

Mines Robotics Labs

Generated by Doxygen 1.8.17

1 Lab 3 Code Documentation	1
2 File Index	3
2.1 File List	3
3 File Documentation	5
3.1 custom_lab_3/custom_lab_3.ino File Reference	5
3.1.1 Detailed Description	7
3.1.2 Macro Definition Documentation	7
3.1.2.1 A	7
3.1.2.2 B	8
3.1.2.3 DegreesPerRev	8
3.1.2.4 dirA	8
3.1.2.5 dirB	8
3.1.2.6 DistancePerRev	8
3.1.2.7 EncoderCountsPerRev	8
3.1.2.8 EncoderMotorLeft	9
3.1.2.9 EncoderMotorRight	9
3.1.2.10 FORWARD	9
3.1.2.11 IN1	9
3.1.2.12 IN2	9
3.1.2.13 IN3	10
3.1.2.14 IN4	10
3.1.2.15 LEFT	10
3.1.2.16 pushButton	10
3.1.2.17 pwmA	10
3.1.2.18 pwmB	10
3.1.2.19 RIGHT	11
3.1.3 Function Documentation	11
3.1.3.1 adjustPWM()	11
3.1.3.2 calculateDesiredCount()	11
3.1.3.3 calculateDesiredCountTurn()	12
3.1.3.4 configure()	12
3.1.3.5 drive()	12
3.1.3.6 driveBackward()	13
3.1.3.7 driveForward()	13
3.1.3.8 idle()	15
3.1.3.9 indexLeftEncoderCount()	15
3.1.3.10 indexRightEncoderCount()	16
3.1.3.11 loop()	16
3.1.3.12 react_forward()	16
3.1.3.13 react_left()	16
3.1.3.14 react_right()	17

3.1.3.15 resetPWM()	17
3.1.3.16 setup()	17
3.1.3.17 turnLeft()	17
3.1.3.18 turnRight()	18
3.1.4 Variable Documentation	19
3.1.4.1 desiredCount	19
3.1.4.2 leftEncoderCount	19
3.1.4.3 leftOutput	19
3.1.4.4 milliSecondsPer90Deg	19
3.1.4.5 motorLeft_PWM	19
3.1.4.6 motorRight_PWM	20
3.1.4.7 moveList	20
3.1.4.8 rightEncoderCount	20
3.1.4.9 rightOutput	20
3.2 custom_lab_3/motors.ino File Reference	20
3.2.1 Function Documentation	20
3.2.1.1 motor_setup()	21
3.2.1.2 run_motor()	21
Index	23

Chapter 1

Lab 3 Code Documentation

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

custom_lab_3/ custom_lab_3.ino	
A basic feedback controlled system for an Arduino based robot	5
custom_lab_3/ motors.ino	20

Chapter 3

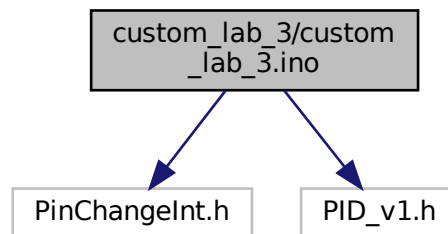
File Documentation

3.1 custom_lab_3/custom_lab_3.ino File Reference

A basic feedback controlled system for an Arduino based robot.

```
#include <PinChangeInt.h>
#include <PID_v1.h>
```

Include dependency graph for custom_lab_3.ino:



Macros

- #define [IN1](#) 9
Libraries for interrupts and PID.
- #define [IN2](#) 10
- #define [IN3](#) 5
- #define [IN4](#) 6
- #define [A](#) 0
Motor control.
- #define [B](#) 1
- #define [pwmA](#) 3
- #define [dirA](#) 12
- #define [pwmB](#) 11

- `#define dirB 13`
- `#define pushButton 2`
Start stop button.
- `#define EncoderCountsPerRev 12.0`
Drive constants - dependent on robot configuration.
- `#define DistancePerRev 51.0`
- `#define DegreesPerRev 27.0`
- `#define EncoderMotorLeft 7`
- `#define EncoderMotorRight 8`
- `#define FORWARD 0`
Enum defines.
- `#define LEFT 1`
- `#define RIGHT -1`

