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.

Сравнение честное, только если меняется **одно**: регулятор. Трасса, старт, скорость, измерения и длина заезда — одни и те же. Каждый регулятор — три заезда.

1. Выберите трассу сравнения: прямой участок и дуга 250 мм из урока 3.2. Отметьте старт.
2. Запиши в журнал список того, что должно оставаться одинаковым во всех заездах. Найди не меньше пяти пунктов.
3. Перерисуй таблицу сравнения.

Регулятор	Заезд	Средняя ошибка	Наибольшая ошибка	Переходов через линию	Сошёл с линии?
Релейный	1, 2, 3				
П	1, 2, 3				
ПД	1, 2, 3				
ПИД	1, 2, 3				

ПРЕДСКАЖИ

Расставь четыре регулятора от лучшего к худшему по средней ошибке и объясни порядок одной фразой.

ДОЛЖНО ПОЛУЧИТЬСЯ

Список одинаковых условий, таблица и прогноз с объяснением — в журнале.

2 Робот измеряет сам

15 мин

Программа ведёт робота заданное время `RUN_MS`, останавливает его и сохраняет итоги заезда в память, которая не стирается без питания. При следующем запуске итоги печатаются в монитор порта. Измерение заезда в шаблоне готово. Твоя часть — функция `controller`: сегодня в ней П-регулятор, на втором занятии добавишь Д и И.

1. Создай скетч `LF2_Compare` по шаблону, перенеси измерения и свой `kp`.
2. Напиши в `controller` только П-часть — как в уроке 3.2.

Шаблон: функция регулятора LF2_Compare_Gaps.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

// Шаблон среднего уровня. Напиши код в разделе «Твоя часть» по целям в комме

#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;
};

// ===== Твоя часть: напиши сам =====

// Поправка П-, ПД- или ПИД-регулятора по ошибке этого шага.
int controller(int mode, int error) {
  // Занятие 1. П: поправка пропорциональна ошибке, коэффициент kp.
  // Занятие 2. Д, режимы 2 и 3: прибавь kd, умноженный на то, насколько ошиб
  // Занятие 2. И, только режим 3: копи ошибку в errorSum, не выпуская сумму
  // В конце запомни ошибку этого шага в lastError: она нужна Д на следующем
  return 0;
}

// ===== Готовое: шкала, моторы, измерение заезда и запуск — не меняй =====

// 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 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);
}

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);
}
```

```
128
129 // Ждёт, пока робот стоит на старте: оба датчика видят белое 1 секунду подряд
130 void waitForStart() {
131   Serial.println("Поставь робота на старт: линия между датчиками");
132   unsigned long whiteSince = millis();
133   while (millis() - whiteSince < 1000) {
134     courseTelemetryPoll();
135     int leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLACK
136     int rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_B
137     if (leftLevel < 70 || rightLevel < 70) {
138       whiteSince = millis();
139     }
140   }
141 }
142
143 void setup() {
144   courseTelemetryBegin();
145   pinMode(LEFT_DIR_PIN, OUTPUT);
146   pinMode(LEFT_SPEED_PIN, OUTPUT);
147   pinMode(RIGHT_DIR_PIN, OUTPUT);
148   pinMode(RIGHT_SPEED_PIN, OUTPUT);
149   drive(FORWARD, 0, FORWARD, 0);
150
151   Result last;
152   EEPROM.get(0, last);
153   if (last.marker == MARKER && last.mode >= 0 && last.mode <= 3) {
154     Serial.println("Прошлый заезд:");
155     printResult(last);
156   } else {
157     Serial.println("Записанных заездов нет");
158   }
159
160   // Без измерений робот не поедет.
161   if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
162     Serial.println("Впиши измерения белого и чёрного");
163     while (true) {
164       courseTelemetryPoll();
165     }
166   }
167   waitForStart();
168   courseWait(2000); // время убрать руки от робота
169
170   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;

