```
#include <Servo.h>

/* SpiroBot
*
*  By: Nestor Sotres
*  for: Art 150, University of Illinois at Chicago
*  Last update: 5/1/14
*
*  Description: Spirobot draws 3 different patterns which are generated randomly. It can also switch
colors
*  while its drawing the pattern and increases the size of the pattern randomly.
*
*  Video Presentation: http://youtu.be/DtZQc3AuFI4
*
*  Note: You may have to recalibrate the servos either manually or programmatically.
*          The funciton readyState() in particular may need recalibration.
*/



/* Global Variables */

// Servo Objects
Servo markerServo;  //servo motor holding the marker
Servo leftServo;    //servo motor on left wheel
Servo rightServo;   //servo motor on right wheel

// Variables for commands
int swipes = 1;             //number of times command will be repeated in Spirograph Mode
int sqr = 3;                //square side length val
int tri = 2.5;              //triangle side length val
float growth = 0.0;         //amount that sides length will grow by
int randComm1 = 0;          //random command
int randComm2 = 0;          //random command
int randComm3 = 0;          //random command
int randComm4 = 0;          //random command

// Menu flags
boolean start = true;       //indicates user started program

// Action flags
boolean swapColor = false;  //used for switching colors

// 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 (2 secs by default)
int turnDel = 1000;    //amount of seconds Spirobot will turn
```

```
/* setup()
*
*    - Setup all paramaters for program
*    - Runs once during lifecycle of program
*/
void setup() {
        Serial.begin(9600);      //open serial port

        // Configure digital pins 9, 10, and 11 to control servo motors.
        leftServo.attach(9);          //cont rotation servo on lef wheel
        rightServo.attach(10);        //cont rotation servo on right wheel
        markerServo.attach(11);       //reg servo on marker

        // Configure digital pin 4 and 13 for LED indicator
        pinMode(4, OUTPUT);
        pinMode(13, OUTPUT);
        // Configure digital pin 2 for button input
        pinMode(2, INPUT);
        //stop both servos from moving by default
        leftServo.write(1510);
        rightServo.write(1510);
        // set servo at 90 by default
        markerServo.write(90);        //elevate marker so no tip touches ground

        //turn LED indicator lights off by default
        digitalWrite(13, LOW);
        digitalWrite(4, LOW);

        //save time at start of program
        savedTime = millis();

}

/* loop()
*    - Runs main program(behavior) in a loop
*    - Called automatically and should never be called explicitly.
*/
void loop() {

        //Ensure Spirobot is in a stopped and ready state
        if(start == true){
                robotStop();              //stop all activity by Spirobot
                markerServo.write(90);    //marker elevated and not touching ground
                //wait for button push
                readyState();
                //set variable indicating Spirobot is no longer at the initial set up
                start = false;
        }
```

```
	/* Draw a SQUARE of random size a random amount of times */
	squarePattern();

	//signify Spirobot is no longer drawing
	greenLED(false);	//turn off green LED
	//wait for push of button to draw next pattern
	readyState();

	/* Draw a TRIANGLE of random size a random amount of times */
	trianglePattern();

	//signify Spirobot is no longer drawing
	greenLED(false);	//turn off green LED
	//wait for push of button to draw next pattern
	readyState();

	/* Draw a Random PATTERN of random size a random amount of times */
	randomPattern();

	//signify Spirobot is no longer drawing
	greenLED(false);		//turn green LED off
	//wait for push of button to start drawing patterns all over again
	readyState();

}

/****************
* PATTERN SET UP *
****************/

/* squarePattern()
*
*   - Sets up variables and number of times to draw pattern
*   - Makes appropriate calls to draw a square pattern
*/
void squarePattern(){

	//get random length of sides (between 2 and 4)
	sqr = getRandom(millis(), 2, 4);
	//get random amount side length will grow by (between 1 and 7)
	growth = getRandom(millis(), 1, 7);
	growth = growth/10;	//reduce growth to a decimal for gradual growth
	//get random number of times to draw the square (between 3 and 6)
	swipes = getRandom(millis(), 3, 6);
	//draw square x times (swipes)
	for(int i=0; i<swipes; i++){
		//signify the Spirobot is drawing
		greenLED(true);		//turn green LED on
		// Swap the marker color
		switchColor();
		drawSquare(sqr, sqr);    //drawing function, sending same measurement for square
		//increase the size of the square's sides
		sqr= sqr+ growth;
	}
}

/* trianglePattern()
*
*    - Sets up variables and number of times to draw pattern
*    - Makes appropriate calls to draw a triangle pattern
*/
void trianglePattern(){

	//get random length of sides (between 1 and 5)
	tri = getRandom(millis(), 1, 5);
	//get random amount side length will grow by (between 2 and 6)
	growth = getRandom(millis(), 2, 6);
	growth = growth / 10;    //reduce growth to a decimal for gradual growth
