Тема 7 · Урок Ф.2

Регуляторы: релейный, П, ПД и ПИД на одной трассе

Робот сам измеряет, насколько точно едет, и команда по числам выясняет, какой регулятор лучше.

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

РЕЗУЛЬТАТ УРОКА

Команда сравнила релейный, П-, ПД- и ПИД-регуляторы на одной трассе и одной скорости по средней ошибке, наибольшей ошибке и числу переходов через линию и объяснила результат. Не нужен для итоговой задачи.

Ты научишься

- Провести честное сравнение: одна трасса, одна скорость, одно изменение за раз, три заезда.
- Объяснить, чем релейный регулятор отличается от П-регулятора.
- Объяснить, на что реагируют Д- и И-части регулятора.
- Сделать вывод о регуляторах по измерениям, а не по впечатлению.

Нужно знать

- 3.1 Датчики линии и первое следование
- 3.2 П-регулятор: плавно и быстро

Понадобится

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

Зачем этот урок

В курсе робот ездил по линии с двумя регуляторами: релейным в уроке 3.1 («колесо едет или стоит») и П-регулятором в уроке 3.2 («чем дальше линия, тем сильнее поворот»). Инженеры используют ещё два: **ПД** и **ПИД**. Какой из четырёх лучше? Сегодня ответим не «на глаз», а по числам: робот сам измерит, насколько точно проехал, а команда сравнит **все четыре регулятора** на одной трассе и одной скорости.

Факультатив не нужен для итоговой задачи. Он нужен тем, кто хочет понять, как устроены настоящие автопилоты, и проверить это опытом.

РАЗМИНКА · ВСПОМНИ

1. Что делает колесо в программе урока 3.1, когда под его датчиком линия?
2. Как в уроке 3.2 считали ошибку и поправку?
3. Почему при подборе `kp` меняли одно число за раз?

1

ЗАНЯТИЕ 1 · 45 МИН

Измеряем, а не на глаз

1 Как сравнить честно

8 мин

Что должен делать робот. Роботу нужно удерживать линию при разных отклонениях. Сравним четыре способа выбирать команды колёсам на тех же датчиках и моторах, чтобы понимать, что меняет именно алгоритм.

Что используем и как это работает. Релейное правило выбирает одно из заранее заданных действий. П-составляющая реагирует на текущую ошибку, Д — на изменение ошибки между шагами, И — на накопленную ошибку. В этом скетче шаг фиксирован, поэтому коэффициенты настроены для его длительности. Накопление ограничивается, чтобы большая прошлая ошибка не давала чрезмерную поправку после возвращения к линии.

Как связать измерение с действием. Проводим заезды с одинаковой базовой командой и на одном поле, сравниваем отклонения и колебания. Больший коэффициент не означает лучший результат. Проверка формулы на числах помогает найти ошибку кода, но не заменяет испытания на трассе; коэффициенты теста не являются настройками вашего робота.

Сначала спланируй опыт, потом пиши код.

1. Выберите трассу сравнения: прямой участок и дуга 250 мм из урока 3.2. Отметьте старт.
2. Запиши в журнал план опыта: - что меняется (одно); - что остаётся одинаковым (не меньше пяти пунктов); - что измеряем и почему этих чисел достаточно, чтобы назвать лучший регулятор; - сколько заездов делаем и какое число берём из трёх, если один заезд сильно отличается.
3. Запиши прогноз: порядок четырёх регуляторов по средней ошибке.

Подсказка: Какое число брать из трёх заездов

Если один заезд из трёх сильно отличается, среднее арифметическое «утянет» его за собой. Есть число, которое на один случайный заезд не реагирует.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале план опыта и прогноз. Ты знаешь, какое число из трёх заездов пойдёт в таблицу.

2 Робот измеряет сам

15 мин

Каркас ниже берёт на себя измерение: итоги заезда копятся в `record(error)`, сохраняются в память после `RUN_MS` и печатаются при следующем запуске. Остальное напишешь сам.

