Section04 - Line Following with PID Control

By using the measured error and PID gain values, the error is corrected, enabling smoother and more precise driving. Compared to when Bang-Bang Control was used, Domabot rarely deviates from the black line and does less frequent yawing, resulting in smoother and more stable movement.

Code

```
//File: lfbb_pid.nxc
//Date: 02/12/2025 22:36
//Desc: Domabot Lego NXT Light Sensor calibration for line following
//      Motor Right(portA), Moter Left(portC), Light Sensor(port4)
//      Domabot yaws back-and-forth to define min and max IR values
//      and display them and calculated threshold value

task main() {

  //Variable declarations-----------------------------------------------------
  bool orangeBTNPushed, rightBTNPushed, leftBTNPushed; //NXT buttons
  int irMin, irMax, irValue, irThresh; //[0,100] light sensor values
  int speed, speedSlow, speedBase; //[0,100] motor speed
  unsigned long endTime; //[ms] for calibration stopwatch

  float Kp, Ki, Kd; //PID gains
  float lE, lEDif, lEInt; //line error and its derivative and integral
  float lEPrev; //line error previous value
  float lECorr; //line related motor power correction
  int speedLeft, speedRight;

  unsigned int result;
  byte fileHandle;
  short bytesWritten;
  string fileName, fileHeader, text;
  string strIteration, strirValue;
  int iteration;

  //Variable initializations--------------------------------------------------
  irMin = 999; //Max light sensor value. Set to high value it'll never reach
  irMax = -1; //Minlight sensor value. Set to low value it'll never reach
  speedBase = 50; //Domabot's default speed. Tune for faster/slower response
  speedSlow = 15; //Used to sweep from white to black and black to white line
  Kp = 2;
  Ki = 0.01;
  Kd = 25;
  lE = lEDif = lEInt = lEPrev = 0.0;
  iteration = 0;
```

```
//File-----------------------------------------------------------------
fileName = "lf_i01.csv";
result = CreateFile(fileName, 1024, fileHandle);

//Overwrite existing file
while (result == LDR_FILEEXISTS) {
  CloseFile(fileHandle);
  DeleteFile(fileName);
  result = CreateFile(fileName, 1024, fileHandle);
}

//Write column headers
fileHeader = "Iteration, Sensor Value";
WriteLnString(fileHandle, fileHeader, bytesWritten);

//Algorithm begins-----------------------------------------------------
TextOut(0, LCD_LINE1, "Orange Btn starts");
do{ //nothing until orng btn pushed
  orangeBTNPushed = ButtonPressed(BTNCENTER, FALSE);
} while(!orangeBTNPushed);

SetSensorLight(IN_4); //initialize IR sensor for 200ms
Wait(200);

//Place light sensor on outside of black track as depicted below
//Assumes IR sensor is on outer border of oval
//Step1: Turn left(CCW) calibration - will rotate IR sensor from white to black

endTime = CurrentTick() + 2000; //2000 msec stopwatch
OnFwd(OUT_A, speedSlow); //Right motor forward, Left motor backward
OnRev(OUT_C, speedSlow); //results in yawing slowly CCW from white to black
while(CurrentTick() < endTime) {
  irValue = Sensor(IN_4);
  if(irValue > irMax) {
    irMax = irValue;
  } else if(irValue < irMin) {
    irMin = irValue;
  } //end if-else
} //end while - and we now have max and min light sensor values

//By now, the light sensor has yawed onto or past the black line
//Step2: So yaw CW to rotate IR sensor from black to white
endTime = CurrentTick() + 3000; //3000msec stopwatch
OnFwd(OUT_C, speedSlow); //Left motor forward, Right motor backward
OnRev(OUT_A, speedSlow); //results in yawing slowly CW from white to black
```

```
while(CurrentTick() < endTime) {
  irValue = Sensor(IN_4);
  if(irValue > irMax) {
    //robot rotates CCW for 3s gathering IR values
    irMax = irValue;
  } else if(irValue < irMin) {
    irMin = irValue;
  } //end if-else
} //end while

//Step3: Calculate and display threshold value until user hits Right Button
Off(OUT_AC);
PlaySound(SOUND_UP);
irThresh = (irMin + irMax)/2; //avg of min and max IR values
ClearScreen();
TextOut(0, LCD_LINE1, "Calibration values");
TextOut(0, LCD_LINE2, FormatNum("irMax = %d", irMax));
TextOut(0, LCD_LINE3, FormatNum("irMin = %d", irMin));
TextOut(0, LCD_LINE4, FormatNum("irThresh = %d", irThresh));
TextOut(0, LCD_LINE6, ">>BTN to proceed");

do{
  rightBTNPushed = ButtonPressed(BTNRIGHT, FALSE);
} while(!rightBTNPushed);
ClearScreen();

//Step4: Find the black line
//Yaw CCW again, towards outermost black edge
endTime = CurrentTick() + 2000; //2000 msec stopwatch
OnFwd(OUT_A, speedSlow); //Right motor fwd, Left motor bwd
OnRev(OUT_C, speedSlow);
while(irValue > irThresh) {
  irValue = Sensor(IN_4); //read light sensor value
} //end whlie

//Step5: PID Line Following
do{
  leftBTNPushed = ButtonPressed(BTNLEFT, FALSE);
  irValue = Sensor(IN_4);

  //Claculate error
  lE = irValue - irThresh;
  lEInt += lE;
  lEDif = lE - lEPrev;
  lECorr = (Kp * lE) + (Ki * lEInt) + (Kd * lEDif);
```

```
    //adjust motor speed
    speedLeft = speedBase - IECorr;
    speedRight = speedBase + IECorr;

    //motor
    OnFwd(OUT_A, speedRight);
    OnFwd(OUT_C, speedLeft);

    IEPrev = IE;

    //Convert data to string and write to file
    strIteration = FormatNum("%d", iteration);
    strirValue = FormatNum("%d", irValue);
    text = StrCat(strIteration, ",", strirValue);

    result = WriteLnString(fileHandle, text, bytesWritten);
    if(result == LDR_EOFEXPECTED) CloseFile(fileHandle);

    iteration++;

  } while(!leftBTNPushed);

  //Step6: User hit Left Arrow BTN
  //User Pushed Left Button, so exit gracefully
  Off(OUT_AC);
  PlaySound(SOUND_DOUBLE_BEEP);
  Wait(SEC_5);
  StopAllTasks();
} //end main
```