	//get random number of times to draw the triangle (between 3 and 5)
	swipes = getRandom(millis(), 3, 5);
	//draw triangle x times (swipes)
	for(int i=0; i<swipes; i++){
		//signify the Spirobot is drawing
		greenLED(true);          //turn green LED on
		// Swap the marker color
		switchColor();
		drawTriangle(tri);      //drawing function for triangle
		//increase the size fo the triangle's sides
		tri = tri+ growth;
	}
}

/* randomPattern()
*
*    - Sets up variables and number of times to draw pattern
*    - Makes appropriate calls to draw a random pattern
*/
void randomPattern(){

	//get random commands for Spirobot to draw a random pattern

	randComm1 = getRandom(millis(), 0, 6);    //generate a random command
	 /*print command attained to console (testing)
	Serial.println("commands: ");
	Serial.print(randComm1);
	*/
```

```
    //delay program for 5 ms to ensure random seed is different enough
    delay(5);
     randComm2 = getRandom(millis(), 0, 6);    //generate a random command
    //print command attained to console (testing)
    //Serial.print(randComm2);
    //delay program for 3 ms to ensure random seed is different enough
    delay(3);
    randComm3 = getRandom(millis(), 0, 6);    //generate a random command
    //print command attained to console (testing)
    //Serial.print(randComm3);
    //delay program for 7 ms to ensure random seed is different enough
    delay(7);
    randComm4 = getRandom(millis(), 0, 6);    //generate a random command
    //print command attained to console (testing)
    //Serial.print(randComm4);
    //get random number of times (swipes) to draw the pattern (between 2 and 5)
    swipes = getRandom(millis(), 2, 5);
    /*print number of swipes (testing)
    Serial.print("  swipes: ");
    Serial.print(swipes); */
    //execute random command x times (swipes)
    for(int i = 0; i<swipes; i++){
        //signify the Spirobot is drawing
        greenLED(true);          //turn green LED on
        //swap the marker color
        switchColor();
        //execute each command once
        commandExec(randComm1);
        commandExec(randComm2);
        commandExec(randComm3);
        commandExec(randComm4);
    }
}


/****************
* DRAW COMMANDS *
****************/

/* drawSquare(int, int)
*
*  - draws a square with values for x and y axis
*/
void drawSquare(int x, int y){
    int a = x * 1000;   //set up seconds to go in x axis
    int b = y * 1000;   //set up seconds to go in y axis
    int turn = 800;     //rotate for 0.8 secs
```

```
        //draw the square
        for(int i=0; i < 2; i++){
                robotFwd(b);        //straight line y axis
                rotateRight(turn);  //rotate 90 deg
                robotFwd(a);        //straight line x axis
                rotateRight(turn);  //rotate 90 deg
        }
}

/* drawTriangle(int)
*
*  - draws a trinagle (sharper angles)
*  - int is the length of the sides of the triangle
*/
void drawTriangle(int x){
        x = x* 1000;         //set up seconds to go straight
        int turn = 1270;     //how many milliseconds the turn will last

        //draw the triangle
        robotFwd(x);        //straight line
        rotateLeft(turn);   //left turn
        robotFwd(x);        //straight line
        rotateLeft(turn);   //left turn
        robotFwd(x);        //straight turn
        rotateLeft(turn);   //left turn completing triangle
}

/* getRandom(long, int, int)
*
*  - returns a random integer number
*/
int getRandom(long seed, int minVal, int maxVal){
        int val = 0;                      //variable to store random number
        randomSeed(seed);                 //set seed
        val = random(minVal, maxVal);      //get and store random value
        return val;
}
```

```
/*****************
*  SYSTEM COMMANDS  *
******************/

/* exitProg()
*
*     - exits program and closes serail connectio
*/
void exitProg(){
        //closing commands [ end() closes serial connection http://arduino.cc/en/Serial/End )
        Serial.println("exiting...");
        delayT();
        Serial.end();
}

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

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

        //current time minus the last saved time
        passedTime = millis() - savedTime;

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

}

/* 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 set to repeat delay loop
    int state = LOW;         //holds button signal
    yellowLED(true);         //yello light indicates currently in delay state
    //enter delay loop and wait for button push from user
     while(ready){
        //stop both servos from moving by default
        leftServo.write(1520);
        rightServo.write(1510);
        //gives the arduino a second chance at reading the pin (more responsive)
        state = digitalRead(2);
        //if button is pushed, exit delay loop
        if(digitalRead(2) == HIGH || state == HIGH){
            ready = false;         //set boolean flag to exit loop
        }
    }
    //warn user that Spirobot will draw again
    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;
        }
        //Serial.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
        }
}
```

