Материалы ученика

Робот-курьер

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

Уровень итоговой задачи: **Средний**

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

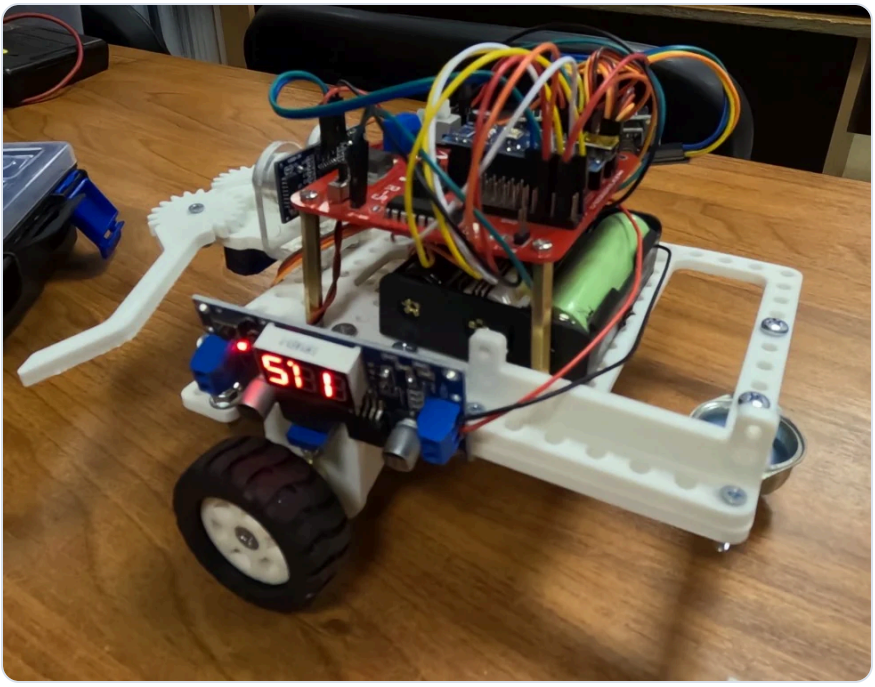
Ничего — это первый урок.

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.

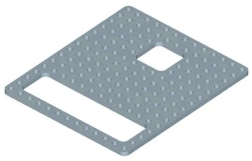
✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

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

2 Разложи детали

7 мин

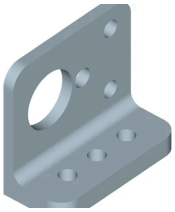
Прежде чем собирать, найди каждую деталь. Так ты не перепутаешь похожие детали и сразу увидишь, если чего-то не хватает.



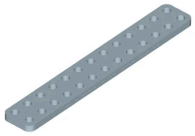
Основа 130 × 150
1 шт.



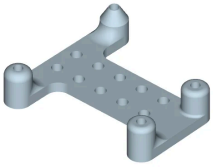
Кронштейн мотора
1 шт.



Кронштейн мотора (зеркальный)
1 шт.



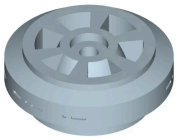
Планка датчиков
1 шт.



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

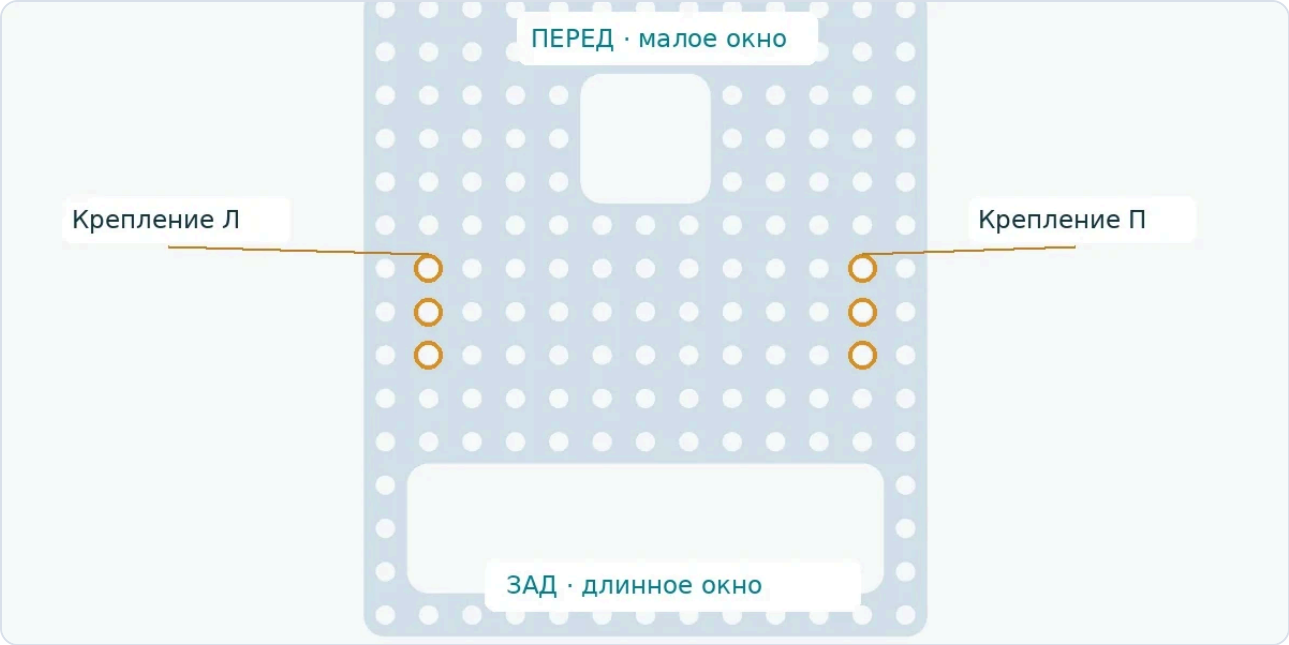
Не заменяй её похожей. Сверь пакет с карточками ещё раз; если детали нет — сообщи взрослому.