Каркас: измерение заезда LF2_Compare_Skeleton.ino

```
// Наблюдение в MyRobotica включается сборщиком; обычный Arduino работает без
#include <math.h>
struct Result; // Тип итогов до прототипов, которые добавляет Arduino.
#if defined(ROBOT_TELEMETRY_BUILD_ID)
#include <RobotTelemetry.h>
robot_telemetry::TelemetryStream courseTelemetry;
#endif

// Каркас для сравнения регуляторов. Измерение заезда и запуск готовы;
// регуляторы напиши сам по требованиям из текста урока.
#include <EEPROM.h>

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;

// Регулятор заезда: 0 — релейный, 1 — П, 2 — ПД, 3 — ПИД.
const int MODE = 1;
int baseSpeed = 100; // одна скорость для всех регуляторов
float kp = 0.5;      // твой kp из урока 3.2
float kd = 0;        // Д: реакция на то, как быстро уходит линия
float ki = 0;        // И: реакция на ошибку, которая копится
const long SUM_LIMIT = 3000;       // предел накопленной ошибки
const unsigned long RUN_MS = 6000; // длина заезда: робот должен остановиться
const unsigned long STEP_MS = 10;  // шаг регулятора: 100 раз в секунду

int lastError = 0;
long errorSum = 0;

// Итоги заезда: хранятся в памяти Arduino и после выключения.
```

```
struct Result {
  int marker;
  int mode;
  int baseSpeed;
  float kp;
  float kd;
  float ki;
  long steps;
  long sumAbs;
  int maxAbs;
  int crossings;
};
// Требование 2: здесь будет int controller(int mode, int error).

// Перенеси свою функцию level из урока 3.2.
int level(int value, int white, int black) {
  return 0;
}

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);
}

// ===== Измерение заезда =====

const int MARKER = 2610; // признак того, что в памяти записан заезд
Result run;
int lastSide = 0;
unsigned long startTime;
unsigned long nextStep;

// Добавляет ошибку шага к итогам заезда.
void record(int error) {
  run.steps++;
  run.sumAbs += abs(error);
  if (abs(error) > run.maxAbs) {
    run.maxAbs = abs(error);
  }
  int side = 0;
  if (error > 10) {
```

```
    side = 1;
  }
  if (error < -10) {
    side = -1;
  }
  if (side != 0 && lastSide != 0 && side != lastSide) {
    run.crossings++;
  }
  if (side != 0) {
    lastSide = side;
  }
}

void printResult(Result r) {
  const char *names[] = {"релейный", "П", "ПД", "ПИД"};
  Serial.print("Регулятор: ");
  Serial.print(names[r.mode]);
  Serial.print("  скорость=");
  Serial.print(r.baseSpeed);
  Serial.print("  kp=");
  Serial.print(r.kp);
  Serial.print("  kd=");
  Serial.print(r.kd);
  Serial.print("  ki=");
  Serial.println(r.ki, 3);
  Serial.print("Средняя ошибка: ");
  Serial.println(r.steps > 0 ? (float)r.sumAbs / r.steps : 0, 1);
  Serial.print("Наибольшая ошибка: ");
  Serial.println(r.maxAbs);
  Serial.print("Переходов через линию: ");
  Serial.println(r.crossings);
}

// Ждёт, пока робот стоит на старте: оба датчика видят белое 1 секунду подряд
void waitForStart() {
  Serial.println("Поставь робота на старт: линия между датчиками");
  unsigned long whiteSince = millis();
  while (millis() - whiteSince < 1000) {
    courseTelemetryPoll();
    int leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLACK
    int rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_B
    if (leftLevel < 70 || rightLevel < 70) {
      whiteSince = millis();
```

```
    }
  }
}

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);

  Result last;
  EEPROM.get(0, last);
  if (last.marker == MARKER && last.mode >= 0 && last.mode <= 3) {
    Serial.println("Прошлый заезд:");
    printResult(last);
  } else {
    Serial.println("Записанных заездов нет");
  }

  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    Serial.println("Впиши измерения белого и чёрного");
    while (true) {
      courseTelemetryPoll();
    }
  }
  waitForStart();
  courseWait(2000); // время убрать руки от робота

  run = {MARKER, MODE, baseSpeed, kp, kd, ki, 0, 0, 0, 0};
  startTime = millis();
  nextStep = startTime;
}

void loop() {
  if (millis() - startTime >= RUN_MS) {
    drive(FORWARD, 0, FORWARD, 0);
    EEPROM.put(0, run);
    Serial.println("Заезд окончен:");
    printResult(run);
    courseTelemetryReport(analogRead(LEFT_SENSOR_PIN), analogRead(RIGHT_SENSO
```

```
    while (true) {
      courseTelemetryPoll();
    }
  }
  while (millis() < nextStep) {
    courseTelemetryPoll();
  }
  nextStep += STEP_MS;

  float observedError = NAN;
  float observedOutput = NAN;
  // Требования 1–3: ошибка, record(error), регулятор по MODE и скорости колё
  // Для графиков сохрани свои результаты в observedError и observedOutput.
  drive(FORWARD, 0, FORWARD, 0);
  courseTelemetryReport(analogRead(LEFT_SENSOR_PIN), analogRead(RIGHT_SENSOR_
                        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, 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);
  }
}
```

