Материалы ученика

Робот-курьер

Собираем колёсного робота и учим его ездить по линии, находить предметы и доставлять их.

Уровень итоговой задачи: **Начальный**

Комплект: Arduino Uno + шилд моторов L298P · версия курса 0.2.0

Содержание

1.1	Собираем робота и оживляем плату
2.1	Моторы: едем и останавливаемся
2.2	Повороты и развороты
3.1	Датчики линии и первое следование
3.2	П-регулятор: плавно и быстро
4.1	Находим и считаем перекрёстки
4.2	Повороты на перекрёстке и маршрут
5.1	Ультразвук: видим груз
5.2	Захват: берём и ставим
6.1	Робот-курьер: план и программа
6.2	Робот-курьер: зачётные заезды
Ф.1	Энкодеры: считаем обороты и точное расстояние
Ф.2	Регуляторы: релейный, П, ПД и ПИД на одной трассе
	Бортовой журнал

Тема 1 · Урок 1.1

Собираем робота и оживляем плату

Собери робота и проверь каждый установленный узел диагностическим скетчем.

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

РЕЗУЛЬТАТ УРОКА

Собранный робот: оба мотора проверены в обе стороны и останавливаются, оба датчика линии реагируют на белое и чёрное. Результаты проверки записаны в журнал.

Ты научишься

- Собрать ходовую часть робота по инструкции и проверить каждый узел.
- Различать перед, зад, левую и правую стороны робота.
- Загрузить программу в Arduino и объяснить, что делают setup() и loop().
- Проверить оба мотора, их остановку и оба датчика линии; отличать команду программы от наблюдаемого результата.

Нужно знать

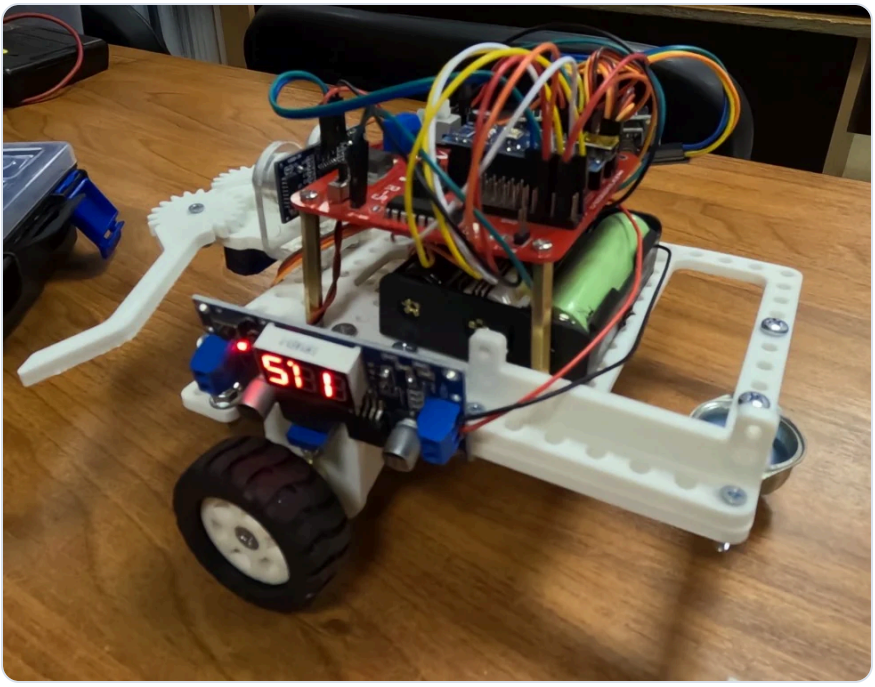
Ничего — это первый урок.

Понадобится

- Основа 130 × 150
- Кронштейн мотора
- Кронштейн мотора (зеркальный)
- Мотор-редуктор × 2
- Колесо × 2
- Шаровая опора
- Держатель Arduino
- Arduino Uno R3
- Шилд моторов L298P (Амперка)
- Планка датчиков
- Датчик линии (аналоговый, Trema) × 2
- Крепление выключателя
- Батарейный отсек моторов с выключателем
- USB-кабель для Arduino
- Крепёж: винты, гайки, шайбы, стойки
- Отвёртка

Зачем этот урок

Весь курс ты будешь работать с одним роботом — своим. Сегодня соберёшь его ходовую часть, установишь «мозг» робота — плату Arduino — и напишешь первую программу. К концу урока ты проверишь каждый установленный узел и запишешь, что он действительно работает.



Так робот будет выглядеть в конце курса. Сегодня собираем основу: платформу, моторы, колёса, плату и датчики линии.

⚠ **БЕЗОПАСНОСТЬ**

Пока идёт сборка, батарейный отсек отключён, а USB-кабель не подключён к компьютеру. Батарейный отсек к шилду заранее подключает взрослый. Моторы ты включаешь и выключаешь только выключателем.

ЗАНЯТИЕ 1 · 45 МИН

1 Ходовая часть

1 Распределите роли

3 мин

Что должен делать робот. Сначала роботу нужно правильно выполнять отдельные команды: вращать каждое колесо, останавливаться и сообщать показания датчиков. До самостоятельной поездки проверим каждый узел по очереди, чтобы искать причину ошибки в одном месте.

Что используем и как это работает. Arduino Uno выполняет программу. Драйвер L298P по её сигналам управляет питанием моторов: вывод DIR задаёт направление, PWM — долю времени подачи питания. Мотор превращает электрическую энергию во вращение, а колесо — в движение. Два датчика отражения снизу помогают различать светлую поверхность и чёрную линию.

Как связать измерение с действием. Команда и результат — разные вещи. Надпись в мониторе означает, что программа дошла до печати; вращение проверяем глазами. На подставке проверим оба направления каждого колеса и остановку, затем сравним показания датчиков над белой и чёрной карточками. Так узнаем, как именно собран наш робот.

Вы работаете вдвоём. Один — **Пилот**: держит детали и отвёртку. Второй — **Штурман**: читает шаг вслух, подаёт детали и проверяет результат по картинке.

1. Решите, кто начинает Пилотом.
2. После каждой детали, которая закреплена винтами, меняйтесь ролями.

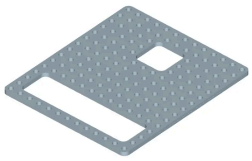
✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Каждый знает свою роль. Штурман держит инструкцию, Пилот — инструмент.

2 Разложи детали

7 мин

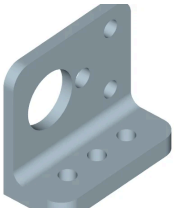
Прежде чем собирать, найди каждую деталь. Так ты не перепутаешь похожие детали и сразу увидишь, если чего-то не хватает.



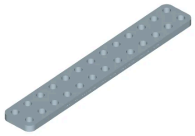
Основа 130 × 150
1 шт.



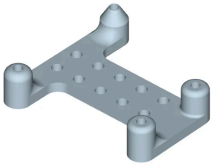
Кронштейн мотора
1 шт.



Кронштейн мотора (зеркальный)
1 шт.



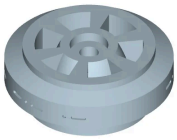
Планка датчиков
1 шт.



Держатель Arduino
1 шт.



Крепление выключателя
1 шт.



Колесо
2 шт.

1. Разложи печатные детали в ряд, как на карточках.
2. Найди два кронштейна моторов. Они похожи, но **зеркальные**: положи их рядом и посмотри — полочки смотрят в разные стороны.
3. Рядом положи покупные детали: Arduino, шилд моторов, два мотора, шаровую опору, два датчика линии, батарейный отсек и пакеты с крепежом.

ДОЛЖНО ПОЛУЧИТЬСЯ

Все детали лежат перед тобой. Ты можешь показать оба зеркальных кронштейна.

Подсказка: Не хватает детали

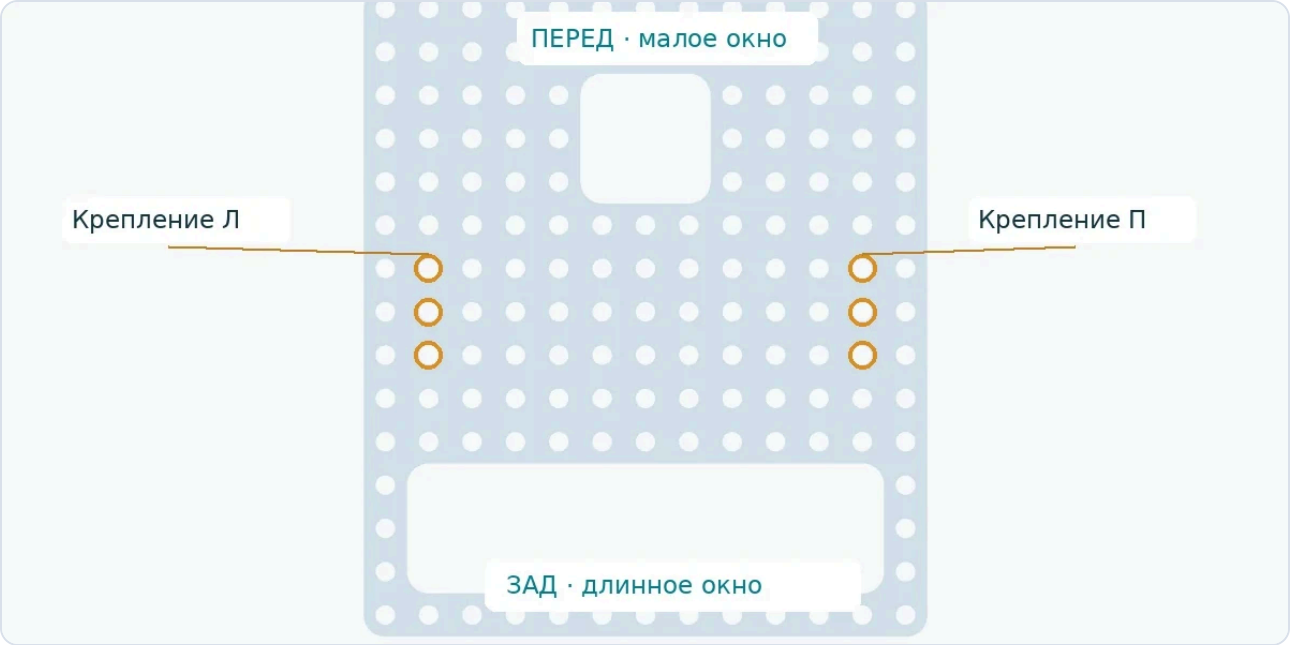
Не заменяй её похожей. Сверь пакет с карточками ещё раз; если детали нет — сообщи взрослому.

3 Потренируйся на одном винте

4 мин

Почти все детали робота держатся на винтах и гайках. Соберём одно соединение на столе.

1. Возьми винт, две шайбы и гайку из пакета «Тренировка».
2. Надень шайбу на винт, проведи винт через две детали, надень вторую шайбу и накрути гайку пальцами.
3. Придержи гайку пальцем и затяни винт отвёрткой. Остановись, как только детали перестали шевелиться.



Сетка отверстий основы. Через эти отверстия будем крепить все узлы.

ДОЛЖНО ПОЛУЧИТЬСЯ

Порядок соединения: винт → шайба → деталь → деталь → шайба → гайка. Детали не болтаются.

Как это устроено: Почему нельзя затягивать изо всех сил

Детали напечатаны из пластика. Если перетянуть винт, пластик продавится или треснет. Признак перетяжки — деталь прогибается или белеет вокруг винта.

4 Договоритесь, где перед

3 мин

Чтобы «левое колесо» всегда значило одно и то же, отметим перед робота.

1. Положи основу плоско: маленькое квадратное окно — дальше от тебя, длинное прямоугольное — ближе.
2. Дальняя сторона — **перед**. Наклей на неё бумажную стрелку «вперёд».
3. Лево и право определяй как водитель, который сидит в роботе и смотрит по стрелке.

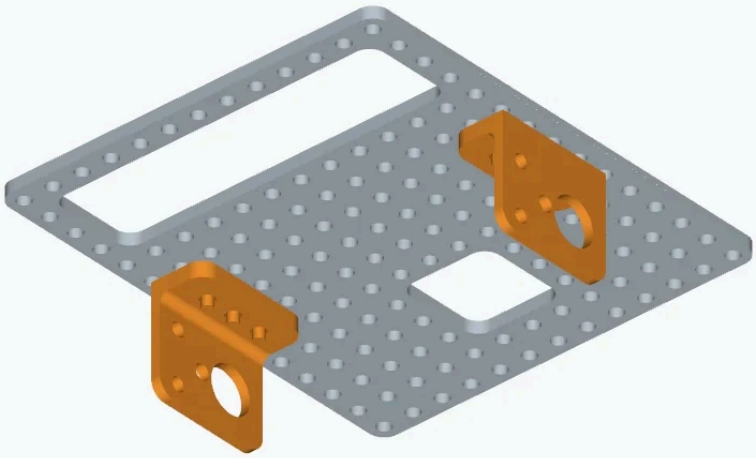
ДОЛЖНО ПОЛУЧИТЬСЯ

На основе есть стрелка «вперёд». Вы оба показываете левую сторону робота одинаково.

5 Прикрепи кронштейны моторов

8 мин

Кронштейны держат моторы снизу основы.



Вид снизу. Новые детали выделены оранжевым.

1. Переверни основу так, чтобы стрелка осталась видна сбоку.
2. Приложи кронштейны к нижней стороне, как на рисунке. Большие круглые отверстия смотрят **наружу** — туда будут выходить оси колёс.
3. Вставь винты сверху, снизу надень шайбы и гайки. Сначала наживи все винты пальцами, потом затяни.

ДОЛЖНО ПОЛУЧИТЬСЯ

Два кронштейна стоят друг напротив друга, большие отверстия — снаружи. Кронштейны не качаются.

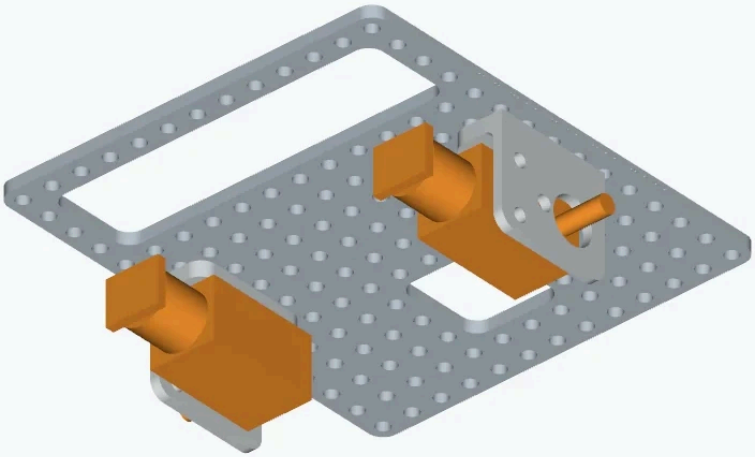
ПРОВЕРЬ СЕБЯ

- [ ] Оба кронштейна смотрят отверстиями наружу.
- [ ] Ни один винт не болтается.

6 Установи моторы

8 мин

Ось мотора — вращающийся стержень, на который надевается колесо. Она должна выйти наружу через отверстие кронштейна.



Корпус мотора внутри, ось — снаружи.

1. Приложи первый мотор к кронштейну изнутри. Ось проходит через большое отверстие наружу.
2. Закрепи мотор винтами из пакета «Моторы». Держи мотор рукой, пока затягиваешь.
3. Так же установи второй мотор.
4. Наклей на провода метки: **Л** — левый мотор, **П** — правый. Сверься со стрелкой «вперёд».

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Оба мотора держатся крепко, оси смотрят влево и вправо, провода подписаны.

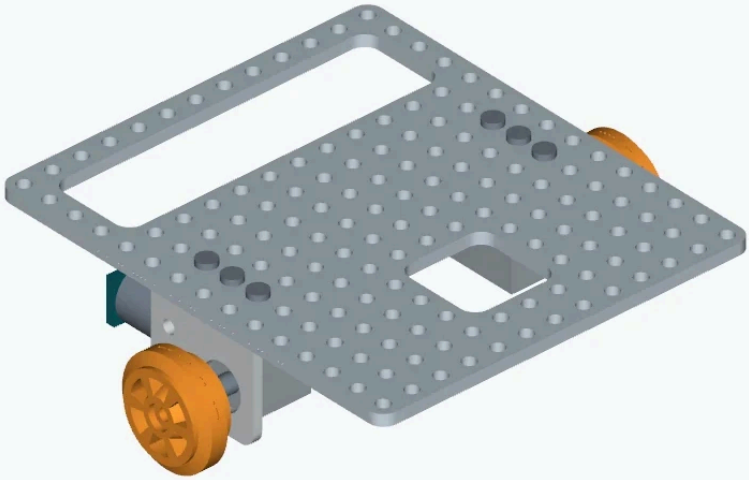
Подсказка: Мотор не встаёт в кронштейн

Не дави и не подпиливай. Проверь, тот ли это кронштейн: может быть, ты взял зеркальный для другой стороны.

7 Надень колёса

5 мин

1. Совмести отверстие колеса с осью мотора. Форма отверстия должна совпасть с формой оси.
2. Надень колесо, придерживая мотор за корпус. Не бей по колесу.
3. Посмотри сбоку: колесо не должно касаться кронштейна или основы.



Между колесом и кронштейном виден зазор.

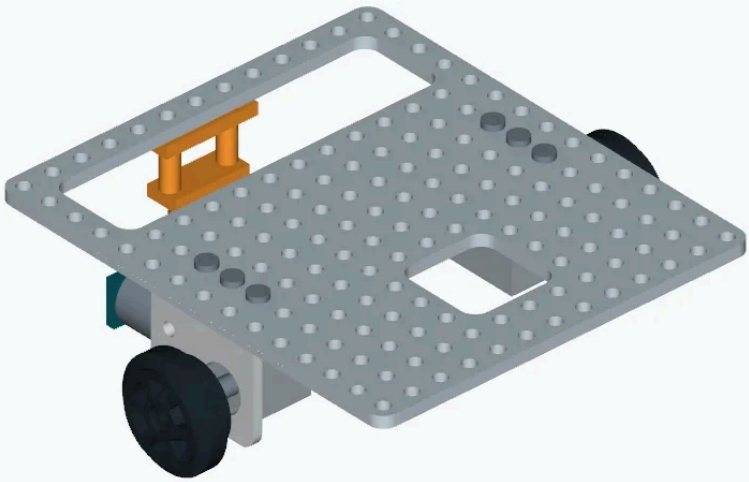
ДОЛЖНО ПОЛУЧИТЬСЯ

Два колеса стоят на осях и ничего не задевают.

8 Добавь заднюю опору и проверь робота

7 мин

Колёс два, поэтому роботу нужна третья точка опоры сзади — шаровая опора.



Опора крепится снизу у задней стороны основы на стойках.

1. Закрепи шаровую опору снизу у задней стороны основы. Стойки из пакета «Опора» подобраны по высоте.
2. Поставь робота на стол. Он должен стоять ровно на двух колёсах и опоре.
3. Аккуратно покачай робота. Ничего не должно болтаться.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Робот стоит устойчиво и ровно. Ходовая часть готова.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу показать перед, левый и правый мотор.
- [ ] Колёса ничего не задевают.
- [ ] Робот стоит ровно.

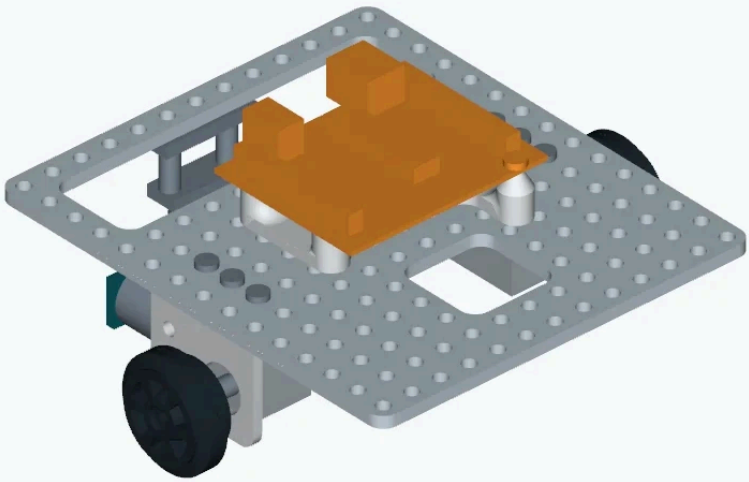
ЗАНЯТИЕ 2 · 45 МИН

2 Электроника и первая программа

9 Установи Arduino

6 мин

Arduino — «мозг» робота. Плата лежит на держателе, чтобы её контакты снизу не касались винтов.



Держатель и плата в центре основы.

1. Закрепи держатель Arduino в центре основы стойками вверх.
2. Положи плату на стойки так, чтобы разъём USB был у края и до него можно было дотянуться.
3. Закрепи плату винтами. Затягивай их по очереди, понемногу.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Плата лежит ровно и не касается винтов основы. Разъём USB свободен.