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.

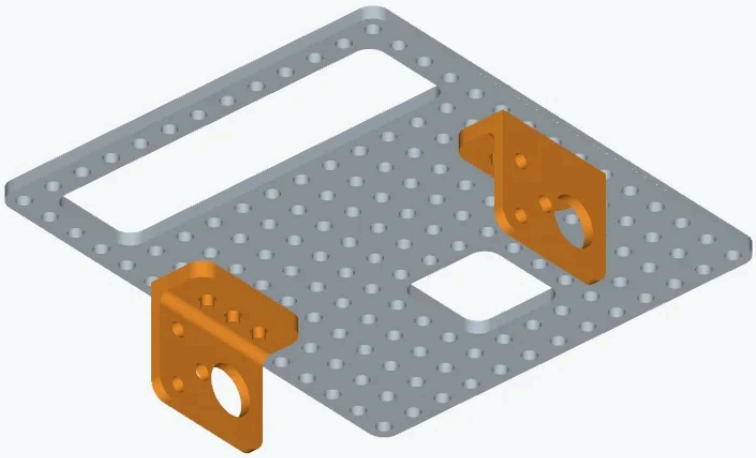
ДОЛЖНО ПОЛУЧИТЬСЯ

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.

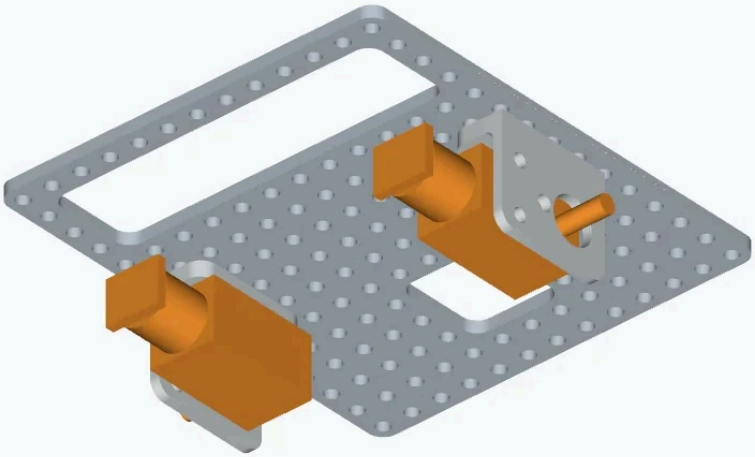
ПРОВЕРЬ СЕБЯ

- [ ] 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.

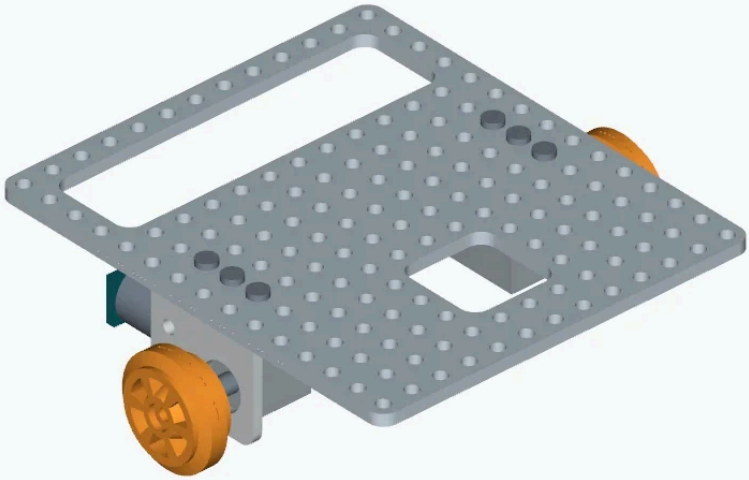
Подсказка: Мотор не встаёт в кронштейн

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.

