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()`?

1

ЗАНЯТИЕ 1 · 45 МИН

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.

Исследуй проблему, прежде чем решать.

1. Запусти свою программу из урока 3.1 на дуге 350 мм, потом на дуге 125 мм. Запиши, как едет робот: рывки, виляние, сход с линии.
2. Объясни в журнале, какую информацию с датчиков программа 3.1 выбрасывает и почему из-за этого робот едет рывками.
3. **До того как читать дальше**, придумай своё правило: как скорости колёс должны зависеть от показаний датчиков. Запиши словами или формулой. В конце урока сравни его с П-регулятором.

Подсказка: Если не знаешь, с чего начать

Программа 3.1 сравнивает показание с порогом и получает «да или нет». А что, если использовать само число?

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале — наблюдение на двух дугах, объяснение причины рывков и твоё правило управления.

2 Общая шкала для двух датчиков

15 мин

Ты не начинаешь с чистого листа: работай в **своей программе из урока 3.1**. В ней уже есть контакты, измерения белого и чёрного и защита от запуска без измерений. Сохрани её копию как `L32_P_Line` — за урок ты превратишь её в П-регулятор, шаг за шагом, по требованиям.

Подсказка: Нет своей программы из урока 3.1?

Возьми каркас: в нём только контакты, управление моторами и защита. Всё остальное напишешь сам.

Каркас П-регулятора L32_P_Line_Skeleton.ino

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

// Каркас П-регулятора. Требования к программе — в тексте урока, шаг з
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;

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;

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);
}

// Требование 1: шкала «чёрное — 0, белое — 100».
int level(int value, int white, int black) {
  return 0;
}

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. Эту защиту не удаляй.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
      courseTelemetryPoll();
    }
  }
  courseWait(2000);
}

void loop() {
  float observedError = NAN;
  float observedOutput = NAN;
  // Требования 2 и 3: ошибка, поправка и скорости колёс.
  // Для графиков сохрани свои результаты в observedError и observedOu
  drive(FORWARD, 0, FORWARD, 0);
  courseTelemetryReport(analogRead(LEFT_SENSOR_PIN), analogRead(RIGHT_
                        observedError, observedOutput);
}

// ===== 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, flo
#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);
  }
}
```

ⓘ Как это устроено: Справка по командам

Команда или тип	Что делает
`analogRead(pin)`	Показание датчика: 0…1023
`analogWrite(pin, v)`	Скорость мотора: `v` от 0 до 255
`digitalWrite(pin, HIGH)` / `LOW`	Направление мотора
`constrain(x, a, b)`	`x`, но не меньше `a` и не больше `b`
`Serial.begin(9600)`	Включает монитор порта, пишут один раз в `setup()`
`Serial.print(x)` / `Serial.println(x)`	Вывод в монитор порта; `println` — с переводом строки
`int` на Arduino Uno	Целые от −32 768 до 32 767
`long` на Arduino Uno	Целые от −2 147 483 648 до 2 147 483 647
`/` с целыми числами	Дробная часть отбрасывается: `7 / 2` даёт `3`

Требование 1. Функция `level(value, white, black)` переводит показание в шкалу 0…100:

- чёрное — 0, белое — 100, между ними — по прямой;
- работает и когда у датчика чёрное — большое число, а белое — маленькое;
- никогда не выходит за 0…100;
- не переполняется на Arduino Uno.

Приёмка. Напиши `level` в тестовом скетче, загрузи его и открой монитор порта. Тест проверяет восемь случаев, в том числе датчик «наоборот» и показания за пределами шкалы. Когда все строки `PASS`, перенеси функцию в свою программу и проверь шкалу обоих датчиков на дорожке.

Тест функции level L32_Level_Test.ino

```ino
// Проверка функции level. Напиши функцию здесь, загрузи и открой монитор пор
// Когда все строки PASS, перенеси level в свою программу.

