```
import processing.serial.*;
import cc.arduino.*;
import ddf.minim.*;

/* SpiroBot - Beta (Hardware and GUI is done, animation is working progress)
*
*  By: Nestor Sotres
*  for: Art 150, University of Illinois at Chicago
*  Last update: 6/20/14
*
*  This is a drawing robot which takes input from a GUI and operates in 3 different modes:
*    1. Demo Mode - Spirobot takes one input from GUI and executes it immediately.
*    2. Path Mode - User enters a combination of moves, Spirobot stores them, and executes them
*                one at a time forming a path
*    3. Spriograph Mode   - User enters a series of commands and Spirobot draws them several times
*                        overlapping the patterns and creating a spirograph design.
*
*  **** READ ME FIRST ****
*
*        In order to run Processing on the Arduino Uno board, you need to do some setup:
*
*        1. Run Arduino, open the Examples > Firmata > StandardFirmata sketch, and upload it to the
*                Arduino board.
*        2. Now you can run Processing sketches on the Arduino Uno board
*
*/

/* Global Variables */

// Aruduino Object
Arduino arduino;
//Minim (sound) Objects
Minim minim;
AudioPlayer player;
AudioPlayer Yoshi;
AudioPlayer Down;
AudioPlayer Tada;
AudioPlayer Applause;
AudioPlayer R2;
AudioPlayer Ready;
// Obj to hold font
PFont f;

//Variables for Screen size
int scrnX = 600;          //size in x direction
int scrnY = 600;          //size in y direction
```

```
// Strings holding messages
String topMssg = "";        //holds message at top of screen
String prompt = ">: ";      //prompt marker
String midMssg = "";        //holds message under prompt
String bottLMssg = "";      //holds message bottom left corner
String bottRMssg = "";      //holds message bottom right corner
//holds moves recorded (x value at end acts as end of list marker)
char[] movesArray = {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'x'};
int currMove = 0;           //marker for movesArray

// Variables for commands
char currComm = ' ';        //holds current command
int num = 99;               //holds command from user
int swipes = 4;             //number of times command will be repeated in Spirograph Mode
int indent = 60;            //indentation

// Menu flags
boolean start = true;       //indicates user started program
boolean intro = true;       //indicates start of program
boolean pathMode = false;   //indicates Spirobot is in Path Mode
boolean spiroMode = false;  //indicates Spirobot is in Spirograph Mode
boolean demoMode = false;   //indicates Spirobot is in Demo Mode
boolean drawBot = false;    //tell system to draw the bot or not
boolean bottomMssg = false; //unlocks bottom menu

// Variables for dealy
int totalTime = 3000;  //total time to wait (3 secs), program delay only
int savedTime;         //variable to hold time
int passedTime = 0;    //time elapsed
int moveDel = 2000;    //amount of seconds Spirobot will move (3 secs by default)
int turnDel = 1000;    //amount of seconds Spirobot will turn
//int y = 10;           //location of text
```

```
/* setup()
*
*    - Setup all paramaters for program
*    - Runs once during lifecycle of program
*/
void setup() {
        //set up size of screen
        size(scrnX, scrnY);
        background(255);        //redraw background
        // Prints out the available serial ports.
        println(Arduino.list());
        //set up arduino to work with processing
        arduino = new Arduino(this, "COM3", 57600);
        // Configure digital pins 9, 10, and 11 to control servo motors.
