Lab 1 Introduction to Arduino IDE and BOE-Bot Shield Kit

Objectives:

1. Learn to use the Arduino Integrated Development Environment (IDE).
2. Write and upload the programs to the Arduino UNO R3 boards.
3. Calibrate and test the continuous rotation servo motors.
4. Test the basic navigation function of the robot

Pre-Lab requirements:

1) **Installation of Arduino IDE to your laptop if needed**
   The latest Arduino IDE is available at https://www.arduino.cc/en/Main/Software .
   Please follow the instruction online to install the software and USB driver.
2) **You need to prepare 5 AA batteries for the robot (rechargeable batteries recommended since it may consumes considerable power).**

Lab Activities:

1. Serial monitor

Open your Arduino IDE, input a sketch to display as follows:

```
/*Lab 1 example 1 */
#define MESSAGE_MAX_LEN 20

void setup()
{
  char message[MESSAGE_MAX_LEN];

  sprintf(message, "Hello, World!\n");

  Serial.begin(9600);
  Serial.write(message);
}

void loop()
{ // for codes that repeat here
}
```

Open the serial monitor to observe the output.

Lab requirement 1: Modify the program so that it continues display the "Hello, world!" once per second. You may use Arduino function *delay(1000)* to achieve the 1 second delay. Please show the demo to the instructor and include your commented program for your report.

2. Calibrate and test the servo motors

Two servo motors are included in each kit. Notice that the servo motor used here is not a standard servo motor, instead a continuous rotation motor from Parallax. The Parallax Continuous Rotation (CR) Servo has the same size and shape as a standard servo, but it is made to keep turning clockwise or counterclockwise. The control signal for a servo is a stream of pulses. The exact duration of these pulses, in fractions of a second, is what determines the position of the servo. Each pulse is from 1300 to 1700 microseconds (µs) in duration.

1300 µs: Turn clockwise
1500 µs: Stops the motor
1700 µs: Turn counterclockwise

The detailed information can be found in the following link:
http://learn.parallax.com/KickStart/900-00008

In this part, you will run a sketch that sends the "stay-still" signal to the servos. You will then use a screwdriver to adjust the servos so that they actually stay still when the pulse width is 1500 µs. This is called centering or neutralizing the servos. After the adjustment, you will run test sketches that will turn the servos clockwise and counterclockwise at various speeds.

Please connect the servo motors to the UNO board as shown in figure 1. We use pin 12 from the UNO board to control the servo motor. You may use +5V power supply from the workbench. ***Please remember to provide common ground between the power supply and the Arduino UNO board***.

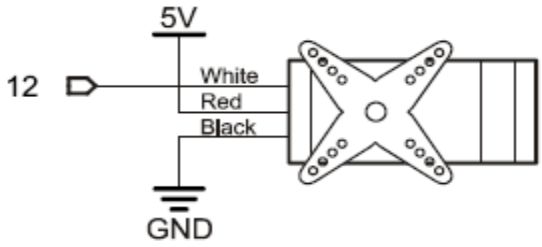

Figure 1 Connection of servo motor

Verify the connections before you upload and run the following sketch:

```
/*
 Robotics with the BOE Shield – ServoStayStill Lab 1-example 2
 Transmit the center or stay still signal on pin 12 for center adjustment.  */
```

```
#include <Servo.h>                  // Include servo library

Servo myServo;                      // Declare the servo for test

void setup()                        // Built-in initialization block
{
 myServo.attach(12);                // Attach right signal to pin 12
 myServo.writeMicroseconds(1500);   // 1.5 ms stay still signal
}

void loop()                         // Main loop auto-repeats
{                                   // Empty, nothing needs repeating
}
```

While running the program, use a screwdriver to ***gently*** adjust the potentiometer in the servo as shown in Figure 2. Don't push too hard! Adjust the potentiometer slightly until you find the setting that makes the servo stop turning, humming or vibrating. The servo should be completely still after the adjustment.

Figure 2 center the servo motor (from Parallax)

Repeat the centering procedure for the second motor.

