```
1  //Main changes from the L298 test code: updated and commented certain variables
     for clarity (please comment your code, folks), added line follower logic to
     keep the robot on-course
2
3
4
5  #include <Encoder.h>
6  #include <Servo.h>
7  //Encoder backRight(2,29);
8  //Encoder backLeft(3,31);
9  Encoder frontRight(18,17);
10 Encoder frontLeft(19,20);
11
12 float enc_br = 1;
13 float enc_bl = 0.84;
14 float enc_fr = 0.8;
15 float enc_fl = 1;
16
17 int mpl = 255; //Left Motor Power
18 int mpr = 255; //Right Motor Power
19 int kpfor; //Proportional control variable for the forward motion of the robot
20
21 int left_speed = 0; //Output variable to control the speed of the left motors
22 int right_speed = 0; //Output variable to control the speed of the right motors
23 int left_back_speed = 0;
24 int right_back_speed = 0;
25 int LimitHit;
26
27 int error = 0; // Error variable. When this value is 0, the robot is centered on
     the line. This value becomes greater than 0 when the robot is too far to the
     right and becomes less than 0 when the robot is too far to the left.
28 int shooterPower = 0;
29 int diag2trigger = 0;
30 int backDiagTrigger = 0;
31
32 //Define all states. Note that certain numbers such as 107 are unused - these
     were from obsolete or otherwise unused states.
33 const int HOME = 101; //All motors stopped, ignore all sensors
34 const int FORWSIDE = 102; //Move sideways, switch to FORWARD upon hitting center
     line
35 const int FORWLINE = 103;  //Move forward along centerline, adjusting bearing as
     needed. Switch to STOP upon hitting half-court line
36 const int BACKLINE = 106; //Move the robot back to the home position to be reset
37 const int BACKUP = 108; //Back the robot up from the half-court line until the
     left and right sensors no longer detect a line. This is to make sure no part of
      the robot is sticking over the side of the half-court line
38 const int STOP = 109; //Stop all movement and prepare the launcher
39 const int FORWSIDE2 = 110; //Same as FORWSIDE but for the opposite court
40
41 int balls = 0; //iterative value to count the number of balls launched during a
     given "SHOOT" state
42
```

```
43
44  Servo LoadServo;
45
46  int state = 101;
47  //PushButton Pins (Ultimately, these pins were not used. We opted to instead
      modify the code and upload)
48  //int LeftCourt = 16, RightCourt = 17, LeftCourt_in, RightCourt_in;
49
50  //Line Follower Sensors Pins
51  int LSense0pin = 39, LSense1pin = 35, LSense2pin = 37; //LS0 is center-left, LS1
      is center, LS2 is center-right (tentative)
52  int LSense3pin = 33; //LS3 is far left sensor
53  int LSenseRpin = 31; //LSR is far right sensor
54
55  int LS[5] = {0,0,0,0,0}; //Overall line sensor data array
56
57  int BRpin = 10; //Pin going to back right motor
58  int FRpin = 5; //Pin going to front right motor
59  int BLpin = 11; //Pin going to back left motor
60  int FLpin = 6; //Pin going to front left motor
61  int FR1pin = 22; //L298 pin for the front right motor. Set to HIGH, this should
      cause the motor to spin forwards
62  int FR2pin = 24; //L298 pin for the front right motor. Set to HIGH, this should
      cause the motor to spin backwards
63  int FL1pin = 26; //L298 pin for the front left motor. Set to HIGH, this should
      cause the motor to spin forwards
64  int FL2pin = 28; //L298 pin for the front left motor. Set to HIGH, this should
      cause the motor to spin backwards
65  int BR1pin = 44; //L298 pin for the back right motor. Set to HIGH, this should
      cause the motor to spin forwards
66  int BR2pin = 46; //L298 pin for the back right motor. Set to HIGH, this should
      cause the motor to spin backwards
67  int BL1pin = 48; //L298 pin for the back left motor. Set to HIGH, this should
      cause the motor to spin forwards
68  int BL2pin = 50; //L298 pin for the back left motor. Set to HIGH, this should