10 Подготовь компьютер

5 мин

Программы для робота пишут в **MyRobotica**. В ней есть уроки и редактор кода.

1. Подключи USB-кабель к Arduino и к компьютеру. Батарейный отсек моторов остаётся отключённым.
2. Открой MyRobotica, курс **«Робот-курьер»** и урок **1.1**. Открой учебный проект урока.
3. В правой части кнопки **Загрузить на робота** выбери своего настоящего робота. В **Подробнее о связи** проверь плату **Arduino Uno** и её порт.
4. В редакторе оставь пустые функции `setup()` и `loop()` и сохрани программу.

ДОЛЖНО ПОЛУЧИТЬСЯ

На плате горит индикатор питания ON. В MyRobotica открыт учебный проект, выбран настоящий робот с платой Arduino Uno и виден его порт.

Подсказка: Не вижу порт

Отключи и снова подключи кабель. Если порт так и не появился, возможно, кабель только заряжает и не передаёт данные. Попроси другой кабель.

11 Проверь загрузку программы

3 мин

Загрузи пустой скетч и прочитай сообщение об успешной загрузке.

Проверка связи с платой L11_Empty.ino

```
1 void setup() {
2 }
3
4 void loop() {
5 }
```

Это проверка платы, USB-связи и загрузки. Она ещё ничего не говорит о моторах, шилде, датчиках и качестве механической сборки. Работоспособность проверим после подключения всех узлов.

12 Надень шилд и подключи моторы

7 мин

Выводы Arduino не могут дать моторам достаточно тока. Поэтому моторы подключаются к **шилду** — плате, которая надевается сверху на Arduino и включает моторы по её командам.

> ⚠ **БЕЗОПАСНОСТЬ**
> Сначала отключи USB-кабель. Ничего не подключай и не отключай, пока плата под питанием.

1. Совмести ножки шилда с гнёздами Arduino и аккуратно надави сверху. Ни одна ножка не должна остаться снаружи или согнуться.
2. Подключи провода моторов к клеммам шилда:

Что подключаем	Куда
Левый мотор Пометь провода буквой «Л».	`Клеммы канала M2 на шилде`
Правый мотор Пометь провода буквой «П».	`Клеммы канала M1 на шилде`
Батарейный отсек (через выключатель) ВЗРОСЛЫЙ Подключает взрослый.	`Клеммы питания моторов на шилде`

3. Батарейный отсек с выключателем уже подключён взрослым. Проверь, что выключатель выключен. Если провода отсека не подключены — сам их не подключай, позови взрослого.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Шилд сидит на Arduino ровно. Провода моторов закреплены в клеммах и подписаны.

13 Установи датчики линии

6 мин

Датчики линии будут «смотреть» на пол и замечать чёрную линию. Ставим их спереди, поперёк робота.

1. Закрепи планку датчиков снизу у переднего края основы на стойках.
2. Закрепи на планке два датчика — левый и правый — чувствительной стороной **вниз**, к полу. Датчики на одинаковой высоте.
3. Подключи датчики к шилду по таблице. Буквы у контактов датчика: S — сигнал, V — питание, G — земля.

Что подключаем	Куда
Левый датчик · S (сигнал)	`A4`
Левый датчик · V (питание)	`5V`
Левый датчик · G (земля)	`GND`
Правый датчик · S (сигнал)	`A5`
Правый датчик · V (питание)	`5V`
Правый датчик · G (земля)	`GND`

4. Уложи провода так, чтобы они не касались колёс.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Два датчика смотрят вниз, их провода подключены и не свисают к колёсам.

14 Проверь все узлы робота

14 мин

Теперь проверим, что каждая часть робота работает: оба колеса и оба датчика. Программа для проверки уже открыта в редакторе справа.

> ⚠ **БЕЗОПАСНОСТЬ**
> Поставь робота на подставку так, чтобы колёса висели в воздухе и не касались стола. Пока выключатель моторов выключен, колёса крутиться не могут.

Программу писать не нужно — она готова. Ты будешь отправлять ей буквы и цифры, а она будет крутить одно колесо или показывать числа датчиков.

Готовая диагностика L11_Diagnostics.ino

```ino
// Программа проверки для робота курса (Arduino Uno + шилд L298P).
// Робот стоит на подставке: колёса висят в воздухе.
// Колесо крутится со скоростью 120 из 255 около 0,3 секунды.
const int LEFT_DIR = 7;
const int LEFT_PWM = 6;
const int RIGHT_DIR = 4;
const int RIGHT_PWM = 5;
const int LEFT_SENSOR = A4;
const int RIGHT_SENSOR = A5;
const int FORWARD = HIGH;
const int BACKWARD = LOW;
const int TEST_PWM = 120;
const unsigned long PULSE_MS = 300;
bool running = false;
unsigned long startedAt = 0;

void reportSensors() {
  Serial.print("LEFT=");
  Serial.print(analogRead(LEFT_SENSOR));
  Serial.print(" RIGHT=");
  Serial.println(analogRead(RIGHT_SENSOR));
}

void beginPulse(int dirPin, int pwmPin, int direction) {
  digitalWrite(dirPin, direction);
  analogWrite(pwmPin, TEST_PWM);
}

// ===== Готовая оболочка — не меняй =====
void stopMotors() {
  analogWrite(LEFT_PWM, 0);
  analogWrite(RIGHT_PWM, 0);
  running = false;
}

void setup() {
  pinMode(LEFT_PWM, OUTPUT);
  pinMode(RIGHT_PWM, OUTPUT);
  stopMotors();
```

```
  pinMode(LEFT_DIR, OUTPUT);
  pinMode(RIGHT_DIR, OUTPUT);
  pinMode(LEFT_SENSOR, INPUT);
  pinMode(RIGHT_SENSOR, INPUT);
  Serial.begin(9600);
  Serial.println("READY: 1=L forward, 2=L backward, 3=R forward, 4=R b
}

void loop() {
  if (running && millis() - startedAt >= PULSE_MS) {
    stopMotors();
    Serial.println("PWM=0; verify physical stop");
  }
  if (!Serial.available()) return;
  char command = Serial.read();
  if (command == 's') {
    stopMotors();
    Serial.println("PWM=0; verify physical stop");
    return;
  }
  // Пока идёт импульс, новые команды не продлевают и не перезапускают
  if (running) return;
  if (command == 'd') {
    reportSensors();
    return;
  }
  if (command < '1' || command > '4') return;
  // Удаляем накопленные команды: один ввод не должен запускать очеред
  while (Serial.available()) Serial.read();
  stopMotors();
  startedAt = millis();
  running = true;
  if (command == '1') beginPulse(LEFT_DIR, LEFT_PWM, FORWARD);
  if (command == '2') beginPulse(LEFT_DIR, LEFT_PWM, BACKWARD);
  if (command == '3') beginPulse(RIGHT_DIR, RIGHT_PWM, FORWARD);
  if (command == '4') beginPulse(RIGHT_DIR, RIGHT_PWM, BACKWARD);
  Serial.println("Observe wheel; no automatic hardware PASS");
}
```

Как это устроено: Что делает программа

- `reportSensors` печатает два числа: что видит левый и правый датчик.
- `beginPulse` включает одно колесо.
- Через 0,3 секунды колесо само останавливается: это делает `stopMotors`.

Как проверить робота

1. **Проверь провода.** Выключатель моторов выключен. Сверь провода с таблицами подключения: каждый на своём месте и не касается колёс. Колёса крутятся рукой свободно.
2. **Загрузи программу.** Нажми **Загрузить на робота**. Внизу открой **Serial Monitor** со скоростью 9600. Появится строка `READY` — плата отвечает. Нажми кнопку Reset на плате: колёса стоят.
3. **Проверь датчики.** Отправь `d` и прочитай два числа. Подложи под левый датчик белую бумагу и отправь `d`, потом чёрную — и снова `d`. Сделай так же с правым датчиком. На белом и чёрном числа должны заметно отличаться.
4. **Проверь колёса.** Включи выключатель моторов. Отправь `1` и смотри на колёса: коротко крутится только левое колесо вперёд и само останавливается. Потом по очереди отправь `2`, `3` и `4`.
5. **Проверь остановку.** Отправь `1` и сразу `s` — колесо останавливается. Выключи выключатель моторов и отправь `1` ещё раз: колёса не крутятся.
6. **Запиши результат** в журнал для каждого узла из таблицы: команду, ответ в Serial Monitor и фактическое вращение или показание датчика. Если узел ещё не проверен, так и запиши: «не проверен». Если что-то не так — выключи выключатель моторов, отключи USB, открой «Если не получилось», исправь и проверь этот узел ещё раз.

Что проверяем	Что сделать	Хорошо, если
Плата	Загрузить программу, нажать Reset	Есть `READY`, колёса стоят
Левое колесо	Отправить `1`, потом `2`	Крутится только левое: вперёд, потом назад
Правое колесо	Отправить `3`, потом `4`	Крутится только правое: вперёд, потом назад
Левый датчик	`d` на белом и на чёрном	Числа заметно разные
Правый датчик	`d` на белом и на чёрном	Числа заметно разные
Остановка	`s` и выключатель моторов	Колесо останавливается

🔧 ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать (выключатель моторов выключен)
Крутится не то колесо	Поменяй местами провода моторов в клеммах M1 и M2
Колесо крутится назад вместо вперёд	Поменяй местами два провода этого мотора в клемме
Колесо не крутится	Проверь, что выключатель включён и провода плотно зажаты в клеммах
Число датчика не меняется	Проверь провода S, V, G и высоту датчика над полом
Нет строки `READY`	Проверь скорость 9600 и выбранный порт
Колесо крутится само после Reset	Проверь, что загружена именно эта программа
Ничего не помогает	Позови взрослого и покажи, на каком пункте остановился

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты проверил каждое колесо в обе стороны, оба датчика и остановку. Результаты записаны в таблицу.

15 Запиши результат в журнал

4 мин

1. Запиши в бортовой журнал дату, имя робота и что получилось сегодня.
2. Отметь, какие узлы прошли проверку сразу, а какие ты исправлял.
3. Сохрани программу проверки — она пригодится, если робот начнёт вести себя странно.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот проверен, результаты записаны в журнал, программа проверки сохранена.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, чем `setup()` отличается от `loop()`.
- [ ] Я проверил оба колеса в обе стороны, остановку и оба датчика.
- [ ] Я записал результаты и прошёл проверку знаний.
- [ ] Я знаю, где у моего робота левый мотор и левый датчик.

Проверь себя

Основное

1. Что подтверждает успешная загрузка скетча?

- ○ Связь с платой и загрузку программы
- ○ Исправность обоих моторов
- ○ Правильную сборку всех узлов

2. Как проверить левый мотор?

- ○ Только включить светодиод
- ○ Отдельно включить его в обе стороны и наблюдать остановку
- ○ Подать команду обоим моторам и не смотреть на колёса

3. Что проверяют у каждого датчика линии?

- ○ Повторяемое различие показаний на белом и чёрном
- ○ Только свечение индикатора
- ○ Только успешную компиляцию

4. Зачем нужен шилд моторов?

- ○ Для ускорения Arduino
- ○ Для питания и управления моторами по командам платы
- ○ Для определения цвета пола

5. Что отключить перед изменением проводки? Выбери все ответы.

- ☐ USB
- ☐ Питание моторов
- ☐ Только монитор порта

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

Ты собрал ходовую часть робота, установил плату и датчики и написал первую программу. Ты подтвердил работу установленных узлов наблюдениями. Проверка знаний обязательна отдельно. В следующем уроке перейдёшь от отдельных проверок на подставке к управляемому движению.

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

Arduino	Плата с микроконтроллером — маленьким компьютером, который выполняет твою программу.
шилд	Плата, которая надевается сверху на Arduino и добавляет ей возможности. Наш шилд управляет моторами.
скетч	Программа для Arduino.
функция	Именованный блок команд. setup и loop — функции.
светодиод	Маленькая лампочка на плате. Светодиод L подключён к выводу 13.
вывод	Контакт платы, которым программа может управлять или с которого может читать сигнал.
мс	Миллисекунда — тысячная доля секунды. 1000 мс = 1 секунда.

Тема 2 · Урок 2.1

Моторы: едем и останавливаемся

Первые движения робота: вперёд, назад и обязательная остановка из программы.

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

РЕЗУЛЬТАТ УРОКА

Робот проезжает вперёд, останавливается и возвращается назад по программе. Движения оформлены функциями forward(), backward() и stopMotors().

Ты научишься

- Объяснить, что задаёт контакт направления и что — контакт скорости.
- Включить и остановить моторы командами digitalWrite и analogWrite.
- Написать функции движения и вызвать их по порядку.
- Объяснить, почему движение без команды остановки не заканчивается.

Нужно знать

- 1.1 Собираем робота и оживляем плату

Понадобится

Мотор-редуктор × 2 | Шилд моторов L298P (Амперка) | Батарейный отсек моторов с выключателем | USB-кабель для Arduino

Зачем этот урок

Робот собран, но пока стоит. Сегодня он поедет. Ты узнаешь, как плата командует моторами, и научишь робота ехать вперёд, назад и — самое важное — останавливаться по программе.

РАЗМИНКА · ВСПОМНИ

1. Сколько раз выполняется `setup()`, а сколько — `loop()`?
2. Что делает `digitalWrite(вывод, HIGH)`?
3. Где на твоём роботе левый мотор?

БЕЗОПАСНОСТЬ

Моторы проверяем на подставке: колёса не касаются стола. Программу загружаем при выключенном питании моторов. Чтобы остановить опыт, выключи питание моторов выключателем.

ЗАНЯТИЕ 1 · 45 МИН

1 Один мотор, потом два

1 Подготовь робота

5 мин

Что должен делать робот. Роботу нужно начать движение, проехать участок и остановиться. Для этого научимся отдельно задавать направление и мощность каждого из двух моторов, а затем объединим команды в понятные действия.

Что используем и как это работает. Моторы подключены через драйвер L298P: вывод Arduino подаёт управляющий сигнал, а ток к мотору поступает через драйвер. DIR выбирает направление. ШИМ быстро чередует включение и выключение питания; большее значение analogWrite означает большую долю включённого времени. Это команда мощности, а не измеренная скорость в сантиметрах в секунду.

Как связать измерение с действием. Если оба колеса крутятся вперёд, робот едет; если одно слабее, он отклоняется. delay только задерживает следующую команду: мотор продолжает выполнять предыдущую. Поэтому в программе нужна отдельная остановка. Сначала наблюдаем колёса на подставке, затем проверяем направление, расстояние и остановку на поле.

1. Поставь робота на подставку так, чтобы оба колеса висели в воздухе.
2. Найди выключатель питания моторов и выключи его.
3. Подключи USB-кабель.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Колёса висят, питание моторов выключено, плата подключена к компьютеру.

2 Включи левый мотор

15 мин

Каждым мотором управляют два контакта. Контакт **направления** решает, куда крутится колесо. Контакт **скорости** решает, насколько сильно: число от 0 (стоит) до 255 (полная мощность). Левый мотор подключён к контактам `7` (направление) и `6` (скорость).

Левый мотор — L21_Left_Motor.ino

```
const int LEFT_DIR_PIN = 7;
const int LEFT_SPEED_PIN = 6;
const int FORWARD = HIGH;

void setup() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);

  digitalWrite(LEFT_DIR_PIN, FORWARD);
  analogWrite(LEFT_SPEED_PIN, 100);
  delay(2000);
  analogWrite(LEFT_SPEED_PIN, 0);
}

void loop() {
}
```

ПРЕДСКАЖИ

Что сделает колесо после загрузки? Сколько секунд оно будет крутиться? Остановится ли оно само? Скажи напарнику до запуска.

1. Набери программу и загрузи её при выключенном питании моторов.
2. Включи питание моторов и нажми кнопку Reset на плате: программа начнётся заново.
3. Сравни, что произошло, со своим прогнозом.

ДОЛЖНО ПОЛУЧИТЬСЯ

Левое колесо крутится вперёд 2 секунды и останавливается.

Как это устроено: Что значит каждая строка

- `const int LEFT_DIR_PIN = …;` — имя для номера контакта. Дальше в программе пишем имя, а не число: так понятнее и меньше ошибок.
- `digitalWrite(LEFT_DIR_PIN, FORWARD);` — выбрать направление «вперёд».
- `analogWrite(LEFT_SPEED_PIN, 100);` — включить мотор с силой 100 из 255.
- `analogWrite(LEFT_SPEED_PIN, 0);` — остановить мотор.
- Всё написано в `setup()`, поэтому выполняется один раз. `loop()` пустой: после остановки робот просто стоит.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Колесо не крутится	Включено ли питание моторов? Нажми Reset после включения.
Колесо крутится назад	Запиши это в журнал — на следующем шаге исправим направлением.
Крутится правое колесо	Провода моторов перепутаны на шилде. Выключи питание и поменяй местами провода в клеммах M1 и M2.

3 Включи оба мотора

15 мин

1. Создай скетч `L21_Both_Motors` и набери программу.

Оба мотора — L21_Both_Motors.ino

```
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;

int speed = 100;

void setup() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);

  digitalWrite(LEFT_DIR_PIN, FORWARD);
  digitalWrite(RIGHT_DIR_PIN, FORWARD);
  analogWrite(LEFT_SPEED_PIN, speed);
  analogWrite(RIGHT_SPEED_PIN, speed);
  delay(1000);

  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void loop() {
}
```

2. Загрузи и запусти на подставке.
3. Поменяй `speed` на 60, потом на 200. Как меняется вращение?

ДОЛЖНО ПОЛУЧИТЬСЯ

Оба колеса крутятся вперёд одну секунду и останавливаются. С большим `speed` колёса крутятся быстрее.

Как это устроено: Почему 0–255, а не километры в час

Число в `analogWrite` — это не скорость, а доля времени, когда мотор включён. При 255 он включён всё время, при 100 — примерно 40 % времени. Настоящая скорость зависит ещё от батареи и пола. Поэтому скорость подбирают опытом.

4 Найди ошибку остановки

10 мин

НАЙДИ ОШИБКУ · РОБОТ НЕ ОСТАНАВЛИВАЕТСЯ

Соседи перенесли команды движения в `loop()` и убрали остановку:

```
void loop() {
  analogWrite(LEFT_SPEED_PIN, speed);
  analogWrite(RIGHT_SPEED_PIN, speed);
  delay(1000);
}
```

Они ждали, что робот проедет одну секунду. Что будет на самом деле и почему? Как исправить?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить: `delay` только ждёт, моторы сами не выключаются, а `loop()` повторяет команды снова и снова.

2

ЗАНЯТИЕ 2 · 45 МИН

Функции движения

5 Оформи движения функциями

15 мин

Писать каждый раз по четыре строки неудобно. Соберём их в **функции**: `forward()`, `backward()`, `stopMotors()`. Внутри них — знакомые команды.

1. Создай скетч `L21_Moves` и набери программу.

Функции движения L21_Moves.ino

```
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

int speed = 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);
}

void forward(int ms) {
  drive(FORWARD, speed, FORWARD, speed);
  delay(ms);
}

void backward(int ms) {
  drive(BACKWARD, speed, BACKWARD, speed);
  delay(ms);
}

void stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void setup() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  delay(2000); // время убрать руки от робота

```

```
  forward(1000);
  stopMotors();
  delay(500);
  backward(1000);
  stopMotors();
}

void loop() {
}
```

2. Найди, где функции **описаны** и где они **вызываются**. Покажи напарнику.

Как это устроено: Как устроена функция с параметром

`void forward(int ms)` — функция с именем `forward` и параметром `ms`. Когда пишем `forward(1000)`, внутри функции `ms` равно 1000. Так одна функция умеет ехать любое время.

`drive(…)` — вспомогательная функция: она сразу задаёт направление и скорость обоих колёс. Ей пользуются остальные функции.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа проверяется без ошибок. Ты можешь показать описание и вызов `forward`.

6 Первый заезд по полу

15 мин

> ⚠ **БЕЗОПАСНОСТЬ**
> Освободи на полу или столе место не меньше метра вперёд и назад. Держи руку рядом с выключателем.

1. Загрузи программу на подставке и проверь: вперёд, пауза, назад, стоп.
2. Поставь робота на пол, включи питание моторов и нажми Reset.
3. Отметь на полу мелом или скотчем точку старта и точку, где робот остановился.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Робот проезжает вперёд, останавливается и возвращается примерно на место старта.

🔧 ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Робот едет не прямо	Моторы не одинаковые — это нормально. Запиши, куда уводит.
Робот вернулся не на старт	Вперёд и назад колёса крутятся чуть по-разному. Запиши разницу.
Робот поехал сразу после загрузки	Так и есть: программа стартует сама. Поэтому в `setup()` есть пауза 2 секунды.

7 Измени программу

10 мин

1. Сделай так, чтобы робот проехал вперёд в два раза дальше, а назад — как раньше.
2. Сделай «челнок»: вперёд — назад три раза подряд с паузами.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, что делает контакт направления и что — контакт скорости.
- [ ] Я могу написать вызов функции `forward` на нужное время.
- [ ] Я знаю, почему без `stopMotors()` робот не остановится.

