Science, Coding, and Robotics: A Complete STEM Experience

Vernier

MEASURE. ANALYZE. LEARN.™

Vernier Software & Technology
www.vernier.com
888.837.6437

Josh Ence
engineering@vernier.com

NSTA National 2019
St. Louis, MO

Driving Outside the Lines

Up to this point, you have used the Line-Follower and Ultrasonic Sensors to guide mBot. But with some simple math and subroutines, you can direct mBot to travel outside the lines!

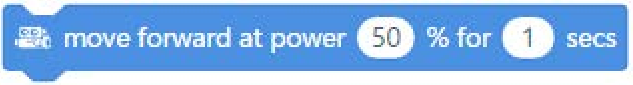
Drive a Specific Path

Most GPS devices have route planners that will determine the best path to a specific destination. You can make your mBot travel along a predefined path without needing a line to follow by controlling the time the motors are on with the blocks in the Action palette.

Consider the challenge of making mBot move in the path of the letter M, such as the one on the side of the robot. mBot could make the M shape by moving in straight lines and making just three turns. Let’s say you want the M shape to be 50 cm tall and are going to start with mBot at the lower left. mBot will need to do the following:

a. Drive straight for 50 cm.
b. Turn right for an angle of more than 90 degrees; we'll use a value of 135 degrees.
c. Drive straight for about half as long as the first straight segment; we'll use a value of 25 cm.
d. Turn left 90 degrees.
e. Drive straight again; we'll use a value of 25 cm to match the previous segment.
f. Turn right again, the same angle as the first turn, 135 degrees.
g. Drive straight for 50 cm.

How can you make mBot move in this pattern? If we know how far mBot travels per second, we can determine how long it needs to drive to cover a specified distance. Likewise, if we know how many degrees mBot turns per second, we can determine how long it needs to turn to sweep through a given angle. Essentially, we can convert the distance and turn angles into "motor on" times. The "move forward" block will help with this determination; you can use it to set a constant speed at which mBot will drive and the duration of time it travels at that speed. This kind of navigation, based on knowing what direction and speed you are going, is called *dead reckoning*. To navigate via dead reckoning requires that you know your vehicle well.



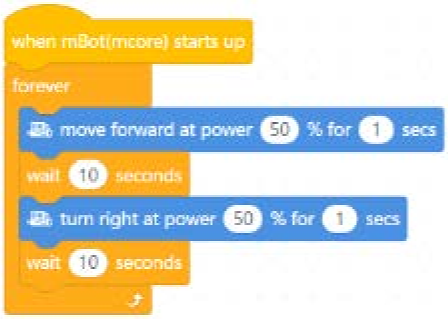
```
move forward at power 50 % for 1 secs
```

FIGURE 1 The "move forward" block allows you to specify the power level and the duration of time that mBot moves.

Measure mBot’s Movement and Turning Rates

We will use the Speed and Turn Rate program to measure mBot's movement and turning rates. These motion measurements will be important in your navigation programs; each will be become a variable in your code. So, here are some tips to consider as you develop your program and make your measurements:

- For this kind of navigation, a slower speeds tends to give you better accuracy. In the Speed and Turn Rate program, mBot runs at a power level of 50 or lower.
- You may have noticed that mBot moves fastest when the batteries are fully charged. Tests have shown that, under use after being fully changed, the battery voltage drops quickly during the first 30 minutes or so and then settles into a more steady, slowly-dropping voltage. Because of this, you will get best results for dead-reckoning navigation if you avoid determining the movement and turning rate with fully charged batteries.
- As with any measurement of this type, it is best to repeat the measurements a few times and calculate an average.



```
when mBot(mcore) starts up
forever
    move forward at power 50 % for 1 secs
    wait 10 seconds
    turn right at power 50 % for 1 secs
    wait 10 seconds
```

FIGURE 2 Speed and Turn Rate program

When you are ready to make your measurements, upload the program to your mBot. While the program is running, determine the following values and record them in Table 1:

1. cmPerSecond: Use a meter stick to measure how far mBot travels forward in one second.
2. degreesPerSecond: Use a protractor to measure how many degrees mBot rotates to the right in one second.