        arduino.pinMode(9, Arduino.SERVO);      //cont rotation, left wheel
        arduino.pinMode(10, Arduino.SERVO);     //cont rotation, right wheel
        arduino.pinMode(11, Arduino.SERVO);     //reg servo, holds color marker
        // Configure digital pin 4 and 13 for LED indicator
        arduino.pinMode(4, Arduino.OUTPUT);     //yellow LED
        arduino.pinMode(13, Arduino.OUTPUT);    //green LED
        // Configure digital pin 2 for button input
        arduino.pinMode(2, Arduino.INPUT);
        //stop both servos from moving by default
        arduino.servoWrite(9, 1510);            //stop left wheel
        arduino.servoWrite(10, 1510);           //stop right wheel
        // set servo at 90 by default
        arduino.servoWrite(11, 90);             //raise marker above ground
        println("servo shoulda moved");
        //turn LED indicator lights off by default
        arduino.digitalWrite(13, Arduino.LOW);  //turn green LED off
        arduino.digitalWrite(4, Arduino.LOW);   //turn yellow LED off
        //set up minim to play sound
        minim = new Minim(this);
        //load sound files
        player = minim.loadFile("3LowRiderR.mp2");
        Yoshi = minim.loadFile("1YoshiTheme.mp3");
        Down = minim.loadFile("2DownOnIt.mp3");
        Tada = minim.loadFile("4Tada.mp3");
        Applause = minim.loadFile("5Applause.mp3");
        R2 = minim.loadFile("6R2D2.mp3");
        Ready = minim.loadFile("7Ready.mp3");


        //font selection
        f = createFont("Arial",20,true);
        textFont(f);
        //save time at start of program
        savedTime = millis();
}
```

```
/* draw()
*    - Runs program's GUI (continuous loop by Processing)
*    - Called automatically and should never be called explicitly.
*/
void draw() {
        //set fill for entire draw program
        fill(0);                    //set color to fill shapes, 0 is black
        //display initial message (done once at start of program)
        if(intro){
                setTopMssg("Spyrobot- Alpha\nBy: Nestor Sotres.\n(s = start , x = exit).");
                //println("inside instructions loop");
                intro= false;               //ensure intro only happens once
        }
        //print top message on GUI
        text(topMssg, 40, 20);        //typically instructions
        text(prompt, 40, 110);        //user input line
        text(midMssg, 40, 130);       //message under input line
        //print bottom message on GUI
        if(bottomMssg == true){
                text(bottLMssg, 40, 550);   //bottom left corner of GUI
                text(bottRMssg, 500, 550);  //bottom right corner of GUI
        }
        //draw the GUI visual representation of Spirobot
        if(drawBot == true){
                fill(204, 102, 0);          //select fill color for body
                rectMode(CENTER);           //select mode to draw body
                rect(scrnX/2, 550, 40, 80);  //draw spirobot
        }

}
```

```
/*****************
*  GUI COMMANDS  *
******************/

/* setTopMssg(String)
*
*    - Sets the string to be displayed at top of GUI
*/
void setTopMssg(String msg){
        topMssg = msg;
        //println("inside setTopMssg()");
}

/* clearTopMssg()
*
*    - Clears the message to be displayed at top of GUI
*/
void clearTopMssg(){
        //println("inside clearTopMssg()");
        topMssg = "";
}

/* setMidMssg(String)
*
*    - Sets the string to be displayed under the prompt of GUI
*/
void setMidMssg(String msg){
        midMssg = msg;
}

/* clearMidMssg()
*
*    - Clears the message displayed under prompt of GUI
*/
void clearMidMssg(){
        midMssg = "";
}

/* setBottMssg(String)
*
*  - Sets string to be displayed at the bottom Left of GUI
*/
void setBottLMssg(String msg){
        bottLMssg = msg;
}
```

```
/* setBottRMssg(String)
*
*  - Sets string to be displayed at the bottom Right of GUI
*/
void setBottRMssg(String msg){
	bottRMssg = msg;
}

/* addMoveMidMssg(char)
*
*  - Displays current move list on GUI for Path and Spirograph modes
*/
void addMoveMidMssg(char c){
	//path mode is true
	if(pathMode == true){
		//can store up to 10 moves
		 if(currMove < 11){
			//add first move with no comma
			if( midMssg == "Moves: "){
				midMssg = midMssg + " " + c;
			}else{
				//add every other move with a comma
				midMssg = midMssg + " ," + c;
				//if 10 moves total notify user its full
