Programming mBot with mBlock

Contents

Assemble and Test mBot

Time required: 60 minutes

"A robot is an autonomous system which exists in the physical world, can sense its environment, and can act on it to achieve some goals." Maja J. Mataric

The mBot is an Arduino based robot. An Arduino board is an open-source hobbyist and prototyping microcontroller with a wide variety of uses from small robots to 3d printers, to electric scooters.

The mBot can be programmed with Scratch based blocks, a mobile device, and Arduino C. It is a fun platform for learning the structures of programming. The mBot allows you to see your "code in motion".

NOTE: Batteries Not Included (Unless you buy the mBot from the WNCC bookstore, you will need batteries.)

STICKY TIP: There is a piece of Velcro to hold the battery pack in. The battery pack fits tight, you really don't need the velcro. If you wish to use the velcro, sut the velcro in half, put it toward the front of the battery pack and the robot. Otherwise, it is hard to get the battery pack out.
Video demonstration

1. Only use AA rechargeable batteries with the robot. Regular alkaline batteries wear down quickly and don't maintain a consistent voltage.
2. Assemble mBot per instructions with kit.
3. Put a battery (**CR2025**) in the remote. You can use the remote to control the robot.

Test the Factory Default Program

This built-in program has three functions.

Remote Control

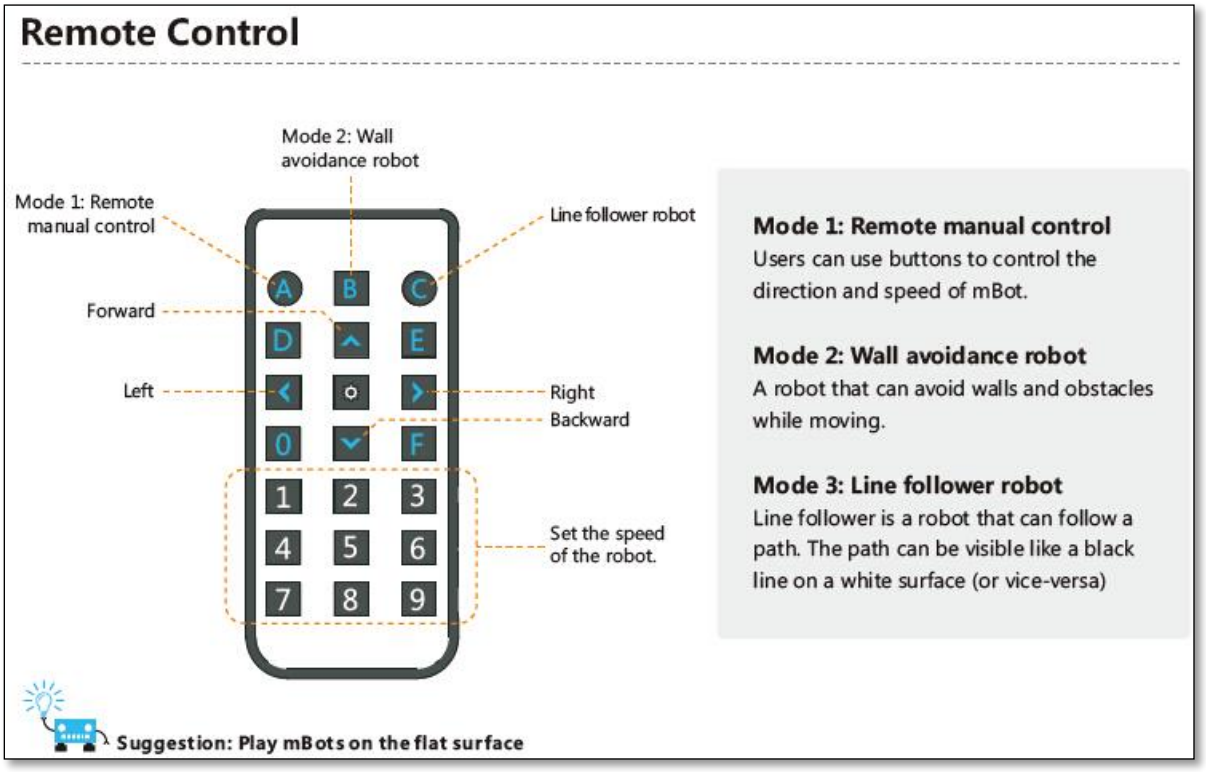
1. The robot starts in remote control mode. Press the arrow keys on the remote to move the robot.
2. To return to remote control, press A, or press the button on the robot. This also serves to stop the robot when it is in either obstacle avoidance, or line following mode.
3. 1-9 changes the speed of the robot.

Obstacle Avoidance

1. Press B on the remote or press the button on the robot.
2. Put your hand in front of the robot while holding it, the "eyes" (ultrasonic sensor) will detect your hand and the wheels should change speed.
3. Put the robot on the floor. It should avoid obstacles, such as furniture, your foot, walls, etc.

Line Following

1. Press C on the remote or press the button again on the robot.
2. Unfold the line follower diagram that came with your mBot. Place the robot on the black line. It should follow the line. And follow the line. And follow the line . . .



Requirements

- Use the remote on the robot to switch modes.
- With the arrow keys, navigate a square and return to the starting point. Change speeds.

- Obstacle avoidance: Show the robot avoiding obstacles. You can use your foot to guide it.
- Line Following: Follow the line on the paper track.

Assignment Submission

- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Get Started with mBlock

Time required: 60 minutes

mBlock is an open-source development environment for Windows and Mac. It is based on MIT's Scratch graphical programming environment. A block-based programming language is an easy way to learn the basic concepts of programming.

1. Go to https://www.mblock.cc/en/download
2. Download and install **mBlock 5**. Accept all default choices. Install the driver when prompted.
3. Run **mBlock**.
4. Select the **Devices** tab→ Click the **x** on the **Codey** icon →Delete it.
5. Click **Add** → add **mBot**.

Blink LED's (First Program)

This first program uses the two LED's toward the front of the robot. This program introduces looping and waiting. Blinking LED's is the traditional Hello World program of an Arduino device.

Knowledge Points

Each block has a different purpose.

forever

The **forever** block causes the program to repeat forever. This is a characteristic of an Arduino microcontroller program. It will keep repeating the loop over and over until powered off.

wait

The **wait** block pauses the program for the specified amount of time. Without the wait block, the LED's would blink so fast you would just see a solid light from the LED's. Remove the wait blocks and see what happens.

LED

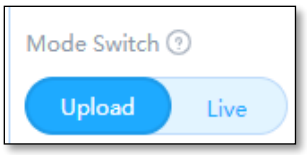
The LED's on the mBot correspond to the standard RGB colors used on the web and other electronic formats. The values range from 0-255.

Requirements

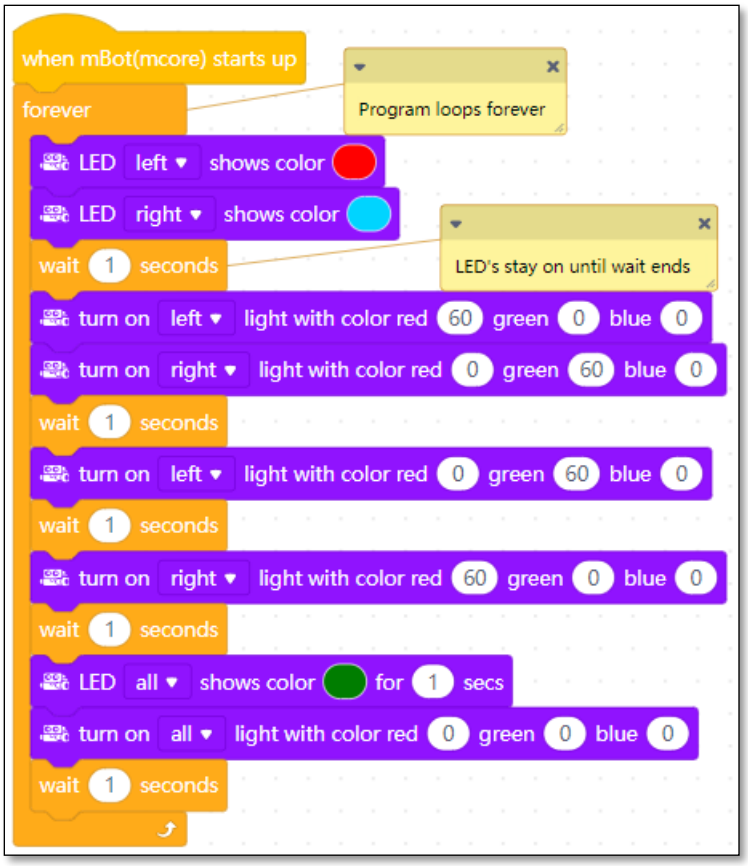
- Program will alternate blinking each LED.
- The program will run immediately and in a continuous loop.
- Each LED will be on for 1 second

Tutorial Assignment

1. Start **mBlock**.
2. Go to the **File** menu → **Save to your computer**.
3. Name the program **Blink LEDs**.
   NOTE: You can't use punctuation in an mBlock file name.
4. Under Mode Switch → Click the **Upload** button to place mBlock into programming mode.

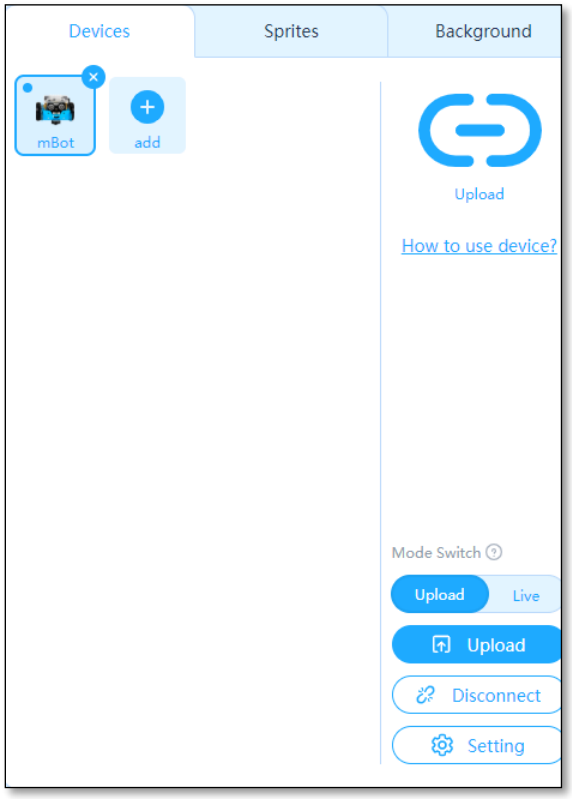


5. Create and test the program as shown.


when mBot(mcore) starts up
forever
Program loops forever
LED left shows color
LED right shows color
wait 1 seconds
LED's stay on until wait ends
turn on left light with color red 60 green 0 blue 0
turn on right light with color red 0 green 60 blue 0
wait 1 seconds
turn on left light with color red 0 green 60 blue 0
wait 1 seconds
turn on right light with color red 60 green 0 blue 0
wait 1 seconds
LED all shows color for 1 secs
turn on all light with color red 0 green 0 blue 0
wait 1 seconds

Upload a Program

1. Power on the robot.
2. Connect the robot to a USB port on your computer with the USB cable that came with the mBot.
3. Select **Connect**. On the USB tab, the correct com port should already be selected. Click **Connect**.
4. Change the **Mode Switch** to **Upload**.
5. Click **Upload**.



Assignment

Add the following to your tutorial project.

- Make up your own colors.
- Change the wait times.

Check Your Understanding

Please answer the following questions.

1. **What is the purpose of the forever block?**

2. **Remove the wait blocks from the program and run it. What happens?**

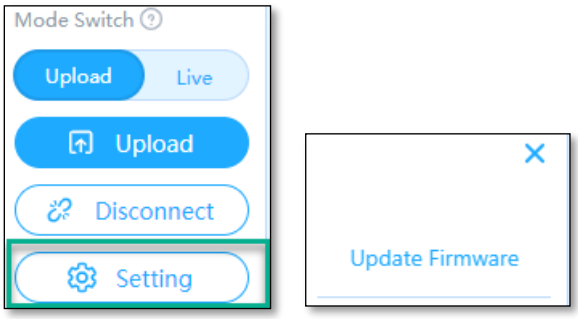
3. **Why does the program set some LED's to 0?**

Factory Default Firmware

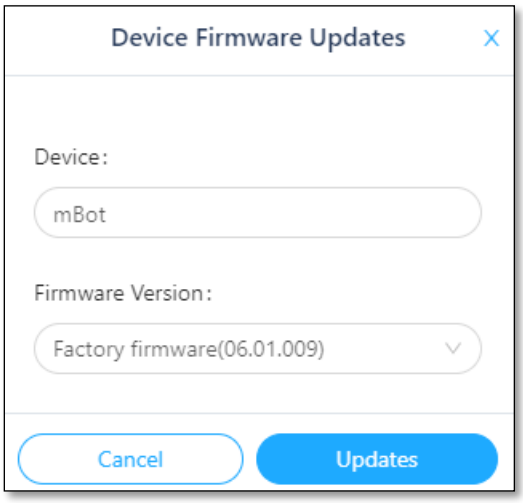
The mBot can only hold one program at a time. You can return your mBot to the original factory software. The following directions show how to upload the factory settings.

1. Power on the robot.
2. Connect the robot to a USB port on your computer with the USB cable.

3. Select **Connect**. On the USB tab, the correct com port should already be selected. Click **Connect**.
4. Change the **Mode Switch** to **Upload**.
5. Click **Setting** → **Update Firmware**



6. Select **Firmware Version** → **Factory firmware**.



7. Click **Updates**.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Random Colors

Time required: 30 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Description

This program shows how to use random numbers to control the LED's on the mBot. Randomization makes programs much more interesting. Randomization is part of game programming. You don't want the game to behave the same way each time you play it.

Understanding

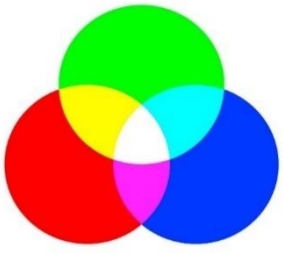
Demonstrate understanding of:

RGB colors, LED's, Random Numbers

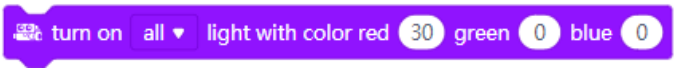
Knowledge Points

Principles of Color and Light Mixing

Each of the onboard LED's are 3 LED's: Red, Green, and Blue, put together in one package. The color of the onboard LED uses the RGB color model. It is a color-adding model, which shows a variety of colors effects by mixing two or three colors in different ratios.

	From the RGB diagram: red+green=yellow green+blue=azure blue+red=pink red+blue+blue=white

Color and Light of the Onboard LED

The color and light values of the onboard LED's use the RGB color map. The larger the value in 

is, the brighter the light is, and mixed color is brighter than a single color.

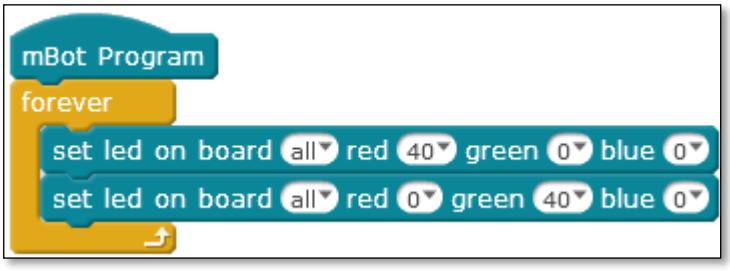
You can use the RGB color table to set the onboard LED color. The onboard LED color is seen more easily when the value is below 40. Divide the RGB value (the 3rd column in the following table) by 10 and round it to determine the on-board LED value (the 4th column in the following table). This keeps the colors proportional.

RGB real color	Name	R.G.B value	Onboard LED value	Block setting
	Cyan4	0 139 139	0 14 14	
	DodgerBlue	30 144 255	3 14 26	
	SpringGreen2	0 238 118	0 24 12	
	Firebrick	178 34 34	18 3 3	
	VioletRed	208 32 144	21 3 14	
	Maroon	176 48 96	18 5 10	
	DarkGoldenrod4	139 101 8	14 10 1	
	Cougar Blue	0 58 112	0 6 11	
	Cougar Gold	249 190 0	25 19 0	

What if You Execute Two LED Blocks of Different Colors?

The mBot runs programs very rapidly, there isn't any "waiting" between "red light" and "blue light". It is the same as lighting red and blue at the same time, you will see yellow. The following two programs show yellow when they are executed. The wait block is needed for a pause between color changes.

Code	Effect

<table>
<tr>
<td> 


</td>
<td>The onboard LED will continually display yellow with either version of the program.</td>
</tr>
</table>

Random Number Block

The random number block generates a random number each time the program is executed. For example, the result of rolling a dice can be considered random since the outcome is inclusive between integer 1-6.

1. **Define the inclusive range of the random number.**
    a. Define the two inclusive ends of the range of the random number. The range can start from a smaller number to a larger number, or vice versa.
    b. Double-click to enter the range directly. A decimal or negative number is also acceptable. 

2. **Random integer and random decimal**. When both numbers are integers (whole numbers), an integer will be selected randomly. If there's a decimal in at least one number, a random decimal will be generated. For example, 

will generate a random decimal.

In this assignment, random numbers are selected as the values of the three primary colors, red, green, and blue. In this way, red, green, and blue colors are combined randomly, producing a random color for the LED light.

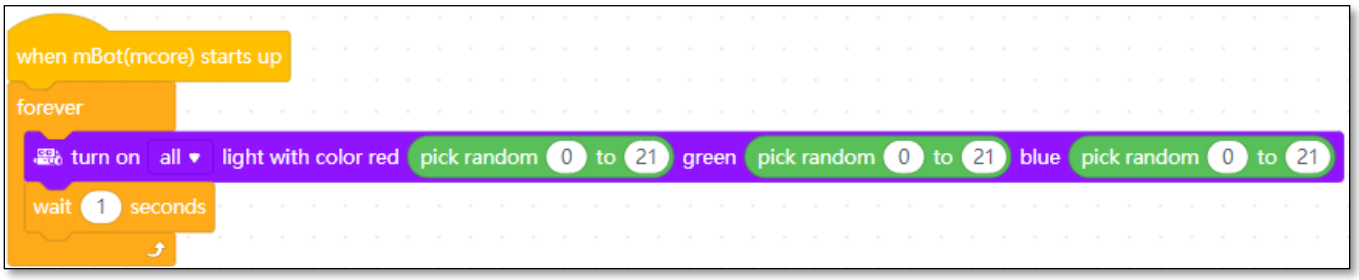
NOTE: LEDs do not accept decimals as brightness. Use integers (whole numbers).

Requirements

The onboard LED lights change randomly every second.

Tutorial

1. Start mBlock. Save the program as **Random Colors**.
2. Create and test the program as shown.



Assignment

Start with your tutorial project and add the following.

1. Change the left and right LEDs on board to show different random colors.
2. Choose a random value for one or two colors, set the others to a static value.
3. Change the range of the random numbers.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Simple Movement

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Description

The program we build in this assignment moves the mBot in four directions at a predetermined speed. The movement blocks, **move forward**, **move backward**, **turn left**, and **turn right** can be set from 0 to 100 percent. Don't set the speed below 30 percent or the motor will stall. 50% will be the power setting we will typically use in our programs.

Understanding

Demonstrate understanding of:

wait until, repeat and loops

Knowledge Points

wait until

This block waits until an event happens, in this case, pressing the ir remote up arrow button.

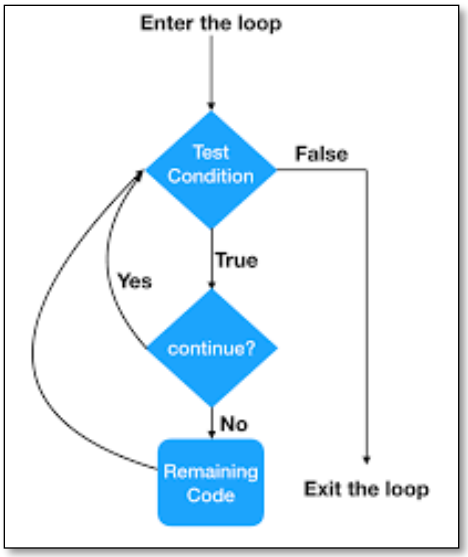
repeat

The repeat block repeats the code inside the block the listed number of times, in this case, 4 times. This simulates mBot waggling.

Repeats are Loops

The repeat block in mBlock is a repetition structure. It is called a loop in most programming languages. This type of loop executes a set number of times, then exits and moves on to the next step.

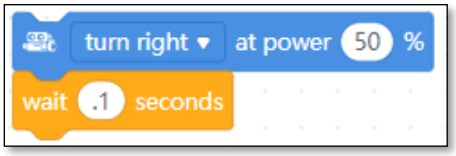
The loop starts at 1 the first time through the loop. Each time the number is incremented, next to 2, then 3, etc., until the test condition returns false. In the example program, when the loop gets to 5, the test condition is false, the program exits the loop and moves on.



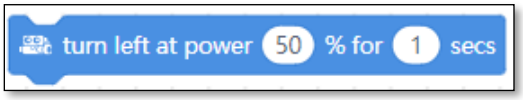
Time Duration for Movement

For the robot to move, there must be a time duration for the wheels to turn. The motors move like this.

1. Electrical control pulse is sent to the motors.
2. The motor starts moving.
3. Time duration must be set for the motor to move. Each of these control blocks have a time duration.
    1. **Wait** .1 seconds will let the motor turn for .1 second.

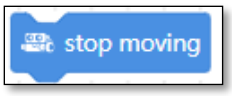


    2. **Turn left at power** for so many seconds. This block has wait built into it.



    3. **Play note**. A note has time duration. One beat is a second. .25 beats is .25 seconds.

4. **Stop moving** sends an electrical control pulse to stop the motors.

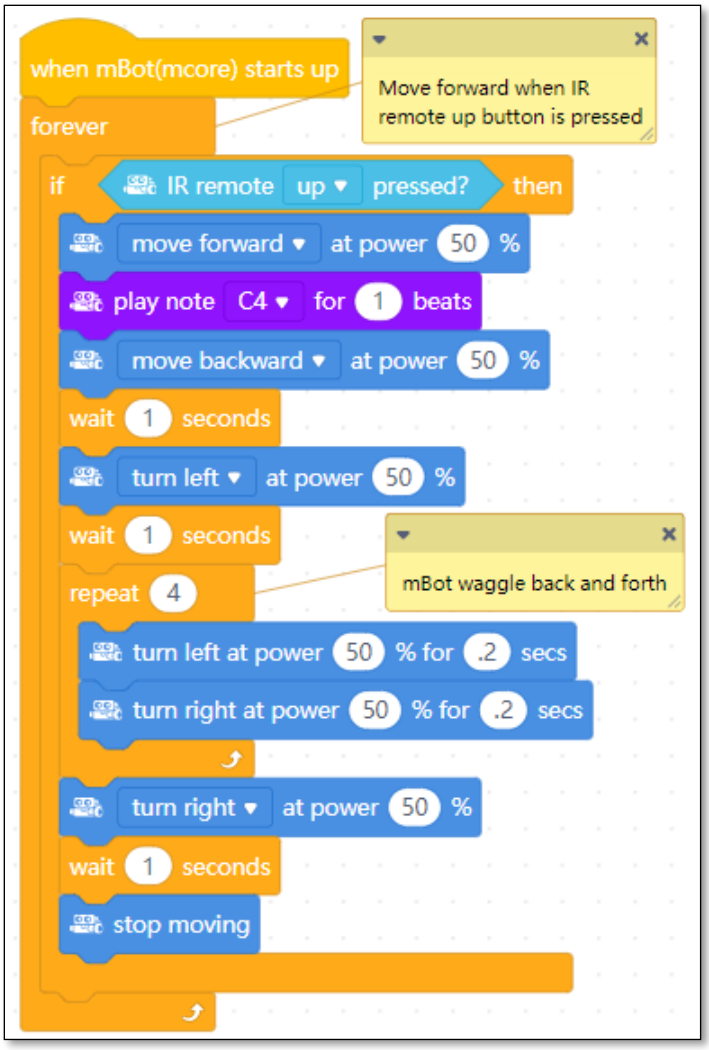


Requirements

The mBot will wait until the up arrow on the remote is pressed, then run through the programmed motions and stop.

Tutorial

1. Start mBlock. Save the program as **Simple Movement**.
2. Create and test the program as shown.



Assignment

Start with your tutorial project and add the following.

- Change the movements to create your own version of the program.
- Change the wait times.
- Make up your own challenge.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.

- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Music and Lights

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Description

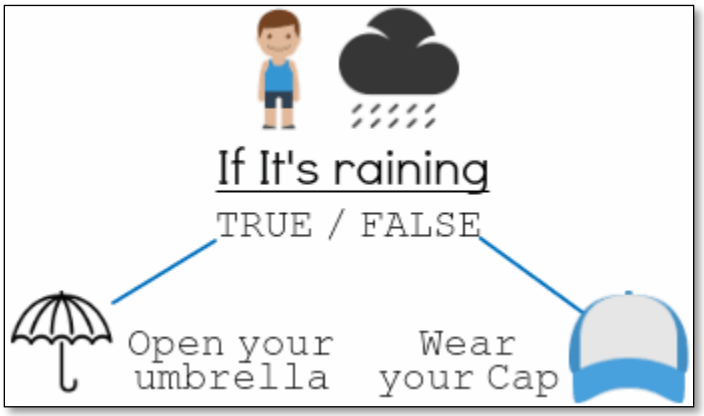
This project adds music along with lights to your mBot. This program introduces a decision structure, the if statement.