Table 1	
cmPerSecond	
degreesPerSecond	

Simplifying with Subroutines

Now that you know how fast mBot drives and turns, you can start determining how long to drive the motor for each segment of the M pattern. For instance, if your value for cmPerSecond is 16 cm/s, how long will mBot need to drive forward to make the first 50 cm segment of the M?

Drive Forward Time = (distance to move(cm)) / (cmPerSecond)

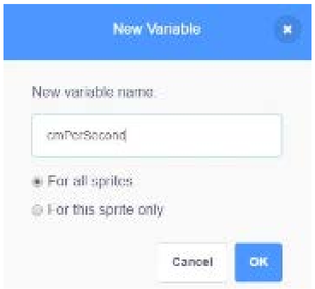
= 50 cm / 16 cm per second

= 3.1 seconds

Since you will often do calculations of this type while you develop the program to make the M shape, this is the perfect opportunity to use a subroutine. We can make a subroutine that will calculate the correct amount of "move forward" time based on a distance value you enter.

To add this to your mBlock code, do the following:

1. Create a "cmPerSecond" variable variable by clicking the Variables palette and then the Make a Variable button (see Figure 3).



New variable name:
cmPerSecond

● For all sprites
○ For this sprite only

FIGURE 3 Name the new variable

2. Create a "moveDistance" block.

   a. Select the My Blocks palette and click the Make a Block button (see Figure 4).
   b. Enter **moveDistance** as the name of the block and then select Options.
   c. To add number input, which makes it possible the program to calculate the "drive forward" time based on a value you enter, select "Add an input number".
   d. Change the number input label from the default "number1" to "cm".

   Once created, "cm" is a variable that can only be used by the "moveDistance" block. When you use the "moveDistance" block in your code, you will specify a value in the "moveDistance" block. That value is stored as the "cm" variable and, any place "cm" is used, mBot will use the value you specified.

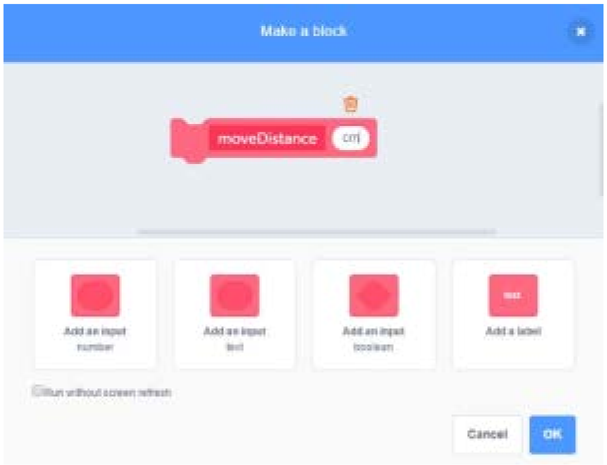
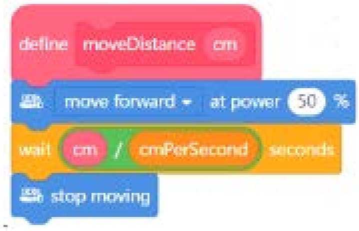


FIGURE 4 Create the "moveDistance" block with a number input. The number input will be the distance you want mBot to move.

3. Finish the "moveDistance" block by adding the rest of the movement code shown in Figure 5. The subroutine is fairly simple:

   a. First, it turns on mBot's motors to run forward.
   b. Next, it performs a quick calculation to determine how long to run the motors ("cm" divided by "cmPerSecond") and waits that amount of time.
   c. Finally, it stops mBot.



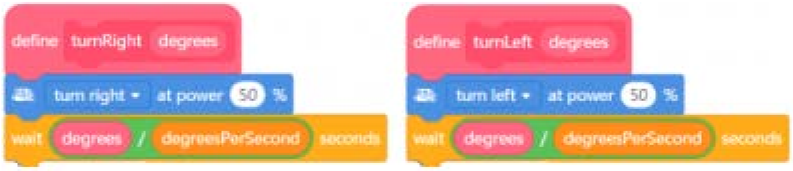
```
define moveDistance cm
move forward at power 50 %
wait cm / cmPerSecond seconds
stop moving
```

FIGURE 5 The complete "moveDistance" block

The great thing about this block is that don't have to do any math calculations yourself; the program does it all for you! When you enter a value for the "distance to travel," the subroutine calculates the correct "motor on" time based on the cmPerSecond rate you set. The program then runs the motor for just that time.