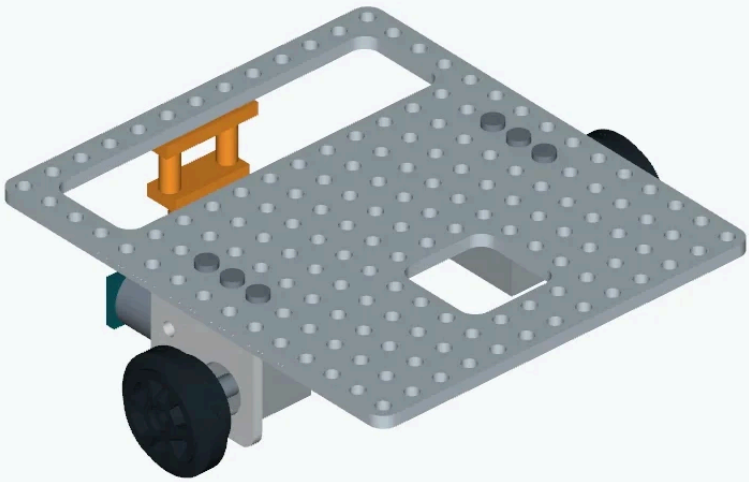
ДОЛЖНО ПОЛУЧИТЬСЯ

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.

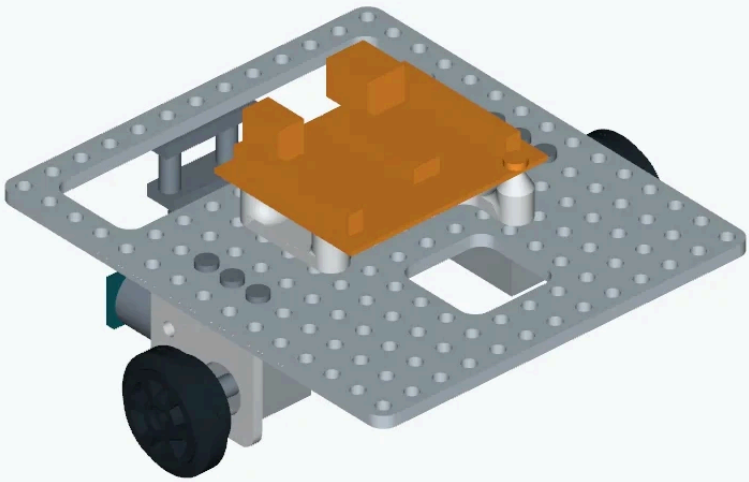
ЗАНЯТИЕ 2 · 45 МИН

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»**, урок **1.1** и его учебный проект.
3. В правой части кнопки **Загрузить на робота** выбери своего настоящего робота. Открой **Подробнее о связи** и проверь плату **Arduino Uno**. Отключи USB-кабель, посмотри, как изменилось состояние связи, и подключи его снова. Запиши имя порта в журнал.
4. В редакторе оставь пустые функции `setup()` и `loop()` и сохрани программу.

ДОЛЖНО ПОЛУЧИТЬСЯ

В MyRobotica открыт учебный проект и выбран настоящий робот с платой Arduino Uno. Имя его порта записано.

Подсказка: Порт так и не появился

Кабель может только заряжать и не передавать 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

```
void setup() {
}

void loop() {
}
```

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.

В программе уже готово всё, кроме двух функций в разделе «Твоя часть». Допиши их:

- `reportSensors` — печатает числа левого и правого датчика с подписями;
- `beginPulse` — включает выбранный мотор в нужную сторону со скоростью `TEST_PWM`. Второй мотор при этом выключен.

Остальную программу не меняй.

Допиши две функции L11_Diagnostics_Gaps.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;

// ===== Твоя часть =====
void reportSensors() {
  // Цель: показать в мониторе подписанные показания левого и правого
}

void beginPulse(int dirPin, int pwmPin, int direction) {
  // Цель: включить только указанный мотор с указанным направлением и
}

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

Подсказка: С чего начать

Найди в справке Arduino `analogRead`, `Serial.print`, `digitalWrite` и

`analogWrite`. Сначала проверь датчики — для этого моторы не нужны.

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

★ Средний уровень

1. Что должна вывести функция чтения датчиков?

- ○ Два подписанных реальных показания
- ○ Всегда слово «исправен»
- ○ Одно значение для двух датчиков

2. Выбран левый мотор. Что делает правый?

