Mines Robotics Labs

Generated by Doxygen 1.8.17

Chapter 1

Lab 3 Code Documentation

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

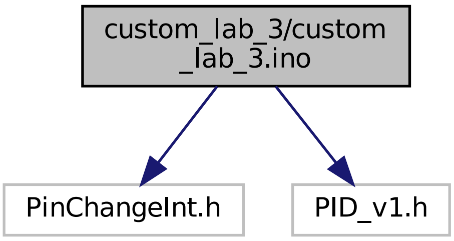
File Documentation

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 shoudl 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.

```
{
    // Compute the pid values
    leftPID.Compute();
    rightPID.Compute();

    // Set the pid values within range
    motorLeft_PWM = constrain(leftOutput, 150, 250);
    motorRight_PWM = constrain(rightOutput, 150, 235);
    Serial.print("Left PWM: ");
    Serial.print(motorLeft_PWM);
    Serial.print(" ");
    Serial.println(leftEncoderCount);
    Serial.print("Right PWM: ");
    Serial.print(motorRight_PWM);
    Serial.print(" ");
    Serial.println(rightEncoderCount);
}
```

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

distance	

Definition at line 109 of file custom_lab_3.ino.

```
{
    double revolutionsRequired = distance / DistancePerRev;

    desiredCount = revolutionsRequired * EncoderCountsPerRev;
    // Reset encoder counts
    leftEncoderCount = 0;
    rightEncoderCount = 0;
    Serial.print("Desired Count: ");
    Serial.println(desiredCount);
}
```

3.1.3.3 calculateDesiredCountTurn()

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

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

Parameters

degrees	

Definition at line 126 of file custom_lab_3.ino.

```
{
    double revolutionsRequired = degrees / DegreesPerRev;
    desiredCount = revolutionsRequired * EncoderCountsPerRev;
    leftEncoderCount = 0;
    rightEncoderCount = 0;
    Serial.print("Desired Count: ");
    Serial.println(desiredCount);
}
```

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.

```
{
    // set up the motor drive ports
    pinMode(pwmA, OUTPUT);
    pinMode(dirA, OUTPUT);
    pinMode(pwmB, OUTPUT);
    pinMode(dirB, OUTPUT);

    pinMode(pushButton, INPUT_PULLUP);

    pinMode(EncoderMotorLeft, INPUT_PULLUP); //set the pin to input
    PCintPort::attachInterrupt(EncoderMotorLeft, indexLeftEncoderCount, CHANGE);

    pinMode(EncoderMotorRight, INPUT_PULLUP); //set the pin to input
    PCintPort::attachInterrupt(EncoderMotorRight, indexRightEncoderCount, CHANGE);
}
```

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.

```
{
    // Iterate over the list jumping by two each time
    for (int i = 0; i < sizeof(moveList); i += 2)
    {
        idle();
        switch (moveList[i])
        {
        case LEFT:
            turnLeft(moveList[i + 1]);
            break;
        case RIGHT:
            turnRight(moveList[i + 1]);
            break;
        case FORWARD:
            driveForward(moveList[i + 1]);
            break;
        default:
            break;
        }
    }
}
```

3.1.3.6 driveBackward()

```
void driveBackward (
            int distance )
```

Drive the bot backwards.

Parameters

distance	

Definition at line 253 of file custom_lab_3.ino.

```
{
    resetPWM();
    calculateDesiredCount(distance);

    // Loop unitl the encoders read correct

    while ((desiredCount - leftEncoderCount) > 3 || (desiredCount - rightEncoderCount) > 3)
    {
        adjustPWM();
        //To drive backward, motors go in the same direction

        if ((desiredCount - leftEncoderCount) > 3)
        {
            run_motor(A, motorLeft_PWM); //change PWM to your calibrations
        }
        if ((desiredCount - rightEncoderCount) > 3)
        {
            run_motor(B, motorRight_PWM); //change PWM to your calibrations
        }
    }

    // motors stop
    run_motor(A, 0);
    run_motor(B, 0);
    Serial.println("Done driving backwards");
    Serial.print("L: ");
    Serial.println(leftEncoderCount);
    Serial.print("R: ");
    Serial.println(rightEncoderCount);
}
```

3.1.3.7 driveForward()

```
void driveForward (
            int distance )
```

Function to drive bot forward until encoders are within range.

Parameters

distance	

Definition at line 215 of file custom_lab_3.ino.

```
{
    Serial.println("Driving Forward...");
    resetPWM();
    calculateDesiredCount(distance);

    // Loop unitl the encoders read correct

    while ((desiredCount - leftEncoderCount) > 3 || (desiredCount - rightEncoderCount) > 3)
    {
        adjustPWM();
        //To drive forward, motors go in the same direction

        if ((desiredCount - leftEncoderCount) > 3)
        {
            run_motor(A, -motorLeft_PWM); //change PWM to your calibrations
        }
        if ((desiredCount - rightEncoderCount) > 3)
        {
            run_motor(B, -motorRight_PWM); //change PWM to your calibrations
        }
    }

    // motors stop
    run_motor(A, 0);
    run_motor(B, 0);
    Serial.println("Done driving forward");
    Serial.print("L: ");
    Serial.println(leftEncoderCount);
    Serial.print("R: ");
    Serial.println(rightEncoderCount);
}
```

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.

```
{
    Serial.println("Idle..");
    while (digitalRead(pushButton) == 1)
        ; // wait for button push
    while (digitalRead(pushButton) == 0)
        ; // wait for button release
    delay(2000); // Give time to move hand
}
```

3.1.3.9 indexLeftEncoderCount()

```
void indexLeftEncoderCount ( )
```

ISR for left encoder.