4. You can create turning blocks following the same format:
   a. Create a "degreesPerSecond" variable in your main program code. You will set this variable to your robot's unique turning rate.
   b. Create "turnRight" and "turnLeft" blocks that accept a number input. The input will be the number of degrees you want mBot to turn.
   c. Define the turning subroutines to do the same math as the "moveDistance" block.

You should have now have two more blocks:

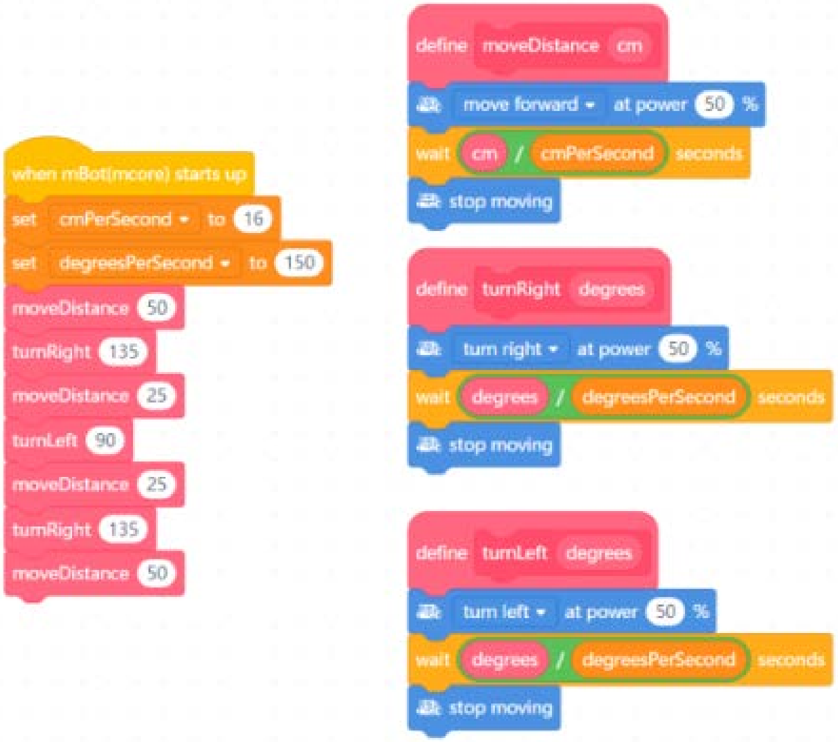


```
define turnRight degrees
turn right ▾ at power 50 %
wait degrees / degreesPerSecond seconds
```

```
define turnLeft degrees
turn left ▾ at power 50 %
wait degrees / degreesPerSecond seconds
```

FIGURE 6 "turnRight" and "turnLeft" blocks turn mBot right and left, respectively, in much the same way the "moveDistance" subroutine drives mBot forward.

With these powerful blocks, also known as subroutines, the program to make mBot move in an **M** path is very simple. The main program is at the left in Figure 7:



```
when mBot(mcore) starts up
set cmPerSecond ▾ to 16
set degreesPerSecond ▾ to 150
moveDistance 50
turnRight 135
moveDistance 25
turnLeft 90
moveDistance 25
turnRight 135
moveDistance 50
```

```
define moveDistance cm
move forward ▾ at power 50 %
wait cm / cmPerSecond seconds
stop moving
```

```
define turnRight degrees
turn right ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

```
define turnLeft degrees
turn left ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

FIGURE 7 Drawing the M program, complete with subroutines

TRY IT OUT

In mBlock, create the Drawing the M program (Figure 7), and upload the program to your mBot. Set the "cmPerSecond" and "degreesPerSecond" variables to the rates you measured when mBot ran the Speed and Turn Rate program.

How well does mBot follow an M path? Attach a pen or marker to mBot with a zip-tie and have mBot drive over a large sheet of paper. Does mBot make an M? Adjust values until it does.