      cause the motor to spin backwards
69
70  int LimitSwitchPin = 13;
71
72  //void PID_Control()
73  //{
74  //
75  //}
76  void setup() {
77    // put your setup code here, to run once:
78    Serial.begin(9600);
79
80    pinMode(BRpin, OUTPUT); //Back Right PWM
81    pinMode(FRpin, OUTPUT); //Front Right PWM
82    pinMode(BLpin, OUTPUT);  //Back Left PWM
83    pinMode(FLpin, OUTPUT);  //Front Left PWM
84
```

```
 85
 86    //pinMode(28, OUTPUT);
 87    //pinMode(30, OUTPUT);
 88
 89
 90  }
 91
 92  int count[3];
 93  int power = 0;
 94
 95  void loop() {
 96    // put your main code here, to run repeatedly:
 97
 98    // Line Sensor Readings
 99    LS[0] = digitalRead(LSense0pin);
100    LS[1] = digitalRead(LSense1pin);
101    LS[2] = digitalRead(LSense2pin);
102    LS[3] = digitalRead(LSense3pin); //Left-most
103    LS[4] = digitalRead(LSenseRpin); //Right-most
104
105    //PushButton Readings (Ultimately ended up being unused; we simply changed the values and reuploaded to reflect our assigned court)
106  //  LeftCourt_in  = digitalRead(LeftCourt);
107  //  RightCourt_in = digitalRead(RightCourt);
108       int LeftCourt_in = 0;
109       int RightCourt_in = 1;
110
111
112
113
114  //NEW AND IMPROVED Error sensor code (now 90% sure to work)
115  if (LS[0] == 0 && LS[1] == 0 && LS[2] == 1){
116    error = 2;
117  }
118  else if (LS[0] == 0 && LS[1] == 1 && LS[2] == 1){
119    error = 1;
120  }
121  //else if (LS[0] == 0 && LS[1] == 1 && LS[2] == 0 && LS[3] == 1){
122  //  error = 0;
123  //}
124  else if (LS[0] == 0 && LS[1] == 1 && LS[2] == 0){
125    error = 0;
126  }
127  else if (LS[0] == 1 && LS[1] == 1 && LS[2] == 0){
128    error = -1;
129  }
130  else if (LS[0] == 1 && LS[1] == 0 && LS[2] == 0){
131    error = -2;
132  }
133
134  //End of Error sensor code
135    //Proportional Control
```

```
136   mpl = 170; //Left Motor Power. This speed is further modified depending on the
        error, seen below.
137   mpr = 170; //Right Motor Power
138   kpfor = 30; //kp value for when the robot moved forward
139   left_speed = mpl + kpfor*error; //Tentative line follower fix: if the left
        sensor goes off (error = -1), make the right motors faster to course-correct
140   right_speed = mpr - kpfor*error;
141
142   //Further modifications to the motor speed depending on how off-course the
        robot currently is
143   if(error == -1) {
144     left_back_speed = mpl*0.75;
145     right_back_speed = 0;
146   }
147   else if(error == 1)  {
148     left_back_speed = 0;
149     right_back_speed = mpr*0.75;
150   }
151   else if(error > 1)  {
152     left_back_speed = 0;
153     right_back_speed = mpr * 1.25;
154   }
155   else if(error < 1)  {
156     left_back_speed = mpl * 1.25;
157     right_back_speed = 0;
158   }
159   else  {
160     left_back_speed = mpl;
161     right_back_speed = mpr;
162   }
163
164 //Debug information regarding the backwards movement of the robot
165 //  Serial.print("LS: ");
166 //  Serial.print(left_back_speed);
167 //  Serial.print("RS :");
168
169 //Ensure all motors are moving forward
170   digitalWrite(FR1pin, HIGH); //Right In1
171   digitalWrite(FR2pin, LOW); //Right In2
172
173   digitalWrite(FL1pin, HIGH); //Right In3
174   digitalWrite(FL2pin, LOW); //Right In4
175
176   digitalWrite(BR2pin, LOW); //Left In2
177   digitalWrite(BR1pin, HIGH); //Left In1
178
179   digitalWrite(BL1pin, HIGH); //Left In3
180   digitalWrite(BL2pin, LOW); //Left In4
181
182
183
184
```

```
185
186   //State Machine
187
188
189   switch(state){
190
191     //All Motors are off in State 0. Robot is in Home position
192     case HOME: //101
193     analogWrite(BRpin,0);
194     analogWrite(FRpin,0);
195     analogWrite(BLpin,0);
196     analogWrite(FLpin,0);
197
198     if(RightCourt_in == 1){
199       state = FORWSIDE2; //Move sideways to the left
200       }
201     else if(LeftCourt_in == 1) {
