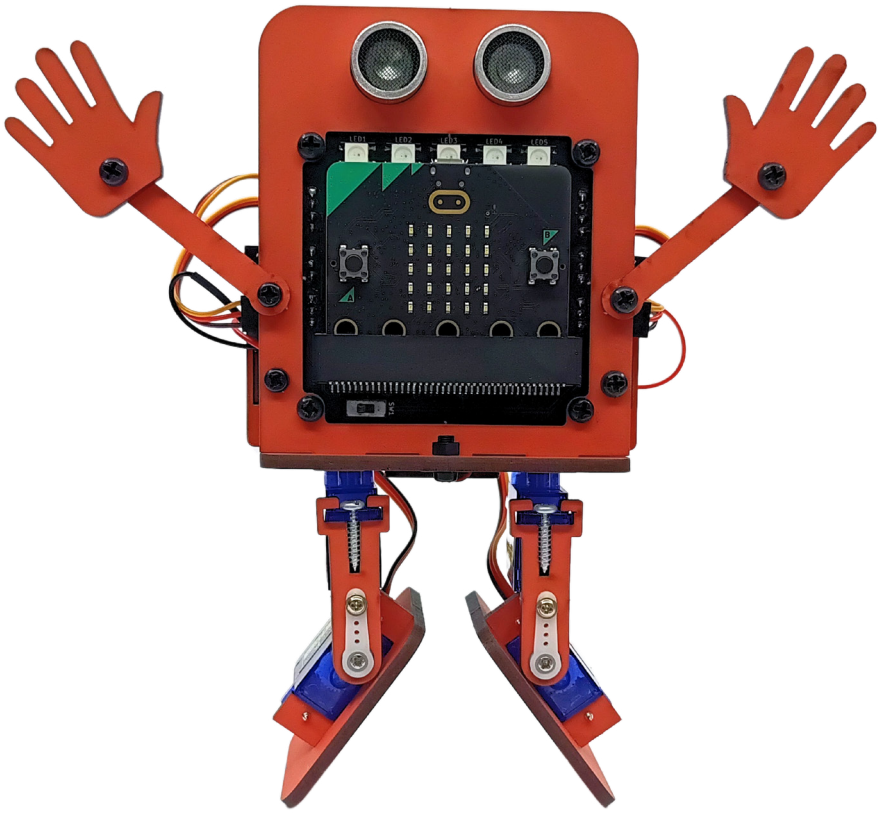
Coding and Robotics

JITTER-BIT



Level 1
Teacher Manual

BOTSHOP
LET THE FUN BEGIN
EST 2014

Copyright:

What We Offer:

Innovative Robots: Dive into the XO-Skeleton series where modular robots come to life with the micro:bit, opening up a world of possibilities.

Comprehensive Training: From student guides to video tutorials, we provide everything you need to turn education into an epic journey.

Sustainability: Our robots are not just smart; they're made from recycled wood, teaching the value of sustainability along the way.

Join the STEM Revolution with us! Whether you're dancing with the Jitter:Bit Dancing Robot, exploring with the Tank-o-Bot, or growing with GrowBotics, you're not just learning coding and engineering—you're becoming an innovator!

Get ready to transform your educational journey from "giving up" to "what do I build next?" with Bot Shop, where learning is not just education; it's an adventure!

Teacher Introduction

1.1 Why Robotics and Coding?

Hello, teacher extraordinaire! Welcome to a world where robots don't just exist in movies, and coding isn't just for computer scientists—it's for you and your students, too! Teaching robotics and coding might seem a bit daunting at first, but trust me, it's like learning to ride a bike. Once you start, it'll become second nature, and the rewards are tremendous.

Why should you teach robotics and coding to kids? Technology touches almost every part of our lives today. From self-driving cars to tiny robots performing surgeries, robotics is reshaping our reality, making things possible that were only a dream a couple of years ago. Coding is the language that powers all these innovations. By introducing your students to these skills early, you're not just teaching them about machines—you're giving them tools to think creatively, solve problems, and build their futures.

Robotics and coding incorporate physics and mathematics. Seeing these two subjects used in real-world applications has tremendous benefits for learners. As they progress through the grades, they will build hydroponic systems that have real-world applications in chemistry and biology, too.

But wait, there's more! Robotics and coding teach critical thinking, logic, and even patience (debugging a robot teaches kids patience in a way few other activities can!). Plus, it's hands-on, so kids aren't just learning—they're building, experimenting, and playing. And let's not forget teamwork. Working together to build a robot or crack a coding challenge encourages collaboration and communication.

1.2 How to Use This Teacher's Guide

This manual will help you every step of the way, breaking down complex concepts into bite-sized, easy-to-teach lessons.

Here's how the guide is structured:

Colour coded chapters: Each chapter in the student and teacher manual is colour coded to make it easy to follow.

Teacher and student manual: Page numbers for each chapter and the questions and answer pages are provided in the teacher manual for easy reference. We've added the answers to the questions on the teacher manual, making it easier for you, the teacher.

Step-by-Step Lessons: Each section builds on the previous section. You will find robotics-specific sections and coding-specific sections. The students will learn about both topics simultaneously, step by step. For example, they first learn about buzzers, then learn how to code a buzzer, and so on.

Activities and Time Allocation: Every lesson includes hands-on activities that your students will love. We've also provided approximate times for each activity so you can plan your classes efficiently.

Bonus activities: The student manual comes with many activities, We will only state each activity here with the estimate time to complete the activity. We however did add additional bonus activities in this teacher manual if there are available time. Teachers can also use many of these activities as homework assignments.

- **Challenges:** The purpose of challenges is to let your students try out other things and not just follow step by step instructions.

- **Teaching Tips:** Throughout the guide, you'll find tips to make teaching robotics and coding easier, whether you're managing a classroom full of excited students or troubleshooting a finicky robot.

- **Troubleshooting:** Remember, troubleshooting is a learning experience all by itself. It makes the students try different things and prepares them for real life, where everything is not working as one thought. With patience and trying different things, you will find a way for things to work as expected, which creates an ever-greater feeling of accomplishment.

- **FAQs:** At the end of each section are common questions and problems you might encounter. These are also great questions for your students to answer.

This guide assumes you know nothing about robotics or coding. It's okay if you've never touched a robot before! You'll learn alongside your students, and by the end, you'll be as comfortable talking about algorithms as you are about your favourite subjects.

1.3 Goals of the Robotics and Coding Curriculum

What do we want your students to achieve by the end of this program?

Goal 1: Understand the Basics of Robotics

By the end of the curriculum, your students should be able to answer questions like:

- What is a robot, and how does it work?
- What are the main components of a robot (e.g., sensors, actuators, processors)?
- Build the Jitter:Bit robot and have the student play with it,

We'll introduce the students to these concepts in a simple and exciting way. They'll learn that robots aren't just cool toys—they can sense, think, and act.

Goal 2: Develop Coding Skills

Coding is the language of robots, and by teaching your students how to code, you're giving them a superpower! By the end of this program, your students will:

- Write their own programs using block-based coding (MakeCode).
- Understand key coding concepts like algorithms, loops, conditionals, and variables.
- Debug simple programs to fix errors.

Goal 3: Build and Program a Robot

You and your students will build an actual robot! They'll assemble a robot from scratch using the Jitter:Bit robot kit. Then, they'll program it to do incredible things like:

- Move forward, backwards, and turn.
- Respond to obstacles using sensors.
- Display animations and make sounds.

This hands-on experience is where the magic happens. There's nothing quite like when a student sees their robot come to life for the first time! They will see firsthand how a microcontroller (the brain) can be placed in something ordinary and make it extraordinary.

Goal 4: Problem-Solving and Creativity

Robotics and coding aren't just about following instructions—they're about thinking critically and creatively. Your students will:

- Solve challenges like programming their robot to navigate a maze.
- Design their own projects, like choreographing a robot dance or inventing a new use for their robot.

These open-ended tasks encourage creativity and help students think like engineers.

Goal 5: Encourage Teamwork and Communication

The students need to suggest things, try things and collaborate. Building and programming robots is a team sport. Throughout the program, your students will work together to brainstorm ideas, divide tasks, and present their projects. They'll learn the importance of communication and respecting each other's contributions.

A Day in the Life of a Robotics Class

Let's paint a picture of what a typical robotics class might look like. Imagine this:

- **10 minutes:** Start with a quick discussion or video about robots in the real world.

- **20 minutes:** Introduce a new concept, like using sensors to detect obstacles. Use simple language and hands-on examples.

- **30 minutes:** Let students dive into an activity, like programming their robot to stop when it gets close to a wall. Walk around the room, offering encouragement and answering questions.

- **10 minutes:** Wrap up with a quick reflection. Ask students what worked, what didn't, and what they're excited to try next time. Also, use the FAQ sections where possible.

- **Activity:** Why Robots Are Awesome

Time: 15 minutes
Objective: Get students excited about robots and coding.
Materials Needed: Access to a video about robots (YouTube has plenty of kid-friendly options).

Instructions:

1. Start by asking students what they think a robot is. Write their answers on the board.
2. Show a short video showcasing different types of robots in action.
3. Discuss what surprised them about the robots? What kinds of tasks would they want a robot to do for them?

FAQ: Welcome to Robotics and Coding

Q: I'm nervous—I've never done this before. Can I really teach robotics and coding?
A: Absolutely! This guide is designed for beginners. You'll learn alongside your students, and there's no pressure to be perfect. Remember, it's okay to say, "I don't know, let's figure it out together." I will not be surprised if you have more fun than the students building these thingamabobs.

Q: What if my students don't seem interested?
A: Start with fun, real-world examples of robots and coding. They'll be hooked once they see how cool it is to make a robot move or light up!

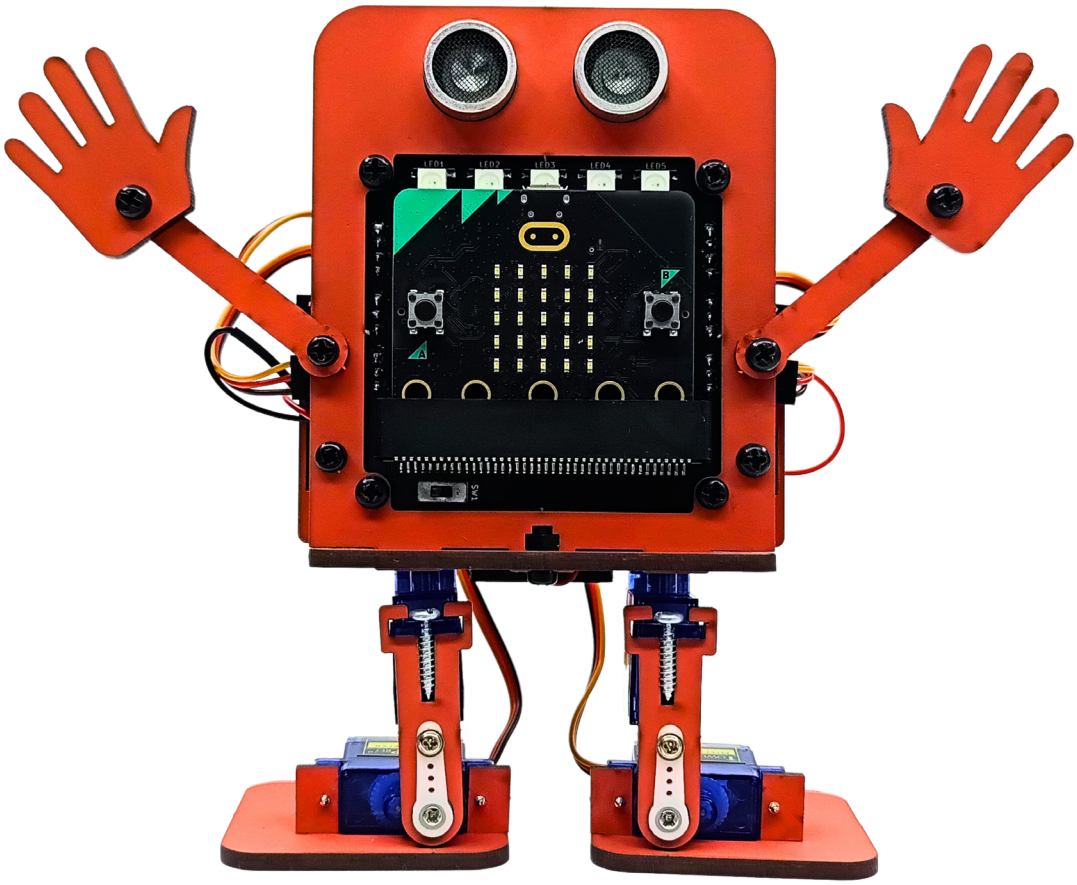
Q: What if I have a tech issue (e.g., the robot won't work)?
A: Don't panic! Check the connections, power, and code. If you're still stuck, consult the troubleshooting section in this guide or ask for help from a tech-savvy colleague.