				if(currMove == 10){
					midMssg = midMssg + " FULL";
				}
			}
		}
	}
	//spirograph mode is true
	if(spiroMode == true){
		//can store up to 10 moves
		if(currMove < 6){
			//add first move with no comma
			if( midMssg == "Moves: "){
				midMssg = midMssg + " " + c;
			}else{
				//add every other move with a comma
				midMssg = midMssg + " ," + c;
				//if 5 moves total notify user its full
				if(currMove == 5){
					midMssg = midMssg + " FULL";
				}
			}
		}
	}
}
```

```
/* clearScreen()
*
*    - Clears the entire GUI by redrawing the background
*/
void clearScreen(){
        background(255);        //redraw background
}

/* exitProg()
*
*      - exits program whenever user presses x
*      - released all assets from memory
*/
void exitProg(){
        clearScreen();
        player.close();
        Yoshi.close();
        Down.close();
        Tada.close();
        Applause.close();
        R2.close();
        Ready.close();
        fill(237, 43, 82);
        text("exiting...", 40, 40);
        //delayT();
        exit();
}

/* resetVals()
*
*    - resets all values in program
*/
void resetVals(){
        //clear screen
        clearScreen();
        // reset flags
        start = true;       //indicates user started program
        intro = true;       //indicates start of program
        pathMode = false;   //indicates Spirobot is in Path Mode
        spiroMode = false;  //indicates Spirobot is in Spirograph Mode
        demoMode = false;   //indicates Spirobot is in Demo Mode
        drawBot = false;    //tell system to draw the bot or not
        bottomMssg = false; //unlocks bottom menu
        //reset curr moves
        currMove= 0;        //reset current move location
        //reset moves array
        for(int i= 0; i<11; i++){
                movesArray[i] = ' ';  //clear moves stored
        }
```

```
        movesArray[10] = 'x';   //reset end marker
}

/* echoKey(char)
*
*     - Displays the character on the screen
*     - Typically the input from the user
*/
void echoKey(char k){
        fill(0);              //set color to black
        text(k, indent, 110);   //display text from user onto GUI
}

/***************
*  TIME DELAYS *
***************/

/* delayT()
*
*     - Delays the program for 3 seconds (default)
*/
void delayT(){

        //current time minus the last saved time
        passedTime = millis() - savedTime;
        //do while the diff between the current time and last saved time is less than the delay wanted
        while(passedTime < totalTime){
                //update the time that has passed
                passedTime = millis() - savedTime;
        }
        println("end of delay");      //testing purposes
        // Save the current time to restart the timer
        savedTime = millis();
}
```

```
/* delayT(int)
*
*     - Accepts an int and delays the program for that amt of time
*     - Delays the program for 3 seconds (default)
*     - Used for Spirobot movement
*/
void delayT(int del){

        //current time minus the last saved time
        passedTime = millis() - savedTime;
        //do while the diff between the current time and last saved time is less than the delay wanted
        while(passedTime < del){
                //update the time that has passed
                passedTime = millis() - savedTime;
        }
        println("end of movement delay");     //testing purposes
        // Save the current time to restart the timer
        savedTime = millis();

}

/******************
*  INPUT CONTROLS *
******************/

/* keyPressed()
*
*     - Method invoked by Processing every time a key is pressed
*     - Saves the current key pressed
*     - Calls main execution program
*/
void keyPressed() {

        if (key == '\n' ) {
                println("hit the enter key");    //testing purposes
                println(currComm);              //testing purposes
                //pressing the new line key executes the command
                commandExec(currComm);
        }else{
                //new key pressed
                clearScreen();    //clear old GUI
                currComm = key;   //store new key
                echoKey(key);     //display new key
        }

}
```

```
/* commandExec(char)
*
*      - main program used to execute all commands
*      - main control function
*      - the 3 different modes (path, spirograph, demo) have been seperated for possible expansion
