Programming with BoeBots

Summer Workshop, McGill University

In this tutorial we will revise what was discussed in the lecture, and then use these skills to get a BoeBot to explore the world autonomously. Note that the Arduino editor program, which we use to program the BoeBot, comes with extensive online tutorials for understanding how to interface with the various sensors and actuators on the BoeBot. Feel free to refer to them at any point.

Hello World

Let's start with the simplest possible program. You will be given the following program called "HelloWorld.ino". The code has been copied in this document to explain exactly how it works.

HelloBoeBot.ino

```
void setup() {
   Serial.begin(9600); // initialize serial communication
}

void loop() {
   // Add code that repeats automatically here.
   Serial.println("Hello World!");
}
```

In the first function called *setup()*, we need to initialize the serial connection to the Arduino through the serial to USB cable connected to the computer. The argument "9600" specifies the baud rate. In general, *setup()* is where we would initialize any parameters or values for our variables. The second function *loop()* is where we put all the code that we wish to run in a continuous loop. In this case, we are simply printing an output through the serial connection to the computer. Opening the serial monitor on the Arduino editor program will display the message "Hello World!" running in a loop.

TASKS:

1. Modify the "HelloWorld.ino" program so that the BoeBot instead repeats the message every 1 second, for 10 seconds. Note that the program should stop printing anything after those 10 seconds. Hint: You will need to use the **for** loop, and the *delay()* command, which takes the time to pause in milliseconds(1000ms = 1sec).

2. Make a new program called "Multiply.ino", which prints the multiplication table of 17. The output should look something like the following:

```
17 x 1 = 17
17 x 2 = 34

..
..
17 x 10 = 170
```

3. Modify your code so that it prints multiplication table for all the integers from 1-10. You must do so without copying the code 10 times.

LED Blink

HIGH and LOW commands can be used to set the voltage on a PIN to HIGH (+5V) or LOW(0V). In this exercise, you will be using the LED that is on the Arduino board itself. To understand how to send the commands to the LED, the program needs to know which PIN is the LED connected to. With the Arduino boards, the LED is connected to PIN 13. Look through the basic Arduino examples Blink to understand how to use the LED.

TASKS:

1. Make a program called “BlinkLED.ino" by modifying the basic Arduino example program to make the LED blink by turning it on for 500ms and then off for 500ms.

2. Modify your program to read an input digit from the user, ranging from 1 to 9, which will define the speed at which the LED should blink every second. For instance, if the speed is set to 5, then the LED should be on 5 times, with a duration of 100ms every time (and 5 times off with the same duration). The input digit should be read through the serial monitor with the function *Serial.read()*. This function will return the ASCII number for the input digit. You may also find it useful to use *Serial.available()* which returns the number of bytes (characters) available for reading from the serial port.

Motion

In this section we will learn how to make the BoeBots move. BoeBots have two servo motors connected to PIN 8 (right motor) and PIN 9 (left motor). You are provided with some sample code in the program called “Square.ino”. Essentially, the program makes the BoeBot move in a square motion, so the functions *forward()* and *right()* are used in this case to move in a square motion. Notice that the program requires the Arduino Servo library for sending out commands for the left and right servos. Go through the provided code to understand the use of the function calls.

TASKS:

1. Add two new functions: Backward and Left, to go backwards and to turn left. Then, using these subroutines make the robot trace 2 squares on the ground. First using Forward and Left subroutines. Then, in the reverse using Backward and Right subroutines. NOTE: You might need to change the delays between the different function calls in order to achieve 90 degree turns. If you have good delays in your program, your robot should be able to return to its starting location.
2. The sonar sensor in front of the BoeBot is mounted on another servo motor, connected to PIN 10. This servo is used to set the direction in which we would like to the sonar to sense obstacles. Unlike the servo motors used to move the robot, this servo motor can only turn from -90 to +90 degrees. Write three subroutines, similar to Forward, Back, Right, and Left subroutines, which control the sonar direction:

   SonarCenter: Turns the sonar to center position.
   SonarRight: Turns the sonar to look to the right side.
   SonarLeft: Turns the sonar to look to the left side.

The Musician, The Explorer, and The Follower

Your BoeBot is equipped with a sonar sensor, which senses distance to an object in front of it, by measuring time it takes for a sound pulse to return after bouncing back from a surface. You will be using the Arduino example “Ping.ino” that prints out the distance (in inches and cm) that the sonar sensor detects. For the following tasks, you are also given some sample code called “Musician.ino” which should help you understand how to use the buzzer on your BoeBot.

TASKS:

1. Modify the above program so that it produces a sound with a musical note proportional to the distance. You will need to figure out the maximum distance that the sonar sees, and then map that distance to a note frequency. The choice of frequencies for the buzzer to play is up to you.

2. Make a new program called “Explorer.ino". The robot goes forward till an obstacle is less than 10cm away. It then turns 180 degrees, looks in 3 different direction using the sonar (-45, 0, 45 degrees), and then goes in the direction which has the most distant obstacle. Can you think of a way to randomize the exploration?