Lab3: Move Robot

Pre-Requisite Knowledge and Skills

1. Understand basic structure of robot programming
2. Be able to write C code.
3. Be able to compile and upload a code.

Learning Objective:

1. Understand how to move Sparki robot using two motors.
2. Be able to program Sparki robot to move forward, backward, turn with an angles, and stop.
3. Understand function and its implementations.

Recommended Running Environment and Software:

1. Computers Running Windows OS, OSX, or Linux
2. SparkiDuino IDE

Instructional Material:

1. Sparki Robot
2. Instructions of this activity
3. Sample codes

Video Demonstration:

1. to be developed

Lab Assessment:

1. Exercises
2. Quiz

Lab Instructions

Before lab starts, check the batteries of Sparki and remote controller.

Sparki has two wheels to move, Left and Right. Sparki can move forward and backward using two wheels together in the same direction with same speed as shown in Figure 1.

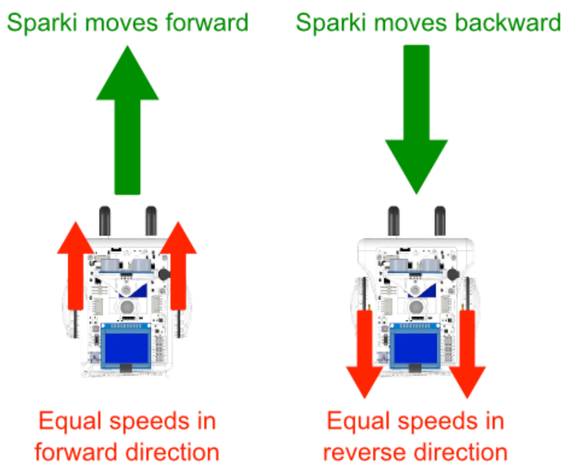
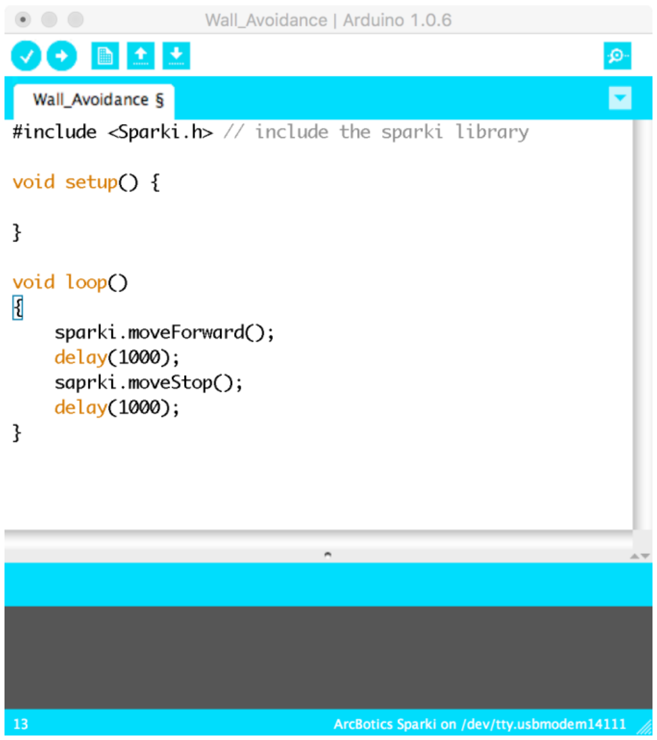

Sparki moves forward
Sparki moves backward
Equal speeds in forward direction
Equal speeds in reverse direction

Figure 1. Sparki move forward and backward

Step 1: Open the SparkiDuino and write a code as shown in Figure 2.