ОСНОВНОЕ · ВПЕРЁД, СТОП, НАЗАД

Робот проезжает вперёд 1 секунду, останавливается и возвращается назад.

Проверь себя

● Основное

1. Что задаёт число в команде analogWrite(LEFT_SPEED_PIN, 100)?

- ○ Скорость в сантиметрах в секунду
- ○ Насколько сильно включён мотор: от 0 до 255
- ○ Сколько секунд крутится мотор

2. Как сделать, чтобы левое колесо крутилось назад?

- ○ Записать в analogWrite отрицательное число
- ○ Поменять значение на контакте направления
- ○ Убрать delay

3. Что сделает робот?

```
1 forward(1000);
2 delay(500);
```

- ○ Проедет 1 секунду и постоит полсекунды
- ○ Проедет полторы секунды и продолжит ехать: команды остановки нет
- ○ Не поедет

4. Зачем описывать движения функциями forward(), backward(), stopMotors()?

- ○ Без функций моторы не работают
- ○ Программа короче, её легче читать и исправлять в одном месте
- ○ Робот едет быстрее

5. Почему в setup() перед движением стоит delay(2000)?

- ○ Чтобы успеть убрать руки и поставить робота
- ○ Чтобы моторы прогрелись
- ○ Без паузы программа не загрузится

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 ездит по твоей программе. Главное: `analogWrite` включает мотор, и он крутится, пока другая команда его не остановит. Функции делают программу короче и понятнее. В следующем уроке робот научится поворачивать.

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

ШИМ	Частое включение и выключение питания мотора. Число от 0 до 255 задаёт, какую долю времени мотор включён.
направление	Состояние контакта направления: одно значение крутит колесо вперёд, другое — назад.
константа	Число с именем, которое не меняется во время работы программы, например номер контакта.
функция	Именованный блок команд. Его можно вызвать много раз, написав имя и скобки.
параметр	Число, которое передают функции в скобках, например время движения.

Тема 2 · Урок 2.2

Повороты и развороты

Робот поворачивает на месте на 90° и разворачивается на 180°. Время поворота подбираешь сам.

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

РЕЗУЛЬТАТ УРОКА

Робот поворачивает на 90° налево и направо и разворачивается на 180°. Время поворотов подобрано и записано в бортовой журнал, повороты оформлены функциями.

Ты научишься

- Объяснить, почему робот крутится на месте, когда колёса вращаются в разные стороны.
- Подобрать время поворота опытом и проверить его повторением.
- Написать функции turnLeft(), turnRight() и turnAround().
- Предсказать, куда приедет робот по последовательности движений.

Нужно знать

- 2.1 Моторы: едем и останавливаемся

Понадобится

Мотор-редуктор × 2 | Шилд моторов L298P (Амперка) | Батарейный отсек моторов с выключателем

Зачем этот урок

Робот умеет ездить вперёд и назад. Но на поле итоговой задачи нужно сворачивать к грузам и разворачиваться. Сегодня робот научится поворачивать на месте, а ты — подбирать числа опытом, как настоящий инженер.

РАЗМИНКА · ВСПОМНИ

1. Какая команда задаёт направление вращения колеса?
2. Что будет, если включить моторы и не написать `stopMotors()`?
3. Чем описание функции отличается от её вызова?

1

ЗАНЯТИЕ 1 · 45 МИН

Поворот на месте

1 Колёса в разные стороны

10 мин

Что должен делать робот. Курьеру нужно менять направление: повернуть к ответвлению и развернуться для обратной поездки. Поворот получим теми же двумя моторами, без отдельного рулевого механизма.

Что используем и как это работает. Когда колёса проходят разные расстояния, корпус поворачивается. Если одно колесо вращается вперёд, а другое назад, робот разворачивается примерно вокруг середины между колёсами. Функция поворота объединяет направления колёс, время вращения и остановку.

Как связать измерение с действием. В этом уроке угол задаём временем работы моторов. Это приближение: заряд батареи, покрытие и проскальзывание меняют результат. Измерим получившийся угол, подберём время и повторим опыт. Имя turnLeft само по себе не гарантирует 90 градусов — это должна подтвердить проверка.

ПРЕДСКАЖИ

Левое колесо крутится назад, правое — вперёд с той же скоростью. Что сделает робот? Покажи руками.

1. Набери скетч `L22_Turn_Test` и загрузи его при выключенном питании моторов.

Проба поворота L22_Turn_Test.ino

```
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

int turnSpeed = 100;
int turnMs = 400; // подбираем: время поворота на 90°

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void setupMotors() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
}

void setup() {
  setupMotors();
  delay(2000);

  // Левое колесо назад, правое вперёд: робот крутится на месте налево
  drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);
  delay(turnMs);
  stopMotors();
}
```

```
40
41 void loop() {
42 }
```

2. Поставь робота на пол, на лист бумаги. Прочерти карандашом линию вдоль его носа.
3. Включи питание моторов и нажми Reset.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот крутится на месте налево и останавливается. Нос повернулся, но, скорее всего, не ровно на 90°.

Как это устроено: Почему робот крутится на месте

Колёса толкают робота в разные стороны с одинаковой силой. Вперёд и назад он не едет — он вращается вокруг точки посередине между колёсами. Чем дольше включены моторы, тем больше угол поворота.

2 Подбери время поворота на 90°

20 мин

На поворот 90° нужно какое-то время в миллисекундах. Узнать его можно только опытом.

1. Начерти на листе две линии под прямым углом (угольник или край тетради).
2. Поставь робота носом вдоль первой линии. Запусти программу.
3. Если робот не довернул — увеличь `turnMs`, если перевернул — уменьши. Меняй понемногу: на 20–50 мс.
4. Когда поворот получился, повтори его три раза. Запиши время в журнал.

Попытка	`turnMs`	Что получилось
1	400	
2		
3		

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот три раза подряд поворачивает примерно на 90°. Время записано в журнале.

Подсказка: Каждый раз поворачивает по-разному

Проверь, что батарея заряжена: от разряженной моторы крутятся медленнее. Пол должен быть ровным и не скользким. Небольшая разница между попытками — нормально.

3 Разворот на 180°

15 мин

ПРЕДСКАЖИ

Если на 90° нужно, например, 400 мс, сколько нужно на 180°? Запиши свой прогноз.

1. Поставь в `turnMs` своё время, умноженное на 2. Проверь разворот.
2. Подправь число, если нужно. Запиши время разворота в журнал.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот разворачивается и смотрит в обратную сторону. Время разворота записано.

Как это устроено: Почему ровно в два раза не всегда получается

В начале поворота моторам нужно время, чтобы разогнаться, а в конце робот немного проскальзывает по инерции. Поэтому время на 180° может быть чуть меньше двух поворотов на 90°. Опыт важнее расчёта.

2

ЗАНЯТИЕ 2 · 45 МИН

Функции поворотов

4 Функции поворотов

12 мин

1. Создай скетч `L22_Turns`. Впиши свои времена из журнала в `turn90Ms` и `turn180Ms`.

Функции поворотов L22_Turns.ino

```
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

int speed = 100;
int turnSpeed = 100;
int turn90Ms = 400;  // впиши своё время из журнала
int turn180Ms = 800; // впиши своё время из журнала

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void setupMotors() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
}

void forward(int ms) {
  drive(FORWARD, speed, FORWARD, speed);
  delay(ms);
  stopMotors();
}

void turnLeft() {
```

```
  drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);
  delay(turn90Ms);
  stopMotors();
}

void turnRight() {
  drive(FORWARD, turnSpeed, BACKWARD, turnSpeed);
  delay(turn90Ms);
  stopMotors();
}

void turnAround() {
  drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);
  delay(turn180Ms);
  stopMotors();
}

void setup() {
  setupMotors();
  delay(2000);

  for (int i = 0; i < 4; i++) {
    forward(1000);
    delay(300);
    turnLeft();
    delay(300);
  }
}

void loop() {
}
```

2. Найди разницу между `turnLeft()` и `turnRight()`. Что поменялось местами?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Программа проверяется без ошибок. Ты можешь объяснить, чем отличаются функции поворота.

5 Квадрат

18 мин

ПРЕДСКАЖИ

Программа повторяет 4 раза: «вперёд 1 секунду, налево». Где окажется робот в конце и куда будет смотреть? Нарисуй путь в журнале.

1. Отметь старт скотчем. Запусти программу на полу.
2. Сравни, где робот остановился, с прогнозом.
3. Насколько робот не доехал до старта? Запиши.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот проезжает квадрат и возвращается близко к старту, носом в ту же сторону.

НАЙДИ ОШИБКУ · КВАДРАТ ПРЕВРАТИЛСЯ В ЛИНИЮ

У соседей робот едет вперёд, крутится, но квадрат не получается: он ездит туда-обратно. В их `turnLeft()` стоит `delay(turn180Ms)`. Что не так?

6 Задания по уровням

15 мин

ОСНОВНОЕ · ПОВОРОТ И РАЗВОРОТ

Робот поворачивает налево и направо на 90° и разворачивается на 180° по твоим временам.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, почему робот крутится на месте.
- [ ] Мои времена поворотов записаны в журнале.
- [ ] Я могу предсказать путь робота по программе из движений.

Проверь себя

Основное

1. Левое колесо крутится вперёд, правое — назад с той же скоростью. Что делает робот?

- ○ Крутится на месте направо
- ○ Крутится на месте налево
- ○ Едет прямо

2. Робот не доворачивает до 90°. Что изменить?

- ○ Немного увеличить время поворота
- ○ Уменьшить время поворота
- ○ Поменять направление правого колеса

3. Почему время поворота подбирают опытом, а не берут из учебника?

- ○ Оно зависит от моторов, батареи и пола конкретного робота
- ○ В учебниках ошибки
- ○ Arduino не умеет считать время

4. Программа: 4 раза «вперёд, налево». Где окажется робот?

- ○ Около старта, носом в ту же сторону
- ○ Далеко впереди старта
- ○ Около старта, носом назад

5. Батарея разрядилась. Что, скорее всего, случится с поворотами по времени?

- ○ Робот будет недоворачивать: моторы крутятся медленнее
- ○ Ничего не изменится
- ○ Робот будет поворачивать 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 по времени, которое ты подобрал опытом. Это простой способ, но он зависит от батареи и пола. В уроке 4.2 робот научится поворачивать точнее — по датчикам: крутиться, пока не найдёт новую линию.

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

поворот на месте	Колёса крутятся в разные стороны, и робот вращается вокруг середины между колёсами.
калибровка	Подбор числа по опыту: например, сколько миллисекунд нужно на поворот 90°.
погрешность	Насколько результат отличается от нужного: например, робот повернул на 85° вместо 90°.

Тема 3 · Урок 3.1

Датчики линии и первое следование

Робот «смотрит» на пол, отличает чёрное от белого и сам держится линии.

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

РЕЗУЛЬТАТ УРОКА

Робот проезжает по чёрной линии с плавным изгибом, поворачивая по сигналам двух датчиков. В бортовом журнале записаны показания датчиков на белом и чёрном.

Ты научишься

- Прочитать показания датчика командой analogRead и вывести их в монитор порта.
- Различать строки Serial Monitor и каналы телеметрии с временем и единицами; сравнивать сохранённые показания модели и настоящего робота.
- Измерить показания на белом и чёрном и объяснить, почему у каждого датчика они свои.
- Записать правило «показание ближе к чёрному — значит, под датчиком линия».
- Составить два правила поворота и проверить их сначала на подставке, затем на дорожке.

Нужно знать

- 1.1 Собираем робота и оживляем плату
- 2.1 Моторы: едем и останавливаемся

Понадобится

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

Зачем этот урок

До сих пор робот делал только то, что ты заранее записал: ехать столько-то, повернуть так-то. Сегодня он начнёт сам принимать решения. Два датчика будут смотреть на пол, и робот будет поворачивать, когда линия уходит в сторону.

РАЗМИНКА · ВСПОМНИ

1. Какая команда задаёт скорость мотора и какие числа она принимает?
2. Что делает `delay(500)`?
3. Какое колесо надо остановить, чтобы робот повернул налево?

ПРЕДСКАЖИ

Как робот может «увидеть» чёрную линию, если у него нет камеры? Обсудите с напарником и запишите догадку в бортовой журнал. В конце урока проверите.

БЕЗОПАСНОСТЬ

Пока меняешь провода, USB-кабель отключён, питание моторов выключено. Опыты с моторами — сначала на подставке: колёса не касаются стола.

ЗАНЯТИЕ 1 · 45 МИН

1 Датчики и их показания

1 Подготовь дорожку и робота

5 мин

Что должен делать робот. Чтобы ехать по дорожке, роботу нужно замечать, когда линия оказалась под левым или правым датчиком. Для этого используем два направленных вниз датчика отражения; они смотрят на пол, а не ищут предметы впереди.

Что используем и как это работает. В датчике есть излучатель и приёмник отражённого света. Поверхности отражают свет по-разному, а приёмник превращает отражённый свет в электрический сигнал. analogRead считывает его как число. Это не готовый ответ «чёрное»: сначала измеряем белое и чёрное для каждого датчика при рабочей высоте.

Как связать измерение с действием. Программа сравнивает чтение с измеренным порогом и выбирает действие колёс. Над линией слева и справа должны получиться разные команды коррекции. Проверим четыре сочетания карточек на подставке, затем движение по линии. Одинаковые показания обоих датчиков ещё не доказывают, что робот стоит правильно: оба могут оказаться и на белом, и на чёрном.

Датчику нужна понятная картинка: светлый фон и тёмная линия.

1. Возьми светлое матовое поле с чёрной линией (или наклей чёрную матовую изоленту на белый лист). Для начала хватит прямой с одним плавным изгибом.
2. Подготовь две карточки: белую и чёрную.
3. Найди на роботе левый и правый датчики. Они смотрят вниз и стоят по обе стороны от линии, когда робот стоит на ней.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Дорожка и карточки готовы. Ты можешь показать левый и правый датчик.

2 Посмотри, что видит датчик

12 мин

Датчик светит на пол и меряет, сколько света вернулось. Плата превращает это в число.

Чтобы увидеть число, плата будет отправлять его на компьютер, а мы откроем **монитор порта**.

1. Создай скетч `L31_Read_Sensors` и набери программу. Левый датчик подключён к `A4`, правый — к `A5`.

Показания двух датчиков — L31_Read_Sensors.ino

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

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

void loop() {
  int leftValue = analogRead(LEFT_SENSOR_PIN);
  int rightValue = analogRead(RIGHT_SENSOR_PIN);

  Serial.print("L=");
  Serial.print(leftValue);
  Serial.print("  R=");
  Serial.println(rightValue);

  delay(200);
}
```

2. Загрузи программу. Открой **Инструменты → Монитор порта** и выбери скорость **9600**.
3. Поднеси под левый датчик белую карточку, затем чёрную. Держи карточку на той же высоте, на которой датчик стоит над полом.

ДОЛЖНО ПОЛУЧИТЬСЯ

В мониторе бегут строки вида `L=… R=…`. Когда меняешь карточку под датчиком, его число заметно меняется.

Как это устроено: Что значит каждая строка

- `analogRead(LEFT_SENSOR_PIN)` — прочитать показание датчика: число от 0 до 1023.
- `int leftValue = …` — сохранить число в переменную `leftValue`. `int` — целое число.
- `Serial.begin(9600)` — начать связь с компьютером. 9600 — скорость, такая же должна быть выбрана в мониторе порта.
- `Serial.print` печатает в той же строке, `Serial.println` — печатает и переходит на новую.
- Текст в кавычках печатается как есть, а имя переменной без кавычек заменяется её числом.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Вместо чисел непонятные символы	В мониторе порта выбери 9600.
Число не меняется	Проверь провод сигнала S и номер входа в программе.
Меняется не тот датчик	Провода левого и правого датчиков перепутаны. Поменяй их местами или номера в программе.

3 Проверь оба датчика

8 мин

ПРЕДСКАЖИ

Будут ли числа левого и правого датчиков одинаковыми над одним и тем же белым полем?

1. Поставь робота на белое поле. Запомни числа L и R.
2. Подведи чёрную карточку только под левый датчик, потом только под правый.
3. Сдвинь робота так, чтобы линия была точно между датчиками. Какие числа сейчас?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты знаешь, какое число у каждого датчика на белом и как оно меняется на чёрном. Когда линия между датчиками, оба показывают «белое».

4 Измерь белое и чёрное

12 мин

Датчики не одинаковые: даже над одним полем их числа разные. Поэтому каждому датчику нужны свои измерения. Такая настройка по своим измерениям называется **калибровкой**.

1. Поставь робота на белое поле. Посмотри на несколько строк и запиши типичное число левого датчика в бортовой журнал: «Левый, белое».
2. Подведи под левый датчик чёрную линию. Запиши «Левый, чёрное».
3. Повтори для правого датчика.
4. Проверь себя: повтори каждое измерение ещё раз. Числа должны быть близки к записанным.

Датчик	Белое	Чёрное
Левый		
Правый		

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале четыре числа. Белое и чёрное у каждого датчика заметно отличаются.

Подсказка: Белое и чёрное почти одинаковые

Датчик может стоять слишком высоко или линия блестит. Опусти датчики ближе к полу и проверь линию в другом месте поля, где нет бликов.

Как это устроено: Почему у одних датчиков чёрное — большое число, а у других — маленькое

Это зависит от устройства датчика. Одни выдают тем большее число, чем больше света вернулось, другие — наоборот. Поэтому мы не запоминаем «чёрное — это много», а измеряем свои числа.

2

ЗАНЯТИЕ 2 · 45 МИН

Робот едет по линии

5 Научи программу замечать линию

10 мин

Как программе решить, что под датчиком линия? Сравним новое показание с нашими двумя измерениями: **если число ближе к чёрному, чем к белому, — под датчиком линия**.

Пример: белое — 100, чёрное — 800. Пришло число 650. До чёрного 150, до белого 550. Ближе к чёрному — значит, линия.

1. Создай скетч `L31_Line_Detect`. Впиши свои четыре числа вместо нулей.

Программа замечает линию L31_Line_Detect.ino

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

// Впиши свои измерения из бортового журнала.
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;

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

void loop() {
  int leftValue = analogRead(LEFT_SENSOR_PIN);
  int rightValue = analogRead(RIGHT_SENSOR_PIN);

  bool leftOnLine = abs(leftValue - LEFT_BLACK) < abs(leftValue - LEF
  bool rightOnLine = abs(rightValue - RIGHT_BLACK) < abs(rightValue -

  Serial.print("L=");
  Serial.print(leftValue);
  Serial.print("  R=");
  Serial.print(rightValue);

  if (leftOnLine) {
    Serial.print("  слева линия");
  }
  if (rightOnLine) {
    Serial.print("  справа линия");
  }

  Serial.println();
  delay(200);
}
```

2. Загрузи программу, открой монитор порта и подводи чёрную карточку под датчики по очереди.

ДОЛЖНО ПОЛУЧИТЬСЯ

Когда под левым датчиком чёрное, в строке появляется «слева линия», под правым — «справа линия».

НАЙДИ ОШИБКУ · ПОЧЕМУ СООБЩЕНИЕ НЕ ПОЯВЛЯЕТСЯ

Команда соседей написала так:

```
bool leftOnLine = abs(leftValue - LEFT_BLACK) > abs(leftValue - LEF
```

Под левым датчиком чёрная карточка, а «слева линия» не печатается. Найди ошибку и объясни, что программа проверяет сейчас.

Как это устроено: Что такое abs и bool

- `abs(число)` убирает знак: `abs(-150)` равно 150. Так мы получаем **расстояние** между двумя числами, не думая, какое из них больше.
- `bool` — переменная, в которой хранится не число, а ответ «да» или «нет» (`true` или `false`).
- `if (leftOnLine) { … }` — если ответ «да», выполнить команды в скобках.

6 Придумай, как поворачивать

7 мин

Робот едет вперёд. Линия должна оставаться между датчиками. Что делать, если линия оказалась под левым датчиком? Значит, робот уехал вправо, и ему надо повернуть налево.

Самый простой способ повернуть налево — **остановить левое колесо**, а правое пусть едет.

1. Поставь выключенного робота на линию. Сдвинь его так, чтобы линия оказалась под левым датчиком. Покажи, какое колесо нужно остановить.
2. То же — для правого датчика.
3. Заполни таблицу в журнале:

Что видят датчики	Левое колесо	Правое колесо	Куда едет робот
Оба белое	вперёд	вперёд	прямо
Левый — линия	?	?	?
Правый — линия	?	?	?
Оба — линия	?	?	?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Таблица заполнена. Ты можешь объяснить, почему остановка одного колеса поворачивает робота.

7 Запиши правила в программу

10 мин

Соберём всё вместе: датчики, наше правило «ближе к чёрному» и моторы из урока 2.1.

1. Создай скетч `L31_First_Line`. Перенеси свои четыре измерения.

