Rock ‘n Rolling Robots (2-lesson plan)

A BEAM Lesson Plan

Written by Matthew Wang

Introduction

As technology has advanced, robots have become more and more commonplace in the world around us; robots have become an avoidable part of 21st century life. Many robots are autonomous: working without live human input or with just human supervision (a human is “out of the loop”). Autonomous robots are often used in manufacturing, operations/logistics, child and elderly care, and in a wide-variety of other commercial and industrial applications. In addition, autonomous robots are closely linked with self-driving cars, machine learning, and artificial intelligence.

Abstract

The goal of this lesson is to teach students about the types of components that go into robots, to introduce them to basic programming concepts via block-based programming, and to tie these two skills together to create and program a working robot. Throughout this activity, they will also practice critical thinking skills, teamwork, and familiarize themselves with assembling basic products.

Materials

Students

For every group of students, they will need the following:

- 1 mBot Robot ($100 USD MSRP, organizational discounts exist)
    - Robot Frame
    - Microcontroller Shell
    - 2 Motors
    - Line Sensor
    - Ultrasonic Sensor
    - 2 Gear Wheels
    - Front Wheel
    - Battery holder
    - USB-A to USB-B cable
    - Assorted screws
    - Assorted wiring
- 1 laptop or desktop computer
- 1 copy of mBlock IDE (free at http://www.mblock.cc/mblock-software/)
- 4 AA Batteries (~ $3 USD)

We recommend that groups range between 2-4 students – however, the group numbers will differ based on the number of materials available.

We recommend that the instructors attach the Velcro strips to the battery holder and the robot beforehand.

Teachers

In theory, nothing is needed to teach this lesson. However, we recommend that the teachers have a projector connected to a computer, where they can project the instruction manual and go through the interface of the mBlock IDE.

Lesson Plan

As a note, this lesson plan can run for at least an hour and a half, to however long instructors wish to give students time to program. Traditionally, we've run this lesson in two parts on two days: the first lesson is just robot assembly, while the second focuses on programming.

Lesson 1: Assembling the Robots

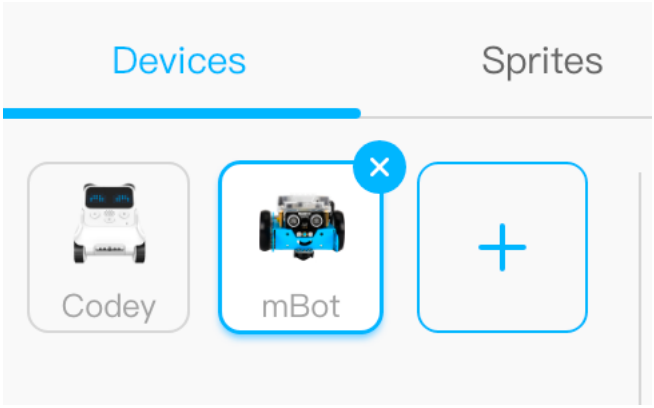
Each robot kit comes with an instruction manual that explains the assembly steps. It's up to the instructors if they feel comfortable with the kids following the instructions or would prefer to walk it through step-by-step in front all of the students. Here are the steps:

1. Setup
    a. Ensure that each kit has everything listed in the materials section – it's possible that some materials have been lost in between lessons.
    b. Explain how the screwdriver in the kit works – some screws are Philips (plus) heads, while some are hex (hexagonal) heads.
2. First, attach the motors to the robot frame, using two long Philips screws per motor.
    a. At this point, ask them to thread the motor wires through the hole at the front of the robot.
    b. If time allows, discuss how electric motors work – have they seen any other electric motors (e.g. electric pencil sharpeners, household appliances, Teslas)? Why are they shaped that way? Why are they on the inside of the car, opposed to the outside?
3. Then, screw each of the rubber gear wheels to the motors, using one small Philips screw per motor.
4. Then, screw in the line sensor and the front wheel into the front of the robot, using two hex screws.
    a. Note that the line sensor's port (similar to an ethernet port) needs to be inside the robot – if not, it will drag when the robot runs.