*      - function first detects what mode the Spirobot is in and then creates appropriate behavior
*/
void commandExec(char com){

  switch(com){
   case 's':
           // If in the Start Menu allow the following
           if(start == true){
                   //user has indicated start running program
                   clearTopMssg();      //reset top message
                   clearMidMssg();      //reset middle message
                   clearScreen();       //clear screen
                   //delayT();
                   //echoKey(key);
                   //start = true;        //indicates user started program
                   setTopMssg("Select Mode: \n a. Draw Path b. Spirograph Mode c. Demo Mode");
                   //unlock bottom messages
                   bottomMssg = true;
                   setBottLMssg("m = Main Menu");
                   setBottRMssg("x = exit");
                   playLowRider();
                    //player.play();
                    //player.loop(2);
           }else{
                   //if you are not in the start menu just clear the screen
                   clearScreen();
                   echoKey(' ');
           }
           break;
   case 'x':
           //user has indicated exit of program
           exitProg();
           break;
   case 'm':
           //user wants to go back to main menu, reset everything
           resetVals();
           //pause all music
           pauseMusic();
           //rewind sound effects
           rewindReady();
           rewindTada();
           rewindR2();
           rewindApplause();
           break;
```

```
case 'a':
        // Path Mode Chosen from Start Menu
        if(start == true){
                clearScreen();
                // Display new menu options
                setTopMssg("Path Mode:     up to 10 moves allowed.\n(8 = forward, 2 = backward, 4 =
                        left turn, 6 = right turn)\n('enter' = save move d = done)");
                setMidMssg("Moves: ");
                //make sure you cant go back by hitting 's' key
                start = false;       //indicate out of Start Menu
                pathMode = true;     //indicate Path Mode is selected
                //turn other modes off
                demoMode = false;
                spiroMode = false;
                drawBot = true;      //tell system to draw the Spirobot
        }
        break;
case 'b':
        // Spirograph Mode Chosen from Start Menu
        if(start == true){
                clearScreen();
                //Display new menu options
                setTopMssg("Spirograph Mode:     up to 5 moves, 4 swipes made\n(8 = forward, 2 =
                        backward, 4 = left turn, 6 = right turn)");
                setMidMssg("Moves: ");
                //make sure you cant go back by hitting 's' key
                start = false;      //indicate out of Start menu
                spiroMode = true;   //indicate Spirograph Mode is selected
                //turn other modes off
                pathMode = false;
                demoMode = false;
                drawBot = true;     //tell system to draw the Spirobot
        }
        break;
case 'c':
       // Demo Mode Chosen from Start Menu
       if(start == true){
                clearScreen();
                //Display new menu options
                setTopMssg("Demo Mode:\n(8 = forward, 2 = backward, 4 = left turn, 6 = right turn)");
                //make sure you cant go back by hitting 's' key
                start = false;       //indicate out of Start menu
                demoMode = true;     //indicate Demo Mode is selected
                //turn other modes off
                pathMode = false;
                spiroMode = false;
       }
       break;
```

```
case 'd':
	//The 'd' key signals that the user is done entering commands for the Spirobot

	//if no moves were entered display message
	if(currMove == 0){
		//setMidMssg("Moves: No moves added! Please add at least 1 move.");
		fill(237, 43, 82);
		text("No moves added! Please add at least 1 move.", 40, 160);
	}else{
		// mark end of list in moves array
		currMove++;	//needed because of the way addMove(char) works
		addMove('x');	//add end of array marker to movesArray
		currMove = 11;	//stops other moves from being added
		pauseMusic();
		playReady();	//play sound "ready?"