Functions

- `void resetPWM ()`
Helper function for setting the PWM back to default value.
- `void indexLeftEncoderCount ()`
ISR for left encoder.
- `void indexRightEncoderCount ()`
ISR for incrementing right encoder.
- `void calculateDesiredCount (int distance)`
Calculate how many encoder counts we expect given the distance provided based on the bot intrinsics.
- `void calculateDesiredCountTurn (int degrees)`
Calculate how many encoder counts we expect given the degrees provided.
- `void turnRight (int degrees)`
Turn bot to given degrees.
- `void turnLeft (int degrees)`
Turn bot right to given degrees.
- `void driveForward (int distance)`
Function to drive bot forward until encoders are within range.
- `void driveBackward (int distance)`
Drive the bot backwards.
- `void configure ()`
Function for configuration of pin states and interrupts.
- `void idle ()`
Default behavior when not driving, waits for the pushButton to be pressed so it can execute next command Blocking function.
- `void adjustPWM ()`
Run the PID loop calculation and set out put to motors output in PWM.
- `void setup ()`
Entry point of program handles serial setup and PID config.
- `void react_left ()`
This is the logic to execute if we hit a push button ideally this is never executed as we should never actually hit the walls.
- `void react_right ()`
- `void react_forward ()`
- `void drive ()`
Main drive execution of program, iterates through moves list executing next move with corresponding distance or degrees.
- `void loop ()`
Loop execution of the program.

Variables

- double `leftEncoderCount` = 0
Lab specific variables.
- double `rightEncoderCount` = 0
- int `motorLeft_PWM` = 180
Default motor pwm values.
- int `motorRight_PWM` = 200
- int `milliSecondsPer90Deg` = 900
Time it takes to move 90 degrees.
- double `desiredCount`
How many encoder counts for given distance.
- int `moveList` [] = {FORWARD, 300, LEFT, 90, FORWARD, 300, LEFT, 90, FORWARD, 300, RIGHT, 90, FORWARD, 900, RIGHT, 90, FORWARD, 600, RIGHT, 90, FORWARD, 300}
- double `leftOutput`
PID values setpoints = desired counts, output = PWM, input = current counts.
- double `rightOutput`

3.1.1 Detailed Description

A basic feedback controlled system for an Arduino based robot.

Author

Christian Prather

Version

0.1

Date

2020-10-21

3.1.2 Macro Definition Documentation

3.1.2.1 A

```
#define A 0
```

Motor control.

Definition at line 28 of file custom_lab_3.ino.

3.1.2.2 B

```
#define B 1
```

Definition at line 29 of file custom_lab_3.ino.

3.1.2.3 DegreesPerRev

```
#define DegreesPerRev 27.0
```

Definition at line 41 of file custom_lab_3.ino.

3.1.2.4 dirA

```
#define dirA 12
```

Definition at line 31 of file custom_lab_3.ino.

3.1.2.5 dirB

```
#define dirB 13
```

Definition at line 33 of file custom_lab_3.ino.

3.1.2.6 DistancePerRev

```
#define DistancePerRev 51.0
```

Definition at line 40 of file custom_lab_3.ino.

3.1.2.7 EncoderCountsPerRev

```
#define EncoderCountsPerRev 12.0
```

Drive constants - dependent on robot configuration.

Definition at line 39 of file custom_lab_3.ino.

3.1.2.8 EncoderMotorLeft

```
#define EncoderMotorLeft 7
```

Definition at line 43 of file custom_lab_3.ino.

3.1.2.9 EncoderMotorRight

```
#define EncoderMotorRight 8
```

Definition at line 44 of file custom_lab_3.ino.

3.1.2.10 FORWARD

```
#define FORWARD 0
```

Enum defines.

Definition at line 51 of file custom_lab_3.ino.

3.1.2.11 IN1

```
#define IN1 9
```

Libraries for interrupts and PID.

Global Defines Motor driver connections

Definition at line 22 of file custom_lab_3.ino.

3.1.2.12 IN2

```
#define IN2 10
```

Definition at line 23 of file custom_lab_3.ino.