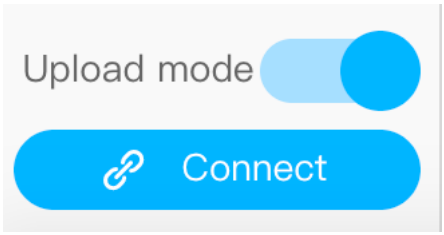
    b. Have the students plug in one wire to the line sensor, and thread the other wire the hole at the front of the robot.
    c. If time allows, discuss how the line sensor works – what technology do they think it uses? Why are there things that look like light bulbs on the line sensor? Have they seen this kind of technology elsewhere?
5. Then, screw in the ultrasonic sensor into the robot, with two hex screws – it should look like eyes are being attached to the "face" of the robot.
    a. Have the students plug in one wire to the ultrasonic sensor, and thread the other wire the hole at the front of the robot.
    b. If time allows, discuss how the ultrasonic sensor works – what do "ultra" and "sonic" mean? Have they encountered ultrasonic sensors in equipment (e.g. some automatic doors, ultrasound)? Are there any analogs in nature (e.g. bats, dolphins)?
6. Then, screw in the four standoffs to the top of the robot – no need to use a screwdriver.
7. Then, attach the battery holder to the top of the robot, aligned with the Velcro strip.
    a. If time allows, discuss how robots are powered. Is it always batteries? Why do we use different kinds of batteries?
8. Then, screw in the robot controller to the top of the standoffs, with four hex screws.
    a. If time allows, discuss what the robot microcontroller is. What is it made of? Why is there a cover on top of it? What else have they seen microcontroller-like items (e.g. a computer, small electric toys)? Is there a biological analogy to a microcontroller?
9. Finally, attach the rest of the wires to the microcontroller:
    a. Plug in the right motor to M1.
    b. Plug in the left motor to M2.
    c. Plug in the line sensor to port 2.
    d. Plug in the ultrasonic sensor to port 3.

Lesson 2: Introducing the Programming

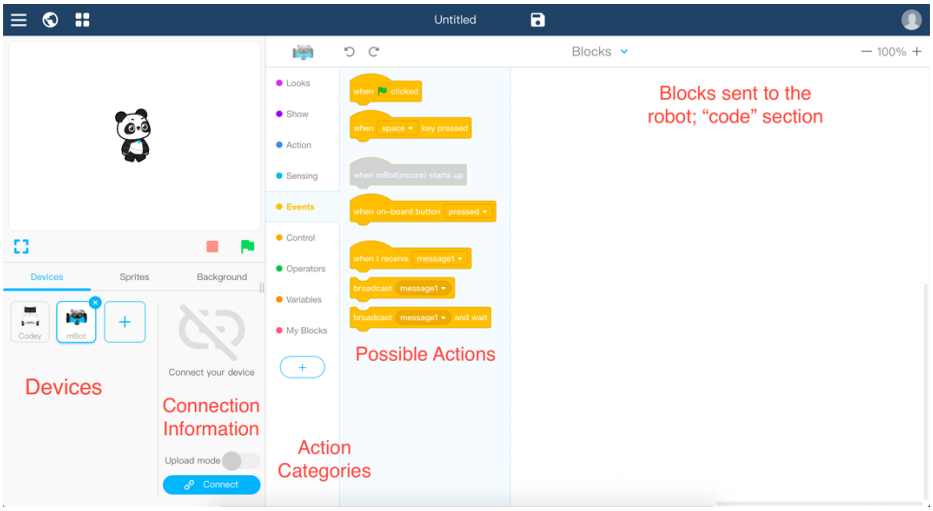
1. Setup
    a. Ensure each student has mBlock IDE installed on their computer
    b. Ensure each student is able to plug in the robot to their computer, via the USB cable.
    c. Have the students create a "mBot" device profile (there should be a + sign) – they must use this device profile to code for the mBots.



d. Have the students tick the “Upload mode” box – this will let them write code for their robots.