Challenge Extension

EXTENSION 1 Draw Another Letter

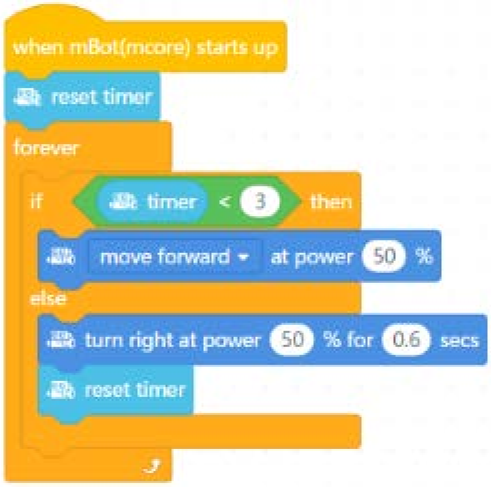
Write a program to make mBot move in the shape of a Z, a V, or a W. If you use the blocks in Figures 5 and 6 and a "calibrated" mBot (i.e., an mBot whose cmPerSecond and degreesPerSecond are known), these programs should be pretty easy.

Timer Command: Driving a Path

Another way to control the motion of mBot is to use mBot’s built-in timer. There are two blocks in the Robots palette that control the timer:

- The "timer" block reports the time in seconds since the timer was last reset. You can use this block in the same ways that you use a variable. For example, you can use it in IF-THEN structures or WAIT UNTIL structures.
- The "reset timer" block resets the timer to 0 seconds.

You can use the "timer" block to make mBot drive in a pattern. In the Timed Turns program in Figure 8, mBot drives forward and makes a turn every 3 seconds. **Note**: You must reset the timer every time you make a turn.



```
when mBot(mcore) starts up
reset timer
forever
    if timer < 3 then
        move forward at power 50 %
    else
        turn right at power 50 % for 0.6 secs
        reset timer
```

Figure 8 Using the "timer" block to control turns in the Timed Turns program

Try It Out

In mBlock, create the Timed Turns program (Figure 8), and upload the program to your mBot. Does mBot turn at regular intervals? How many degrees is each turn? What pattern does mBot make?

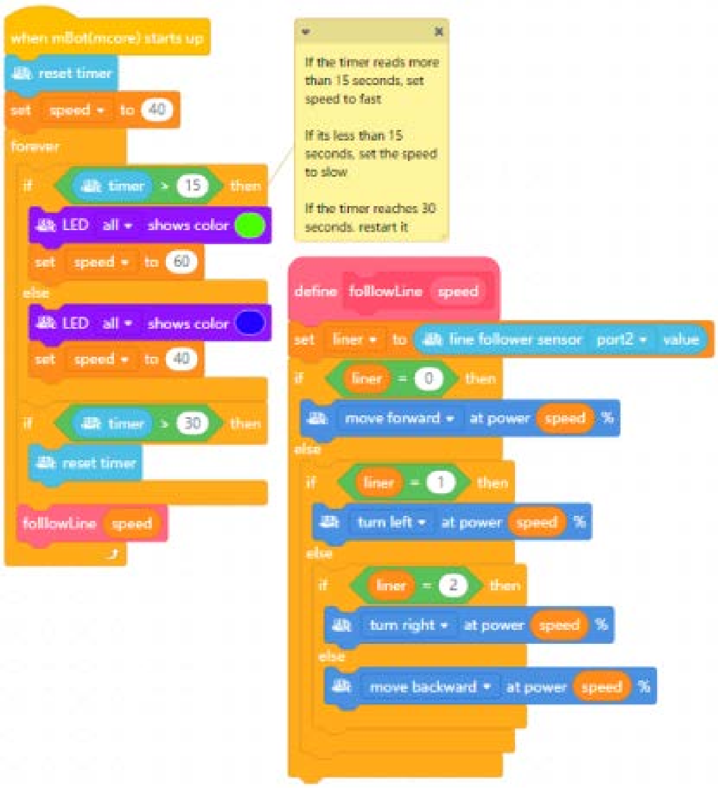
Challenge Extension

Extension 2 Make mBot Move in a Square Pattern

Start with the Timed Turns program (Figure 8) and adjust the time in the "wait" block so that mBot turns 90 degrees each time. Use the turn rate you determined in the Speed and Turn Rate program to determine the correct amount of time. Does mBot drive in a square pattern?