Knowledge Points

if then block, tone block

Decisions, Decisions: The if then Block

In life, people need to choose or carry out different tasks according to conditions, such as wearing T shirts in hot weather or padded clothes in cold weather. The air temperature is a condition. Whether you wear T shirt or padded clothes depends on the judgment of the air temperature.



Robots need to make decisions. For a robot, if the button is pressed, it turns on the LED's. If the button is not pressed, it does nothing. Whether the button is pressed or not is a condition and whether to turn on the LED's or play sounds is the executed result after judgment.

In 

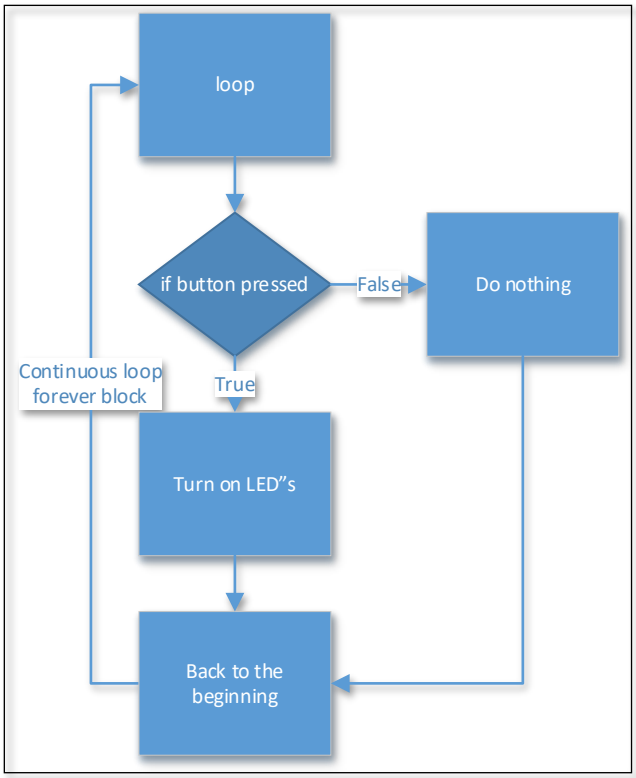
, if the "if" condition  is true, execute the block script within. In this program example, the condition is "whether the onboard button is pressed"; when it is true, the onboard buzzer sounds a tone and when it is not true, nothing is done.

What if I do not use 

?

To test the status of the onboard button all the time when the program is running, you need to use the "forever" block. If you do not use "forever", the program, when started, will run through the program once and then stop. Pressing the button will not control the program.

The following diagram shows the typical execution of an if statement in an Arduino type of program. It is continuously looping waiting for something to happen.



Play Tone Block

There is one drop-down menu in the play tone block 

defining the pitch of the note. The length of the note is defined in beats.

1. C/D/E/F/G/A/B in the tone menu defines the name of the tone, mapping to Do/Re/Mi/Fa/So/La/Ti of C major. The number behind a tone stands for different pitches, C4 for standard middle C, and C5 for a higher octave and C3 for a lower octave.
2. The beats option is how long the note is played. Each beat is a whole note, or 1000 milliseconds.

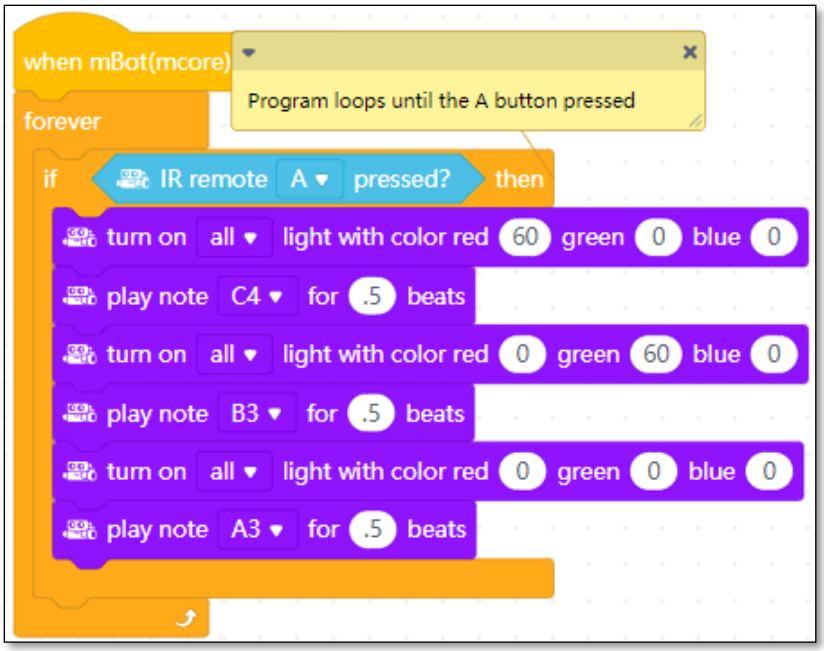
Double (two whole notes)	2 beats	2 seconds, 2000 milliseconds
Whole	1 beat	1000 milliseconds
Half	.5 beat	500 milliseconds
Quarter	.25 beat	250 milliseconds
Eight	.125 beat	125 milliseconds

Requirements

- The program will run when you press the number A button.
- The program will play 3 notes and change the LED colors at the same time, then wait until the remote button is pressed again.

Tutorial Assignment

1. Start mBlock. Save the program as **Music and Lights**.
2. Complete and test the program as pictured with the requirements listed.



Assignment

Start with your tutorial project and add the following.

- Add more notes, change their durations.
- Add more LED color changes, flashing and blinking.
- Change the sequence from alternating to a different pattern, both on at the same time, both off, etc.

Check Your Understanding

Please answer the following question.

1. **Why doesn't this program have wait blocks?**

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Ambulance

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

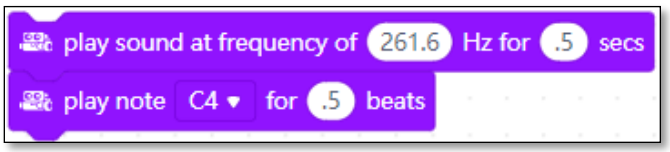
Understanding

frequency of sound, code blocks

Ambulance Sounds

Ambulance sound effects consist of two tones, a high frequency sound of 950Hz and low frequency sound of 700Hz. The high frequency sound lasts for 0.6 second and the low frequency sound 0.4 seconds. The high and low frequency sounds are played alternatively. Double click to enter either the frequency for the note, or the milliseconds for the sound. There are 1000 milliseconds in a second.

Use of the Tone Playing Block



For this project, the tone within the tone-playing block is the sound frequency in Hz. You can double click a note value to enter the frequency value for the buzzer. For example, the frequency of C4 is 261.6Hz, so both blocks are equivalent.

The rhythm within a tone-playing block is the duration of a sound in beats or seconds. Each beat is 1 second, or 1000 milliseconds. You can double click the beat value to enter a duration in milliseconds.

Code Blocks

A Block is a chunk of modular code that can be reused in the program without having to write the code again and again. You can break your code into several blocks, rather than one big, long program.

Requirements

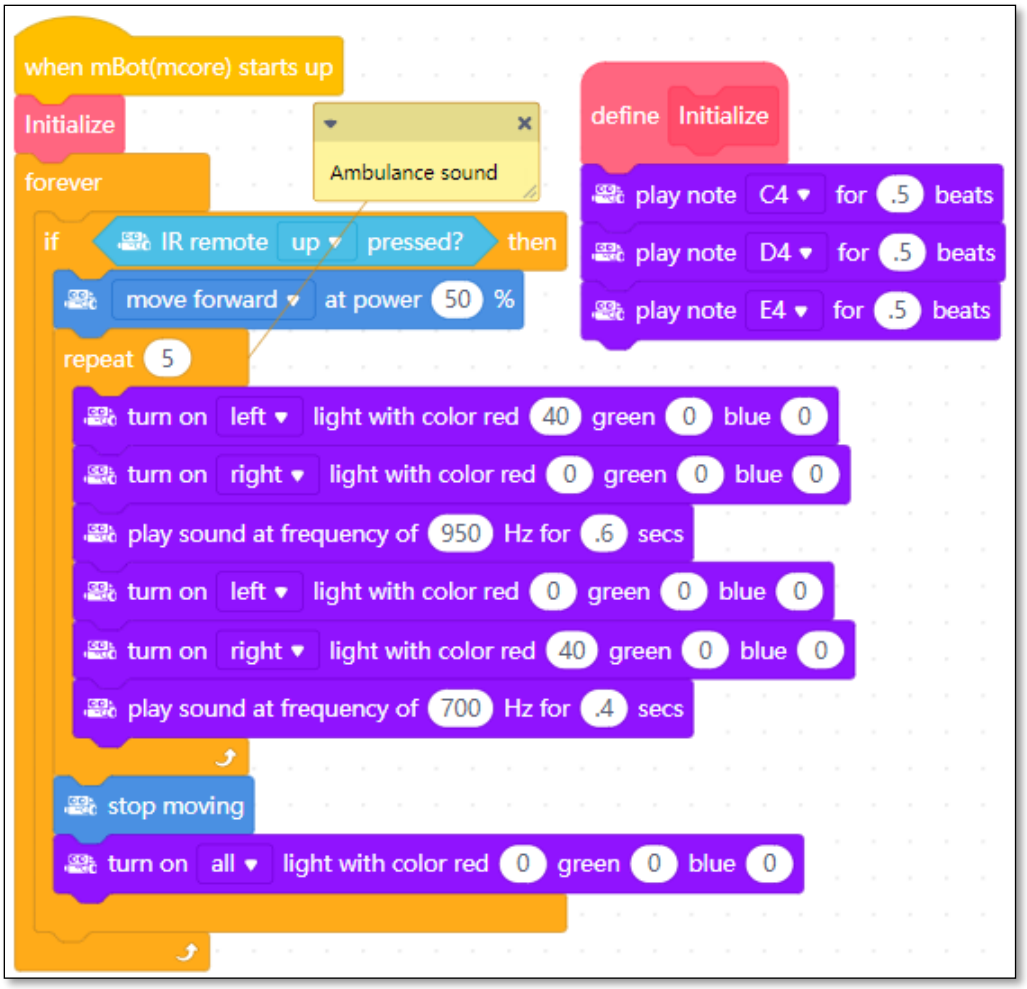
- The program will run when you press the up arrow on the IR remote.
- The program will play the ambulance sound 5 times while moving forward, then wait until the remote button is pressed again.

Tutorial Assignment

We are going to create an Initialize block to let the world know when the mBot is ready for action.

1. Start mBlock. Save the program as **Ambulance**.
2. Go to **My Blocks**, Click **Make a Block**.
3. Name the Block **Initialize**. Click **OK**.
4. This places **define Initialize** on the programming area.

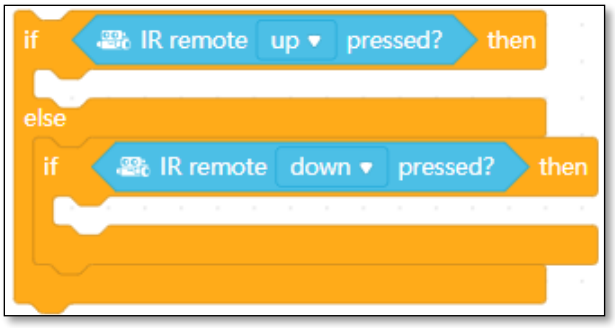
5. Go to **My Blocks**, **drag Initialize** as shown in the program.
6. Complete and test the program as shown.



Assignment

Start with your tutorial project and add the following.

1. Create your own Initialize block to introduce your mBot to the world, be creative!
2. In the Ambulance program, replace the **if then** block with an **if then else** block. Put an **if then** block in the else part of the **if then else** block. They should stack as shown below.



3. The program will keep checking each condition until one of them is true, then execute the interior code block.
4. Right Click on the code in the first if then block: choose **duplicate**.
5. Drag the copy of the code block to the second **if then** block.
6. Assign the down arrow to the **if** condition.
7. Change the code in the second block to run the mBot backwards when the down arrow on the remote is pressed.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Oh, Susannah

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Description

The following music program plays a part of Oh, Susannah, a common folk song.

Understanding

Demonstrate understanding of:

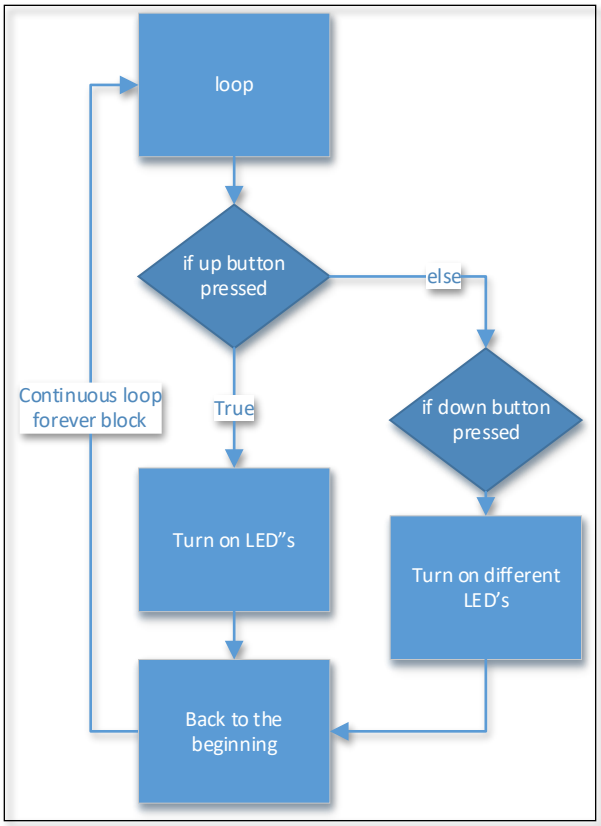
Debounce, playing music, if then else decisions

Knowledge Point

Debounce: When an IR remote button is pressed, it may make contact a couple of times. By putting in a pause with a wait block, you eliminate the bouncing effect of the key. This is called debouncing. Try the program without the wait and see what it does.

If Then Else

If then else extends our decision making. If then else is mutually exclusive. This means that only one of the choices or conditions can be true. You can stack multiple if else statements together.

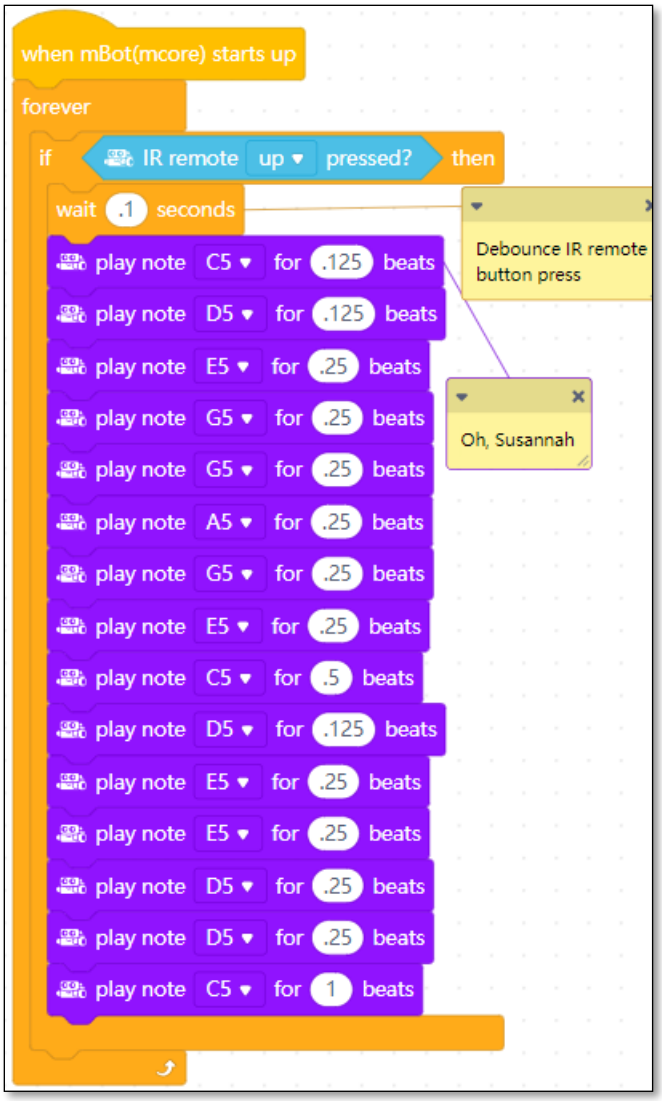


Requirements

- The program will run when you press the up arrow on the IR remote.
- The program will play a part of Oh, Susannah, then wait until the remote button is pressed again.

Tutorial Assignment

1. Start mBlock. Save the program as **Oh, Susannah**.
2. Complete and test the programs as pictured with the requirements listed.



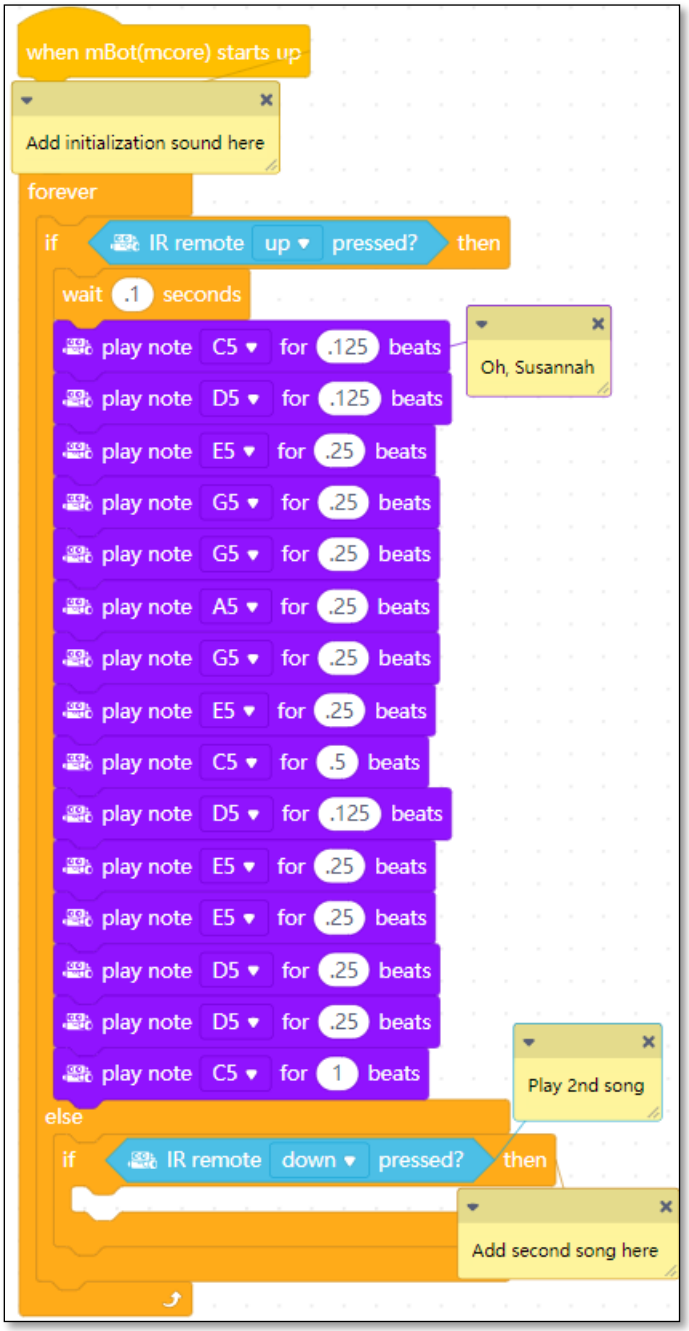
Assignment

Start with your tutorial project and add the following.

- Add an Initialize block with a short sequence of notes to the beginning of the program before the forever loop. This snippet of music will play automatically to indicate the program has been initialized and the mBot is ready to go.
- Add the Initialize block where the example program says **Add initialization sound here.**
- Add movements in between the notes. The mBot can move and play at the same time. Don't put the movements too close together, every 5 notes or more.

- Add another **if then else** control structures to your program as shown in the example. Use if else to finish the control block. See the example where it says Play 2nd song and Add 2nd song here.
- Add another part of a song of your choosing to where the screenshot says **Add second song here**. (It doesn't have to be a complete song, just a short piece of it) or make up a short little song. There is plenty of sheet music available on the web with notes and names of notes to help you figure out a different song. Please don't use a program that someone else created, create your own program.
- Here is a web site to get you started with a known song. https://noobnotes.net

Example of how to add another song.



Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.

- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Simple Remote Control

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Description

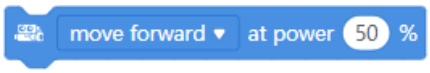
The IR remote control can be programmed to control the mBot's movement and other program commands. This project will get you started with remote control movement.

Understanding

Demonstrate understanding of:

IR remote, remote control, movement blocks

Knowledge Point

	The forward-moving rotation speed is -100%~100%. A positive number represents moving forward, and a negative one moving backward. 0 speed represents stop. The bigger the number is, the quicker it moves. In the tutorial example the speed is set to 50%. Do not use a speed of less than 30%, or the motor will stall.

What if the rotation speed for moving forward is beyond 100%?

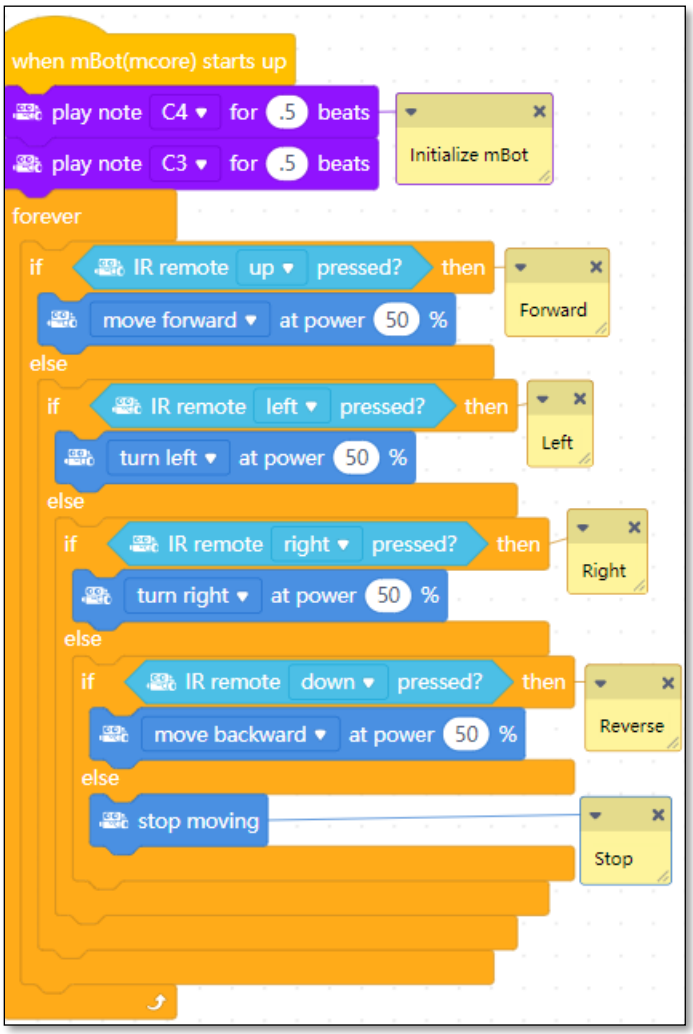
If the speed is set beyond 100%, it will be automatically set as 100% which is the maximum speed.

Requirements

- The robot will move in the direction of the arrow keys on the remote, then stop when the keys are released.
- The nested if then else decision structure only allows one key to be accessed at a time.

Tutorial Assignment

1. Start mBlock. Save the program as **Simple Remote Control**.
2. Complete and test the program as pictured with the requirements listed.



Assignment

Start with your tutorial project and add the following.

- Use LED's to indicate direction and movement.
- Use very short sounds to indicate changes in direction.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.

- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Get Started with MakeBlock

Time required: 45 minutes

The MakeBlock app is available for Android and iOS. You can remote control the mBot and create programs.

NOTE: The Factory Default Program must be loaded on the mBot for MakeBlock to work. This program is the only way to enable Bluetooth communication with the mBot.

POWER: The MakeBlock app requires power to be entered differently. 0 is stop, 255 is maximum. 127 in MakeBlock would be the same at 50% in the mBlock application.

Watch Video: Get Started with MakeBlock

Load the Factory Default Program

Whenever you load a new program, the factory default program is erased. This procedure reloads the factory default program if needed. You need to reload the factory default program to use the Bluetooth connection.

1. Run **mBlock**.
2. Power on the robot.
3. Connect the robot to a USB port on your computer with the USB cable.
4. Select **Connect**.
5. Click **Setting → Update Firmware**.
6. Select **Firmware Version → Factory Firmware**.
7. Click **Updates.** The default program loads.