3.1.2.13 IN3

```
#define IN3 5
```

Definition at line 24 of file custom_lab_3.ino.

3.1.2.14 IN4

```
#define IN4 6
```

Definition at line 25 of file custom_lab_3.ino.

3.1.2.15 LEFT

```
#define LEFT 1
```

Definition at line 52 of file custom_lab_3.ino.

3.1.2.16 pushButton

```
#define pushButton 2
```

Start stop button.

Definition at line 36 of file custom_lab_3.ino.

3.1.2.17 pwmA

```
#define pwmA 3
```

Definition at line 30 of file custom_lab_3.ino.

3.1.2.18 pwmB

```
#define pwmB 11
```

Definition at line 32 of file custom_lab_3.ino.

3.1.2.19 RIGHT

```
#define RIGHT -1
```

Definition at line 53 of file custom_lab_3.ino.

3.1.3 Function Documentation

3.1.3.1 adjustPWM()

```
void adjustPWM ( )
```

Run the PID loop calculation and set out put to motors output in PWM.

Definition at line 324 of file custom_lab_3.ino.

```
325 {
326     // Compute the pid values
327     leftPID.Compute();
328     rightPID.Compute();
329
330     // Set the pid values within range
331     motorLeft_PWM = constrain(leftOutput, 150, 250);
332     motorRight_PWM = constrain(rightOutput, 150, 235);
333     Serial.print("Left PWM: ");
334     Serial.print(motorLeft_PWM);
335     Serial.print(" ");
336     Serial.println(leftEncoderCount);
337     Serial.print("Right PWM: ");
338     Serial.print(motorRight_PWM);
339     Serial.print(" ");
340     Serial.println(rightEncoderCount);
341 }
```

3.1.3.2 calculateDesiredCount()

```
void calculateDesiredCount (
    int distance )
```

Calculate how many encoder counts we expect given the distance provided based on the bot intrinsics.

Parameters

<i>distance</i>	
-----------------	--

Definition at line 109 of file custom_lab_3.ino.

```
110 {
111     double revolutionsRequired = distance / DistancePerRev;
112
113     desiredCount = revolutionsRequired * EncoderCountsPerRev;
114     // Reset encoder counts
115     leftEncoderCount = 0;
116     rightEncoderCount = 0;
117     Serial.print("Desired Count: ");
118     Serial.println(desiredCount);
119 }
```

3.1.3.3 calculateDesiredCountTurn()

```
void calculateDesiredCountTurn (
    int degrees )
```

Calculate how many encoder counts we expect given the degrees provided.

Parameters

<i>degrees</i>	
----------------	--

Definition at line 126 of file custom_lab_3.ino.

```
127 {
128     double revolutionsRequired = degrees / DegreesPerRev;
129     desiredCount = revolutionsRequired * EncoderCountsPerRev;
130     leftEncoderCount = 0;
131     rightEncoderCount = 0;
132     Serial.print("Desired Count: ");
133     Serial.println(desiredCount);
134 }
```

3.1.3.4 configure()

```
void configure ( )
```

Function for configuration of pin states and interrupts.

Definition at line 288 of file custom_lab_3.ino.

```
289 {
290     // set up the motor drive ports
291     pinMode(pwmA, OUTPUT);
292     pinMode(dirA, OUTPUT);
293     pinMode(pwmB, OUTPUT);
294     pinMode(dirB, OUTPUT);
295
296     pinMode(pushButton, INPUT_PULLUP);
297
298     pinMode(EncoderMotorLeft, INPUT_PULLUP); //set the pin to input
299     PCintPort::attachInterrupt(EncoderMotorLeft, indexLeftEncoderCount, CHANGE);
300
301     pinMode(EncoderMotorRight, INPUT_PULLUP); //set the pin to input
302     PCintPort::attachInterrupt(EncoderMotorRight, indexRightEncoderCount, CHANGE);
303 }
```

3.1.3.5 drive()

```
void drive ( )
```

Main drive execution of program, iterates through moves list executing next move with corresponding distance or degrees.