Timer Command: Controlling mBot’s Speed

The most common use of "timer" block is to cause an event to happen at regular intervals while other code is running. For example, in the program in Figure 9, Fast-Slow Line Following, mBot’s speed changes every 15 seconds:



```
when mBot(mcore) starts up
reset timer
set speed ▾ to 40
forever
    if timer > 15 then
        LED all ▾ shows color ●
        set speed ▾ to 60
    else
        LED all ▾ shows color ●
        set speed ▾ to 40
    if timer > 30 then
        reset timer
    followLine speed
```

If the timer reads more than 15 seconds, set speed to fast

If its less than 15 seconds, set the speed to slow

If the timer reaches 30 seconds, restart it

```
define followLine speed
set liner ▾ to line follower sensor port2 ▾ value
if liner = 0 then
    move forward ▾ at power speed %
else
    if liner = 1 then
        turn left ▾ at power speed %
    else
        if liner = 2 then
            turn right ▾ at power speed %
        else
            move backward ▾ at power speed %
```

FIGURE 9 In the Fast-Slow Line Following program, the "timer" block triggers a speed change every 15 seconds.

TRY IT OUT

In mBlock, create the Fast-Slow Line Following program (Figure 9), and upload the program to your mBot. What happens if you change the values in the "If timer > 15 then" and "If timer > 30 then" statements?

Driving Outside the Lines

Overview

During this activity, students learn to navigate their mBot using dead reckoning; in other words, controlling time the motors are on to specify the distance traveled and degrees turned. Students calculate the driving speed and turning rate of their mBots. They then use those values to create a program for mBot that allows mBot to drive in a path the shape of the letter M. Additionally, students learn how to use mBlock "timer" block in place of a "wait" block.

Objectives

- Practice writing, running, and troubleshooting mBot code.
- Determine the driving speed and turning rate of mBot.
- Apply the relationship, distance = rate × time, to determine how long to run the motors in order to make mBot move a given distance.
- Practice writing subroutines to simplify code.
- Learn how to use the "timer" block in place of the "wait" block.

Computer Science Teachers Association (CSTA) Standards

2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-AP-13	Decompose problems and subproblems into parts to facilitate design, implementation, and review of programs.
2-AP-14	Create procedures with parameters to organize code and make it easier to reuse.
2-AP-17	Systematically test and refine programs using a range of test cases.
2-AP-19	Document programs in order to make them easier to follow, test, and debug.

Materials

- mBot
- USB cable (or, for Bluetooth connection, Makeblock Bluetooth Dongle (Vernier order code MB-BLE))
- computer (PC/Mac) **or** Chromebook with mBlock software installed
- black permanent marker
- zip-ties (or similar) to attach the black marker to mBot
- large paper or plastic sheet for mBot to draw on

Tips

1. In the Electronic Resources you will find a PDF of the student pages, as well as sample programs for this activity. The PDF allows you to print the activity for your students or distribute the activity electronically. The sample programs are fully functioning; you can open them in mBlock, and then upload them to mBot. Sign in to your account at **vernier.com/account** to access the Electronic Resources.

2. Students will likely have noticed that the mBot moves faster when the batteries are fully charged. Tests have shown that the battery voltage drops quickly when it is first used after charging and then settles into a more steady, slowly-dropping voltage after a half hour or so of use. Because of this, you will get best results for this kind of dead reckoning navigation if you avoid using fully charged batteries.

