```
/**
   @file custom_lab_3.ino
   @author Christian Prather
   @brief A basic feedback controlled system for an Arduino based robot
   @version 0.1
   @date 2020-10-21

*/

/*! \mainpage Lab 3 Code Documentation
 *
 */


/// Libraries for interrupts and PID
#include <PinChangeInt.h>
#include <PID_v1.h>

/// Global Defines

/// Motor driver connections
#define IN1 9
#define IN2 10
#define IN3 5
#define IN4 6

/// Motor control
#define A 0
#define B 1
#define pwmA 3
#define dirA 12
#define pwmB 11
#define dirB 13

/// Start stop button
#define pushButton 2

/// Drive constants - dependent on robot configuration
#define EncoderCountsPerRev 12.0
#define DistancePerRev 51.0
#define DegreesPerRev 27.0

#define EncoderMotorLeft 7
#define EncoderMotorRight 8

/// Lab specific variables
double leftEncoderCount = 0;
double rightEncoderCount = 0;

/// Enum defines
#define FORWARD 0
#define LEFT 1
#define RIGHT -1

/// Default motor pwm values
int motorLeft_PWM = 180;
int motorRight_PWM = 200;

/// Time it takes to move 90 degrees
int milliSecondsPer90Deg = 900;
```

```
61
62 /// How many encoder counts for given distance
63 double desiredCount;
64
65 // Global array for tracking move order (move, distance) or (move, degree)
66 int moveList[] = {FORWARD, 300, LEFT, 90, FORWARD, 300, LEFT, 90, FORWARD,
   300, RIGHT, 90, FORWARD, 900, RIGHT, 90, FORWARD, 600, RIGHT, 90, FORWARD,
   300};
67
68 /**
69    @brief PID values
70    setpoints = desired counts, output = PWM, input = current counts
71 */
72 double leftOutput, rightOutput;
73 PID leftPID(&leftEncoderCount, &leftOutput, &desiredCount, 2, 5, 2, DIRECT);
74 PID rightPID(&rightEncoderCount, &rightOutput, &desiredCount, 2, 5, 2,
   DIRECT);
75
76 /**
77    @brief Helper function for setting the PWM back to default value
78 */
79 void resetPWM()
80 {
81     motorLeft_PWM = 180;
82     motorRight_PWM = 200;
83 }
84
85 /**
86    @brief ISR for left encoder
87 */
88 void indexLeftEncoderCount()
89 {
90     leftEncoderCount++;
91     //Serial.println("Left Encoder ++");
92 }
93
94 /**
95    @brief ISR for incrementing right encoder
96 */
97 void indexRightEncoderCount()
98 {
99     rightEncoderCount++;
100    //Serial.println("Right Encoder ++");
101 }
102
103 /**
104    @brief Calculate how many encoder counts we expect given the distance
   provided
105    based on the bot intrinsics
106
107    @param distance
108 */
109 void calculateDesiredCount(int distance)
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 }
120 
121 /**
122  * @brief Calculate how many encoder counts we expect given the degrees provided
123  *
124  * @param degrees
125  */
126 void calculateDesiredCountTurn(int degrees)
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 }
135 
136 /**
137    @brief Turn bot to given degrees
138 
139    @param degrees
140 */
141 void turnRight(int degrees)
142 {
143     resetPWM(); // Reset pwm
144     calculateDesiredCountTurn(degrees);
145     // While the encoders are not correct adjust PWM with PID loop
146     // Loop unitl 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 }
172 
173 /**
174    @brief Turn bot right to given degrees
175 
```

```
   @param degrees
*/
void turnLeft(int degrees)
{
    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);
}

/**
   @brief Function to drive bot forward until encoders are within range

   @param distance
*/
void driveForward(int distance)
{
    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
```

```
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 }
247 
248 /**
249    @brief Drive the bot backwards
250 
251    @param distance
252 */
253 void driveBackward(int distance)
254 {
255     resetPWM();
256     calculateDesiredCount(distance);
257 
258     // Loop unitl 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 }
284 
285 /**
286    @brief Function for configuration of pin states and interrupts
287 */
288 void configure()
289 {
290     // set up the motor drive ports
291     pinMode(pwmA, OUTPUT);
292     pinMode(dirA, OUTPUT);
293     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);
}

/**
   @brief Default behavior when not driving, waits for the pushButton to
   be pressed so it can execute next command
   Blocking function
*/
void idle()
{
    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
}

/**
   @brief Run the PID loop calculation and set out put to motors output in
PWM

*/
void adjustPWM()
{
    // 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);
}

/**
   @brief Entry point of program handles serial setup and PID config
*/
void setup()
{
    Serial.begin(9600);
    Serial.println("Setting up.....");
    configure();
```

```
351     leftPID.SetMode(AUTOMATIC);
352     rightPID.SetMode(AUTOMATIC);
353 }
354 
355 /**
356    @brief This is the logic to execute if we hit a push button
357    ideally this is never executed as we shoudl never actually hit the walls
358 */
359 void react_left()
360 {
361     // TODO: Check which button was hit
362 
363     driveBackward(20);
364     turnRight(30);
365 }
366 void react_right()
367 {
368     // TODO: Check which button was hit
369 
370     driveBackward(20);
371     turnLeft(30);
372 }
373 void react_forward()
374 {
375     // TODO: Check which button was hit
376     driveBackward(50);
377 }
378 
379 /**
380    @brief Main drive execution of program, iterates through moves list
    executing
381    next move with corresponding distance or degrees
382 */
383 void drive()
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 }
405 
406 /**
407    @brief Loop execution of the program
408 */
409 void loop()
```

```
410 {
411     drive();
412 }
413
```