		readyState();	//ready state waits for a button press
		//readyDelay();	//ready state is just a delay
		if(spiroMode == true){
			print("Spiro mode true, swipes: ");
			playDown();	//after user pushes button start soundtrack
			//println(swipes);
			//run the number of swipes input by user
			for(int s = swipes; s > 0; s--){
				print("swipes: ");
				println(s);
				moveExec();	//execute the moves storred in the movesArray
				//after each swhipe (except the last) repeat the moves
				if(s != 1){
					readyState();	//ready state waits for a button press
					robotSwipe();	//behavior to do after each swipe (turn right)
				}

			}
			pauseMusic();	//after all swipes are done, pause the music
			finishedYay();
		}else{
			playYoshi();	//play soundTrack
			moveExec();	//execute the moves storred in the movesArray
			finishedYay();
		}


	}
	printPath();
	break;
case 'z':
	//turn marker left
	arduino.servoWrite(11, 0);
	break;
```

```
case 'p':
	//turn marker right
	arduino.servoWrite(11, 175);
	break;
case 't':
	//raise the marker above ground
	arduino.servoWrite(11, 90);
	break;
case '2':
	// Move Spirobot Backward
	if(pathMode == true){
		//increment move counter
		currMove++;
		// Add move to display on GUI
		addMoveMidMssg('2');
		// Add move to array of moves
		addMove('2');
		//robotBkwd(moveDel);
		//lServoBack();
		//rServoBack();
	}else if(spiroMode == true){
		//increment move counter
		currMove++;
		// Add move to display on GUI
		addMoveMidMssg('2');
		// Add move to array of moves
		addMove('2');
	}else if(demoMode == true){
		pauseMusic();
		playR2();
		rewindR2();
		//immediately move robot backward
		robotBkwd(moveDel);
	}
	break;
case '4':
	// Turn Spirobot left
	if(pathMode == true){
		//increment move counter
		currMove++;
		// Add move to display on GUI
		addMoveMidMssg('4');
		// Add move to array of moves
		addMove('4');
		//robotLeft(moveDel);
	}else if(spiroMode == true){
		//increment move counter
		currMove++;
		// Add move to display on GUI
```

```
                addMoveMidMssg('4');
                // Add move to array of moves
                addMove('4');
        }else if(demoMode == true){
                pauseMusic();
                playR2();
                rewindR2();
                //immediately move robot left
                robotLeft(turnDel);
        }
        break;
case '5':
        // Stop Spirobot
        if(pathMode == true){
                robotStop();
          //lServoStop();
          //rServoStop();
        }else if(spiroMode == true){
          robotStop();
        }
        else if(demoMode == true){
          pauseMusic();
          playR2();
          rewindR2();
          //immediately stop robot
          robotStop();
        }
        break;
case '6':
        // Turn Spirobot right
        if(pathMode == true){
                //increment move counter
                currMove++;
                // Add move to display on GUI
                addMoveMidMssg('6');
                // Add move to array of moves
                addMove('6');
                //robotRight(moveDel);
        }else if(spiroMode == true){
                //increment move counter
                currMove++;
                // Add move to display on GUI
                addMoveMidMssg('6');
                // Add move to array of moves
                addMove('6');
        }else if(demoMode == true){
                pauseMusic();
                playR2();
                rewindR2();
```

```
                //immediately move robot right
                robotRight(turnDel);
        }
        break;
  case '8':
        // Move Spirobot Forward
        if(pathMode == true){
                //increment move counter
                currMove++;
                // Add move to display on GUI
                addMoveMidMssg('8');
                // Add move to array of moves
                addMove('8');
                //robotFwd(moveDel);
                //lServoFwd();
                //rServoFwd();
        }else if(spiroMode == true){
                //increment move counter
                currMove++;
                // Add move to display on GUI
                addMoveMidMssg('8');
                // Add move to array of moves
                addMove('8');
        }else if(demoMode == true){
                pauseMusic();
                playR2();
                rewindR2();
                //immediately move robot forward
                robotFwd(moveDel);
        }
        break;
  default:
         //print debugging message
         println("reached default statement in commandExec(char com)");
         break;


  }
}
```

```
/* finishedYay()
*
*  - spirobot is finished drawing
*  - play finished sounds
*  - do little dance
*/
void finishedYay(){
        pauseMusic();   //after moves are executed stop pause music
        //rewind audio
         rewindReady();
        rewindTada();
        rewindApplause();
        playTada();           // play tada! sound
        //set booleans for end dance
        boolean back = true;
        boolean spin = true;
        delayT(2500);         //pause to indicate finished
        playApplause();       //you deserve some applause!