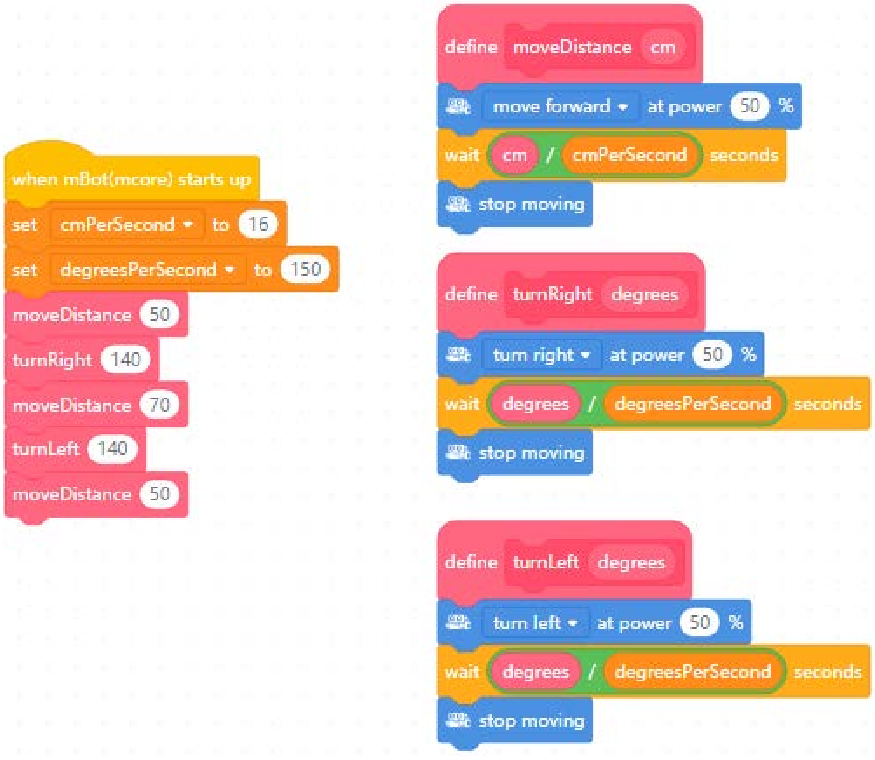
3. When students are measuring the drive speed and turning rate of their mBot, encourage them to repeat the measurements a few times and then calculate the average.

Answers to the Challenge Extensions

EXTENSION 1 Draw Another Letter

Once you have the subroutine blocks set up, it is pretty straightforward to write programs to make other letters.

Note Students may need to adjust the values for cmPerSecond and degreesPerSecond, depending on the battery state and the surface on which you are working.



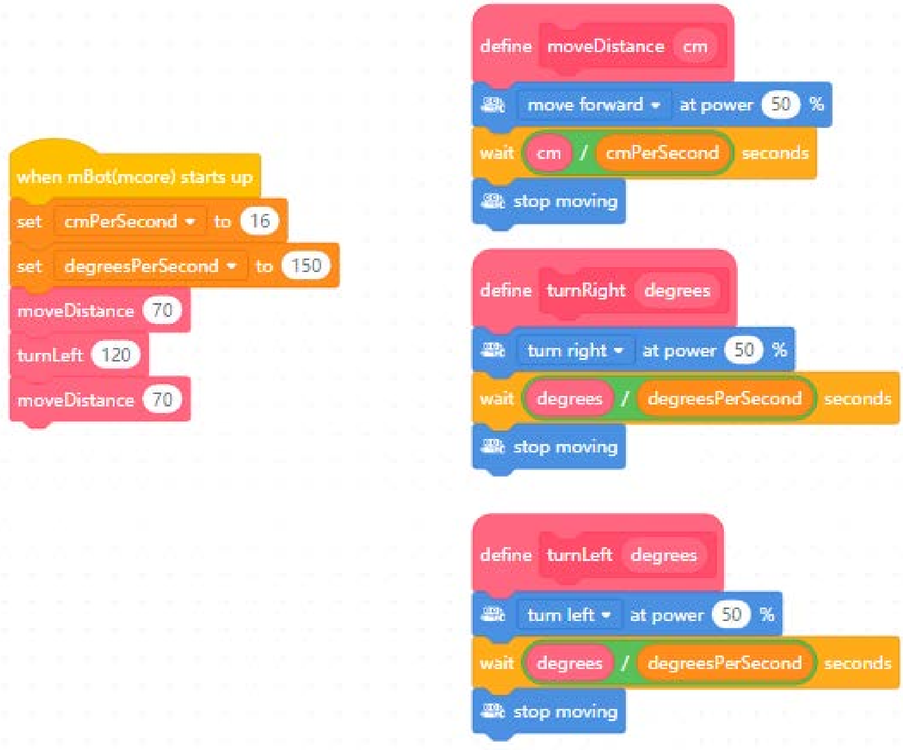
```
when mBot(mcore) starts up
set cmPerSecond ▾ to 16
set degreesPerSecond ▾ to 150
moveDistance 50
turnRight 140
moveDistance 70
turnLeft 140
moveDistance 50
```

```
define moveDistance cm
move forward ▾ at power 50 %
wait cm / cmPerSecond seconds
stop moving
```

```
define turnRight degrees
turn right ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