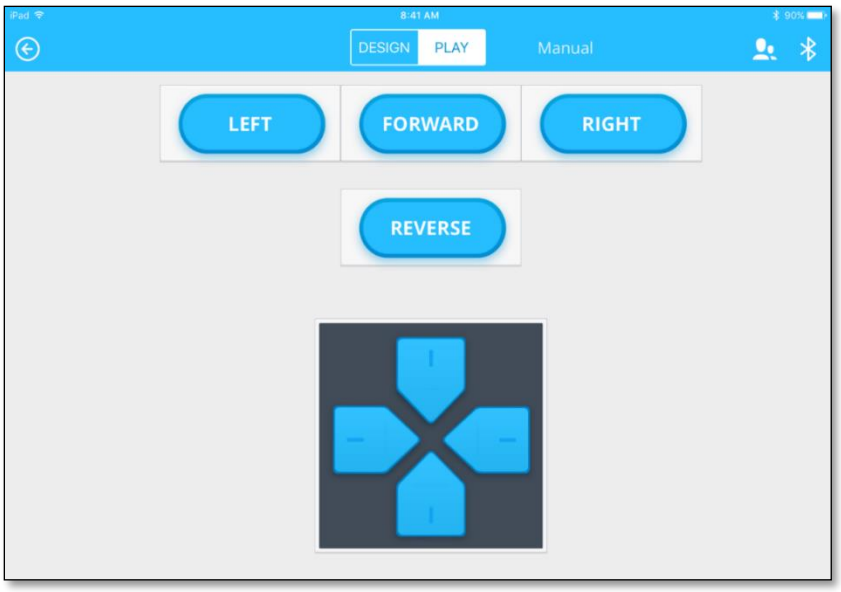
Tutorial Assignment Requirements

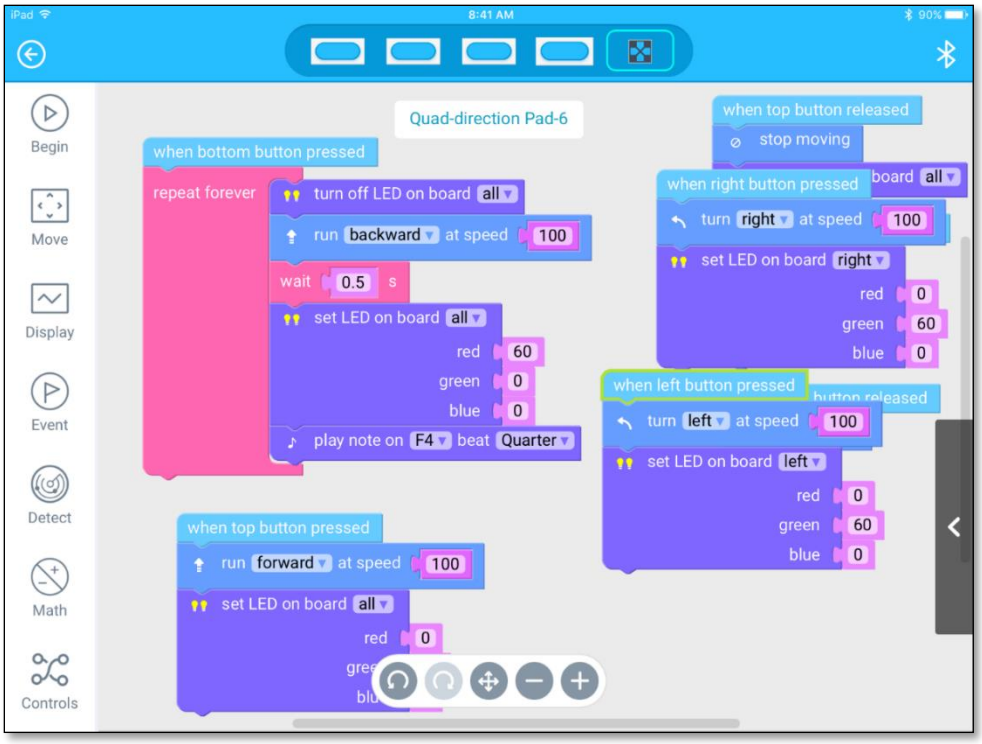
- Download and install the **MakeBlock** app from Google Play or iTunes.
- Pair your portable device with your robot.
- Use **Play** to remote control your robot with the pre-built app.
- Go to **Create** to create a basic app as pictured that makes your robot go forward, backward, right and left. The app is just a sample, as long as your Quad-Direction control and Buttons work, that is good.
- Use 4 buttons and the Quad-Direction control from the Custom category.

- Tap on the buttons and Quad-Direction controls to bring up the code and naming view.

Assignment Submission

- Attach a screenshot from your device of your interface and code for the program.
- The assignment is demonstrated in class or a link to a YouTube video recording showing your robot going through its motions is placed in the submission area in BlackBoard.





Robot Soccer

It's time for some fun with robot soccer!!! Use MakeBlock on your phone to control your robot in a game of soccer.

Assignment Submission

The assignment is demonstrated in class or a link to a YouTube video recording showing your robot going through its motions is placed in the submission area in BlackBoard.

Remote Control with Backup Sounds

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Description

This program simulates the backup sound of a dump truck or payloader.

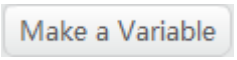
Understanding

Demonstrate understanding of:

variables

Knowledge Points

Define a Variable

Go to **Variables**. Click 

to create a variable. Enter the name of the variable and create it. 4 blocks relating to it will appear automatically.

The first two blocks are used to define the variable value and the varied value of the variable. The other two blocks are rarely used.

Use of Variables

Variables are values that change when a program runs. A variable has a name and a value.

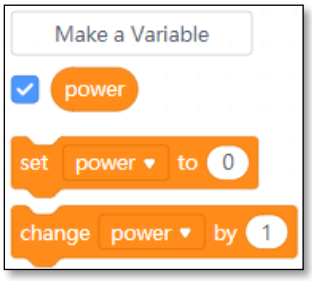
In mBlock, we can imagine that a variable is a box containing data. A program can store, change, and extract the data within the box. The name of the box is "variable name" and the data in the box is "variable value". When you create a variable, the box is empty. When you set a variable value, you fill the box with data. When you change the value, you take the data out and put in new data.

This assignment introduces variables. By creating a variable named power, you can change the power everywhere in the program by changing the set power block. If you set power to 60, everywhere that the power variable is used, is set to 60.

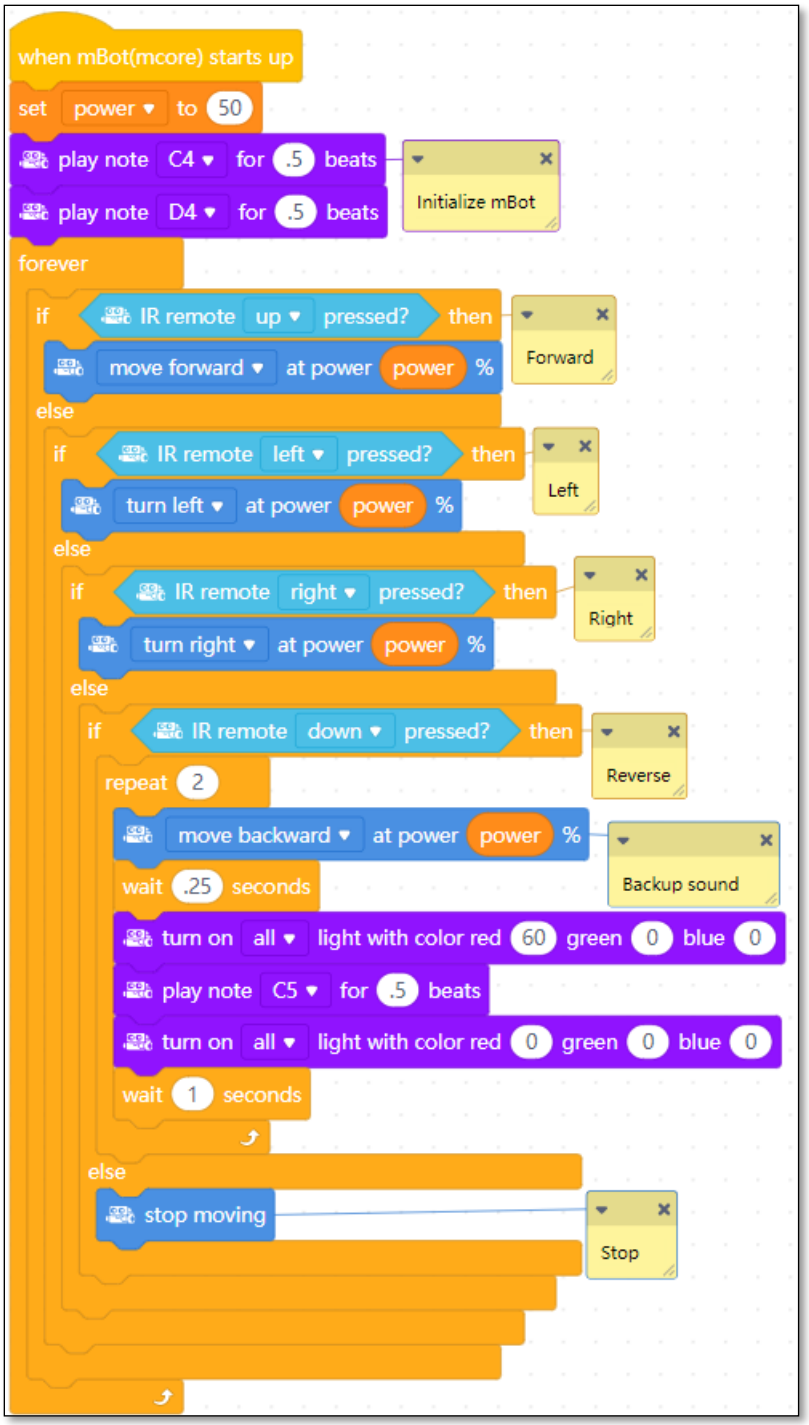
Requirements

- The robot will move in the direction of the arrow keys on the remote, then stop when the keys are released.

Tutorial Assignment



1. Start mBlock. Save the program as **Remote Control with Backup Sounds**.
2. Go to **Variables →** Click **Make a Variable.**
3. Name the variable **power**.
4. Drag the variable blocks as shown in the program.
5. Complete and test the program as pictured with the requirements listed.



Assignment

Start with your tutorial project and add the following.

1. Add a variation to the Backup Sound to your Remote Control with Backup Sounds program.

2. Add LED lights to indicate direction and movement.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Simple Obstacle Avoidance (Stay Away from Me!)

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

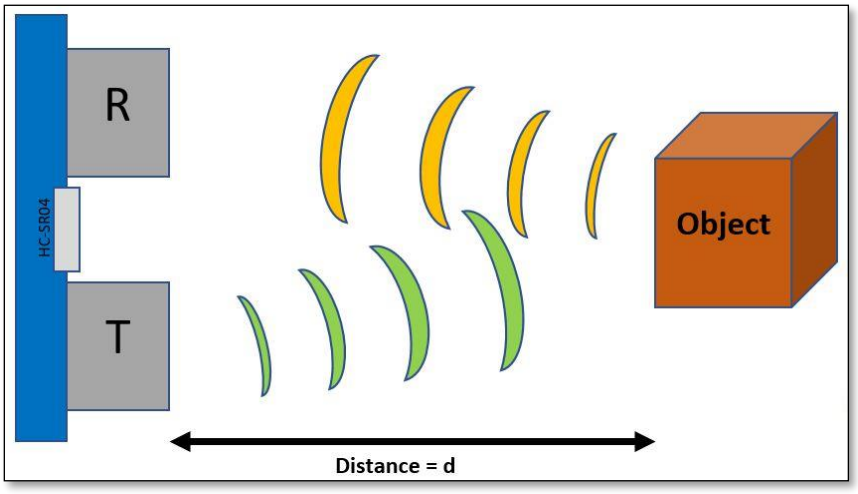
Understanding

Demonstrate understanding of:

Ultrasonic sensor, relational operators

Knowledge Points

The robot has an ultrasonic sensor on the front that detects objects and how far away they are. One "eye" emits ultrasonic sound waves while the other receives the signals bounced back. The distance is calculated based on how long it takes for the sound to return, much like a sonar.



Relational Operators

Relational operators test for true or false by comparing one value to another. In this program we will compare the distance the sensor detects to the distance that we have set.

Operator	Interpretation	Examples	Result
>	Greater than	9 < 10 10 < 10	true false
>=	Greater than or equal to	10 >= 10 10 >= 11	true false
<	Less than	9 < 10 10 < 10	true false
<=	Less than or equal to	10 <= 10 10 <=-9	true true
==	Equal to	9 == 9	true
!=	Not equal to	9 != 9	false

Program Description

The mBot Ultrasonic Sensor has a range from 3 cm to 400 cm, with a 30-degree angle of detection.

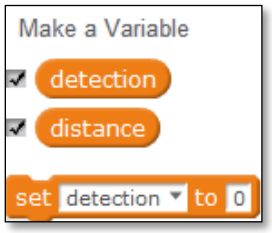
This program changes LED colors based on the distance of an object. The ultrasonic sensor assigns its reading to the distance variable. The distance variable is compared to the detection variable. When the object is within detection distance, the red LED's lights up, otherwise the LED's are blue.

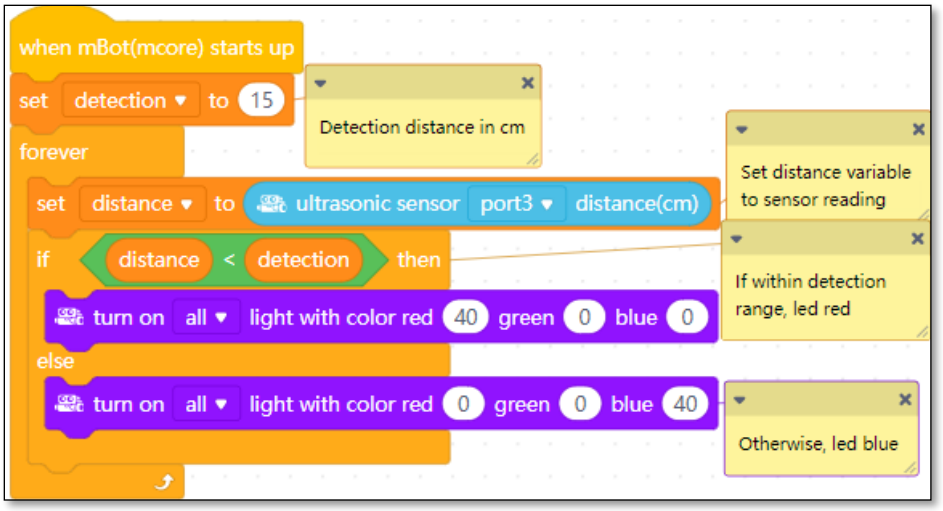
Requirements

- When an object gets within 15 cm (6"), the LED's change from Green to Red.
- Check the accuracy of the sensor with a ruler.

Tutorial Assignment

1. Start mBlock. Save the program as **Simple Obstacle Avoidance**.
2. Go to **Variables**, and create a variable called **detection** and one called **distance**.
3. This program introduces comparison operators, which are in the **Operators** category.
4. Complete and test the program as shown.



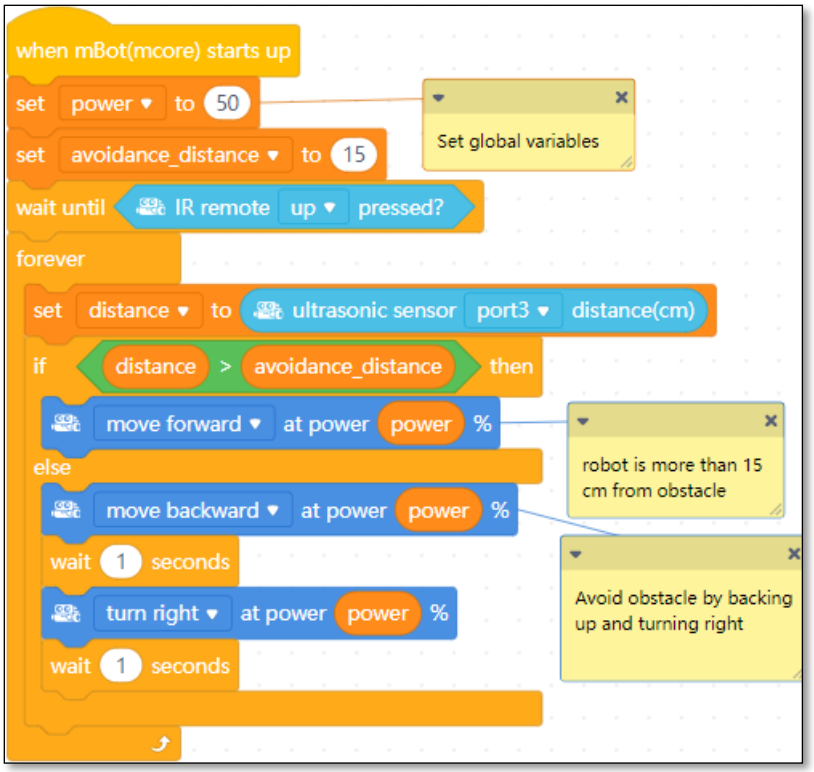


Requirements

- The robot detects an object within 15 cm, backs up, turns right, and moves forward.
- Test obstacle avoidance with your foot.
- Test the robot with an obstacle course of boxes or something similar.

Tutorial Part 2

1. Modify your program as shown for real obstacle avoidance.



Assignment

Start with your tutorial project and add the following.

- Add a sound and/or lights when an object is detected.
- Change the avoidance movement when an object is detected.
- Experiment with the detection distance.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Keep Away

Time required: 60 minutes

- Keep Away Tutorial Video
- Keep Away Assignment Video

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Understanding

Demonstrate understanding of:

Ultrasonic sensor, relational operators

Knowledge Points

An ultrasonic sensor can detect the distance from the object in front of it. A critical value is the distance between the object in front and mBot's ultrasonic sensor can be defined as the threshold to determine whether mBot should move forward (a threshold is a value of the condition under which an object is changed, which is also called critical value).

repeat until Block

The repeat until block is another type of loop. In other programming languages this is called a while loop. This loops keeps going until a condition is met. In the example program, the keep away part of the program will continue to repeat until the set button is pressed. The program exits the loop. The mBot stops moving.

In this example the value of the ultrasonic sensor indicates the distance between mBot and the object in front of it. Given the threshold of 15cm, mBot will keep moving forward until its distance from the object is less than 15cm; the mBot will stop immediately when its distance from the object is less than 15cm.

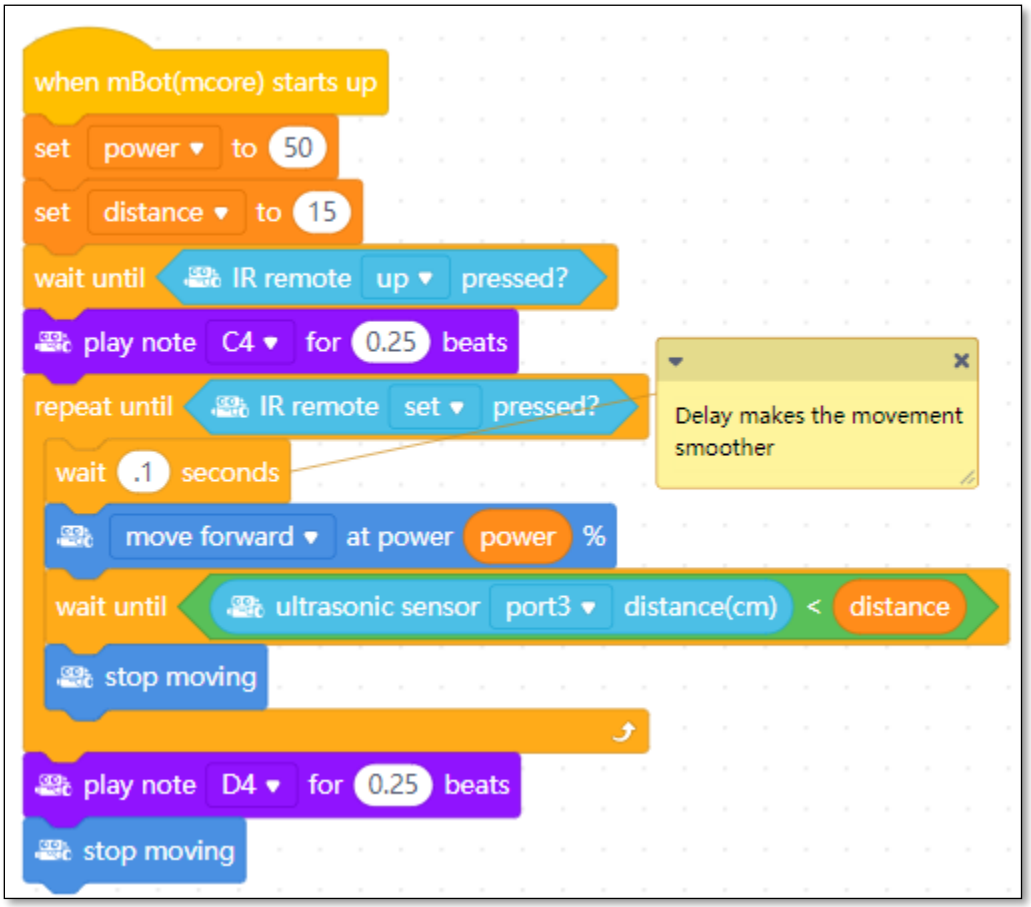
Requirements

- The robot detects an object within 15 cm and stops.
- When the object is moved, the mBot starts moving forward.

- Test the keep away with your foot.

Tutorial Assignment

1. Start mBlock. Save the program as **Keep Away**.
2. Complete and test the program as pictured with the requirements listed.



Assignment

Start with your tutorial project and add the following.

- Add lights when an object is detected.
- Add a sound when an object is detected. Make it very short. You will want to replace the wait with the sound.
- Make the robot also move backwards if the barrier is moved closer to the mBot.

NOTE: An if else statement will allow your robot to go two different directions based on the distance detected. You will want one distance for moving backwards, another distance for moving forward.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Light Sensor (Nightime Dance Party!)

Time required: 60 minutes

Introduction of the Light Sensor

Sensors are used to detect events or changes in the environment and send information to the electronic components of other electronic devices. While the program is running and debugging, it is often required to collect real-time sensor values to help us understand the environment light, sound, distance and other information.

Light sensor value range: 0~1000, exposed under sunshine (> 500), evening (0 ~ 100), lighting (100 to 500).

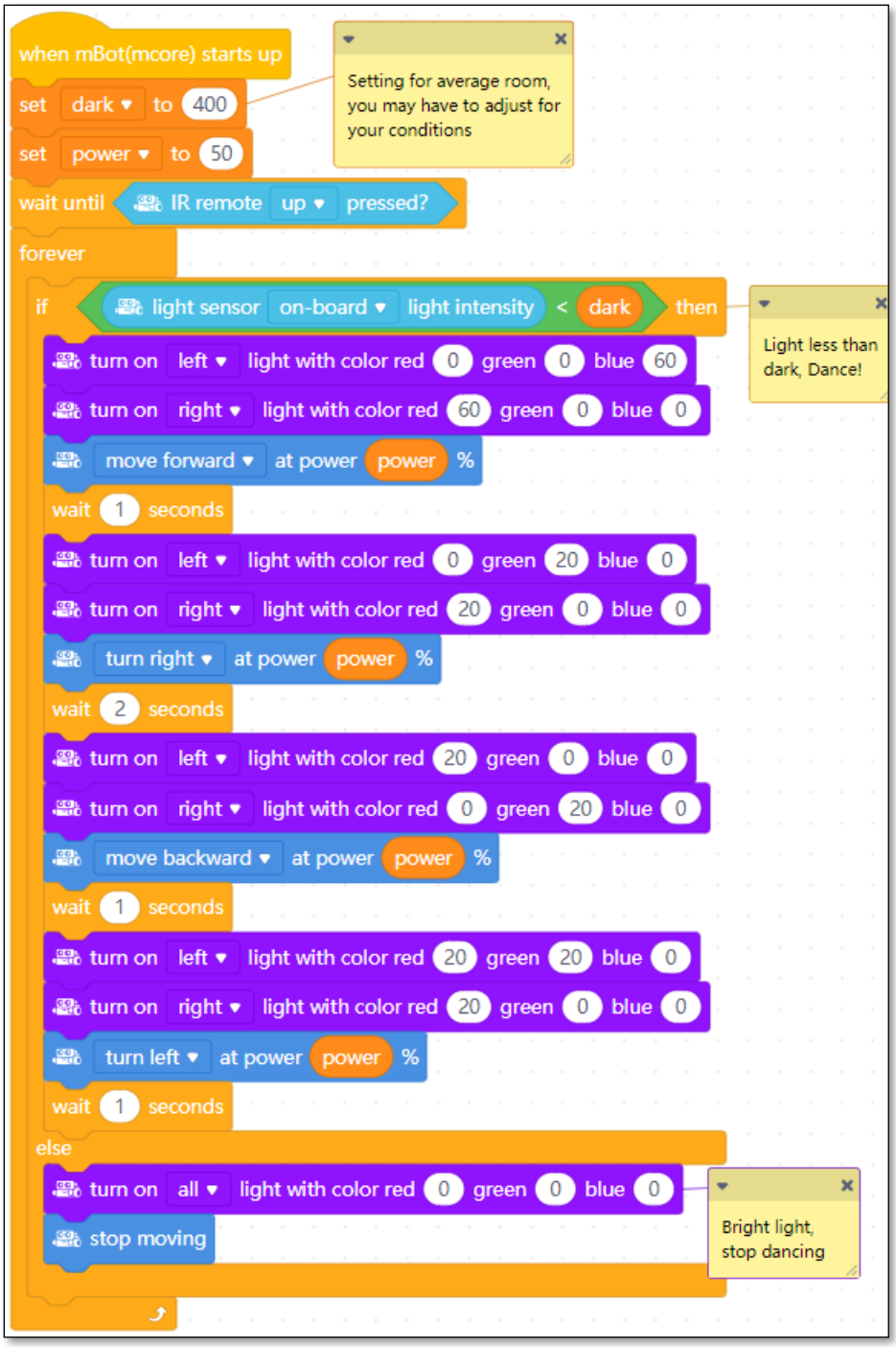
The following program will give you a robot bedtime dance party!

Tutorial Assignment

1. Start mBlock. Save the program as **Light Sensor (Bedtime Dance Party!)**.
2. Complete and test the program as pictured with the requirements listed.