        //execute dance ints are in milsecs (1,000 ms = 1 sec)
        if(back){
                robotBkwd(3000);  //move back for 3 seconds
                back = false;
        }
        if(back == false && spin == true){
                robotLeft(5000);
        }

        //may not need this while loop
        while(Tada.isPlaying()){
                println("tada is playing");
                //do a little dance

        }

}
```

```
/*****************
*  DATA HANDLING *
******************/

/* addMove(char)
*
*    - Adds move to the moveArray
*    - The two modes, pathMode and spiroMode have bee split for future possible expansion
*/
void addMove(char move){
        //Path mode and up to 10 moves
        if(pathMode == true && currMove < 11){
                //add the move to the movesArray
                movesArray[currMove-1] = move;
        }else if(spiroMode == true && currMove < 6){
                //Spirograph mode and up to 5 moves
                //add the move to the movesArray
                movesArray[currMove-1] = move;
        }
}

/* printPath()
*
*    - Prints the moveArray to the command line
*    - Primarily for testing purposes
*/
void printPath(){
        int cnt = 0;
        print("moves: ");
        //print moves array with standard loop
        while(movesArray[cnt] != 'x'){
                //print("(pos: ");        //testing
                //print(cnt);              //testing
                //print(" )");             //testing
                print(movesArray[cnt]);
                print(" ,");
                cnt++;
        }
        println(" end of list");
}
```

```
/* moveExec()
*
*    - Executes the moves from the moveArray
*/
void moveExec(){
          int cnt = 0;
          //light LED indicating robot is in motion
          greenLED(true);
          //go through moves array until encounter end of array marker
  while(movesArray[cnt] != 'x'){
                    //execute the encountered move
    switch(movesArray[cnt]){
      case '2':
            //Move Back
            robotBkwd(moveDel);
            break;
      case '4':
            //Move Left
            robotLeft(turnDel);
            break;
      case '6':
            //Move Right
            robotRight(turnDel);
            break;
      case '8':
            //Move Forward
            robotFwd(moveDel);
            break;
      default:
            //print debugging message
            println("reached default statment in moveExec()");
            break;

    } // end case statement
    //increment counter to next move
    cnt++;
  } //end while loop
  //turn off LED indicating robot is not in motion
    greenLED(false);
}
```

```
/*********************
*  SPIROBOT CONTROLS *
*********************/

/*robotSwipe()
*
*  - executes behaviors after each swipe in spirograph mode
*  - turn right
*  - turn is in millisecs (1,000 ms = 1 sec)
*/
void robotSwipe(){
        //for now rotate right
        robotRight(3000);
}

/*robotFwd(int)
*  -moves robot forward for 'del' secs
*  -1,000 milliseconds in a second (delay() is in millisecs)
*/
void robotFwd(int del){
        // set saved time for timing purposes
        savedTime= millis();
        //println("inside robotFwd()");
        //print("del = ");
        //println(del);
        //print("savedTime = ");
        //println(savedTime);
        //current time minus the last saved time
        passedTime = millis() - savedTime;
        //print("initial passedTime = ");
        //println(passedTime);
        //move forward while diff between current time and last saved time is less than delay wanted
        while(passedTime < del){
                //update the time that has passed
