UNIVERSITY OF NORTHERN IOWA

TECH 3157 MICROCONTROLLERS APPLICATIONS SECTION 01

LAB 1 : INTRODUCTION TO ARDUINO IDE AND BOE-BOT SHIELD KIT

PROFESSOR: Dr. JIN ZHU

STUDENT: DIDAN JUNQUEIRA RIBEIRO

SEPTEMBER 17, 2015
CEDAR FALLS, IA

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.

Equipment

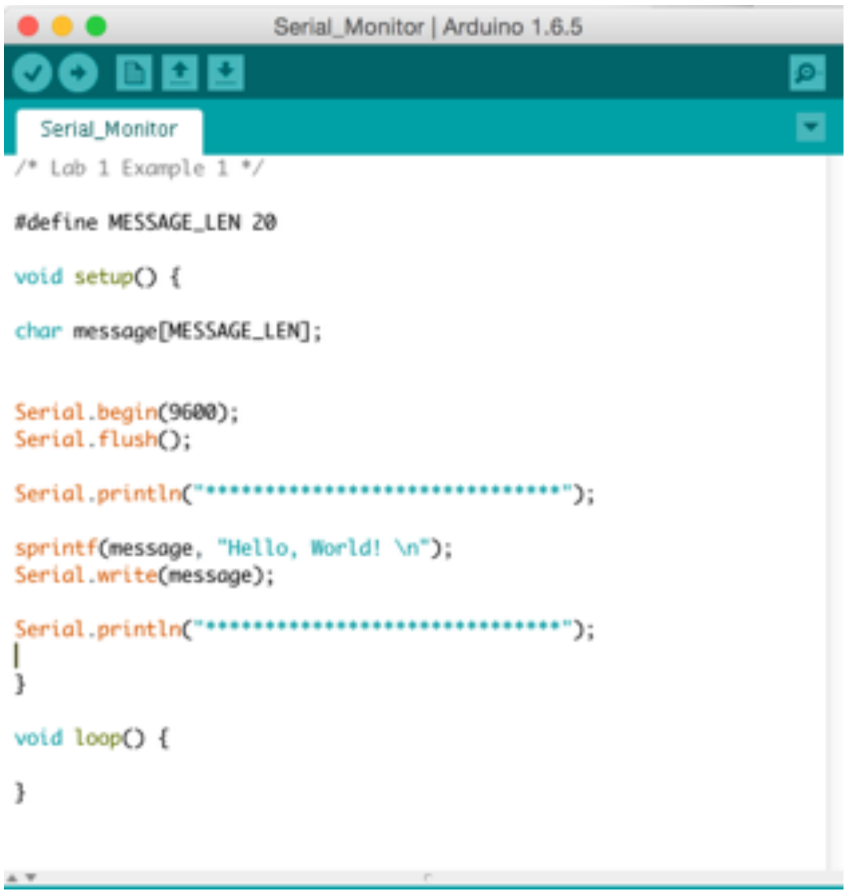
BOE-BOT Shield Kit

Theory

In this first lab, wasn’t necessary to have any previous knowledge about how to make codes on Arduino IDE. However, have a previous background should be interesting to understand the codes used on this first lab as how to print in the serial monitor, a simple message like “Hello, World”. Moreover, how to test servo motors after they get calibrated and how to test the basic navigation function of the robot using the BOE-BOT Shield Kit from Parallax.