- ○ Остаётся выключенным
- ○ Повторяет левый
- ○ Работает в обратную сторону

3. Когда настраивают выводы и выключают моторы при запуске?

- ○ В setup до ожидания команд
- ○ После первого движения
- ○ Только после сигнала ученика

4. Что доказывает PWM=0?

- ○ Фактическую неподвижность колеса
- ○ Команду выключения; остановку ещё нужно наблюдать
- ○ Исправность проводки

5. Что делать, если датчик выдаёт постоянное значение?

- ○ Засчитать узел
- ○ Поставить произвольный порог
- ○ Проверить подключение и повторить опыт

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).

Команда	Что делает
`pinMode(контакт, OUTPUT);`	Разрешает плате управлять контактом
`digitalWrite(контакт, FORWARD);`	Выбирает направление «вперёд»
`analogWrite(контакт, сила);`	Включает мотор с силой от 0 до 255; 0 — стоп
`delay(мс);`	Ждёт указанное число миллисекунд

1. Создай скетч `L21_Left_Motor` по шаблону и напиши `setup()` сам: что должно произойти, сказано в комментариях.

Шаблон: левый мотор L21_Left_Motor_Gaps.ino

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

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

// ===== Твоя часть: напиши сам =====

void setup() {
  // Подготовь оба контакта левого мотора.
  // Левое колесо крутится вперёд 2 секунды с силой 100 из 255 и оста
}

void loop() {
}
```

ПРЕДСКАЖИ

Что будет, если забыть команду остановки? Запиши ответ и проверяй его только на подставке.

2. Загрузи программу при выключенном питании моторов.
3. Включи питание моторов и нажми кнопку Reset на плате: программа начнётся заново.

Подсказка: С чего начать

Сначала оба контакта должны стать выходами, потом — направление, потом сила.

ДОЛЖНО ПОЛУЧИТЬСЯ

Левое колесо крутится вперёд 2 секунды и останавливается. Прогноз записан.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Колесо не крутится	Настроены ли контакты как выходы? Включено ли питание моторов? Нажми Reset.
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

Правый мотор устроен так же, как левый: контакт направления `4` и контакт скорости `5`. Имена всех четырёх контактов в шаблоне уже есть.

1. Создай скетч `L21_Both_Motors` по шаблону и напиши `setup()` сам. Свою программу левого мотора используй как образец.

Шаблон: оба мотора L21_Both_Motors_Gaps.ino

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

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;

// ===== Твоя часть: напиши сам =====

void setup() {
  // Подготовь контакты обоих моторов.
  // Оба колеса крутятся вперёд со скоростью speed одну секунду и ост
}

void loop() {
}
```

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

ПРЕДСКАЖИ

Ты включаешь колёса по очереди: сначала левое, потом правое. Заметно ли, что они трогаются не одновременно? Почему?

Подсказка: Колёса ведут себя по-разному

Сравни команды для левого и правого мотора строка за строкой: для каждого нужны настройка, направление, сила и остановка.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

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 ПОЧТИ НЕ ОСТАНАВЛИВАЕТСЯ

Соседи перенесли движение в `loop()` и добавили остановку:

```
void loop() {
  analogWrite(LEFT_SPEED_PIN, speed);
  analogWrite(RIGHT_SPEED_PIN, speed);
  delay(1000);
  analogWrite(LEFT_SPEED_PIN, 0);
  analogWrite(RIGHT_SPEED_PIN, 0);
}
```

Они ждали: секунду едет, потом стоит. На подставке колёса крутятся почти без остановки. Почему?

1. Запиши ответ в журнал.
2. Предложи **два** разных исправления и запиши их кодом: одно — без `loop()`, второе — оставив движение в `loop()`.

Подсказка: Второе исправление

Можно ли после остановки «застрять» на месте, чтобы `loop()` больше не повторялся? Вспомни защиту из урока 1.1… или пустой бесконечный цикл `while (true) { }`.

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить, почему остановка длится мгновение, и в журнале два исправления.

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()`. Функции `drive` и `forward` в шаблоне — образец.

1. Создай скетч `L21_Moves` по шаблону. Напиши `backward`, `stopMotors` и движение в `setup()` сам.

Шаблон: функции движения L21_Moves_Gaps.ino

```
// Шаблон среднего уровня. Напиши код в разделе «Твоя часть» по целям
// Функции drive и forward ниже — образец.

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;

// ===== Твоя часть: напиши сам =====

void backward(int ms) {
  // Оба колеса крутятся назад со скоростью speed ms миллисекунд.
}

void stopMotors() {
  // Оба мотора стоят.
}

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

  // Вперёд 1 секунду, остановка, пауза полсекунды, назад 1 секунду, с
}

// ===== Образец — не меняй =====

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);
```

```
40 }
41
42 void forward(int ms) {
43   drive(FORWARD, speed, FORWARD, speed);
44   delay(ms);
45 }
46
47 void loop() {
48 }
```

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

Подсказка: Функция назад

Чем «назад» отличается от «вперёд»? В программе уже есть константа для этого направления.