// Требование 1: шкала «чёрное — 0, белое — 100».
int level(int value, int white, int black) {
  return 0;
}

int failed = 0;

void expect(int value, int white, int black, int expected) {
  int actual = level(value, white, black);
  Serial.print(actual == expected ? "PASS " : "FAIL ");
  Serial.print("level(");
  Serial.print(value);
  Serial.print(", ");
  Serial.print(white);
  Serial.print(", ");
  Serial.print(black);
  Serial.print(") = ");
  Serial.print(actual);
  Serial.print(", ожидаем ");
  Serial.println(expected);
  if (actual != expected) {
    failed++;
  }
}

void setup() {
  Serial.begin(9600);

  // Датчик, у которого чёрное — большое число, а белое — маленькое.
  expect(780, 120, 780, 0);
  expect(120, 120, 780, 100);
  expect(450, 120, 780, 50);
  // За пределами шкалы.
  expect(900, 120, 780, 0);
  expect(50, 120, 780, 100);
  // Датчик, у которого наоборот: белое — большое число, чёрное — маленькое.
  expect(120, 900, 120, 0);
  expect(510, 900, 120, 50);
```

```
  expect(900, 900, 120, 100);

  if (failed == 0) {
    Serial.println("Все проверки пройдены");
  } else {
    Serial.print("Ошибок: ");
    Serial.println(failed);
  }
}

void loop() {
}
```

Подсказка: Идея (1 из 3)

Шкала — это доля пути от чёрного до белого. Весь путь — разница белого и чёрного.

Подсказка: План (2 из 3)

1. Насколько показание ушло от чёрного?
2. Раздели на длину всего пути и умножь на 100.
3. Обрежь результат границами шкалы.

Подсказка: Фрагмент кода (3 из 3)

`long result = (long)(value - black) * 100 / (white - black);` — почему здесь `long` и почему умножение стоит раньше деления?

ДОЛЖНО ПОЛУЧИТЬСЯ

Тест показывает «Все проверки пройдены». На дорожке шкала показывает около 0 на чёрном и около 100 на белом.

3 Считаем ошибку

15 мин

Требование 2. Программа считает ошибку — число со знаком: насколько и в какую сторону линия ушла от середины. Знак выбираешь ты, но потом поправка должна поворачивать робота **к линии**.

1. Добавь расчёт ошибки и временный вывод `leftLevel`, `rightLevel`, ошибки в монитор порта.
2. Заполни таблицу и запиши в журнал своё правило знака одной фразой.

Где линия	leftLevel	rightLevel	Ошибка
Посередине			
Под левым			
Под правым			

3. Разберись с краем: линия ушла **за пределы обоих датчиков**. Что покажет ошибка и что тогда сделает робот? Запиши, чем это опасно на крутой дуге.

Подсказка: Идея (1 из 2)

Если линия ровно посередине, оба датчика видят одно и то же — ошибка должна быть ноль.

Подсказка: Проверка знака (2 из 2)

Выбери: `left - right` или `right - left`. Запиши, какое колесо должно ускориться, когда линия слева, — это понадобится в требовании 3.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена, правило знака записано. Ты знаешь, что видит регулятор, когда линия ушла за оба датчика.

2

ЗАНЯТИЕ 2 · 45 МИН

П-регулятор на дорожке

4 От ошибки к скоростям

10 мин

Требование 3. Робот едет вперёд со скоростью `baseSpeed` и поворачивает к линии тем сильнее, чем больше ошибка:

- поправка пропорциональна ошибке, коэффициент `kp`;
- `kp` и `baseSpeed` — именованные переменные в начале скетча, их подбирают на дорожке;
- скорости колёс всегда в пределах 0…255;
- без измерений белого и чёрного колёса стоят (защита из урока 3.1 остаётся).

Приёмка на бумаге. Посчитай скорости колёс для своей программы:

`baseSpeed`	`kp`	Ошибка	Левое	Правое
100	0.5	−40		
200	0.8	60		
200	1.0	100		

В последней строке одно колесо упирается в границу. Насколько слабее робот повернёт, чем «хотел»? Запиши в журнал, что можно с этим сделать.

Подсказка: Идея (1 из 3)

Одно колесо получает базовую скорость плюс поправку, другое — минус поправку.

Подсказка: План (2 из 3)

1. Ошибка из требования 2.
2. Поправка = коэффициент × ошибка.
3. Левое и правое колесо: база ± поправка, каждое — в границах 0…255.
4. Отправить скорости на моторы.

Подсказка: Фрагмент кода (3 из 3)

`int leftSpeed = constrain(baseSpeed + correction, 0, 255);` — а что для правого? Проверь знак по своему правилу из требования 2.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование 3, таблица посчитана, ограничение записано в журнал.

5 Проверь на подставке

7 мин

Приёмка требования 3 на подставке.

1. Временно выведи в монитор порта ошибку, поправку и обе скорости колёс.
2. Подводи линию под левый датчик, правый и посередине. Совпадает ли направление поправки с твоим правилом знака из требования 2?
3. Найди положение линии, при котором одно колесо упирается в 0 или 255. Сравни скорости в мониторе с тем, что ты посчитал в таблице требования 3.
4. Отключи измерения (нули) — колёса стоят? Верни измерения.

ДОЛЖНО ПОЛУЧИТЬСЯ

Направление поправки, упор в границу и защита проверены по монитору порта. Расхождения с таблицей объяснены.

6 Подбери коэффициент

18 мин

Подбери `kp` и скорость **методом**, а не наугад, и докажи выбор записями.

1. При `baseSpeed = 100` начни с `kp = 0.1` и удваивай: 0.1, 0.2, 0.4, 0.8... Найди наименьший `kp`, при котором робот начинает вилять **на прямой**.
2. Рабочий `kp` возьми примерно вдвое меньше. Проверь его на дуге своего уровня.
3. Поднимай `baseSpeed` шагами по 20. Для каждой скорости — три заезда. Запомни последнюю скорость, на которой робот прошёл все три.
4. Запиши в журнал таблицу и вывод: почему именно эта пара, что ограничивает скорость.
5. Вернись к правилу, которое ты придумал в первом шаге. Чем оно похоже на П-регулятор и чем отличается? Что бы ты в нём изменил теперь?

`kp`	`baseSpeed`	Заезды (из 3)	Что видно

Подсказка: Робот виляет при любом kp

Проверь, не упирается ли колесо в 255 или 0 на дуге — вспомни последнюю строку таблицы из требования 3.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица с методом подбора, пара `kp – baseSpeed`, проверенная тремя заездами, вывод с обоснованием и сравнение своего правила с П-регулятором.

7 Задания по уровням

10 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · КРУТАЯ S-СВЯЗКА**

Пройти две дуги радиусом 125 мм подряд в разные стороны, как на продвинутом поле. Начни с пары `kp — скорость`, выбранной в шаге подбора. Если она не удерживает S-связку, измени один параметр и дополни существующую таблицу результатом; заново переписывать удачные и неудачные попытки не нужно.

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

★★ Продвинутый уровень

1. На чёрном и у края линии шкала верна, но для показания 50 при white = 120 и black = 780 функция возвращает неверное число. В чём ошибка?