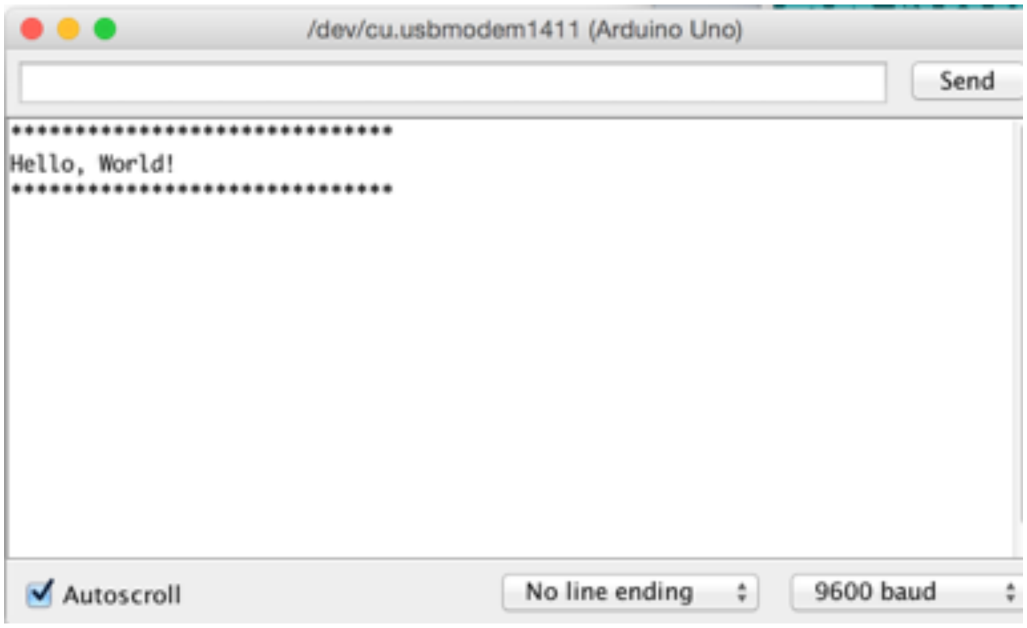
Procedure and Design

The first part of the the Lab 1 was display in the serial monitor the message "Hello, World" using the code as example available on Lab1 file. In addition, as Lab requirement 1 was modify that program so that the message "Hello, World" should continues display once per second using function delay(1000). It was done as following below:



```
/* Lab 1 Example 1 */

#define MESSAGE_LEN 20

void setup() {

char message[MESSAGE_LEN];


Serial.begin(9600);
Serial.flush();

Serial.println("********************************");

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

Serial.println("********************************");
|
}

void loop() {

}
```

Figure 1: Code to display the message "Hello, world".



```
******************************
Hello, World!
******************************
```

Figure 2: Message "Hello World" displayed on Serial Monitor

In the second part of the Lab 1 was asked to calibrate and test the servo motors included on BOE-BOT Shield Kit. As 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. If the servo is tested using 1300 µs, it will turn clockwise. If is tested using a pulse of 1500 µs, the servo stops the motor; Finally, if the servo is tested using a pulse of 1700 µs, it will turn counterclockwise. In order to calibrate the servo was sent a pulse of 1500 µs to check if both servo motors stop turning, meaning that them are calibrated if do not turn, humming or vibrate. For lab requirement 2 was asked to let the motor run for 30 seconds given pulse width: 1000µs, 1300µs, 1450µs, 1550µs, 1700µs, 2000µs. Was used the following connection of servo motor and code available on Lab1:

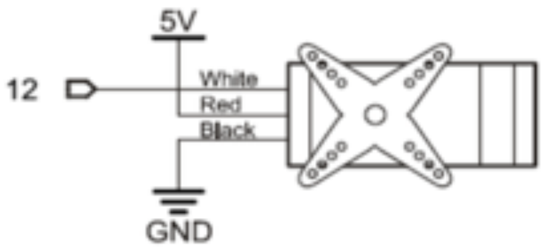
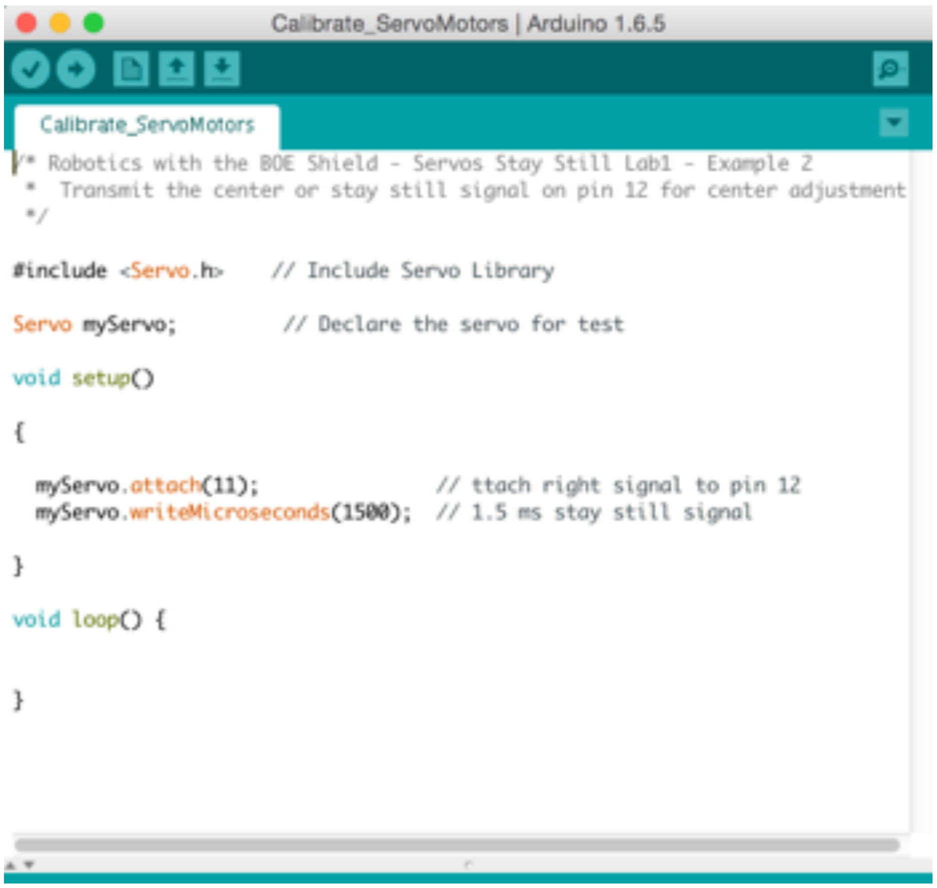


