<CODERS>
Missouri State University

<CODERS>

Lesson Name: Micro:bit Radio Communication

School District:

Teacher Name:

Intended Grade Level: All

Differentiation: N/A

Accommodations:

Visually impaired students should be partnered with another student to participate.

Material(s):

- Computer with internet access
- 2 - Micro:bits
- 2 – Micro:bit to Computer cables (micro USB to USB A)

Time allotment (varies according to grade level):

- TBD

Alignment with CODERS module(s):

(Micro:bit)

MLS Standards:

4.AP.M.01 – Decompose (break down) large problems into smaller, manageable subproblems to facilitate the program development process.

5.AP.M.01 – Decompose (break down) large problems into smaller, manageable subproblems and then into a precise sequence of instructions.

4.AP.M.02 – With grade-appropriate complexity, modify, remix or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.

5.AP.M.02 – With grade-appropriate complexity, modify, remix or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.

4.AP.PD.01 – Use an iterative and collaborative process to plan the development of a program that includes user preferences while solving simple problems.

5.AP.PD.01 – Use an iterative and collaborative process to plan the development of a program that includes other perspectives and user preferences while solving simple problems.

Objectives:

- Code the micro:bits to send and receive data via the micro:bit’s radio functionality.

Background information / Activation of Prior Knowledge:

- Access the MakeCode classroom.

- Basic computational thinking skills.
- Basic knowledge of block programming.

Introduction / Anticipatory Set:

Download the receiver and sender programs or make them yourself by following the **"Guided Practice (We do)"** section. Look over the programs and develop familiarity with them. Group students into pairs and assign them a radio group number.

Teaching (I do):

Introduce what the goal of this project is (to send and receive data via the radio functionality) and how you will do it (by using two micro:bits and writing two programs: a sender and receiver program). The sender sends data, and the receiver receives and displays the data. Then depending on students' skill level/demeanor, there are options:

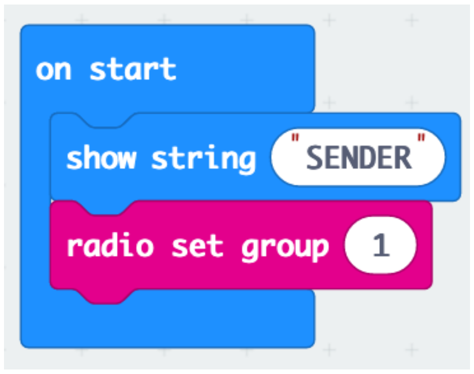
- Assign students the **"Writing Integration"** from the **"Group / Independent Practice (You do)"** section. After they have completed that assignment, you can quickly guide them through the **"Guided Practice (We do) section #2"** to ensure everyone is on the same page.

- Provide students a portion of the sender program (**"Guided Practice (We do) section #2. a: i-iv"**) and/or receiver program (**"Guided Practice (We do) section #2. b: i-iv"**) and have them finish writing the code for one or both programs. Afterward, you can proceed with **"Guided Practice (We do) section #3"**.

Guided Practice (We do):

1. **Plan and design micro:bit programs.**
    a. Lead students through the planning process.
        i. Use a flowchart to show what is necessary for the functionality of both the sender and receiver programs.

2. **Code**
    a. Program "Sender" micro:bit.
        i. Using MakeCode, set up and code the first micro:bit.
        ii. Name the project "Sender".
        iii. The on start event will display the title and function of the micro:bit in all caps, "SENDER".
        iv. Set up a radio group using the radio set group. Both micro:bits need the same radio group. This action is akin to the concept of tuning into a radio station.



```
on start
    show string "SENDER"
    radio set group 1
```

