```
//Comal Virdi
 // Group 3
 #include <CPutil.h>
 #include <CPE123_EncoderLib.h>

 // Simple sketch to just test a motor

 // Define our pins
 const int leftMotorPin1 = 7;
 const int leftMotorPin2 = 6;
 const int rightMotorPin1 = 5;
 const int rightMotorPin2 = 4;
 const int myButton = 10;

 const int rightEncoderPin1 = 20;
 const int rightEncoderPin2 = 21;
 const int leftEncoderPin1 = 2;
 const int leftEncoderPin2 = 3;

 Button button(myButton);

 void setup()
 {
  // put your setup code here, to run once:

  Serial.begin(9600);
  setupMessage(__FILE__, "Simple Motor Test sketch");
  delay(500);

  motorSetup();
  encoderSetup(rightEncoderPin1, rightEncoderPin2, leftEncoderPin1,
leftEncoderPin2);
  waitOnButton(button);

  /*Serial.println ("test 1");
  while (robotBackward(60, 250) != true)
  {}
  delay (4000);

  Serial.println ("test 1.5");
  while (robotBackward(100, 150) != true)
  {
    Serial.println ("test 2");
  }
  delay (1000);
```

```
  Serial.println ("test 3");
  */

 }



 void loop()
 {
  drivingControl();
 }

 void drivingControl()
 {
  enum {START, STRAIGHT1, LEFT, STRAIGHT2, RIGHT1, STRAIGHT3, RIGHT2, STRAIGHT4,
STOP};
  static int state = START;


  switch(state)
  {
    case START:
      state = STRAIGHT1;
    break;

    case STRAIGHT1:
     if (robotForward(60, 250))
      {
       state = LEFT;
       Serial.println("straight1");
      }
    break;

    case LEFT:
     if (robotLeft(45, 250))
      {
         state = STRAIGHT2;
         Serial.println("left");
      }
    break;

    case STRAIGHT2:
      if(robotForward(40, 250))
       {
         state = RIGHT1;
```

```
        Serial.println("straight2");
       }
    break;

    case RIGHT1:
      if (robotRight(90, 250))
        {
         state = STRAIGHT3;
         Serial.println("right1");
        }
    break;

    case STRAIGHT3:
      if (robotForward(50, 250))
       {
         state = RIGHT2;
         Serial.println("straight3");
       }
    break;

    case RIGHT2:
      if (robotRight(45, 250))
      {
         state = STRAIGHT4;
         Serial.println("right2");
      }
    break;

    case STRAIGHT4:
      if (robotForward(50, 250))
       {
         state = STOP;
         Serial.println("straight4");
       }
    break;

    case STOP:
      robotStop();
      // state = START;
    break;
  }
}

void motorTest()
{
```

```
 print2 ("count for 20 cm:" , calcDistance(20));
 print2 ("count for 480 cm:" , calcDistance(480));
 print2 ("count for 700 cm:" , calcDistance(700));
 print2 ("count for 20 degrees:" , calcAngle(20));
 print2 ("count for 48 degrees:" , calcAngle(48));
 print2 ("count for 70 degrees:" , calcAngle(70));

 delay(1000);
 }

void motorSetup()
{

  // Initalize the pins for output
  pinMode(leftMotorPin1, OUTPUT);
  pinMode(leftMotorPin2, OUTPUT);
  pinMode(rightMotorPin1, OUTPUT);
  pinMode(rightMotorPin2, OUTPUT);

   // Stop the motor
  analogWrite(leftMotorPin1, 0);
  analogWrite(leftMotorPin2, 0);
  analogWrite(rightMotorPin1, 0);
  analogWrite(rightMotorPin2, 0);

  robotStop();

}


int robotForward(int distanceInCm, int aSpeed)
{

 enum {START, STOP};
 static int state = START;
 int returnValue = false;
 static unsigned long totalCount = 0;

 switch(state)
  {
    case START:
      totalCount = calcDistance(distanceInCm) + rightEncoderCount();
      returnValue = false;
      state = STOP;
```

```
    break;

    case STOP:

     if (rightEncoderCount() >= totalCount)
      {
        robotStop();
        returnValue = true;
        state = START;
      }
     else
      {
        leftMotorForward(aSpeed);