Definition at line 383 of file custom_lab_3.ino.

```
384 {
385     // Iterate over the list jumping by two each time
```



```

386     for (int i = 0; i < sizeof(moveList); i += 2)
387     {
388         idle();
389         switch (moveList[i])
390         {
391             case LEFT:
392                 turnLeft(moveList[i + 1]);
393                 break;
394             case RIGHT:
395                 turnRight(moveList[i + 1]);
396                 break;
397             case FORWARD:
398                 driveForward(moveList[i + 1]);
399                 break;
400             default:
401                 break;
402         }
403     }
404 }

```

3.1.3.6 driveBackward()

```

void driveBackward (
    int distance )

```

Drive the bot backwards.

Parameters

<i>distance</i>	
-----------------	--

Definition at line 253 of file custom_lab_3.ino.

```

254 {
255     resetPWM();
256     calculateDesiredCount (distance);
257
258     // Loop until the encoders read correct
259
260     while ((desiredCount - leftEncoderCount) > 3 || (desiredCount - rightEncoderCount) > 3)
261     {
262         adjustPWM();
263         //To drive backward, motors go in the same direction
264
265         if ((desiredCount - leftEncoderCount) > 3)
266         {
267             run_motor(A, motorLeft_PWM); //change PWM to your calibrations
268         }
269         if ((desiredCount - rightEncoderCount) > 3)
270         {
271             run_motor(B, motorRight_PWM); //change PWM to your calibrations
272         }
273     }
274
275     // motors stop
276     run_motor(A, 0);
277     run_motor(B, 0);
278     Serial.println("Done driving backwards");
279     Serial.print("L: ");
280     Serial.println(leftEncoderCount);
281     Serial.print("R: ");
282     Serial.println(rightEncoderCount);
283 }

```

3.1.3.7 driveForward()

```

void driveForward (
    int distance )

```

Function to drive bot forward until encoders are within range.

Parameters

<i>distance</i>	
-----------------	--

Definition at line 215 of file custom_lab_3.ino.

```

216 {
217     Serial.println("Driving Forward...");
218     resetPWM();
219     calculateDesiredCount(distance);
220
221     // Loop until the encoders read correct
222
223     while ((desiredCount - leftEncoderCount) > 3 || (desiredCount - rightEncoderCount) > 3)
224     {
225         adjustPWM();
226         //To drive forward, motors go in the same direction
227
228         if ((desiredCount - leftEncoderCount) > 3)
229         {
230             run_motor(A, -motorLeft_PWM); //change PWM to your calibrations
231         }
232         if ((desiredCount - rightEncoderCount) > 3)
233         {
234             run_motor(B, -motorRight_PWM); //change PWM to your calibrations
235         }
236     }
237
238     // motors stop
239     run_motor(A, 0);
240     run_motor(B, 0);
241     Serial.println("Done driving forward");
242     Serial.print("L: ");
243     Serial.println(leftEncoderCount);
244     Serial.print("R: ");
245     Serial.println(rightEncoderCount);
246 }
```

3.1.3.8 idle()

```
void idle ( )
```

Default behavior when not driving, waits for the pushButton to be pressed so it can execute next command Blocking function.

Definition at line 310 of file custom_lab_3.ino.

```

311 {
312     Serial.println("Idle..");
313     while (digitalRead(pushButton) == 1)
314         ; // wait for button push
315     while (digitalRead(pushButton) == 0)
316         ; // wait for button release
317     delay(2000); // Give time to move hand
318 }
```

3.1.3.9 indexLeftEncoderCount()

```
void indexLeftEncoderCount ( )
```

ISR for left encoder.

Definition at line 88 of file custom_lab_3.ino.

```

89 {
90     leftEncoderCount++;
91     //Serial.println("Left Encoder ++");
92 }
```

3.1.3.10 indexRightEncoderCount()

```
void indexRightEncoderCount ( )
```

ISR for incrementing right encoder.

Definition at line 97 of file custom_lab_3.ino.

```
98 {  
99     rightEncoderCount++;  
100     //Serial.println("Right Encoder ++");  
101 }
```

3.1.3.11 loop()

```
void loop ( )
```

Loop execution of the program.