Teacher Notes:

Student Manual, page 13-20

Section 1:

Introduction to Coding and Robotics



Teacher Tips - "What Are Robotics and Coding?"

1. Start with engagement:

- Begin the lesson with a fun question: "Can you name a robot you've seen in a movie or TV show? What did it do?"
- Show a short video clip or an image of robots in action (e.g., a robot vacuum cleaner or a robotic arm in a factory). This will grab the student's attention and set the tone for learning.

2. Use analogies to simplify concepts:

- Relate robots to everyday items: "A robot is like a smart toy or a tool that can follow instructions."
- Compare coding to a language: "Coding is like writing instructions for a robot, just like writing a to-do list for a friend."
- Explain algorithms with relatable examples like making a sandwich or brushing your teeth.

3. Break down the outcomes into bite-sized discussions:

For each key outcome, guide the discussion with open-ended questions:

- ***What is a robot?*** Ask students to describe a robot they've seen and identify its "brain," sensors, or movement parts.
- ***What is coding?*** Let students give examples of commands they'd give a robot.
- ***What is an algorithm?*** Use a hands-on demo (e.g., make a sandwich before the class) and intentionally mix up the steps to highlight why order matters.

4. Make it interactive:

- ***"Let's Pretend" Activity:*** Turn this into a group exercise where students act as "robots" and others give instructions (like an algorithm).
- Use props like a ball, toy box, or paper cut-outs to simulate robot actions.

5. Reinforce key concepts visually:

- Create a diagram on the board showing the main parts of a robot (brain, sensors, motors, wheels/legs).
- Use flowcharts or simple drawings to show an algorithm (e.g., move forward →, see the toy →, stop →, pick it up).

6. Encourage creativity:

- When asking students to draw a robot, give them examples of how different robots look (a robotic car, arm, or humanoid). Emphasize functionality over artistic skills.
- For algorithms, let them develop unique ideas for tasks (e.g., watering a plant, serving food, or playing music).

7. Relate it to real life:

Discuss how robots and coding impact their daily lives, such as:

- Robots helping in hospitals or factories.
- Apps and gadgets they use at home (e.g., Alexa, automated lights).

8. Assess understanding with simple Q&A:

Ask questions to summarize key concepts:

- "Why is it important to follow the steps of an algorithm in the right order?"
- "What might happen if a robot skipped a step in its instructions?"

9. Emphasize problem-solving:

Remind students that coding and creating algorithms is about solving problems step by step. It's okay to make mistakes as long as they learn to debug and try again.

10. Provide constructive feedback:

When students create their algorithms or draw robots, focus on encouraging creativity and clear logical thinking. Highlight effort and improvement rather than perfection.
These tips ensure the chapter is engaging, interactive, and effective in introducing robotics and coding to students!

Student manual, page 20

Section 1: Questions and Answers

1. **What is a robot?**
   Answer: A robot is a special machine that can follow instructions to do different tasks, such as moving around, picking things up, or helping people. It is made of parts like a brain (computer), sensors, motors, and wheels or legs.

2. **What is coding?**
   Answer: Coding is how we talk to robots and computers using a special language called "code." When you code, you write instructions for the robot to follow.

3. **Why are robotics and coding important?**
   Answer: Robotics and coding are important because they help us in our daily lives. They help us build cars and houses, clean up messes, explore new places, and make life easier and more fun. Learning robotics and coding helps develop skills for the future.

4. **What is an algorithm?**
   Answer: An algorithm is a list of steps to solve a problem or do something. It is like a recipe that tells you what to do step by step.

5. **Why is the order of steps important in an algorithm?**
   Answer: The order of steps is important because the algorithm needs to work correctly if some steps are done before others. For example, you need to put on your socks before your shoes.

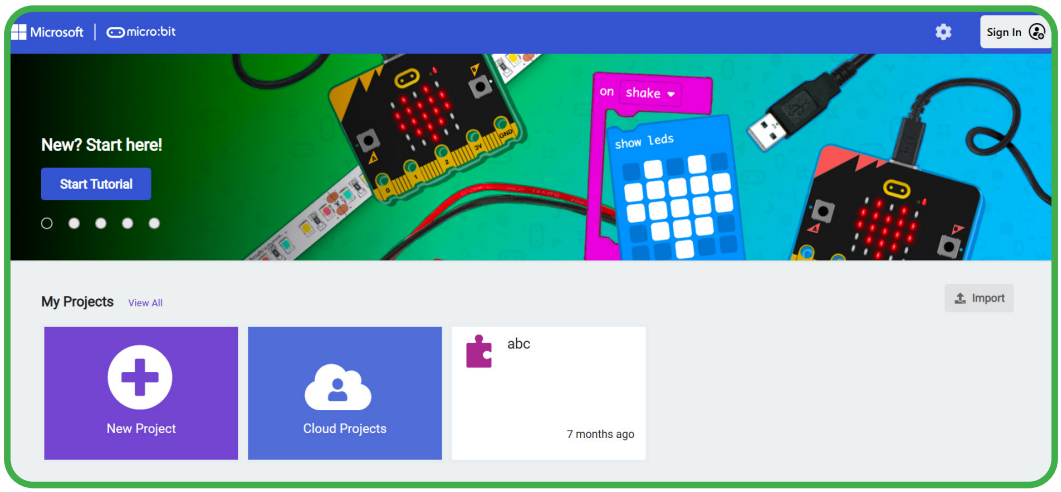
6. **Give an example of an algorithm for a robot.**
   Answer: An example could be: 1. Move forward until you see the ball. 2. Stop when you reach the ball. 3. Open your claw (or hand). 4. Grab the ball. 5. Lift the ball.

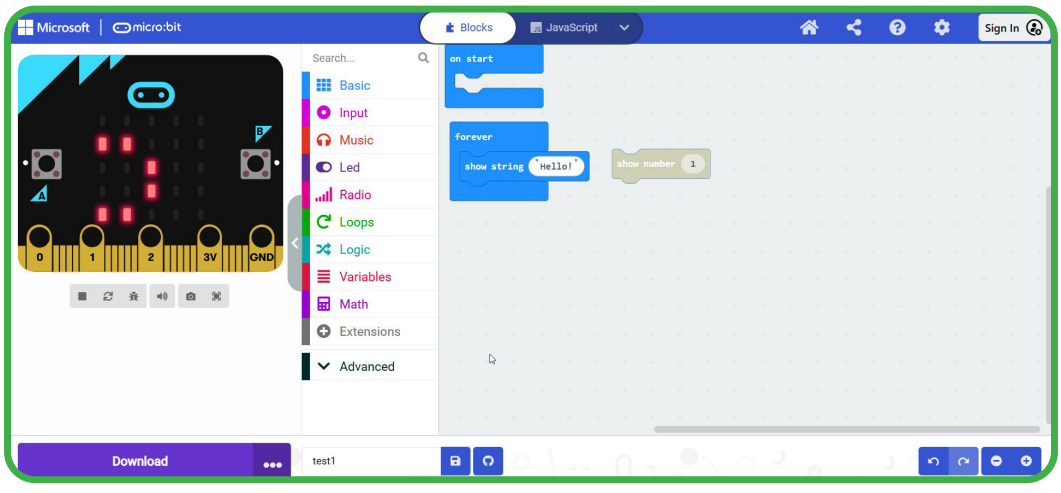
Teacher Notes:

Student Manual, page 21-27

Section 2:

Block Based Programming using the micro:bit (Robot Brain)





Teachers note - Coding.

Real-Life Examples of Coding.

- Have you ever wondered how your phone knows when to unlock? That's coding.
- Or how a robot vacuum avoids bumping into walls? Also coding.
- Even video games and animated movies are powered by code.

Why Teach Kids to Code?

- Coding encourages problem-solving and logical thinking.
- It's a gateway to understanding how technology works.
- It's fun and creative—students can see their ideas come to life!

Bonus Activity: Writing an Algorithm

Time: 20 minutes
Objective: Help students understand algorithms through a relatable task.
Instructions:

1. Ask students to write an algorithm for making a peanut butter and jelly sandwich.
2. Example:
   Step 1: Get two slices of bread.
   Step 2: Spread peanut butter on one slice.
   Step 3: Spread jelly on the other slice.
   Step 4: Put the slices together.
   Step 5: Eat the sandwich!
3. Discuss how robots follow instructions literally. For example, the robot might get confused if the algorithm says, "Spread jelly," but doesn't mention using a knife.

Teaching Tip:

Please encourage students to test their algorithms by acting them out or having a friend follow their steps. It's fun to see if their instructions are clear and logical.

Bonus Activity: Exploring MakeCode
Time: 10 minutes
Objective: Familiarize students with the MakeCode interface.
Instructions:

1. Open MakeCode and start a new project.
2. Ask students to explore the toolbox and find blocks they think are interesting. 3. Let them drag blocks into the workspace and watch the simulator to see what happens.

Teaching Tip:

Encourage students to experiment without fear. Remind them: "You can't break anything with code—every mistake is a chance to learn!"

Debugging: What to Do When Things Go Wrong

Mistakes are part of the coding journey. Debugging is the process of finding and fixing errors in code.

Common Debugging Scenarios

- The LED matrix doesn't light up.
    - Check if the Micro:Bit is properly connected and powered.
    - Ensure the code was downloaded correctly.
- The wrong message appears.
    - Double-check the text in your "show string" block.

Teaching Tip:

Celebrate mistakes! Remind students that every coder, even professionals, makes errors. Debugging is how we learn and grow.

FAQ: First Steps in Coding

Q: What if my students don't understand coding right away?
A: That's okay! Start slow and use real-world analogies to explain concepts. Coding takes practice, and the goal is to make it fun and approachable.

Q: Can we skip the MakeCode simulator and go straight to the Micro:Bit?
A: The simulator is a great way to test code quickly, but if students are eager to see their programs on the Micro:Bit, let them dive in!

Q: How can I keep fast learners engaged?
A: Offer bonus challenges, like creating a short animation or using multiple "show string" blocks to display a conversation.