Первое движение по линии L31_First_Line.ino

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

// Впиши свои измерения из бортового журнала.
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;

int speed = 100; // начальная скорость, подбираем на подставке

void setup() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);

  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
  digitalWrite(LEFT_DIR_PIN, FORWARD);
  digitalWrite(RIGHT_DIR_PIN, FORWARD);

  // Без измерений робот не поедет: программа остановится здесь.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }

  delay(2000); // время убрать руки от робота
}

void loop() {
  int leftValue = analogRead(LEFT_SENSOR_PIN);
```

```
  int rightValue = analogRead(RIGHT_SENSOR_PIN);

  bool leftOnLine = abs(leftValue - LEFT_BLACK) < abs(leftValue - LEF
  bool rightOnLine = abs(rightValue - RIGHT_BLACK) < abs(rightValue -

  int leftSpeed = speed;
  int rightSpeed = speed;

  if (leftOnLine) {
    leftSpeed = 0;
  }
  if (rightOnLine) {
    rightSpeed = 0;
  }

  analogWrite(LEFT_SPEED_PIN, leftSpeed);
  analogWrite(RIGHT_SPEED_PIN, rightSpeed);
}
```

2. Найди в `loop()` два правила из твоей таблицы. Объясни напарнику, что делает каждое.
3. Нажми **Проверить**.

> **Как это устроено: Почему скорость задаётся заново в каждом повторе**
>
> В начале `loop()` обе скорости снова становятся равны `speed`. Если бы мы этого не делали, колесо, остановленное один раз, так бы и стояло, даже когда датчик снова увидит белое.

> **Как это устроено: Зачем строка while (true)**
>
> Если ты забыл вписать измерения, белое и чёрное равны нулю, и программа не может отличить линию. Тогда она «застревает» в пустом бесконечном цикле, и робот не поедет. Это защита от случайного старта.

8 Проверь на подставке

6 мин

> ⚠ **БЕЗОПАСНОСТЬ**
> Загружай программу при выключенном питании моторов. Колёса не касаются стола.

1. Поставь робота на подставку. Загрузи программу.
2. Включи питание моторов. Оба колеса должны крутиться вперёд.
3. Подведи чёрную карточку под левый датчик: левое колесо остановится. Убери карточку: колесо снова поедет. Проверь правый датчик.
4. Выключи питание моторов.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Каждое колесо останавливается, когда под его датчиком чёрное, и снова едет на белом.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Колёса не крутятся	Проверь, что вписаны измерения и включено питание моторов.
Колесо крутится назад	Проверь `FORWARD` и подключение мотора.
Останавливается не то колесо	Перепутаны датчики или моторы. Проверь по монитору порта и по меткам Л/П.

9 Первый заезд по линии

8 мин

1. Выключи питание моторов, отключи USB-кабель.
2. Поставь робота на прямой участок: линия между датчиками.
3. Включи питание и наблюдай. Будь готов выключить питание, если робот уходит с дорожки.
4. Попробуй проехать изгиб. Запиши в журнал, что получилось.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Робот едет вдоль линии и проходит плавный изгиб. Он двигается рывками — это нормально для такого способа.

● **ОСНОВНОЕ · ПРЯМАЯ И ПЛАВНЫЙ ИЗГИБ**

Робот проходит прямую и плавный изгиб по двум правилам и не теряет линию.

Проверь себя

● Основное

1. Почему мы измеряем белое и чёрное для каждого датчика отдельно?

- ○ Так требует Arduino
- ○ Датчики не одинаковые: над одним полем их числа отличаются
- ○ Чтобы программа работала быстрее

2. Белое — 100, чёрное — 800. Датчик показывает 300. Что под датчиком?

- ○ Линия: число ближе к чёрному
- ○ Белое поле: число ближе к белому
- ○ Нельзя сказать

3. Робот едет вперёд, линия оказалась под правым датчиком. Что сделает наша программа?

- ○ Остановит левое колесо, робот повернёт налево
- ○ Остановит правое колесо, робот повернёт направо
- ○ Остановит оба колеса

4. Что напечатает этот код, если leftValue равно 512?

```
1 Serial.print("L=");
2 Serial.println(leftValue);
```

- ○ L=leftValue
- ○ L=512
- ○ "L="512

5. Зачем в начале loop() снова писать leftSpeed = speed и rightSpeed = speed?

- ○ Чтобы колесо поехало снова, когда датчик опять увидит белое
- ○ Чтобы робот ехал быстрее
- ○ Без этого программа не загрузится

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 научился сам выбирать, какое колесо остановить, по показаниям двух датчиков. Мы не угадывали числа, а измерили их на своём поле. Рывки при движении — цена простого правила «едет или стоит». В уроке 3.2 робот научится поворачивать плавно: тем сильнее, чем дальше линия ушла в сторону.

Вернись к своей догадке из начала урока. Подтвердилась ли она?

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

датчик линии	Датчик, который светит на пол и измеряет, сколько света отразилось. Белое отражает много, чёрное — мало.
показание	Число, которое программа получает от датчика. У нашей платы это число от 0 до 1023.
монитор порта	Панель Serial Monitor в MyRobotica, где видны сообщения, которые плата отправляет на компьютер.
калибровка	Настройка программы по измерениям своего робота и своего поля.
переменная	Именованное место в памяти, где программа хранит число.
условие	Проверка if: команды в фигурных скобках выполняются, только если проверка верна.
телеметрия	Передача именованных измерений с временем и единицами, чтобы строить графики и сохранять наблюдения.
АЦП	Аналого-цифровое преобразование: напряжение на входе платы превращается в число от 0 до 1023.

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.

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 рабо
#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, fl
#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 рабо
#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
}
```

```
81
82 void courseTelemetryPoll() {
83 #if defined(ROBOT_TELEMETRY_BUILD_ID)
84   courseTelemetry.update();
85 #endif
86 }
87
88 // NAN означает «ещё не вычислено», а не нулевую ошибку или поправку.
89 void courseTelemetryReport(int leftRaw, int rightRaw, float error, fl
90 #if defined(ROBOT_TELEMETRY_BUILD_ID)
91   using namespace robot_telemetry::wire;
92   courseTelemetry.setUnsigned(Line0, leftRaw);
93   courseTelemetry.setUnsigned(Line1, rightRaw);
94   if (isnan(error)) courseTelemetry.setMissing(LineError);
95   else courseTelemetry.setFloat(LineError, error);
96   if (isnan(output)) courseTelemetry.setMissing(ControlOutput);
97   else courseTelemetry.setFloat(ControlOutput, output);
98   courseTelemetry.update();
99 #endif
100 }
101
102 void courseWait(unsigned long durationMs) {
103   unsigned long started = millis();
104   while (millis() - started < durationMs) {
105     courseTelemetryPoll();
106     delay(1);
107   }
108 }
```

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.

Тема 4 · Урок 4.1

Находим и считаем перекрёстки

Робот замечает перекрёсток, останавливается на нём и не считает один перекрёсток дважды.

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

РЕЗУЛЬТАТ УРОКА

Робот едет по линии с П-регулятором и останавливается на нужном по счёту перекрёстке. Каждый перекрёсток считается один раз.

Ты научишься

- Описать, что показывают датчики на перекрёстке.
- Записать условие перекрёстка и объяснить выбор порога.
- Объяснить, почему один перекрёсток может посчитаться много раз, и исправить это.
- Остановить робота на перекрёстке с заданным номером.

Нужно знать

- 3.2 П-регулятор: плавно и быстро

Понадобится

Датчик линии (аналоговый, Trema) × 2

Зачем этот урок

На полях итоговой задачи к грузам ведут ответвления от перекрёстков. Чтобы свернуть в нужное место, робот должен замечать перекрёстки и знать, какой по

счёту перед ним. Сегодня робот научится находить, считать и проезжать перекрёстки.

РАЗМИНКА · ВСПОМНИ

1. Чему равна ошибка П-регулятора, когда линия ровно посередине?
2. Что показывает датчик по шкале 0–100 на чёрном?
3. Что делает `constrain`?

1

ЗАНЯТИЕ 1 · 45 МИН

Перекрёсток глазами датчиков

1 Что видят датчики на перекрёстке

12 мин

Что должен делать робот. Курьер должен заметить перекрёсток, доехать до него и при необходимости остановиться. Отдельный датчик перекрёстка не нужен: используем оба датчика линии и их показания вместе.

Что используем и как это работает. На обычной узкой дорожке датчики находятся по её краям. На широкой поперечной полосе оба могут одновременно увидеть чёрное. Программа проверяет это сочетание и решает, продолжать ли движение. Пока датчики остаются над одной полосой, многократные чтения относятся к одному перекрёстку, а не к нескольким.

Как связать измерение с действием. Сначала проверим признаки на неподвижном роботе, затем въезд и выезд с полосы. Для счёта нужно отделить обнаружение перекрёстка от проезда за его границу. Иначе быстрый цикл насчитает одну и ту же полосу много раз. Другая широкая тёмная область тоже может дать такой сигнал: датчики не знают географию поля.

ПРЕДСКАЖИ

Робот подъезжает к перекрёстку. Что покажут оба датчика по шкале 0–100, когда он окажется на поперечной линии?

1. Загрузи скетч `L32_Error` из урока 3.2.
2. Медленно провези робота руками по тренировочной дорожке через перекрёсток. Смотри на монитор порта.
3. Запиши в журнал показания: на линии посередине, на перекрёстке, после перекрёстка.

ДОЛЖНО ПОЛУЧИТЬСЯ

На перекрёстке оба датчика показывают почти 0: под ними чёрная поперечная линия.

2 Условие перекрёстка

10 мин

Запишем правило: **перекрёсток — когда оба датчика видят чёрное**. «Чёрное» — показание меньше порога 30.

```
bool onCrossing() {
  return leftLevel < 30 && rightLevel < 30;
}
```

`&&` читается как «и»: условие верно, только если верны обе части.

Подсказка: Почему 30, а не 0

Даже на чёрном показание редко ровно 0, а при движении робот видит край линии. Порог 30 надёжно отделяет «чёрное» от «края», но его можно подобрать по своим измерениям.

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить, почему на изгибе линии условие не срабатывает, а на перекрёстке — срабатывает.

3 Стоп на перекрёстке

20 мин

1. Создай скетч `L41_Crossing_Stop`. Перенеси измерения, `kp` и `baseSpeed` из журнала.

Стоп на перекрёстке L41_Crossing_Stop.ino

```
const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

// Измерения и настройки из журнала (уроки 3.1 и 3.2).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;
float kp = 0.5;
int baseSpeed = 100;

int leftLevel = 100;
int rightLevel = 100;

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

void readSensors() {
  leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
  rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
}

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
```

```
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void followLineStep() {
  int error = leftLevel - rightLevel;
  int correction = kp * error;
  drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
        FORWARD, constrain(baseSpeed - correction, 0, 255));
}

void setupRobot() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }
  delay(2000);
}

// Перекрёсток: оба датчика видят чёрное.
bool onCrossing() {
  return leftLevel < 30 && rightLevel < 30;
}

// Ехать по линии, пока не найдём перекрёсток.
void followLineToCrossing() {
  while (true) {
    readSensors();
    if (onCrossing()) {
      return;
    }
    followLineStep();
  }
}

void setup() {
```

```
  setupRobot();
  followLineToCrossing();
  stopMotors();
}

void loop() {
}
```

2. Найди функцию `followLineToCrossing()`. Объясни напарнику, как из неё выходит программа.
3. Поставь робота на линию перед перекрёстком и запусти.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
>
> Робот едет по линии и останавливается на первом перекрёстке.

> ⓘ **Как это устроено: Как работает followLineToCrossing**
>
> `while (true)` — повторять всегда. Внутри: прочитать датчики; если перекрёсток — `return`, то есть выйти из функции; иначе сделать шаг П-регулятора. После выхода программа продолжается со строки `stopMotors();` в `setup()`.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Робот проезжает перекрёсток	Снизь `baseSpeed` или подними порог до 40
Робот останавливается на изгибе	Порог слишком высокий, попробуй 20
Робот не едет	Вписаны ли измерения из журнала?

2

ЗАНЯТИЕ 2 · 45 МИН

Считаем перекрёстки

4 Найди ошибку счёта

8 мин

НАЙДИ ОШИБКУ · РОБОТ НАСЧИТАЛ 40 ПЕРЕКРЁСТКОВ

Соседи хотели считать перекрёстки так:

```
void loop() {
  readSensors();
  if (onCrossing()) {
    count = count + 1;
  }
  followLineStep();
}
```

Робот проехал один перекрёсток, а `count` стал 40. Почему?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить: пока робот едет по широкой чёрной полосе, условие верно много раз подряд, и каждый раз счётчик растёт.

5 Проехать перекрёсток

10 мин

Чтобы посчитать перекрёсток один раз, его нужно **проехать**: ехать прямо, пока датчики не съедут с чёрного.

```
void passCrossing() {
  drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
  do {
    readSensors();
  } while (onCrossing());
  delay(100);
}
```

`do { … } while (условие)` — сделать и повторять, пока условие верно. Короткая пауза в конце уводит датчики подальше от края поперечной линии.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты можешь объяснить, когда закончится цикл в `passCrossing()`.

6 Стоп на нужном перекрёстке

17 мин

1. Создай скетч `L41_Count`. В `TARGET` укажи, на каком перекрёстке остановиться.

Счёт перекрёстков L41_Count.ino

```
const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

// Измерения и настройки из журнала (уроки 3.1 и 3.2).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;
float kp = 0.5;
int baseSpeed = 100;

int leftLevel = 100;
int rightLevel = 100;

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

void readSensors() {
  leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
  rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
}

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
```

```
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void followLineStep() {
  int error = leftLevel - rightLevel;
  int correction = kp * error;
  drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
        FORWARD, constrain(baseSpeed - correction, 0, 255));
}

void setupRobot() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }
  delay(2000);
}

// Перекрёсток: оба датчика видят чёрное.
bool onCrossing() {
  return leftLevel < 30 && rightLevel < 30;
}

// Ехать по линии, пока не найдём перекрёсток.
void followLineToCrossing() {
  while (true) {
    readSensors();
    if (onCrossing()) {
      return;
    }
    followLineStep();
  }
}

// Проехать перекрёсток прямо, пока датчики не съедут с чёрного.
```

```
81  void passCrossing() {
82    drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
83    do {
84      readSensors();
85    } while (onCrossing());
86    delay(100);
87  }
88
89  const int TARGET = 1; // на каком перекрёстке остановиться
90
91  void setup() {
92    setupRobot();
93    for (int i = 1; i < TARGET; i++) {
94      followLineToCrossing();
95      passCrossing();
96    }
97    followLineToCrossing();
98    stopMotors();
99  }
100
101 void loop() {
102 }
```

ПРЕДСКАЖИ

`TARGET = 2`. Сколько раз выполнится цикл `for`? Что робот сделает на первом перекрёстке?

2. Проверь на дорожке с тремя перекрёстками: `TARGET` = 1, 2, 3.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот останавливается ровно на перекрёстке с номером `TARGET`.

7 **Задания по уровням** 10 мин

● **ОСНОВНОЕ · ПЕРВЫЙ ПЕРЕКРЁСТОК**

Доехать по линии до первого перекрёстка и остановиться на нём.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить условие перекрёстка.
- [ ] Я знаю, почему перекрёсток нужно проехать, прежде чем искать следующий.
- [ ] Я могу остановить робота на нужном перекрёстке.

Проверь себя

● Основное

1. По какому признаку программа узнаёт перекрёсток?

- ○ Оба датчика одновременно видят чёрное
- ○ Один датчик видит чёрное
- ○ Оба датчика видят белое

2. Что вернёт выражение leftLevel < 30 && rightLevel < 30, если leftLevel = 10, rightLevel = 80?

- ○ true — это перекрёсток
- ○ false — чёрное только слева
- ○ Ошибку

3. Почему счётчик перекрёстков растёт много раз на одном перекрёстке?

- ○ Пока робот на чёрной полосе, условие верно в каждом повторе loop()
- ○ Датчики сломаны
- ○ Так работает П-регулятор

4. Что делает return внутри followLineToCrossing()?

- ○ Останавливает моторы
- ○ Выходит из функции, программа продолжается после её вызова
- ○ Начинает программу заново

5. **TARGET = 3. Сколько перекрёстков робот проедет прямо до остановки?**

- ○ Два
- ○ Три
- ○ Ни одного

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 находит перекрёстки и считает их без повторов. Следующий шаг — сворачивать на перекрёстке в нужную сторону и проходить маршрут.

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

перекрёсток	Место, где линию пересекает другая линия. Оба датчика одновременно видят чёрное.
порог	Число для сравнения: показание меньше 30 по шкале 0–100 считаем чёрным.
цикл while	Повторяет команды, пока условие верно.
return	Немедленно завершает функцию и возвращает программу туда, откуда функцию вызвали.

Тема 4 · Урок 4.2

Повороты на перекрёстке и маршрут

Робот сворачивает с перекрёстка на нужную линию и проходит маршрут из нескольких решений.

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

РЕЗУЛЬТАТ УРОКА

Робот поворачивает с перекрёстка налево или направо до новой линии, разворачивается на линии и проходит маршрут из нескольких решений, записанных в программе по порядку.

Ты научишься

- Объяснить, почему поворот по датчикам надёжнее поворота по времени.
- Подобрать выезд на центр перекрёстка, чтобы после поворота робот оказался на линии.
- Записать маршрут последовательностью решений «налево / направо / прямо».
- Проверить маршрут на поле и найти, на каком решении он ломается.

Нужно знать

- 4.1 Находим и считаем перекрёстки
- 2.2 Повороты и развороты

Понадобится

Датчик линии (аналоговый, Trema) × 2

Зачем этот урок

Робот находит перекрёсток. Теперь он должен **свернуть** — к грузу, к месту доставки или дальше по дороге. В уроке 2.2 повороты были по времени и зависели от батареи. Сегодня робот будет поворачивать по датчикам: крутиться, пока не найдёт новую линию.

РАЗМИНКА · ВСПОМНИ

1. Почему поворот по времени получается разным при разряженной батарее?
2. Что делает `passCrossing()`?
3. Как робот понимает, что под ним перекрёсток?

ЗАНЯТИЕ 1 · 45 МИН

1 Поворот по датчикам

1 Как повернуть точно

8 мин

Что должен делать робот. Роботу нужно пройти несколько перекрёстков и выбрать нужное направление на каждом. Используем датчики линии для обнаружения событий и заранее записанный план маршрута для выбора действия.

Что используем и как это работает. Датчики отвечают на вопрос «что сейчас под роботом», а маршрут — «что делать следующим». Программа доезжает до перекрёстка, выполняет поворот или проезд прямо, выходит с поперечной полосы и переходит к следующему пункту. Массив или последовательность вызовов хранит порядок, но не измеряет положение робота.

Как связать измерение с действием. Проверяем маршрут по участкам: отмечаем, какой перекрёсток найден и какое действие выполнено. Если робот не обнаружил полосу или повернул не туда, следующие команды будут выполняться уже в другом месте. Поэтому сначала проверим каждую функцию, затем соберём целый маршрут.

ПРЕДСКАЖИ

Робот стоит на перекрёстке и крутится налево. Как он может сам понять, что повернул достаточно и уже смотрит вдоль новой линии?

1. Обсудите с напарником. Поставьте робота на перекрёсток и медленно поверните руками.
2. Следите за левым датчиком: когда он окажется над новой линией?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить идею: крутиться, пока датчик со стороны поворота не увидит новую линию.

2 Выезд на центр перекрёстка

12 мин

Датчики стоят **впереди** колёс. Когда датчики заметили перекрёсток, колёса ещё до него не доехали. Если повернуть сразу, робот окажется сбоку от новой линии. Поэтому сначала надо проехать вперёд так, чтобы колёса встали над центром перекрёстка.

1. Поставь робота так, чтобы датчики были на поперечной линии.
2. Измерь линейкой, сколько ещё ехать колёсам до центра перекрёстка. Запиши в журнал.
3. Время этого проезда — `centerMs` — подберём на следующем шаге.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты знаешь, почему перед поворотом нужен выезд на центр.

3 Поворот на перекрёстке

22 мин

1. Создай скетч `L42_Turn_At_Crossing`. Перенеси измерения, `kp` и `baseSpeed`.