Definition at line 409 of file custom_lab_3.ino.

```
410 {  
411     drive();  
412 }
```

3.1.3.12 react_forward()

```
void react_forward ( )
```

Definition at line 373 of file custom_lab_3.ino.

```
374 {  
375     // TODO: Check which button was hit  
376     driveBackward(50);  
377 }
```

3.1.3.13 react_left()

```
void react_left ( )
```

This is the logic to execute if we hit a push button ideally this is never executed as we should never actually hit the walls.

Definition at line 359 of file custom_lab_3.ino.

```
360 {  
361     // TODO: Check which button was hit  
362  
363     driveBackward(20);  
364     turnRight(30);  
365 }
```

3.1.3.14 react_right()

```
void react_right ( )
```

Definition at line 366 of file custom_lab_3.ino.

```
367 {  
368     // TODO: Check which button was hit  
369  
370     driveBackward(20);  
371     turnLeft(30);  
372 }
```

3.1.3.15 resetPWM()

```
void resetPWM ( )
```

Helper function for setting the PWM back to default value.

Definition at line 79 of file custom_lab_3.ino.

```
80 {  
81     motorLeft_PWM = 180;  
82     motorRight_PWM = 200;  
83 }
```

3.1.3.16 setup()

```
void setup ( )
```

Entry point of program handles serial setup and PID config.

Definition at line 346 of file custom_lab_3.ino.

```
347 {  
348     Serial.begin(9600);  
349     Serial.println("Setting up.....");  
350     configure();  
351     leftPID.SetMode(AUTOMATIC);  
352     rightPID.SetMode(AUTOMATIC);  
353 }
```

3.1.3.17 turnLeft()

```
void turnLeft (  
    int degrees )
```

Turn bot right to given degrees.

Parameters

<i>degrees</i>	
----------------	--

Definition at line 178 of file custom_lab_3.ino.

```

179 {
180     resetPWM();
181     calculateDesiredCountTurn(degrees);
182
183     // Loop until the encoders read correct
184
185     while ((desiredCount - leftEncoderCount) > 3)
186     {
187         adjustPWM();
188         //To drive forward, motors go in the same direction
189
190         if ((desiredCount - leftEncoderCount) > 3)
191         {
192             run_motor(A, motorLeft_PWM); //change PWM to your calibrations
193         }
194         if ((desiredCount - rightEncoderCount) > 3)
195         {
196             run_motor(B, -motorRight_PWM); //change PWM to your calibrations
197         }
198     }
199
200     // motors stop
201     run_motor(A, 0);
202     run_motor(B, 0);
203     Serial.println("Done driving Left");
204     Serial.print("L: ");
205     Serial.println(leftEncoderCount);
206     Serial.print("R: ");
207     Serial.println(rightEncoderCount);
208 }

```

3.1.3.18 turnRight()

```

void turnRight (
    int degrees )

```

Turn bot to given degrees.

Parameters

<i>degrees</i>

Definition at line 141 of file custom_lab_3.ino.

```

142 {
143     resetPWM(); // Reset pwm
144     calculateDesiredCountTurn(degrees);
145     // While the encoders are not correct adjust PWM with PID loop
146     // Loop until the encoders read correct
147
148     while ((desiredCount - rightEncoderCount) > 3)
149     {
150         adjustPWM();
151         //To drive forward, motors go in the same direction
152
153         if ((desiredCount - leftEncoderCount) > 3)
154         {
155             run_motor(A, -motorLeft_PWM); //change PWM to your calibrations
156         }
157         if ((desiredCount - rightEncoderCount) > 3)
158         {
159             run_motor(B, motorRight_PWM); //change PWM to your calibrations
160         }
161     }
162
163     // motors stop
164     run_motor(A, 0);
165     run_motor(B, 0);
166     Serial.println("Done driving Right");
167     Serial.print("L: ");
168     Serial.println(leftEncoderCount);
169     Serial.print("R: ");
170     Serial.println(rightEncoderCount);
171 }

```

3.1.4 Variable Documentation

3.1.4.1 desiredCount

```
double desiredCount
```

How many encoder counts for given distance.

Definition at line 63 of file custom_lab_3.ino.