Requirements

- The program will run when you press the robot's remote control.
- There will be variety in movement, sights, and sounds.
- The dance will start when the lights go out. Stop when the lights turn on.



Assignment

Start with your tutorial project and add the following.

- Get creative and create your own version of Bedtime Dance Party!

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Fire Engine

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Understanding

Demonstrate understanding of:

loops

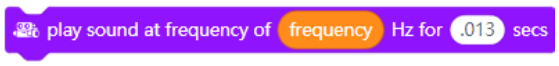
Knowledge Points

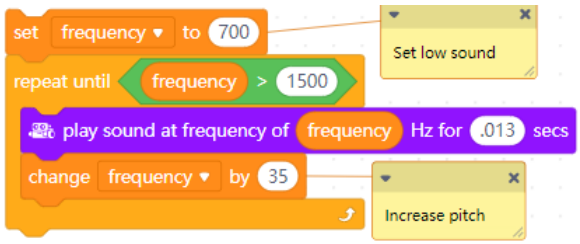
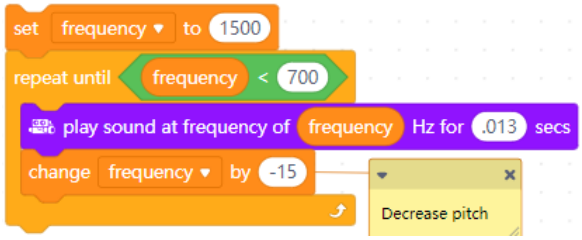
This program simulates the sound of a fire engine siren.

Fire Engine Sire Simulation

The fire engine siren's low frequency sound is between 650Hz and 750Hz, and its high frequency sound is between 1450Hz and 1550Hz. The siren sound is generated by repeating the following pattern: the low frequency sound amplifies to a high frequency sound in 1.5 seconds, and then drops back to the lower frequency in 3.5 seconds. Therefore, the fire engine siren sounds can be programmatically simulated as follows:

Set the low frequency to be 700Hz, then set the high frequency sound to be 1500HZ, repeatedly playing the buzzer in a range from 700Hz to 1500Hz and then back to 700Hz. The ratio of amplification time to the drop time is 1.5:3.5, which is 3:7, so the ratio of frequency amplification to the drop needs to be 7:3. By tuning the sound time and amplification vs. drop's amplitude, the fire engine siren is simulated.

	Define variable frequency as the frequency of a tone and set the sounding duration as 12ms (12 here is for example only, and you can set it to any value you think proper).
	Set the lower limit of variable frequency as 700 and the upper limit of frequency as 1500.

	This block indicates repeated run of the blocks within it until the condition  is established. When the condition is established, it will break the loop and run the following blocks.
	This block means: set the initial value of tone frequency as 700Hz and sound it, and then increase the frequency by 35Hz each time and sound it until above 1500Hz, where it breaks the loop and the frequency will not be increased anymore.
	This block means: set the initial value of tone frequency as 1500Hz and sound it, and then decrease the frequency by 15Hz each time and sound it until below 700Hz, where it breaks the loop and the frequency will not be decreased anymore.

Why "repeat it until frequency > 1500", not "frequency = 1500"?

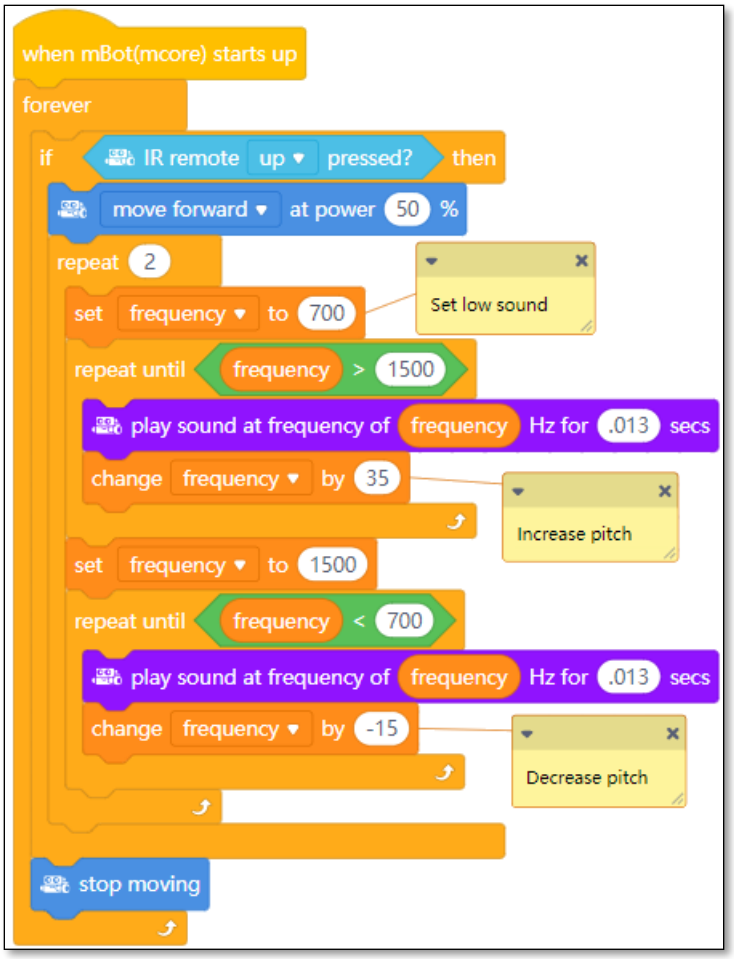
That is because in the example of simulating the fire engine sound effects, it is hard to define the sound frequency each time when it is increased from 700Hz to 1500Hz and what increment it should be each time the frequency is increased. If we set frequency=1500, the final frequency should reach 1500 so that it can break the loop, or the frequency will be increased again and again, making it hard to debug. We use frequency >1500, and when the frequency is above 1500, the loop will be broken and the following program decreasing the frequency will be executed.

Tutorial Assignment

1. Start mBlock. Save the program as **Fire Engine**.
2. Complete and test the program as pictured with the requirements listed.

Requirements

- The program runs as shown.



Assignment

Start with your tutorial project and add the following.

Keep the fire siren. Add a police car sound or another siren sound. How you switch between the two sirens is up to you. It can be automatic or controlled by the IR remote.

How to simulate police car sound effects: Low-frequency sound is set between 650Hz and 750Hz and high-frequency sound between 1450Hz to 1550Hz. It takes 230 ms to raise a low-frequency sound to a high-frequency one and then 100 ms to lower a high-frequency sound to a low-frequency one.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.

- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Light Sensor Blackout Warning with Blocks (Nighttime Dance Party 2)

Time required: 60 minutes

Please read the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Understanding

Demonstrate understanding of:

code blocks, light sensor

Knowledge Points

We will do some math to determine the normal light in a room when the program first runs. If the light level falls below 80% of the room illumination, the robot sounds an alarm!

Requirements

- The program will run when you power on the robot.
- When the lights are off, the robot does a dance.
- When the lights are on, the robot is quiet.

Tutorial Assignment

1. Start mBlock. open **Nighttime Dance Party**, save the program as Nighttime **Dance Party 2**.
2. To make a code block, go to **My Blocks**, Click **Make a Block**.
3. Name the Block **Initialize**. Click **OK**.
4. **define Initialize** is placed on the programming area.
5. Go to **My Blocks**, **drag Initialize** where you want to run it.

6. See the 3rd example.

7. Add and integrate the program pictured to the Bedtime Dance Party program.

   a. Create the new code.

   b. Break apart your dance party, and put it into the new blocks.

   c. The idea is to break the code into blocks, rather than one long string of code.

8. Test the program.


when mBot(mcore) starts up
wait until IR remote up pressed?
Initialize
repeat until IR remote set pressed?
CheckForBlackout
turn on all light with color red 0 green 0 blue 0
stop moving
define Initialize
set accumulator to 0
Initialize the value of the variable
repeat 10
set lightIntensity to light sensor on-board light intensity
Store sensor reading in variable
set accumulator to accumulator + lightIntensity
Accumulate the readings
set average to accumulator / 10
Calculate average reading
set lightThreshold to average * .8
Set variable to 80% of ambient light
play note C4 for 0.25 beats
play note D4 for 0.25 beats
play note E4 for 0.25 beats
define CheckForBlackout
If room gets darker
if light sensor on-board light intensity < lightThreshold then
move forward at power 50 %
turn on all light with color red 0 green 60 blue 0
play note C4 for 0.25 beats
play note D4 for 0.25 beats
Light, sound and movement
else
stop moving
Stop when room is light
turn on all light with color red 0 green 0 blue 0

Assignment

Start with your tutorial project and add the following.

- Change the notes, movement, etc.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Separate Motor Control

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Understanding

Demonstrate understanding of:

Separate motor control, variables

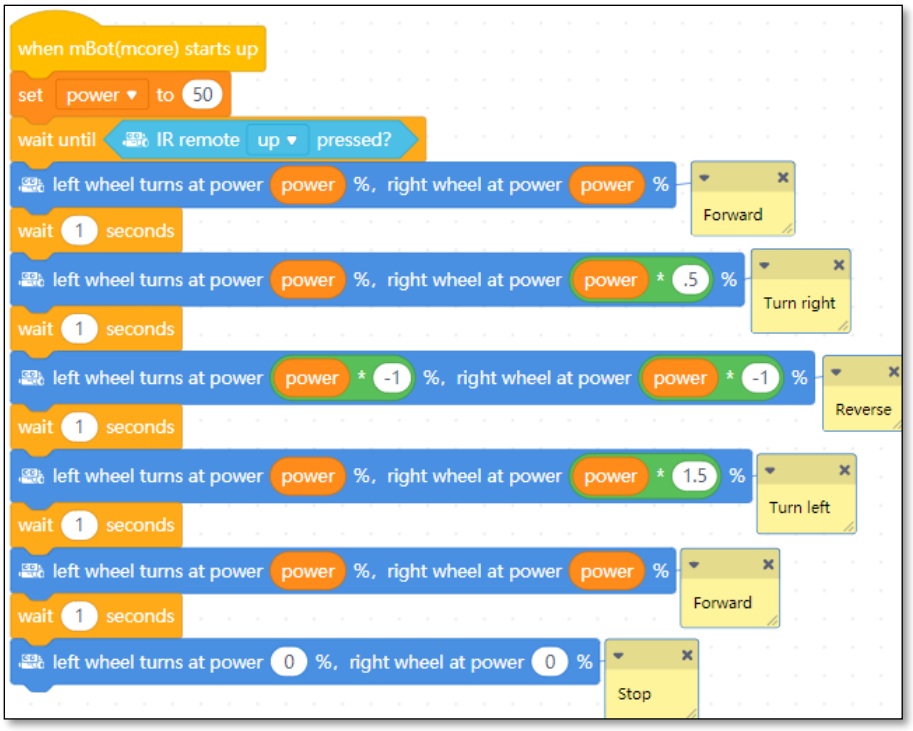
The following example controls the motors separately. By varying the power to each motor, the mBot can make different types of turns.

Requirements

- The program will run when you press the robot's remote-control button.
- Turn like a CAR.

Tutorial Assignment

1. Start mBlock. Save the program as **Separate Motor Control**.
2. Complete and test the program as pictured with the requirements listed.



Assignment

Start with your tutorial project and add the following.

- Add car differential turns as shown in the figure.
- Add tank turns as shown in the figure.

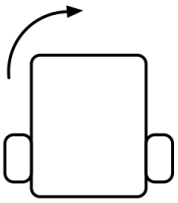


Figure 2: Car

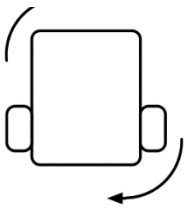


Figure 2: Tank

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Simple Line Following (What's My Line?)

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Understanding

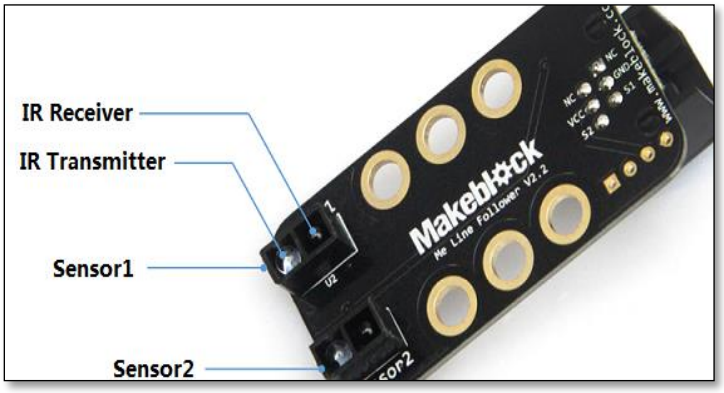
Demonstrate understanding of:

Line-follower sensor, if then else

Knowledge Points

Principles of the Line-follower Sensor

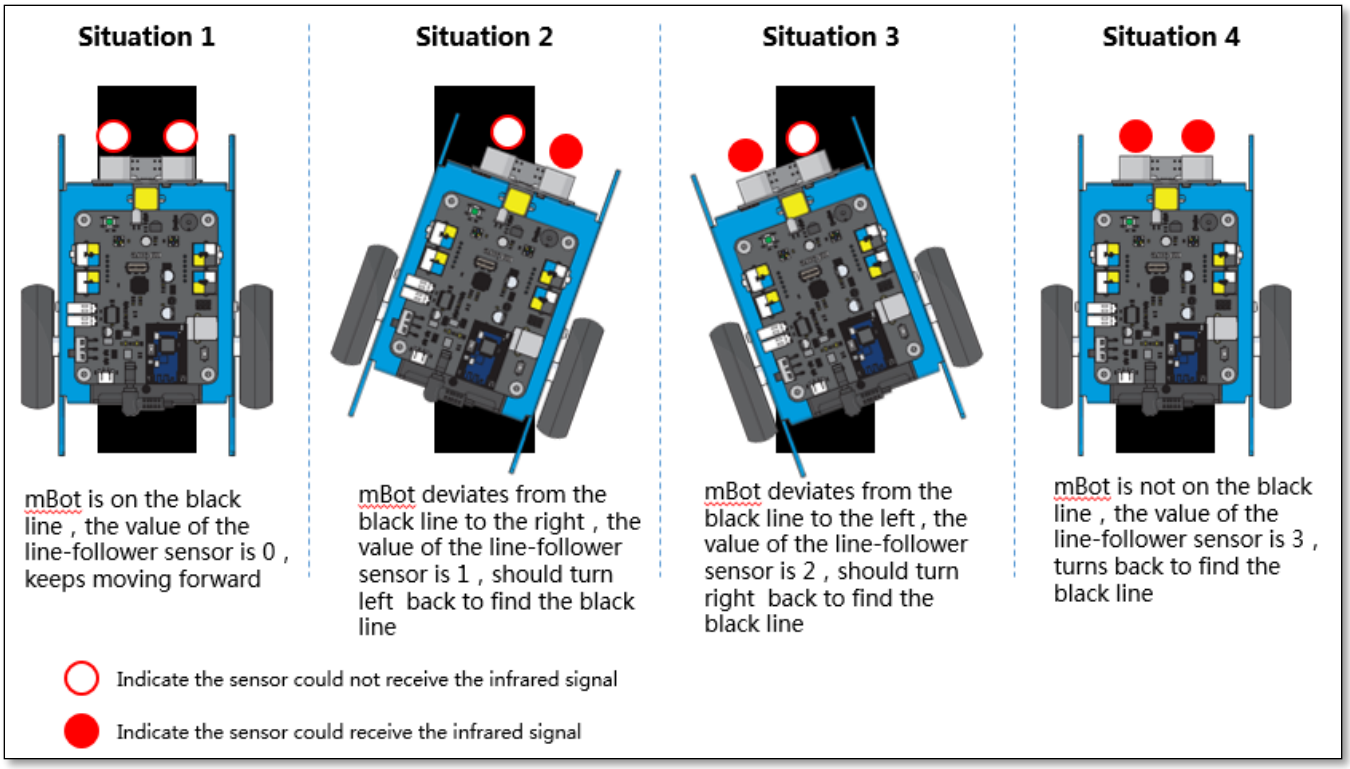
The line-follower sensor is below the robot (see the attached diagram), which consists of two sensors, Sensor 1 and 2, each consisting of an infrared emitter and an infrared receiver (see the attached diagram). As it is often used to keep the robot moving straight, it is called a line-follower sensor. Its detection range is 1 to 2 cm.



The infrared emitter continually emits infrared light during the mBot moving:

- If the infrared light is reflected (encountering white or other light color surfaces), the receiver receives the infrared signal and output the value 1 (now you can see the blue LED on the back of the line-follower sensor is lighted);
- If the infrared light is absorbed or cannot be reflected, the receiver will not receive the infrared signal but output the value 0.

The mBot line-follower sensors can detect a white line on a black surface, or a black line on a white surface.



- **Situation 1:** Line follower = 0. Both sensors detect a line indicated by both blue lights shutting off.
- **Situation 2:** Line follower = 1. The right sensor no longer detects a line indicated by the right blue light turning on. In order to get the mBot back on the line, therefore, we turn the mBot left until both sensors are activated and the mBot continues moving forward.
- **Situation 3:** Line follower = 2. The left sensor no longer detects a line indicated by the left blue light turning on. We turn the mBot right until both sensors are activated and the mBot continues moving forward again.
- **Situation 4:** Line follower = 3. Both sensors no longer detect a line. Run backward until the robot detects a line.

Sensor Position	**Value**
Both sensors over the line	0
Right sensor off line	1
Left sensor off line	2

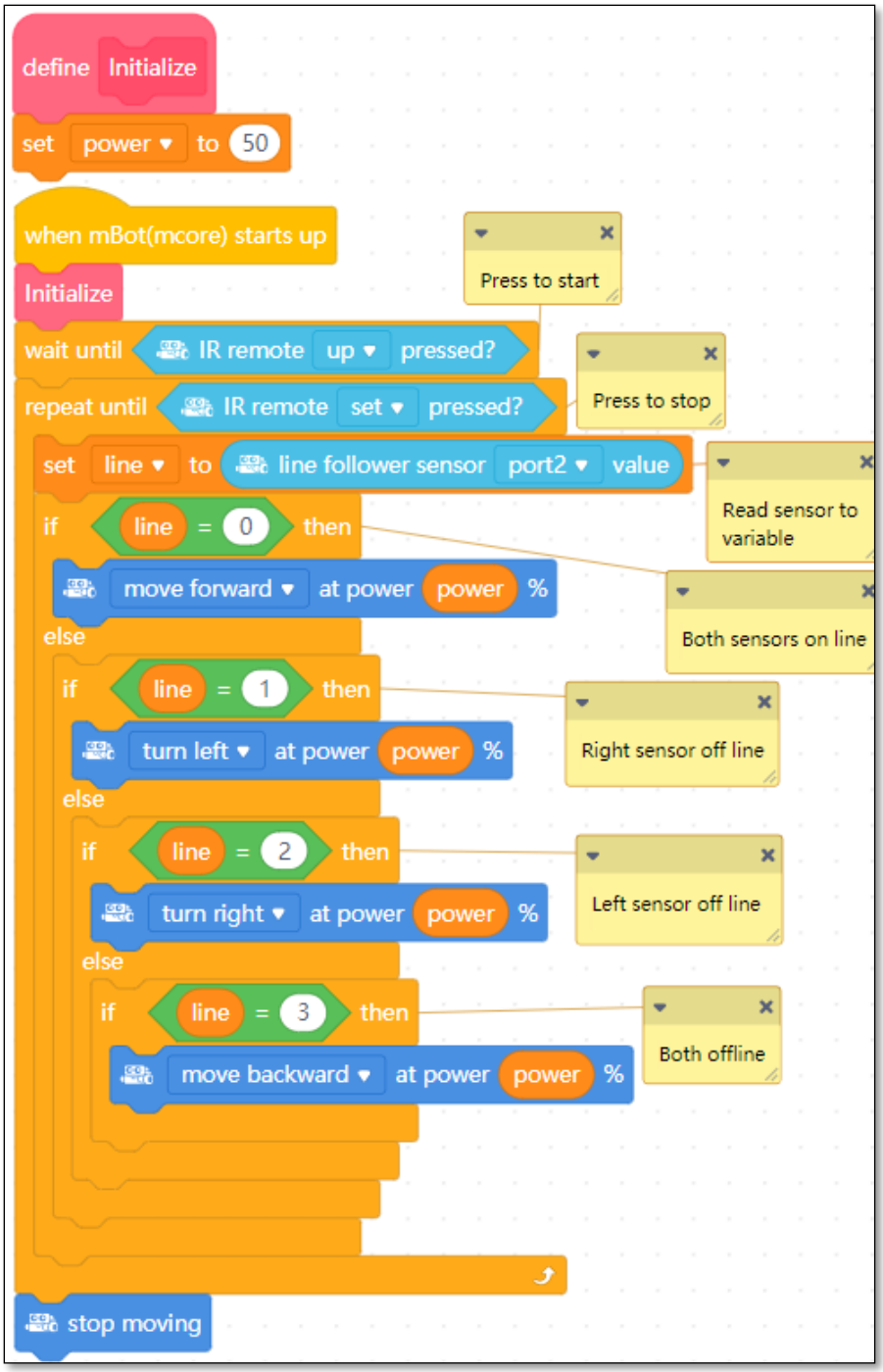
Both sensors off line	3

Line Follower Track: A paper track came with the mBot in the box. A line follower track can also be made with foam board and black tape. Automotive cloth wiring harness, electrical tape, or duct tape works well.

Tutorial

Create the follow **Simple Line Following** program.

Test with the paper line following track that came with your mBot, or make your own.



Assignment

1. Add LED's to the program to indicate direction.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.

- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Move Along an S-shaped Track

Time required: 60 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Understanding

Demonstrate understanding of:

Using Code Blocks

In this example the module block 

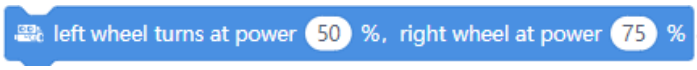
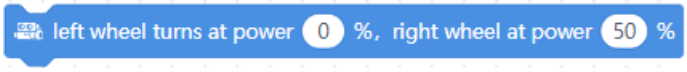
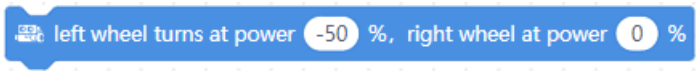
is defined to let the mBot move in a S-shaped track.

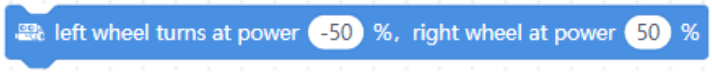
Define the module directive to combine several blocks. The program executes the defined module directive, i.e. calling its defined block behavior. This can make the program simpler and easier to read.

Use "Differential Speed" to Control mBot's Turns

When the right wheel is faster than the left one, mBot turns left; when the left wheel speed is faster than the right one, mBot turns right.

Watch mBot's turning by testing the following scripts.

Script	Type of Turning Motion
	Differential turn, turning left while moving forward
	Car turn, turning left on the left wheel
	Car turn, turning left on the right wheel, the left wheel goes backwards

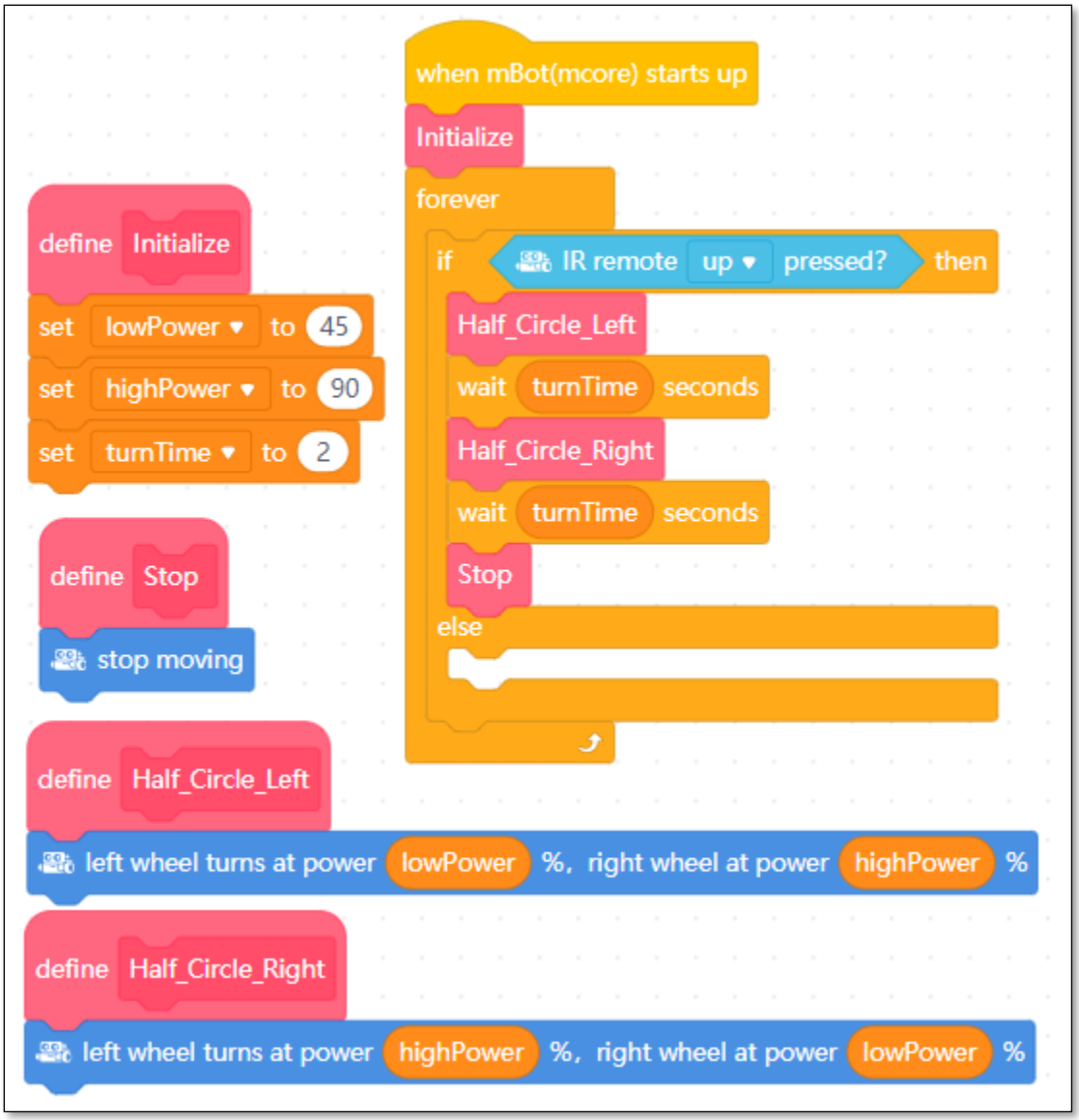
	Tank turn, turning left in place.