Поворот на перекрёстке L42_Turn_At_Crossing.ino

```
const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

// Измерения и настройки из журнала (уроки 3.1 и 3.2).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;
float kp = 0.5;
int baseSpeed = 100;

int leftLevel = 100;
int rightLevel = 100;

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

void readSensors() {
  leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
  rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
}

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
```

```
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void followLineStep() {
  int error = leftLevel - rightLevel;
  int correction = kp * error;
  drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
        FORWARD, constrain(baseSpeed - correction, 0, 255));
}

void setupRobot() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }
  delay(2000);
}

// Перекрёсток: оба датчика видят чёрное.
bool onCrossing() {
  return leftLevel < 30 && rightLevel < 30;
}

// Ехать по линии, пока не найдём перекрёсток.
void followLineToCrossing() {
  while (true) {
    readSensors();
    if (onCrossing()) {
      return;
    }
    followLineStep();
  }
}

// Проехать перекрёсток прямо, пока датчики не съедут с чёрного.
```

```
void passCrossing() {
  drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
  do {
    readSensors();
  } while (onCrossing());
  delay(100);
}

int turnSpeed = 100;
int centerMs = 250; // подбираем: проехать от датчиков до центра пере
int leaveMs = 200;  // время, чтобы датчик ушёл со старой линии

// Выехать на центр перекрёстка: датчики стоят впереди колёс.
void toCenter() {
  drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
  delay(centerMs);
  stopMotors();
}

// Крутиться на месте, пока датчик со стороны поворота не найдёт новую
void turnUntilLine(bool left) {
  if (left) {
    drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);
  } else {
    drive(FORWARD, turnSpeed, BACKWARD, turnSpeed);
  }
  delay(leaveMs);
  do {
    readSensors();
  } while ((left ? leftLevel : rightLevel) > 30);
  stopMotors();
}

void turnLeftAtCrossing() {
  toCenter();
  turnUntilLine(true);
}

void turnRightAtCrossing() {
  toCenter();
  turnUntilLine(false);
```

```
}

// Развернуться на линии: крутиться, пока линия снова не окажется под
void turnAroundOnLine() {
  turnUntilLine(true);
}

// Ехать по линии заданное время.
void followLineFor(unsigned long ms) {
  unsigned long start = millis();
  while (millis() - start < ms) {
    readSensors();
    followLineStep();
  }
  stopMotors();
}

void setup() {
  setupRobot();
  followLineToCrossing();
  turnLeftAtCrossing();
  followLineFor(1500);
  turnAroundOnLine();
  followLineToCrossing();
  stopMotors();
}

void loop() {
}
```

2. Разберите функцию `turnUntilLine`. Зачем в ней `delay(leaveMs)` перед поиском линии?
3. Запусти на дорожке с перекрёстком. Подбери `centerMs`: после поворота робот должен уверенно поехать по новой линии.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Робот доезжает до перекрёстка, поворачивает налево и едет по новой линии.

Как это устроено: Зачем leaveMs

В начале поворота датчик ещё над старой линией и сразу «нашёл бы» её. Поэтому первые `leaveMs` миллисекунд робот просто крутится, не глядя на датчики, — чтобы датчик ушёл со старой линии. Потом ищем новую.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
После поворота робот сбоку от линии	Подбери `centerMs`
Робот крутится бесконечно	Датчик не видит линию: проверь порог 30 и измерения
Робот останавливается сразу	Увеличь `leaveMs`

ЗАНЯТИЕ 2 · 45 МИН

2 Маршрут из решений

4 Разворот на линии

10 мин

В конце ответвления, у груза или места доставки, роботу нужно развернуться и вернуться к перекрёстку. Разворот — тот же поворот по датчикам, только линия теперь сзади.

1. В скетче найди вызов `turnAroundOnLine()`. Проверь на дорожке: проехать по ответвлению, развернуться, вернуться к перекрёстку.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот разворачивается, находит ту же линию и возвращается к перекрёстку.

5 Маршрут по порядку

22 мин

Маршрут — это решения на перекрёстках по порядку. Робот не видит, какой это перекрёсток: он просто выполняет следующее решение.

1. Создай скетч `L42_Route`. Начерти в журнале тренировочную дорожку и путь робота.

Маршрут L42_Route.ino

```
const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

// Измерения и настройки из журнала (уроки 3.1 и 3.2).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;
float kp = 0.5;
int baseSpeed = 100;

int leftLevel = 100;
int rightLevel = 100;

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

void readSensors() {
  leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
  rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
}

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
```

```
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void followLineStep() {
  int error = leftLevel - rightLevel;
  int correction = kp * error;
  drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
        FORWARD, constrain(baseSpeed - correction, 0, 255));
}

void setupRobot() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }
  delay(2000);
}

// Перекрёсток: оба датчика видят чёрное.
bool onCrossing() {
  return leftLevel < 30 && rightLevel < 30;
}

// Ехать по линии, пока не найдём перекрёсток.
void followLineToCrossing() {
  while (true) {
    readSensors();
    if (onCrossing()) {
      return;
    }
    followLineStep();
  }
}

// Проехать перекрёсток прямо, пока датчики не съедут с чёрного.
```

```
void passCrossing() {
  drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
  do {
    readSensors();
  } while (onCrossing());
  delay(100);
}

int turnSpeed = 100;
int centerMs = 250; // подбираем: проехать от датчиков до центра пере
int leaveMs = 200;  // время, чтобы датчик ушёл со старой линии

// Выехать на центр перекрёстка: датчики стоят впереди колёс.
void toCenter() {
  drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
  delay(centerMs);
  stopMotors();
}

// Крутиться на месте, пока датчик со стороны поворота не найдёт новую
void turnUntilLine(bool left) {
  if (left) {
    drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);
  } else {
    drive(FORWARD, turnSpeed, BACKWARD, turnSpeed);
  }
  delay(leaveMs);
  do {
    readSensors();
  } while ((left ? leftLevel : rightLevel) > 30);
  stopMotors();
}

void turnLeftAtCrossing() {
  toCenter();
  turnUntilLine(true);
}

void turnRightAtCrossing() {
  toCenter();
  turnUntilLine(false);
```

```
}

// Развернуться на линии: крутиться, пока линия снова не окажется под
void turnAroundOnLine() {
  turnUntilLine(true);
}

// Ехать по линии заданное время.
void followLineFor(unsigned long ms) {
  unsigned long start = millis();
  while (millis() - start < ms) {
    readSensors();
    followLineStep();
  }
  stopMotors();
}

// Маршрут: решения на перекрёстках по порядку.
void setup() {
  setupRobot();

  followLineToCrossing();
  turnLeftAtCrossing();   // решение 1: налево

  followLineToCrossing();
  passCrossing();         // решение 2: прямо

  followLineToCrossing();
  turnRightAtCrossing();  // решение 3: направо

  followLineToCrossing();
  stopMotors();           // конец маршрута
}

void loop() {
}
```

ПРЕДСКАЖИ

По программе: налево, прямо, направо. Нарисуй, где робот остановится на дорожке.

2. Запусти. Если маршрут сломался — на каком решении? Почему?

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот проходит маршрут из трёх решений и останавливается там, где ты предсказал.

НАЙДИ ОШИБКУ · РОБОТ ПОВОРАЧИВАЕТ НЕ НА ТОМ ПЕРЕКРЁСТКЕ

У соседей после `turnLeftAtCrossing()` сразу стоит `turnRightAtCrossing()`, без `followLineToCrossing()` между ними. Что сделает робот?

6 Задания по уровням

13 мин

ОСНОВНОЕ · ПОВОРОТ И РАЗВОРОТ

Повернуть с перекрёстка налево и направо и уехать по новой линии; развернуться на линии и вернуться.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, как робот понимает, что поворот закончен.
- [ ] Я могу записать маршрут решениями по порядку.
- [ ] Я нахожу, на каком решении сломался маршрут.

Проверь себя

● Основное

1. Почему поворот «до линии» надёжнее поворота по времени?

- ○ Он заканчивается, когда датчик видит новую линию, и не зависит от батареи
- ○ Он быстрее
- ○ Он не использует моторы

2. Зачем перед поворотом проехать немного вперёд (centerMs)?

- ○ Датчики впереди колёс: колёса должны встать над центром перекрёстка
- ○ Чтобы набрать скорость
- ○ Чтобы проверить батарею

3. Что случится, если в turnUntilLine убрать delay(leaveMs)?

- ○ Робот сразу остановится: датчик ещё видит старую линию
- ○ Робот повернёт быстрее и точнее
- ○ Ничего не изменится

4. **Маршрут: налево, прямо, направо. Что выполнит робот на втором перекрёстке?**

- ○ Проедет прямо
- ○ Повернёт налево
- ○ Повернёт направо

5. **char route[] = {'L', 'S', 'R'}; Чему равно route[2]?**

- ○ 'S'
- ○ 'R'
- ○ 'L'

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 сворачивает по датчикам и проходит маршрут из решений. Для итоговой задачи осталось научиться видеть груз и брать его захватом.

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

маршрут	Последовательность решений на перекрёстках: налево, направо или прямо.
центр перекрёстка	Точка, где пересекаются линии. Для поворота на месте над ней должны оказаться колёса.
массив	Список значений под одним именем: route[0], route[1] и так далее.
millis()	Сколько миллисекунд прошло с включения платы. Позволяет ехать заданное время, не останавливая программу.

Тема 5 · Урок 5.1

Ультразвук: видим груз

Робот измеряет расстояние звуком и останавливается перед грузом, не путая «нет эха» с «груз рядом».

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

РЕЗУЛЬТАТ УРОКА

Робот едет по линии и останавливается на заданном расстоянии от груза. Если эхо не вернулось, программа не принимает это за «груз рядом».

Ты научишься

- Объяснить, как ультразвуковой датчик измеряет расстояние.
- Вывести расстояние в монитор порта и проверить его линейкой.
- Отличить отсутствие эха от маленького расстояния.
- Остановить робота перед грузом по измерению, а не по времени.

Нужно знать

- 3.2 П-регулятор: плавно и быстро

Понадобится

Ультразвуковой датчик расстояния | Датчик линии (аналоговый, Trema) × 2

Зачем этот урок

Груз стоит в конце ответвления. Робот должен подъехать к ней и остановиться ровно так, чтобы захват мог её взять. Если ехать «по времени», робот то не доедет, то врежется. Сегодня он научится видеть груз ультразвуком — так же, как летучая мышь видит препятствия.

РАЗМИНКА · ВСПОМНИ

1. Как робот останавливается на перекрёстке в программе из урока 4.1?
2. Что делает `break` в цикле? (Если не помнишь — разберём сегодня.)
3. Какие две команды нужны, чтобы робот ехал по линии с П-регулятором?

1

ЗАНЯТИЕ 1 · 45 МИН

Измеряем расстояние

1 Как работает ультразвук

8 мин

Что должен делать робот. Курьеру нужно обнаружить предмет впереди, оценить расстояние до него и остановиться перед грузом. Датчики линии смотрят вниз и эту задачу не решают. Добавим ультразвуковой датчик: он поможет измерить расстояние, но не отличит груз от стены.

Что используем и как это работает. Передатчик датчика излучает короткую серию ультразвуковых колебаний, которые человек не слышит. Звук отражается от предмета, а приёмник улавливает эхо. Arduino запускает измерение импульсом на TRIG и измеряет длительность сигнала ECHO. Это электрический сигнал о времени измерения, а не сам звук. Путь звука двойной: до предмета и обратно. Поэтому расстояние равно скорости звука, умноженной на время и делённой на два.

Как связать измерение с действием. В учебной программе время в микросекундах делится на 58, чтобы получить приблизительные сантиметры: около 29 мкс на сантиметр в одну сторону и столько же обратно. Например, 580 мкс соответствует примерно 10 см. Целочисленный результат отбрасывает дробную часть; температура и отражающая поверхность тоже влияют на измерение. Сравниваем его с линейкой от передней поверхности датчика.

Если измерение не получилось. Наклонная поверхность может отразить звук в сторону, мягкая — поглотить, а слишком близкий или далёкий предмет — не дать пригодного эха. Поэтому «нет эха» означает «расстояние неизвестно», а не ноль и не свободный путь. В коде для этого есть отдельное значение −1. Программа принимает решение об остановке только после проверки измерения. Базовый пример при −1 продолжает движение — это ограничение учебного алгоритма, которое разберём в задании на потерю эха.

Датчик посылает короткий щелчок ультразвука и слушает эхо. Чем дальше предмет, тем дольше звук летит туда и обратно. По времени полёта считаем расстояние.

ПРЕДСКАЖИ

Звук пролетает 1 см примерно за 29 микросекунд. Предмет в 10 см. Сколько времени пройдёт до возвращения эха? (Не забудь: туда и обратно.)

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить: время эха → расстояние. До предмета в 10 см и обратно — около 580 мкс.

2 Расстояние в мониторе порта

20 мин

1. Посмотри номера контактов TRIG и ECHO на карточке у робота. Карточки нет — проследи провода от выводов TRIG и ECHO датчика до платы и прочитай номера у гнёзд. Запиши в журнал.
2. Создай скетч `L51_Distance` и впиши номера контактов вместо `-1`.

Расстояние в мониторе порта L51_Distance.ino

```
// Контакты ультразвукового датчика: впиши номера из таблицы проводки
const int TRIG_PIN = -1;
const int ECHO_PIN = -1;

// Расстояние в сантиметрах или -1, если эхо не вернулось.
int readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000);
  if (echoUs == 0) {
    return -1;
  }
  return echoUs / 58;
}

void setupSonar() {
  // Без контактов датчика программа дальше не пойдёт.
  if (TRIG_PIN < 0 || ECHO_PIN < 0) {
    while (true) {
    }
  }
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

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

void loop() {
  int distance = readDistanceCm();
  if (distance < 0) {
    Serial.println("нет эха");
  } else {
    Serial.print(distance);
```

```
    Serial.println(" см");
  }
  delay(200);
}
```

3. Загрузи, открой монитор порта. Ставь груз на 5, 10, 20 и 30 см по линейке. Запиши показания.

По линейке	Показание датчика
5 см	
10 см	
20 см	
30 см	

ДОЛЖНО ПОЛУЧИТЬСЯ

Показания датчика близки к линейке, разница — 1–2 см.

Как это устроено: Как работает readDistanceCm

- На TRIG подаём короткий импульс 10 мкс — датчик щёлкает.
- `pulseIn(ECHO_PIN, HIGH, 25000)` ждёт эхо не дольше 25 000 мкс и возвращает время в микросекундах. Если эхо не вернулось — возвращает 0.
- Делим время на 58: это 29 мкс на сантиметр туда и 29 обратно.
- Если эха нет, функция возвращает −1: «измерения нет».

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Программа ничего не печатает	Вписаны ли номера TRIG и ECHO вместо −1?
Всегда «нет эха»	Перепутаны TRIG и ECHO или датчик не получает питание
Числа прыгают	Груз должна стоять прямо напротив датчика

3 «Нет эха» — не «рядом»

15 мин

1. Направь датчик в пустоту или на мягкую ткань под углом. Что печатает монитор?
2. Поднеси ладонь вплотную к датчику, ближе 2 см. Что печатает?

НАЙДИ ОШИБКУ · РОБОТ ЗАКРЫЛ ЗАХВАТ В ПУСТОТЕ

Соседи написали остановку так: `if (readDistanceCm() <= 5) { stopMotors(); }`. Робот иногда останавливается посреди пустой дороги. Почему? Что вернёт функция, если эха нет?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить, почему условие остановки должно проверять, что расстояние больше нуля.

2

ЗАНЯТИЕ 2 · 45 МИН

Стоп перед грузом

4 Едем к грузу и останавливаемся

17 мин

1. Создай скетч `L51_Stop_At_Can`. Перенеси измерения, `kp`, `baseSpeed` и контакты датчика.

Стоп перед грузом L51_Stop_At_Can.ino

```
const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

// Измерения и настройки из журнала (уроки 3.1 и 3.2).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;
float kp = 0.5;
int baseSpeed = 100;

int leftLevel = 100;
int rightLevel = 100;

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

void readSensors() {
  leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
  rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
}

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
```

```
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void followLineStep() {
  int error = leftLevel - rightLevel;
  int correction = kp * error;
  drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
        FORWARD, constrain(baseSpeed - correction, 0, 255));
}

void setupRobot() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }
  delay(2000);
}

// Контакты ультразвукового датчика: впиши номера из таблицы проводки
const int TRIG_PIN = -1;
const int ECHO_PIN = -1;

// Расстояние в сантиметрах или -1, если эхо не вернулось.
int readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000);
  if (echoUs == 0) {
    return -1;
  }
  return echoUs / 58;
}
```

```
81
82  void setupSonar() {
83    // Без контактов датчика программа дальше не пойдёт.
84    if (TRIG_PIN < 0 || ECHO_PIN < 0) {
85      while (true) {
86      }
87    }
88    pinMode(TRIG_PIN, OUTPUT);
89    pinMode(ECHO_PIN, INPUT);
90  }
91
92  const int STOP_CM = 5; // на каком расстоянии от груза остановиться
93
94  void setup() {
95    setupSonar();
96    setupRobot();
97
98    while (true) {
99      int distance = readDistanceCm();
100     if (distance > 0 && distance <= STOP_CM) {
101       break;
102     }
103     readSensors();
104     followLineStep();
105   }
106   stopMotors();
107 }
108
109 void loop() {
110 }
```

2. Поставь груз в конце линии. Запусти робота с другого конца.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Робот едет по линии и останавливается перед грузом, не касаясь его.

ⓘ **Как это устроено: Что делает break**

`break` немедленно выходит из цикла `while`. После цикла программа выполняет `stopMotors()`. Так «стоп по расстоянию» оказывается в одном месте программы.

5 Точная остановка

15 мин

1. Измерь линейкой, на каком расстоянии от груза остановился робот.
2. Подбери `STOP_CM` и `baseSpeed`, чтобы робот останавливался там, где удобно захвату.
3. Повтори три раза. Запиши в журнал `STOP_CM` и фактическое расстояние.

Подсказка: Робот останавливается дальше или ближе, чем STOP_CM

Робот не останавливается мгновенно: он ещё немного катится. Чем выше скорость, тем дальше. Можно перед грузом ехать медленнее или поставить `STOP_CM` чуть больше.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот три раза подряд останавливается на нужном расстоянии от груза.

6 Задания по уровням

13 мин

ОСНОВНОЕ · СТОП В 5 СМ

Остановиться в 5 см от груза, стоящего в конце линии.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, как датчик измеряет расстояние.
- [ ] Я знаю, почему −1 нельзя принимать за «груз рядом».
- [ ] Мой робот останавливается перед грузом по измерению.

Проверь себя

Основное

1. Как ультразвуковой датчик узнаёт расстояние?

- ○ По времени, за которое звук долетает до предмета и возвращается
- ○ По яркости отражённого света
- ○ По громкости эха

2. **readDistanceCm() вернула −1. Что это значит?**

- ○ Груз вплотную к датчику
- ○ Эхо не вернулось — измерения нет
- ○ Датчик измерил 1 см

3. **Какое условие остановки правильное?**

- ○ distance <= 5
- ○ distance > 0 && distance <= 5
- ○ distance == 5

4. **Робот всегда останавливается на 3 см ближе, чем STOP_CM. Почему?**

- ○ После команды остановки робот ещё немного катится
- ○ Датчик врёт
- ○ Не хватает памяти

5. **Три измерения: 12, 85, 13. Какая медиана?**

- ○ 13
- ○ 36
- ○ 85

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 видит груз и останавливается перед ним по измерению. Осталось научиться его брать — в следующем уроке.

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

ультразвук	Звук такой высоты, что человек его не слышит. Датчик посылает щелчок и ждёт эхо.
эхо	Звук, отразившийся от предмета и вернувшийся к датчику.
pulseIn	Команда, которая измеряет, сколько микросекунд длился сигнал на входе.
мкс	Микросекунда — миллионная доля секунды.
медиана	Среднее по величине из нескольких чисел: из 4, 25 и 5 это 5.

Тема 5 · Урок 5.2

Захват: берём и ставим

Робот подъезжает к грузу, берёт его захватом, увозит и ставит в нужном месте.

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

РЕЗУЛЬТАТ УРОКА

Робот подъезжает к грузу по ультразвуку, берёт её, разворачивается, везёт по линии и ставит на другом конце. Действия оформлены функциями pickUp() и putDown().

Ты научишься

- Объяснить, как сервопривод поворачивается на заданный угол.
- Найти углы «открыто» и «закрыто» для своего захвата опытом.
- Соблюдать порядок: остановиться — закрыть; открыть — подождать — отъехать.
- Собрать перевозку груза из функций прошлых уроков.

Нужно знать

- 5.1 Ультразвук: видим груз
- 4.2 Повороты на перекрёстке и маршрут

Понадобится

Захват с сервоприводом | Ультразвуковой датчик расстояния | Датчик линии (аналоговый, Trema) × 2

Зачем этот урок

Робот умеет доезжать до груза. Остался последний навык курьера — взять груз, довезти и аккуратно поставить. Сегодня робот соберёт всё вместе: линия, ультразвук и захват.

РАЗМИНКА · ВСПОМНИ

1. Какое условие остановки перед грузом правильное и почему?
2. Как робот разворачивается на линии?
3. Что делает `followLineFor(1500)`?

БЕЗОПАСНОСТЬ

Не держи пальцы в захвате, когда включено питание. Сервопривод может резко повернуться при загрузке программы.

ЗАНЯТИЕ 1 · 45 МИН

1 Сервопривод и захват

1 Как работает сервопривод

8 мин

Что должен делать робот. Курьер должен взять груз, удержать его при перевозке и отпустить в нужном месте. Для этого добавим механический захват с сервоприводом. Ультразвук помогает подъехать, но сам ничего не захватывает.

Что используем и как это работает. Внутри сервопривода есть мотор, редуктор, датчик положения и управляющая схема. По управляющим импульсам он получает заданное положение вала и старается его удержать. Библиотека Servo формирует эти импульсы; gripper.write задаёт положение, а не скорость вращения. Рычаг превращает поворот вала в движение губок захвата.