3.1.4.2 leftEncoderCount

```
PID leftPID & leftEncoderCount = 0
```

Lab specific variables.

Definition at line 47 of file custom_lab_3.ino.

3.1.4.3 leftOutput

```
double leftOutput
```

PID values setpoints = desired counts, output = PWM, input = current counts.

Definition at line 72 of file custom_lab_3.ino.

3.1.4.4 milliSecondsPer90Deg

```
int milliSecondsPer90Deg = 900
```

Time it takes to move 90 degrees.

Definition at line 60 of file custom_lab_3.ino.

3.1.4.5 motorLeft_PWM

```
int motorLeft_PWM = 180
```

Default motor pwm values.

Definition at line 56 of file custom_lab_3.ino.

3.1.4.6 motorRight_PWM

```
int motorRight_PWM = 200
```

Definition at line 57 of file custom_lab_3.ino.

3.1.4.7 moveList

```
int moveList[] = {FORWARD, 300, LEFT, 90, FORWARD, 300, LEFT, 90, FORWARD, 300, RIGHT, 90, FORWARD, 900, RIGHT, 90, FORWARD, 600, RIGHT, 90, FORWARD, 300}
```

Definition at line 66 of file custom_lab_3.ino.

3.1.4.8 rightEncoderCount

```
PID rightPID & rightEncoderCount = 0
```

Definition at line 48 of file custom_lab_3.ino.

3.1.4.9 rightOutput

```
double rightOutput
```

Definition at line 72 of file custom_lab_3.ino.

3.2 custom_lab_3/motors.ino File Reference

Functions

- void [motor_setup](#) ()
- void [run_motor](#) (int motor, int pwm)

3.2.1 Function Documentation

3.2.1.1 motor_setup()

```
void motor_setup ( )
```

Definition at line 1 of file motors.ino.

```
2 {
3   // if using dual motor driver
4   // define driver pins as outputs
5   pinMode(IN1, OUTPUT);
6   pinMode(IN2, OUTPUT);
7   pinMode(IN3, OUTPUT);
8   pinMode(IN4, OUTPUT);
9   // initialize all pins to zero
10  digitalWrite(IN1, 0);
11  digitalWrite(IN2, 0);
12  digitalWrite(IN3, 0);
13  digitalWrite(IN4, 0);
14  return;
15 } // end function
```

3.2.1.2 run_motor()

```
void run_motor (
    int motor,
    int pwm )
```

Definition at line 19 of file motors.ino.

```
20 {
21   int dir = (pwm / abs(pwm)) > 0; // returns if direction is forward (1) or reverse (0)
22   pwm = abs(pwm); // only positive values can be sent to the motor
23
24   switch (motor)
25   { // find which motor to control
26     case A: // if A, write A pins
27       if (dir)
28       { // If dir is forward
29         analogWrite(IN1, pwm); // IN1 is the forward pwm pin
30         digitalWrite(IN2, LOW); // IN2 is low
31       }
32       else
33       {
34         digitalWrite(IN1, LOW); // IN1 is low
35         analogWrite(IN2, pwm); // IN2 is the reverse pwm pin
36       } // end if
37       break; // end case A
38     case B: // if B, write B pins
39       if (dir)
40       { // if dir is forward
41         analogWrite(IN3, pwm); // IN3 is the forward pwm pin
42         digitalWrite(IN4, LOW); // IN4 is low
43       }
44       else
45       {
46         digitalWrite(IN3, LOW); // IN3 is low
47         analogWrite(IN4, pwm); // IN4 is the reverse pwm pin
48       } // end if
49       break; // end case B
50     } // end switch case
51   return;
52 } // end function
```


Index

A

custom_lab_3.ino, [7](#)

adjustPWM

custom_lab_3.ino, [11](#)

B

custom_lab_3.ino, [7](#)

calculateDesiredCount

custom_lab_3.ino, [11](#)

calculateDesiredCountTurn

custom_lab_3.ino, [12](#)

configure

custom_lab_3.ino, [12](#)

custom_lab_3.ino

A, [7](#)

adjustPWM, [11](#)

B, [7](#)

calculateDesiredCount, [11](#)

calculateDesiredCountTurn, [12](#)

configure, [12](#)