Requirements

- Use differential turning to trace out the letter S.
- The program will trace out an S shaped track when you press the robot's remote-control button.

Tutorial Assignment

1. Start mBlock. Save the program as **S-shaped Track**.
2. Complete and test the program as pictured with the requirements listed.
3. The program uses two half circles to create an S. One is a left turn half circle, the other is a right turn half circle.
4. Change the turnTime variable to make an S.



Assignment

Start with your tutorial project and add the following.

- Move the robot in a circle. Use a different remote button to trigger this part of the program.
- Have the robot trace an infinity symbol. Create this shape with a different remote button press.
  Hint: This is two S's connected.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Smart Line Following (Follow the Black Ink Road)

Time required: 60 minutes

There are some tweaks that can be made to make the line following smoother. We are going to create a **turnLeft** block, and **turnRight** block to make the line following smoother. We will use a couple of variations of car turns instead of tank turns. Car turns are smoother.

Boolean (Flag) Variables

This program introduces the concept of Boolean (flag) variables. Boolean (flag) variables allow the mBot to keep track of something, to give it the ability to remember something. The **turningLeft** variable keeps track of the last direction the mBot turned.

- If the mBot turned left last, **turningLeft** is set to 1 or true
- If the mBot turned right last, **turningLeft** is set to 0 or false.
- If the mBot last turned left and lost the line, it will keep turning left until it finds the line.

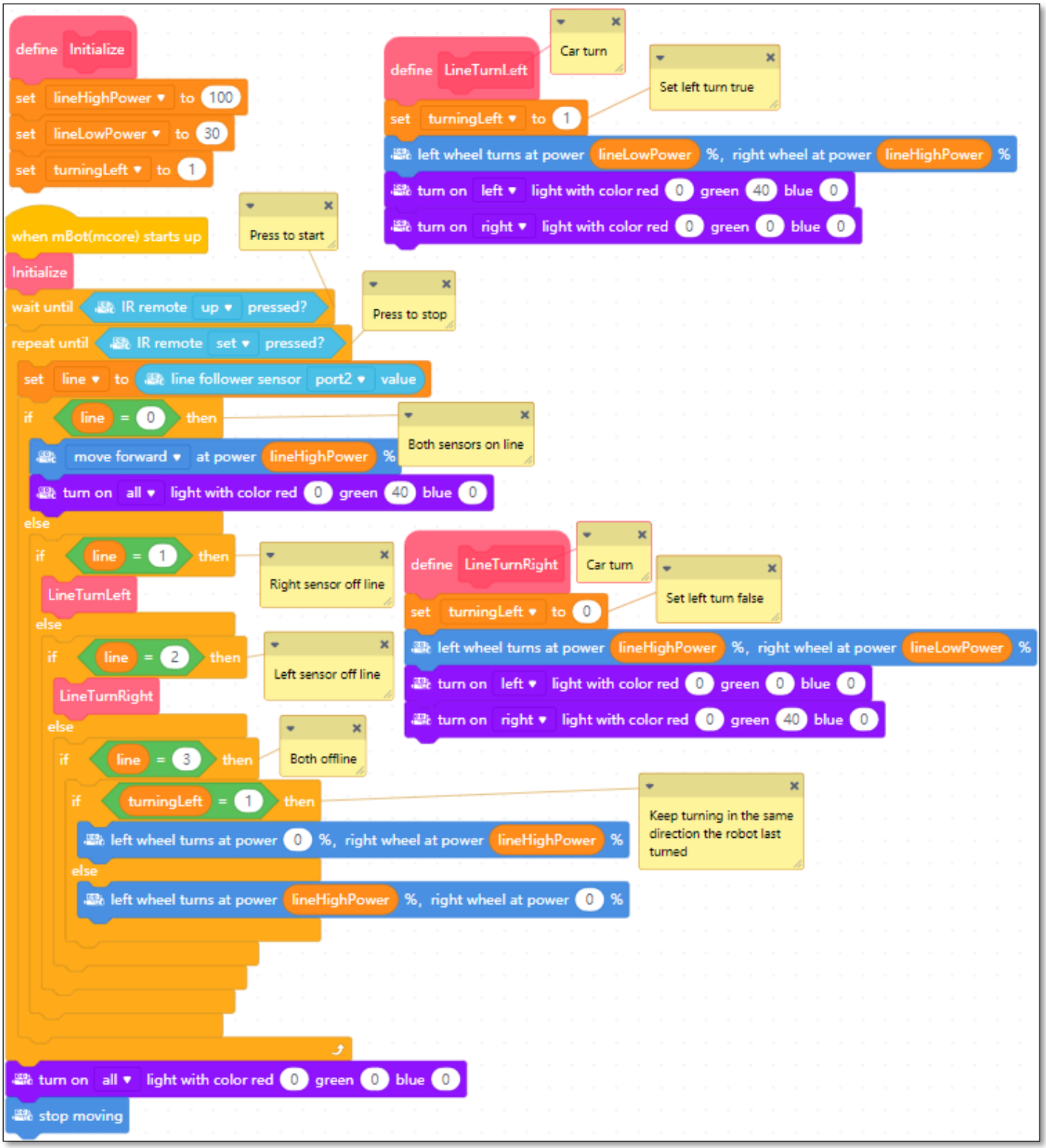
Remembering which direction it turned last gives the mBot a better chance of finding the line again.

Tutorial Assignment

1. Start mBlock, open Simple Line Following. Save the program as **Smart Line Following**.
2. Complete and test the program as pictured with the requirements listed.

Requirements

- Accurately follow the line (doesn't get off course)


define Initialize
set lineHighPower to 100
set lineLowPower to 30
set turningLeft to 1
when mBot(mcore) starts up
Initialize
wait until IR remote up pressed?
repeat until IR remote set pressed?
set line to line follower sensor port2 value
if line = 0 then
move forward at power lineHighPower %
turn on all light with color red 0 green 40 blue 0
else
if line = 1 then
LineTurnLeft
else
if line = 2 then
LineTurnRight
else
if line = 3 then
if turningLeft = 1 then
left wheel turns at power 0 %, right wheel at power lineHighPower %
else
left wheel turns at power lineHighPower %, right wheel at power 0 %
turn on all light with color red 0 green 0 blue 0
stop moving
Press to start
Press to stop
Both sensors on line
Right sensor off line
Left sensor off line
Both offline
Keep turning in the same direction the robot last turned
define LineTurnLeft
Car turn
Set left turn true
set turningLeft to 1
left wheel turns at power lineLowPower %, right wheel at power lineHighPower %
turn on left light with color red 0 green 40 blue 0
turn on right light with color red 0 green 0 blue 0
define LineTurnRight
Car turn
Set left turn false
set turningLeft to 0
left wheel turns at power lineHighPower %, right wheel at power lineLowPower %
turn on left light with color red 0 green 0 blue 0
turn on right light with color red 0 green 40 blue 0

Assignment

Start with your tutorial project and add the following.

- Change the LED's to indicate right and left turn and going straight ahead.
- Don't add notes while your mBot is line following. That slows down the sensor readings.
- Try to make the line following smoother and able to follow sharper turns. Don't completely stop the wheel in the car turn, reduce the power. Don't go below 50. Try reducing one side and increasing the other. Try speeding up when you are going straight.
- Time trials. Let's see who can go through the line following tracks with the best time!

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Stay Inside the Line

Time required: 60 minutes

- Stay Inside the Line Demo Video

The line following sensor can be used for other tasks besides following a line. The sensor detects dark or light. This can be used to keep the mBot inside a circle or other shape that is outlined in black. You can use your line following track or some sort of black tape to outline a shape. Automotive cloth wiring harness tape works well. You could also use a piece of paper or the back of the mBot paper track on a dark floor.

Problem

We want our mBot to detect black lines and avoid them.

The following table shows the different readings from the line sensor. Our logic for staying on the line is to provide actions to get back to 0, both sensors on the line. In this project, we want both sensors to be off the line. If the sensor reading is 3, we can avoid any black lines.

Sensor Position	Value
Both sensors over the line	0
Right sensor off the line	1
Left sensor off the line	2
Both sensors off the line	3

Algorithm

An algorithm is a step-by-step set of instructions for solving a problem or performing a task. It's like a recipe that guides you through a series of actions to achieve a specific goal.

Pseudocode

A simple way to outline the steps of an algorithm for a computer program without using specific programming language syntax. It's like a rough draft of a program's logic. Start with writing clear and concise English-like statements to describe what your code should do.

Pseudocode solution to problem.

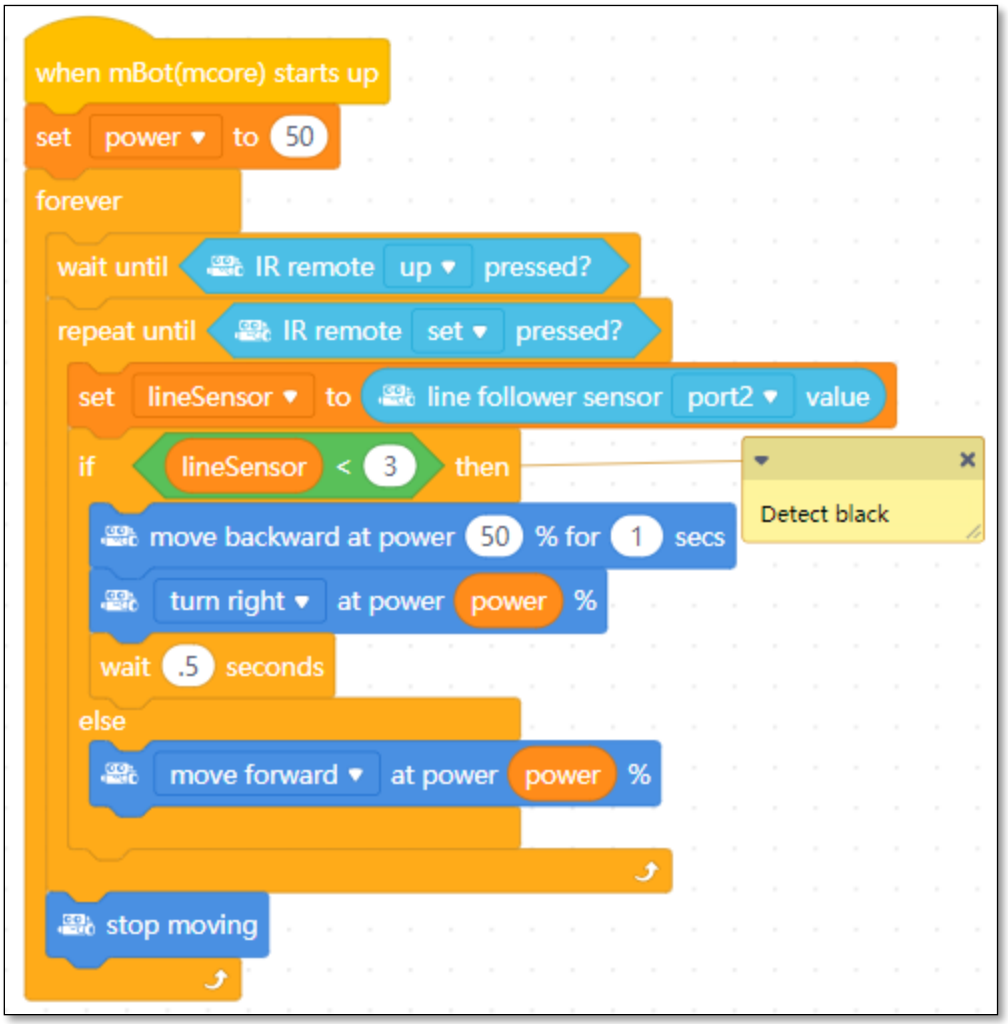
```
if linefollowing sensor reads 3
    robot is off the line, keep going
else if linefollowing sensor reads less than 3 (0, 1, 2)
    avoid obstacle
```

Requirements

- Stay inside the line.

Tutorial Assignment

1. Start mBlock. Save the program as **Stay Inside the Line**.
2. Complete and test the program as pictured with the requirements listed.



Assignment

Start with your tutorial project and add the following.

- Add a light and a short sound to indicate hitting the line.
- Randomly use a different maneuver to avoid the line.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Calibrate Distance and Square (Straight on 'til Morning)

Second star to the right and **straight on 'til morning**

Time required: 60 minutes

NOTE: The mBot is not an accurate robot. As the batteries discharge and the conditions change, it will behave differently. We can change power and time. Just try to get close. We will use this program to calibrate our mBot for future programs.

This program will allow us to calibrate our mBot for the following:

1. Adjust the **COMP** compensation factor percentage for the robot to go straight.
2. Distance by adjusting the **time** constant to drive 48”.
3. Adjust turns by adjusting **turnTime360** to have the mBot start and start in approximately the same orientation while making a square.

Debouncing

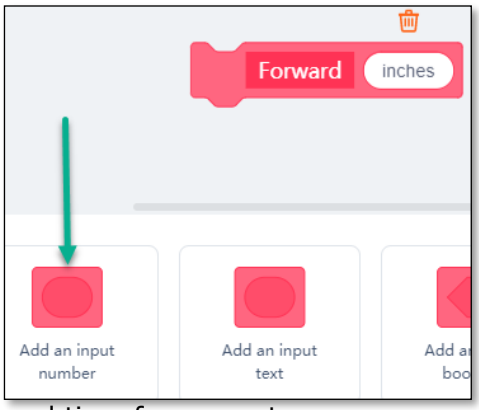
Putting in a short wait allows for more accuracy scanning for the remote signal and movement. A small wait while changing directions allows the robot to temporarily stop, then turn more.

When you press a button on the remote, it is impossible to press it once. It makes contact several times, bouncing off the contact. It may make contact on one side, then the other, then it settles down. The mBot scans for each ir code so fast that it can mistake one code for another. Debouncing is putting a tiny delay in between scanning for the ir codes to ensure smooth operation.

Code Blocks with Arguments/Parameters

Code blocks with arguments allow us to create reusable code that will behave differently depending on what we feed them. Arguments/parameters allow us to change the function of the block based on the information we pass to the block.

1. To make a code block, go to **My Blocks**, Click **Make a Block**.
2. Name the Block **Forward**.
3. Click Add an input number. Type **inches**. Click **OK**.
4. To use the block, go to **My Blocks**, drag the block to where you want to run it.



We will determine how much time it takes your mBot to move a certain distance at a certain power. We can input that distance and time for accurate movement.

avgSpeed (inches per second) = (Distance(inches) / Time)

Example: (48/7.4) = 6.5 inches per second

Tutorial Assignment

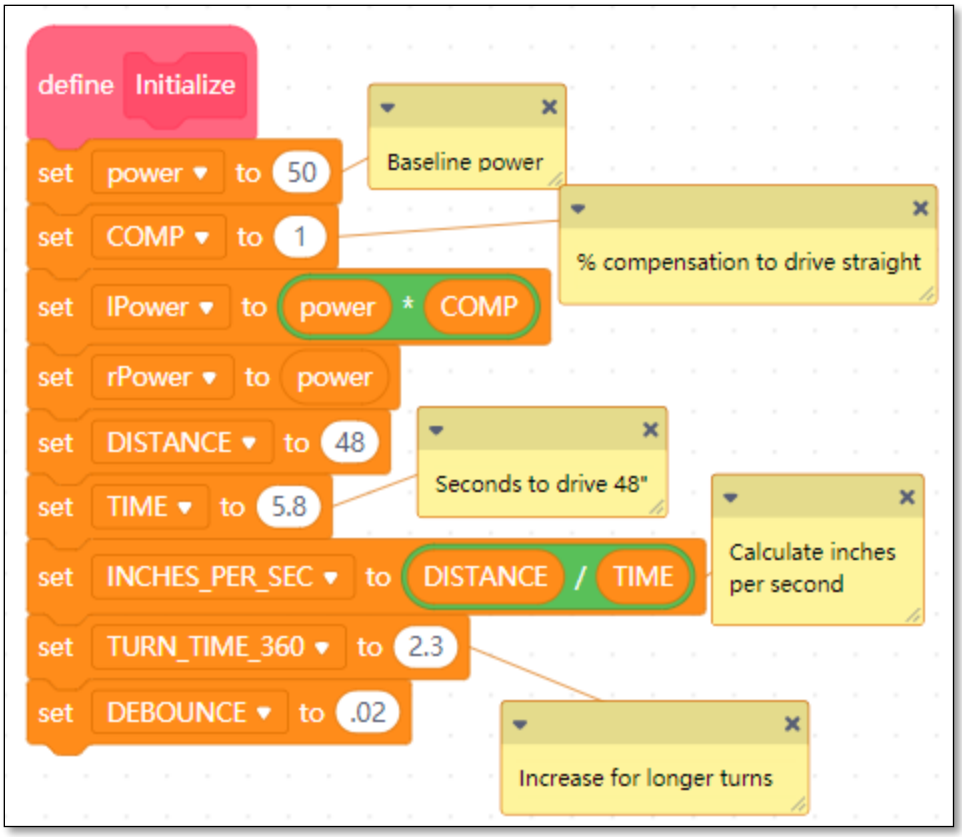
1. Start mBlock. Save the program as **Calibrate Distance and Square**.
2. Complete and test the sample program and add the requirements listed.

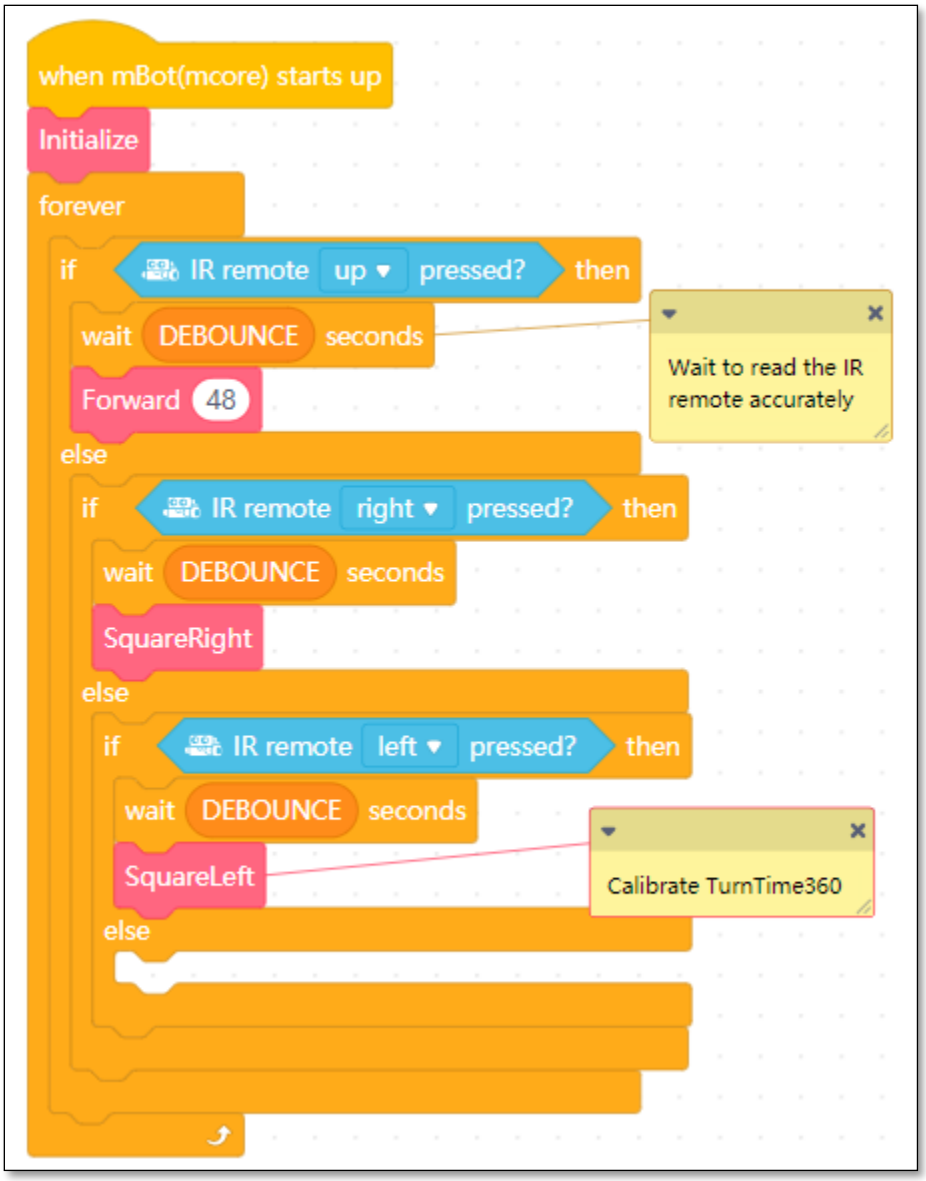
Requirements

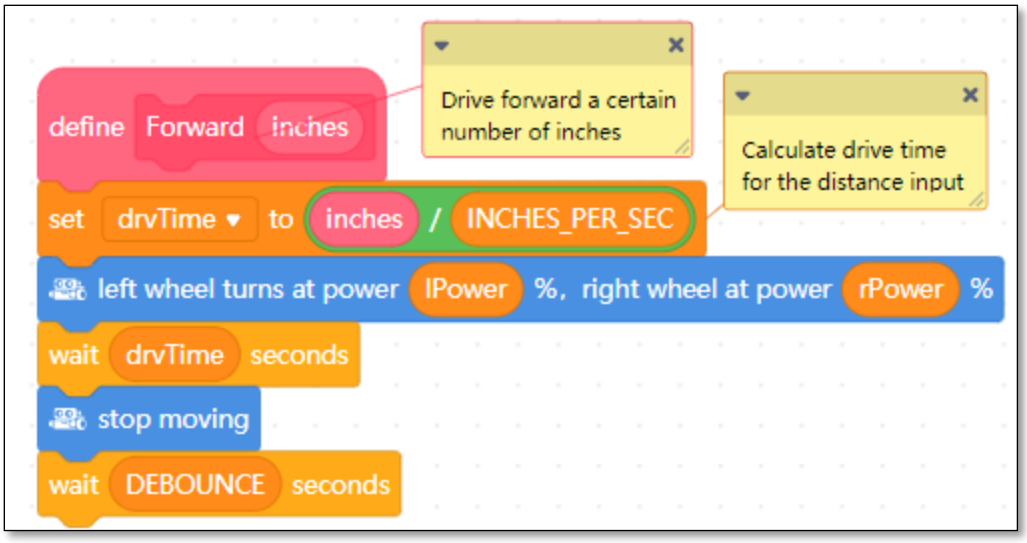
NOTE: The mBot will never be able to do these requirements perfectly. Just get close.

Create a program with the following blocks each triggered by a different remote button.