```
int level(int value, int white, int black) {
  int result = (value - black) * 100 / (white - black);
  return constrain(result, 0, 100);
}
```

- ○ (50 − 780) × 100 = −73 000 не помещается в int Arduino Uno
- ○ constrain не работает с отрицательными числами
- ○ Нельзя делить на отрицательное число

2. Что вернёт эта функция для показания 450 при white = 120 и black = 780?

```
int level(int value, int white, int black) {
  long result = (long)(value - black) / (white - black) * 100;
  return constrain(result, 0, 100);
}
```

- ○ 50
- ○ 0
- ○ 100

3. Ошибку посчитали как rightLevel - leftLevel, а поправку по-прежнему прибавляют к левому колесу. Линия ушла под левый датчик. Что сделает робот?

- ○ Повернёт от линии и потеряет её
- ○ Повернёт к линии, как обычно
- ○ Остановится

4. На крутой дуге линия ушла за пределы обоих датчиков. Что сделает П-регулятор из урока?

- ○ Оба датчика видят белое, ошибка около нуля — робот поедет прямо и потеряет линию
- ○ Ошибка станет максимальной — робот резко развернётся к линии
- ○ Сработает защита — робот остановится

5. **baseSpeed = 200, kp = 1.0, ошибка = 100, скорости ограничены 0…255. Что будет с поворотом?**

- ○ Поворот слабее расчётного: левое колесо упёрлось в 255, разница колёс 155 вместо 200
- ○ Поворот такой же, как рассчитан: constrain на него не влияет
- ○ Робот повернёт в другую сторону

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.