DegreesPerRev, [8](#)

desiredCount, [19](#)

dirA, [8](#)

dirB, [8](#)

DistancePerRev, [8](#)

drive, [12](#)

driveBackward, [13](#)

driveForward, [13](#)

EncoderCountsPerRev, [8](#)

EncoderMotorLeft, [8](#)

EncoderMotorRight, [9](#)

FORWARD, [9](#)

idle, [15](#)

IN1, [9](#)

IN2, [9](#)

IN3, [9](#)

IN4, [10](#)

indexLeftEncoderCount, [15](#)

indexRightEncoderCount, [15](#)

LEFT, [10](#)

leftEncoderCount, [19](#)

leftOutput, [19](#)

loop, [16](#)

milliSecondsPer90Deg, [19](#)

motorLeft_PWM, [19](#)

motorRight_PWM, [19](#)

moveList, [20](#)

pushButton, [10](#)

pwmA, [10](#)

pwmB, [10](#)

react_forward, [16](#)

react_left, [16](#)

react_right, [16](#)

resetPWM, [17](#)

RIGHT, [10](#)

rightEncoderCount, [20](#)

rightOutput, [20](#)

setup, [17](#)

turnLeft, [17](#)

turnRight, [18](#)

custom_lab_3/custom_lab_3.ino, [5](#)

custom_lab_3/motors.ino, [20](#)

DegreesPerRev

custom_lab_3.ino, [8](#)

desiredCount

custom_lab_3.ino, [19](#)

dirA

custom_lab_3.ino, [8](#)

dirB

custom_lab_3.ino, [8](#)

DistancePerRev

custom_lab_3.ino, [8](#)

drive

custom_lab_3.ino, [12](#)

driveBackward

custom_lab_3.ino, [13](#)

driveForward

custom_lab_3.ino, [13](#)

EncoderCountsPerRev

custom_lab_3.ino, [8](#)

EncoderMotorLeft

custom_lab_3.ino, [8](#)

EncoderMotorRight

custom_lab_3.ino, [9](#)

FORWARD

custom_lab_3.ino, [9](#)

idle

custom_lab_3.ino, [15](#)

IN1

custom_lab_3.ino, [9](#)

IN2

custom_lab_3.ino, [9](#)

IN3

custom_lab_3.ino, [9](#)

IN4

- custom_lab_3.ino, [10](#)
- indexLeftEncoderCount
 - custom_lab_3.ino, [15](#)
- indexRightEncoderCount
 - custom_lab_3.ino, [15](#)
- LEFT
 - custom_lab_3.ino, [10](#)
- leftEncoderCount
 - custom_lab_3.ino, [19](#)
- leftOutput
 - custom_lab_3.ino, [19](#)
- loop
 - custom_lab_3.ino, [16](#)
- milliSecondsPer90Deg
 - custom_lab_3.ino, [19](#)
- motor_setup
 - motors.ino, [20](#)
- motorLeft_PWM
 - custom_lab_3.ino, [19](#)
- motorRight_PWM
 - custom_lab_3.ino, [19](#)
- motors.ino
 - motor_setup, [20](#)
 - run_motor, [21](#)
- moveList
 - custom_lab_3.ino, [20](#)
- pushButton
 - custom_lab_3.ino, [10](#)
- pwmA
 - custom_lab_3.ino, [10](#)
- pwmB
 - custom_lab_3.ino, [10](#)
- react_forward
 - custom_lab_3.ino, [16](#)
- react_left
 - custom_lab_3.ino, [16](#)
- react_right
 - custom_lab_3.ino, [16](#)
- resetPWM
 - custom_lab_3.ino, [17](#)
- RIGHT
 - custom_lab_3.ino, [10](#)
- rightEncoderCount
 - custom_lab_3.ino, [20](#)
- rightOutput
 - custom_lab_3.ino, [20](#)
- run_motor
 - motors.ino, [21](#)
- setup
 - custom_lab_3.ino, [17](#)
- turnLeft
 - custom_lab_3.ino, [17](#)
- turnRight
 - custom_lab_3.ino, [18](#)