        rightMotorForward(aSpeed);
      }
    break;
  }

 return returnValue;
}

int robotBackward(int distanceInCm, int aSpeed)
{

 enum {START, STOP};
 static int state = START;
 int returnValue = false;
 static unsigned long totalCount = 0;

 switch(state)
  {
    case START:
      totalCount = calcDistance(distanceInCm) + rightEncoderCount();
      returnValue = false;
      state = STOP;
    break;

    case STOP:

     if (rightEncoderCount() >= totalCount)
      {
        robotStop();
        returnValue = true;
        state = START;
      }
```

```
    else
      {
        leftMotorBackward(aSpeed);
        rightMotorBackward(aSpeed);
      }
    break;
  }

 return returnValue;
}

int robotLeft(int turnAngle, int aSpeed)
{

 enum {START, STOP};
 static int state = START;
 int returnValue = false;
 static unsigned long expectedTransitions =0;

 switch(state)
  {
    case START:
      expectedTransitions = calcAngle(turnAngle) +rightEncoderCount();
      returnValue = false;
      state = STOP;
    break;

    case STOP:

     if (rightEncoderCount() >= expectedTransitions)
      {
        robotStop();
        returnValue = true;
        state = START;
      }
     else
      {
        leftTurn(aSpeed);
      }
    break;
  }

 return returnValue;
}
```

```
int robotRight(int turnAngle, int aSpeed)
{

 enum {START, STOP};
 static int state = START;
 int returnValue = false;
 static unsigned long expectedTransitions = 0;

 switch(state)
  {
    case START:
     expectedTransitions = calcAngle(turnAngle) + leftEncoderCount();
     returnValue = false;
     state = STOP;
    break;

    case STOP:

    if (leftEncoderCount() >= expectedTransitions)
     {
        robotStop();
        returnValue = true;
        state = START;
     }
    else
     {
        rightTurn(aSpeed);
     }
    break;
  }

 return returnValue;
}

int robotSpin(int spinTime)
{
 enum {FORWARD, STOP};
 static int state = FORWARD;
 static MSTimer timer;
 int returnValue = false;

 switch(state)
 {
    case FORWARD:
      timer.set(spinTime);
```

```
      returnValue = false;
      state = STOP;
    break;

    case STOP:

     if (timer.done())
      {
        robotStop();
        returnValue = true;
        state = FORWARD;
      }
     else
      {
        rightMotorForward(250);
        leftMotorBackward(250);
      }
    break;
 }

 return returnValue;
}
void motorControl(int pin1, int pin2, int aSpeed)
 {
  analogWrite(pin1, aSpeed);
  analogWrite(pin2, 0);
 }

void leftMotorForward(int aSpeed)
 {
  motorControl(leftMotorPin1, leftMotorPin2, aSpeed);
 }

void rightMotorForward(int aSpeed)
 {
  motorControl(rightMotorPin1, rightMotorPin2, aSpeed);
 }

void leftMotorBackward(int aSpeed)
 {
  motorControl(leftMotorPin2, leftMotorPin1, aSpeed);
 }

void rightMotorBackward(int aSpeed)
 {
```

```
  motorControl(rightMotorPin2, rightMotorPin1, aSpeed);
 }

void leftMotorStop()
 {
  motorControl(leftMotorPin1, leftMotorPin2, 0);
 }

void rightMotorStop()
 {
  motorControl(rightMotorPin1, rightMotorPin2, 0);
 }

void leftTurn(int aSpeed)
 {
  rightMotorForward(aSpeed);
  leftMotorStop();
 }

void rightTurn(int aSpeed)
 {
  leftMotorForward(aSpeed);
  rightMotorStop();
 }

void robotStop()
 {
  leftMotorStop();
  rightMotorStop();
 }

unsigned long calcAngle(int aAngle)
 {
  return aAngle * 37;
 }

unsigned long calcDistance(unsigned long aDistance)
 {
  return aDistance * 111;
 }

void waitOnButton(Button & button)
 {
  Serial.println ("Waiting on Button Push");
  while (button.wasPushed() == false)
```

```
  { } // note infinite loop until button is pushed
}
```