurance Directorate
Uriah Kofi Otoo	Senior Curriculum Development Officer (Art and Design Foundation & Studio)
Nii Boye Tagoe	Senior Curriculum Development Officer (History)
Juliet Owusu-Ansah	Senior Curriculum Development Officer (Social Studies)
Eric Amoah	Senior Curriculum Development Officer (General Science)
Ayuuba Sullivan Akudago	Senior Curriculum Development Officer (Physical Education & Health)
Godfred Asiedu Mireku	Senior Curriculum Development Officer (Mathematics)
Samuel Owusu Ansah	Senior Curriculum Development Officer (Mathematics)
Thomas Kumah Osei	Senior Curriculum Development Officer (English)
Godwin Mawunyo Kofi Senanu	Assistant Curriculum Development Officer (Economics)
Joachim Kwame Honu	Principal Standards, Assessment and Quality Assurance Officer
Jephtar Adu Mensah	Senior Standards, Assessment and Quality Assurance Officer
Richard Teye	Senior Standards, Assessment and Quality Assurance Officer
Nancy Asieduwaa Gyapong	Assistant Standards, Assessment and Quality Assurance Officer
Francis Agbalenyo	Senior Research, Planning, Monitoring and Evaluation Officer
Abigail Birago Owusu	Senior Research, Planning, Monitoring and Evaluation Officer

NaCCA Team	
Name of Staff	**Designation**
Ebenezer Nkuah Ankamah	Senior Research, Planning, Monitoring and Evaluation Officer
Joseph Barwuah	Senior Instructional Resource Officer
Sharon Antwi-Baah	Assistant Instructional Resource Officer
Dennis Adjasi	Instructional Resource Officer
Samuel Amankwa Ogyampo	Corporate Affairs Officer
Seth Nii Nartey	Corporate Affairs Officer
Alice Abbew Donkor	National Service Person

Subject	**Writer**	**Designation/Institution**
Home Economics	Grace Annagmeng Mwini	Tumu College of Education
	Imoro Miftaw	Gambaga Girls' SHS
	Jusinta Kwakyewaa (Rev. Sr.)	St. Francis SHTS
Religious Studies	Dr. Richardson Addai-Mununkum	University of Education Winneba
	Dr. Francis Opoku	Valley View University College
	Aransa Bawa Abdul Razak	Uthmaniya SHS
	Godfred Bonsu	Prempeh College
RME	Anthony Mensah	Abetifi College of Education
	Joseph Bless Darkwa	Volo Community SHS
	Clement Nsorwineh Atigah	Tamale SHS
Arabic	Dr. Murtada Mahmoud Muaz	AAMUSTED
	Dr. Abas Umar Mohammed	University of Ghana
	Mahey Ibrahim Mohammed	Tijjaniya Senior High School
French	Osmanu Ibrahim	Mount Mary College of Education
	Mawufemor Kwame Agorgli	Akim Asafo SHS
Performing Arts	Dr. Latipher Osei Appiah-Agyei	University of Education Winneba
	Desmond Ali Gasanga	Ghana Education Service
	Chris Ampomah Mensah	Bolgatanga SHS, Winkogo

Subject	Writer	Designation/Institution
Art and Design Studio and Foundation	Dr. Ebenezer Acquah	University for Education Winneba
	Seyram Kojo Adipah	Ghana Education Service
	Dr. Jectey Nyarko Mantey	Kwame Nkrumah University of Science and Technology
	Yaw Boateng Ampadu	Prempeh College
	Kwame Opoku Bonsu	Kwame Nkrumah University of Science and Technology
	Dzorka Etonam Justice	Kpando Senior High Sschool
Applied Technology	Dr. Sherry Kwabla Amedorme	AAMUSTED
	Dr. Prosper Mensah	AAMUSTED
	Esther Pokuah	Mampong Technical College of Education
	Wisdom Dzidzienyo Adzraku	AAMUSTED
	Kunkyuuri Philip	Kumasi SHTS
	Antwi Samuel	Kibi Senior High School
	Josiah Bawagigah Kandwe	Walewale Technical Institute
	Emmanuel Korletey	Benso Senior High Technical School
	Isaac Buckman	Armed Forces Senior High Technical School
	Tetteh Moses	Dagbon State Senior High School
	Awane Adongo Martin	Dabokpa Technical Institute
Design and Communication Technology	Gabriel Boafo	Kwabeng Anglican SHTS
	Henry Agmor Mensah	KASS
	Joseph Asomani	AAMUSTED
	Kwame Opoku Bonsu	Kwame Nkrumah University of Science and Technology
	Dr. Jectey Nyarko Mantey	Kwame Nkrumah University of Science and Technology
	Dr. Ebenezer Acquah	University for Education Winneba
Business Studies	Emmanuel Kodwo Arthur	ICAG
	Dr. Emmanuel Caesar Ayamba	Bolgatanga Technical University
	Ansbert Baba Avole	Bolgatanga Senior High School, Winkogo
	Faustina Graham	Ghana Education Service, HQ
	Nimako Victoria	SDA Senior High School, Akyem Sekyere

Subject	Writer	Designation/Institution
Agriculture	Dr. Esther Fobi Donkoh	University of Energy and Natural Resources
	Prof. Frederick Adzitey	University for Development Studies
	Eric Morgan Asante	St. Peter's Senior High School
Agricultural Science	David Esela Zigah	Achimota School
	Prof. J.V.K. Afun	Kwame Nkrumah University of Science and Technology
	Mrs. Benedicta Carbiliba Foli	Retired, Koforidua Senior High Technical School
Government	Josephine Akosua Gbagbo	Ngleshie Amanfro SHS
	Augustine Arko Blay	University of Education Winneba
	Samuel Kofi Adu	Fettehman Senior High School
Economics	Dr. Peter Anti Partey	University of Cape Coast
	Charlotte Kpogli	Ho Technical University
	Benjamin Agyekum	Mangoase Senior High School
Geography	Raymond Nsiah Asare	Methodist Girls' High School
	Prof. Ebenezer Owusu Sekyere	University for Development Studies
	Samuel Sakyi Addo	Achimota School
History	Kofi Adjei Akrasi	Opoku Ware School
	Dr. Anitha Oforiwah Adu-Boahen	University of Education Winneba
	Prince Essiaw	Enchi College of Education
Ghanaian Language	David Sarpei Nunoo	University of Education Winneba, Ajumako
	Catherine Ekua Mensah	University of Cape Coast
	Ebenezer Agyemang	Opoku Ware School
Physical Education and Health	Paul Dadzie	Accra Academy
	Sekor Gaveh	Kwabeng Anglican Senior High Technical School
	Anthonia Afosah Kwaaso	Junkwa Senior High School
	Mary Aku Ogum	University of Cape Coast
Social Studies	Mohammed Adam	University of Education Winneba
	Simon Tengan	Wa Senior High Technical School
	Jemima Ayensu	Holy Child School

Subject	Writer	Designation/Institution
Computing and Information Communication Technology (ICT)	Victor King Anyanful	OLA College of Education
	Raphael Dordoe Senyo	Ziavi Senior High Technical School
	Kwasi Abankwa Anokye	Ghana Education Service, SEU
	Millicent Heduvor	STEM Senior High School, Awaso
	Dr. Ephriam Kwaa Aidoo	University for Education Winneba
	Dr. Gaddafi Abdul-Salaam	Kwame Nkrumah University of Science and Technology
English Language	Esther O. Armah	Mangoase Senior High School
	Kukua Andoh Robertson	Achimota School
	Alfred Quaittoo	Kaneshie Senior High Technical School
	Benjamin Orrison Akrono	Islamic Girls' Senior High School
	Fuseini Hamza	Tamale Girls' Senior High School
Intervention English	Roberta Emma Amos-Abanyie	Ingit Education Consult
	Perfect Quarshie	Mawuko Girls Senior High School
	Sampson Dedey Baidoo	Benso Senior High Technical School
Literature-in-English	Blessington Dzah	Ziavi Senior High Technical School
	Angela Aninakwah	West African Senior High School
	Juliana Akomea	Mangoase Senior High School
General Science	Dr. Comfort Korkor Sam	University for Development Studies
	Saddik Mohammed	Ghana Education Service
	Robert Arhin	SDA SHS, Akyem Sekyere
Chemistry	Ambrose Ayikue	St. Francis College of Education
	Awumbire Patrick Nsobila	Bolgatanga SHS, Winkogo
	Bismark Tunu	Opoku Ware School
	Gbeddy Nereus Anthony	Ghanata Senior High School
Physics	Dr. Linus Labik	Kwame Nkrumah University of Science and Technology
	Henry Benyah	Wesley Girls High School
	Sylvester Affram	Kwabeng Anglican SHS
Biology	Paul Beeton Damoah	Prempeh College
	Maxwell Bunu	Ada College of Education
	Ebenezer Delali Kpelly	Wesley Girls' SHS
	Doris Osei-Antwi	Ghana National College
Mathematics	Edward Dadson Mills	University of Education Winneba
	Zacharia Abubakari Sadiq	Tamale College of Education
	Collins Kofi Annan	Mando SHS

Subject	Writer	Designation/Institution
Additional Mathematics	Dr. Nana Akosua Owusu-Ansah	University of Education Winneba
	Gershon Mantey	University of Education Winneba
	Innocent Duncan	KNUST SHS
Intervention Mathematics	Florence Yeboah	Assin Manso SHS
	Mawufemor Adukpo	Ghanata SHS
	Jemima Saah	Winneba SHS
Robotics	Dr. Eliel Keelson	Kwame Nkrumah University of Science and Technology
	Dr. Nii Longdon Sowah	University of Ghana
	Isaac Nzoley	Wesley Girls High School
Engineering	Daniel K. Agbogbo	Kwabeng Anglican SHTS
	Prof. Abdul-Rahman Ahmed	Kwame Nkrumah University of Science and Technology
	Valentina Osei-Himah	Atebubu College of Education
Aviation and Aerospace Engineering	Opoku Joel Mintah	Altair Unmanned Technologies
	Sam Ferdinand	Afua Kobi Ampem Girls' SHS
Biomedical Science	Dr. Dorothy Yakoba Agyapong	Kwame Nkrumah University of Science and Technology
	Jennifer Fafa Adzraku	Université Libre de Bruxelles
	Dr. Eric Worlawoe Gaba	Br. Tarcisius Prosthetics and Orthotics Training College
Manufacturing Engineering	Benjamin Atribawuni Asaaga	Kwame Nkrumah University of Science and Technology
	Dr. Samuel Boahene	Kwame Nkrumah University of Science and Technology
	Prof Charles Oppon	Cape Coast Technical University
Spanish	Setor Donne Novieto	University of Ghana
	Franklina Kabio Danlebo	University of Ghana
	Mishael Annoh Acheampong	University of Media, Art and Communication
Assessment	Benjamin Sundeme	St. Ambrose College of Education
	Dr. Isaac Amoako	Atebubu College of Education
Curriculum Writing Guide Technical Team	Paul Michael Cudjoe	Prempeh College
	Evans Odei	Achimota School