Как связать измерение с действием. Углы «открыто» и «закрыто» подбираем для своей механики небольшими изменениями; чужие числа могут упереть механизм в ограничитель. Сначала останавливаем колёса, затем закрываем захват и даём ему время. Команда закрытия не доказывает, что груз удерживается: это проверяем наблюдением. Подключение и питание сверяем с подтверждённой схемой комплекта.

Обычный мотор крутится, пока включён. **Сервопривод** поворачивается на заданный угол и держит его. Внутри у него мотор, шестерёнки и датчик положения. Командуем углом в градусах: `gripper.write(90)`.

ПРЕДСКАЖИ

Что сделает сервопривод после `gripper.write(90)`, а потом `gripper.write(40)`?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить, чем сервопривод отличается от мотора колеса.

2 Найди углы «открыто» и «закрыто»

22 мин

У каждого захвата свои углы. Найдём их опытом, как время поворота в уроке 2.2.

1. Посмотри контакт сервопривода на карточке у робота и впиши его вместо `-1`.

Проба угла L52_Servo_Test.ino

```
#include <Servo.h>

// Контакт сервопривода захвата: впиши номер из таблицы проводки своей
const int SERVO_PIN = -1;

int angle = 90; // меняй понемногу, чтобы найти «открыто» и «закрыто»

Servo gripper;

void setup() {
  if (SERVO_PIN < 0) {
    while (true) {
    }
  }
  gripper.attach(SERVO_PIN);
  gripper.write(angle);
}

void loop() {
}
```

2. Начни с `angle = 90`. Загрузи. Меняй угол по 10°: найди положение, где захват открыт так, что груз свободно входит.
3. Найди угол, где захват держит груз крепко, но не давит изо всех сил.
4. Запиши оба угла в журнал.

Подсказка: Сервопривод гудит или дрожит

Захват упёрся в груз или в собственный упор и не может дойти до угла. Отступи на 5–10° от этого положения: сервопривод не должен работать в упор.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале записаны углы «открыто» и «закрыто», при которых груз свободно входит и надёжно держится.

3 Порядок действий важен

15 мин

Сопоставь функции `pickUp()` и `putDown()` в итоговом скетче с правилом порядка (следующее занятие):

- **Взять:** сначала остановиться, потом закрыть захват.
- **Поставить:** открыть захват, подождать, пока он раскроется, и только потом отъехать.

НАЙДИ ОШИБКУ · ГРУЗ УЕЗЖАЕТ С РОБОТОМ

У соседей `putDown()` такая: `gripper.write(OPEN_ANGLE);` и сразу `drive(BACKWARD, …)`. Груз то падает, то уезжает вместе с роботом. Почему? Как исправить?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить, зачем пауза после открытия захвата.

2

ЗАНЯТИЕ 2 · 45 МИН

Перевозка груза

4 Взять и поставить

20 мин

1. Создай скетч `L52_Pick_Place`. Перенеси измерения, `kp`, `baseSpeed`, контакты датчика и сервопривода, свои углы.

Взять и поставить L52_Pick_Place.ino

```
#include <Servo.h>

const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
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;
const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;

// Измерения и настройки из журнала (уроки 3.1 и 3.2).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;
float kp = 0.5;
int baseSpeed = 100;

int leftLevel = 100;
int rightLevel = 100;

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

void readSensors() {
  leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
  rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
}

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 stopMotors() {
  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
}

void followLineStep() {
  int error = leftLevel - rightLevel;
  int correction = kp * error;
  drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
        FORWARD, constrain(baseSpeed - correction, 0, 255));
}

void setupRobot() {
  pinMode(LEFT_DIR_PIN, OUTPUT);
  pinMode(LEFT_SPEED_PIN, OUTPUT);
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  stopMotors();
  // Без измерений робот не поедет.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
    }
  }
  delay(2000);
}

// Контакты ультразвукового датчика: впиши номера из таблицы проводки
const int TRIG_PIN = -1;
const int ECHO_PIN = -1;

// Расстояние в сантиметрах или -1, если эхо не вернулось.
int readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000);
  if (echoUs == 0) {
    return -1;
  }
```

```
  return echoUs / 58;
}

void setupSonar() {
  // Без контактов датчика программа дальше не пойдёт.
  if (TRIG_PIN < 0 || ECHO_PIN < 0) {
    while (true) {
    }
  }
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

int turnSpeed = 100;
int centerMs = 250; // подбираем: проехать от датчиков до центра пере
int leaveMs = 200;  // время, чтобы датчик ушёл со старой линии

// Выехать на центр перекрёстка: датчики стоят впереди колёс.
void toCenter() {
  drive(FORWARD, baseSpeed, FORWARD, baseSpeed);
  delay(centerMs);
  stopMotors();
}

// Крутиться на месте, пока датчик со стороны поворота не найдёт новую
void turnUntilLine(bool left) {
  if (left) {
    drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);
  } else {
    drive(FORWARD, turnSpeed, BACKWARD, turnSpeed);
  }
  delay(leaveMs);
  do {
    readSensors();
  } while ((left ? leftLevel : rightLevel) > 30);
  stopMotors();
}

void turnLeftAtCrossing() {
  toCenter();
  turnUntilLine(true);
```

```
}

void turnRightAtCrossing() {
  toCenter();
  turnUntilLine(false);
}

// Развернуться на линии: крутиться, пока линия снова не окажется под
void turnAroundOnLine() {
  turnUntilLine(true);
}

// Ехать по линии заданное время.
void followLineFor(unsigned long ms) {
  unsigned long start = millis();
  while (millis() - start < ms) {
    readSensors();
    followLineStep();
  }
  stopMotors();
}

// Сервопривод захвата: впиши контакт и свои углы из журнала.
const int SERVO_PIN = -1;
const int OPEN_ANGLE = 90;
const int CLOSED_ANGLE = 40;
const int STOP_CM = 5;
Servo gripper;

void setupGripper() {
  // Без контакта сервопривода программа дальше не пойдёт.
  if (SERVO_PIN < 0) {
    while (true) {
    }
  }
  gripper.attach(SERVO_PIN);
  gripper.write(OPEN_ANGLE);
  delay(500);
}

// Ехать по линии и остановиться перед грузом по ультразвуку.
```

```
void approachCan() {
  while (true) {
    int distance = readDistanceCm();
    if (distance > 0 && distance <= STOP_CM) {
      break;
    }
    readSensors();
    followLineStep();
  }
  stopMotors();
}

// Сначала остановиться, потом закрыть захват.
void pickUp() {
  stopMotors();
  gripper.write(CLOSED_ANGLE);
  delay(600);
}

// Открыть захват, подождать и только потом отъехать назад.
void putDown() {
  stopMotors();
  gripper.write(OPEN_ANGLE);
  delay(600);
  drive(BACKWARD, baseSpeed, BACKWARD, baseSpeed);
  delay(300);
  stopMotors();
}

const unsigned long CARRY_MS = 1500; // сколько ехать по линии с грузо

void setup() {
  setupSonar();
  setupGripper();
  setupRobot();

  approachCan();
  pickUp();
  turnAroundOnLine();
  followLineFor(CARRY_MS);
  putDown();
```

```
204 }
205
206 void loop() {
207 }
```

2. Поставь груз в конце линии, робота — на другом конце. Запусти.

ПРЕДСКАЖИ

Прочитай `setup()` и расскажи по шагам, что сделает робот.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот подъезжает, берёт груз, разворачивается, везёт его по линии и ставит.

5 Надёжность

12 мин

1. Повтори перевозку три раза подряд. Где робот ошибается чаще всего?
2. Подбери `STOP_CM` так, чтобы груз оказывался в середине захвата.
3. Подбери `CARRY_MS`, чтобы груз ставился в нужном месте.

ДОЛЖНО ПОЛУЧИТЬСЯ

Три перевозки подряд без помощи рук. Значения записаны в журнал.

6 Задания по уровням

13 мин

ОСНОВНОЕ · ПЕРЕВОЗКА ПО ЛИНИИ

Взять груз с конца линии и поставить его на другом конце той же линии.

ПРОВЕРЬ СЕБЯ

- [ ] Мои углы «открыто» и «закрыто» записаны в журнале.
- [ ] Я соблюдаю порядок: остановиться — закрыть; открыть — подождать — отъехать.
- [ ] Робот перевозит груз без помощи рук.

Проверь себя

Основное

1. Чем сервопривод отличается от мотора колеса?

- ○ Поворачивается на заданный угол и держит его
- ○ Крутится быстрее
- ○ Работает без питания

2. В каком порядке правильно взять груз?

- ○ Закрыть захват на ходу, потом остановиться
- ○ Остановиться, потом закрыть захват
- ○ Отъехать назад и закрыть захват

3. Зачем в putDown() пауза после открытия захвата?

- ○ Чтобы захват успел раскрыться и не утащил груз
- ○ Чтобы зарядилась батарея
- ○ Пауза не нужна

4. Сервопривод гудит, захват дрожит у груза. Что сделать?

- ○ Поставить угол «закрыто» на 5–10° мягче, чтобы не давить в упор
- ○ Увеличить скорость робота
- ○ Убрать delay

5. Почему углы захвата подбирают опытом?

- ○ Они зависят от конструкции захвата и размера груза
- ○ Углы у всех сервоприводов разные каждый день
- ○ Так требует библиотека Servo

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

У робота есть все умения итоговой задачи: ехать, поворачивать, держаться линии, находить перекрёстки, видеть груз и перевозить его. Дальше — итоговый проект.

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

сервопривод	Мотор, который поворачивается на заданный угол и держит его.
угол	Положение рычага сервопривода в градусах, обычно от 0 до 180.
библиотека	Готовый набор команд, который подключают строкой #include. Servo — библиотека для сервоприводов.

Тема 6 · Урок 6.1

Робот-курьер: план и программа

Итоговая задача, часть 1: выбираешь уровень, планируешь маршрут и собираешь программу из проверенных частей.

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

РЕЗУЛЬТАТ УРОКА

Выбран уровень итоговой задачи, записан и проверен план маршрута. Программа проходит весь маршрут хотя бы в пробном заезде, работающие версии сохранены.

Ты научишься

- Выбрать уровень итоговой задачи по своим выполненным заданиям ●, ★ и ★★.
- Разбить задачу на участки и записать план маршрута до написания кода.
- Собрать программу из функций прошлых уроков и проверять её по частям.
- Сохранять рабочую версию после каждой проверенной части.

Нужно знать

- 2.1 Моторы: едем и останавливаемся
- 2.2 Повороты и развороты
- 3.1 Датчики линии и первое следование
- 3.2 П-регулятор: плавно и быстро
- 4.1 Находим и считаем перекрёстки
- 4.2 Повороты на перекрёстке и маршрут
- 5.1 Ультразвук: видим груз
- 5.2 Захват: берём и ставим

Понадобится

- Ультразвуковой датчик расстояния
- Захват с сервоприводом
- Датчик линии (аналоговый, Trema) × 2

Твоя задача

Ты — инженер службы доставки. Такие роботы-курьеры уже ездят по тротуарам и складам: сами находят дорогу, забирают груз и привозят его по адресу. Твой робот должен сам выехать со старта, перевезти два груза на новые места и остановиться в финише. Во время заезда ему никто не помогает: всё решает твоя программа.

Новых команд здесь нет. Всё нужное ты уже делал: ездил и поворачивал, держался линии с П-регулятором, находил перекрёстки, видел груз ультразвуком и работал захватом. Сейчас нужно собрать эти умения в одну программу. В уроке 6.2 — зачётные заезды.

РАЗМИНКА · ВСПОМНИ

1. Чем `setup()` отличается от `loop()` и где должен выполняться маршрут, чтобы после финиша робот стоял?
2. Как программа понимает, что под роботом перекрёсток?
3. Что делать, если ультразвук не получил эхо?

1

ЗАНЯТИЕ 1 · 45 МИН

Уровень и план

1 Выбери уровень

6 мин

Что должен делать робот. Теперь роботу нужно доставить два груза и закончить поездку на финише. Нового датчика не добавляем: объединяем уже проверенные возможности — движение, поиск линии и перекрёстков, подъезд по ультразвуку и захват.

Что используем и как это работает. План — последовательность действий с условиями завершения. Для каждого участка запиши, какие показания нужны, какое действие выполняют моторы или захват и по какому признаку переходить дальше. Функции позволяют назвать эти действия и проверить их по отдельности; вызов функции передаёт управление её командам и затем возвращает его в маршрут.

Как связать измерение с действием. Перед сборкой маршрута проверь каждую функцию своим коротким опытом. Пустая функция в заготовке не выполняет действие, даже если программа успешно собирается. После последнего участка нужна явная остановка; повторение маршрута в loop без решения автора запустило бы доставку заново.

Задача есть в трёх уровнях. Правила и критерии оценки у них общие, а поля разные. Уровень выбирают по тому, какие задания уроков ты уже выполнил: задания ★ готовили к среднему полю, ★★ — к продвинутому.

Уровень итоговой задачи	Какие задания уроков нужны
Начальный S-образная линия, два перекрёстка, порядок действий известен заранее.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2
Средний Сеть дорог, четыре перекрёстка, проезд прямо и повторные встречи с перекрёстками.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2 ★ Средний уровень 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2
Продвинутый Шесть перекрёстков, крутые S-дуги, маршрут по карточке составляешь сам.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2 ★ Средний уровень 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2 ★★ Продвинутый уровень 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2

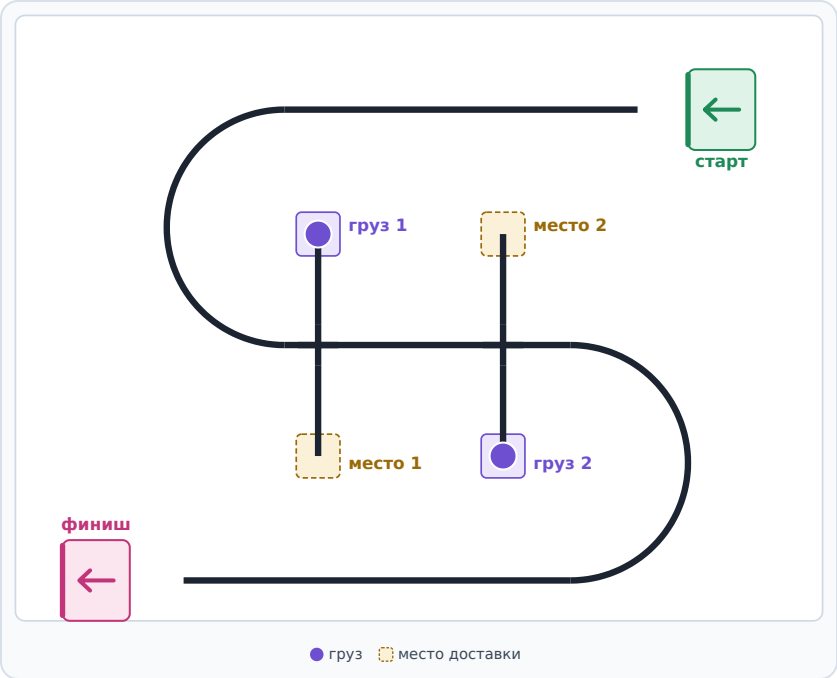
1. Открой бортовой журнал и посмотри, какие задания ●, ★ и ★★ у тебя получились.
2. Выбери уровень сам и отметь его ниже. Если сомневаешься, начни с начального: после зачёта можно попробовать следующий.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Уровень выбран и записан в журнал.

2 Разберись в задании своего уровня

8 мин



Начальное поле.

1. Выехать из зоны старта влево и найти линию.
2. По линии пройти левую дугу и доехать до первого перекрёстка.
3. Перевезти груз 1 с верхнего конца ответвления на нижний — в место 1.
4. Доехать до второго перекрёстка и перевезти груз 2 с нижнего конца на верхний — в место 2.
5. Пройти правую дугу, выехать с конца линии в зону финиша и остановиться.

У старта и финиша есть короткие участки без линии: там робот едет прямо по твоим командам.

1. Покажи пальцем на схеме: старт, оба груза, оба места доставки, финиш.
2. Прочитай критерии оценки в конце урока. За что дают больше всего баллов?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты можешь пересказать задание своего уровня своими словами.

3 Проверь умения по отдельности

10 мин

> ⚠ **БЕЗОПАСНОСТЬ**
> После загрузки старой программы робот может сразу поехать. Загружай программы при выключенном питании моторов, первый запуск — на подставке.

Если хотя бы одно умение не работает, вся программа не поедет. Проверь каждое коротким опытом.

1. Оба колеса крутятся вперёд и назад; моторы стоят, когда выключены (урок 2.1).
2. Робот поворачивает на 90° и разворачивается на 180° (урок 2.2).
3. Робот едет по линии с П-регулятором и проходит дуги своего поля (урок 3.2).
4. Робот находит перекрёсток и поворачивает с него на нужную линию (уроки 4.1, 4.2).
5. Ультразвук видит груз на нужном расстоянии (урок 5.1).
6. Захват берёт, держит и отпускает груз (урок 5.2).

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Все умения проверены. То, что не работает, ты сначала чинишь по разделу «Если не получилось» своего урока.

4 Составь план маршрута

16 мин

Хороший инженер сначала планирует, потом пишет код. Самое сложное — повороты на обратном пути: «налево» и «направо» считаются от носа робота, а не от тебя.

Участок	Что делает робот	Какая функция
Старт → линия	Едет прямо до линии	`driveStraight()`
…		

1. Заполни таблицу плана в бортовом журнале. Одна строка — один участок.
2. Для каждого перекрёстка запиши: откуда приехал, куда повернул, что делаешь дальше.
3. Проведи выключенного робота по плану руками. Куда смотрит его нос после каждого разворота?
4. Запиши, где скорее всего будет ошибка и как ты это проверишь.

В плане семь участков: выезд, линия до перекрёстка 1, груз 1, место 1, перекрёсток 2, груз 2 и место 2, финиш.

Подсказка: Робот развернулся — а что теперь «налево»?

Встань и повернись так же, как робот. Твоя левая рука показывает его «налево».

ДОЛЖНО ПОЛУЧИТЬСЯ

План записан. Напарник прошёл по нему с выключенным роботом и не нашёл ошибок.

2

ЗАНЯТИЕ 2 · 45 МИН

Программа по частям

5 Подготовь заготовку

6 мин

Программу собираем из функций. Функция — действие с понятным именем, например `turnLeft()`. Когда маршрут записан вызовами функций, его легко сверить с планом.

Заготовка с функциями L61_Courier.ino

```
// Робот-курьер: моя программа итоговой задачи.
// Это заготовка, а не решение. Наполни функции своим проверенным кодо
// из прошлых уроков и проверяй каждую отдельно, прежде чем собирать м

// --- Моторы (урок 2.1) ---
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;

// --- Датчики линии (уроки 3.1 и 3.2) ---
const int LEFT_SENSOR_PIN = A4;
const int RIGHT_SENSOR_PIN = A5;
// Перенеси сюда свои измерения белого и чёрного и коэффициент П-регул

// --- Ультразвук (урок 5.1) и захват (урок 5.2) ---
// Перенеси сюда контакты и настройки из своих проверенных скетчей.

void stopMotors() {
  // Остановить оба мотора (урок 2.1).
}

void driveStraight() {
  // Проехать прямо проверенное время (урок 2.1), например въезд в зон
}

void turnLeft() {
  // Повернуть налево на 90° (урок 2.2) или с перекрёстка до линии (ур
}

void turnRight() {
  // Повернуть направо.
}

void turnAround() {
  // Развернуться на 180° (урок 2.2).
}

```

```
void followLineToCrossing() {
  // Ехать по линии с П-регулятором до перекрёстка (уроки 3.2, 4.1).
  // После перекрёстка сначала съедь с широкой чёрной области.
}

void approachCan() {
  // Ехать по линии и остановиться перед грузом по ультразвуку (урок 5
}

void pickUp() {
  // Остановить моторы и закрыть захват (урок 5.2).
}

void putDown() {
  // Открыть захват и отпустить груз (урок 5.2).
}

void setup() {
  // Подготовь выводы, датчики и захват. Убедись, что моторы стоят.

  // Здесь — твой маршрут: вызывай функции в порядке своего плана.

  stopMotors();
}

void loop() {
  // Задание выполняется один раз в setup(). После финиша робот стоит
}
```

1. Открой заготовку и сохрани её под своим именем.
2. Перенеси в функции свой проверенный код из прошлых уроков.

Как это устроено: Зачем делить программу на функции

Если поворот работает неправильно, исправлять нужно одно место — саму функцию, а не каждый поворот на маршруте. А маршрут в `setup()` читается почти как план из журнала.

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · МАРШРУТ МАССИВОМ**

Запиши решения на перекрёстках массивом, например `char route[] = {'L', 'S', 'R'};`, и выполняй их по очереди в цикле, как в уроке 4.2. Тогда для новой карточки достаточно поменять одну строку.

6 Часть 1: от старта до первого груза

12 мин

1. В `setup()` вызови функции первого участка: выезд, линия, перекрёсток, поворот, подъезд к грузу.
2. Проверь на поле только эту часть.
3. Работает — сохрани копию: «рабочая версия 1».