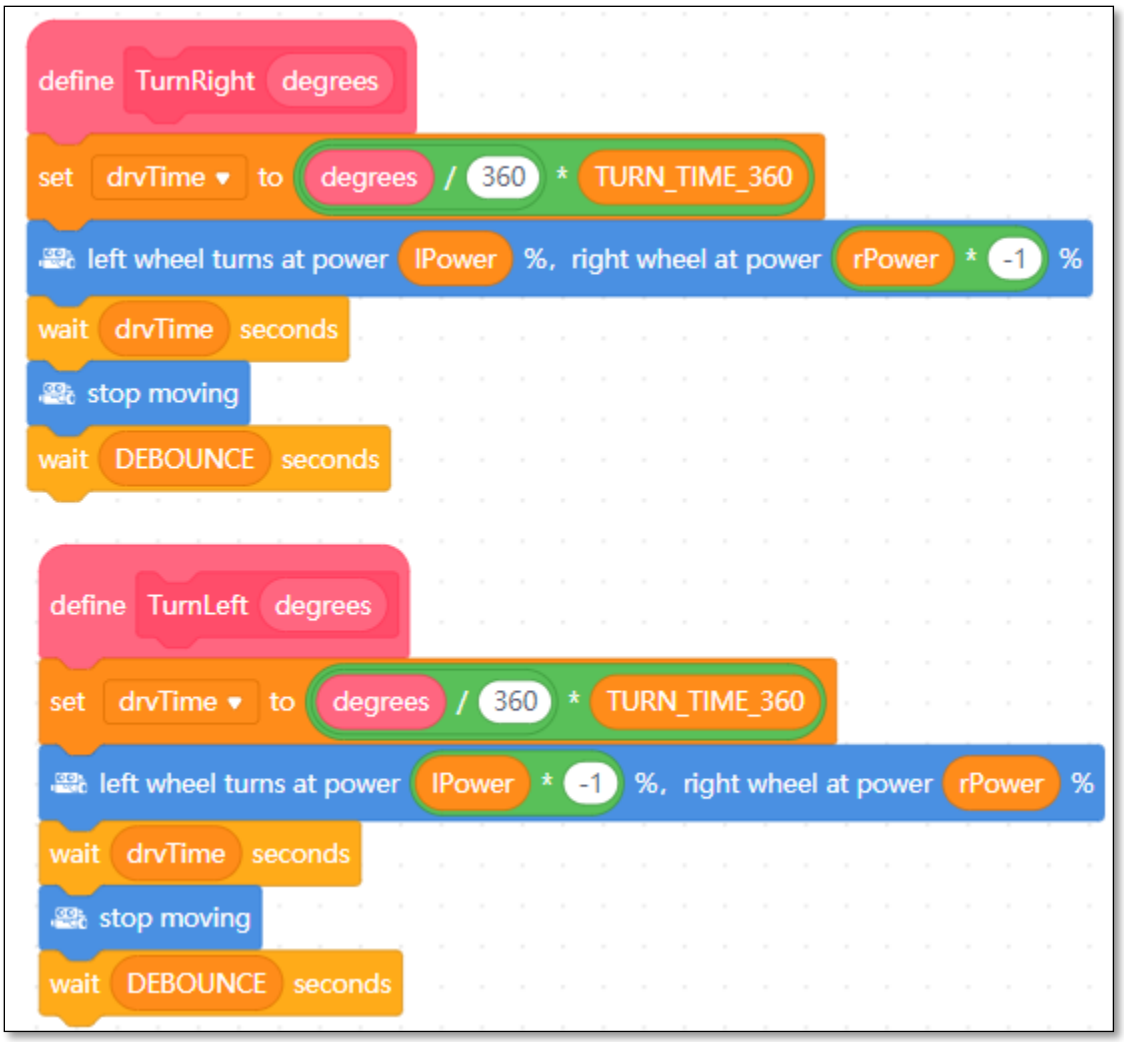
- A block that moves forward 48”.
    a. If your mBot turns to the left, adjust **COMP** to 1.05 to start. Keep adjusting the COMP until your mBot runs fairly straight.
    b. Adjust **time** until the robot moves forward 48”.
- A block that does a square turning to the right.
- A block that does a square turning to the left.
    a. Adjust the **TurnTime360** until the right square and left square are more or less the same.

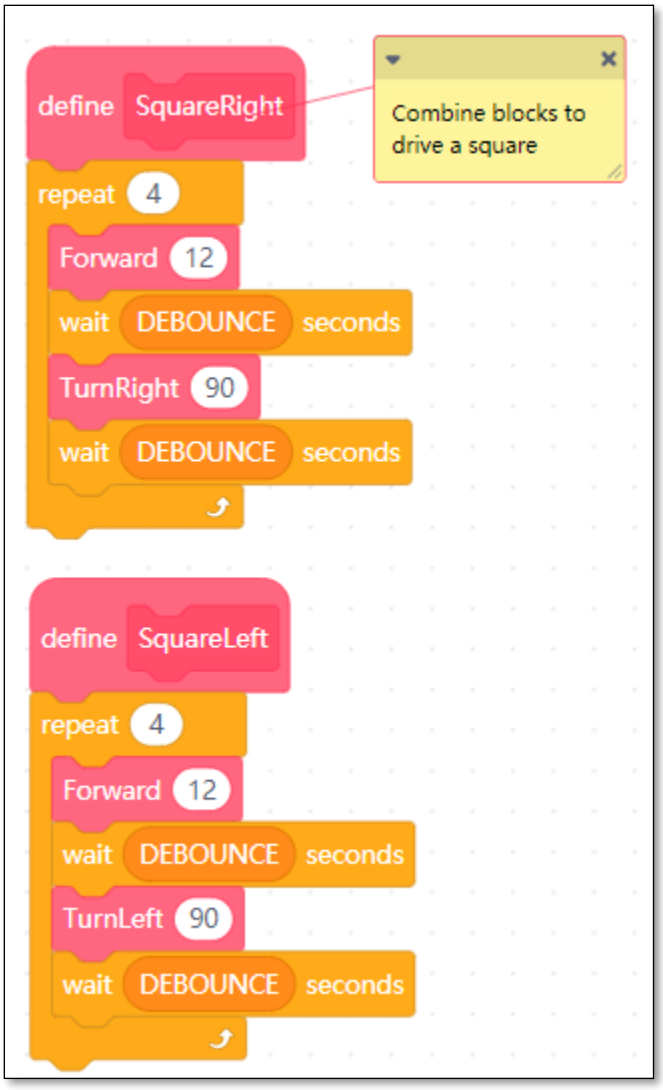

define Initialize
set power to 50
Baseline power
set COMP to 1
% compensation to drive straight
set IPower to power * COMP
set rPower to power
set DISTANCE to 48
set TIME to 5.8
Seconds to drive 48"
Calculate inches per second
set INCHES_PER_SEC to DISTANCE / TIME
set TURN_TIME_360 to 2.3
set DEBOUNCE to .02
Increase for longer turns


when mBot(mcore) starts up
Initialize
forever
if
IR remote
up
pressed?
then
wait
DEBOUNCE
seconds
Forward
48
Wait to read the IR remote accurately
else
if
IR remote
right
pressed?
then
wait
DEBOUNCE
seconds
SquareRight
else
if
IR remote
left
pressed?
then
wait
DEBOUNCE
seconds
SquareLeft
Calibrate TurnTime360
else



NOTE: Notice that in the set drvTime to: **degrees / 360** is grouped together.





Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Driving School

Time required: 120 minutes

NOTE: The mBot is not an accurate robot. As the batteries discharge and the conditions change, it will behave differently. We can change power and time. Just try to get close.

We can accurately move and turn. We will combine the movement and turning programs into one. There isn't an example program, it is up to you to figure it out.

Charge your batteries. Calibrate your robot.

It is time to put your mBot through its paces. Can you pass the driving tests?

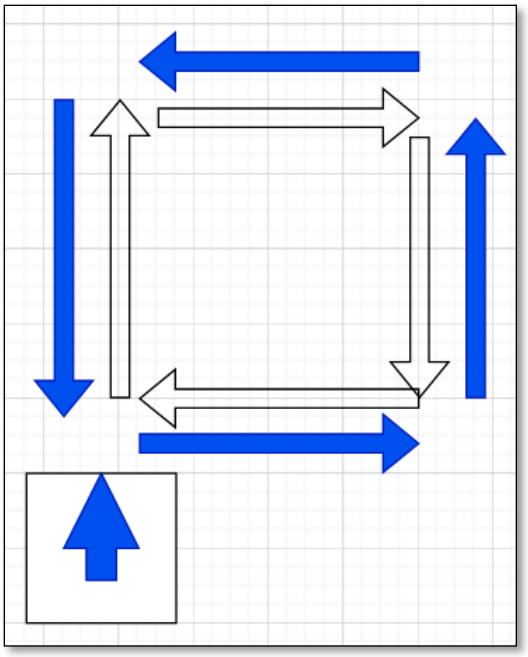
Assignment

- Complete each shape in a separate code block.
- Assign each block/shape to a different remote-control button. Use Simple Remote Control as an example.
  Hint: Use if else to stack your remote-control keys.
- Open **Calibrate Distance and Square** and save the program as **Driving School**.

Requirements

1. **Square** - your robot will trace the path of a square that is 1-foot square. It will start and end in the same place and the same orientation.
2. **Rectangle** - your robot will trace the path of a rectangle that is 1-foot x 2-foot. It will start and end in the same place and the same orientation.
3. **Sentry** - your robot will trace a 1-foot square around an object. Start the square one way, then turn around and go back the other way. Return to the beginning point and orientation.
   Sentry Demo Video
4. **Retrace** - move in a 1-foot square forward, and then move in reverse to retrace that same square backwards to the beginning point and orientation. One solution would be to build a Reverse block that uses negative numbers for motor movement.
   Retrace Demo Video

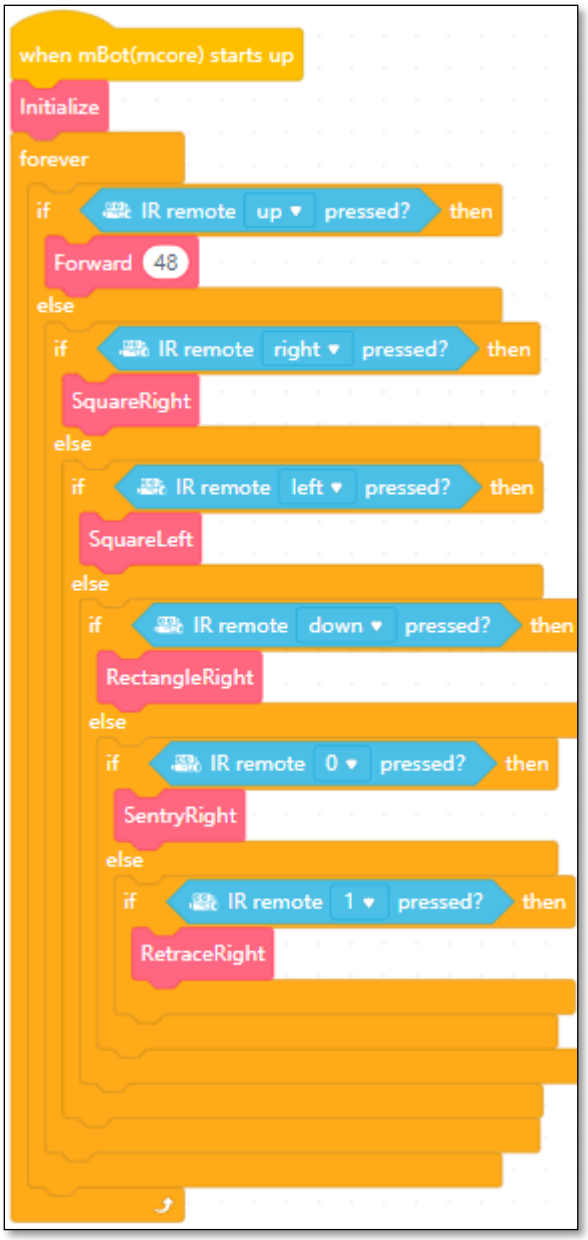
It is a good idea to sketch out on a piece of paper the turns and straight traveling you have to do.



Pseudocode Example Square

```
repeat 4 times
    forward 100
    right 90
```

This is what the main part of your program should look like. Each shape will have its own separate block. Try to build from the parts you already have to build the separate shape blocks.



Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Driving School Part 2

Time required: 120 minutes

NOTE: The mBot is not an accurate robot. As the batteries discharge and the conditions change, it will behave differently. We can only change is power and time. Just try to get close.

Charge your batteries. Calibrate your robot.

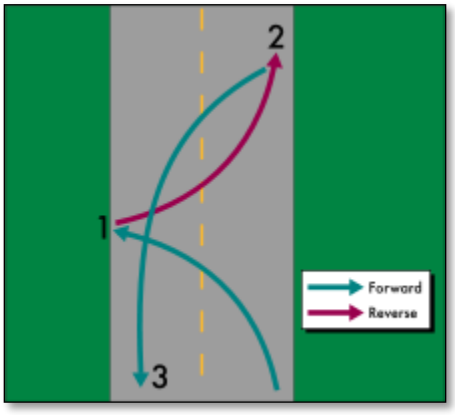
Assignment

- Complete each shape in a separate code block.
- Assign each block/shape to a different remote-control button. Use Simple Remote Control as an example.
  Hint: Use if else to stack your remote-control keys.
- Open the completed **Driving School** program.
- Save the program as **Driving School 2**

Requirements

1. **ForwardReverse** - Move forward 12”, turn 180°, move backwards 12” (which will be the same direction), turn 180° again, and then continue to move forward 12”. The robot should move in one direction but do part of the trip moving backwards.
   ForwardReverse Video Demo
2. **Octagon** - Move your robot in a 12” octagon. Each turn is a 45° angle. Start and end in the same place and the same orientation.
   Octagon Video Demo
3. **Equilateral Triangle** – Move your robot in a 12” equilateral triangle. Start and end in the same place and the same orientation. An equilateral triangle has an inside angle of 60 degrees. Subtract that from 180 degrees to find out how far the robot should turn for each side.
   Equilateral Triangle Video Demo
4. **5-Point Star** – Teach your robot to trace a 5-point 12” star. Start and end at the same location and orientation. Look up the inside angle and subtract from 180 degrees.
   5-Point Star Video Demo

5. **3-PointTurn** - Using 3 or more turns, teach your robot how to make a 3-point turn, like a regular car. You don't have to do curves, you can use straight angles if you wish.



Optional Challenges

1. Modify the program to trace the outline of a 12" pentagon.
2. Modify the program to trace the outline of a 12" hexagon.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Obstacle Avoidance with Warning and Random Turns (Look Out!)

Time required: 60 minutes

Tutorial Assignment

1. Start mBlock. Save the program as **Obstacle Avoidance with Warning and Random Turns**.
2. Complete and test the program as pictured with the requirements listed.

Requirements

- The robot randomly turns left or right to avoid an obstacle and gives a visual warning.
- Test obstacle avoidance with your foot.


when mBot(mcore) starts up
Initialize
wait until IR remote up pressed?
repeat until IR remote set pressed?
turn on all light with color red 60 green 0 blue 0
set distance to ultrasonic sensor port3 distance(cm)
if distance < 40 then
turn on all light with color red 20 green 20 blue 0
if distance < 15 then
turn on all light with color red 60 green 0 blue 0
move backward at power 50 % for 1 secs
if pick random 1 to 100 < 50 then
turn on left light with color red 0 green 0 blue 0
turn on right light with color red 0 green 60 blue 0
turn right at power 50 % for 1 secs
else
turn on left light with color red 0 green 60 blue 0
turn on right light with color red 0 green 0 blue 0
turn left at power 50 % for 1 secs
move forward at power 50 %
turn on all light with color red 0 green 0 blue 0
stop moving
define Initialize
set POWER to 50
Baseline power
Within 40cm of object, display warning
Within 15cm avoid object
Randomly turn right or left

Assignment

Start with your tutorial project and add the following.

- Play a very short sound and change LED color when an object is detected or when avoiding an obstacle. A long sound during detection or avoidance can cause the avoidance not to work properly.
- Experiment with the detection distance.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

Smart Obstacle Avoidance

Time required: 90 minutes

By combining 90 degree turns and ultrasonic sensor readings, your robot can determine which way to go when it senses an obstacle. This program uses more blocks (functions) and a Boolean variable to track whether an obstacle has been detected or not.

A Boolean variable is either true or false. mBlock doesn't have Boolean variables, we use a 0 for false or 1 for true.

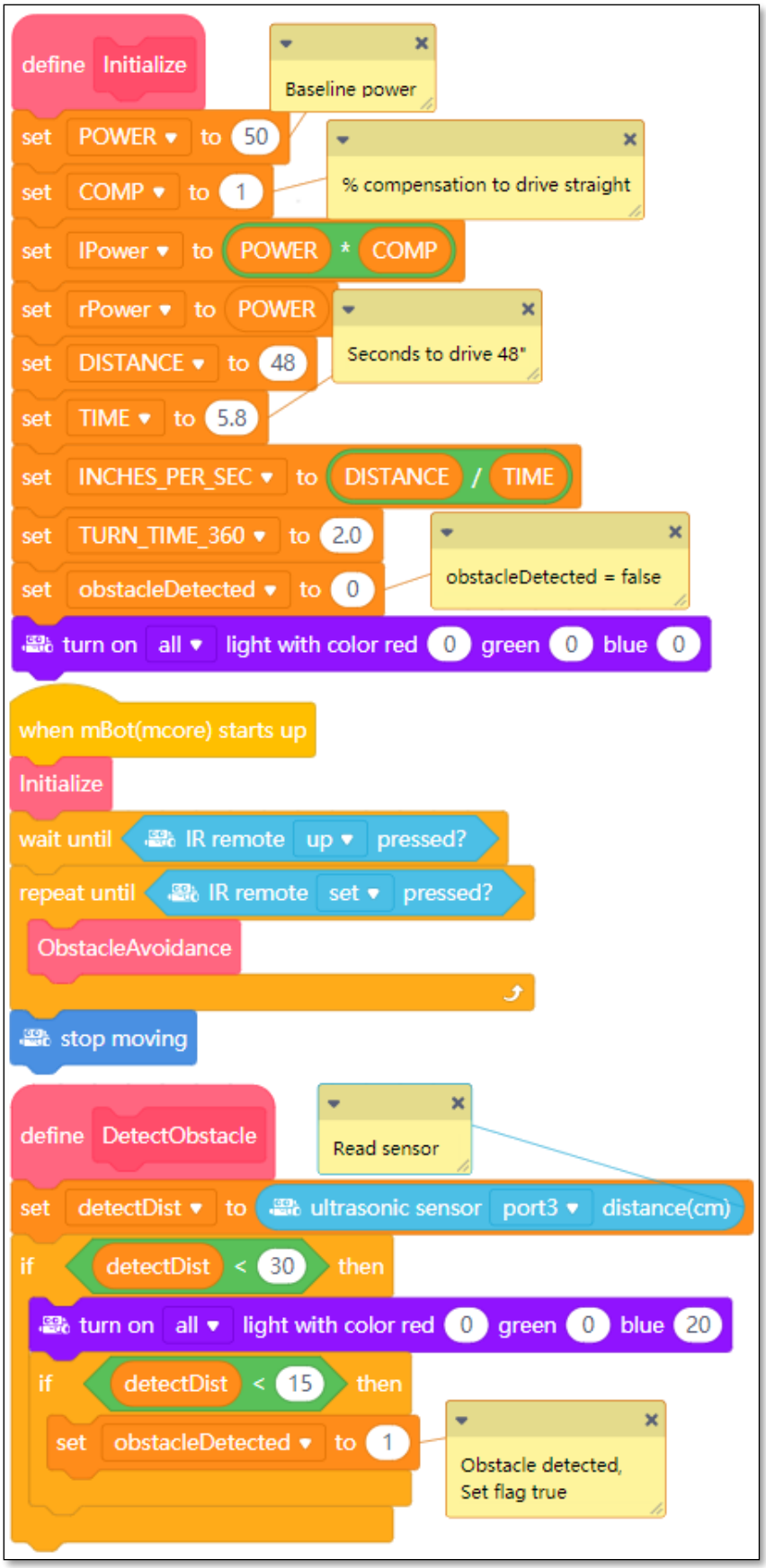
A Boolean variable (also known as a flag) keeps track of the state of the mBot, it allows the mBot to "remember" something for later use. This program will remember whether an obstacle has been detected or not.

Tutorial Assignment

1. Start mBlock.
2. Open **Calibrate Distance and Square**
3. Save the program as **Smart Obstacle Avoidance**.
4. Complete and test the program as pictured with the requirements listed.

Requirements

- When it detects an obstacle: turn right, take a sensor reading, turn left, take a sensor reading. Turn the robot in the direction that has the longest distance.
- Use the accurate turn and movement programs created earlier to make turns and movement more accurate.
- Test obstacle avoidance with your foot.


define Initialize
Baseline power
set POWER to 50
set COMP to 1
% compensation to drive straight
set IPower to POWER * COMP
set rPower to POWER
set DISTANCE to 48
Seconds to drive 48"
set TIME to 5.8
set INCHES_PER_SEC to DISTANCE / TIME
set TURN_TIME_360 to 2.0
set obstacleDetected to 0
obstacleDetected = false
turn on all light with color red 0 green 0 blue 0
when mBot(mcore) starts up
Initialize
wait until IR remote up pressed?
repeat until IR remote set pressed?
ObstacleAvoidance
stop moving
define DetectObstacle
Read sensor
set detectDist to ultrasonic sensor port3 distance(cm)
if detectDist < 30 then
turn on all light with color red 0 green 0 blue 20
if detectDist < 15 then
set obstacleDetected to 1
Obstacle detected,
Set flag true


define TurnRight degrees
set drvTime to degrees / 360 * TURN_TIME_360
left wheel turns at power lPower %, right wheel at power rPower * -1 %
wait drvTime seconds
stop moving
define TurnLeft degrees
set drvTime to degrees / 360 * TURN_TIME_360
left wheel turns at power lPower * -1 %, right wheel at power rPower %
wait drvTime seconds
stop moving
define ObstacleAvoidance
turn on all light with color red 0 green 20 blue 0
left wheel turns at power lPower %, right wheel at power rPower %
DetectObstacle
if obstacleDetected = 1 then
AvoidObstacle
define LookLeft
Find the distance to the obstacle on the left
TurnLeft 90
wait .2 seconds
set detectL to ultrasonic sensor port3 distance(cm)
define AvoidObstacle
turn on all light with color red 20 green 0 blue 0
LookLeft
LookRight
If L > R, Go left
if detectL > detectR then
TurnLeft 180
define LookRight
Find the distance to the obstacle on the right
TurnRight 180
wait .2 seconds
set detectR to ultrasonic sensor port3 distance(cm)
set obstacleDetected to 0
turn on all light with color red 0 green 20 blue 0
Obstacle avoided, set flag false

Assignment

Start with your tutorial project and add the following.

- Setup a maze with available objects, see if your robot can navigate through the maze.
- When the robot moves forward, if there is an obstacle ahead (e.g. 50 cm away), the robot will be alerted and turn on an alarm light and/or very short sound.
- As an optional challenge: As the obstacle gets closer, the short alarm sound and light frequency will gradually accelerate until the robot turns away.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

State Machine (Flags)

Time required: 30 minutes

This program demonstrates how to store the state of the machine (mBot) in a flag variable. Flags are a way of keeping track of the state or history of a robot. The robot can then access that history and decide based on that history. Flags allow for fast switching and checking of task states. Checking a flag is a common Arduino practice for modular programming.

In this program, we are changing modes, which allows the remote buttons to be reused for other code blocks. This is how the default program that came with the mBot works. When the mBot is in **ModeA**, you can set remote button actions in that code block. Switch to **ModeB**, the buttons can have other actions in that code block.

How it Works

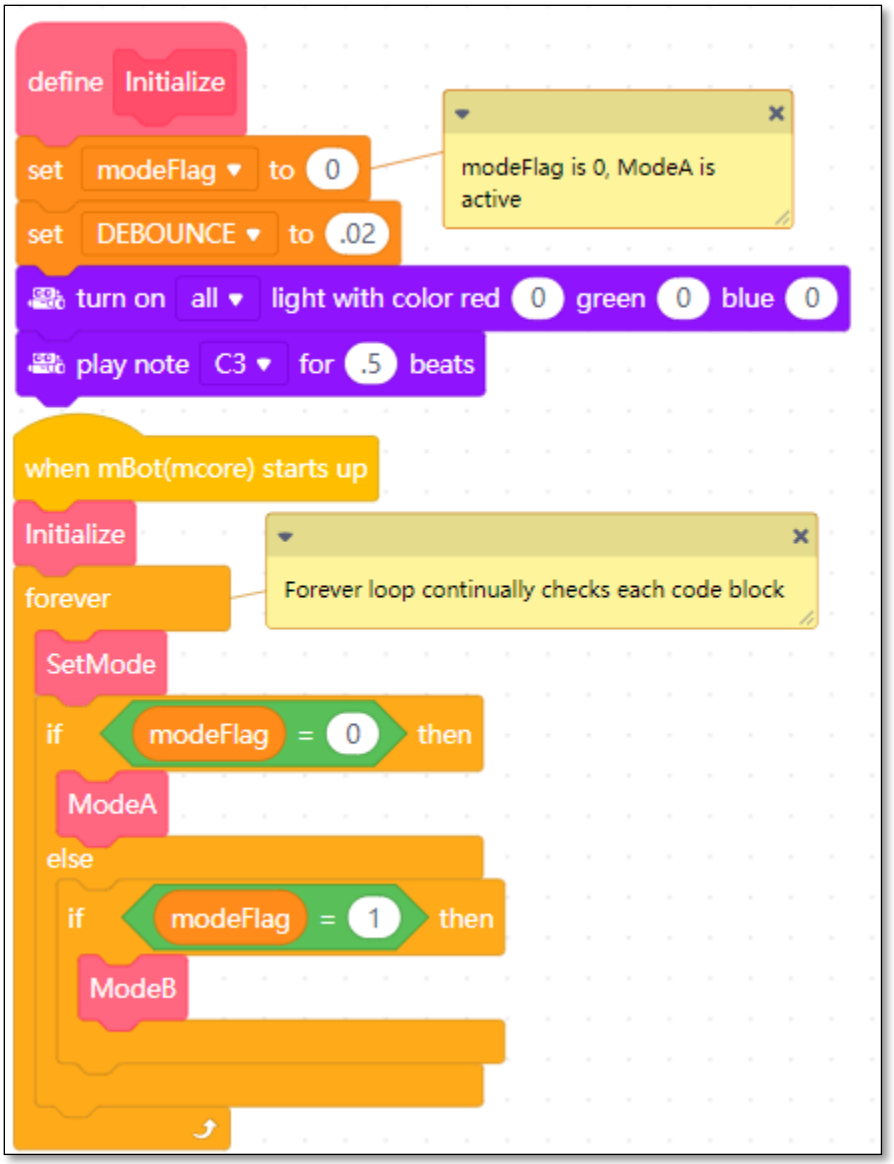
1. The **modeFlag** is set to 0 in the **Initialize** block. **ModeA** code block is active.
2. The forever loop checks for a remote key press in the **SetMode** block.
3. The **SetMode** block changes the **modeFlag** to 1.
4. The **ModeA and ModeB** code blocks keep testing for a **modeFlag** change. When **ModeB** sees the **modeFlag** = 1, it executes and **ModeA** stops.
5. When **SetMode** changes the **modeflag** to 0, we go back to **ModeA**.