                passedTime = millis() - savedTime;
                //print("inside while loop. passedTime = ");
                //println(passedTime);
                //move robot forward
                lServoFwd();
                rServoFwd();
        }
        println("end of robotFwd(int)");      //testing purposes
        // Save the current time to restart the timer
        savedTime = millis();
        //wait x seconds
        //delayT(del);
        //stop the robot
        robotStop();
}
```

```
/*robotBkwd(int)
*
*  - moves robot backward for 'del' secs
*  - 1,000 milliseconds in a second (del is in millisecs)
*/
void robotBkwd(int del){
	// set saved time for timing purposes
	savedTime= millis();
	//current time minus the last saved time
	passedTime = millis() - savedTime;
	//move forward while the diff between current time and last saved time is less than delay wanted
	while(passedTime < del){
		//update the time that has passed
		passedTime = millis() - savedTime;

		//move robot forward
		lServoBack();
		rServoBack();
	}
	println("end of robotBkwd(int)");	//testing purposes
	// Save the current time to restart the timer
	savedTime = millis();
	//stop the robot
	robotStop();
}

/*robotLeft(int)
*  -turns robot left for 'del' secs (enough to do a quarter turn)
*  -1,000 milliseconds in a second (del is in millisecs)
*/
void robotLeft(int del){
	// set saved time for timing purposes
	savedTime= millis();
	//current time minus the last saved time
	passedTime = millis() - savedTime;
	//move left while diff between current time and last saved time is less than delay wanted
	while(passedTime < del){
		//update the time that has passed
		passedTime = millis() - savedTime;
		//move robot left
		lServoBack();
		rServoFwd();
	}
	println("end of robotLeft(int)");	//testing purposes
	// Save the current time to restart the timer
	savedTime = millis();
	//stop the robot
	robotStop();
}
```

```
/*robotRight(int)
*  - turns robot right for 'del' secs (enough to do a quarter turn)
*  - 1,000 milliseconds in a second (del is in millisecs)
*/
void robotRight(int del){
	// set saved time for timing purposes
	savedTime= millis();
	//current time minus the last saved time
	passedTime = millis() - savedTime;
	 //move right while diff between current time and last saved time is less than delay wanted
	while(passedTime < del){
		//update the time that has passed
		passedTime = millis() - savedTime;

		//move robot right
		lServoFwd();
		rServoBack();
	}
	println("end of robotRight(int)");	//testing purposes
	// Save the current time to restart the timer
	savedTime = millis();

	//stop the robot
	robotStop();
}

/*robotStop()
*  -stops both servos immediately
*/
void robotStop(){
	lServoStop();
	rServoStop();
}
```

```
/************************
*  INDIV SERVO CONTROLS *
************************/

/*lServFwd()
*  -rotates servo "forward" (relative position)
*  -1700 is max for forward motion
*/
void lServoFwd(){
        arduino.servoWrite(9, 1560);
}

/*lServoBack
*  -rotates servo "backward" (relative position)
*  -1300 is max for backward motion
*/
void lServoBack(){
        arduino.servoWrite(9, 1460);
}

/*lServoStop()
*  -stops left servo from rotating
*/
void lServoStop(){
        arduino.servoWrite(9, 1510);
}

/*rServFwd()
*  -rotates servo "forward" (relative position)
*  -1300 is max for forward motion
*/
void rServoFwd(){
        arduino.servoWrite(10, 1460);
}

/*rServoBack
*  -rotates servo "backward" (relative position)
*  -1700 is max for backward motion
*/
void rServoBack(){
        arduino.servoWrite(10, 1560);
}
```

```
/*rServoStop()
*  -stops left servo from rotating
*  -1510 is stop (calibration from manuf.)