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Робот выезжает со старта, доезжает по линии до груза 1 и останавливается перед ним.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что проверить
Робот не находит линию после старта	Смотрит ли нос робота вдоль линии? Хватает ли длины выезда?
Робот останавливается на дуге	Дуга — не перекрёсток. Проверь условие поиска перекрёстка.
Робот проезжает перекрёсток	Скорость велика или условие перекрёстка не срабатывает.
После поворота робот теряет линию	Проверь, что поворот заканчивается, когда датчик нашёл новую линию.

7 Часть 2: первая доставка

12 мин

1. Допиши: захват, разворот, путь к месту 1, отпускание, возвращение к дороге.
2. Проверь только эту часть: поставь робота перед грузом 1 и запусти.
3. Работает — сохрани «рабочую версию 2».

Подсказка: Когда закрывать захват?

Сначала останови моторы, потом закрывай захват. Подъезд заканчивай по измерению ультразвука. Если эха нет, это не значит «груз совсем рядом».

Подсказка: Груз уезжает вместе с роботом

Отъезжай только после того, как захват полностью открылся.

ДОЛЖНО ПОЛУЧИТЬСЯ

Груз 1 стоит в месте 1, захват открыт, робот на дороге и смотрит туда, куда нужно дальше.

8 Часть 3: вторая доставка и финиш

15 мин

1. Допиши вторую доставку и путь к финишу. Груз 2 стоит в другом месте, поэтому повороты будут другими — сверься с планом.
2. В конце — прямой въезд в финиш и остановка. После финиша робот должен стоять.
3. Запусти весь маршрут от старта. Сохрани «рабочую версию 3».

Подсказка: Как понять, что пора въезжать в финиш

Чёрной метки финиша на поле нет. Используй последний ориентир — конец дуги или последний перекрёсток — и проверенное время прямого движения. Два «белых» показания ещё не значат, что линия кончилась: она может быть между датчиками.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот проходит весь маршрут хотя бы в одной пробе. Все рабочие версии сохранены.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу показать в программе каждый участок своего плана.
- [ ] Я знаю, какая часть работает ненадёжно и что я проверю в следующем уроке.

Критерии оценки

Зачётные заезды пройдут в уроке 6.2 по этим критериям.

Как оценивается заезд

Максимум **100** | Зачёт от **70**

Критерий	Баллы	Макс.
Прямой выезд со старта	5 — Сам выехал вперёд из зоны старта. 10 — Доехал до линии и продолжил задание без подталкивания.	**10**
Движение по линии	10 — Терял линию, но сам возвращался на неё и продолжал маршрут. 20 — Прошёл все участки линии и дуги по датчикам, не теряя линию.	**20**
Выбор дороги на перекрёстках	5 — За каждое из четырёх решений на перекрёстках, которые совпали с планом.	**20**
Груз 1: обнаружение	5 — Остановился перед грузом по ультразвуку.	**5**
Груз 1: захват	5 — Взял груз захватом и увёз, а не вытолкнул.	**5**
Груз 1: доставка	5 — Груз стоит и частично находится в нужной зоне. 10 — Груз стоит целиком в нужной зоне, захват отпущен.	**10**
Груз 2: обнаружение	5 — Остановился перед грузом по ультразвуку.	**5**
Груз 2: захват	5 — Взял груз захватом и увёз, а не вытолкнул.	**5**
Груз 2: доставка	5 — Груз стоит и частично находится в нужной зоне. 10 — Груз стоит целиком в нужной зоне, захват отпущен.	**10**
Прямой въезд и остановка	5 — После линии сам въехал вперёд в зону финиша. 10 — Остановился внутри зоны и стоит с выключенными моторами не меньше 3	**10**

Критерий	Баллы	Макс.
	секунд.	

Обязательные условия зачёта

- Команда объясняет свою программу: показывает прямое движение, расчёт П-регулятора, остановку по ультразвуку, управление захватом и строку, которая завершает поездку.
- Изгибы пройдены по показаниям датчиков линии с П-регулятором.
- Подъезд к грузу заканчивается по измерению ультразвука, а не только по времени.
- После старта робота никто не трогает и не отправляет ему команд.

Правила

- Скорость не оценивается: нет бонуса за быстрый заезд.
- Отрицательных штрафов нет. Невыполненное действие просто не получает баллов.
- Груз в чужой зоне или доставленная не по порядку — 0 за её доставку.
- После помощи руками автономный заезд окончен. Продолжать можно для тренировки, но баллы не добавляются.
- Каждая попытка оценивается отдельно. Лучшие части разных попыток не складываются.
- Аварийная остановка ради безопасности не штрафуется.

Проверь себя

● Основное

1. **Робот ехал на восток и развернулся на 180°. Теперь ему нужно повернуть на север. Куда он поворачивает?**

- ○ Налево
- ○ Направо
- ○ Прямо

2. **Робот нашёл перекрёсток и повернул. Почему перед поиском следующего перекрёстка нужно сначала съехать с чёрной области?**

- ○ Иначе программа сразу найдёт тот же самый перекрёсток ещё раз
- ○ Иначе сядет батарея
- ○ Так требует П-регулятор

3. **Поворот налево оформлен функцией turnLeft() и вызывается четыре раза. Поворот работает неправильно. Сколько мест нужно исправить?**

- ○ Одно — саму функцию
- ○ Четыре — каждый вызов
- ○ Пять

4. **Зачем собирать программу по частям и сохранять копию после каждой работающей части?**

- ○ Если появится ошибка, понятно, в какой части её искать, и можно вернуться к рабочей версии
- ○ Так программа быстрее загружается
- ○ Так требует Arduino

5. **На среднем поле робот проезжает перекрёсток B несколько раз. Как программа понимает, что делать на нём в этот раз?**

- ○ Читает букву на поле
- ○ Знает, какое по счёту решение маршрута сейчас выполняет
- ○ Всегда поворачивает налево

★ **Средний уровень**

1. **На среднем поле одни и те же перекрёстки встречаются несколько раз. Что поможет не перепутать решения?**

- ○ Записать маршрут списком решений по порядку и сверить его со схемой
- ○ Всегда поворачивать налево
- ○ Проезжать все перекрёстки без решений

2. **У строки плана «проехать перекрёсток Б прямо» нет своей функции. Что сделать до сборки маршрута?**

- ○ Найти или дописать функцию для этого действия и проверить её отдельно
- ○ Пропустить строку плана
- ○ Убрать перекрёсток из плана

3. **Часть 1 сработала один раз из трёх. Можно ли переходить к части 2?**

- ○ Да: часть 2 исправит часть 1
- ○ Нет: сначала найти, что срывается, и добиться трёх из трёх
- ○ Да: один раз уже достаточно

4. **Весь маршрут сорвался в части 3. Как быстрее проверить исправление?**

- ○ Каждый раз запускать маршрут от старта
- ○ Переписать часть 1
- ○ Поставить робота в начало части 3 и запустить только её

5. Почему маршрут записывают в setup(), а не в loop()?

- ○ После финиша программа не должна начинать маршрут заново
- ○ loop() не умеет вызывать функции
- ○ setup() выполняется быстрее

★★ Продвинутый уровень

1. Карточка маршрута: налево, направо, прямо. Какая запись верна?

- ○ char route[] = {'L', 'R', 'S'};
- ○ char route[] = {'L', 'S', 'R'};
- ○ char route[] = {'R', 'L', 'S'};

2. Для новой карточки пришлось менять функцию turnLeftAtCrossing(). Что это говорит о программе?

- ○ Всё в порядке: так и должно быть
- ○ Маршрут не отделён от действий: данные и функции перемешаны
- ○ Карточка составлена неверно

3. **Функция подъезда ждёт груз, а груз забыли поставить. Что должно уберечь робота?**

- ○ Ничего: робот сам остановится
- ○ П-регулятор
- ○ Защита от бесконечного ожидания: остановка по времени или в конце линии

4. **Часть 1 проходит, если робот стоит на старте ровно, и срывается, если он на 2 см левее. Где слабое место?**

- ○ Выезд со старта и поиск линии
- ○ Захват
- ○ Ультразвук

5. Маршрут по карточке из четырёх решений обрывается раньше. Почему?

```
char route[] = {'L', 'S', 'R', 'L'};
for (int i = 0; i < 3; i++) {
  followLineToCrossing();
  decide(route[i]);
}
```

- ○ decide не понимает 'S'
- ○ Цикл выполняет только три решения: последнее не выполнится
- ○ Элемента route[3] не существует

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

У тебя есть план и программа, которая проходит маршрут. В уроке 6.2 ты сделаешь её надёжной и проведёшь зачётные заезды.

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

функция	Именованный блок команд, который можно вызывать много раз: например, turnLeft().
план маршрута	Таблица участков пути: что делает робот и какая функция за это отвечает.
рабочая версия	Копия программы, в которой всё написанное уже проверено на поле.

Тема 6 · Урок 6.2

Робот-курьер: зачётные заезды

Итоговая задача, часть 2: делаешь программу надёжной, проводишь зачётные заезды и разбираешь результат.

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

РЕЗУЛЬТАТ УРОКА

Проведены две зачётные попытки, каждая оценена по листу. Команда объясняет свою программу и рассказывает, какую ошибку нашла и как исправила.

Ты научишься

- Находить причину ошибки по наблюдению и менять за раз одну вещь.
- Проверять, что программа работает при немного других условиях старта.
- Провести автономный заезд по правилам и оценить его по критериям.
- Объяснить устройство своей программы и следующий шаг улучшения.

Нужно знать

- 6.1 Робот-курьер: план и программа

Понадобится

Ультразвуковой датчик расстояния | Захват с сервоприводом | Датчик линии (аналоговый, Trema) × 2

Зачем этот урок

Программа, которая прошла маршрут один раз, — ещё не готовая программа. Сегодня ты сделаешь её надёжной, проведёшь две зачётные попытки и разберёшь результат, как настоящий инженер.

РАЗМИНКА · ВСПОМНИ

1. Какой участок твоего плана в прошлый раз работал ненадёжно?
2. Почему после помощи руками в зачётной попытке баллы продолжения не засчитываются?

ЗАНЯТИЕ 1 · 45 МИН

1 Отладка и надёжность

1 Пробный заезд целиком

8 мин

Что должен делать робот. Задача робота — повторяемо выполнить доставку по своему плану. Задача команды — доказать это наблюдениями и объяснить, почему программа выбирает каждое действие. Используем готового робота, поле, журнал и свою проверенную программу.

Что используем и как это работает. Во время заезда работает цепочка: датчик даёт сигнал, программа проверяет условие, исполнитель выполняет команду, новое измерение показывает изменение. Ошибка может возникнуть на любом звене. Отладка начинается с первого места, где наблюдение расходится с прогнозом: например, датчик увидел полосу, но условие её не распознало.

Как связать измерение с действием. Записывай ожидаемое действие и фактический результат, меняй одну причину за раз и повторяй тот же участок. Разовый успех не доказывает повторяемость. Успешная сборка доказывает допустимость программы для компилятора, а доставка и остановка подтверждаются только наблюдением самого заезда.

1. Поставь грузы на места, робота — в зону старта.
2. Запусти весь маршрут и не вмешивайся.
3. Запиши в журнал, на каком участке плана робот впервые сделал не то, что нужно.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты знаешь, какой участок работает ненадёжно, — или что весь маршрут прошёл.

2 Найди и исправь ошибку

25 мин

Отладка идёт по кругу из четырёх шагов. Главное правило: **меняй одну вещь за раз**, иначе не поймёшь, что помогло.

Шаг	Что делаешь	Пример
Наблюдаю	Описываю, что робот сделал	Робот проехал мимо груза 2
Причина	Предлагаю одну возможную причину	Не досчитал перекрёсток: проехал его, не заметив
Одно изменение	Меняю одну вещь	Снижаю скорость перед перекрёстком
Проверка	Запускаю только этот участок	Робот останавливается на перекрёстке 3 раза из 3

1. Заполни такую таблицу в журнале для своей ошибки.
2. Проверяй исправление на своём участке, а не всем маршрутом: так быстрее.
3. Когда участок работает три раза подряд, проверь весь маршрут и сохрани новую рабочую версию.

Подсказка: Не понимаю, что делает программа

Временно добавь вывод в монитор порта: какое решение маршрута выполняется и что видят датчики. Перед зачётом этот вывод можно оставить — он не мешает.

ДОЛЖНО ПОЛУЧИТЬСЯ

Ошибка найдена и исправлена, в журнале записан путь к причине.

3 Проверь надёжность

12 мин

На зачёте условия будут чуть другими: робот поставлен на старт не идеально, батарея немного разряжена. Проверь заранее.

1. Поставь робота на старт чуть левее, потом чуть правее. Маршрут проходит?
2. Если робот стал ехать медленнее, чем на пробах, попроси преподавателя проверить батарею. После замены или зарядки заново подбери время прямых участков и поворотов.
3. Проведи весь маршрут два раза подряд без исправлений между запусками.

Два полных маршрута подряд с разных положений на старте.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа проходит маршрут два раза подряд. Рабочая версия для зачёта сохранена.

ЗАНЯТИЕ 2 · 45 МИН

2 Зачёт и разбор

4 Зачётная попытка 1

10 мин

1. Расставь грузы на исходные места по схеме своего поля.
2. Перед стартом договоритесь, за какими четырьмя решениями на перекрёстках следить.
3. Один запуск. Дальше робот едет сам: нельзя трогать его, поправлять грузы или отправлять команды.
4. Если нужна помощь, останови робота. Безопасность важнее баллов.

ДОЛЖНО ПОЛУЧИТЬСЯ

Попытка записана в лист оценки.

5 Исправь, если нужно

8 мин

Между попытками можно исправлять программу. Используй круг отладки: наблюдение → причина → одно изменение → проверка участка.

ДОЛЖНО ПОЛУЧИТЬСЯ

Исправление проверено на участке и сохранено — или ты оставил программу без изменений.

6 Зачётная попытка 2

10 мин

Проведи вторую попытку по тем же правилам. Каждая попытка оценивается отдельно: лучшие части разных попыток не складываются.

ДОЛЖНО ПОЛУЧИТЬСЯ

Обе попытки записаны в лист оценки.

7 Разбор результата

12 мин

1. Сначала оцени свою попытку по листу оценки сам, потом сверь оценку с преподавателем.
2. Покажи в своей программе: прямое движение, расчёт П-регулятора, остановку по ультразвуку, управление захватом и строку, которая завершает поездку.
3. Расскажи об одной ошибке, которую ты нашёл и исправил. Как ты понял, в чём причина?
4. Запиши в журнал: какое умение у твоего робота работает лучше всего, а какое ты бы улучшил дальше.
5. Сохрани итоговый скетч и лист оценки.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, как устроена моя программа, по своему плану.
- [ ] Я знаю, какое умение нужно улучшить и как я бы это сделал.

Критерии оценки

Как оценивается заезд

Максимум **100** | Зачёт от **70**

Критерий	Баллы	Макс.
Прямой выезд со старта	5 — Сам выехал вперёд из зоны старта. 10 — Доехал до линии и продолжил задание без подталкивания.	**10**
Движение по линии	10 — Терял линию, но сам возвращался на неё и продолжал маршрут. 20 — Прошёл все участки линии и дуги по датчикам, не теряя линию.	**20**
Выбор дороги на перекрёстках	5 — За каждое из четырёх решений на перекрёстках, которые совпали с планом.	**20**
Груз 1: обнаружение	5 — Остановился перед грузом по ультразвуку.	**5**
Груз 1: захват	5 — Взял груз захватом и увёз, а не вытолкнул.	**5**
Груз 1: доставка	5 — Груз стоит и частично находится в нужной зоне. 10 — Груз стоит целиком в нужной зоне, захват отпущен.	**10**
Груз 2: обнаружение	5 — Остановился перед грузом по ультразвуку.	**5**
Груз 2: захват	5 — Взял груз захватом и увёз, а не вытолкнул.	**5**
Груз 2: доставка	5 — Груз стоит и частично находится в нужной зоне. 10 — Груз стоит целиком в нужной зоне, захват отпущен.	**10**
Прямой въезд и остановка	5 — После линии сам въехал вперёд в зону финиша. 10 — Остановился внутри зоны и стоит с выключенными моторами не меньше 3	**10**

Критерий	Баллы	Макс.
	секунд.	

Обязательные условия зачёта

- Команда объясняет свою программу: показывает прямое движение, расчёт П-регулятора, остановку по ультразвуку, управление захватом и строку, которая завершает поездку.
- Изгибы пройдены по показаниям датчиков линии с П-регулятором.
- Подъезд к грузу заканчивается по измерению ультразвука, а не только по времени.
- После старта робота никто не трогает и не отправляет ему команд.

Правила

- Скорость не оценивается: нет бонуса за быстрый заезд.
- Отрицательных штрафов нет. Невыполненное действие просто не получает баллов.
- Груз в чужой зоне или доставленная не по порядку — 0 за её доставку.
- После помощи руками автономный заезд окончен. Продолжать можно для тренировки, но баллы не добавляются.
- Каждая попытка оценивается отдельно. Лучшие части разных попыток не складываются.
- Аварийная остановка ради безопасности не штрафуется.

Проверь себя

● Основное

1. Оба датчика линии показывают «белое». Что это значит?

- ○ Линия точно закончилась
- ○ Линия может быть между датчиками или уже сбоку — одних этих показаний мало
- ○ Датчики сломались

2. Ультразвуковой датчик не получил эхо. Как поступить программе?

- ○ Считать, что груз вплотную, и закрыть захват
- ○ Считать измерение неудачным и не закрывать захват
- ○ Ехать дальше с максимальной скоростью

3. Робот проехал мимо груза. Что делаешь первым?

- ○ Сразу меняю скорость, коэффициент и расстояние остановки
- ○ Смотрю, что происходило, предлагаю одну причину и меняю одну вещь
- ○ Переписываю программу заново

4. **Во время зачётной попытки робот сбился, и ты поставил его обратно на линию. Что с попыткой?**

- ○ Продолжается со штрафом
- ○ Автономная попытка окончена: продолжать можно только для тренировки
- ○ Начинается заново без последствий

5. **После финиша робот снова поехал. В чём вероятная причина?**

- ○ Маршрут записан в loop() и повторяется
- ○ Села батарея
- ○ Датчики не откалиброваны

★ **Средний уровень**

1. **На среднем поле робот выбрал на перекрёстке не то решение. Где искать причину сначала?**

- ○ Сверить маршрут в программе со схемой и порядком перекрёстков
- ○ Перекалибровать ультразвуковой датчик
- ○ Заменить захват

2. Зачем перед зачётной попыткой делать тренировочный заезд?

- ○ Проверить робота на этом поле с текущим зарядом батареи
- ○ Тренировочный заезд идёт в зачёт
- ○ Без него робот не включится

3. Робот проехал мимо груза 2. Какая запись в круге отладки — «причина», а не «наблюдение»?

- ○ Робот проехал мимо груза 2
- ○ Робот остановился у финиша
- ○ Робот не заметил перекрёсток и не досчитал его

4. Не знаешь, на каком участке ошибка. Как найти его за 2–3 запуска?

- ○ Запустить первую половину маршрута, потом половину оставшейся части
- ○ Запустить весь маршрут три раза
- ○ Поменять всё сразу и посмотреть

5. Почему между попытками меняют только одну вещь?

- ○ Две вещи нельзя загрузить в Arduino за раз
- ○ Иначе непонятно, что помогло, а что навредило
- ○ Так требует лист оценки

★★ Продвинутый уровень

1. На крутой S-дуге робот теряет линию только со свежезаряженной батареей. Что вероятнее всего?

- ○ Робот едет быстрее, и подобранного kp уже мало: уменьшить скорость или подобрать kp заново
- ○ Сломался датчик линии
- ○ Неверный маршрут

2. Что записать в журнал после неудачного заезда? Выбери все верные ответы.

- ☐ Где именно робот сбился
- ☐ Какую одну вещь ты изменишь
- ☐ Только итоговые баллы

3. **Гипотеза: «робот пропускает перекрёсток B из-за скорости». Какой опыт может её опровергнуть?**

- ○ Запустить весь маршрут ещё раз
- ○ Поменять сразу порог и скорость
- ○ Снизить скорость только перед B и проверить пять раз: если пропуски остались, гипотеза неверна

4. **Диагностика печатает номер решения на каждом перекрёстке. Робот повернул не там, а последним напечатано «решение 4», хотя по плану это перекрёсток 5. Что это значит?**

- ○ Один перекрёсток робот не заметил, и решения сдвинулись на одно
- ○ Решение 4 записано в программе неверно
- ○ Перепутаны левый и правый датчики

5. **В отчёте: «участок 3 прошёл 4 раза из 10». Какой вывод верен?**

- ○ Участок 3 надо убрать из маршрута
- ○ Участок 3 ненадёжен — улучшать его первым
- ○ Участок надёжен: он же проходил

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

Ты прошёл весь путь инженера: разобрал задачу, составил план, собрал программу из проверенных частей, нашёл и исправил ошибки и проверил результат на поле. Если зачёт сдан, попробуй следующий уровень или факультативные уроки — энкодеры и ПИД-регулятор.