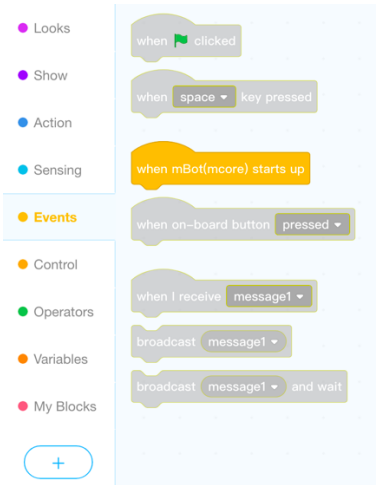


2. Introduce the mBot interface, with these annotated sections:



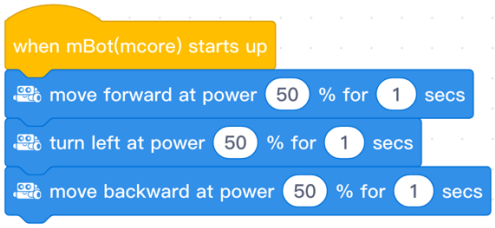
a. Discuss the sequential order of programming – similar to a to-do list. In general, computers can only follow orders one at a time (simplification).

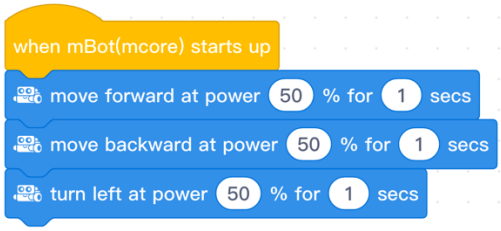
3. Discuss the overall concept of sequential programming – that is, telling the robot to do certain steps in order. In Scratch/mBlock instructions are rectangular blocks that can chain to each other via small tabs on each block. Each block is executed after its previous block is done.
4. Introduce the “Events” tab, which starts a block of code if an event happens – either when the robot turns on, or when a button on the board is pressed etc.



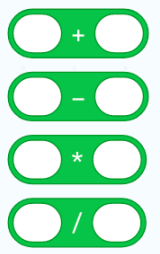
   a. Drag the "when mBot(mcore) starts up" block into the code section. Note the tab at the bottom of the block: the rest of the code that students want to execute should go under this block.

5. Introduce the "Action" tab, which controls the movement of the robots. Chain a few movement controls together and ask the students what they think the robot would do (and where it would end up relative to its starting position).
   a. A good exercise to run through is test their understanding of sequential code – what would be the difference between the following two blocks?

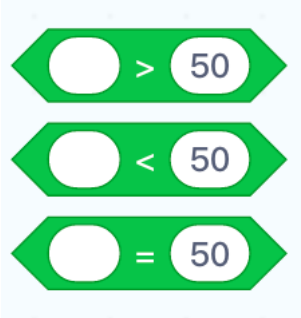




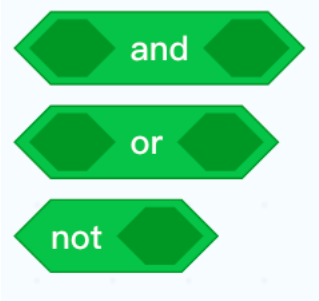
6. Introduce the "Operators" tab – this lets students perform simple mathematical operations, and most importantly, evaluate expressions.
   a. Objects that are circular evaluate to numbers (or words). Students can type into white boxes, use existing operators to get new values, or get input values from sensors.



b. Objects that are hexagonal take in inputs and evaluate to either true or false (in programming, these are Boolean expressions). The most basic operators are greater than, less than, and equal to – they all take in circular inputs (i.e. numbers).

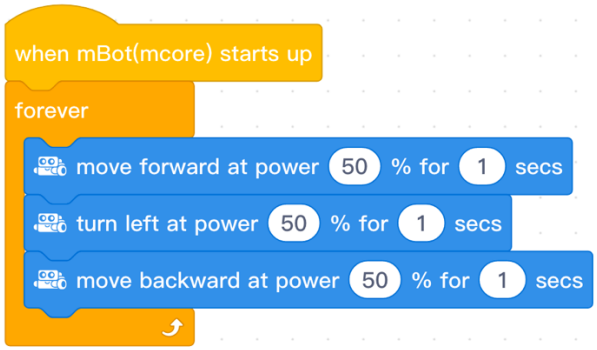


i. If time permits, introduce the more complex Boolean operators, AND, OR, and NOT. They each take in another expression and evaluate them against each other.