*/
void rServoStop(){
        arduino.servoWrite(10, 1510);;  // Stop
}

/*************************
*  LED INDICATOR CONTROLS *
*************************/

/* greenLED(boolean)
*
*  - on/off switch for green LED
*  - mostly used to indicate Spirobot is in motion
*/
void greenLED(boolean s){
        //on off switch for green LED
        if(s){
                //turn on green LED
                arduino.digitalWrite(13, Arduino.HIGH);
        }else{
                //turn off green LED
                arduino.digitalWrite(13, Arduino.LOW);
        }
}

/* yellowLED(boolean)
*
*  - on/off switch for yellow LED
*  - mostly used to indicate Spirobot is in waiting
*/
void yellowLED(boolean s){
        //on off switch for yellow LED
        if(s){
                 //turn on yellow LED
                arduino.digitalWrite(4, Arduino.HIGH);
        }else{
                //turn off yellow LED
                arduino.digitalWrite(4, Arduino.LOW);
        }
}
```

```
/* readyState()
*
*      - ready state indicated by yellow LED on
*      - spirobot waiting for button to be pushed to continue
*      - basically a delay until button is pushed
*/
void readyState(){
        boolean ready = true;       //flag for while loop
        int state = Arduino.LOW;    //holds signal from push button
        yellowLED(true);            //turn on yellow LED
        println("inside readyState() button push delay");
        //pauseMusic();
        /*if(arduino.digitalRead(2) == Arduino.LOW){
                println("button is not pushed");
        }
        if(arduino.digitalRead(2) == Arduino.HIGH){
                println("button is PUSHED!!!");
        }*/
        while(ready){
                //gives the arduino a second chance at reading the pin (more responsive)
                state = arduino.digitalRead(2);
                //gives the arduino a sort of delay so that it can read the pin again and
                //not get caught in an endless loop. A sort of quirk if you will.
                //Weird, it HAS to be a print statement. Maybe because of the Java Platform
                //its running in the background.
                print(" ");
                //print("inside while loop");
                if(arduino.digitalRead(2) == Arduino.HIGH || state == Arduino.HIGH){
                        ready = false;
                        println("**** READY SHOULD BE FALSE ****");
                }
        }
        println("out of while loop");
        blinkDelay();  //blinks light indicating button push accepted
        readyDelay();  //delay allows user to back off before robot moves again
        //turn off yello LED to indicate out of ready mode
        yellowLED(false);

}
```

```
/* readyDelay()
*
*    - ready state indicated by yellow LED on
*    - after a delay of 2 secs (2000 value below) spirobot proceeds with movement
*    - basically a delay of x seconds with yellow light indicator
*/
void readyDelay(){
        //indicate in ready mode
        yellowLED(true);
        // set saved time for timing purposes
        savedTime= millis();
        //current time minus the last saved time
        passedTime = millis() - savedTime;
        //do while the diff between the current time and last saved time is less than the delay wanted
        while(passedTime < 2000){
                //update the time that has passed
                passedTime = millis() - savedTime;
        }
        println("end of readyDelay())");      //testing purposes
        // Save the current time to restart the timer
        savedTime = millis();
        //indicate end of ready mode
        yellowLED(false);

}

/* blinkDelay()
*
*    - blinks yellow LED 3 times
*/
void blinkDelay(){
        //blink light 3 times
        for(int i = 0; i < 3; i++){
                yellowLED(false);
                delayT(300);     //allows light to flicker
                yellowLED(true);
                delayT(300);     //allows light to flicker
        }

}
```

```
/*****************
*  AUDIO CONTROLS *
*****************/

/* pauseMusic()
*
*    - simply pauses all music
*/

void pauseMusic(){
        if(player.isPlaying()){
                player.pause();
        }
        if(Yoshi.isPlaying()){
                Yoshi.pause();
        }
        if(Down.isPlaying()){
                Down.pause();
        }
        if(Tada.isPlaying()){
                Tada.pause();
        }
        if(Applause.isPlaying()){
                Applause.pause();
        }
        if(R2.isPlaying()){
                R2.pause();
        }
        if(Ready.isPlaying()){
                Ready.pause();
        }
}
```

```
/* Play individual sounds*/
void playLowRider(){
	player.play();
}
void playYoshi(){
	Yoshi.play();
}
void playDown(){
	Down.play();
}
void playTada(){
	Tada.play();
}
void playApplause(){
	Applause.play();
}
void playR2(){
	R2.play();
}
void playReady(){
	Ready.play();
}

/*Rewind individual sounds*/
void rewindReady(){
	Ready.rewind();
}
void rewindTada(){
	Tada.rewind();
}
void rewindApplause(){
	Applause.rewind();
}
void rewindR2(){
	R2.rewind();
}
```