Congratulations—you've taken your first steps into the world of coding! In the next section, "Building Blocks of Coding," we'll explore important coding concepts like inputs, outputs, variables, and loops. Let me know when you're ready for the next section!

Student manual, page 27

Section 2: Questions and Answers

1. **What is block-based programming?**
   Answer: Block-based programming is writing code using blocks that snap together like puzzle pieces. Each block represents a step the robot can take.

2. **What software will we use for block-based programming with the Micro:bit?**
   Answer: We will be using MakeCode software.

3. **What is a project in the MakeCode software?**
   Answer: Each time you code something new, it is called a project.

4. **What is the purpose of the simulator area in the MakeCode platform?**
   Answer: The simulator area shows you what your code will do on a computer without uploading the code to an actual microcontroller.

5. **Where do you find the different coding blocks in MakeCode?**
   Answer: You find the coding blocks in the toolbox area.

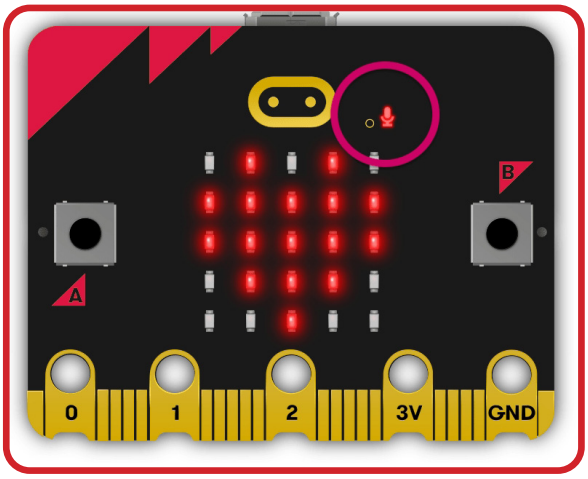
6. **How do you download your code to the Micro:bit?**
   Answer: After you make your code and are happy with the simulator result, you connect the micro:bit to your computer with a USB cable and click the blue download button.

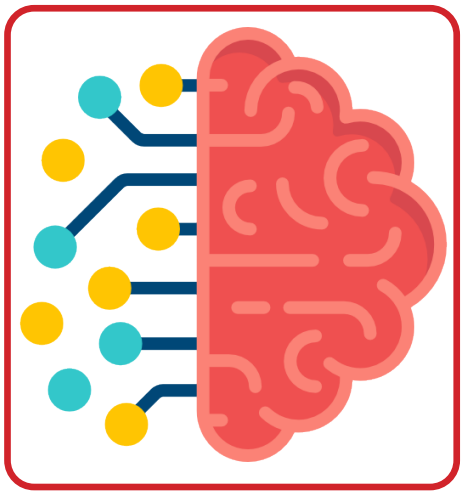
Teacher Notes:

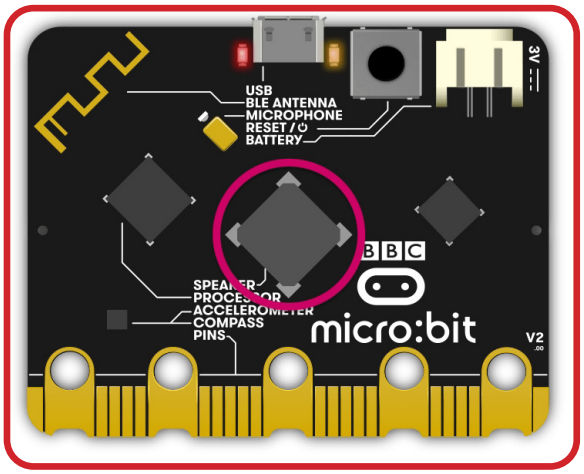
Student Manual, page 29-35

Section 3.

The micro:bit microcontroller (The robot brain)







Teacher Tips: The Micro:Bit

Activity 1: The LED Matrix Adventure

Time: 15 minutes
Objective: Get students comfortable with programming the LED matrix.
Instructions:

1. Open the MakeCode platform on your computer.
2. Create a new project and drag the "show LEDs" block onto the workspace into the "forever" block.
3. In the block, select a pattern for the LEDs to light up (e.g., a heart, smiley face, or star).
4. Click the download button and save the program to your micro:bit. Watch as your design lights up the LED matrix!

Bonus Activity 2: Button Magic

Time: 20 minutes
Objective: Teach students to use buttons A and B to control the micro:bit.
Instructions:

1. In MakeCode, drag a "on button A pressed" block to the workspace.
2. Inside this block, add a "show string" block and type a message (e.g., "Hello!").
3. Add a similar block for button B with a different message (e.g., "Bye!").
4. Download the code and press the buttons to see the messages appear on the LED matrix.

Bonus Activity 3: Shake It Up!

Time: 20 minutes
Objective: Introduce students to the accelerometer.

Instructions:

1. Drag the "on shake" block into the workspace in Make-Code.
2. Inside the block, add a "show icon" block inside the block and select a fun icon (e.g., a ghost or diamond).
3. Download the code and shake the micro:bit to see the icon appear. Students will love discovering how movement triggers an event!

Common Troubleshooting Tips

Even superheroes have their off days, and so will your micro:bit. Here are some common problems and how to fix them:

- **Problem:** The micro:bit isn't lighting up.
  Solution: Check the USB cable and ensure it's plugged in securely. Try another USB port or cable.
- **Problem:** The program isn't working.
  Solution: Double-check the code for errors and ensure you downloaded it to the micro:bit correctly.

FAQs: Meet Your micro:bit

Q: What if I don't have enough micro:bits for every student?
A: No problem! Divide the class into small groups and let them take turns. Collaboration is a great way to encourage teamwork.

Q: Can I use the micro:bit without a computer?
A: Yes, but you'll need to load programs onto it beforehand. You can also connect it to tablets or smartphones via Bluetooth.

Q: My students want to do more advanced things with the micro:bit. How do I help them?
A: Great! Check out additional resources like MakeCode tutorials on the MakeCode website or explore extensions like sound modules and sensors to expand their projects.

In the next section, "First Steps in Coding," we'll explore programming concepts further and get your students coding like pros.

Student manual, page 35

Section 3: Questions and Answers

1. **What is a microcontroller?**
   Answer: A microcontroller stores the code you wrote, does calculations, sends power to connected devices, reads data from sensors, sends data to speakers, and stores data in memory.

2. **What is a sensor?**
   Answer: A sensor is an electronic component that measures weight, temperature, distance, and movement.

3. **What is the difference between input and output for a microcontroller?**
   Answer: Input is how the microcontroller "listens" and gets information from buttons and sensors. Output is how the microcontroller "talks" by turning on lights, playing sounds, or moving motors.

4. **Name three inputs that the micro:bit has.**
   Answer: The micro:bit has buttons, a microphone, and sensors like an accelerometer, touch sensor, microphone and magnetic field sensor.

5. **Name three outputs that the micro:bit has.**
   Answer:The micro:bit has an LED matrix, a speaker, and pins for connecting to external devices.

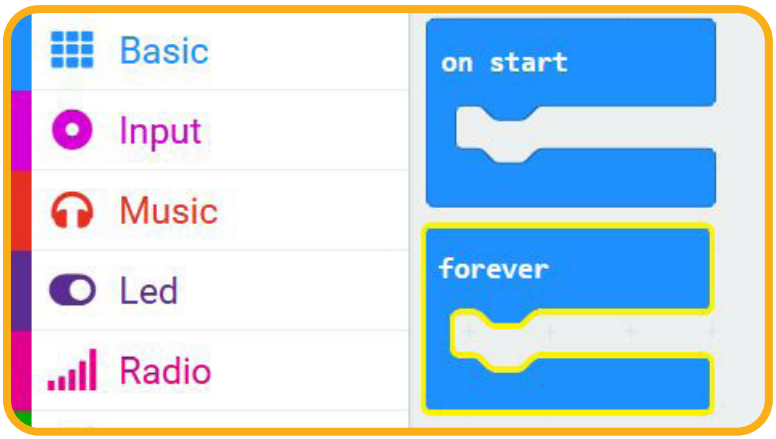
6. **What are two ways to power the micro:bit?**
   Answer: The micro:bit can be powered with a USB cable from your computer or with batteries.

Teacher Notes:

Student Manual, page 37-43

Section 4:

Use Variables and the Forever Block



Teacher tips - Variables

Why Are Variables Useful?

- They make programs more flexible and efficient.
- Robots can "remember" things, like how many steps they've taken or the last button pressed.

Teaching Tip:

Explain how loops save time and make programs efficient. Without loops, we'd need to write "show icon" and "clear screen" over and over again.

Bringing It All Together

Once students are comfortable with inputs, outputs, variables, conditionals, and loops, challenge them to combine these concepts into a single program. For example:

- Use Button A to increase a counter.
- Use Button B to reset the counter.
- Display the counter value on the LED matrix in a loop.

FAQ: Building Blocks of Coding

Q: My students are struggling to understand variables. How can I help?
A: Use real-life examples. For instance, explain that a variable is like a locker where you store your books (data) and can retrieve them later.

Q: Can I skip loops and come back to them later?
A: Loops are fundamental but can be revisited as students become more comfortable with coding. Start simple and gradually build complexity.

Q: How do I keep students engaged during these foundational lessons?
A: Encourage experimentation. Let students modify the examples (e.g., change icons, adjust loop timing) to see immediate results on the micro:bit.

Now that your students understand the basics of a microcontroller and the building blocks of coding, they are ready for the next chapter.

Student manual, page 43

Section 4: Questions and Answers

1. **What is a variable in coding?**
   Answer: A variable is like a data storage space in your robot's mind or your computer where you can keep different information or special items, which can be changed and used repeatedly.

2. **What are the "on start" and "forever" blocks?**
   Answer: The "on start" block contains code that runs once when the microcontroller powers up, while the "forever" block contains code that repeats until the microcontroller is switched off.

3. **Why are variables useful?**
   Answer: Variables help keep track of things, shorten code, and allow you to experiment by changing the values they store.

4. **What are some rules for naming variables?**
   Answer: Variable names should not contain spaces or special characters, except for underscores, and if using two words, the second word starts with a capital letter or use and underscore between them.

5. **How do you change the value of a variable in block-based coding?**
   Answer: You use a "set" block to put an initial value in a variable and a "change" block to add or subtract numbers from the variable.

6. **What is the difference between number and text variable types?**
   Answer: A number variable stores numbers for use in math, and a text variable stores strings of characters.

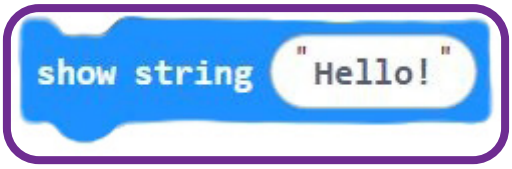
Teacher Notes:

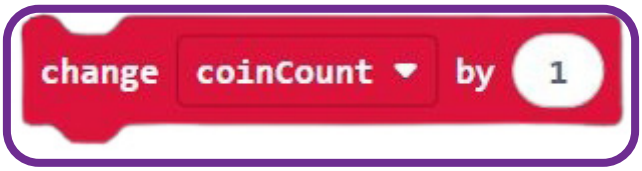
Teacher Notes:

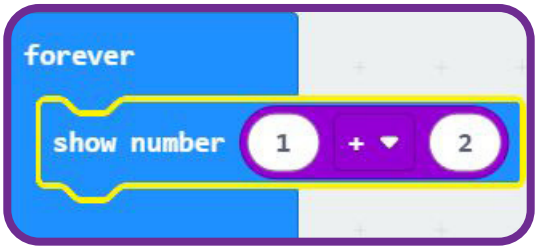
Student Manual, page 45-50

Session 5:

Creating and Working with Text Variables







Teacher Tips for the Chapter: "Creating and Working with Text Variables"

1. Simplify the outcomes with relatable examples:

Before diving into technical details, explain the chapter outcomes in simple terms:

- **Strings:** "Text in coding, like names, words, or sentences, is called strings. Think of it like typing words on a phone."
- **Text variables:** "A variable is like a container where we can store something. A text variable stores words instead of numbers."
- **Using text variables:** "Once we store something in a text variable, we can use it to make the micro:bit display names, messages, or other text."

