LESSON 10

Tilt Remote Control Car

60 minutes (2 periods) Programming Car Behaviours

WHAT THIS LESSON DOES

In this step-by-step lesson, you'll learn to control a Move Motor car using a Microbit as a remote controller. You'll program the Microbit to send directional commands to the car based on its tilt. You'll also create code for the car to respond to these commands, resulting in a fully remote-controlled car.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand and apply the concept of radio communication between two Microbits.
- Programme a Microbit to send specific messages based on different gestures.
- Develop the ability to programme a Move Motor car to respond to different messages received.
- Test and debug the code to ensure the car responds correctly to the remote control.
- Extend the project by adding additional features such as LED light changes.

BEFORE THE BELL

- Count out working pairs: 2 micro:bits and 1 Move Motor car per pair or group. Charge cars and fit batteries the day before.
- Open https://makecode.microbit.org on the school network and confirm students can create projects and add the kitronik-move-motor extension.
- Plan unique radio group numbers per pair so neighbouring cars do not steal each other's messages.
- Clear floor space for test runs and keep spare USB cables ready for downloads.

Tilt Remote Control Car

HOW THE LESSON RUNS

- Introduction
- Program the remote
- Send message 'forward'
- Send messages 'left', 'right' & 'reverse'
- Send message 'stop'
- Download the code
- Create a new project
- Receive message 'forward'
- Receive message 'left'
- Receive message 'right'
- Receive message 'reverse'
- Receive message 'stop'
- Download the code
- Challenge

TEACHING MOVES

- 1. Introduction. Hold up both micro:bits and the car. State the aim in one line: tilt one board, the other drives the car. Name the kit each pair needs before anyone opens MakeCode.
- 2. Program the remote. Get everyone on a new MakeCode project named for the remote. Add radio.setGroup first. Assign each pair their own group number and write it on the board.
- 3. Send message 'forward'. Show LogoDown as tilt forward. Students add the gesture handler that sends "forward" and shows a North arrow so they can see the trigger fire.
- 4. Send messages 'left', 'right' & 'reverse'. Have them copy the same pattern for TiltLeft, TiltRight and LogoUp. Stress exact string spelling: left, right, reverse. Match each arrow to the tilt.
- 5. Send message 'stop'. Add ScreenUp to send "stop" and show the No icon. Ask why a flat, face-up rest position is a safer stop gesture than another tilt.
- 6. Download the code. Download to the remote micro:bit only. Before touching the car, pairs tilt through all five gestures and confirm the matching arrows or stop icon.
- 7. Create a new project. New project for the car board. Add the kitronik-move-motor extension, then set the same radio group as that pair's remote. No shared group across pairs.
- 8. Receive message 'forward'. Build onRadio receivedString with an if that compares to the text block "forward". On match, call the Move Motor forward drive. Show where the Text block lives under Advanced.
- 9. Receive message 'left'. Expand the if with else if for "left" and wire the left turn motor block. Keep the compare strings identical to what the remote sends.
- 10. Receive message 'right'. Add else if for "right" and the right turn. If a pair's car veers the wrong way, check string spelling before blaming the motors.
- 11. Receive message 'reverse'. Add else if for "reverse" and the reverse motor call. Remind them LogoUp on the remote is reverse, not forward.
- 12. Receive message 'stop'. Final else if for "stop" that stops both motors. Without this branch the car keeps the last motion when the remote goes flat.
- 13. Download the code. Flash the car micro:bit, power the car, and test one gesture at a time. If a direction fails, compare send and receive strings for that word only.
- 14. Challenge. For pairs whose drive works, add Move Motor LED colour changes per received message so the car shows state as well as motion.

WHAT TO WATCH FOR

- Students set different radio group numbers on the remote and the car, so nothing is received even though both programs download cleanly. Before the first live test, have each pair read both setGroup values aloud and match them to the number you assigned them.
- Gesture orientation is confused: LogoDown vs LogoUp, or holding the remote so left/right are swapped relative to the car. Agree a grip: logo at the top edge away from the body for forward. Retest arrows on the remote alone before blaming the car code.
- A capital letter or spelling mismatch (Forward vs forward) means the if never matches, so the car ignores a message that was sent. Put the five exact lowercase strings on the board. Students check sendString and the receive compare text character by character.

DIFFERENTIATION

- Emerging: Pair them with a partner whose remote arrows already work, and let them flash and test one gesture at a time. Stay on the remote project until all five gestures show the right feedback before opening the car project.
- Developing: Give a checklist: group number, five send strings, five receive branches, stop last. They tick as each live test passes. If left and right are swapped, have them explain their tilt grip back to you before editing code.
- Proficient: Add LED colour changes per message as in the challenge once drive and stop are reliable. Have them teach another pair how they debugged a failed direction using string matching only.