  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;
  record(error);

  float observedOutput = NAN; // у релейного алгоритма нет пропорциональной п
  int leftSpeed;
  int rightSpeed;
  if (MODE == 0) {
    // Релейный, как в уроке 3.1: колесо над линией стоит.
    leftSpeed = leftLevel < 50 ? 0 : baseSpeed;
    rightSpeed = rightLevel < 50 ? 0 : baseSpeed;
  } else {
    int correction = controller(MODE, error);
    observedOutput = correction;
    leftSpeed = constrain(baseSpeed + correction, 0, 255);
    rightSpeed = constrain(baseSpeed - correction, 0, 255);
  }
  drive(FORWARD, leftSpeed, FORWARD, rightSpeed);
  courseTelemetryReport(leftRaw, rightRaw, error, observedOutput);
}
```

```
214
215 // ===== Готовое: передача наблюдений — не меняй =====
216 void courseTelemetryBegin() {
217 #if defined(ROBOT_TELEMETRY_BUILD_ID)
218   using namespace robot_telemetry::wire;
219   courseTelemetry.begin((1UL << Line0) | (1UL << Line1) |
220                         (1UL << LineError) | (1UL << ControlOutput));
221 #else
222   Serial.begin(9600);
223 #endif
224 }
225
226 void courseTelemetryPoll() {
227 #if defined(ROBOT_TELEMETRY_BUILD_ID)
228   courseTelemetry.update();
229 #endif
230 }
231
232 // NAN означает «ещё не вычислено», а не нулевую ошибку или поправку.
233 void courseTelemetryReport(int leftRaw, int rightRaw, float error, float outp
234 #if defined(ROBOT_TELEMETRY_BUILD_ID)
235   using namespace robot_telemetry::wire;
236   courseTelemetry.setUnsigned(Line0, leftRaw);
237   courseTelemetry.setUnsigned(Line1, rightRaw);
238   if (isnan(error)) courseTelemetry.setMissing(LineError);
239   else courseTelemetry.setFloat(LineError, error);
240   if (isnan(output)) courseTelemetry.setMissing(ControlOutput);
241   else courseTelemetry.setFloat(ControlOutput, output);
242   courseTelemetry.update();
243 #endif
244 }
245
246 void courseWait(unsigned long durationMs) {
247   unsigned long started = millis();
248   while (millis() - started < durationMs) {
249     courseTelemetryPoll();
250     delay(1);
251   }
252 }
```

3. Поставь `MODE = 1`, загрузи, поставь робота на старт. Подбери `RUN_MS` так, чтобы робот останавливался до конца дорожки, и больше его не меняй.

4. Найди в готовой части функцию `record` и ответь в журнале: когда она добавляет переход через линию и зачем там порог 10?

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

> 💡 **Подсказка: Робот не трогается со старта**
>
> Программа ждёт, пока оба датчика целую секунду видят белое (больше 70 по шкале). Посмотри шкалу датчиков на старте.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> `controller` возвращает П-поправку, `RUN_MS` подобрано, в журнале — ответ про `record`.

3 Релейный против П

20 мин

1. Сделай по три заезда с `MODE = 0` и `MODE = 1`. Итоги — в таблицу.
2. Для каждого регулятора выбери из трёх средних ошибок **среднюю по величине** — она и пойдёт в сравнение.
3. Ответь в журнале: почему у релейного больше переходов через линию? Объясни через то, какие скорости получают колёса.

> 💡 **Подсказка: Почему средняя по величине**
>
> Если один заезд случайно вышел плохим, среднее арифметическое изменится сильно, а средняя по величине — нет.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Таблица для двух регуляторов заполнена, выбраны средние по величине, объяснение переходов записано.

ЗАНЯТИЕ 2 · 45 МИН

2 ПД, ПИД и общий итог