Требование 1. Одна константа `MODE` выбирает регулятор: 0 — релейный, 1 — П, 2 — ПД, 3 — ПИД. Всё остальное во всех режимах одинаково.

- Релейный — по правилу урока 3.1, но на шкале 0...100: колесо, под датчиком которого линия, стоит, другое едет со скоростью `baseSpeed`.
- П, ПД и ПИД получают поправку из функции `controller(mode, error)` и делят её между колёсами, как в уроке 3.2.
- Каждый шаг ошибка передаётся в `record(error)` — во всех четырёх режимах.

Сегодня напиши `level`, ошибку, релейный режим и П-часть `controller`. Д и И — на втором занятии.

1. Перенеси `level` и ошибку из урока 3.2.
2. Выполни требование 1 для режимов 0 и 1. Подбери `RUN_MS` так, чтобы робот останавливался до конца дорожки.
3. Найди в `record` порог 10. Запиши в журнал, что изменится в числе переходов, если порог убрать, и почему.

> ⚠ **БЕЗОПАСНОСТЬ**
>
> Подключай кабель, держа робота в руке или на подставке: на белом столе программа примет стол за старт.

> 💡 **Подсказка: Робот не трогается со старта**
>
> `waitForStart` ждёт белого по шкале `level`. Пока `level` возвращает 0, робот не стартует никогда — это защита каркаса.

ДОЛЖНО ПОЛУЧИТЬСЯ

Режимы 0 и 1 работают, `RUN_MS` подобрано, в журнале — ответ про порог.

3 Релейный против П

20 мин

1. Проведи опыт по своему плану для режимов 0 и 1.
2. Сравни не только средние ошибки, но и **разброс** трёх заездов: насколько повторяется результат каждого регулятора?
3. Запиши в журнал объяснение через устройство регулятора: почему у релейного больше переходов и наибольшая ошибка, хотя оба видят одни и те же датчики.

ДОЛЖНО ПОЛУЧИТЬСЯ

Данные двух регуляторов, разброс и объяснение — в журнале. Прогноз из первого шага проверен для первой пары.

2

ЗАНЯТИЕ 2 · 45 МИН

ПД, ПИД и общий итог

4 ПД: тормозим поворот заранее

14 мин

Требование 2. Функция `int controller(int mode, int error)`:

- режим 1 — только П-часть;
- режимы 2 и 3 — плюс Д-часть: `kd`, умноженный на изменение ошибки с прошлого шага;
- режим 3 — плюс И-часть: ошибки копятся в `errorSum`, сумма не выходит за `±SUM_LIMIT`, поправка `ki × errorSum`; в других режимах сумма не копится;
- прошлая ошибка обновляется на каждом шаге во всех режимах.

Приёмка. Напиши `controller` в тестовом скетче, загрузи и открой монитор порта. Когда все строки `PASS`, перенеси функцию в свою программу. Сегодня тест проверяет и И-часть — значит, её тоже нужно написать.