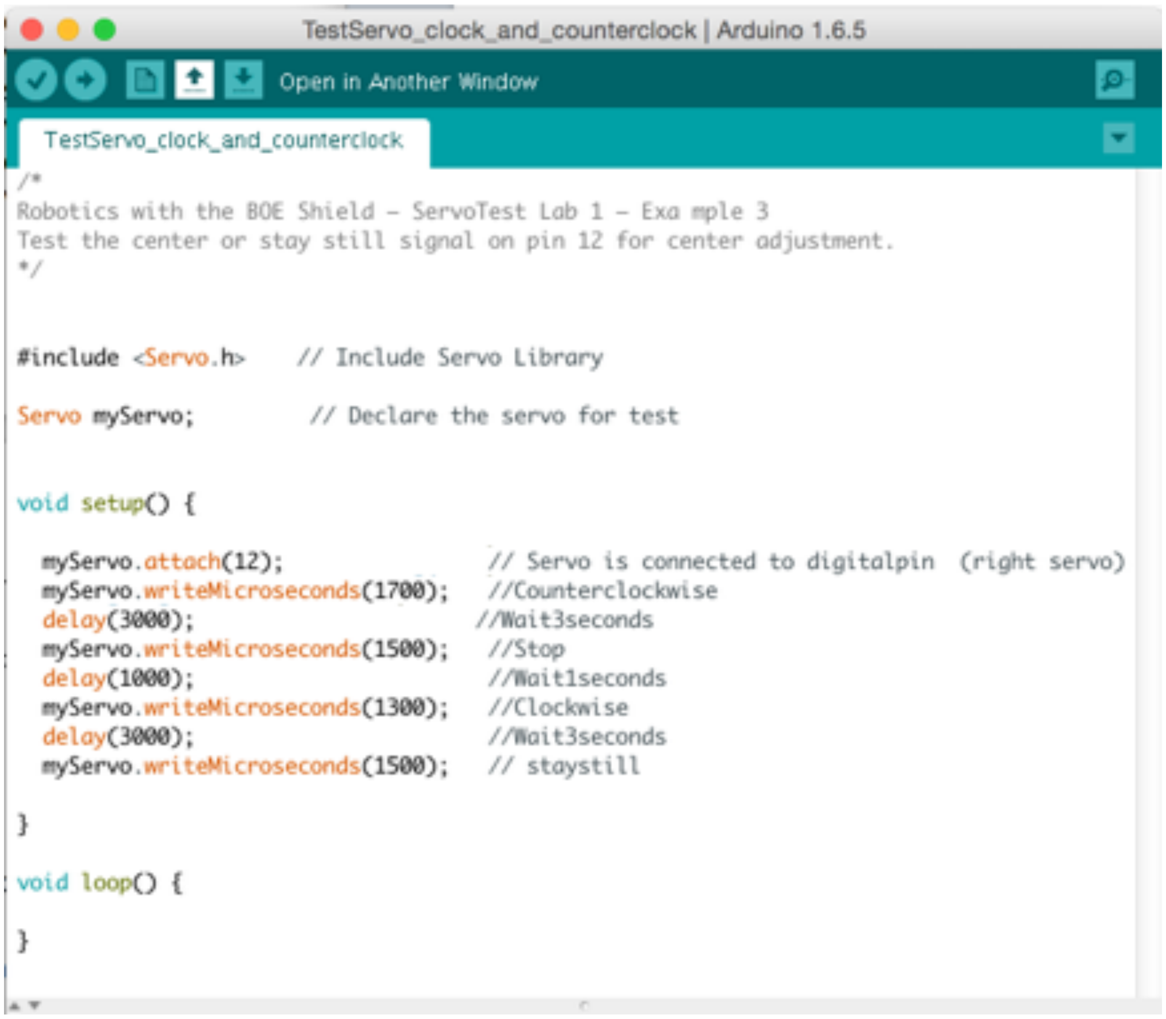
Figure 3: Connections of servo motor.



```
/* Robotics with the BOE Shield - Servos Stay Still Lab1 - 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()

{

  myServo.attach(11);                // ttach right signal to pin 12
  myServo.writeMicroseconds(1500);  // 1.5 ms stay still signal

}

void loop() {


}
```

Figure 4: Transmit the signal on pin 11 (left servo) for center adjustment. The same was done using pin 12 (right servo).

Next, was sent a signal for both servo motors to turn clockwise and counterclockwise according what the code ask.


TestServo_clock_and_counterclock | Arduino 1.6.5
Open in Another Window
TestServo_clock_and_counterclock

```
/*
Robotics with the BOE Shield - ServoTest Lab 1 - Exa mple 3
Test 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() {

  myServo.attach(12);                   // Servo is connected to digitalpin  (right servo)
  myServo.writeMicroseconds(1700);      //Counterclockwise
  delay(3000);                         //Wait3seconds
  myServo.writeMicroseconds(1500);      //Stop
  delay(1000);                          //Wait1seconds
  myServo.writeMicroseconds(1300);      //Clockwise
  delay(3000);                          //Wait3seconds
  myServo.writeMicroseconds(1500);      // staystill

}

void loop() {

}
```

Figure 5: Test the center or stay still signal on pin 12 (right servo) for center adjustment. The same was done for pin 11 (left servo).