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

/* commandExec(int)
*
*      - main program used to execute all commands
*      - main control function
*      - the 3 different modes (path, spirograph, demo) have been seperated for possible expansion
*/
void commandExec(int com){

  switch(com){
          case 0:
                    // Swap the color
                    switchColor();
                    break;
          case 1:
                    // Move Spirobot Backward
                    robotBkwd(moveDel);
                    break;
          case 2:
                    // Turn Spirobot left
                    rotateLeft(turnDel);
                    break;
          case 3:
                    // Stop Spirobot
                    robotStop();
                    break;
          case 4:
                    // Turn Spirobot right
                    rotateRight(turnDel);
                    break;
          case 5:
                    // Move Spirobot Forward
                    robotFwd(moveDel);
                    break;
          default:
                    // Print debugging message
                    Serial.println("reached default statemet in commandExec(char com)");
                    break;
          }//end switch
}
```

```
/*************************
*  COLOR MARKER CONTROLS *
*************************/

/* switchColor()
*
*  - switches between markers when called upon
*/
void switchColor(){

        if(swapColor == true){
                //switch to left
                markerServo.write(0);
                swapColor = false;     //swap flag
        }else if(swapColor ==false){
                //switch to right
                markerServo.write(175);
                swapColor = true;      //swap flag
        }
}

/* noColor()
*
*   - centers the servo so the robot is not actively drawing
*/
void noColor(){
        markerServo.write(90);    //servo at 90deg raises marker
}

/*********************
*  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
        rotateRight(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();
        //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;
                //move robot forward
                lServoFwd(1560);
                rServoFwd(1460);
        }
        // Save the current time to restart the timer
        savedTime = millis();
        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;
        //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;

                //move robot forward
                lServoBack(1460);
                rServoBack(1560);
        }
        // Save the current time to restart the timer
        savedTime = millis();

        //stop the robot
        robotStop();
}
```

```
/*rotateLeft(int)
*  -turns robot left for 'del' secs (enough to do a quarter turn)
*  -1,000 milliseconds in a second (del is in millisecs)
*/
void rotateLeft(int del){
        // set saved time for timing purposes
        savedTime= millis();
        //current time minus the last saved time
        passedTime = millis() - savedTime;
        //turn 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;

                //create a left turn
                lServoBack(1460);    //move left wheel back
                rServoFwd(1460);     //move right wheel forward
        }
        // Save the current time to restart the timer
        savedTime = millis();

        //stop the robot
        robotStop();
}

/*rotateRight(int)
*  -turns robot right for 'del' secs (enough to do a quarter turn)
*  -1,000 milliseconds in a second (del is in millisecs)
*/
void rotateRight(int del){
        // 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 < del){
                //update the time that has passed
                passedTime = millis() - savedTime;

                //move robot right
                lServoFwd(1560);
                rServoBack(1560);
        }
        // Save the current time to restart the timer
        savedTime = millis();

        //stop the robot
        robotStop();
}
```

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

/* finishedYay()
*
*  - spirobot is finished drawing
*  - do little dance
*/
void finishedYay(){

        //set booleans for end dance
        boolean back = true;
        boolean spin = true;
        delayT(2500);        //pause to indicate finished
        //execute dance ints are in milsecs (1,000 ms = 1 sec)
        if(back){
                robotBkwd(2000);  //move back for 2 seconds
                back = false;
        }
        if(back == false && spin == true){
                rotateLeft(5000);
        }
}
```

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

/*lServFwd(int)
*  -rotates servo "forward" (relative position)
*  -1700 is max for forward motion (1560 was default)
*/
void lServoFwd(int val){
        leftServo.write(val);
}

/*lServoBack(int)
*  -rotates servo "backward" (relative position)
*  -1300 is max for backward motion (1460 was default)
*/
void lServoBack(int val){
        leftServo.write(val);
}

/*lServoStop()
*  -stops left servo from rotating
*/
void lServoStop(){
        leftServo.write(1510);
}

/*rServFwd(int)
*  -rotates servo "forward" (relative position)
*  -1300 is max for forward motion (1460was default)
*/
void rServoFwd(int val){
        rightServo.write(val);
}

/*rServoBack(int)
*  -rotates servo "backward" (relative position)
*  -1700 is max for backward motion (1560 was default)
*/
void rServoBack(int val){
        rightServo.write(val);
}
```

```
/*rServoStop()
*  -stops left servo from rotating
*  -1510 is stop (calibration from manuf.)
*/
void rServoStop(){
        rightServo.write(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
                digitalWrite(13, HIGH);
        }else{
                //turn off green LED
                digitalWrite(13, 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
                digitalWrite(4, HIGH);
        }else{
                //turn off yellow LED
                digitalWrite(4, LOW);
        }
}
```