Тест функции controller LF2_Controller_Test.ino

```
// Проверка функции controller. Напиши функцию здесь, загрузи и открой монито
// Когда все строки PASS, перенеси controller в свою программу.
// Коэффициенты в тесте свои, удобные для счёта; на дорожке будут твои.

float kp = 0.5;
float kd = 2.0;
float ki = 0.25;
const long SUM_LIMIT = 200;

int lastError = 0;
long errorSum = 0;

// Требование 2: режим 1 — П, 2 — ПД, 3 — ПИД.
int controller(int mode, int error) {
  return 0;
}

int failed = 0;

void reset() {
  lastError = 0;
  errorSum = 0;
}

void expect(const char *name, int actual, int expected) {
  Serial.print(actual == expected ? "PASS " : "FAIL ");
  Serial.print(name);
  Serial.print(": получили ");
  Serial.print(actual);
  Serial.print(", ожидаем ");
  Serial.println(expected);
  if (actual != expected) {
    failed++;
  }
}

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

  reset();
  expect("П, ошибка 40", controller(1, 40), 20);
```

```
  expect("П, ошибка -40", controller(1, -40), -20);

  reset();
  expect("ПД, первый шаг 10", controller(2, 10), 25);
  expect("ПД, ошибка выросла до 20", controller(2, 20), 30);
  expect("ПД, ошибка осталась 20", controller(2, 20), 10);

  reset();
  expect("ПИД, первый шаг 10", controller(3, 10), 27);
  expect("ПИД, снова 10", controller(3, 10), 10);

  // Накопленная ошибка не выходит за SUM_LIMIT.
  reset();
  int last = 0;
  for (int i = 0; i < 5; i++) {
    last = controller(3, 100);
  }
  expect("ПИД, пять шагов по 100", last, 100);
  reset();
  for (int i = 0; i < 5; i++) {
    last = controller(3, -100);
  }
  expect("ПИД, пять шагов по -100", last, -100);

  // В режиме П ошибка не копится, но прошлая ошибка запоминается.
  reset();
  controller(1, 100);
  controller(1, 100);
  controller(1, 100);
  expect("ПИД после трёх шагов П", controller(3, 100), 75);

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

void loop() {
}
```

Затем подбери `kd` методом: начни с 2.5 и удваивай до первого дёргания на прямой, возьми предыдущее значение. Три заезда — в таблицу.

Подсказка: Идея (1 из 3)

Скорость ухода линии — разница между ошибкой сейчас и ошибкой на прошлом шаге.

Подсказка: План (2 из 3)

1. П-часть.
2. Если режим 2 или 3 — прибавить Д-часть.
3. Если режим 3 — обновить сумму, ограничить её, прибавить И-часть.
4. Запомнить ошибку.

Подсказка: Фрагмент кода (3 из 3)

`errorSum = constrain(errorSum + error, -SUM_LIMIT, SUM_LIMIT);` — почему тест «пять шагов по 100» без этой строки даёт 175?

ДОЛЖНО ПОЛУЧИТЬСЯ

Тест: «Все проверки пройдены». `kd` подобран методом, три заезда ПД записаны.

5 ПИД: что даёт накопленная ошибка

11 мин

1. Поставь `MODE = 3`, подбери `ki` методом: от 0.005, удваивая. Три заезда с лучшим — в таблицу.
2. Опыт с пределом: один заезд с `SUM_LIMIT = 30000`. Опиши, что происходит после дуги, и объясни через значение `errorSum`.
3. Ответь в журнале: в каких условиях И-часть даёт выигрыш и есть ли они на нашей трассе?

Подсказка: Когда И-часть полезна

Подумай о долгой ошибке одного знака: длинная дуга одного радиуса или робот, у которого одно колесо всегда крутится медленнее другого.

ДОЛЖНО ПОЛУЧИТЬСЯ

`ki` подобран, опыт с пределом описан, условия пользы И-части записаны.