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

отладка	Поиск и исправление ошибки: наблюдаем, находим причину, меняем одну вещь и проверяем снова.
автономный заезд	Заезд, в котором робот после старта едет сам: никто его не трогает и не отправляет команд.
надёжность	Программа работает не один раз случайно, а каждый раз — даже если робот поставлен на старт чуть иначе.

Тема 7 · Урок Ф.1

Энкодеры: считаем обороты и точное расстояние

Робот считает обороты колёс и проезжает отмеренное расстояние, а не «сколько успеет за время».

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

РЕЗУЛЬТАТ УРОКА

Программа считает обороты колёс энкодерами и проезжает отмеренное расстояние. Число импульсов на оборот измерено, точность проверена линейкой. Не нужен для итоговой задачи.

Ты научишься

- Объяснить, как энкодер превращает оборот колеса в импульсы.
- Измерить число импульсов на оборот колеса.
- Пересчитать расстояние в импульсы через длину окружности колеса.
- Проехать отмеренное расстояние и оценить точность линейкой.

Нужно знать

- 2.1 Моторы: едем и останавливаемся
- 3.2 П-регулятор: плавно и быстро

Понадобится

Колесо × 2 | Мотор-редуктор × 2 | USB-кабель для Arduino

Зачем этот урок

В уроке 2.1 робот ехал «по времени»: моторы включались на две секунды. Но сколько он проедет за две секунды, зависит от заряда батарей, пола и веса. **Энкодер** считает обороты колеса, поэтому программа может выбирать момент остановки по вращению колёс, а не только по времени. Точность пути всё равно нужно проверить линейкой.

Для урока нужен дополнительный модуль: энкодер на каждое колесо. В основной комплект курса он не входит; без него урок не провести. Факультатив не нужен для итоговой задачи.

РАЗМИНКА · ВСПОМНИ

1. Как в уроке 2.1 робот проезжал заданный участок?
2. Почему при тех же двух секундах робот с севшими батареями проезжает меньше?
3. Что такое П-регулятор из урока 3.2 в одной фразе?

ЗАНЯТИЕ 1 · 45 МИН

1 Считаем импульсы

1 Почему время — плохая линейка

8 мин

Что должен делать робот. Роботу нужно проехать заданное расстояние точнее, чем по секундомеру. Для этого понадобятся дополнительные энкодеры: в базовом комплекте их нет. Они измеряют вращение колёс, а не расстояние до препятствия.

Что используем и как это работает. Энкодер выдаёт импульсы при вращении. Счётчик накапливает их; число импульсов за оборот связывает счёт с оборотами колеса. Длина окружности связывает обороты с предполагаемым путём. Прерывание позволяет быстро увеличивать счётчик при импульсе, даже когда основная программа занята другой работой.

Как связать измерение с действием. Измерим импульсы на оборот и используем диаметр своего колеса. На AVR чтение многобайтного счётчика прерывание может разорвать посередине, поэтому копируем его при кратко выключенных прерываниях. Проверим результат линейкой: при проскальзывании колесо вращается, а корпус проходит меньше, поэтому энкодер не даёт абсолютной координаты.

1. Вспомни программу из урока 2.1, которая едет вперёд по времени. Запусти её дважды: на полной скорости и на скорости вдвое меньше. Время одинаковое.
2. Отметь, где робот остановился оба раза. Измерь линейкой оба пути и запиши в журнал.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты видишь: при одном и том же времени робот проезжает разное расстояние.

2 Как устроен энкодер

12 мин

Энкодер — это диск с прорезями на оси колеса и датчик-«вилка». Внутри вилки с одной стороны светит светодиод, с другой — приёмник. Когда мимо проходит прорезь, свет попадает на приёмник, и датчик посылает **импульс**. Один оборот колеса — столько импульсов, сколько прорезей на диске.

1. Посчитай прорези на диске энкодера. Запиши в журнал.
2. Найди в таблице проводки своего робота, к каким контактам подключены энкодеры. Запиши номера.

⚠ **БЕЗОПАСНОСТЬ**

Не подключай энкодеры при включённом питании. Если в таблице проводки контакты энкодеров не указаны — подключение делает преподаватель.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — число прорезей и номера контактов левого и правого энкодеров.

ⓘ **Как это устроено: Что такое прерывание**

Импульсы приходят быстро — десятки в секунду. Если программа занята чем-то ещё, она может их пропустить. Поэтому счёт делает **прерывание**: на каждый импульс Arduino сама откладывает основную программу, прибавляет 1 к счётчику и возвращается обратно. На Arduino Uno так умеют только контакты 2 и 3.

3 Импульсов на один оборот

22 мин

1. Создай скетч `LF1_Count`. Впиши номера контактов энкодеров.

Счёт импульсов LF1_Count.ino

```ino
// Контакты энкодеров: впиши номера из таблицы проводки своего робота
// На Arduino Uno счёт по прерыванию работает только на контактах 2 и
const int LEFT_ENCODER_PIN = -1;
const int RIGHT_ENCODER_PIN = -1;

// Счётчики импульсов. volatile: их меняет прерывание.
volatile long leftTicks = 0;
volatile long rightTicks = 0;

// Arduino вызывает её сама на каждый импульс левого энкодера.
void countLeft() {
  leftTicks++;
}

// То же для правого энкодера.
void countRight() {
  rightTicks++;
}

void setupEncoders() {
  // Без контактов энкодеров программа дальше не пойдёт.
  if (LEFT_ENCODER_PIN < 0 || RIGHT_ENCODER_PIN < 0) {
    Serial.println("Впиши контакты энкодеров");
    while (true) {
    }
  }
  pinMode(LEFT_ENCODER_PIN, INPUT);
  pinMode(RIGHT_ENCODER_PIN, INPUT);
  attachInterrupt(digitalPinToInterrupt(LEFT_ENCODER_PIN), countLeft,
  attachInterrupt(digitalPinToInterrupt(RIGHT_ENCODER_PIN), countRigh
}

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

void loop() {
  // Читаем оба счётчика, пока прерывания выключены: число не изменит
```

```
  noInterrupts();
  long left = leftTicks;
  long right = rightTicks;
  interrupts();

  Serial.print("Левое: ");
  Serial.print(left);
  Serial.print("  правое: ");
  Serial.println(right);
  delay(200);
}
```

2. Загрузи и открой монитор порта. Поверни колесо рукой: число должно расти.
3. Сделай мелом или маркером метку на колесе. Прокрути колесо рукой **ровно 10 оборотов** по метке. Раздели число импульсов на 10 — это импульсов на оборот.
4. Повтори для второго колеса. Запиши оба числа в журнал.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — импульсов на оборот для каждого колеса. Обычно это число прорезей на диске.

🔧 ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
«Впиши контакты энкодеров»	Впиши номера контактов вместо −1
Число не растёт	Проверь провода и что диск проходит через щель датчика
Число растёт, когда колесо стоит	Колесо остановилось на краю прорези: сдвинь его чуть-чуть

2

ЗАНЯТИЕ 2 · 45 МИН

Точное расстояние

4 Миллиметры в импульсы

15 мин

За один оборот колесо проезжает длину окружности: `π × диаметр`. Диаметр колеса — 45 мм. Чтобы проехать 500 мм, нужно `500 / (π × 45)` оборотов, а импульсов — в `TICKS_PER_REV` раз больше.

1. Посчитай на бумаге: сколько импульсов на 500 мм, если на оборот 20 импульсов?
2. Создай скетч `LF1_Distance`, впиши контакты и своё число импульсов на оборот.

Точное расстояние LF1_Distance.ino

```
// Контакты энкодеров: впиши номера из таблицы проводки своего робота
// На Arduino Uno счёт по прерыванию работает только на контактах 2 и
const int LEFT_ENCODER_PIN = -1;
const int RIGHT_ENCODER_PIN = -1;

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;

const int TICKS_PER_REV = 0; // импульсов на оборот колеса: измерь и
const int WHEEL_DIAMETER_MM = 45;
const int DISTANCE_MM = 500; // сколько проехать
int speed = 100;

volatile long leftTicks = 0;
volatile long rightTicks = 0;

void countLeft() {
  leftTicks++;
}

void countRight() {
  rightTicks++;
}

// Сколько импульсов нужно, чтобы проехать mm миллиметров.
long ticksFor(int mm, int ticksPerRev, int diameterMm) {
  float revolutions = mm / (PI * diameterMm);
  return (long)(revolutions * ticksPerRev + 0.5);
}

// Сколько импульсов в среднем насчитали оба колеса.
long averageTicks() {
  noInterrupts();
  long sum = leftTicks + rightTicks;
  interrupts();
  return sum / 2;
```

```
}

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 setupEncoders() {
  // Без контактов энкодеров программа дальше не пойдёт.
  if (LEFT_ENCODER_PIN < 0 || RIGHT_ENCODER_PIN < 0) {
    Serial.println("Впиши контакты энкодеров");
    while (true) {
    }
  }
  pinMode(LEFT_ENCODER_PIN, INPUT);
  pinMode(RIGHT_ENCODER_PIN, INPUT);
  attachInterrupt(digitalPinToInterrupt(LEFT_ENCODER_PIN), countLeft,
  attachInterrupt(digitalPinToInterrupt(RIGHT_ENCODER_PIN), countRigh
}

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

  // Без числа импульсов на оборот робот не поедет.
  if (TICKS_PER_REV <= 0) {
    Serial.println("Впиши число импульсов на оборот");
    while (true) {
    }
  }
  delay(2000); // время убрать руки от робота

  long target = ticksFor(DISTANCE_MM, TICKS_PER_REV, WHEEL_DIAMETER_MM
  drive(FORWARD, speed, FORWARD, speed);
```

```
  while (averageTicks() < target) {
  }
  drive(FORWARD, 0, FORWARD, 0);
  delay(1000); // робот докатывается

  Serial.print("Цель, импульсов: ");
  Serial.println(target);
  Serial.print("Насчитано после остановки: ");
  Serial.println(averageTicks());
}

void loop() {
}
```

3. Поставь робота на пол у метки старта, включи питание моторов. Робот проедет и остановится. Измерь путь линейкой от метки до той же точки корпуса.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты посчитал: около 71 импульса. Робот проехал примерно 50 см, путь измерен.

5 Почему робот проезжает дальше

15 мин

После команды «стоп» робот ещё немного катится — как велосипед, если перестать крутить педали.

1. Сделай три заезда на 500 мм. Запиши пути в журнал.
2. Уменьши `speed` до 70 и сделай ещё три заезда. Стало точнее?

`speed`	Заезд 1	Заезд 2	Заезд 3
100			
70			

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена. Ты видишь, на какой скорости робот точнее.

6 Задания по уровням

12 мин

● **ОСНОВНОЕ · РОВНО 50 СМ**

Проехать 500 мм с ошибкой не больше 2 см в трёх заездах из трёх.

★ **СРЕДНИЙ УРОВЕНЬ · ПРЯМО НА МЕТР**

Проехать 1 м, выравнивая колёса по разнице импульсов: если левое насчитало больше — замедлить левое и ускорить правое, как П-регулятор из урока 3.2. Робот уходит от прямой не больше чем на 3 см.

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · РОВНО 90 ГРАДУСОВ**

Повернуть на месте на 90°: колёса крутятся в разные стороны, каждое проходит четверть окружности диаметром в колею робота — 90 мм. Посчитать импульсы, проверить угол угольником. Ошибка — не больше 10°.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, как энкодер считает обороты.
- [ ] Я могу перевести миллиметры в импульсы.
- [ ] Я могу объяснить, почему робот проезжает чуть дальше цели и как это уменьшить.

Проверь себя

● Основное

1. Что считает энкодер?

- ○ Импульсы: сколько прорезей диска прошло мимо датчика
- ○ Расстояние до препятствия
- ○ Заряд батареи

2. На диске 20 прорезей, на каждую — один импульс. Сколько импульсов за 3 оборота колеса?

- ○ 60
- ○ 23
- ○ 20

3. Какой путь проходит колесо диаметром 45 мм за один оборот?

- ○ Около 141 мм: π × 45
- ○ 45 мм
- ○ 90 мм

4. Почему езда «по времени» неточна?

- ○ Скорость зависит от заряда батарей и пола, а время — нет
- ○ Arduino неправильно считает время
- ○ Моторы всегда крутятся одинаково

5. Почему после остановки робот проезжает чуть больше цели?

- ○ Он ещё немного катится по инерции
- ○ Энкодер считает лишние прорези
- ○ Колесо становится больше

★ Средний уровень

1. 20 импульсов на оборот, колесо 45 мм. Сколько импульсов нужно на 500 мм?

- ○ Около 71
- ○ Около 222
- ○ 11

2. **Зачем счётчик импульсов объявлен как volatile long?**

- ○ Его меняет прерывание, и программа должна каждый раз читать его заново
- ○ Чтобы он сохранялся после выключения питания
- ○ Чтобы счёт шёл быстрее

3. **За одно время левое колесо насчитало 300 импульсов, правое — 270. Куда уводит робота?**

- ○ Вправо: левое колесо проехало больше
- ○ Влево: левое колесо проехало больше
- ○ Робот едет прямо

4. **Как точнее измерить число импульсов на оборот?**

- ○ Прокрутить колесо 10 оборотов по метке и разделить счёт на 10
- ○ Прокрутить один оборот на глаз
- ○ Посчитать прорези и не проверять

5. Почему поправка «остановиться на 5 импульсов раньше» перестаёт работать на другой скорости?

- ○ Чем быстрее робот едет, тем дальше он катится после остановки
- ○ На другой скорости меняется диаметр колеса
- ○ На другой скорости энкодер не работает

★★ Продвинутый уровень

1. Робот поворачивает на месте на 90°, колея 90 мм. Какой путь проходит каждое колесо?

- ○ Около 71 мм: четверть окружности диаметром 90 мм
- ○ Около 141 мм
- ○ 90 мм

2. Функция верна на 500 мм, но для 2000 мм и 40 импульсов на оборот даёт неверное число. В чём ошибка?

```
long ticksFor(int mm, int ticksPerRev, int diameterMm) {
  int ticks = mm * ticksPerRev / (3.14 * diameterMm);
  return ticks;
}
```

- ○ 2000 × 40 = 80 000 не помещается в int Arduino Uno
- ○ Нельзя делить на 3.14
- ○ Функция должна возвращать int

3. Колесо стоит на краю прорези, а счётчик растёт. Что происходит?

- ○ Сигнал дрожит около порога, и датчик даёт лишние импульсы
- ○ Колесо незаметно крутится
- ○ Прерывание срабатывает само по таймеру

4. Почему цель сравнивают со средним двух счётчиков, а не с одним левым?

- ○ Колёса крутятся немного по-разному; среднее ближе к пути середины робота
- ○ Левый энкодер всегда врёт
- ○ Так программа работает быстрее

5. Счётчик long читают в loop() без noInterrupts(). Что может случиться?

- ○ Прерывание изменит число посередине чтения, и программа увидит смесь старого и нового значения
- ○ Ничего: Arduino читает long за один шаг
- ○ Энкодер перестанет считать

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

Энкодер превращает оборот колеса в импульсы, а импульсы — в пройденный путь. Робот едет не «сколько успеет», а сколько нужно, при любом заряде батарей.

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

энкодер	Датчик оборотов: диск с прорезями на оси колеса и датчик, который даёт импульс на каждую прорезь.
импульс	Короткий сигнал энкодера: прорезь прошла мимо датчика.
прерывание	Arduino откладывает основную программу и сразу выполняет короткую функцию, когда на контакт пришёл импульс.
volatile	Пометка переменной, которую меняет прерывание: программа каждый раз читает её заново.
длина окружности	Путь колеса за один оборот: π × диаметр.

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 рабо
#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 оста
const unsigned long STEP_MS = 10;  // шаг регулятора: 100 раз в секун

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 =====
```

```
81
82  const int MARKER = 2610; // признак того, что в памяти записан заезд
83  Result run;
84  int lastSide = 0;
85  unsigned long startTime;
86  unsigned long nextStep;
87
88  // Добавляет ошибку шага к итогам заезда.
89  void record(int error) {
90    run.steps++;
91    run.sumAbs += abs(error);
92    if (abs(error) > run.maxAbs) {
93      run.maxAbs = abs(error);
94    }
95    int side = 0;
96    if (error > 10) {
97      side = 1;
98    }
99    if (error < -10) {
100     side = -1;
101   }
102   if (side != 0 && lastSide != 0 && side != lastSide) {
103     run.crossings++;
104   }
105   if (side != 0) {
106     lastSide = side;
107   }
108 }
109
110 void printResult(Result r) {
111   const char *names[] = {"релейный", "П", "ПД", "ПИД"};
112   Serial.print("Регулятор: ");
113   Serial.print(names[r.mode]);
114   Serial.print("  скорость=");
115   Serial.print(r.baseSpeed);
116   Serial.print("  kp=");
117   Serial.print(r.kp);
118   Serial.print("  kd=");
119   Serial.print(r.kd);
120   Serial.print("  ki=");
121   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
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, LEF
    int rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE,
    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(RIGI
    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 пропорциона
  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, fl
#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 рабо
#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 оста
const unsigned long STEP_MS = 10;  // шаг регулятора: 100 раз в секун

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 =====
```

```
81
82  const int MARKER = 2610; // признак того, что в памяти записан заезд
83  Result run;
84  int lastSide = 0;
85  unsigned long startTime;
86  unsigned long nextStep;
87
88  // Добавляет ошибку шага к итогам заезда.
89  void record(int error) {
90    run.steps++;
91    run.sumAbs += abs(error);
92    if (abs(error) > run.maxAbs) {
93      run.maxAbs = abs(error);
94    }
95    int side = 0;
96    if (error > 10) {
97      side = 1;
98    }
99    if (error < -10) {
100     side = -1;
101   }
102   if (side != 0 && lastSide != 0 && side != lastSide) {
103     run.crossings++;
104   }
105   if (side != 0) {
106     lastSide = side;
107   }
108 }
109
110 void printResult(Result r) {
111   const char *names[] = {"релейный", "П", "ПД", "ПИД"};
112   Serial.print("Регулятор: ");
113   Serial.print(names[r.mode]);
114   Serial.print("  скорость=");
115   Serial.print(r.baseSpeed);
116   Serial.print("  kp=");
117   Serial.print(r.kp);
118   Serial.print("  kd=");
119   Serial.print(r.kd);
120   Serial.print("  ki=");
121   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
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, LEF
    int rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE,
    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(RIGI
    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 пропорциона
  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, fl
#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.

★ **СРЕДНИЙ УРОВЕНЬ · ПД ЛУЧШЕ П**

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

★★ **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

● 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

★ **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

★★ Продвинутый уровень

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

```
1 int controller(int mode, int error) {
2   float correction = kp * error;
3   if (mode >= 2) {
4     correction += kd * (error - lastError);
5   }
6   return correction;
7 }
```

- ○ 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.

Печатный лист команды

Бортовой журнал робота

Записывайте сюда всё, что измерили и решили. Эти значения понадобятся в следующих уроках.

Имя робота	
Команда	
Комплект	Arduino Uno + шилд моторов L298P

Выполненные уровни заданий

Отмечай, какие задания получились. По этим отметкам в уроке 6.1 выбирается уровень итоговой задачи.

Урок	● Основное	★	★★
2.2 Повороты			
3.1 Датчики линии			
3.2 П-регулятор			
4.1 Перекрёстки			
4.2 Маршрут			
5.1 Ультразвук			
5.2 Захват			

Урок 1.1 · Сборка

Дата
Что получилось
Что было трудно

Урок 2.1 · Моторы

Скорость `speed`, при которой оба колеса уверенно трогаются	Куда уводит робота при движении прямо

Урок 2.2 · Повороты

Поворот	Время, мс	Проверено три раза
90°		
180°		

Урок 3.1 · Датчики линии

Догадка в начале урока: как робот видит линию?

_____

Датчик	Подключён к	Белое	Чёрное
Левый	A4		
Правый	A5		

Что видят датчики	Левое колесо	Правое колесо	Куда едет робот
Оба белое	вперёд	вперёд	прямо
Левый — линия			
Правый — линия			
Оба — линия			

Скорость `speed`	Прошёл изгиб?	Что заметили

Урок 3.2 · П-регулятор

`kp`	`baseSpeed`	Что видно

Выбрано: `kp` = ____ , `baseSpeed` = ____

Урок 4.1–4.2 · Перекрёстки и маршрут

Порог перекрёстка	`centerMs`	`leaveMs`

Урок 5.1–5.2 · Ультразвук и захват

TRIG	ECHO	`STOP_CM`	Сервопривод	Угол «открыто»	Угол «закрыто»

Урок 6.1 · План итоговой задачи

Уровень поля: начальный / средний / продвинутый. Карточка (для продвинутого): ____

Участок	Что делает робот	Какая функция

Где скорее всего будет ошибка и как я это проверю:

_____

Урок 6.2 · Отладка и зачёт

Наблюдаю	Причина	Одно изменение	Проверка

Попытка	Баллы	Что исправили после неё
1		
2		

Что у робота работает лучше всего и что я улучшу дальше:

_____