202       state = FORWSIDE; //Move sideways to the right
203     }
204     else {
205       state = HOME; //We messed something up if this happens
206       }
207      break;
208
209     //The robot stops and begins launching balls
210     case STOP: //109
211     //Disable the four motors
212     analogWrite(BRpin,0);
213     analogWrite(FRpin,0);
214     analogWrite(BLpin,0);
215     analogWrite(FLpin,0);
216
217     //Initialize the ball-loading servo
218     LoadServo.attach(9);
219
220     //Initialize the launcher motor and set the motor speed, while keeping the
        motor disabled
221     digitalWrite(3,LOW);
222     shooterPower = 90;
223     analogWrite(2,shooterPower);
224
225     //Fully extend the loader servo (maximum angle is 180°) to prevent balls from
         entering the launcher prematurely
226     LoadServo.write(180);
227
228     //Loop to give the STOP state an end-case.
229     while (balls < 5) {
230
231       analogWrite(2, shooterPower);
232
233       //Allow the launcher to reach full speed by delaying for two seconds
234       delay(2000);
```

```
235
236       //After the two-second spinning up, rotate the loader servo such that a
            ball rolls into the launcher and hold that position for 0.75 seconds to
            prevent the ball from getting pinched
237       LoadServo.write(90);
238       delay(750);
239
240       //After 0.75 seconds, close the launching servo to prevent more than one
            ball from entering the launcher and hold for an extra second to allow the
             ball to reach the shooter and get launched
241       LoadServo.write(180);
242       delay(1000);
243
244       //Disable the motor to reset it. This is done since the motor would
            steadily lose power each time a ball was introduced unless its power was
            cycled for each shot.
245       analogWrite(2, 0);
246       delay(2000);
247
248       //Increment the loop variable
249       balls = balls + 1;
250     }
251
252     //Determine when no more balls are present on the robot
253     if(balls >= 5)  {
254       //Disable the launcher motor and send to the next state
255       analogWrite(2,0);
256       state = BACKLINE;
257       frontRight.write(0);
258       frontLeft.write(0);
259     }
260
261
262
263
264     //analogWrite(9,255);
265
266     break;
267
268     case FORWSIDE: //102; Move robot to the right until it reaches the centerline
269
270     //Set the correct motor direction for each motor. This allows the mechanum
          wheels to slide the robot sideways without requiring a turn.
271     digitalWrite(FR1pin, LOW); //Right In1
272     digitalWrite(FR2pin, HIGH); //Right In2
273
274     digitalWrite(FL1pin, HIGH); //Right In3
275     digitalWrite(FL2pin, LOW); //Right In4
276
277     digitalWrite(BR2pin, LOW); //Left In2
278     digitalWrite(BR1pin, HIGH); //Left In1
279
```

```
280       digitalWrite(BL1pin, LOW); //Left In3
281       digitalWrite(BL2pin, HIGH); //Left In4
282
283       //Write the speed of the motors. Due to a combination of weight imbalance and
             the right-side motors being different from the left-side motors, different
             speeds needed to be written to each side
284       analogWrite(BRpin,200);
285       analogWrite(FRpin,200);
286       analogWrite(BLpin,255);//229.5
287       analogWrite(FLpin,255);//242.25
288
289       //If the center line sensor detects the centerline, switch to the FORWLINE
            state. Otherwise, maintain current state
290       if(LS[1] == 1){
291         state = FORWLINE;
292       }
293       else  {
294         state = FORWSIDE;
295       }
296
297
298       break;
299
300       case FORWSIDE2: //110; Similar concept as the FORWSIDE state, though the
            robot is instructed to move forward for a short amount of time to avoid a
            patch of uneven surface present in the right court.