Подсказка: Как вызвать функцию

Вызов — имя функции и значение параметра в скобках, в конце `;`.

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.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

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, отметь старт скотчем, включи питание моторов и нажми Reset.
3. Повтори три раза. Каждый раз отметь, где робот остановился, и измерь линейкой, на сколько сантиметров он ушёл от старта вперёд-назад и вбок.

Заезд	Вперёд или назад, см	Вбок, см	Куда уводит
1			
2			
3			

4. Одинаково ли робот ошибается в трёх заездах? Запиши вывод.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Таблица из трёх заездов и вывод: куда и насколько уводит твоего робота.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Робот едет не прямо	Моторы не одинаковые — это нормально. Запиши, куда уводит.
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, стоп, назад, стоп три раза подряд, — написав вызовы `forward`, `stopMotors`, `backward` только один раз. Нужен цикл `for` из урока 1.1.

ПРЕДСКАЖИ

Вернётся ли робот после трёх челноков точно на старт? Запиши прогноз, потом отметь старт и финиш на полу.

Подсказка: Как устроен for

`for (int i = 0; i < 3; i++) { … }` повторяет команды в скобках три раза.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот выполняет оба задания. В журнале — прогноз и насколько робот ушёл от старта после трёх челноков.

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.

СРЕДНИЙ УРОВЕНЬ · ЧЕЛНОК БЕЗ КОПИРОВАНИЯ

Сделай «челнок» из трёх повторов, написав вызовы `forward`, `stopMotors`, `backward` только один раз. Подсказка: цикл `for` из урока 1.1.

Проверь себя

★ Средний уровень

1. Нужно повторить «вперёд — стоп — назад» три раза, написав вызовы функций только один раз. Что подойдёт?

- ○ Поместить вызовы в цикл for из трёх повторов
- ○ Написать функцию loop() три раза
- ○ Добавить delay(3000) после вызовов

2. Скорость уменьшили со 200 до 100, а время езды вперёд осталось 1 секунда. Что изменится?

- ○ За ту же секунду робот проедет меньше
- ○ Робот проедет то же расстояние быстрее
- ○ Робот поедет назад

3. В программе забыли остановить правый мотор. Что сделает робот после секунды движения?

- ○ Правое колесо продолжит крутиться, и робот закружится
- ○ Оба колеса остановятся, потому что левое стоит
- ○ Программа не загрузится

4. Вызвали forward(1000), а сразу за ним — backward(1000) без stopMotors() между ними. Что получится?

- ○ Ошибка при проверке программы
- ○ Робот резко сменит направление без остановки
- ○ Робот постоит секунду и поедет назад

5. После трёх челноков робот оказался в 6 см от старта, а после одного — в 2 см. Почему ошибка растёт?

- ○ Цикл for с каждым разом прибавляет время
- ○ delay становится длиннее при каждом повторе
- ○ Небольшая разница путей вперёд и назад складывается в каждом повторе

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

Робот должен повернуть налево на месте. Какое колесо должно крутиться вперёд, а какое назад? Запиши до запуска.

1. Создай скетч `L22_Turn_Test` по шаблону. Функции `drive` и `stopMotors` из урока 2.1 уже есть. Напиши поворот в `setup()` по своему прогнозу.