```
define turnLeft degrees
turn left ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

FIGURE 1 Making a Z shape



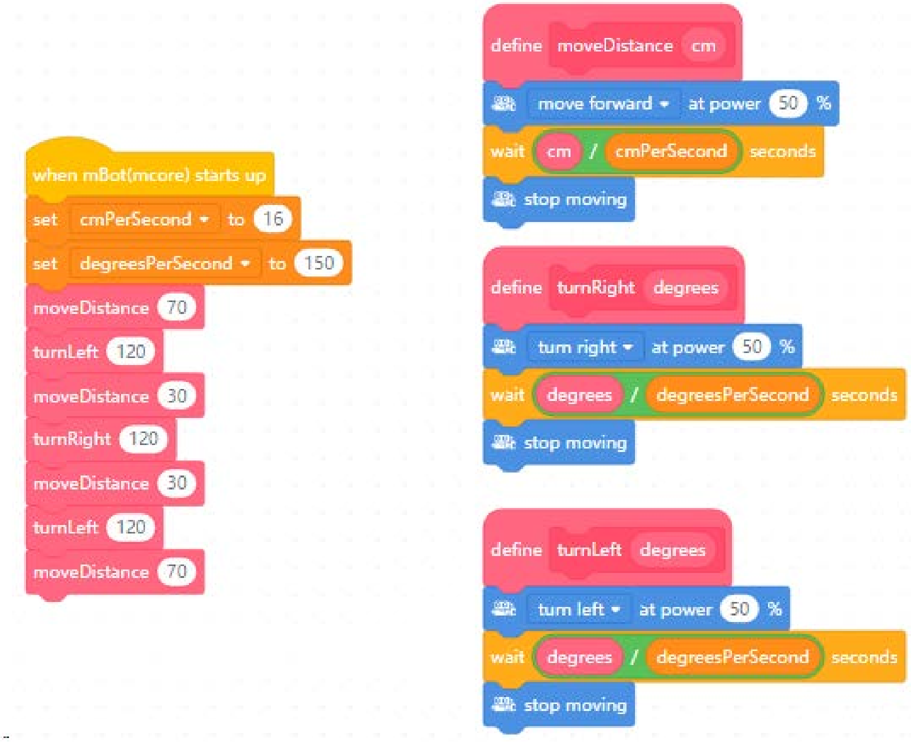
```
when mBot(mcore) starts up
set cmPerSecond ▾ to 16
set degreesPerSecond ▾ to 150
moveDistance 70
turnLeft 120
moveDistance 70
```

```
define moveDistance cm
move forward ▾ at power 50 %
wait cm / cmPerSecond seconds
stop moving
```

```
define turnRight degrees
turn right ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

```
define turnLeft degrees
turn left ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

FIGURE 2 Making a V shape



```
when mBot(mcore) starts up
set cmPerSecond ▾ to 16
set degreesPerSecond ▾ to 150
moveDistance 70
turnLeft 120
moveDistance 30
turnRight 120
moveDistance 30
turnLeft 120
moveDistance 70

define moveDistance cm
move forward ▾ at power 50 %
wait cm / cmPerSecond seconds
stop moving

define turnRight degrees
turn right ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving

define turnLeft degrees
turn left ▾ at power 50 %
wait degrees / degreesPerSecond seconds
stop moving
```

FIGURE 3 Making a W shape

EXTENSION 2 Make mBot Move in a Square Pattern

This extension is fairly simple. All students need to do is adjust the settings used in the Timed Turns program so that the turns are 90 degrees. Note the time in the turn right block controls the angle of each turn and the time in the if statement controls the size of the square.