301         if(diag2trigger == 0) {
302
303          //Ensure all motors are moving forward
304           digitalWrite(FR1pin, HIGH); //Right In1
305           digitalWrite(FR2pin, LOW); //Right In2
306
307           digitalWrite(FL1pin, HIGH); //Right In3
308           digitalWrite(FL2pin, LOW); //Right In4
309
310           digitalWrite(BR2pin, LOW); //Left In2
311           digitalWrite(BR1pin, HIGH); //Left In1
312
313           digitalWrite(BL1pin, HIGH); //Left In3
314           digitalWrite(BL2pin, LOW); //Left In4
315
316           //Move the robot forward out of the home position for a short time. This
                was done since the surface directly to the side of the home position
                caused major slipping when the robot tried to move sideways.
317           analogWrite(BRpin,125);
318           analogWrite(FRpin,125);
319           analogWrite(BLpin,125);
320           analogWrite(FLpin,125);
321
322           delay(4000);
323           diag2trigger = 1;
324         }
```

```
325         //Set the correct motor direction for each motor. This allows the mechanum
              wheels to slide the robot sideways without requiring a turn.
326        digitalWrite(FR1pin, HIGH); //Right In1
327        digitalWrite(FR2pin, LOW); //Right In2
328
329        digitalWrite(FL1pin, LOW); //Right In3
330        digitalWrite(FL2pin, HIGH); //Right In4
331
332        digitalWrite(BR2pin, HIGH); //Left In2
333        digitalWrite(BR1pin, LOW); //Left In1
334
335        digitalWrite(BL1pin, HIGH); //Left In3
336        digitalWrite(BL2pin, LOW); //Left In4
337
338
339
340       //WRITE TO MOTORS HERE
341       analogWrite(BRpin,175);
342       analogWrite(FRpin,175);
343       analogWrite(BLpin,230);
344       analogWrite(FLpin,230);
345
346       if(LS[1] == 1){
347         state = FORWLINE;
348       }
349       else  {
350         state = FORWSIDE2;
351       }
352
353
354       break;
355
356       case FORWLINE: //103; this is the forward line-following state. The robot is
            to follow the centerline until reaching the half-court line, at which point
             it backs up slightly and beings shooting.
357
358       digitalWrite(FR1pin, HIGH); //Right In1
359       digitalWrite(FR2pin, LOW); //Right In2
360
361       digitalWrite(FL1pin, HIGH); //Right In3
362       digitalWrite(FL2pin, LOW); //Right In4
363
364       digitalWrite(BR2pin, LOW); //Left In2
365       digitalWrite(BR1pin, HIGH); //Left In1
366
367       digitalWrite(BL1pin, HIGH); //Left In3
368       digitalWrite(BL2pin, LOW); //Left In4
369
370
371       //WRITE TO MOTORS HERE
372       //Instead of writing constant values to the motor speed, we use variables
            that are modified by the line-following code above.
```

```
373       analogWrite(BRpin,right_speed);
374       analogWrite(FRpin,right_speed);
375       analogWrite(BLpin,left_speed);
376       analogWrite(FLpin,left_speed);
377
378       //If all five line sensors are activated (Only possible if the robot reached
            the half-court line), move to the BACKUP state.
379       if (LS[0] == 1 && LS[1] == 1 && LS[2] == 1 && LS[3] == 1 && LS[4] == 1)  {
380         state = BACKUP;
381       }
382       else  {
383         state = FORWLINE;
384       }
385
386       break;
387
388       case BACKUP: //108; Robot tends to overshoot the half-court line slightly.
            This state inches it back over the line.