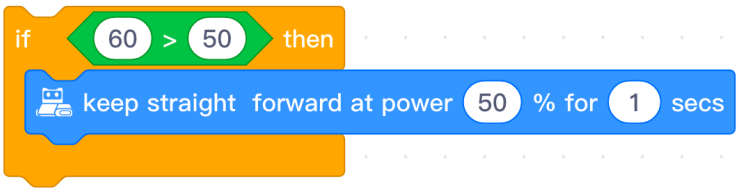
7. Briefly introduce the "Sensing" tab, which contains several circular and hexagonal inputs that students can use to get information from sensors. Likely, the only important sensor to introduce is the ultrasonic sensor, which returns a numerical value based on distance. Note the importance of what port the sensor is connected to!

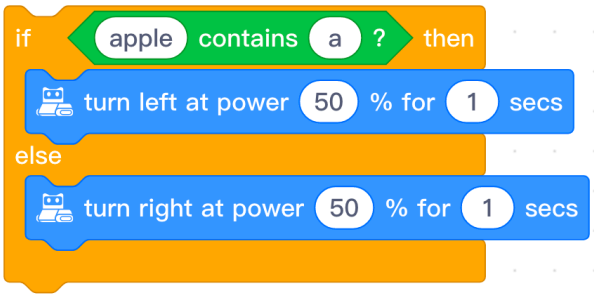
8. Introduce the "Control" tab, which is a simplified version of basic control flow paradigms in computer science. Run through a few key concepts (if, if/else, loops) with both real-life examples and code blocks. This is often the trickiest part of the lesson – spend extra time to ensure the kids understand these concepts.
   a. The "forever" control block is like a "while true" loop in programming, a common control flow used for event loops and real-time controls. Students can place other blocks inside the forever construct – they will execute sequentially and then jump back to the top, forever.



i. Ask students if they can come up with examples of when a forever loop would be used in real life.
ii. Metaphorical examples of forever loops can include many human bodily functions (e.g. breathing, digestion), or electronics that are always on and listening for input
iii. Good (simplified) examples of forever loops in control systems include video games (checking if a button is pressed, rendering frames, etc.), robots (checking sensors for inputs while they're always on), and phones (checking if they've received a notification, cell signal).
iv. If time permits, introduce the "repeat" and "repeat until" blocks, which combine the concept of the forever loop and if statements.

b. The "if then" and "if then else" blocks function like if statements in programming, a very important concept to allow for conditional code (i.e. doing different things based on inputs). Everything inside the block construct will only execute if the condition (the expression in the hexagon) is true.





i. Ask students if they can come up with any examples of an if or if-else statement in real life.
ii. Common examples include "if it's raining, bring an umbrella outside", "if I'm hungry, eat food", "if there's something in front of me, go around it", etc.

iii. If statements are often used in control systems to process outside input; examples include distance sensors (automatic doors, moving robots), light sensors (automatic lights, barcode scanners), sound sensors (clapping lights, smart assistants), and human inputs (computers, phones, buttons).

9. Quickly introduce the "Show" tab – students can use it to change the colour of lights on their robot, or to play a sound.
10. Finally, quickly mention how students will upload their code to their robot:
    a. Ensure the robot is physically connected to the computer via USB
    b. After toggling the robot onto upload mode, hit the "Connect" button
    c. When the students are ready, hit "Upload". Ensure that the robot's wheels are not touching the ground.
11. Now, the students have all the knowledge they need to start programming their robot. Present to them a goal for their robot to aim for, and have groups work on programming their robot to meet that task. Teachers should split up and help groups who are struggling.
12. Examples of possible activities:
    a. Robot Dancing (easy): using a combination of moving the robot, turning lights on/off, and playing sounds, have students create a dance routine for their robot. Tests understanding of sequential programming and robot movement.
    b. No-Bump Robot (medium): using information from the ultrasonic sensor, make the robot move around the room without ever bumping into a wall. Tests understanding of sequential programming, robot movement, using sensors, and control flow.
        i. Example code: