Тема 3 · Урок 3.2

П-регулятор: плавно и быстро

Робот поворачивает тем сильнее, чем дальше ушла линия, и проходит дуги без рывков.

2 × 45 мин | 1–2 ученика на робота | Практика

РЕЗУЛЬТАТ УРОКА

Робот плавно едет по линии с П-регулятором и проходит дугу своего уровня. Коэффициент и скорость подобраны и записаны в журнал.

Ты научишься

- Перевести показания двух датчиков в общую шкалу от 0 до 100.
- Посчитать ошибку как разницу показаний и объяснить её знак.
- Объяснить, как поправка меняет скорости колёс.
- Подобрать коэффициент и скорость по наблюдениям на дорожке.

Нужно знать

- 3.1 Датчики линии и первое следование
- 2.1 Моторы: едем и останавливаемся

Понадобится

Датчик линии (аналоговый, Trema) × 2 | USB-кабель для Arduino

Зачем этот урок

В уроке 3.1 робот ехал по линии рывками: колесо либо крутилось, либо стояло. На крутых дугах итоговой задачи так не проехать. Сегодня робот научится поворачивать **чуть-чуть**, когда линия ушла немного, и **сильно**, когда ушла далеко. Так работают автопилоты и роботы-курьеры.

РАЗМИНКА · ВСПОМНИ

1. Как программа из урока 3.1 решала, что под датчиком линия?
2. Почему робот в уроке 3.1 ехал рывками?
3. Что делает функция `drive()`?

ЗАНЯТИЕ 1 · 45 МИН

1 Ошибка вместо «да или нет»

1 Плавно — это как?

8 мин

Что должен делать робот. Роботу нужно двигаться по линии плавнее: при маленьком отклонении чуть поправлять курс, а при большом — сильнее. Используем те же датчики отражения и моторы, добавив П-регулятор — правило расчёта поправки.

Что используем и как это работает. Сначала переводим измерения каждого датчика в общую шкалу по его белому и чёрному. Разность уровней называется ошибкой. Коэффициент kp умножает ошибку: чем больше отклонение, тем больше поправка. Одному колесу прибавляем её к базовой команде, у другого вычитаем. Новое измерение позволяет снова скорректировать движение — это обратная связь.

Как связать измерение с действием. Слишком малая поправка не успевает вернуть робота, слишком большая вызывает колебания. Меняем один параметр за раз и сравниваем траекторию и показания на одинаковой трассе. Нулевая ошибка означает равные показания, но не всегда правильное положение: оба датчика могут потерять линию.

Представь, что ты ведёшь велосипед по линии. Если линия ушла чуть влево, ты чуть поворачиваешь руль. Если сильно — поворачиваешь сильнее. Ты не бросаешь руль из крайнего положения в крайнее.

1. Встань с напарником: один — «робот», второй держит линию (шнур или полоску бумаги) и плавно смещает её влево и вправо.
2. «Робот» поворачивает корпус тем сильнее, чем дальше линия от середины.

Если ты один: положи на стол изогнутый шнур и веди вдоль него ладонью — это «робот». Поворачивай ладонь тем сильнее, чем дальше шнур ушёл от её середины.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты можешь объяснить: сила поворота зависит от того, насколько далеко ушла линия.

2 Общая шкала для двух датчиков

15 мин

У каждого датчика свои числа белого и чёрного. Чтобы сравнивать их, переведём оба в одну шкалу: **чёрное — 0, белое — 100**, край линии — где-то посередине.

1. Создай скетч `L32_Error`. Перенеси из журнала свои измерения белого и чёрного.

Шкала и ошибка L32_Error.ino

```
// Наблюдение в MyRobotica включается сборщиком; обычный Arduino работает без
#include <math.h>
#if defined(ROBOT_TELEMETRY_BUILD_ID)
#include <RobotTelemetry.h>
robot_telemetry::TelemetryStream courseTelemetry;
#endif

const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;

// Измерения белого и чёрного из журнала (урок 3.1).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;

// Переводит показание в шкалу: чёрное — 0, белое — 100.
int level(int value, int white, int black) {
  long result = (long)(value - black) * 100 / (white - black);
  return constrain(result, 0, 100);
}

void setup() {
  courseTelemetryBegin();
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    Serial.println("Впиши измерения белого и чёрного");
    while (true) {
      courseTelemetryPoll();
    }
  }
}

void loop() {
  int leftRaw = analogRead(LEFT_SENSOR_PIN);
  int rightRaw = analogRead(RIGHT_SENSOR_PIN);
  int leftLevel = level(leftRaw, LEFT_WHITE, LEFT_BLACK);
  int rightLevel = level(rightRaw, RIGHT_WHITE, RIGHT_BLACK);
  int error = leftLevel - rightLevel;

  courseTelemetryReport(leftRaw, rightRaw, error, NAN);
  Serial.print("L=");
```

```
  Serial.print(leftLevel);
  Serial.print("  R=");
  Serial.print(rightLevel);
  Serial.print("  ошибка=");
  Serial.println(error);
  courseWait(200);
}

// ===== Готовое: передача наблюдений — не меняй =====
void courseTelemetryBegin() {
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  using namespace robot_telemetry::wire;
  courseTelemetry.begin((1UL << Line0) | (1UL << Line1) |
                        (1UL << LineError) | (1UL << ControlOutput));
#else
  Serial.begin(9600);
#endif
}

void courseTelemetryPoll() {
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  courseTelemetry.update();
#endif
}

// NAN означает «ещё не вычислено», а не нулевую ошибку или поправку.
void courseTelemetryReport(int leftRaw, int rightRaw, float error, float outp
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  using namespace robot_telemetry::wire;
  courseTelemetry.setUnsigned(Line0, leftRaw);
  courseTelemetry.setUnsigned(Line1, rightRaw);
  if (isnan(error)) courseTelemetry.setMissing(LineError);
  else courseTelemetry.setFloat(LineError, error);
  if (isnan(output)) courseTelemetry.setMissing(ControlOutput);
  else courseTelemetry.setFloat(ControlOutput, output);
  courseTelemetry.update();
#endif
}

void courseWait(unsigned long durationMs) {
  unsigned long started = millis();
  while (millis() - started < durationMs) {
    courseTelemetryPoll();
```

```
    delay(1);
  }
}
```

2. Загрузи, открой монитор порта. Проверь: на белом оба числа около 100, на чёрном — около 0.
3. Медленно проведи край линии под левым датчиком. Как меняется число?

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Оба датчика показывают около 100 на белом и около 0 на чёрном, на краю линии — промежуточные числа.

> ⓘ **Как это устроено: Как работает функция level**
>
> `value - black` — насколько показание отличается от чёрного. `white - black` — весь путь от чёрного до белого. Их отношение, умноженное на 100, — «сколько процентов пути до белого». Формула работает, даже если у датчика чёрное — большое число, а белое — маленькое. `constrain(…, 0, 100)` обрезает случайные выбросы за границы шкалы.

3 Считаем ошибку

15 мин

Ошибка — разница показаний: `error = leftLevel - rightLevel`. Когда линия ровно посередине, оба датчика видят одинаково и ошибка равна нулю.

ПРЕДСКАЖИ

Линия ушла под левый датчик. Какой знак будет у ошибки — плюс или минус? А если под правый?

1. Посмотри на строку с ошибкой в мониторе порта.
2. Подвинь линию под левый датчик, потом под правый. Запиши в журнал знак и примерные числа.

Где линия	leftLevel	rightLevel	Ошибка
Посередине			
Под левым			
Под правым			

ДОЛЖНО ПОЛУЧИТЬСЯ

Линия слева — ошибка отрицательная, справа — положительная, посередине — около нуля.

2

ЗАНЯТИЕ 2 · 45 МИН

П-регулятор на дорожке

4 От ошибки к скоростям

10 мин

Поправка — ошибка, умноженная на коэффициент: `correction = kp * error`. К левому колесу поправку прибавляем, у правого — отнимаем.

1. Создай скетч `L32_P_Line`, перенеси измерения.

П-регулятор L32_P_Line.ino

```
// Наблюдение в MyRobotica включается сборщиком; обычный Arduino работает без
#include <math.h>
#if defined(ROBOT_TELEMETRY_BUILD_ID)
#include <RobotTelemetry.h>
robot_telemetry::TelemetryStream courseTelemetry;
#endif

const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;

// Измерения белого и чёрного из журнала (урок 3.1).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;

// Переводит показание в шкалу: чёрное — 0, белое — 100.
int level(int value, int white, int black) {
  long result = (long)(value - black) * 100 / (white - black);
  return constrain(result, 0, 100);
}

const int LEFT_DIR_PIN = 7;
const int LEFT_SPEED_PIN = 6;
const int RIGHT_DIR_PIN = 4;
const int RIGHT_SPEED_PIN = 5;
const int FORWARD = HIGH;

float kp = 0.5;     // сила поворота: подбираем на дорожке
int baseSpeed = 100; // скорость на прямой

void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
  digitalWrite(LEFT_DIR_PIN, leftDir);
  digitalWrite(RIGHT_DIR_PIN, rightDir);
  analogWrite(LEFT_SPEED_PIN, leftSpeed);
  analogWrite(RIGHT_SPEED_PIN, rightSpeed);
}

void setup() {
  courseTelemetryBegin();
  pinMode(LEFT_DIR_PIN, OUTPUT);
```

```
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  drive(FORWARD, 0, FORWARD, 0);

  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
      courseTelemetryPoll();
    }
  }
  courseWait(2000);
}

void loop() {
  int leftRaw = analogRead(LEFT_SENSOR_PIN);
  int rightRaw = analogRead(RIGHT_SENSOR_PIN);
  int leftLevel = level(leftRaw, LEFT_WHITE, LEFT_BLACK);
  int rightLevel = level(rightRaw, RIGHT_WHITE, RIGHT_BLACK);

  int error = leftLevel - rightLevel;
  int correction = kp * error;

  int leftSpeed = constrain(baseSpeed + correction, 0, 255);
  int rightSpeed = constrain(baseSpeed - correction, 0, 255);
  drive(FORWARD, leftSpeed, FORWARD, rightSpeed);
  courseTelemetryReport(leftRaw, rightRaw, error, correction);
}

// ===== Готовое: передача наблюдений — не меняй =====
void courseTelemetryBegin() {
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  using namespace robot_telemetry::wire;
  courseTelemetry.begin((1UL << Line0) | (1UL << Line1) |
                        (1UL << LineError) | (1UL << ControlOutput));
#else
  Serial.begin(9600);
#endif
}

void courseTelemetryPoll() {
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  courseTelemetry.update();
```

```
#endif
}

// NAN означает «ещё не вычислено», а не нулевую ошибку или поправку.
void courseTelemetryReport(int leftRaw, int rightRaw, float error, float outp
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  using namespace robot_telemetry::wire;
  courseTelemetry.setUnsigned(Line0, leftRaw);
  courseTelemetry.setUnsigned(Line1, rightRaw);
  if (isnan(error)) courseTelemetry.setMissing(LineError);
  else courseTelemetry.setFloat(LineError, error);
  if (isnan(output)) courseTelemetry.setMissing(ControlOutput);
  else courseTelemetry.setFloat(ControlOutput, output);
  courseTelemetry.update();
#endif
}

void courseWait(unsigned long durationMs) {
  unsigned long started = millis();
  while (millis() - started < durationMs) {
    courseTelemetryPoll();
    delay(1);
  }
}
```

2. Посчитай на бумаге: `baseSpeed = 100`, `kp = 0.5`, ошибка −40. Какие скорости получат колёса?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты посчитал: поправка −20, левое колесо 80, правое 120 — робот поворачивает налево, к линии.

ⓘ **Как это устроено: Зачем constrain**

При большой ошибке скорость может получиться −30 или 300. Мотор понимает только 0…255. `constrain(число, 0, 255)` ставит границы: меньше нуля станет 0, больше 255 — 255.

5 Проверь на подставке

7 мин

1. Поставь робота на подставку, загрузи программу, включи питание моторов.
2. Подведи линию под левый датчик: левое колесо замедлится, правое ускорится.
3. Подведи под правый — наоборот. Линия посередине — колёса крутятся одинаково.

ДОЛЖНО ПОЛУЧИТЬСЯ

Колёса меняют скорость плавно, а не «стоп — едет».

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Колёса не крутятся	Вписаны ли измерения? Включено ли питание моторов?
Ускоряется не то колесо	Проверь знак ошибки по монитору порта и формулы скоростей

6 Подбери коэффициент

18 мин

Коэффициент `kp` подбирают на дорожке. Меняй **одно число за раз** и смотри, что изменилось.

1. Поставь робота на дорожку с дугой. Попробуй `kp` = 0.2, 0.5, 0.8 при одной скорости.
2. Запиши в журнал, как робот ведёт себя при каждом значении.
3. Выбери лучший `kp` и попробуй увеличить `baseSpeed`.

`kp`	`baseSpeed`	Что видно
0.2	100	
0.5	100	
0.8	100	

Как это устроено: Как понять, куда менять kp

- Робот плавно уходит с дуги наружу — поправка слишком слабая, увеличь `kp`.
- Робот виляет из стороны в сторону даже на прямой — поправка слишком сильная, уменьши `kp`.
- Хороший `kp` — робот держится линии на дуге и почти не виляет на прямой.

ДОЛЖНО ПОЛУЧИТЬСЯ

Подобраны `kp` и `baseSpeed`, при которых робот проходит дугу. Числа записаны в журнал.

7 Задания по уровням

10 мин

ОСНОВНОЕ · ДУГА НАЧАЛЬНОГО ПОЛЯ

Пройти дугу радиусом 350 мм, как на начальном поле итоговой задачи, не теряя линию.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить знак ошибки, когда линия слева и справа.
- [ ] Я могу посчитать скорости колёс по ошибке, kp и baseSpeed.
- [ ] Я подобрал kp по наблюдениям, меняя одно число за раз.

Проверь себя

Основное

1. Левый датчик показывает 20, правый — 100. Чему равна ошибка error = leftLevel - rightLevel?

- ○ 80
- ○ −80
- ○ 120

2. baseSpeed = 100, kp = 0.5, ошибка = 40. Какие скорости у колёс?

- ○ Левое 120, правое 80
- ○ Левое 80, правое 120
- ○ Оба 140

3. Робот сильно виляет даже на прямой. Что сделать с kp?

- ○ Увеличить
- ○ Уменьшить
- ○ Сделать отрицательным

4. Зачем переводить показания датчиков в шкалу 0–100?

- ○ Чтобы показания двух разных датчиков можно было сравнивать
- ○ Чтобы программа работала быстрее
- ○ Arduino не понимает чисел больше 100

5. Что делает constrain(baseSpeed + correction, 0, 255)?

- ○ Не даёт скорости выйти за пределы 0…255
- ○ Делит скорость на 255
- ○ Останавливает робота

Отвечай на вопросы своего уровня. Нужно верных ответов: Начальный — 4 из 5; Средний — 4 из 5; Продвинутый — 4 из 5. Ответы проверит преподаватель или программа курса.

Итог

Робот поворачивает пропорционально ошибке — это П-регулятор. Он плавнее правила «стоп — едет» и проходит крутые дуги. Дальше робот научится находить перекрёстки на линии.

Как подтвердить результат

Для каждого опыта сохраняй условия, версию программы, наблюдение и вывод. Сравнивая попытки, меняй одну причину за раз; неудачные попытки тоже остаются в журнале. Отметка результата относится к уже выполненным действиям, повторять их ради отметки не нужно. Если результат не получен, запиши расхождение и следующую проверяемую гипотезу. Модель, измерения настоящего робота и объяснение — разные виды доказательств.

Словарь урока

ошибка	Насколько линия ушла от середины: разница показаний левого и правого датчиков.
поправка	На сколько изменить скорости колёс, чтобы вернуться к линии.
коэффициент	Число kp: во сколько раз поправка больше или меньше ошибки.
П-регулятор	Правило «поправка пропорциональна ошибке»: больше ушли — сильнее поворачиваем.
constrain	Команда, которая не даёт числу выйти за границы, например за 0…255.