2. Begin with a fun introduction:

- **Ask the students:** "If you could make your micro:bit display anything, what would it be?"
- **Write their suggestions** on the board (e.g., their favorite pet's name, a funny phrase). This builds excitement for the activity.

3. Use step-by-step guided practice:

- Demonstrate the steps to create a text variable using the micro:bit editor on a projector or screen.
- Walk through each action slowly, explaining why each step is necessary (e.g., "We drag the set block because we're telling the micro:bit what to do with our variable").

4. Relate coding concepts to real life:

- Explain variables as "containers" or "labels" that hold information.
- Use a real-world analogy for joining text, like joining two Lego pieces together or combining "first name" and "last name" to create a full name.

5. Highlight common mistakes and solutions:

- Mention that new variables are always created as number variables by default. Teach them to check this and change it to a text variable if needed.
- Emphasize careful dragging of blocks into the right places. For instance, show what happens if the join block is left empty or in the wrong position.

6. Reinforce learning with hands-on activities:

- **Activity 1:** Creating a Text Variable
  Guide students to create a variable (e.g., "dogName") and input a value like "Buddy." Encourage them to experiment with other pet names.
- **Activity 2:** Joining Text Variables
  Help them combine two variables, like "Name" and "Surname," to display a full name.
- **Challenge:** Ask them to create their own combinations (e.g., a greeting like "Hello" + "World" or "My dog's name is " + "Buddy").

7. Encourage creativity

Allow students to personalize the variables they create. For example, instead of "John" and "Smith," they can use their own names or fun words like "Super" and "Coder."

8. Ask engaging questions throughout:

- "What would happen if you joined the words 'Hello' and 'World'?"
- "Can you think of a time when joining text could be useful? (e.g., creating usernames or messages)"
- "Why do you think we need to use the set block before showing the text?"

9. Summarize with Q&A:

Use the chapter's questions to review key points:

1. What is another name for text in coding? (Answer: Strings)
2. How do you create a text variable? (Answer: Use the set block and add a text block.)
3. How do you join text variables together? (Answer: Use the join block.)
4. What is an example of a text variable? (Answer: A name, like "John.")
5. Which block is useful to display a text variable on the micro:bit screen? (Answer: Show string block.)
6. What is the purpose of the set block? (Answer: To assign a value to a variable.)

10. Provide constructive feedback during activities:

- Check their code to ensure proper placement of blocks.
- Encourage debugging if something doesn't work: "Let's check if the variable has a value set correctly."
- Celebrate creativity and problem-solving efforts.

These tips will make the chapter interactive, engaging, and practical for students learning to create and work with text variables!

Student manual, page 50

Section 5: Questions and Answers

1. **What is another name for text in coding?**
   Answer: In coding, text is also called strings.

2. **How do you create a text variable?**
   Answer: You create a new variable and then change it from a number variable to a text variable by dragging the " " variable block into the value slot of your variable set block.

3. **How do you join text variables together?**
   Answer: You can use the join block from the text toolbox to join text variables together.

4. **What is an example of a text variable?**
   Answer: An example of a text variable is "dogName" or "Surname" that contains your dogs name or your surname.

5. **If you want to show a text variable on the micro:bit LED screen, which block is useful?**
   Answer: The "show string" block can be used to display text.

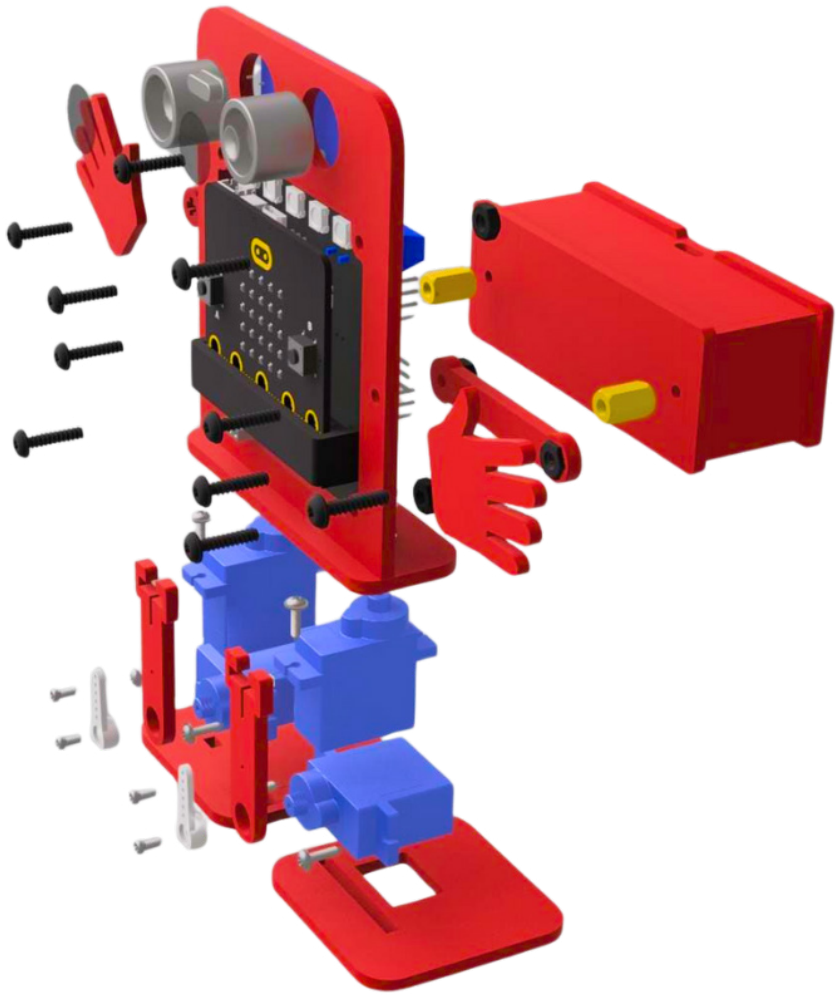
6. **What is the purpose of the set block when working with text variables?**
   Answer: The set block allows you to assign a value to the text variable.

Teacher Notes:

Student Manual, page 51-72

Section 6:

Building a robot



Teacher tips - Building Your First Robot: The Jitter:Bit

We have done some great exercises and lots of learning up to now. But it all means little until the student can apply it in real life. What better way than using a real dancing robot!
It's time to roll up your sleeves and build your first robot—the Jitter:Bit dancing robot! This section combines all the coding concepts your students have learned so far with hands-on assembly and real-world problem-solving. By the end of this section, your students will have built a functional robot and programmed it to perform basic movements. How exciting is that?

We'll cover everything from identifying the parts in the Jitter:Bit kit to testing the robot once it is built. You'll also get detailed instructions, fun activities, and troubleshooting tips to ensure a smooth and enjoyable experience.

5.1 What Is the Jitter:Bit Robot?

The Jitter:Bit robot is an entry-level learning robot powered by the Micro:Bit and includes components like servos and sensors to help students understand how robots move and interact with their environment.

Why the Jitter:Bit?

- **Simple Design:** Perfect for beginners, with easy-to-assemble parts.
- **Interactive Learning:** Combines hardware and software to create a hands-on experience.
- **Room for Creativity:** Once the robot is built, students can program it to do all sorts of cool things!

5.2 Unpacking the Jitter:Bit Kit

Before we start building, let's get familiar with the parts in the kit. Lay everything on a table so students can see what they'll be working with.

What's in the Box?

1. **Chassis:** The robot's body, which holds everything together
2. **Servos:** Four servo motors, servo motors, are specialized motors that can be controlled more accurately.
3. **micro:bit:** The microcontroller.
4. **Ultrasonic Sensor:** Acts as the robot's "eyes," detecting obstacles.
5. **Fasteners**: Screws, nuts, and bolts to assemble the parts.
6. **micro:bit Holder:** A secure place to attach the micro:bit.
7. **Battery pack:** Rechargeable battery pack

Activity: Identify the Parts

Time: 10 minutes
Objective: Help students learn the names and functions of the robot parts.
Instructions:

1. Show each part to the class and explain its purpose (e.g., "This is the ultrasonic sensor; it helps the robot detect obstacles.").
2. Pass the parts around so students can examine them.
3. Create a simple matching activity where students label the parts with their names and functions.

Teaching Tip:

Encourage students to think of the robot as a living thing. For example, the chassis is its body, the servos make the legs move, and the micro:bit is its brain.

5.3 Step-by-Step Assembly

Now, let's put the Jitter:Bit together! Follow these instructions carefully, and your students will have a fully assembled robot by the end of the session.

Tools You'll Need:

A small screwdriver (included in some kits).

Activity: Build the Robot
Time: 45 minutes
Objective: Assemble the Jitter:Bit robot in small groups.
Instructions:

1. Divide the class into groups of 3–4 students.
2. Assign roles within each group (e.g., builder, parts manager, quality checker).
3. Guide the groups through the assembly process step by step.

Teaching Tip:
Walk around the room as students assemble their robots. Offer encouragement and help troubleshoot any issues. Celebrate milestones, like attaching the wheels or mounting the sensor.
Building instructions Video www.botshop.co.za/jitterbit

5.4 Testing the Build

Once the robots are assembled, it's time to test them! Testing ensures that all the components are correctly connected and functioning.

Testing Checklist:

1. **Power On the micro:bit:**
   Ensure the micro:bit is securely attached and powered via USB or a battery pack.
2. **Default program:**
   The default program is available here if it is not loaded yet: www.botshop.co.za/jitterbit 404 error.
3. **Test the Ultrasonic Sensor:**
   Place your hand in front of the ultrasonic sensor. The robot should move backwards.
4. **Test servo motors alignment:**
   The Jitter:Bit will move on a load hand clap in front of it.
5. **Inspect for Loose Parts:**
   Ensure all screws and connections are secure.

5.5 Troubleshooting Assembly

If something doesn't work, don't worry—troubleshooting is part of the process! Here are common problems and solutions:

- **Problem:** The robot doesn't move.
  Solution: Check motor connections and ensure they're plugged into the correct pins and the connectors are the right way up.
- **Problem:** The ultrasonic sensor isn't detecting objects.
  Solution: Verify the wiring and ensure the sensor is mounted correctly.
- **Problem:** Loose parts or rattling.
  Solution: Tighten all screws and nuts.

Teaching Tip:

Discuss real-world applications of obstacle detection, like self-driving cars or robotic vacuum cleaners.

FAQ: Building Your First Robot

Q: What if some parts are missing from the kit?
A: Contact the supplier immediately. In the meantime, use a spare kit or pair up groups to share components.

Q: My students are struggling to assemble the robot. How can I help?
A: Break the process into smaller steps and demonstrate each one. Offer extra guidance to groups that need it.

Q: Can we decorate the robots?
A: Absolutely! Let students personalize their robots with stickers, markers, or even googly eyes. It makes the experience more engaging and fun.