In the third part of the lab, was asked to test the basic navigation functions of BOE robot, including foward, backward and turns. Moreover, was used a piezo buzzer to indicate the start of navigation. For Lab requirement 3, was asked to modify the program to ask the robot to move forward in full speed for a long period and use fine tune if necessary for the robot can move straightly forward and backward for 15 seconds, respectively. Was used the following Piezo Buzzer Connection and code available from Lab 1:

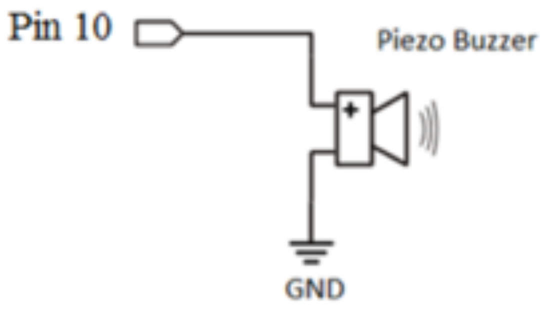


Figure 6: Piezo Buzzer Connection

Next, was used the following code for robot basic navigation:

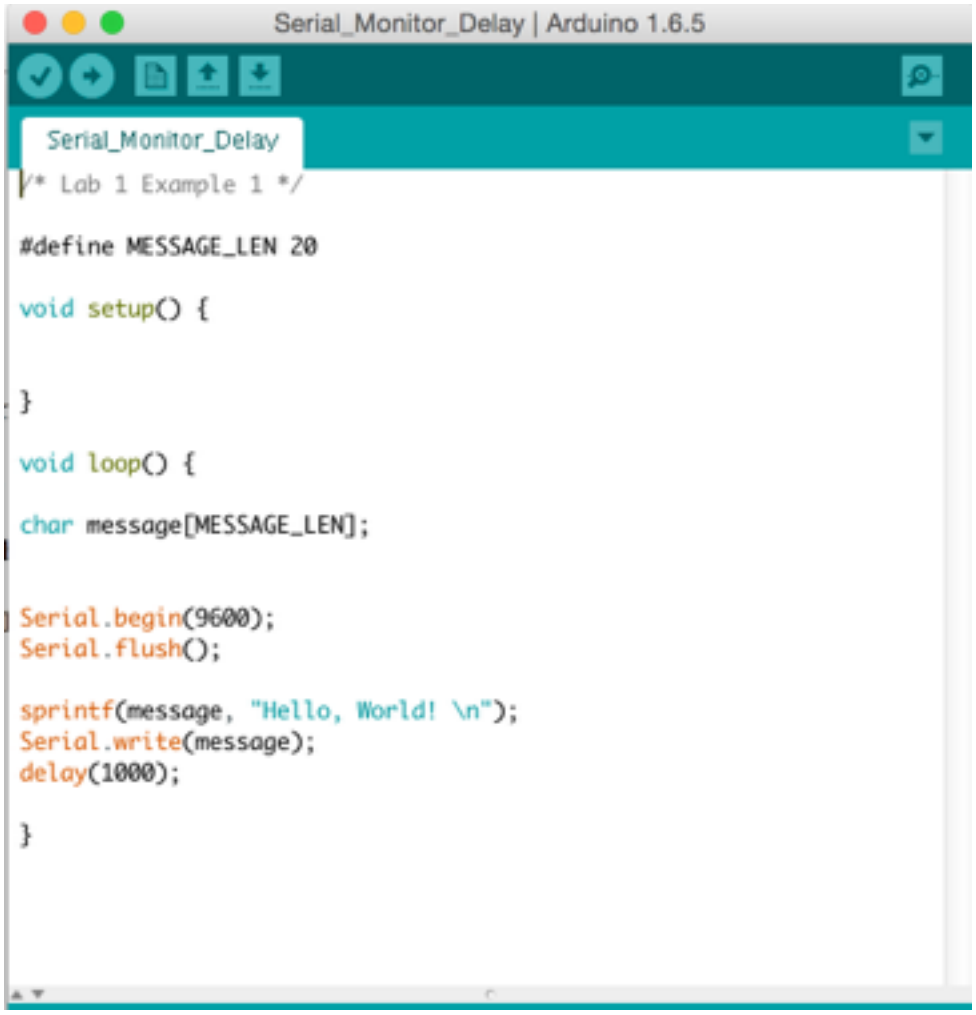
```
/*TECH 3157 Lab 1 example 4

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 servi 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() {


  tone(10,3000,1000);     // Play tone for 1 second on pin2

  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()

{

}
```

```
void forward (unsigned int time)   // Full-speed Forward function

{
servoLeft.writeMicroseconds(1700);
servoRight.writeMicroseconds(1400);
delay(time);    // Maneuver for time ms
}

void turnLeft(unsigned int time)  // Left turn function

{
servoLeft.writeMicroseconds(1300);
servoRight.writeMicroseconds(1300);
delay(time);  // Maneuver for time ms
}

void turnRight(unsigned int time)   // Right turn function

{
servoLeft.writeMicroseconds(1700);
servoRight.writeMicroseconds(1700);
delay(time);     // Maneuver for time ms
}

void backward(unsigned int time)

{
servoLeft.writeMicroseconds(1300);
servoRight.writeMicroseconds(1700);
delay(time);
}

void disableServos()

{
servoLeft.detach();
servoRight.detach();
}
// End of lab 1 Example4
```

