LESSON 11

Traffic Lights and Car Communication

60 minutes (2 periods) Integration Projects

WHAT THIS LESSON DOES

In this step-by-step lesson, you'll learn how to code a set of traffic lights and a robot car using Microbits. You'll program the traffic lights to run through a sequence and broadcast their state, which the car will receive and respond to accordingly. The lesson also includes a challenge to modify the code so the car only responds when close to the traffic lights.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand and apply the concept of radio communication between Microbits.
- Program a sequence of traffic light signals using code blocks.
- Develop skills to broadcast and receive specific messages based on traffic light states.
- Control the movement of a robot car based on received messages.
- Enhance problem-solving skills by modifying the code to respond based on the proximity of the car to the traffic lights.

BEFORE THE BELL

- Day before: count out per working pair or group 2 charged micro:bits, 1 Move Motor car and 1 Traffic Lights kit. Power-check both micro:bits and the car.
- Confirm https://makecode.microbit.org opens on the school network and that USB cables or download routes work on the class machines.
- Have screws ready to fix one micro:bit to the traffic lights kit; seat the second micro:bit in the car before students code.

Traffic Lights and Car Communication

HOW THE LESSON RUNS

Introduction
Program the traffic lights
Add the Stopbit extension
Program the sequence
Broadcast the state
Program the car
Receive the message
Download the code
Challenge

TEACHING MOVES

- 1. Introduction. Name the two roles out loud: lights micro:bit broadcasts the state, car micro:bit receives and drives. Hand out kit only after students can point to which micro:bit goes where.
- 2. Program the traffic lights. Open MakeCode, new project named for the lights. Set radio group 1 together. If several groups share the room, give each pair its own group number and write it on the board.
- 3. Add the Stopbit extension. Send everyone to Extensions, search stopbit, click the traffic lights kit image. Check that Kitronik STOP:bit blocks now show in the toolbox before they build the sequence.
- 4. Program the sequence. Build a forever loop: green 10 seconds, yellow 3 seconds, red 5 seconds. Read the timings back once so nobody swaps the pauses.
- 5. Broadcast the state. Beside each light state, send the exact strings go, ready to stop and stop. Stress matching spelling. Download only to the micro:bit fixed on the lights.
- 6. Program the car. New MakeCode project for the car. Add the kitronik-move-motor extension, then set the same radio group number as that pair's lights.
- 7. Receive the message. On received string: go drives at 40, ready to stop at 20, stop cuts the motors. Check the three strings match the lights project letter for letter.
- 8. Download the code. Power the lights first and watch one full cycle on the floor. Then power the car a few metres away and look for speed changes tied to each light state.
- 9. Challenge. Show received packet signal strength: about -28 is strong and close, -128 is weak and far. Gate motor actions so they run only above a threshold such as -50.

WHAT TO WATCH FOR

- Car and lights sit on different radio groups, so messages never arrive and students think the receive code is broken. Read both projects' radio.setGroup values side by side. Same number on both, and unique per pair if the room is busy.
- Students treat any download as success and do not notice one micro:bit is unpowered or out of range. Test routine: lights powered and cycling first, then car on, then watch one full green-yellow-red pass before debugging code.
- On the challenge, students reverse signal strength and think a more negative number means closer. Write on the board: closer to -28 is stronger and nearer, closer to -128 is weaker and farther. Try one threshold together.

DIFFERENTIATION

- Emerging: Sit them with a partner whose lights already cycle. Their job is radio group match plus one working receive branch before adding the rest. Stay for the Extensions step: stopbit on the lights project, kitronik-move-motor on the car project, then leave them on the sequence.
- Developing: Have them prove go, ready to stop and stop strings match on both projects before they touch speeds or the challenge. If the car only partly responds, ask them to show you which received-string branch runs for the light they can see.
- Proficient: Point them at the signal-strength challenge: car only slows or stops when strength is above a threshold they choose and can justify. Ask them to tune the threshold by walking the car closer and farther and reporting when behaviour changes.