Шаблон: проба поворота L22_Turn_Test_Gaps.ino

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

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°

// ===== Твоя часть: напиши сам =====

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

  // Робот крутится на месте налево turnMs миллисекунд и останавливае
}

// ===== Готовое из урока 2.1 — не меняй =====

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 loop() {
}
```

2. Поставь робота на пол, на лист бумаги. Прочерти карандашом линию вдоль его носа. Включи питание моторов и нажми Reset.
3. Измени программу так, чтобы робот повернул **направо**. Проверь.

> **Подсказка: Робот едет, а не крутится**
>
> Если оба колеса крутятся в одну сторону, робот едет. Для поворота на месте нужны разные направления.

> **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Робот крутится на месте налево, а после твоего изменения — направо. Прогноз проверен.

2 Подбери время поворота на 90°

20 мин

На поворот 90° нужно какое-то время в миллисекундах. Узнать его можно только опытом. Как подбирать — реши сам, но меняй **только** `turnMs` и записывай каждую попытку.

1. Начерти на листе две линии под прямым углом. Поставь робота носом вдоль первой.
2. Подбирай `turnMs`, пока робот не повернёт на 90°. В журнале — не меньше четырёх попыток и объяснение, почему ты выбрал следующее число.
3. Когда поворот получился, повтори его три раза.

Попытка	`turnMs`	Что получилось	Что поменять дальше и почему
1			
2			
3			
4			

Подсказка: С какого числа начать

Посмотри, сколько повернулся робот в пробе с `turnMs = 400`. Больше 90° или меньше?

Подсказка: Каждый раз поворачивает по-разному

Проверь, что батарея заряжена, а пол ровный и не скользкий. Небольшая разница между попытками — нормально.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот три раза подряд поворачивает примерно на 90°. В журнале четыре попытки с объяснением и итоговое время.

3 Разворот на 180°

15 мин

ПРЕДСКАЖИ

Сколько миллисекунд нужно на разворот на 180°, если ты знаешь своё время на 90°? Запиши прогноз.

1. Проверь прогноз на полу и подправь число, если нужно. Запиши время разворота.
2. Ответь в журнале: получилось ровно вдвое больше, чем на 90°? Если нет, как ты это объяснишь?

Подсказка: Почему не ровно вдвое

Подумай, что происходит в начале поворота, когда колёса только трогаются, и в конце, когда моторы уже выключены.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот разворачивается и смотрит в обратную сторону. Время разворота и объяснение записаны.

2

ЗАНЯТИЕ 2 · 45 МИН

Функции поворотов

4 Функции поворотов

12 мин

1. Создай скетч `L22_Turns` по шаблону. Впиши свои времена из журнала в `turn90Ms` и `turn180Ms`.
2. Напиши `turnLeft`, `turnRight`, `turnAround` и квадрат в `setup()` сам. Функция `forward` из урока 2.1 уже есть — пользуйся ей как образцом.

Шаблон: функции поворотов L22_Turns_Gaps.ino

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

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

// ===== Твоя часть: напиши сам =====

void turnLeft() {
  // Поворот на месте налево на 90° по времени turn90Ms; в конце колё
}

void turnRight() {
  // Поворот на месте направо на 90° по времени turn90Ms; в конце коле
}

void turnAround() {
  // Разворот на месте на 180° по времени turn180Ms; в конце колёса с
}

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

  // Квадрат: четыре раза — вперёд 1 секунду, пауза, поворот налево, п
}

// ===== Готовое из урока 2.1 — не меняй =====

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 loop() {
}
```

3. Нажми **Проверить**.

Подсказка: Чем отличаются повороты

Сравни свою пробу поворота: какие колёса и в какую сторону крутятся налево, а какие — направо? А при развороте?

Подсказка: Четыре раза

Повторить одно и то же четыре раза помогает цикл `for` из урока 1.1.

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». Нарисуй путь в журнале. Где окажется робот в конце и куда будет смотреть?

1. Отметь старт скотчем. Запусти программу на полу три раза.
2. Каждый раз измерь, на сколько сантиметров робот не доехал до старта, и на сколько градусов примерно повёрнут его нос.
3. Объясни в журнале, почему ошибка в конце больше, чем ошибка одного поворота.

ДОЛЖНО ПОЛУЧИТЬСЯ

Три заезда с замерами и объяснение накопления ошибки.

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

★ **СРЕДНИЙ УРОВЕНЬ · РАЗВОРОТ ОБРАТНО НА ЛИНИЮ**

Наклей на пол полоску изоленты. Поставь робота на неё, сделай разворот на 180°. Робот должен снова стоять вдоль полоски, а нос — уходить от неё не дальше её ширины. Так разворот будет работать в итоговой задаче.

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

★ Средний уровень

1. Разворот на 180° длится 900 мс, но робот немного недоворачивает. Что сделать?

- ○ Немного увеличить время разворота
- ○ Уменьшить время разворота
- ○ Увеличить время езды вперёд

2. Зачем проверять разворот на полоске изоленты?

- ○ Видно, насколько нос робота ушёл от линии после разворота
- ○ Изолента увеличивает сцепление колёс
- ○ На изоленте робот поворачивает быстрее

3. При turnMs = 400 робот повернул на 70°. Какое число попробовать следующим?

- ○ Оставить 400 и поменять turnSpeed
- ○ Больше 400, например 480
- ○ Меньше 400, например 320

4. Почему разворот на 180° может длиться не ровно вдвое дольше поворота на 90°?

- ○ При долгом повороте батарея даёт больше силы
- ○ delay не принимает больших чисел
- ○ Разгон в начале и доворот по инерции в конце почти не зависят от угла

5. Чем turnRight() отличается от turnLeft()?

- ○ Колёса крутятся в обратные стороны: левое вперёд, правое назад
- ○ Время поворота вдвое больше
- ○ Скорость колёс отрицательная

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.

1

ЗАНЯТИЕ 1 · 45 МИН

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. Для заданий среднего уровня на дорожке понадобится и перекрёсток.