Student manual, page 72

Section 6: Questions and Answers

1. **What are fasteners in robotics?**
   Answer: Fasteners are things like screws, nuts, and bolts that hold a robot's parts together.

2. **What is a servo motor?**
   Answer: A servo motor is a type of motor that can move to a specific position very precisely.

3. **What does an ultrasonic distance sensor do?**
   Answer: An ultrasonic distance sensor sends out sound waves and listens for the echo to determine an object's distance.

4. **What is the role of fasteners when building a robot?**
   Answer: Fasteners help to prevent a robot's pieces from falling apart when it moves around.

5. **How does a servo motor help a robot?**
   Answer: A servo motor helps a robot do precise movements, like waving or hitting a specific spot.

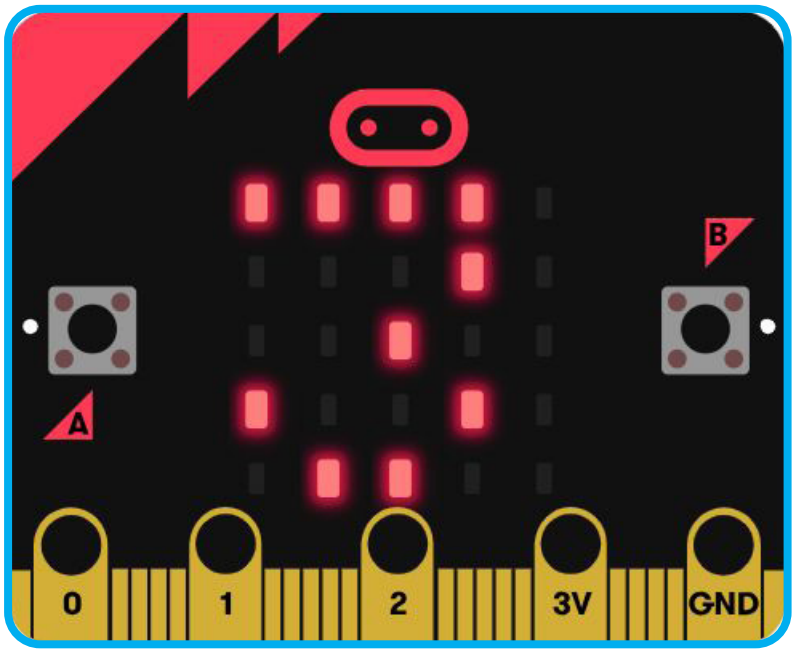
6. **Why is a distance sensor useful for a robot?**
   Answer: A distance sensor helps the robot know where to go without bumping into things.

Teacher Notes:

Student Manual, page 73-78

Session 7:

Inputs and Outputs



Teacher tips - Inputs & outputs

Welcome to the foundational building blocks of coding! This section is where your students start to understand how robots interact with the world and make decisions. We'll explore essential coding concepts like inputs, outputs, variables, conditionals, and loops while keeping things fun and hands-on. By the end of this section, your students will see how these "blocks" combine to create powerful, intelligent programs.

Keep telling the students that they can not break the micro:bit or the robot with code. This will make them feel more comfortable trying out different blocks.

Teaching Tip:

Use real-world examples to explain inputs and outputs. For example, "When you press a key on a piano (input), it makes a sound (output)."

Student manual, page 78

Section 7: Questions and Answers

1. **What are inputs in robotics?**
   Answer: Inputs are like the robot's senses that help the microcontroller understand what's happening outside.

2. **Give three examples of inputs.**
   Answer: Three examples of inputs are buttons, heat sensors and distance sensors.

3. **What are the outputs of robotics?**
   Answer: Outputs are actions the microcontroller can take, like moving parts, making sounds, or lighting up LEDs.

4. **Give three examples of outputs.**
   Answer: Three examples of outputs are LED lights, motors, and speakers.

5. **When do the "on button press" blocks work?**
   Answer: The "on button press" blocks work independently, constantly monitoring the buttons for a press in the background.

6. **How can we use the LED screen as output?**
   Answer: One of many examples can be to use the LED screen to display a letter based on which button is pressed.

Teacher Notes:

Student Manual, page 79-83

Section 8:

Number Variables

Teacher Tips for the Chapter: "Number Variables and Counters"

1. Build on prior knowledge:

- Start by recalling the previous lesson on text variables. Ask the students, "What do you remember about variables? What were we able to do with text variables?"
- Explain that number variables are the default type and are even easier to work with because they don't require changing the variable type.

2. Relate concepts to real-world examples:

- Explain that a number variable is like a score in a video game or the number of coins collected in a treasure hunt.
- Use simple, relatable examples like counting steps with a pedometer or keeping track of points in a game to explain counters.

3. Use visual demonstrations:

- Show how to create a variable in MakeCode using a live demonstration or screenshots. Highlight how the variable defaults to a number.
- Emphasize that the value can only be set or changed when the set block is dragged into the work area, not in the variable toolbox folder.

4. Highlight important concepts with examples:

- Explain why you cannot do mathematics with text variables: "Imagine trying to add the word 'ten' to the number 5. The microcontroller won't understand!"

- Use a coin collecting robot example to demonstrate how to set an initial value for a variable (e.g., coinCount = 0) and how to increment it with the change block.

5. Guide students through hands-on activities:

- **Activity 1:** Create a counter
  Walk students step-by-step through creating a variable (Counter), setting its default value, and making it count up continuously.
- **Activity 2:** Explore the pause block
  Have students experiment with changing the pause block value to see how it affects the counter's speed. Ask, "What happens if you make the pause very short? Very long?"
- **Activity 3:** Change the increment
  Let them modify the change block to count by 2s, 5s, or other increments instead of 1.

6. Reinforce learning with engaging challenges:

- **Challenge 1:** Ask students to create a counter that counts down instead of up (e.g., starting from 10 and subtracting 1 each time).
- **Challenge 2:** Create a counter that displays only even numbers (e.g., 0, 2, 4, 6).
- **Challenge 3:** Add a second variable (e.g., Multiplier) and use it to multiply the counter value before displaying it.

7. Promote critical thinking with questions:

Use the provided questions to check understanding:

1. **What is the default type of a newly created variable?**
   Answer: Number variable
2. **Can you do mathematics with a text variable?**
   Answer: No
3. **How do you set a default value for a number variable?**
   Answer: Drag the set block into the work area and assign the default value.
4. **How can a variable value change automatically?**
   Answer: Use the change block to increment or decrement the value.
5. **What is a counter in coding?**
   Answer: A counter keeps track of a value by increasing or decreasing it automatically.
6. **What is the purpose of the pause block when you are working with counters?**
   Answer: It controls the speed of the counter by pausing for a set amount of time between updates.

8. Provide visual and verbal feedback:

- Watch students' code as they work and offer encouragement: "Great job! Your counter is working perfectly!"
- Troubleshoot common mistakes, like forgetting to drag the variable into the show number block or leaving the pause block out of the forever loop.

9. Relate counters to real-life applications:

Discuss how counters are used in everyday life, such as:

- Tracking steps in a fitness tracker
- Counting scores in games.
- Keeping track of inventory in stores.

10. Wrap up with reflection:

- Ask students to explain in their own words: "What did you learn about counters today? Can you think of another way we could use a counter in coding?"
- End with a challenge question to spark curiosity: "What if we made a counter that counts only odd numbers? How would we do that?"

These tips will make the lesson engaging, hands-on, and easy to follow, ensuring students grasp the concepts of number variables and counters effectively!

Student manual, page 83

Section 8: Questions and Answers

1. **What is the default type of a newly created variable?**
   Answer: The default type of a new variable is a number variable

2. **Can you do mathematics with a text variable?**
   Answer: No, you cannot do mathematics with a text variable, even if it contains numbers. You must use a number-type variable.

3. **How do you set a default value for a number variable?**
   Answer: Once created, you can add a default value to the variable using the "set" block.

4. **How can a variable value change automatically?**
   Answer: You can use the "change" block to automatically add or subtract a number from a variable's current value.

5. **What is a counter in coding?**
   Answer: A counter is a variable that increases by one each time it is activated. It is commonly used to track how often something has occurred.

6. **What is the purpose of the "pause" block when you are working with counters?**
   Answer: The "pause" block can be used to adjust how quickly the counter changes.

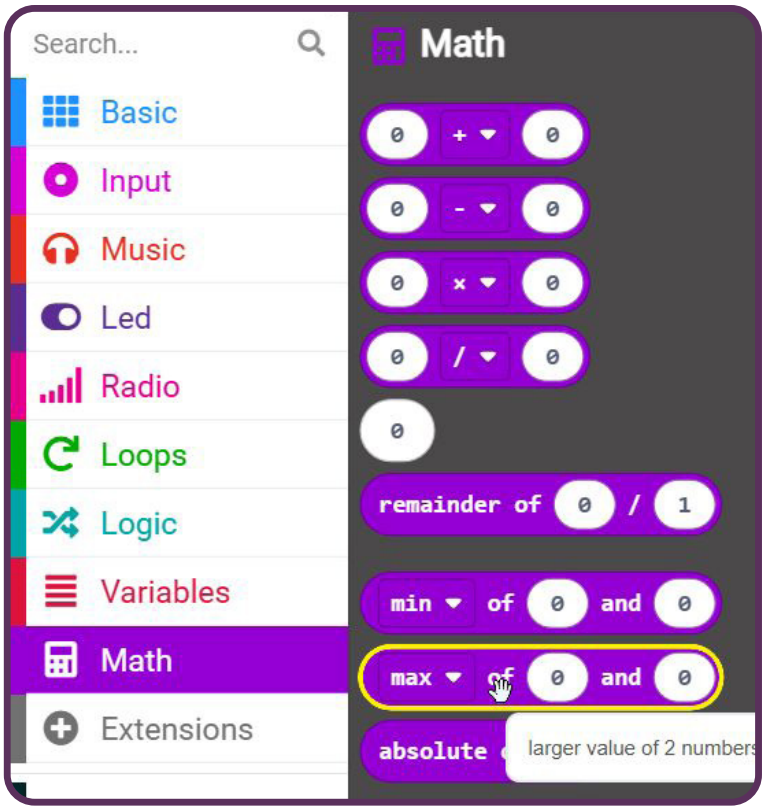
Teacher Notes:

Teacher Notes:

Student Manual, page 85-89

Section 9:

Use Operators like Add, Subtract, Multiply



Teacher Tips:
Using Operators (Add, Subtract, Multiply, Divide)

1. Start with Real-World Connections

Begin the lesson by relating math operations to real-life situations:

- **Addition (+)** → "If you have 5 apples and get 3 more, how many do you have?"
- **Subtraction (-)** → "You had 10 chocolates, but you ate 4. How many are left?"
- **Multiplication (×)** → "Each friend gets 3 stickers, and you have 4 friends. How many stickers do you need?"
- **Division (÷)** → "You have 12 cookies and 3 friends. How many cookies does each friend get?"
- Encourage students to share their own examples.

2. Use Hands-On Demonstrations

- Open MakeCode and show how to drag and drop the math blocks.
- Demonstrate each operator with simple numbers (e.g., 5 + 3 = 8).
- Let students predict answers before showing results on the micro:bit.

3. Explain How the micro:bit Thinks

- The micro:bit processes math inside its brain before showing the result.
- Use an analogy: "The micro:bit is like a calculator—it does the math first, then tells us the answer."
- Show how to connect a calculation block to a display block to see results.