6 Сравнение всех четырёх

12 мин

1. Составь итоговую таблицу всех четырёх регуляторов с разбросом трёх заездов.
2. Напиши вывод из трёх частей: что показали числа, чем это объясняется в устройстве регуляторов, что можно было сделать нечестно и как ты это исключил.
3. Сравни результат с прогнозом и планом опыта из первого шага.

ДОЛЖНО ПОЛУЧИТЬСЯ

Итоговая таблица и вывод из трёх частей — в журнале.

7 Задания по уровням

8 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · ПРЕДЕЛЬНАЯ СКОРОСТЬ**

Найти для каждого из четырёх регуляторов наибольшую `baseSpeed`, на которой он проходит трассу три раза из трёх: повышать шагами по 20. Составить рейтинг по скорости и сравнить его с рейтингом по средней ошибке.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, чем отличаются релейный, П, ПД и ПИД.
- [ ] Я сравнил регуляторы, меняя только одно и делая по три заезда.
- [ ] Я могу назвать лучший регулятор для нашей трассы и подтвердить это числами.

Проверь себя

★★ Продвинутый уровень

1. С этой функцией ПД ведёт себя как П с увеличенным kp. Почему?

```
int controller(int mode, int error) {
  float correction = kp * error;
  if (mode >= 2) {
    correction += kd * (error - lastError);
  }
  return correction;
}
```

- ○ lastError нигде не обновляется и остаётся 0, поэтому Д-часть равна kd × ошибка
- ○ kd умножается на kp
- ○ Д-часть работает только в режиме 3

2. В ПИД убрали ограничение суммы. После длинной дуги влево робот на прямой долго уходит влево за линию. Почему?

- ○ Сумма ошибок, накопленная на дуге, продолжает поворачивать робота, пока ошибки другого знака её не «отработают»
- ○ Д-часть перестаёт работать без ограничения
- ○ Датчики на прямой видят хуже

3. Средняя ошибка и переходы: релейный 30 и 40, П 14 и 6, ПД 9 и 2, ПИД 10 и 3. Какой вывод честный?

- ○ На этой трассе и скорости ПД лучше всех по обоим числам, ПИД выигрыша не дал
- ○ ПИД лучше всех, потому что в нём больше частей
- ○ Релейный лучше: у него больше переходов, значит он чаще поправляется

4. Почему на трассе из дуг в разные стороны И-часть почти не помогает?

- ○ Ошибка часто меняет знак: сумма не успевает стать полезной и запаздывает за трассой
- ○ И-часть работает только на прямой
- ○ На дугах сумма всегда равна нулю

5. В программе написали correction -= kd * (error - lastError). Что будет с ПД?

- ○ Д-часть будет раскачивать робота: он завиляет сильнее, чем с П
- ○ Ничего: знак Д-части не важен
- ○ Робот поедет точнее

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

Релейный регулятор знает только «да или нет», П-регулятор — «насколько», Д-часть — «как быстро», И-часть — «как долго». Больше частей не значит лучше: решают

измерения на твоей трассе и скорости.

Как подтвердить результат

Результат опыта связывает условия, версию программы, наблюдение и вывод. Сравнение попыток с одним изменением помогает проверить причину, включая неудачные попытки. Отметка относится к уже выполненной практике; повтор ради отметки не нужен. Расхождение с прогнозом задаёт следующую проверяемую гипотезу. Модель, измерения настоящего робота и объяснение — разные виды доказательств.

Словарь урока

релейный регулятор	Правило «всё или ничего»: колесо либо едет, либо стоит, как в уроке 3.1.
П-часть	Поправка, пропорциональная ошибке: kp × ошибка. Регулятор из урока 3.2.
Д-часть	Поправка на то, как быстро меняется ошибка: kd × (ошибка сейчас − ошибка на прошлом шаге).
И-часть	Поправка на ошибку, накопленную за время: ki × сумма ошибок.
средняя ошибка	Насколько в среднем линия уходила от середины за заезд, без учёта стороны. Чем меньше, тем точнее робот едет.
шаг регулятора	Постоянный промежуток между расчётами поправки, здесь 10 мс.