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.

Чтобы сравнить регуляторы, всё остальное должно быть **одинаковым**: одна дорожка, одно место старта, одна скорость, одни и те же измерения белого и чёрного. Меняется только регулятор. Каждый регулятор едет **три раза**: один заезд может оказаться случайным.

1. Выберите трассу сравнения: прямой участок и дуга 250 мм из урока 3.2. Отметьте карандашом место старта.
2. Перерисуй в журнал таблицу — её вы заполните к концу урока.

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

Какой регулятор, по-твоему, проедет точнее всех, а какой — хуже всех? Запиши в журнал.

ДОЛЖНО ПОЛУЧИТЬСЯ

Трасса и старт отмечены, таблица перерисована, прогноз записан.

2 Робот измеряет сам

15 мин

Программа ведёт робота заданное время `RUN_MS`, а потом останавливается и сохраняет итоги заезда в память Arduino. Память не стирается, когда выключаешь питание. В начале каждого запуска программа печатает в монитор порта итоги **прошлого** заезда.

Что печатает робот	Что значит
Средняя ошибка	Насколько в среднем линия уходила от середины. Чем меньше, тем точнее
Наибольшая ошибка	Самый сильный уход от линии за заезд
Переходов через линию	Сколько раз линия перескочила с одной стороны на другую: виляние

1. Создай скетч `LF2_Compare`, перенеси измерения из журнала и свой `kp` из урока 3.2.

Четыре регулятора и измерение заезда LF2_Compare.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

#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;
};

// 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);
}

// Поправка П-, ПД- или ПИД-регулятора по ошибке этого шага.
int controller(int mode, int error) {
  float correction = kp * error;
  if (mode >= 2) {
    correction += kd * (error - lastError);
  }
  if (mode == 3) {
    errorSum = constrain(errorSum + error, -SUM_LIMIT, SUM_LIMIT);
    correction += ki * errorSum;
  }
  lastError = error;
  return correction;
}

// ===== 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;

  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);
```

```
}

// ===== 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. Поставь `MODE = 1` (П-регулятор), загрузи программу.

3. Поставь робота на старт: линия между датчиками. Через секунду программа увидит старт, ещё через две — робот поедет.
4. Робот должен останавливаться **до конца дорожки**. Если доезжает до края — уменьши `RUN_MS`. Подобрал — больше `RUN_MS` не меняй.

> ⚠ **БЕЗОПАСНОСТЬ**
> Подключай кабель, держа робота в руке или на подставке. Если поставить его на белый стол, программа примет стол за старт и робот поедет.

5. Возьми робота в руки, подключи кабель и открой монитор порта: появятся итоги прошлого заезда.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> `RUN_MS` подобрано, робот останавливается до конца дорожки, в мониторе порта видны итоги заезда.

> ⓘ **Как это устроено: Почему робот ждёт старта**
>
> Когда открываешь монитор порта, Arduino перезапускается. Если бы робот сразу ехал, он уехал бы со стола и записал бы «заезд» поверх настоящего. Поэтому программа ждёт, пока оба датчика целую секунду видят белое: так бывает, когда робот стоит на старте.

3 Релейный против П

20 мин

1. Поставь `MODE = 0` — релейный регулятор из урока 3.1. Сделай три заезда. После каждого подключи кабель и перепиши итоги в таблицу.
2. Поставь `MODE = 1` — П-регулятор. Ещё три заезда.
3. Сравни строки. Какой регулятор вилял больше? У какого меньше средняя ошибка?

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
«Записанных заездов нет» после заезда	Заезд не доехал до конца `RUN_MS`: питание отключили раньше. Повтори
Робот поехал прямо со стола	Ты подключил кабель, когда робот стоял на белом. Держи его в руке
Релейный сходит с дуги	Это тоже результат: запиши «да» в «Сошёл с линии?»

ДОЛЖНО ПОЛУЧИТЬСЯ

В таблице по три заезда релейного и П-регулятора. Ты можешь сказать, какой точнее и по какому числу это видно.

2

ЗАНЯТИЕ 2 · 45 МИН

ПД, ПИД и общий итог

4 ПД: тормозим поворот заранее

14 мин

П-регулятор поворачивает, пока линия не вернулась на середину, и часто проскакивает её: робот виляет. **Д-часть** смотрит, **как быстро** меняется ошибка. Если линия быстро возвращается к середине, Д-часть заранее ослабляет поворот — как велосипедист, который начинает выпрямлять руль ещё до того, как выехал на линию.