1. Возьми светлое матовое поле с чёрной линией или наклей чёрную матовую изоленту на белый лист: прямая, плавный изгиб и один перекрёсток.
2. Подготовь две карточки: белую и чёрную.
3. Найди на роботе левый и правый датчики. Поставь робота на линию и проверь: линия проходит между датчиками и не касается ни одного из них?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Дорожка с перекрёстком и карточки готовы. Линия помещается между датчиками.

2 Посмотри, что видит датчик

12 мин

Датчик светит на пол и меряет, сколько света вернулось. Плата превращает это в число, а чтобы увидеть его, программа отправляет число на компьютер.

Команда	Что делает
`analogRead(контакт)`	Показание датчика: число от 0 до 1023
`Serial.begin(9600);`	Включает связь с компьютером, один раз
`Serial.print(x);`	Печатает в той же строке
`Serial.println(x);`	Печатает и переходит на новую строку

1. Создай скетч `L31_Read_Sensors` по шаблону и напиши обе части сам. Левый датчик подключён к `A4`, правый — к `A5`.

Шаблон: показания двух датчиков L31_Read_Sensors_Gaps.ino

```
1  // Шаблон среднего уровня. Напиши код в разделе «Твоя часть» по целям
2
3  const int LEFT_SENSOR_PIN = A4;
4  const int RIGHT_SENSOR_PIN = A5;
5
6  // ===== Твоя часть: напиши сам =====
7
8  void setup() {
9    // Включи связь с компьютером на той же скорости, что в мониторе пор
10 }
11
12 void loop() {
13   // Пять раз в секунду печатай одну строку: L=число левого датчика  R
14 }
```

2. Загрузи программу и открой монитор порта со скоростью **9600**.
3. Поднеси под левый датчик белую карточку, затем чёрную. Держи карточку на той же высоте, на которой датчик стоит над полом.

ПРЕДСКАЖИ

Что напечатает монитор, если имя переменной в `Serial.print` взять в кавычки?

Подсказка: Текст и число в одной строке

Текст в кавычках печатается как есть, а имя переменной без кавычек заменяется её числом.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Вместо чисел непонятные символы	Скорость в мониторе порта и в программе должна совпадать: 9600.
Все числа в одной длинной строке	Нет перехода на новую строку.
Меняется не тот датчик	Провода левого и правого датчиков перепутаны.

3 Проверь оба датчика

8 мин

ПРЕДСКАЖИ

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

1. Заполни таблицу для четырёх положений робота:

Положение	L	R
Оба над белым		
Чёрное только под левым		
Чёрное только под правым		
Линия точно между датчиками		

2. Почему в последней строке оба числа «белые», хотя робот стоит на линии? Запиши.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена, прогноз проверен, ответ про положение «между датчиками» записан.

4 Измерь белое и чёрное

12 мин

Датчики не одинаковые: даже над одним полем их числа разные. Поэтому каждому датчику нужны свои измерения — **калибровка**.

1. Для каждого датчика запиши по три показания на белом и три на чёрном, в разных местах дорожки.
2. Выбери типичное число для каждой клетки и запиши, почему именно его.

Датчик	Белое: 3 числа	Белое	Чёрное: 3 числа	Чёрное
Левый				
Правый				

Подсказка: Числа в разных местах разные

Освещение и поле неодинаковые. Берут число, близкое к остальным, а не самое удачное.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале таблица с повторами и четыре выбранных числа. Белое и чёрное заметно отличаются.

2

ЗАНЯТИЕ 2 · 45 МИН

Робот едет по линии

5 Научи программу замечать линию

10 мин

Программе нужно правило: когда считать, что под датчиком линия? Сравни новое показание с двумя измерениями: **если оно ближе к чёрному, чем к белому, — под датчиком линия**.

ПРЕДСКАЖИ

Белое — 100, чёрное — 800, пришло число 650. Линия под датчиком или нет? Посчитай до запуска: сколько от 650 до чёрного и сколько до белого?

1. Создай скетч `L31_Line_Detect` по шаблону и впиши свои четыре числа вместо нулей.
2. Напиши `loop()` сам: решение «линия или нет» для каждого датчика и печать.

Шаблон: программа замечает линию L31_Line_Detect_Gaps.ino

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

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;

// ===== Твоя часть: напиши сам =====

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

  // Для каждого датчика реши, линия ли под ним (правило «ближе к чёр
  // Печатай строку L=… R=…, а если под датчиком линия — допиши «слева
  // Пять строк в секунду.
}

// ===== Готовое из прошлого шага — не меняй =====

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

3. Загрузи программу, открой монитор порта и подводи чёрную карточку под датчики по очереди.
4. Ответь в журнале: что начнёт печатать программа, если в ней перепутать белое и чёрное местами? Почему?