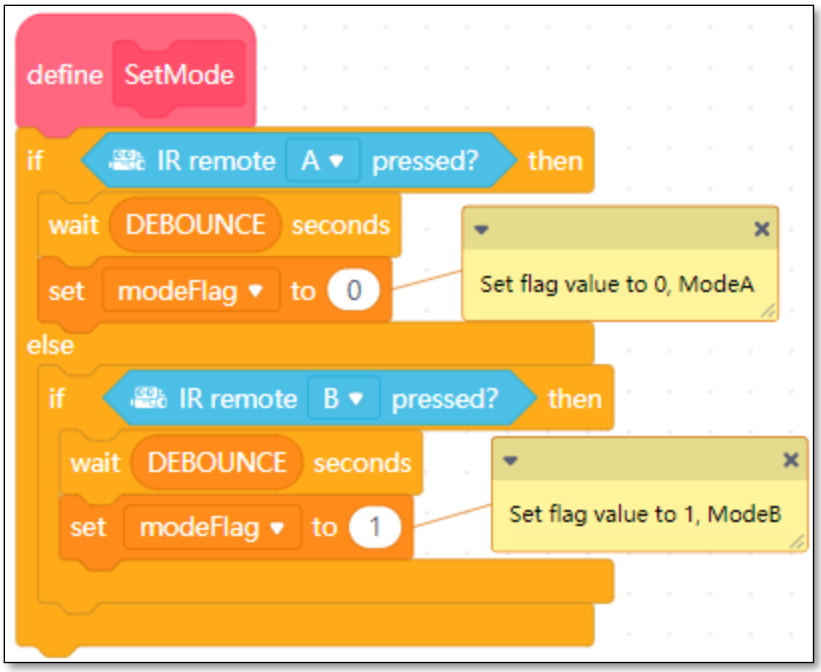
Tutorial Assignment

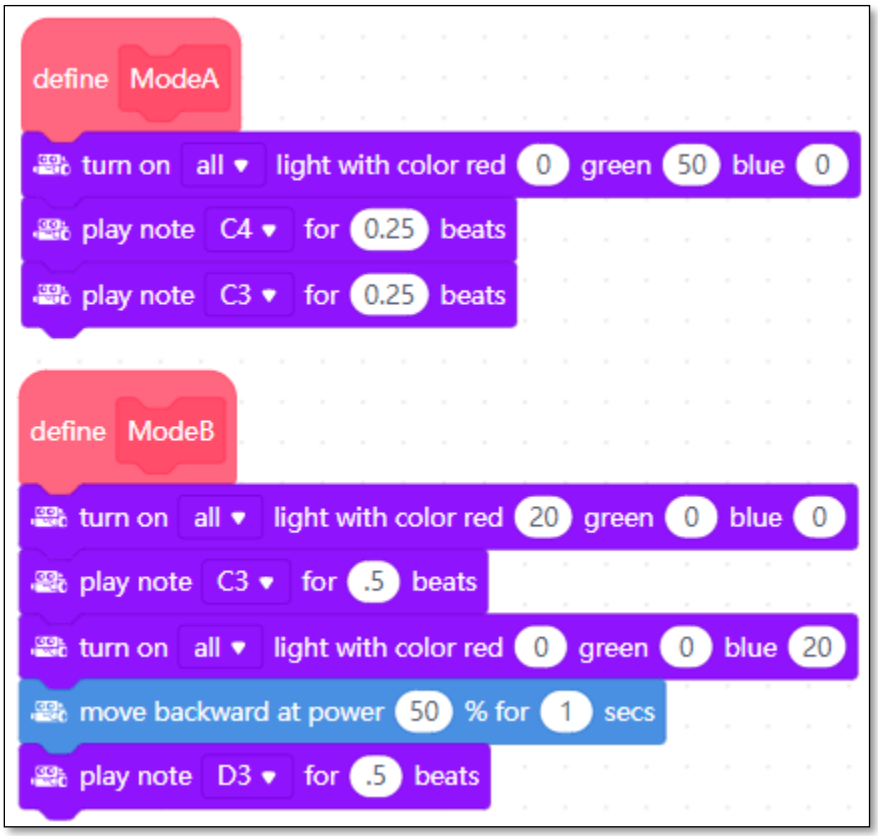
1. Start mBlock.
2. Save the program as **State Machine**.
3. Complete and test the program as pictured with the requirements listed.

Requirements

- Create and test the program.


define Initialize
set modeFlag to 0
modeFlag is 0, ModeA is active
set DEBOUNCE to .02
turn on all light with color red 0 green 0 blue 0
play note C3 for .5 beats
when mBot(mcore) starts up
Initialize
forever
Forever loop continually checks each code block
SetMode
if modeFlag = 0 then
ModeA
else
if modeFlag = 1 then
ModeB





Assignment

Start with your tutorial project and add the following.

- Add **ModeC** to the program. The Flag value would be 2.
- Have **ModeC** do something else.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

mBot Default Program Part 1: Remote Control

Time required: 120 minutes

NOTE: Use your own COMP, TIME, and TURN_TIME_360 numbers from the Calibrate Distance and Calibrate program. These programs were tested on my mBot, not yours. Your mBot will run differently.

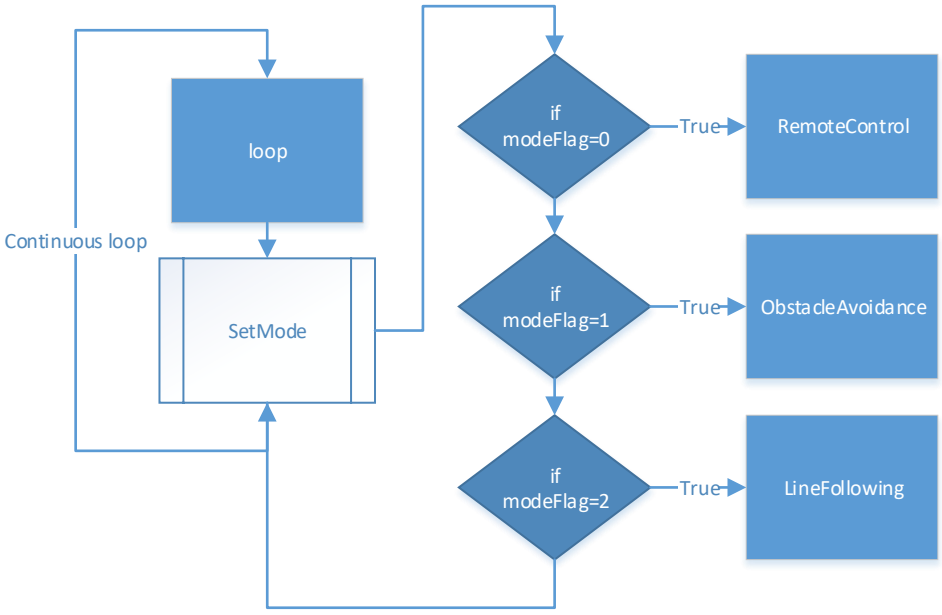
This program is the first of 3 steps to build a full featured mBot default program. This program uses the machine state/flag concept introduced in the Machine State assignment.

This program adds remote control features for the first part of the default program.

The SetSpeed numbers are a percentage of total power, 100 is 100%, 30 is 30%.

Knowledge Points

1. The program continuously watches or scans for input from the remote control by looping.
2. If the **modeFlag** is changed to 0, the **RemoteControl** portion of the program is active.
3. If it changes to 1, the **ObstacleAvoidance** is active. And so on.



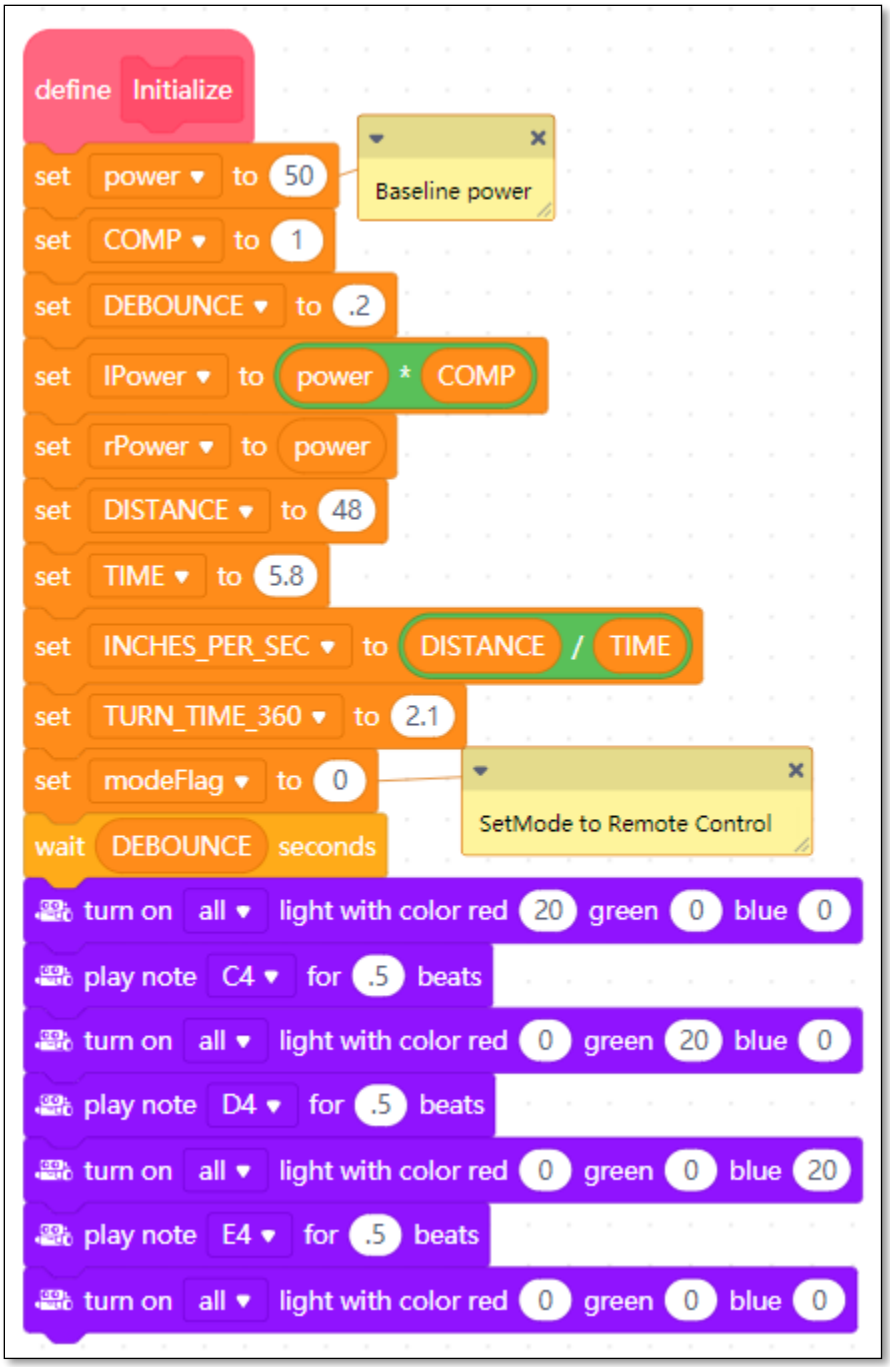
Tutorial Assignment

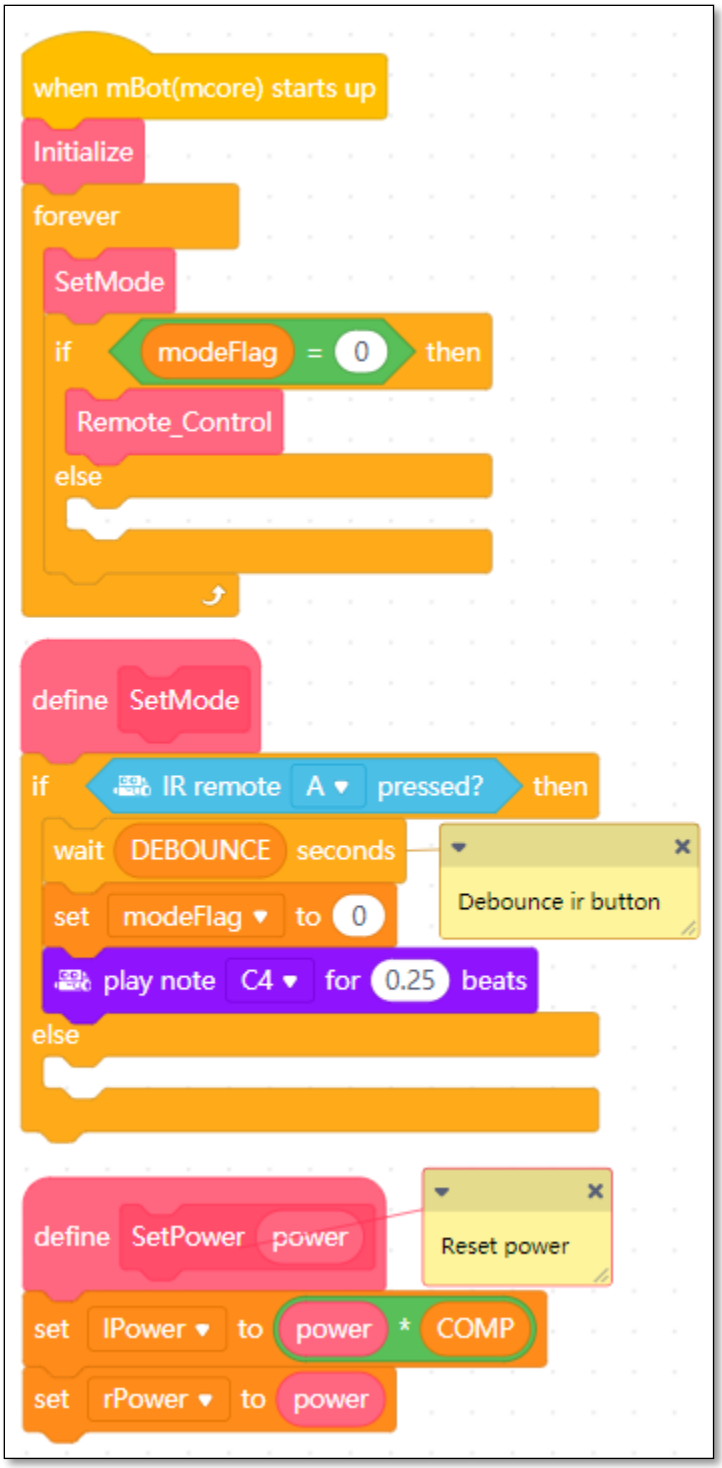
1. Start mBlock and open **Driving School 2**. Save as **mBot-Default-Program-Part-1 Remote Control**.
2. Complete and test the program as pictured with the requirements listed.

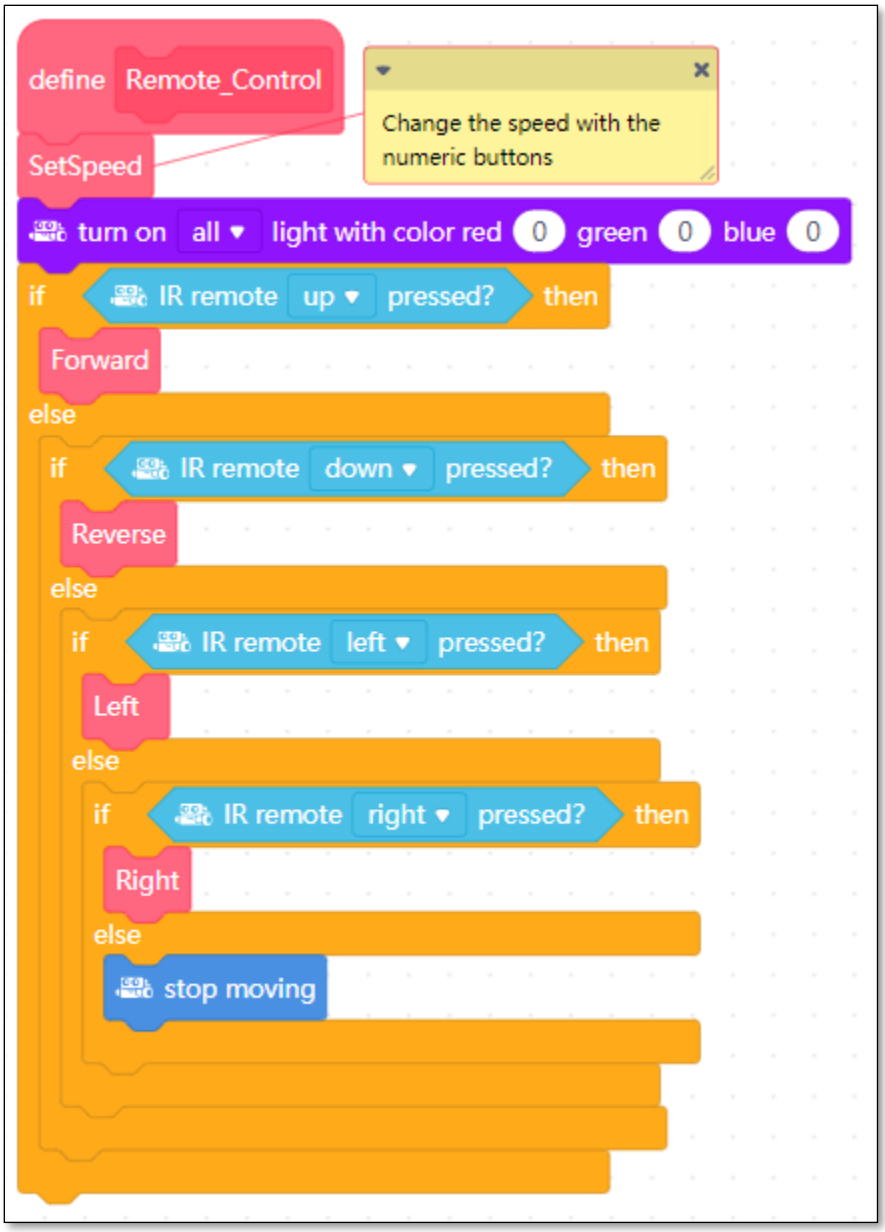
Requirements

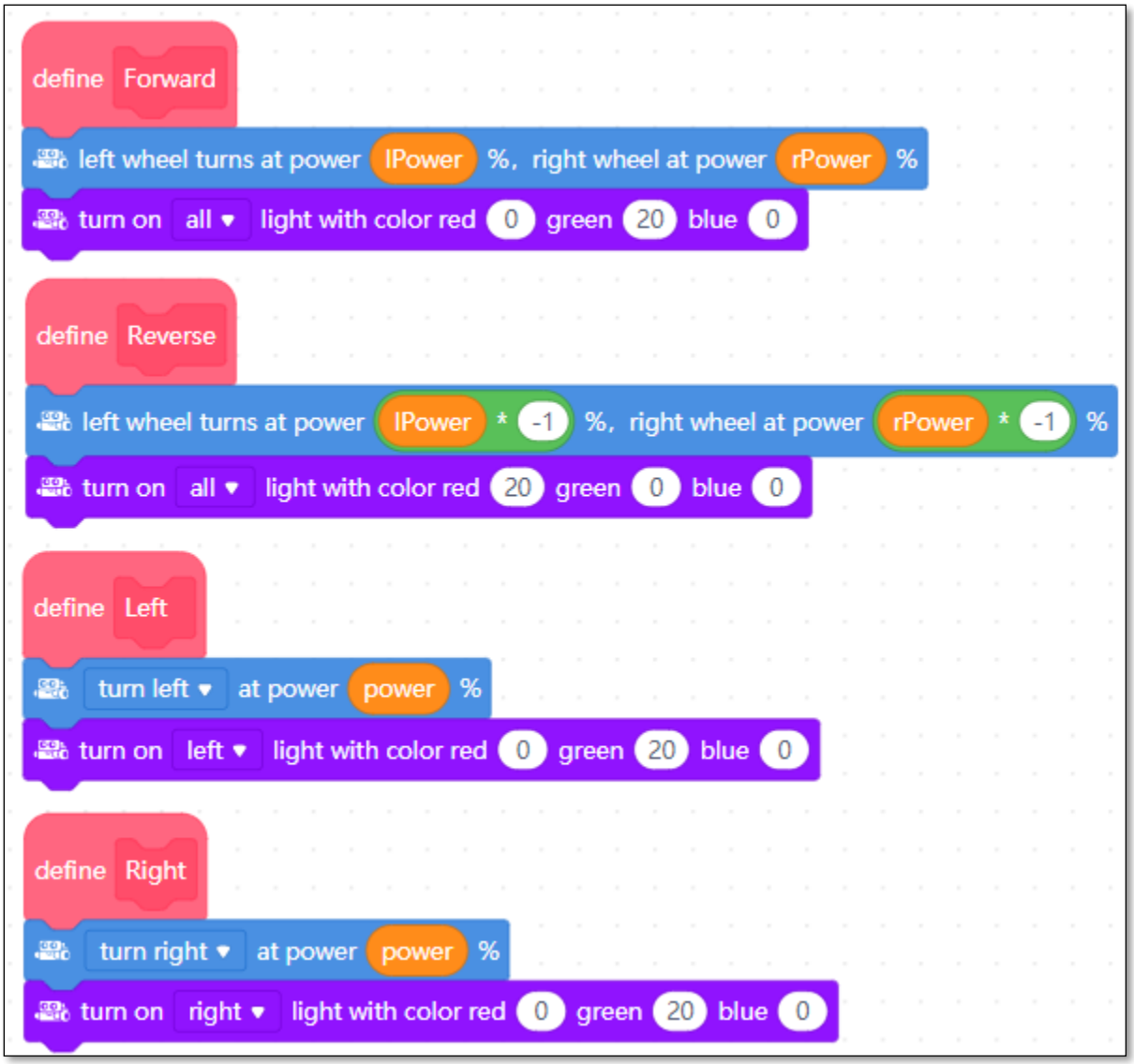
- Create and test the program.
- Button A starts remote control mode. The modeFlag = 0
- Set the speed of the robot with the number keys.

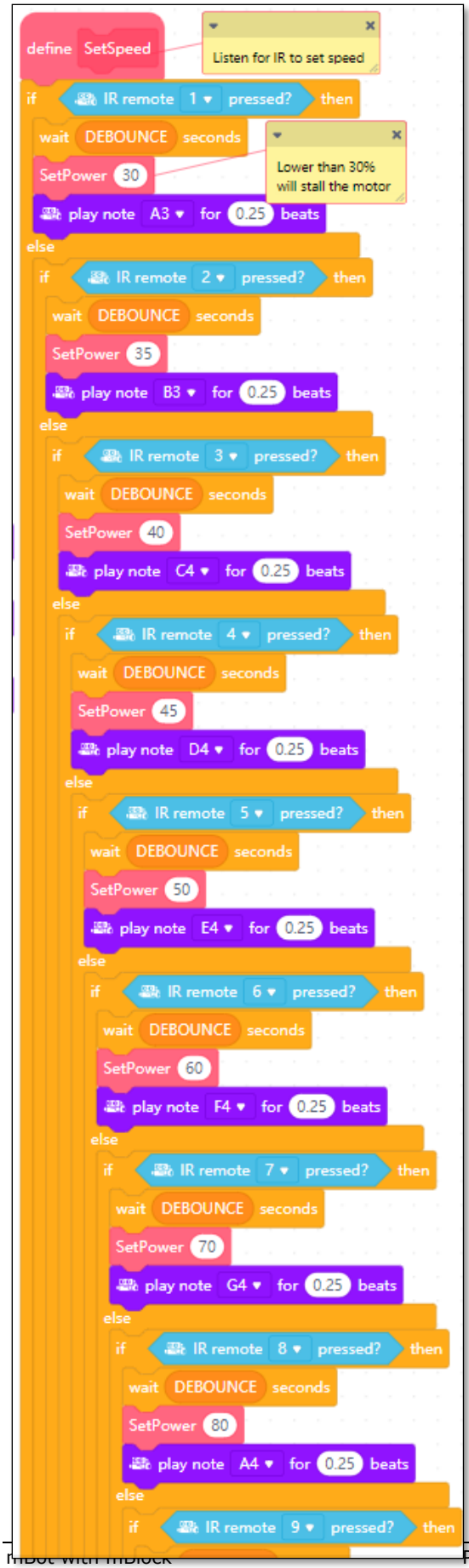
NOTE: The program below is shown in several screen shots to make the resolution big enough to see.