Функция регулятора LF2_Compare.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

#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;
};

// 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);
}

// Поправка П-, ПД- или ПИД-регулятора по ошибке этого шага.
int controller(int mode, int error) {
  float correction = kp * error;
  if (mode >= 2) {
    correction += kd * (error - lastError);
  }
  if (mode == 3) {
    errorSum = constrain(errorSum + error, -SUM_LIMIT, SUM_LIMIT);
    correction += ki * errorSum;
  }
  lastError = error;
  return correction;
}

// ===== 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;

  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);
```

```
}

// ===== 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);
  }
}
```

Поправка ПД: `kp × ошибка + kd × (ошибка сейчас − ошибка на прошлом шаге)`.

1. Посчитай на бумаге: `kp = 0.5`, `kd = 4`, на прошлом шаге ошибка была 10, сейчас 25.
2. Поставь `MODE = 2` и попробуй `kd` = 5, 10 и 20 — по одному заезду на дорожке. Выбери `kd`, при котором робот меньше всего виляет.
3. С выбранным `kd` сделай три заезда и запиши итоги в таблицу.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты посчитал поправку: 12,5 + 60 = 72,5, программа отбросит дробную часть — 72. Выбран `kd`, в таблице три заезда ПД.

ⓘ **Как это устроено: Почему kd такой большой**

Регулятор считает 100 раз в секунду. За одну сотую секунды ошибка меняется всего на несколько единиц, поэтому множитель `kd` нужен больше, чем `kp`.

5 ПИД: что даёт накопленная ошибка

11 мин

И-часть складывает ошибки шаг за шагом. Если линия долго остаётся с одной стороны — например, на длинной дуге, — сумма растёт, и робот поворачивает всё сильнее. Чтобы сумма не росла бесконечно, её ограничивает `SUM_LIMIT`.

1. Оставь свой `kd`, поставь `MODE = 3` и попробуй `ki` = 0.005, 0.01 и 0.02 — по одному заезду.
2. Выбери лучший `ki` и сделай три заезда в таблицу.

? **ПРЕДСКАЖИ**

Будет ли ПИД точнее ПД на нашей трассе? Запиши прогноз до заездов.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В таблице три заезда ПИД. Прогноз проверен.

6 Сравнение всех четырёх

12 мин

1. Выпиши для каждого регулятора среднюю ошибку из трёх заездов — среднюю по величине.
2. Расставь регуляторы от лучшего к худшему.

Место	Регулятор	Средняя ошибка	Переходов через линию
1			
2			
3			
4			

3. Сравни с прогнозом из начала урока. Что совпало, что нет?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Четыре регулятора расставлены по местам, прогноз проверен.

7 Задания по уровням

8 мин

● **ОСНОВНОЕ · ВСЕ ЧЕТЫРЕ РЕГУЛЯТОРА**

Заполнить таблицу сравнения всех четырёх регуляторов на одной скорости, по три заезда, и назвать лучший 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

● Основное

1. Почему все регуляторы сравнивают на одной трассе и одной скорости?

- ○ Иначе непонятно, что повлияло на результат: регулятор или условия
- ○ Так заезды проходят быстрее
- ○ Регуляторы работают только на одной скорости

2. Что делает релейный регулятор из урока 3.1 с колесом, под датчиком которого линия?

- ○ Останавливает его
- ○ Немного замедляет, тем сильнее, чем дальше линия
- ○ Крутит назад

3. Чем П-регулятор отличается от релейного?

- ○ Поправка растёт вместе с ошибкой, а не переключается «всё или ничего»
- ○ П-регулятор не использует датчики линии
- ○ П-регулятор всегда едет быстрее

4. Что показывает «средняя ошибка» в итогах заезда?

- ○ Насколько в среднем линия уходила от середины за весь заезд
- ○ Ошибку в последний миг заезда
- ○ Сколько раз робот сошёл с линии

5. На что реагирует Д-часть регулятора?

- ○ На то, как быстро меняется ошибка
- ○ На ошибку, накопленную за весь заезд
- ○ На заряд батареи

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.