4. Break Down the Steps for Displaying Results

- Drag the math operator block (+, -, ×, ÷) into the workspace.
- Add two numbers inside the operator block.
- Drag a show number block from the Basic toolbox.
- Place the math block inside the show number block.
- Place everything inside a forever block so it keeps displaying the result.

5. Hands-On Activities

- **Activity 1: Basic Math on the micro:bit**
  - Let students add, subtract, multiply, and divide numbers using blocks.
  - Encourage them to try different numbers and predict answers before running the program.
- **Activity 2: Changing One Number Dynamically**
  - Create a counter that adds 1 every second (0, 1, 2, 3…).
  - Modify it to multiply the counter by 2 (0, 2, 4, 6…).
  - Ask: "What happens if we subtract instead of adding?"
- **Activity 3: Real-Life Problem Solving**
  - Challenge students to create a "Sharing Candies" program using division.
  - Example: "You have 20 candies and 4 friends. How many does each friend get?"

6. Address Common Mistakes

- Forgetting to place the math block inside the show number block.
- Putting the calculation in the wrong order (e.g., dividing 3 by 12 instead of 12 by 3).
- Forgetting to use a forever block, which keeps displaying the answer.

7. Encourage Critical Thinking with Questions

1. **What are math operators used for?**
   Answer: They allow robots to do math, like adding and subtracting numbers.
2. **Name the four math operators.**
   Answer: Addition (+), Subtraction (-), Multiplication (×), and Division (÷).
3. **How do you display mathematical results on the micro:bit screen?**
   Answer: Use the show number block.
4. **How do you connect a calculation block and a display block?**
   Answer: Drag the math block inside the show number block.
5. **Why do you put the calculation blocks in a forever block?**
   Answer: To continuously update and display the result.

8. Wrap Up with a Fun Challenge!

Ask students:

Can you create a program that counts from 1 to 10 and multiplies each number by 2 before displaying it?

These tips will make the lesson interactive, easy to understand, and fun for students learning math operators with the micro:bit!

Student manual, page 89

Section 9: Questions and Answers

1. **What are math operators used for?**
   Answer: Math operators are used to perform calculations on numbers.

2. **Name the four math operators.**
   Answer: The four math operators are addition (+), subtraction (-), multiplication (×), and division (÷).

3. **How do you display mathematical results on the micro:bit screen?**
   Answer: You use the "show number" block to display numbers on the LED screen.

4. **How do you connect a calculation block and a display block?**
   Answer: You drag the calculation block to the display block to connect them.

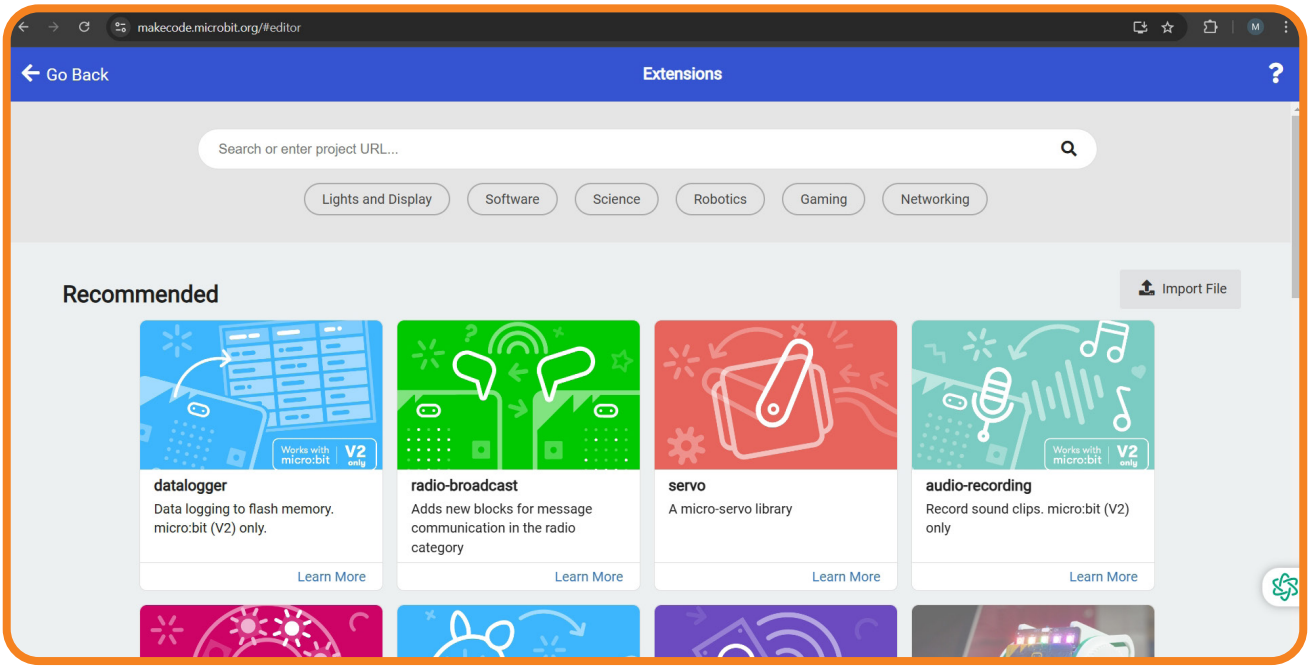
5. **Why do you put the calculation blocks in a "forever" block?**
   Answer: So that it will always show the calculation or result.

Teacher Notes:

Student Manual, page 91-130

Section 10:

Programming a Robot



Teacher Tips: Programming a Robot

Key Learning Outcomes:

- Add and use extensions in MakeCode.
- Program servos, sensors, and LED lights.
- Test and debug robotic functions.
- Explore demo functions like the Scare, Jitter, and Dance Function.

Teacher Tips:

- Explain Extensions: Compare them to app add-ons that expand the micro:bit's capabilities.
- Guide Through Servo Control: Show different movement angles and let students experiment.
- Demonstrate Sensors: Explain how ultrasonic sensors measure distance using sound waves.

Break Down Demo Code Functions:

- **Scare Function:** Teach students how sound sensors trigger movements.
- **Jitter Function:** Explain loops and rapid servo movements.
- **Dance Function:** Show how sequences control robotic behaviours.

Interactive Activities:

- **Activity 1:** Program the Jitter:Bit to move forward and stop when it detects an obstacle.
- **Activity 2:** Create a light show using NeoPixels.
- **Activity 3:** Make the robot "wave" by programming the servo motor.
- **Activity 4:** Customize the Dance Function by adding LED effects.

Common Troubleshooting Tips:

- If the servo doesn't move, check the wiring and MakeCode blocks.
- If the ultrasonic sensor isn't detecting objects, verify the connection to the correct pins.
- If the LEDs don't light up, confirm the NeoPixel extension is installed correctly.

Final Notes:

- Encourage students to experiment and explore beyond the given activities.
- Reinforce problem-solving skills by allowing students to de-bug their own code.
- Foster creativity by having students design their own robot behaviours.
- These teacher notes serve as a quick reference guide for effectively teaching each chapter of the manual.

Student manual, page 130

Section 10: Questions and Answers

1. **What is the Jitter:Bit?**
   Answer: The Jitter:Bit is a robot you build to learn to code robots.

2. **In previous sections, did we build the Jitter:Bit or code it too?**
   Answer: In previous sections, we built the Jitter:Bit, uploaded default code to the robot brain, and played with it.

3. **Which parts of the Jitter:Bit robot will we use to learn new coding concepts in this section?**
   Answer: In this section, we will use the servos and distance sensors to learn new coding.

4. **Does this section focus only on building robots or coding as well?**
   Answer: This section focuses on coding the Jitter:Bit robot.

5. **Which sections prepared the learner for coding this robot?**
   Answer: Section 1 to Section 9 prepared the learner for this coding.

6. **What two parts of the robot will we use to explore more coding?**
   Answer: We will use the servos and distance sensor.

Teacher Notes:

Teacher Notes:

Student Manual, page 131-139

Section 11:

Microcontroller Parts and Its Blocks

Teachers tips - LEDs sound

Bonus Activity: Light Up Your Robot

Time: 20 minutes
Objective: Teach students to create and display images on the LED matrix.
Instructions:

1. Open MakeCode and create a new project.
2. Drag a "show icon" block from the Basic toolbox into the "on start" block.
3. Select a built-in icon (e.g., heart, smiley face) from the dropdown menu.
4. Test it by downloading the code to the micro:bit.

Bonus Challenge:

Ask students to combine multiple "show icon" blocks in a "forever" loop to create a blinking effect.

Activity: Scrolling Messages
Time: 15 minutes
Objective: Program the LED matrix to display scrolling text.
Instructions:

1. Drag a "show string" block into the "on start" block.
2. Type a short message inside the block, like "Hello, World!"
3. Test the code by downloading it to the micro:bit.

Teaching Tip:

Let students personalize their messages, like displaying their names or a fun greeting.

Activity: LED Animations

Time: 30 minutes
Objective: Create an animated sequence using the LED matrix.
Instructions:

1. Drag multiple "show LEDs" blocks into a "forever" loop.
2. In each block, select different patterns for the LEDs.
3. Add "pause" blocks between each "show LEDs" block to control the animation speed.
4. Test the code and watch the animation play.

Bonus Challenge:

Encourage students to create an animation that tells a short story or shows a countdown.

7.2 Making Your Robot make sounds

Robots don't just move and blink—they can make sounds, too! Adding sound to your robot can turn it into a beeping, buzzing, or musical creation. The micro:bit can control a buzzer or speaker to produce tones and melodies.

How Sound Works

1. Frequency: Controls the pitch of the sound (high or low).
2. Duration: Determines how long the sound plays.
3. Volume: Adjusts how loud the sound is.

Activity: Basic Beeps

Time: 15 minutes
Objective: Teach students how to create simple beeping sounds.

Instructions:

1. Connect a buzzer or speaker to the micro:bit using alligator clips (e.g., GND and P0).
2. Open MakeCode and create a new project.
3. Drag a "play tone" block from the Music toolbox into the "on start" block.
4. Select a note (e.g., middle C) and set the duration (e.g., 500 ms).
5. Test the code by downloading it to the micro:bit.

Activity: Play a Melody

Time: 20 minutes
Objective: Program the robot to play a melody when it starts.
Instructions:

1. Drag multiple "play tone" blocks into the "on start" block.
2. Arrange the tones to create a simple melody.
3. Add "pause" blocks between the tones for proper timing.
4. Test the code on the robot.

Bonus Challenge:

Ask students to create their own unique melodies and perform a "robot concert" for the class.

7.4 Troubleshooting Lights and Sounds

- **Problem:** The LED matrix doesn't light up.
  Solution: Check the connections and ensure the micro:bit is powered correctly.
- **Problem:** The buzzer doesn't make a sound.
  Solution: Verify that the buzzer is connected to the correct pin and ground.
- **Problem:** The timing of lights and sounds is off.
  Solution: Adjust the "pause" blocks to fine-tune the sequence.

FAQ: Lights, Sounds, and Action

Q: Can I use the LED matrix and sound together in one program?
A: Absolutely! The micro:bit can handle multiple tasks at once. Include both "show icon" and "play tone" blocks in your code.
Q: What if my robot's sounds are too quiet?
A: Check the buzzer or speaker connection. If possible, use a powered speaker for louder sounds.
Q: How do I make animations smoother?
A: Reduce the "pause" duration between "show LEDs" blocks to make the transitions faster.

With lights, sounds, and movement combined, your students' robots are now bursting with personality! In the next section, "Advanced Coding Concepts," we'll dive into using math, complex logic, and loops to make robots even smarter. Let me know when you're ready for the next section!