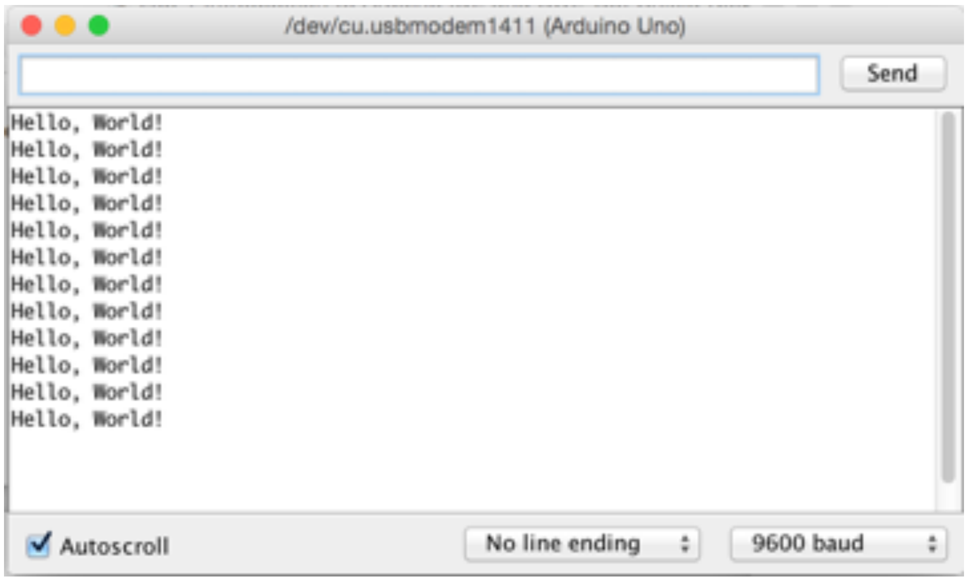
Observation and Results

The following figures below contain the results for lab requirement 1:



```
/* Lab 1 Example 1 */

#define MESSAGE_LEN 20

void setup() {


}

void loop() {

char message[MESSAGE_LEN];


Serial.begin(9600);
Serial.flush();

sprintf(message, "Hello, World! \n");
Serial.write(message);
delay(1000);

}
```