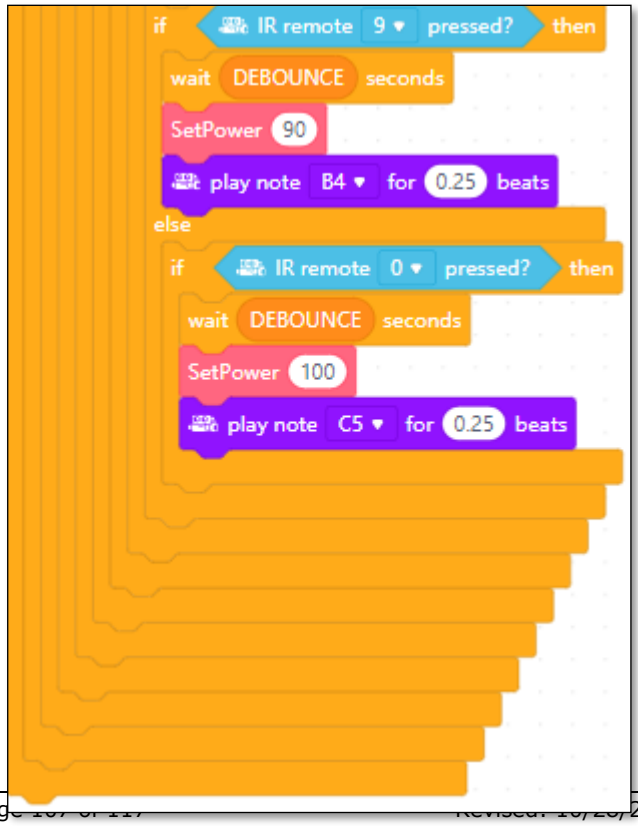

define Initialize
set power to 50
Baseline power
set COMP to 1
set DEBOUNCE to .2
set IPower to power * COMP
set rPower to power
set DISTANCE to 48
set TIME to 5.8
set INCHES_PER_SEC to DISTANCE / TIME
set TURN_TIME_360 to 2.1
set modeFlag to 0
SetMode to Remote Control
wait DEBOUNCE seconds
turn on all light with color red 20 green 0 blue 0
play note C4 for .5 beats
turn on all light with color red 0 green 20 blue 0
play note D4 for .5 beats
turn on all light with color red 0 green 0 blue 20
play note E4 for .5 beats
turn on all light with color red 0 green 0 blue 0


when mBot(mcore) starts up
Initialize
forever
SetMode
if modeFlag = 0 then
Remote_Control
else
define SetMode
if IR remote A pressed? then
wait DEBOUNCE seconds
Debounce ir button
set modeFlag to 0
play note C4 for 0.25 beats
else
define SetPower power
Reset power
set lPower to power * COMP
set rPower to power


define Remote_Control
SetSpeed
Change the speed with the numeric buttons
turn on all light with color red 0 green 0 blue 0
if IR remote up pressed? then
Forward
else
if IR remote down pressed? then
Reverse
else
if IR remote left pressed? then
Left
else
if IR remote right pressed? then
Right
else
stop moving


define Forward
left wheel turns at power lPower %, right wheel at power rPower %
turn on all light with color red 0 green 20 blue 0
define Reverse
left wheel turns at power lPower * -1 %, right wheel at power rPower * -1 %
turn on all light with color red 20 green 0 blue 0
define Left
turn left at power power %
turn on left light with color red 0 green 20 blue 0
define Right
turn right at power power %
turn on right light with color red 0 green 20 blue 0


define SetSpeed
Listen for IR to set speed
if IR remote 1 pressed? then
wait DEBOUNCE seconds
SetPower 30
Lower than 30% will stall the motor
play note A3 for 0.25 beats
else
if IR remote 2 pressed? then
wait DEBOUNCE seconds
SetPower 35
play note B3 for 0.25 beats
else
if IR remote 3 pressed? then
wait DEBOUNCE seconds
SetPower 40
play note C4 for 0.25 beats
else
if IR remote 4 pressed? then
wait DEBOUNCE seconds
SetPower 45
play note D4 for 0.25 beats
else
if IR remote 5 pressed? then
wait DEBOUNCE seconds
SetPower 50
play note E4 for 0.25 beats
else
if IR remote 6 pressed? then
wait DEBOUNCE seconds
SetPower 60
play note F4 for 0.25 beats
else
if IR remote 7 pressed? then
wait DEBOUNCE seconds
SetPower 70
play note G4 for 0.25 beats
else
if IR remote 8 pressed? then
wait DEBOUNCE seconds
SetPower 80
play note A4 for 0.25 beats
else
if IR remote 9 pressed? then


if IR remote 9 pressed? then
wait DEBOUNCE seconds
SetPower 90
play note B4 for 0.25 beats
else
if IR remote 0 pressed? then
wait DEBOUNCE seconds
SetPower 100
play note C5 for 0.25 beats

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

mBot Default Program Part 2: Obstacle Avoidance

Time required: 120 minutes

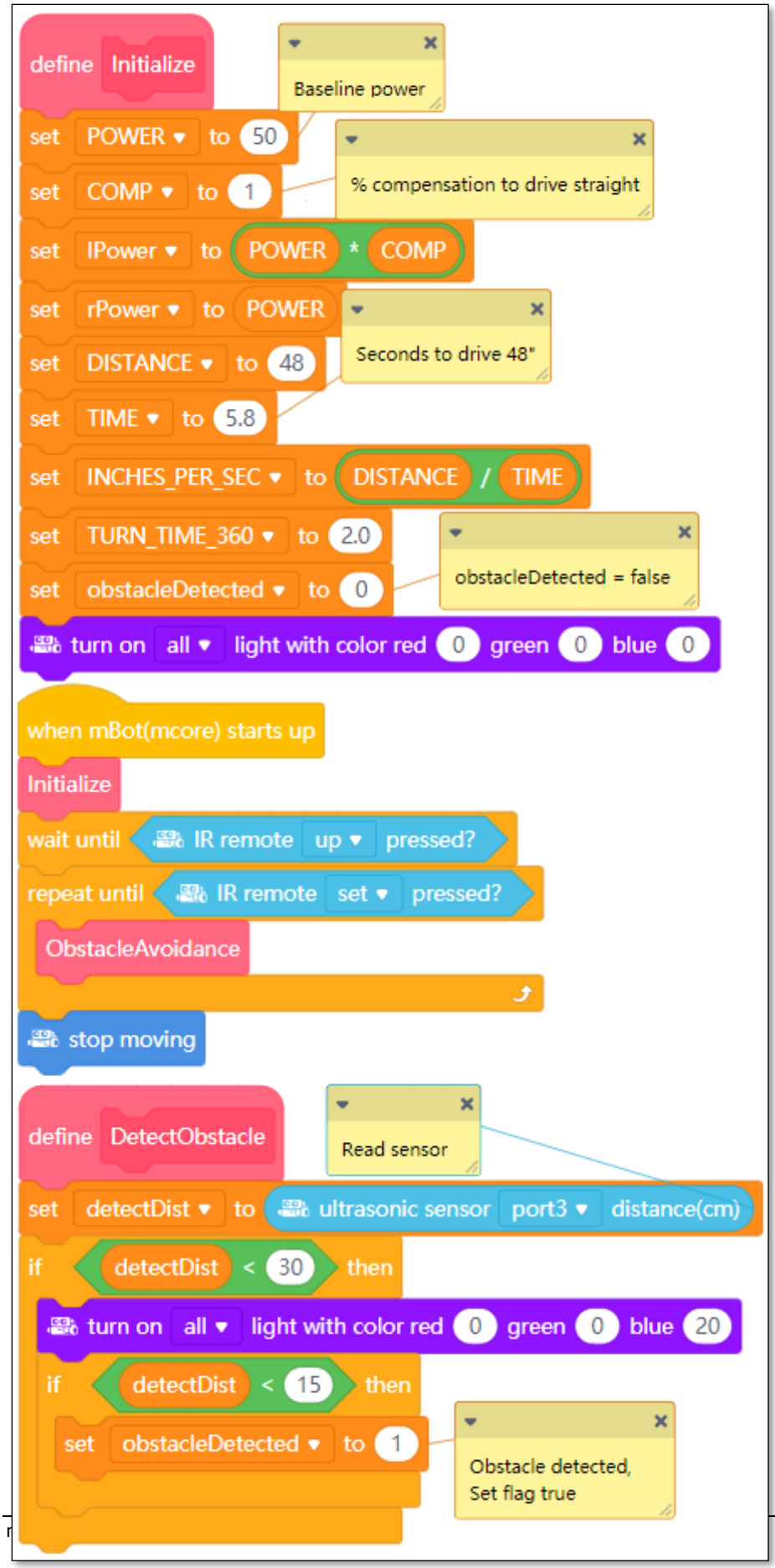
This program adds the Smart Obstacle Avoidance program we did earlier to our default program.

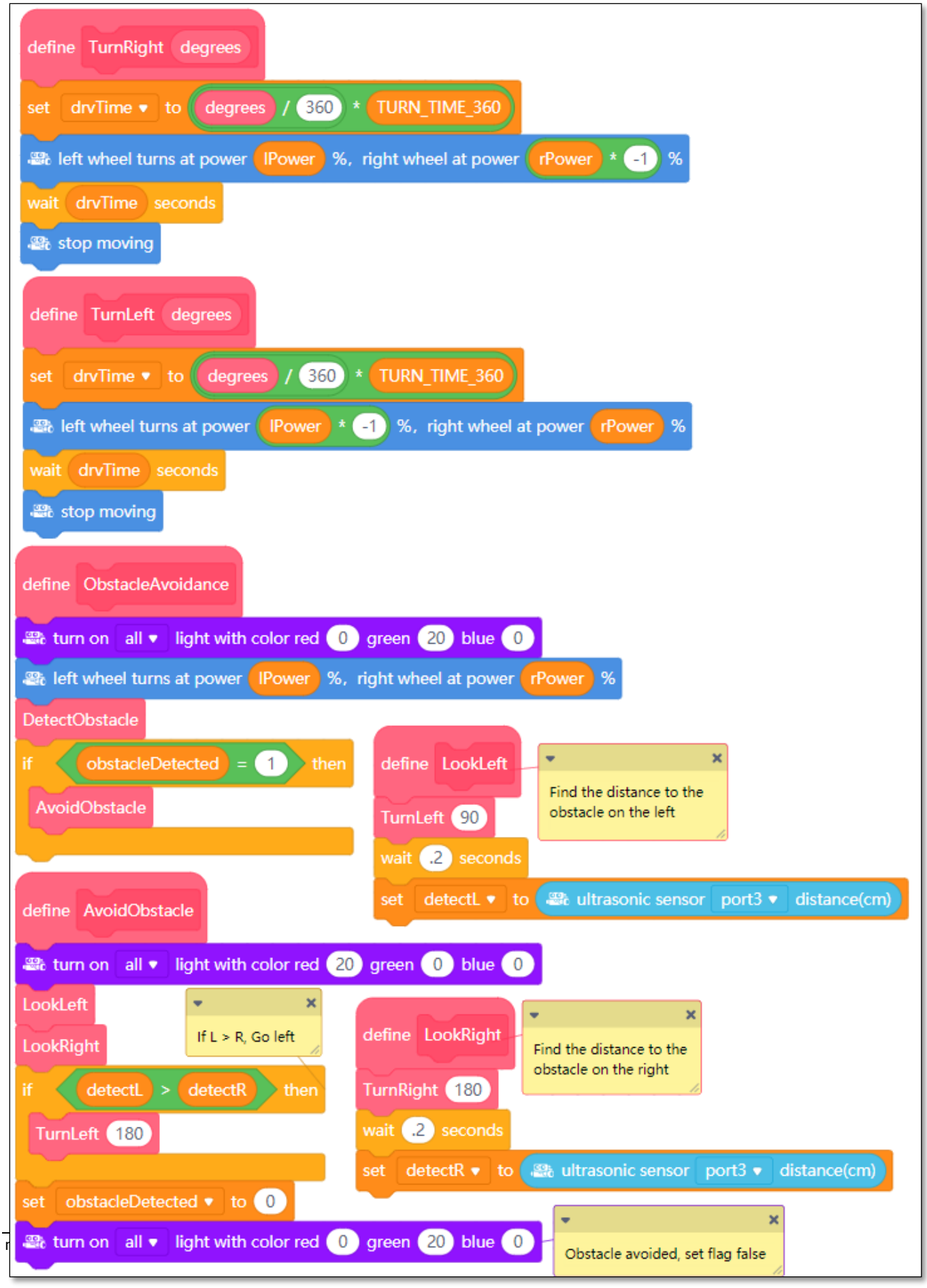
Tutorial Assignment

1. Start mBlock → Open **mBot Default Program Part 1 Remote Control**
2. Save to your computer as **mBot Default Program Part 2 Obstacle Avoidance**
3. Complete and test the program with the requirements listed.

Requirements

- Use **if then else** blocks to extend the **SetMode** code block to include Button B setting the obstacle avoidance mode.
- Button A starts remote control mode: modeFlag = 0
- Button B starts obstacle avoidance mode: modeFlag = 1
- Integrate **Smart Obstacle Avoidance** into the default program. The original code is shown below.
- Create and test the program.


define Initialize
Baseline power
set POWER to 50
% compensation to drive straight
set COMP to 1
set IPower to POWER * COMP
set rPower to POWER
Seconds to drive 48"
set DISTANCE to 48
set TIME to 5.8
set INCHES_PER_SEC to DISTANCE / TIME
set TURN_TIME_360 to 2.0
obstacleDetected = false
set obstacleDetected to 0
turn on all light with color red 0 green 0 blue 0
when mBot(mcore) starts up
Initialize
wait until IR remote up pressed?
repeat until IR remote set pressed?
ObstacleAvoidance
stop moving
define DetectObstacle
Read sensor
set detectDist to ultrasonic sensor port3 distance(cm)
if detectDist < 30 then
turn on all light with color red 0 green 0 blue 20
if detectDist < 15 then
set obstacleDetected to 1
Obstacle detected,
Set flag true


define TurnRight degrees
set drvTime to degrees / 360 * TURN_TIME_360
left wheel turns at power lPower %, right wheel at power rPower * -1 %
wait drvTime seconds
stop moving
define TurnLeft degrees
set drvTime to degrees / 360 * TURN_TIME_360
left wheel turns at power lPower * -1 %, right wheel at power rPower %
wait drvTime seconds
stop moving
define ObstacleAvoidance
turn on all light with color red 0 green 20 blue 0
left wheel turns at power lPower %, right wheel at power rPower %
DetectObstacle
if obstacleDetected = 1 then
AvoidObstacle
define LookLeft
Find the distance to the obstacle on the left
TurnLeft 90
wait .2 seconds
set detectL to ultrasonic sensor port3 distance(cm)
define AvoidObstacle
turn on all light with color red 20 green 0 blue 0
LookLeft
LookRight
If L > R, Go left
if detectL > detectR then
TurnLeft 180
define LookRight
Find the distance to the obstacle on the right
TurnRight 180
wait .2 seconds
set detectR to ultrasonic sensor port3 distance(cm)
set obstacleDetected to 0
turn on all light with color red 0 green 20 blue 0
Obstacle avoided, set flag false

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

mBot Default Program Part 3: Line Following

Time required: 120 minutes

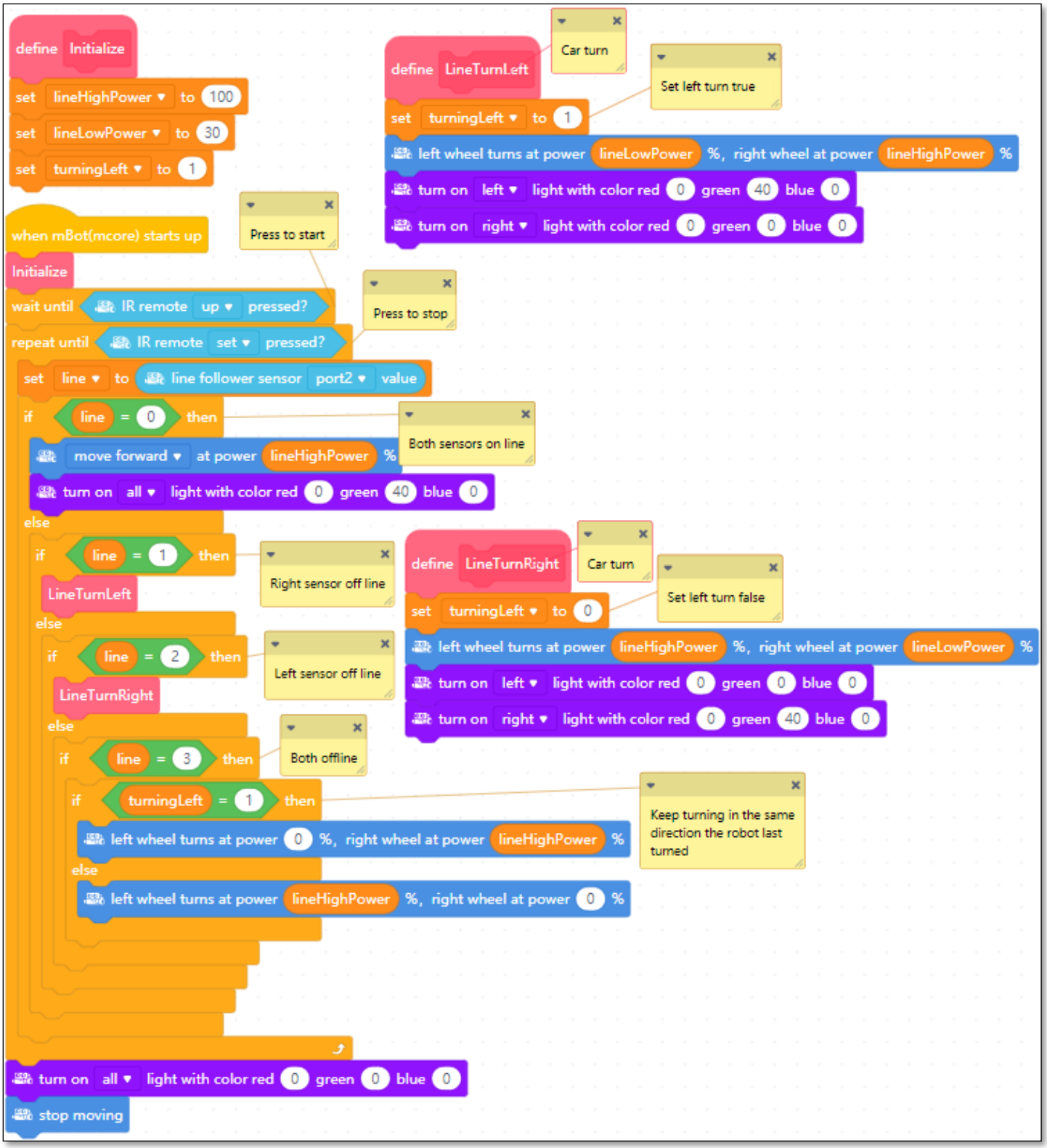
This program adds line following features to our default program.

Tutorial Assignment

1. Start mBlock. Open **mBot Default Program Part 2 Obstacle Avoidance**.
2. Save as **mBot Default Program Part 3 Line Following**.
3. Complete and test the program with the requirements listed.

Requirements

- Integrate the Smart Line Following assignment with your current mBot Default Program.
    - From the Smart Line Following program: remove the Initialize block, wait until, repeat until, and run forward at speed 0.
- Create separate turn functions for the line following. Experiment with the turns to speed up and smooth out the line following. Use factors to change leftPower and rightPower. By doing this, you can press A, use the Set_Speed block to change the speed, then press C and the Line Following will work at the new speed.
- Button A starts remote control mode. The modeFlag = 0
- Button B starts obstacle avoidance mode. The modeFlag = 1
- Button C starts line following mode. The modeFlag = 2
- Create and test the program.


define Initialize
set lineHighPower to 100
set lineLowPower to 30
set turningLeft to 1
when mBot(mcore) starts up
Initialize
wait until IR remote up pressed?
repeat until IR remote set pressed?
set line to line follower sensor port2 value
if line = 0 then
move forward at power lineHighPower %
turn on all light with color red 0 green 40 blue 0
else
if line = 1 then
LineTurnLeft
else
if line = 2 then
LineTurnRight
else
if line = 3 then
if turningLeft = 1 then
left wheel turns at power 0 %, right wheel at power lineHighPower %
else
left wheel turns at power lineHighPower %, right wheel at power 0 %
turn on all light with color red 0 green 0 blue 0
stop moving
Press to start
Press to stop
Both sensors on line
Right sensor off line
Left sensor off line
Both offline
Keep turning in the same direction the robot last turned
define LineTurnLeft
Car turn
set turningLeft to 1
Set left turn true
left wheel turns at power lineLowPower %, right wheel at power lineHighPower %
turn on left light with color red 0 green 40 blue 0
turn on right light with color red 0 green 0 blue 0
define LineTurnRight
Car turn
set turningLeft to 0
Set left turn false
left wheel turns at power lineHighPower %, right wheel at power lineLowPower %
turn on left light with color red 0 green 0 blue 0
turn on right light with color red 0 green 40 blue 0

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.

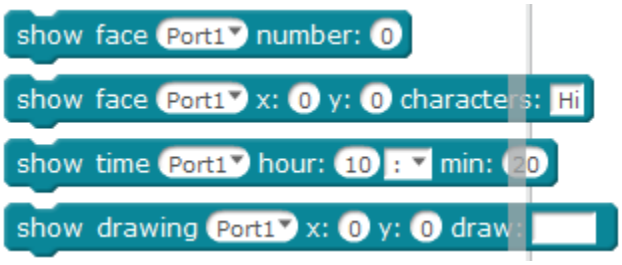
In Class Project: Me LED Matrix 8x16 Module

Time required: 30 minutes

This project demonstrates an addon for the mBot. This program adds an Me LED Matrix 8x16 Module to the mBot. You can draw arrows, put in text, make faces and lots of other creative, fun things.

Knowledge

The LED Matrix has four blocks.



In this tutorial we will use the show drawing to put arrows for direction. We will also use show face characters to show stop.

Tutorial Assignment

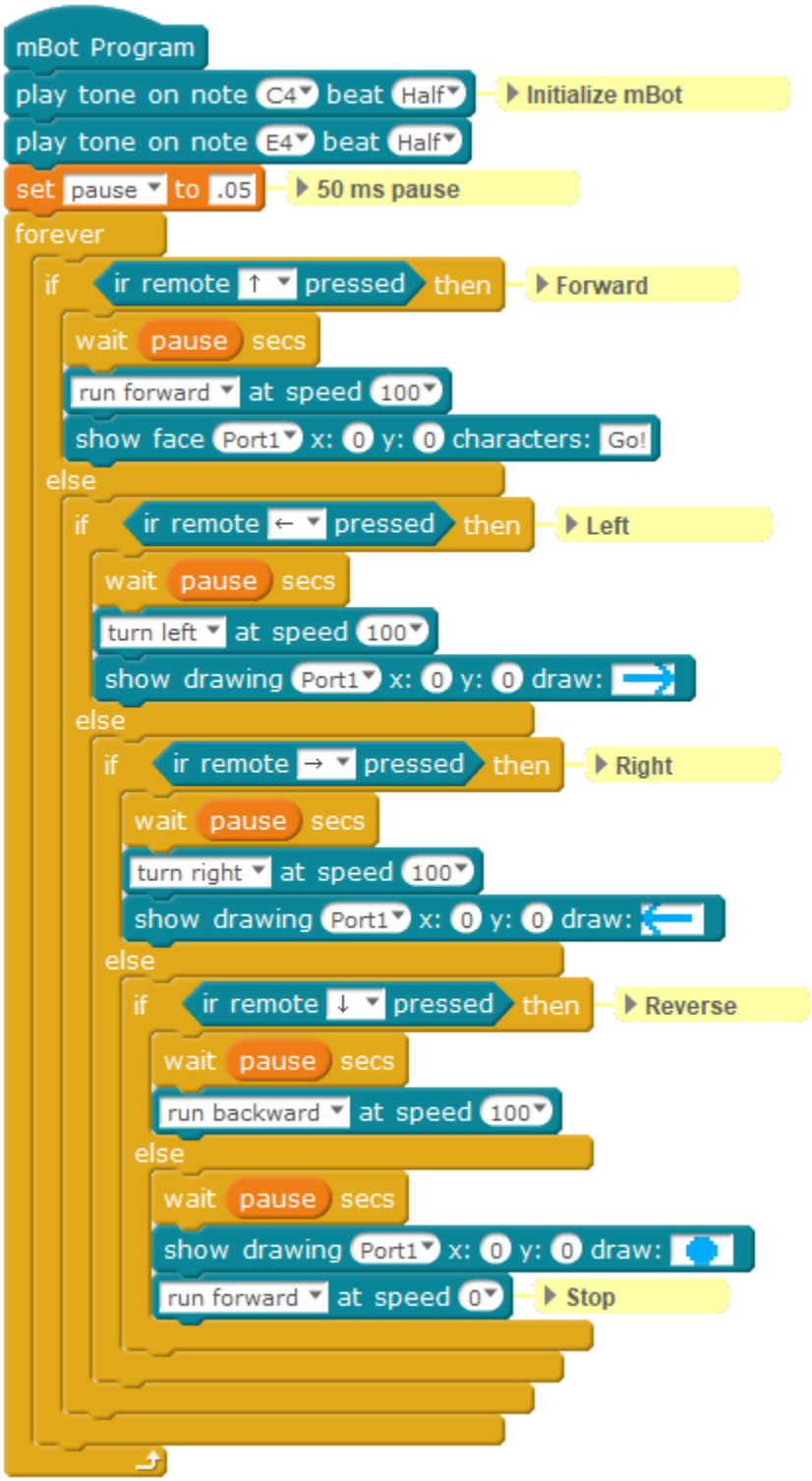
1. Start mBlock. Start a new mBlock program, save as **Remote with LED Matrix**.
2. Complete and test the program as pictured with the requirements listed.

Requirements

- Make the following changes to the Remote Control part of the program as shown in the example.
- The 50 ms pause is to allow the LED to light and motors to run. Without the pause, the program won't work.
- Come up with something creative for Stop.

Assignment Submission

- Attach the program, and submit the assignment in Blackboard.
- The assignment is demonstrated in class. Online students submit the completed program.


mBot Program
play tone on note C4 beat Half
Initialize mBot
play tone on note E4 beat Half
set pause to .05
50 ms pause
forever
if ir remote ↑ pressed then
Forward
wait pause secs
run forward at speed 100
show face Port1 x: 0 y: 0 characters: Go!
else
if ir remote ← pressed then
Left
wait pause secs
turn left at speed 100
show drawing Port1 x: 0 y: 0 draw:
else
if ir remote → pressed then
Right
wait pause secs
turn right at speed 100
show drawing Port1 x: 0 y: 0 draw:
else
if ir remote ↓ pressed then
Reverse
wait pause secs
run backward at speed 100
else
wait pause secs
show drawing Port1 x: 0 y: 0 draw:
run forward at speed 0
Stop