Definition at line 88 of file custom_lab_3.ino.

```
{
    leftEncoderCount++;
    //Serial.println("Left Encoder ++");
}
```

3.1.3.10 indexRightEncoderCount()

```
void indexRightEncoderCount ( )
```

ISR for incrementing right encoder.

Definition at line 97 of file custom_lab_3.ino.

```
{
    rightEncoderCount++;
    //Serial.println("Right Encoder ++");
}
```

3.1.3.11 loop()

```
void loop ( )
```

Loop execution of the program.

Definition at line 409 of file custom_lab_3.ino.

```
{
    drive();
}
```

3.1.3.12 react_forward()

```
void react_forward ( )
```

Definition at line 373 of file custom_lab_3.ino.

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

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 shoudl never actually hit the walls.

Definition at line 359 of file custom_lab_3.ino.

```
{
    // TODO: Check which button was hit

    driveBackward(20);
    turnRight(30);
}
```

3.1.3.14 react_right()

```
void react_right ( )
```

Definition at line 366 of file custom_lab_3.ino.

```
{
    // TODO: Check which button was hit

    driveBackward(20);
    turnLeft(30);
}
```

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.

```
{
    motorLeft_PWM = 180;
    motorRight_PWM = 200;
}
```

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.

```
{
    Serial.begin(9600);
    Serial.println("Setting up.....");
    configure();
    leftPID.SetMode(AUTOMATIC);
    rightPID.SetMode(AUTOMATIC);
}
```

3.1.3.17 turnLeft()

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

Turn bot right to given degrees.

Parameters

degrees	

Definition at line 178 of file custom_lab_3.ino.

```
{
    resetPWM();
    calculateDesiredCountTurn(degrees);

    // Loop unitl the encoders read correct

    while ((desiredCount - leftEncoderCount) > 3)
    {
        adjustPWM();
        //To drive forward, motors go in the same direction

        if ((desiredCount - leftEncoderCount) > 3)
        {
            run_motor(A, motorLeft_PWM); //change PWM to your calibrations
        }
        if ((desiredCount - rightEncoderCount) > 3)
        {
            run_motor(B, -motorRight_PWM); //change PWM to your calibrations
        }
    }

    // motors stop
    run_motor(A, 0);
    run_motor(B, 0);
    Serial.println("Done driving Left");
    Serial.print("L: ");
    Serial.println(leftEncoderCount);
    Serial.print("R: ");
    Serial.println(rightEncoderCount);
}
```

3.1.3.18 turnRight()

```
void turnRight (
             int degrees )
```

Turn bot to given degrees.

Parameters

degrees	

Definition at line 141 of file custom_lab_3.ino.

```
{
    resetPWM(); // Reset pwm
    calculateDesiredCountTurn(degrees);
    // While the encoders are not correct adjust PWM with PID loop
    // Loop unitl the encoders read correct

    while ((desiredCount - rightEncoderCount) > 3)
    {
        adjustPWM();
        //To drive forward, motors go in the same direction

        if ((desiredCount - leftEncoderCount) > 3)
        {
            run_motor(A, -motorLeft_PWM); //change PWM to your calibrations
        }
        if ((desiredCount - rightEncoderCount) > 3)
        {
            run_motor(B, motorRight_PWM); //change PWM to your calibrations
        }
    }

    // motors stop
    run_motor(A, 0);
    run_motor(B, 0);
    Serial.println("Done driving Right");
    Serial.print("L: ");
    Serial.println(leftEncoderCount);
    Serial.print("R: ");
    Serial.println(rightEncoderCount);
}
```

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.

```
{
  // if using dual motor driver
  // define driver pins as outputs
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  // initialize all pins to zero
  digitalWrite(IN1, 0);
  digitalWrite(IN2, 0);
  digitalWrite(IN3, 0);
  digitalWrite(IN4, 0);
  return;
} // end function
```

3.2.1.2 run_motor()

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

Definition at line 19 of file motors.ino.

```
{
  int dir = (pwm / abs(pwm)) > 0; // returns if direction is forward (1) or reverse (0)
  pwm = abs(pwm);                 // only positive values can be sent to the motor

  switch (motor)
  {       // find which motor to control
  case A: // if A, write A pins
    if (dir)
    {                         // If dir is forward
      analogWrite(IN1, pwm);  // IN1 is the forward pwm pin
      digitalWrite(IN2, LOW); // IN2 is low
    }
    else
    {
      digitalWrite(IN1, LOW); // IN1 is low
      analogWrite(IN2, pwm);  // IN2 is the reverse pwm pin
    }                         // end if
    break;                    // end case A
  case B:                     // if B, write B pins
    if (dir)
    {                         // if dir is forward
      analogWrite(IN3, pwm);  // IN3 is the forward pwm pin
      digitalWrite(IN4, LOW); // IN4 is low
    }
    else
    {
      digitalWrite(IN3, LOW); //IN3 is low
      analogWrite(IN4, pwm);  // IN4 is the reverse pwm pin
    }                         // end if
    break;                    // end case B
  }                           // end switch case
  return;
} // end function
```

Index