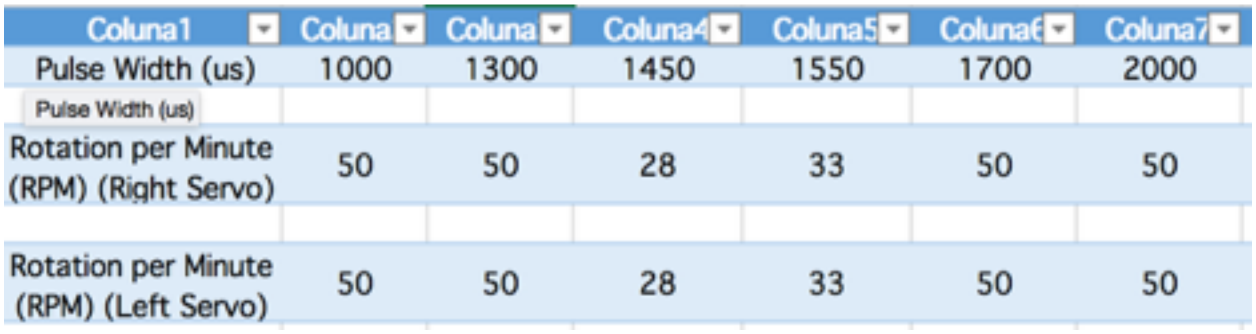
Figure 7: Code to display the message “Hello, World” once per second



```
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

Figure 8: Message “Hello World” displayed on Serial Monitor once per second

The following table contain the results for lab requirement 2:



Coluna1	Coluna	Coluna	Coluna4	Coluna5	Coluna6	Coluna7
Pulse Width (us)	1000	1300	1450	1550	1700	2000
Pulse Width (us)						
Rotation per Minute (RPM) (Right Servo)	50	50	28	33	50	50
Rotation per Minute (RPM) (Left Servo)	50	50	28	33	50	50

Table 1: Results for lab requirement 2

As displayed above, when was sent a pulse off 1000 µs, was obtained 25 rotation during 30 seconds. For 1 minute, that rotation was multiplied by 2 giving 50 rotation per minute. The same procedure was done for other pulse width. The code used was the same showed on figure 5, but testing each pin per time on the required pulse width to get one by one its RPM.

Finally, for lab requirement 3 was used the code below and the robot could forward and backward in a straight as much as possible.

```
/*TECH 3157 Lab 1 example 4

Robotbasicnavigation: Move forward full-speed,turnleft,turnright,and

full-speedbackward

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);  //Leftturnfor adurationoftime(ms)

void turnRight(unsigned int time);  //Rightturnfor adurationoftime(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() {


  tone(10,3000,1000);     // Play tone for 1 second on pin2

  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(10000);          // Go forward for 2.5 seconds

  turnLeft(0);          // Turn left for 0.8 seconds

  turnRight(0);         // Turn right for 0.8 seconds

  backward(10000);         // go backward for 2.5 seconds


  disableServos();        // Stay still indefinitely


}


void loop()

{
}

void forward (unsigned int time)   // Full-speed Forward function

{
servoLeft.writeMicroseconds(1627);
servoRight.writeMicroseconds(1300);
delay(time);    // Maneuver for time ms
}

void turnLeft(unsigned int time)  // Left turn function

{
servoLeft.writeMicroseconds(1300);
servoRight.writeMicroseconds(1300);
delay(time);  // Maneuver for time ms
}

void turnRight(unsigned int time)   // Right turn function

{
servoLeft.writeMicroseconds(1700);
```

```
servoRight.writeMicroseconds(1700);

delay(time);     // Maneuver for time ms

}


void backward(unsigned int time)


{

servoLeft.writeMicroseconds(1300);

servoRight.writeMicroseconds(1630);

delay(time);

}


void disableServos()


{

servoLeft.detach();

servoRight.detach();

}


// End of lab 1 Exampl4
```

Conclusion

All the lab parts was done successful. All the steps was followed as in the lab 1 manual and in this way was possible got all the results expected from it. It applies for lab requirements which was accomplish the goal asked for it. The unique trouble was about put the robot to forward and backward in a perfect straight line with a 100% of precision. It was impossible but a range between 97%-99% was achieved.