```
1
2  /* Dead Reckoning
3    20200117
4    Program to get the ardbot to drive a predefined path
5    jsteele, mshapiro, klarsen
6  */
7
8  // Complete Tasks in Initial_Robot_Testing.ino first!!
9
10 /* Program TODO LIST
11   1) Change the milliSecondsPerCM constant to the value you found in testing
12   2) Change the milliSecondsPer90Deg constant to the value you found in
   testing
13   3) Change the PWM of the forward function to the values you found in
   testing
14   4) Write code for the button pause functionality
15   5) Write your own turn function
16
17   Note: type // to make a single line comment
18         Comments are for the future you to understand how you wrote your code
19 */
20
21 // Preprocessor Definitions
22
23 // If you have a kit with the moto shield, set this to true
24 // If you have the Dual H-Bridge controller w/o the shield, set to false
25 #define SHIELD false
26
27 // Defining these allows us to use letters in place of binary when
28 // controlling our motor(s)
29 #define A 0
30 #define B 1
31
32 //SHIELD Pin varables
33 #define motorApwm 3
34 #define motorAdir 12
35 #define motorBpwm 11
36 #define motorBdir 13
37
38 //Driver Pin variable
39 #define IN1 9
40 #define IN2 10
41 #define IN3 5
42 #define IN4 6
43
44 #define FORWARD 0
45 #define LEFT 1
46 #define RIGHT -1
47 #define pushButton 2
48 #define A 1
49 #define B 2
50 #define pwmA 3
51 #define dirA 12
52 #define pwmB 11
53 #define dirB 13
54
55 // PWM values for the motors
56 #define motorA_PWM 185
57 #define motorB_PWM 200
58
```

```
59 #define SHIELD 0
60
61 // the following converts centimeters into milliseconds as long datatype
62 #define milliSecondsPerCM 54     //CHANGE THIS ACCORDING TO YOUR BOT
63 #define milliSecondsPer90Deg 900 //CHANGE THIS ACCORDNING TO YOUR BOT
64
65 // the itemized list of moves for the robot as a 1D array
66 // this setup assumes that all the turns are 90 degrees and that all motions
   are pairs of drives and turns.
67 int moves[] = {140, LEFT, 90, RIGHT, 60, RIGHT, 180, RIGHT, 100, LEFT, 60,
   RIGHT, 100, RIGHT, 150, RIGHT};
68
69 // RIGHT param t is delay time calculated from dist and speed ratio
70 void turnRight(int t)
71 {
72   run_motor(A, -motorA_PWM); //set this to a number between -255 and 255
73   run_motor(B, motorB_PWM);  //set this to a number between -255 and 255
74   delay(t);                  //set this to a time in ms for the motors to run
75   run_motor(A, 0);           //motors stop
76   run_motor(B, 0);
77 }
78
79 // LEFT param t is delay time calculated from dist and speed ratio
80 void turnLeft(int t)
81 {
82   run_motor(A, motorA_PWM);  //set this to a number between -255 and 255
83   run_motor(B, -motorA_PWM); //set this to a number between -255 and 255
84   delay(t);                  //set this to a time in ms for the motors to run
85   run_motor(A, 0);           //motors stop
86   run_motor(B, 0);
87 }
88
89 void setup()
90 {
91   // set up the motor drive ports
92   pinMode(pwmA, OUTPUT);
93   pinMode(dirA, OUTPUT);
94   pinMode(pwmB, OUTPUT);
95   pinMode(dirB, OUTPUT);
96   // make the pushbutton's pin an input:
97   pinMode(pushButton, INPUT_PULLUP); //CHANGE TO INPUT_PULLUP
98   // initialize serial communication at 9600 bits per second:
99   Serial.begin(9600);
100 }
101
102 void loop()
103 {
104   int i, dist, dir;
105   long time;
106   while (digitalRead(pushButton) == 1)
107     ;
108   while (digitalRead(pushButton) == 0)
109     ;
110   //This for loop steps (or iterates) through the array 'moves'
111   for (i = 0; i < sizeof(moves) / 2; i = i + 2)
112   {
113
114     while (digitalRead(pushButton))
115     {
116       // Do nothing but wait
```

```
      //Serial.println("Waiting");
    }

    delay(250);
    //Forward Leg of each step
    Serial.print("Step #:");
    Serial.println(i);
    dist = moves[i];
    Serial.print("Forward for");
    time = Forward(dist);
    Serial.print(time);
    Serial.println(" ms");
    delay(1000);

    //Turn Leg of each step
    Serial.print("Step #:");
    Serial.println(i + 1);
    dir = moves[i + 1];
    if (dir == LEFT)
    {
      time = Turn(90);
      Serial.print("turning LEFT ");
      Serial.print(time);
      Serial.println(" ms");
    }
    else
    {
      time = Turn(-90);
      Serial.println("turning RIGHT ");

      Serial.print(time);
      Serial.println(" ms");
    } // end of else motions conditional
    delay(1000);

  } // end of for loop
  Serial.println("That's All Folks!");
  delay(1000);
  exit(i);
} // the end

////////////////////////////////////////////
unsigned long Forward(int distance)
{
  unsigned long t;
  t = distance * milliSecondsPerCM; //Time to keep motors on

  //To drive forward, motors go in the same direction
  run_motor(A, motorA_PWM); //change PWM to your calibrations
  run_motor(B, motorB_PWM); //change PWM to your calibrations
  delay(t);
  run_motor(A, 0);
  run_motor(B, 0);
  return (t);
}

////////////////////////////////////////////
unsigned long Turn(int degrees)
{
  unsigned long t;
```

```
  int sign = degrees / abs(degrees);              //Find if left or right
  t = (abs(degrees) / 90) * milliSecondsPer90Deg; //Time to keep motors on

  if (sign == -1)
  {
    turnLeft(t);
  }
  else
  {
    turnRight(t);
  }

  return (t);
}

```