Student manual, page 139

Section 11: Questions and Answers

1. **What are some additional electronic components often found on microcontroller boards?**
   Answer: Some additional components are Sensors, LEDs, USB ports, and battery connectors.

2. **What are two ways that a micro:bit can be powered?**
   Answer: The Micro:bit can be powered with a USB cable or with batteries.

3. **What does LED stand for?**
   Answer: LED stands for Light Emitting Diode.

4. **What is a matrix in the context of LEDs?**
   Answer: A matrix is like a grid of rows and columns where LEDs are placed to display numbers, text and graphics.

5. **What are X and Y coordinates used for in the LED matrix?**
   Answer: The X and Y coordinates specify which LED in the matrix will light up. Rows are X, and Columns are Y.

6. **What can you use a speaker or buzzer for in robotics?**
   Answer: Speakers and buzzers can make sounds for alerts or notifications or even play basic tunes.

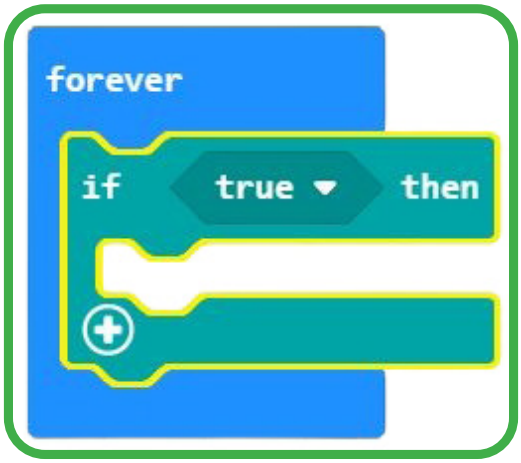
Teacher Notes:

Teacher Notes:

Student Manual, page 141-146

Section 12:

More Operators and the if Statements



Teachers tips - operators

Teaching Tip:

Explain how math is used in real-world robotics, such as calculating distances or adjusting speeds for self-driving cars.

Activity: Displaying Math Results

Time: 15 minutes
Objective: Teach students to use the LED matrix to show the result of a math operation.
Instructions:

1. Drag a "show number" block to the workspace.
2. Inside the block, add a math operation (e.g., 5 + 3).
3. Test the code to see the result (8) displayed on the LED matrix.

Bonus Challenge:

Program the robot to display the distance from the ultrasonic sensor multiplied by 2.

8.4 Debugging Advanced Code

As programs become more complex, the potential for errors increases. Debugging helps identify and fix these issues.

Common Issues and Fixes:

- **Problem:** The robot doesn't respond to conditions.
  Solution: Check the logic in the conditional blocks and ensure all variables are initialized.
- **Problem:** Math calculations are incorrect.
  Solution: Verify the math blocks and test smaller program parts to isolate errors.

FAQ: Advanced Coding Concepts

Q: How do I explain complex conditionals to younger students?
A: Use real-life examples, like: "If it's raining and cold, wear a raincoat. If it's raining but warm, use an umbrella."
Q: What if my students don't grasp loops right away?
A: Start with simple "forever" loops and gradually introduce more conditional-based loops. Practice makes perfect!
Q: How do I challenge students who are advancing faster?
A: Encourage them to create their own mini-projects, like a robot maze solver or a robot that reacts to multiple sensors simultaneously.

With these advanced concepts, your students are now coding robots that are not only intelligent but also adaptable. Next, we'll move on to Capstone Projects, where your students will apply everything they've learned to create something truly amazing. Let me know when you're ready for the next section!

Student manual, page 146

Section 12: Questions and Answers

1. **What are operators in coding?**
   Answer: Operators are symbols or keywords that tell the computer to do something with numbers or information.

2. **Name three types of operators.**
   Answer: Three types of operators are math operators, comparison operators, and logical operators.

3. **What does the "if then" command do in coding?**
   Answer: The "if then" command tells the computer to do something if a specific condition is true.

4. **What are comparison operators used for?**
   Answer: Comparison operators are used to compare two or more values.

5. **What is an example of a comparison operator?**
   Answer: Examples of comparison operators are =, >, and <.

6. **If you are using the "If" block, where do you find the correct operator?**
   Answer: You find the correct operator in the logic toolbox folder.

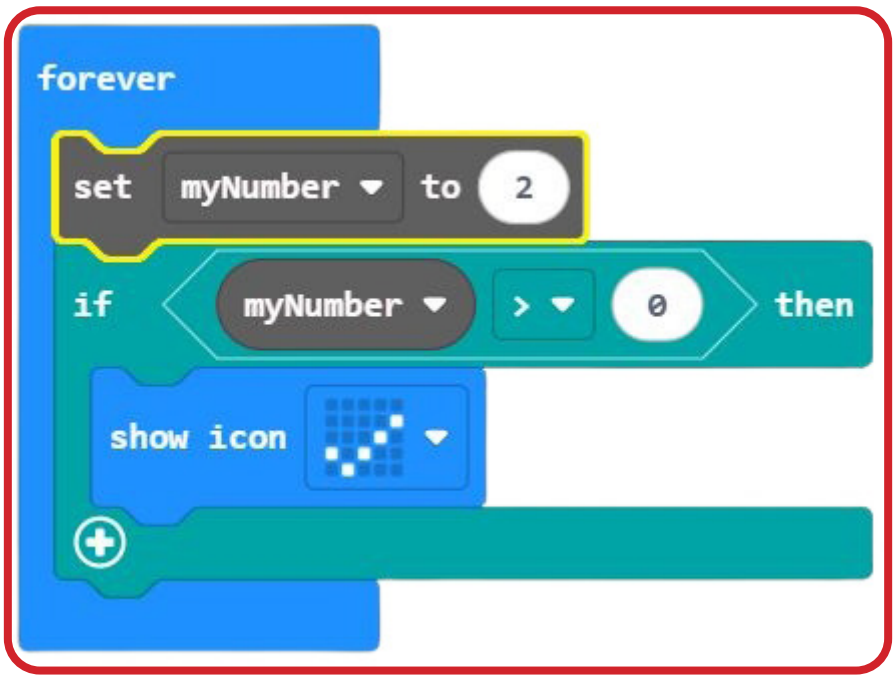
Teacher Notes:

Teacher Notes:

Student Manual, page 147-151

Session 13:

More Control Blocks



Teachers tips - More Control Blocks

Key Learning Outcomes:

- Understand event triggers and how they detect actions.
- Use repeat blocks to simplify tasks by repeating actions a set number of times.
- Customize repetitions in MakeCode.

Teacher Tips:

- Introduce Event Triggers with Real-World Examples: Compare them to light sensors that turn on streetlights at night or motion detectors for security alarms.
- Demonstrate the Touch Sensor: Have students try pressing a button versus touching the micro:bit logo to feel the difference.
- Explain Why Event Triggers Are Outside the Forever Loop: Show how they continuously wait for an action rather than running in a loop.

Live Coding Activity:

- Have students program an event trigger that displays a heart when the micro:bit is touched.
- Extend the activity by making the heart flash multiple times using a repeat block.
- Challenge: Ask students to create their own event triggers, such as making a sound or changing the LED display when touched.

Discussion Questions:

- Why are touch sensor and button blocks not inside the forever loop?
- How is a touch sensor different from a button?
- What are some practical applications of event triggers in daily life?

Final Notes:

- Encourage students to experiment and explore beyond the given activities.
- Reinforce problem-solving skills by allowing students to debug their own code.
- Foster creativity by having students design their own robot behaviors.

Student manual, page 151

Section 13: Questions and Answers

1. **What is a touch sensor?**
   Answer: A touch sensor is a sensor that detects when you touch it with your finger.

2. **What is an event trigger?**
   Answer: An event trigger is code that is used when a particular event occurs, such as pressing a touch sensor or a button.

3. **Why do touch sensors and button blocks exist outside the "forever" loop?**
   Answer: Touch sensors and button blocks exist outside of the forever loop because they constantly monitor for user interaction.

4. **What is the repeat block used for?**
   Answer: The repeat block repeats the blocks you drag into it as many times as you want.

5. **What does the "on logo press" block do?**
   Answer: The "on logo press" block is activated when you touch the touch sensor on the micro:bit.

6. **What is an example of a practical way to use the repeat block?**
   Answer: The repeat block is used to repeat a specific action, like flashing an LED.

Teacher Notes:

1. Teaching Tips and Classroom Management

Robotics and coding are exciting but can also be challenging—especially when managing curious minds, wires, buzzing robots, and the occasional coding error. This section focuses on strategies to help you create an engaging, organized, and supportive classroom environment. Classroom management might be a bit different than usual in robotics and coding. The tips in this section will help to keep students on task, improve teamwork, or troubleshoot common issues so that your robotics lessons run smoothly.

1.1 Managing Groups and Time

Robotics and coding are naturally collaborative, so group work is a big part of learning. Managing group dynamics and keeping students on track will maximize their learning and minimize frustration.

Group Work ideas

- **Small Teams work best:** Divide the class into groups of 3–4 students. This size ensures everyone has a role and feels involved.
- **Assign Roles to Learners:** Having different roles keep tasks organized. Examples:
  - ***Coder:*** Focuses on programming.
  - ***Builder:*** Assembles and modifies the robot.
  - ***Tester:*** Ensures the robot works as expected.
- **Rotate Roles:** Always encourage the students to try different roles throughout the lesson or project. Changing roles ensures everyone gains well-rounded skills.

Activity: Team Contract
Time: 15 minutes
Objective: Help teams establish ground rules for working together.

Instructions:

1. Provide each team with a worksheet to create a team contract.
2. **Ask them to agree on:**
    - How they will communicate and share ideas.
    - How they'll handle disagreements.
    - What to do if someone is stuck or off-task.

Time Management Tips

- **Set Clear Goals:** Start each session by outlining the objectives and tasks for the day.
- **Use Timers:** Break activities into chunks and use a timer to keep everyone on schedule. For example, allocate:
    - 10 minutes for brainstorming.
    - 30 minutes for building/programming.
    - 10 minutes for testing.
- **Plan for Wrap-Ups:** Allow students at least 5–10 minutes at the end of each session to reflect, tidy up, and store their robots.

Teaching Tip:

Have a "time-out station" where students can test their robots without distracting others. This will help manage classroom noise and ensure focused collaboration.

1.2 Encouraging Creativity and Problem-Solving

Robotics isn't just about following instructions—it's about thinking critically and creatively. Encourage students to approach challenges with curiosity and persistence.

Fostering Creativity

- **Ask Open-Ended Questions:**
  - "What else could your robot do?"
  - "How could you solve this problem differently?"
- **Encourage Personalization:**
  - Let students decorate their robots or add unique features to them.
  - Incorporate their interests, like programming the robot to do other things not in the assignments.
- **Celebrate Experiments:**
  - Praise efforts, even if the robot doesn't work perfectly. Remind students that trial and error is part of the process and that they can not damage the robot with coding.

Activity: Brainstorm a Robot Superpower

Time: 10 minutes
Objective: Encourage students to think creatively about what their robot could do.
Instructions:

1. Ask each group to imagine their robot has a "superpower."
2. Have them sketch or describe the superpower and how it would work.
3. Discuss as a class how coding and sensors could bring their ideas to life.

Teaching Tip:
Use real-world examples of innovative robots, like robotic pets or delivery drones, to inspire your students.

1.3 Troubleshooting Common Challenges

Time: 10 minutes
Objective: Encourage students to think creatively about what their robot could do.