Next we will test the servo motors for both clockwise and counterclockwise turns using the following program:

```
/*
 Robotics with the BOE Shield – ServoTest Lab 1 – Example 3
 Test the center or stay still signal on pin 12 for center adjustment.
 */

#include <Servo.h>
Servo myServo;      // Create Servo object to control the servo

void setup() {
        myServo.attach(12);                 // Servo is connected to digital pin 12
        myServo.writeMicroseconds(1700);    // Counter clockwise
        delay(3000);                        // Wait 3 seconds
        myServo.writeMicroseconds(1500);    // Stop
        delay(1000);                        // Wait 1 seconds
        myServo.writeMicroseconds(1300);    // Clockwise
        delay(3000);                        // Wait 3 seconds
```

```
        myServo.writeMicroseconds(1500);  // stay still
        }

void loop() {
}
```

Run the program and you should observe that the motor turns in counter clockwise for 3 seconds, stop for a second, then turns clockwise for 3 second.

Lab requirement 2: Please modify the pulse width to the following values and let the motor run for 30 seconds at the given pulse width: 1000 µs, 1300 µs, 1450 µs, 1550 µs, 1700 µs, 2000 µs (put the robot upside down for observation). Observe the outputs (you may use a small post it to help you to calculate) and determine the approximate RPM for each case. Do you observe any difference in RPM for different pulse width? Please describe your observation and compare the results with the figure 4 (available at http://learn.parallax.com/node/207).

Pulse width (µs)	1000	1300	1450	1550	1700	2000
Rotation per Minute (RPM) (right servo)						
Rotation per Minute (RPM) (left servo)						

3. Test the basic navigation function of the robot

In the part we will test the basic navigation functions of your BOE robot, including forward, backward, and turns. Connect the right servo motor to pin12 and the left one to pin11 via the servo port.

We will use a piezo buzzer to indicate the start of navigation, either from external reset or due to brown-out reset. When the supply voltage drops below a given level a device needs to function properly, the device may shuts down until the supply voltage returns to normal. It is called Brown-out reset and typically happens when batteries are already running low, and the servos suddenly demand more power. For example, if the BOE Shield-Bot changes from full speed forward to full speed backward, the servos need a large current to stop the servos and then go the other direction. The output voltage dips enough to cause brownout. A piezoelectric element is a crystal that changes shape slightly when voltage is applied to it. Applying high and low voltages at a rapid rate causes the crystal to rapidly change shape. The resulting vibration in turn vibrates the air around it, and this is what our ear detects as a tone. Every rate of vibration makes a different tone. Connect the piezo buzzer as shown in Figure 3.

