Making the ActivityBot360 robot move

[JJK: Updated 2023 for ActivityBot 360 robots] [Updated 2026 for NRC Maze lessons JSL]

You will need SimpleIDE software installed on your computer to program the ActivityBot360 robot. Mr. La Favre will assist you in installing the software. **Please do not attempt to install the software without assistance.**

Connect the computer to the ActivityBot360

Plug the USB cable into your computer and into the ActivityBot360
Turn the power switch on the ActivityBot360 to position 1

Our 1st ActivityBot360 program, moving straight

This program will drive the ActivityBot360 forward for 2 seconds and then stop it.

```
// Our first ActivityBot program

#include "simpletools.h"   // needed for the pause() function
#include "abdrive360.h"    // needed for the drive_speed() function

int main( )
   {
   drive_speed(64, 64);    // drive both wheels at speed 64
   pause(2000);            // wheels will spin for 2 seconds (2000 ms)
   drive_speed(0, 0);      // drive both wheels at speed 0, stop
   }
```

Notes

Two arguments are required in the **drive_speed()** function. The first number is the speed of rotation of the left wheel, the second number is the speed of rotation of the right wheel. The wheels are divided into 64 "ticks." The code **drive_speed(64, 64)** means "drive both wheels forward at a speed of 64 ticks per second." When the wheels are rotating at a speed of 64 ticks per second, they are rotating at one rotation per second. The circumference of the wheel is 208 mm. When both robot wheels rotate at speed 64, the robot will travel forward at a speed of 208 mm per second. For each second of time, the robot will move forward 208 mm. If the robot moves forward for 2 seconds, the robot will travel a distance of 416 mm (2 X 208). However, this does not account for the acceleration and deceleration at the start and end of the robot movement. If you make a measurement of the distance actually traveled, it may be slightly different than 416 mm.

Our 2nd ActivityBot program, turning

This program will drive the ActivityBot forward for 2 seconds and then stop it. It will then turn around 180° and drive to the starting point and stop.

```
// Our second ActivityBot program

#include "simpletools.h"   // needed for the pause() function
#include "abdrive360.h"    // needed for the drive_speed() function

int main( )
   {
   drive_speed(64, 64);    // both wheels at speed 64
   pause(2000);            // drive for 2 seconds (2000 ms)
   drive_speed(0, 0);      // stop robot
   pause(1000);            // stay stopped for 1 second (1000 ms)
   drive_speed(32, -32);   // left wheel forward and right wheel backward
   pause(1250);            // turn for 1.25 seconds (1250 ms)
   drive_speed(64, 64);    // both wheels at speed 64
   pause(2000);            // drive for 2 seconds (2000 ms)
   drive_speed(0, 0);      // stop robot
   }
```

Notes

In a subsequent lesson you will learn more about the kinds of turns the robot can make. In this lesson you will cause the robot to make a **zero-radius** turn. This type of turn is coded by making one wheel move forward while the other wheel moves backwards, both at the same speed. The robot will rotate with a center point of rotation located at the center point between the two wheels. If we make the robot turn at wheel speeds of 32 and -32, then it should take about 1.25 seconds to turn 180 degrees. You may need to slightly adjust the pause time to make the robot turn exactly 180 degrees.

Our 3rd ActivityBot program, using the Ping sensor

This program will drive the ActivityBot forward until it senses it is 20 centimeters (about 8 inches) away from a wall and then it will stop.

```
// Our third ActivityBot program

#include "simpletools.h"          // needed for the pause() function
#include "abdrive360.h"           // needed for the drive_speed() function
#include "ping.h"                 // needed for the ping_cm() function

int main( )
   {
   while(ping_cm(17) > 20)        // robot more than 20 cm from wall in front?
      {
      drive_speed(64, 64);        // drive both wheels at speed 64
      pause(50);                  // wait a short time (50 milliseconds)
      }                           // end of while loop, go back to while line
   drive_speed(0, 0);             // stop robot
   }
```

Notes

The function **ping_cm()** returns the number of centimeters from the sensor to a wall in front of the sensor. The function takes one argument inside the parentheses: **`(ping_cm(17)`** The argument needs to be the port number that the ping sensor is connected to on the robot. In the case of GEAR robots, I have wired the front-facing ping sensor to port 17. The code **`(ping_cm(17)`** causes the robot to make a ping sensor measurement with the ping sensor connected to port 17.

This code: **`while(ping_cm(17) > 20)`** marks the start of a while loop. At the start of each loop, the robot makes a ping measurement and then compares the value of the measurement to a distance of 20 cm. If the distance measured is more than 20 cm, then the loop will continue to run, causing the robot to move forward for 50 milliseconds then make another ping measurement. If the distance measured is 20 cm or less, then the while loop terminates and the robot executes the next line under the last line of the while loop, **`drive_speed(0, 0);`** which causes the robot to stop.

File Name: FirstLessonActivityBot360.pdf
Web location: https://www.lafavre.us/robotics/FirstLessonActivityBot360.pdf
Last date edited: 3/19/26