389
390       //Reverse L298 motor directions to go backwards
391       digitalWrite(FR1pin, LOW); //Right In1
392       digitalWrite(FR2pin, HIGH); //Right In2
393
394       digitalWrite(FL1pin, LOW); //Right In3
395       digitalWrite(FL2pin, HIGH); //Right In4
396
397       digitalWrite(BR2pin, HIGH); //Left In2
398       digitalWrite(BR1pin, LOW); //Left In1
399
400       digitalWrite(BL1pin, LOW); //Left In3
401       digitalWrite(BL2pin, HIGH); //Left In4
402
403       //WRITE TO MOTORS HERE
404       analogWrite(BRpin,125);
405       analogWrite(FRpin,125);
406       analogWrite(BLpin,125);
407       analogWrite(FLpin,125);
408
409       //After half a second of travel, begin launching
410       delay(500);
411       state = STOP;
412
413       break;
414
415
416
417
418
419
420
421       case BACKLINE: //106
422       //INPUT TO MOTORS TO MAKE IT BACK TO REAR OF COURT
```

```
423     //GO TO BACKDIAG ONCE THE LIMITSWITCH IS HIT
424
425     digitalWrite(FR1pin, LOW); //Right In1
426     digitalWrite(FR2pin, HIGH); //Right In2
427
428     digitalWrite(FL1pin, LOW); //Right In3
429     digitalWrite(FL2pin, HIGH); //Right In4
430
431     digitalWrite(BR2pin, HIGH); //Left In2
432     digitalWrite(BR1pin, LOW); //Left In1
433
434     digitalWrite(BL1pin, LOW); //Left In3
435     digitalWrite(BL2pin, HIGH); //Left In4
436
437       //This code tells the robot to turn by a preset amount before going
            straight backwards, depending on which court it's in.
438       //Encoder counts were used in lieu of line-following due to the placement
            of the line sensors at the front making line-following difficult while
            moving backwards.
439       if(backDiagTrigger == 0 && RightCourt_in == 1) {
440           analogWrite(BLpin,255*.84);
441           analogWrite(FLpin,255*.84);
442 //          Serial.println(frontLeft.read());
443           if(frontLeft.read() < -1200) {
444             backDiagTrigger = 1;
445             frontRight.write(0);
446             frontLeft.write(0);
447           }
448       } else if(backDiagTrigger == 0 && LeftCourt_in == 1) {
449           analogWrite(BRpin,255);
450           analogWrite(FRpin,255);
451 //          Serial.println(frontRight.read());
452           if(frontRight.read() < -3000) {
453             backDiagTrigger = 1;
454             frontRight.write(0);
455             frontLeft.write(0);
456           }
457       } else if(frontLeft.read() < -9000 && RightCourt_in == 1) {
458           analogWrite(BRpin,255);
459           analogWrite(FRpin,255);
460           analogWrite(BLpin,0);
461           analogWrite(FLpin,0);
462       } else if(frontRight.read() < -16000 && LeftCourt_in == 1) {
463           analogWrite(BRpin,0);
464           analogWrite(FRpin,0);
465           analogWrite(BLpin,255*.84);
466           analogWrite(FLpin,255*.84);
467       } else if(frontRight.read()/75 < frontLeft.read()/45) {
468           analogWrite(BRpin,100);
469           analogWrite(FRpin,100);
470           analogWrite(BLpin,125*.84);
471           analogWrite(FLpin,125*.84);
```

```
472       } else if(frontRight.read()/75 > frontLeft.read()/45) {
473           analogWrite(BRpin,125);
474           analogWrite(FRpin,125);
475           analogWrite(BLpin,100*.84);
476           analogWrite(FLpin,100*.84);
477       } else if(frontRight.read()/75 == frontLeft.read()/45){
478           analogWrite(BRpin,125);
479           analogWrite(FRpin,125);
480           analogWrite(BLpin,125*.84);
481           analogWrite(FLpin,125*.84);
482       }
483     //Debug information regarding the motor encoder values
484     Serial.print("FR: ");
485     Serial.println(frontRight.read());
486     Serial.print("FL: ");
487     Serial.println(frontLeft.read());
488
489     //Stop motors upon activation of the limit switch on the robot's rear
490     LimitHit = digitalRead(LimitSwitchPin);
491
492     if(LimitHit == 0) {
493       //state = BACKDIAG2;
494       analogWrite(BRpin,0);
495       analogWrite(FRpin,0);
496       analogWrite(BLpin,0);
497       analogWrite(FLpin,0);
498     }
499     else{
500       state = BACKLINE;
501     }
502
503     break;
504
505     }
506     //Debugging information regarding the current state of the robot
507     Serial.println(state);
508     }
509
510
```