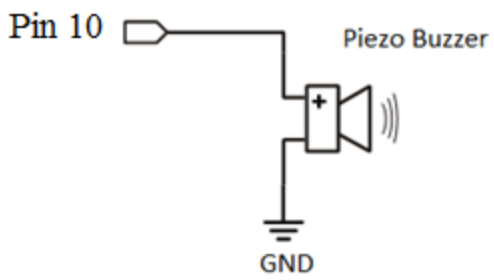

Figure 3 Piezo Buzzer Connection

Please run the following program and test your robot. The robot should move forward for 2.5 seconds, turn left, then turn right, and finally move backward to the original location.

```
/*TECH 3157 Lab 1 example 4
  Robot basic navigation: Move forward full-speed, turn left, turn right, and full-speed backward
  pin 11: control signal to Left servo
  pin 12: control signal to right servo
  pin 10: tone to the piezo buzzer */

#include <Servo.h>                    // Include servo library

void forward(unsigned int time);      //full speed forward for a duration of time (ms)
void turnLeft(unsigned int time);     // Left turn for  a duration of time (ms)
void turnRight(unsigned int time);    // Right turn for  a duration of time (ms)
void backward(unsigned int time);     // Full-speed Backward for a duration of time (ms)
void disableServos();                 // Halt servo signals

Servo servoLeft;                      // Declare left and right servos
Servo servoRight;

void setup()                          // Built-in initialization block
{
  tone(10, 3000, 1000);               // Play tone for 1 second on pin 2
  delay(1000);                        // Delay to finish tone

  servoLeft.attach(11);               // Attach left signal to pin 11
  servoRight.attach(12);              // Attach right signal to pin 12

  forward(2500);                      // Go forward for 2.5 seconds
  turnLeft(800);                      // Turn left for 0.8 seconds
  turnRight(800);                     // Turn right for 0.8 seconds
  backward(2500);                     // go backward for 2.5 seconds
```

```
  disableServos();                    // Stay still indefinitely
}

void loop()                           // Main loop auto-repeats
{                                     // Empty, nothing needs repeating
}

void forward(unsigned int time)       // Full-speed Forward function
{
  servoLeft.writeMicroseconds(1700);  // Left wheel counerclockwise
  servoRight.writeMicroseconds(1300); // Right wheel clockwise
  delay(time);                        // Maneuver for time ms
}

void turnLeft(unsigned int time)      // Left turn function
{
  servoLeft.writeMicroseconds(1300);  // Left wheel clockwise
  servoRight.writeMicroseconds(1300); // Right wheel clockwise
  delay(time);                        // Maneuver for time ms
}

void turnRight(unsigned int time)     // Right turn function
{
  servoLeft.writeMicroseconds(1700);  // Left wheel counterclockwise
  servoRight.writeMicroseconds(1700); // Right wheel counterclockwise
  delay(time);                        // Maneuver for time ms
}

void backward(unsigned int time)      // Full-speed Backward function
{
  servoLeft.writeMicroseconds(1300);  // Left wheel clockwise
  servoRight.writeMicroseconds(1700); // Right wheel counterclockwise
  delay(time);                        // Maneuver for time ms
}

void disableServos()                  // Halt servo signals
{
  servoLeft.detach();                 // Stop sending servo signals
  servoRight.detach();
}
//end of Lab 1 example 4
```

Lab requirement 4:

Read the program carefully. You may find out that the robot does not return to its exact original location after running the above example. Modify the program to ask the robot to move forward in full speed for a long period (such as 15 seconds). You may observe the robot curves to left or right slightly, instead of moving in a straight line. It is due to the slightly different RPM for the left and right servo motors. If the robot turn to left slightly, you can slow down the right servo slightly by adjust the pulse width sent to right servo (you may adjust it in step of 5 μs). Using the Figure 4 as a reference for your adjustment. Figure 4 shows the relationship between the rotational velocity in Revolution Per Minute (RPM) and the pulse width, where a negative RPM means clockwise rotation, and a positive one means counter-clockwise). Please fine tune your robot so that it can move straightly forward and backward for 15 seconds, respectively. Show the demo to the instructor. Include your observation and pulse width values needed for move straightly forward and backward for your report.

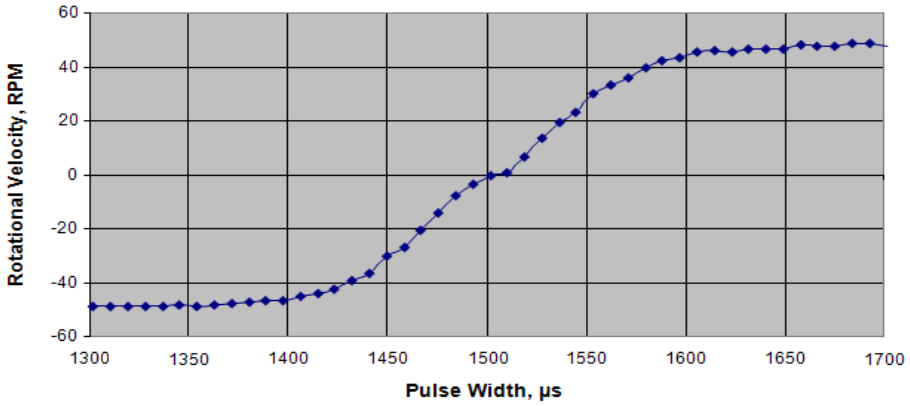

Figure 4 Transfer curve for Parallax Continuous Rotation Servo motor 900-00008 (from www. Parallax.com)