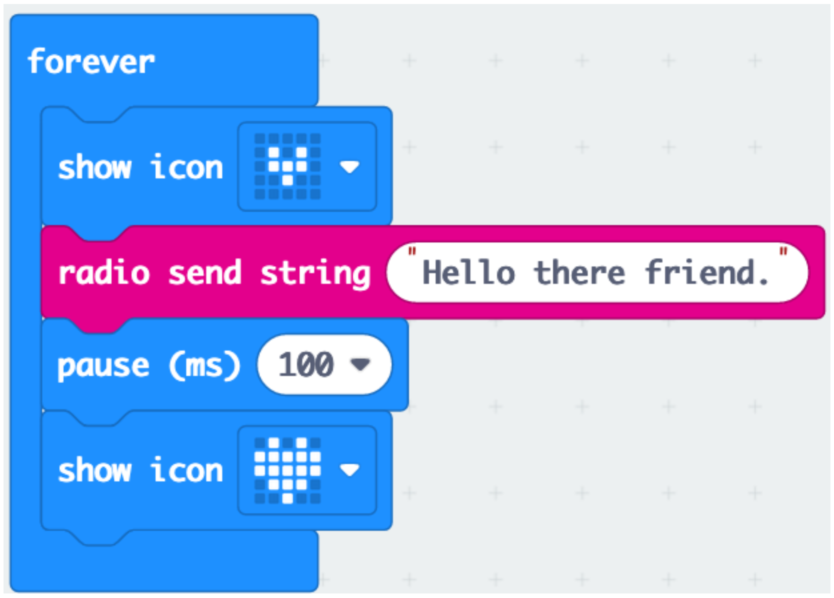
        v. The forever loop will send a "string" to **ALL** other micro:bits that are receiving radio signals in the same radio group.

vi. To do this, put a radio send string in the forever loop and insert a string message in the blank.

vii. To monitor where the micro:bit is at in executing the code. Place a show icon block before and after the radio send the string.

viii. To make it even easier to monitor the sender program, add a pause (ms) block between the radio send string and the show icon block. Choosing 100 from the drop-down menu.

ix. The program is now ready to be flashed to the sender micro:bit.



```
forever
    show icon
    radio send string "Hello there friend."
    pause (ms) 100
    show icon
```

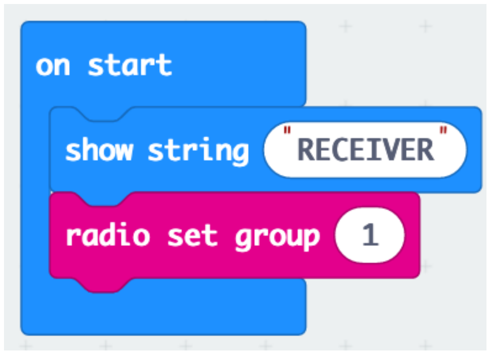
b. Program “Receiver” micro:bit.

i. Using MakeCode setup and code the second micro:bit.

ii. This micro:bit will remain connected to the computer through the USB cable and MakeCode to monitor the data being received. Name the project "Receiver".

iii. The on start event will display the title and function of the micro:bit in all caps, "RECEIVER". Similar to the sender program.

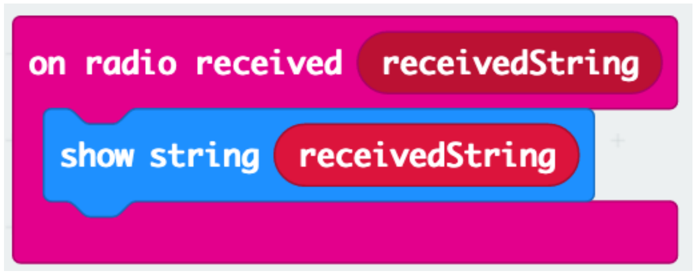
iv. Set up a radio group using the radio set group block. Both micro:bits need to be in the same radio group.


```
on start
    show string "RECEIVER"
    radio set group 1
```

v. The on radio received string event will constantly monitor radio signals from the radio group.

vi. Insert the show string block in the on radio received string event and put the receivedString value in the blank.

vii. The receiver program is now ready to be flashed to the receiver micro:bit.


```
on radio received receivedString
    show string receivedString
```

3. **Testing the sender and receiver programs.**
    a. Both micro:bits are plugged in to a computer and run both programs.

4. **Debugging**
    a. In the previous section, were any issues found in the program? Was the sender sending data? Was the receiver receiving and displaying the data?
    b. Go back to MakeCode and make the changes needed to fix the bugs observed in the testing phase.
    c. Test the two programs again and repeat the debugging process if needed.

Group / Independent Practice (You do):

Writing Integration:

Prompt paired students to write out instructions to program the functionality of the micro:bit’s radio communication. Then have the paired students swap instructions with each other and write code based only on the written instructions. Have students test the programs their partner wrote, write down the bugs, and return the instructions to their partner. The original writer can then modify their instructions to fix the bugs encountered previously.

Notes, Reflections, Attachments