4 ПД: тормозим поворот заранее

14 мин

П-регулятор часто проскакивает линию: поправка не ослабевает, пока ошибка не станет нулём. **Д-часть** реагирует на то, **как быстро меняется ошибка**, и гасит поворот заранее.

1. Допиши Д-часть в `controller` по цели в шаблоне.

ПРЕДСКАЖИ

`kp = 0.5`, `kd = 4`, на прошлом шаге ошибка 10, сейчас 25. Какую поправку вернёт `controller` для `MODE = 2`? Посчитай до запуска.

2. Поставь `MODE = 2`. Подбери `kd`: начни с 2.5 и удваивай, по одному заезду. Запиши, при каком `kd` робот начал дёргаться на прямой, и возьми предыдущее значение.
3. С выбранным `kd` — три заезда в таблицу.

Подсказка: Где взять прошлую ошибку

Она уже лежит в `lastError`, если ты запоминаешь её в конце функции.

Подсказка: Робот дёргается даже при маленьком kd

Проверь знак: Д-часть прибавляется, а изменение ошибки — «сейчас минус раньше».

ДОЛЖНО ПОЛУЧИТЬСЯ

Прогноз совпал: 72. Д-часть написана, `kd` подобран удвоением, три заезда записаны.

5 ПИД: что даёт накопленная ошибка

11 мин

И-часть копит ошибку. Пока линия долго с одной стороны, сумма растёт и усиливает поворот. Опасность — сумма продолжает тянуть в ту же сторону и после дуги.

1. Допиши И-часть в `controller` по цели в шаблоне.
2. Поставь `MODE = 3`, подбери `ki`: 0.005, 0.01, 0.02. Три заезда с лучшим — в таблицу.
3. Временно поставь `SUM_LIMIT = 30000` и проедь один раз. Что изменилось после дуги? Верни 3000.

Подсказка: Как не выпустить сумму за предел

Та же команда, что держит скорость колёс в 0…255, подойдёт и для суммы.

ДОЛЖНО ПОЛУЧИТЬСЯ

И-часть написана, `ki` подобран, три заезда записаны. Ты видел, что делает робот без предела суммы.

6 Сравнение всех четырёх

12 мин

1. Составь итоговую таблицу: регулятор, средняя по величине средняя ошибка, наибольшая ошибка, переходы, сходы.
2. Назови лучший регулятор **для этой трассы и скорости** и обоснуй двумя числами.
3. Ответь: оправдала ли И-часть своё усложнение? Почему?
4. Какой регулятор ты взял бы для итоговой задачи и почему?

ДОЛЖНО ПОЛУЧИТЬСЯ

Итоговая таблица, обоснованный вывод и ответы — в журнале.

7 Задания по уровням

8 мин

★ **СРЕДНИЙ УРОВЕНЬ · ПД ЛУЧШЕ П**

Подобрать `kd`, при котором ПД проходит трассу с меньшей средней ошибкой, чем П, и объяснить по числам почему.

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

★ Средний уровень

1. kp = 0.5, kd = 4. На прошлом шаге ошибка была 10, сейчас 25. Какую поправку вернёт ПД-регулятор (дробная часть отбрасывается)?

- ○ 72
- ○ 12
- ○ 160

2. Почему ПД меньше виляет на выходе из дуги, чем П?

- ○ Когда ошибка быстро уменьшается, Д-часть заранее ослабляет поворот
- ○ Д-часть увеличивает скорость обоих колёс
- ○ Д-часть выключает моторы на выходе из дуги

3. Зачем ограничивать сумму ошибок errorSum значением SUM_LIMIT?

- ○ Иначе на долгой дуге сумма вырастет и после дуги будет тянуть робота мимо линии
- ○ Arduino не умеет хранить числа больше 3000
- ○ Чтобы робот ехал быстрее

4. Три заезда ПИД дали среднюю ошибку 8, 9 и 21. Какое число взять в сравнение?

- ○ 9 — среднее по величине; заезд с 21 разобрать отдельно
- ○ 8 — лучший заезд
- ○ 38 — сумму всех трёх

5. Зачем регулятор считает с постоянным шагом STEP_MS?

- ○ Д сравнивает ошибки соседних шагов: при разном шаге одна и та же разница значила бы разную скорость ухода линии
- ○ Иначе перегреются моторы
- ○ Иначе монитор порта не успевает печатать

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.