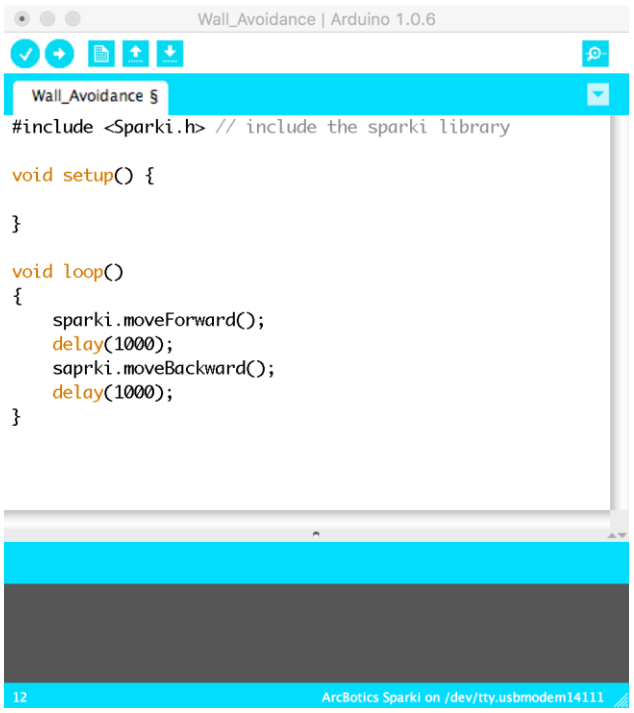

Wall_Avoidance | Arduino 1.0.6
Wall_Avoidance §
13
ArcBotics Sparki on /dev/tty.usbmodem14111

```
#include <Sparki.h> // include the sparki library

void setup() {

}

void loop()
{
    sparki.moveForward();
    delay(1000);
    saprki.moveStop();
    delay(1000);
}
```

Figure 2. Sparki move forward code

Step 2: Compile and upload the code. Then, disconnect the cable and turn the Sparki on. Observe how Sparki moves.

Step 3: Now, let's make Sparki move forward and backward over and over. Write a code as shown in Figure 3.


Wall_Avoidance | Arduino 1.0.6
Wall_Avoidance §
12
ArcBotics Sparki on /dev/tty.usbmodem14111

```
#include <Sparki.h> // include the sparki library

void setup() {

}

void loop()
{
    sparki.moveForward();
    delay(1000);
    saprki.moveBackward();
    delay(1000);
}
```

Figure 3. Sparki move forward and backward code

Step 4: Compile and upload the code. Then, disconnect the cable and turn the Sparki on. Observe how Sparki moves.

Given functions top move Sparki in "Sparki.h" header file are "sparki.moveForward(), sparki.moveBackward(), sparki.moveLeft(), sparki.moveRight(), and sparki.moveStop()".

Step 5: Let's make Sparki move more with turns. Write a code as shown in Figure 4.

```
#include <Sparki.h>                 // include the sparki library

void setup()
{
}

void loop()
{
    sparki.moveForward();                   // move the robot forward
    delay(1000);                            // wait a second (1000 milliseconds)

    sparki.moveBackward();          // move the robot backward
    delay(1000);

    sparki.moveRight(45);                   // rotate the robot clockwise in 45 degree

    sparki.moveLeft(45);            // rotate the robot counter-clockwise in 45 degree

    sparki.moveStop();              // stop all robot wheels
    delay(2000);                             // wait two seconds (2000 milliseconds)
}
```

Figure 4. Sparki move more code

Step 4: Compile and upload the code. Then, disconnect the cable and turn the Sparki on. Observe how Sparki moves.

Step 6: Write a code to move Sparki zig-zag movement as shown in Figure 5.

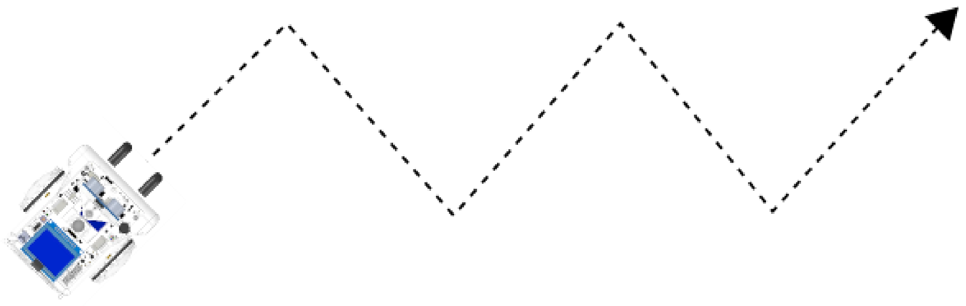


Figure 5. Zig-zag move

<u>**Discussion**</u>

- **What is function and why is/are the advantage(s) of using function?**

  Code reusability – same multiple lines of code can be reusing in any place in the code

  Code sharing – same multiple lines of code can be shared with other programmers

  Code readability – function makes code more simple in main function and look simple and easy to follow the long code

  Procedural abstraction – function works as black box and it can be known form its name and parameters that it expects

  Etc.

- **How many lines of codes are required in Step 6?**

  It needs only four lines of code if the argument is used for each function. See sample code.

- **Why minimization or simplicity is important?**

  If the code or process is too complicated, it is harder to debug or protect it. Therefore, it would be more benefit if it can be minimized or simplified.