Instructions:

1. Ask each group to imagine their robot has a "superpower."
2. Have them sketch or describe the superpower and how it would work.
3. Discuss as a class how coding and sensors could bring their ideas to life.

Teaching Tip:
Use real-world examples of innovative robots, like robotic pets or delivery drones, to inspire your students.

Solution: Verify that the sensor is mounted correctly and that the wiring is secure. Test it with a known object and ensure that the code is reading the correct pin.

Coding Errors

- **Problem:** The program doesn't work as expected.
- **Solution:** Debug systematically:
  - Break the program into smaller parts.
  - Test each block of code separately.
  - Use the MakeCode simulator to identify errors.
- **Problem:** The robot moves erratically.
- **Solution:** Check the logic in loops and conditionals. Ensure motor speed settings are balanced.

Classroom Challenges

- **Problem:** Some students finish early.
- **Solution:** Offer bonus challenges, like adding a new feature to their robot or optimizing their code for efficiency.
- **Problem:** Some students struggle to stay engaged.
- **Solution:** Pair them with a peer mentor or give them smaller, manageable tasks to build confidence.

Teaching Tip:

Create a "robotics toolbox" with extra materials (e.g., spare wires, screws) and reference sheets for debugging common issues.

1.4 Building a Supportive Classroom Culture

Create a positive classroom environment that encourages students to take risks and embrace learning through experimentation. Make the sessions fun and encourage coding failures; they're great opportunities to gain experience.

Promoting Collaboration

- Peer Feedback: Teach students to give constructive feedback. Use phrases like, "I like how you did this, but what if you tried this too?"
- Group Reflection: After each session, have teams share what worked, what didn't, and what they learned.

Activity: Robot Show-and-Tell
Time: 15 minutes
Objective: Foster a sense of accomplishment and teamwork.

Instructions:

1. Ask each team to demonstrate their robot's progress at the end of a lesson.
2. Encourage classmates to ask questions or offer ideas for improvement.
3. Celebrate each group's unique approach.

Encouraging "not giving up"

- Remind students that mistakes are learning opportunities. Share stories of famous inventors and engineers who failed before succeeding.
- Use phrases like, "What did you learn from this?" or "How can we fix this together?"

FAQ: Teaching Tips and Classroom Management

Q: How do I manage students who dominate group activities?
A: Rotate roles within the team to ensure everyone contributes equally. Gently remind dominant students to listen to their teammates' ideas.
Q: What if there's only one robot for the whole class?
A: Use a rotation system where groups work with the robot while others focus on coding simulations in MakeCode.
Q: How do I handle technical issues I can't solve?
A: Don't be afraid to say, "Let's figure this problem out as a team" Use online forums and documentation, or ask a tech-savvy colleague for help.

With these teaching tips and classroom management strategies, you're well-equipped to guide your students through a fun and productive robotics experience. The "Additional Resources and Glossary" section will provide tools and terms to support your teaching journey. Let me know when you're ready!

2. Additional Resources and Glossary

This section provides additional resources to support your teaching and help your students dive deeper into robotics and coding.

Whether it's finding tutorials, troubleshooting guides, or learning new terminology, these tools will enhance your classroom experience. We've also included a glossary of key terms to ensure you and your students speak the same "robotics language."

2.1 Online Tutorials and Tools

Here's a curated list of online resources to help you expand your knowledge and give your students extra challenges. These are great for advanced learners, troubleshooting, or exploring new ideas.

For Teachers:

3. Everything Jitter:Bit related:

Many videos that include building videos, example robot code and much more. Botshop.co.za/jitterbit

1. **micro:bit Official Website**
   - https://microbit.org
   - A treasure trove of tutorials, lesson plans, and ideas for integrating the micro:bit into your curriculum.
2. **MakeCode for micro:bit**
   - https://makecode.microbit.org.
   - The official coding platform with sample projects and simulations to experiment with.
3. **Edutopia: Teaching Robotics**
   - https://www.edutopia.org.
   - Articles and tips for creating an engaging STEM classroom.

Reflection Sheet ideas after class that can be great homework.

- **What Did I Learn Today?**
  - Students summarize what they learned, what challenges they faced, and how they overcame them.
- **What Would I Do Differently?**
  - Encourages students to think critically about how to improve their approach next time.

3.1 Fun Facts About Robots

Use these facts to inspire your students and spark curiosity:

1. The word "robot" comes from the Czech word robota, which means "forced labor."
2. The first industrial robot, Unimate, was created in 1961 to work on an assembly line.
3. Robots explore places humans can't, like the deep ocean and outer space (e.g., NASA's Mars rovers).
4. There's a robot in Japan called Pepper that can understand and respond to human emotions.
5. The smallest robot in the world is the size of a grain of rice!

3.2 Glossary of Key Terms

Here's a list of essential terms to help you and your students navigate the world of robotics and coding.

General Terms

- Algorithm: A step-by-step set of instructions to solve a problem.
- Code: The language we write to instruct a computer or robot.
- Debugging: Finding and fixing errors in a program.
- Microcontroller: A small computer that controls a robot (e.g., the micro:bit).

Coding Terms

- **Conditional:** A statement that runs code only if a specific condition is true (e.g., "If distance < 10 cm, then stop").
- **Loop:** A command that repeats actions (e.g., "forever" or "repeat 10 times").
- **Variable:** A container that stores data, like a number or a word.

Robotics Terms

- **Actuator:** A component that moves or controls parts of the robot (e.g., motors, wheels).
- **Sensor:** A device that gathers information from the environment (e.g., ultrasonic sensor, accelerometer).
- **Servo Motor:** A motor that can move to specific positions is used for precise movement.
- **Chassis:** The body or frame of the robot that holds all components together.

3.3 FAQ: Additional Resources and Glossary

Q: How do I decide which resources are best for my class?
A: Start with beginner-friendly resources, like the micro:bit and MakeCode tutorials. Gradually introduce more advanced tools as your students gain confidence.
Q: What if my students want to go beyond what we cover in class?
A: Encourage them to explore the online tools and projects listed above. You can also set up a robotics club or offer extra credit for independent projects.
Q: How do I make technical terms more straightforward to understand?
A: Use analogies and real-world examples. For instance, explain a "variable" as a backpack where a robot stores information.

With these additional resources and a handy glossary, you're fully equipped to answer questions, inspire curiosity, and support your students in their robotics journey. This concludes the teacher's guide! If you need help customizing lesson plans or adding new content, let me know. Happy teaching!

4. Final Section: Bringing It All Together

Congratulations! You've reached the final section of the teacher's guide. By now, you've explored the fascinating world of robotics and coding, gained the skills to guide your students, and discovered how to troubleshoot and manage your classroom effectively. This final section is all about reflection, building excitement for future projects, and ensuring that you and your students leave this experience feeling accomplished and inspired.

4.1 Reflecting on the Robotics Journey

Reflection is a crucial part of learning. It helps students consolidate what they've learned and consider how they can apply it in the future. Let's reflect on what you and your students have achieved.

Activity: Reflecting as a Class

Time: 15 minutes
Objective: Summarize what the students have learned and celebrate their accomplishments.
Instructions:

1. **Discuss Achievements: Ask students to share:**
    - Their favourite part of the robotics program.
    - The biggest challenge they overcame.
    - One thing they're proud of.
2. **Create a Class Mural:**
    - Provide a large poster board or whiteboard.
    - Ask each student to write or draw something they learned or enjoyed about robotics.
3. **Highlight Growth:**
    - Share observations about how students developed teamwork, problem-solving, and creativity skills.

Teaching Tip:

Celebrate every milestone, big or small. Recognize not just the students who excelled technically but also those who showed persistence, creativity, or leadership.

4.2 Setting the Stage for Future Projects

Robotics is a journey, not a destination. Inspire your students to think about how they can continue exploring robotics and coding in the future.

Discuss Real-World Applications

- Robots are used in healthcare, agriculture, space exploration, and more.
- Show videos or articles about real-world robots, like Boston Dynamics' Spot or NASA's Mars rovers.

Introduce Advanced Topics

- Artificial Intelligence (AI): How robots can learn and make decisions.
- IoT (Internet of Things): Connecting robots to the internet for advanced functionality.
- 3D Printing: Creating custom parts for robots.

Activity: Dream Big

Time: 20 minutes
Objective: Encourage students to imagine what they'd like to create in the future.
Instructions:

1. Ask students to think of a problem they'd like to solve using a robot.
2. Have them draw or describe their dream robot.
3. Discuss as a class how coding and robotics could bring these ideas to life.

Teaching Tip:

Encourage students to join robotics clubs, participate in competitions, or explore additional online courses to keep learning.

4.3 Sharing Knowledge with the Community

One of the best ways to deepen understanding is by sharing it. Encourage your students to showcase their work and inspire others.

Organize a Robotics Showcase

- Invite parents, teachers, and other students to a "Robotics Expo."
- Set up stations where each group presents their robot and explains how it works.
- Include live demos of robots navigating mazes, dancing, or reacting to sensors.

Create a Class Portfolio

- Compile photos, videos, and reflections from the robotics lessons.
- Share the portfolio digitally with parents and administrators to highlight the students' achievements.

Activity: Teach a Friend

Time: 30 minutes
Objective: Help students solidify their knowledge by teaching someone else.

Instructions:

1. Pair each student with a younger buddy or a peer from another class.
2. Have them explain a simple robotics concept or demonstrate their robot.
3. Encourage questions and discussion.

4.4 Celebrating Success

End your robotics program on a high note by celebrating the hard work and creativity of your students.

Awards and Certificates

- Create fun awards for each group or individual, such as:
  - "Best Robot Design"
  - "Most Creative Programmer"
  - "Best Debugger"
- Provide certificates of completion for all students.

Robot Graduation Party

- Organize a small celebration with snacks and decorations.
- Let students take turns presenting their robots one last time.
- Share a slideshow of photos from the program.

4.5 FAQ: Final Reflections

Q: How can I keep students engaged with robotics after this program?
A: Suggest joining local robotics clubs, participating in competitions like FIRST Lego League, or exploring advanced robotics kits and projects.

Q: What if students want to explore topics I don't know about?
A: Encourage independent learning! Direct them to online resources, tutorials, or even books about robotics.
Q: How do I maintain excitement for future classes?
A: Build on this program by introducing more advanced challenges or starting a year-long robotics project.

Final Words

Teaching robotics and coding is a powerful way to inspire young minds and equip them with the skills they need for the future. Remember, this is just the beginning! You've opened the door to a world of creativity, innovation, and problem-solving.

Thank you for being an amazing guide on this journey. You've made a lasting impact on your students, and who knows? One day, one of them might build the next Mars rover or design a life-saving robot.
Keep exploring, keep learning, and most importantly—keep having fun with robotics!

This concludes the teacher's guide! If you need additional support, lesson plan customization, or extra materials, just let me know.

Teacher Notes:

About Bot Shop:

Vision: Bot Shop aims to make education fun, sparking curiosity and creativity rather than feeling like a chore.

Mission: Simple yet impactful, we make education fun and entertaining, turning **"giving up"** into **"what do I build next?"** Through the X-O-Skeleton series, we transform the micro:bit microcontroller into engaging learning tools.

Location and Production: Proudly made, designed, engineered, and manufactured educational robots in South Africa by Bot Shop.

Our products are tailored to meet the educational needs of South African learners and educators, combining advanced technology with innovative design.

Contact Us

Email: sales@botshop.co.za
Phone: +27 12 002 1651
Website: www.botshop.co.za