Подсказка: Как измерить расстояние между числами

От 650 до 800 — это `800 - 650`. Но программа заранее не знает, какое число больше. Команда `abs(…)` убирает знак: `abs(650 - 800)` равно

150.

Подсказка: Да или нет

Ответ «да или нет» хранит тип `bool`. Команды в `if (…) { … }` выполняются, только если ответ «да».

ДОЛЖНО ПОЛУЧИТЬСЯ

«слева линия» и «справа линия» появляются, только когда под датчиком чёрное. Прогноз про 650 сошёлся, ответ про перепутанные числа записан.

НАЙДИ ОШИБКУ · «СЛЕВА ЛИНИЯ» НА БЕЛОМ ПОЛЕ

Команда соседей написала так:

```
bool leftOnLine = abs(leftValue - LEFT_BLACK) < abs(leftValue - RIG
```

Иногда «слева линия» печатается, когда под левым датчиком белое поле. Найди ошибку и объясни, почему она проявляется не всегда.

6 Придумай, как поворачивать

7 мин

Робот едет вперёд, и линия должна оставаться между датчиками. Если линия оказалась под одним датчиком, робот уехал в сторону. Колесо пока умеет только ехать или стоять. Как вернуть робота к линии?

1. Поставь выключенного робота на линию. Сдвинь его так, чтобы линия оказалась под левым датчиком. Куда робот уехал? Куда ему повернуть? Какое колесо для этого 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	?	?	?

4. Объясни в журнале одной фразой, почему остановка одного колеса поворачивает робота.

Подсказка: Какое колесо останавливать

Вспомни урок 2.2: левое колесо стоит, правое едет. Вокруг какого колеса повернётся робот и в какую сторону?

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена, объяснение записано. Ты показал на выключенном роботе оба поворота.

7 Запиши правила в программу

10 мин

Соберём всё вместе: датчики, правило «ближе к чёрному» и моторы из урока 2.1. Колёса будут слушаться правил из твоей таблицы.

1. Создай скетч `L31_First_Line` по шаблону. Впиши свои четыре измерения. Настройка моторов и защита уже готовы.
2. Напиши `loop()` сам: перенеси свои условия из `L31_Line_Detect`, задай колёсам скорости по своей таблице правил и отправь их на моторы.

Шаблон: первое движение по линии L31_First_Line_Gaps.ino

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

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

// ===== Твоя часть: напиши сам =====

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

  // Реши, линия ли под каждым датчиком, — перенеси свои условия из L
  // Колёса выполняют твою таблицу правил: линия под датчиком — его ко
  // Колесо, которое правило остановило, снова едет, когда его датчик
}

// ===== Готовое: настройка и защита — не меняй =====

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
}
```

ПРЕДСКАЖИ

Что случится, если задать скорость колёс один раз в `setup()`, а не в начале каждого повтора `loop()`?

3. Нажми **Проверить**.

Подсказка: Колесо остановилось и больше не едет

Когда правило снова разрешает колесу ехать? Откуда в каждом повторе берётся его скорость?

Подсказка: Правило колеса

`if (проверка) { … }` выполняет команды в скобках, только если проверка верна. Сверься со своей таблицей: какую переменную проверить и какую скорость поменять?

ДОЛЖНО ПОЛУЧИТЬСЯ

`loop()` написан, **Проверить** проходит без ошибок, ответ на прогноз записан в журнал.

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, включи питание моторов.
2. Проверь все четыре строки своей таблицы правил: подставляй карточки так, чтобы датчики видели нужное. Записывай, что делают колёса.

Что видят датчики	Ждал	Увидел
Оба белое		
Левый — линия		
Правый — линия		
Оба — линия		

3. Проверь свой прогноз про скорость, заданную в `setup()`. Выключи питание моторов.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Таблица заполнена, все строки совпали с правилами. Прогноз проверен.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Колёса не крутятся	Написан ли `loop()`? Вписаны ли измерения? Включено ли питание моторов?
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. Включи питание. Будь готов выключить его, если робот уходит с дорожки.
3. Пройди дорожку три раза при `speed = 100`, потом три раза при 80 и при 130. Записывай, сколько раз робот прошёл изгиб.

speed	Прошёл из 3	Что видно
80		
100		
130		

4. Почему робот едет рывками? Запиши одной фразой.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Таблица заездов при трёх скоростях и объяснение рывков.

★ **СРЕДНИЙ УРОВЕНЬ · ЕДЕМ ЧЕРЕЗ ПЕРЕКРЁСТОК**

Сейчас, если оба датчика видят чёрное, робот останавливается. На среднем поле через перекрёстки нужно проезжать прямо. Измени программу так, чтобы при чёрном под обоими датчиками оба колеса ехали вперёд. Проверь на дорожке с одним перекрёстком.

Проверь себя

★ Средний уровень

1. **На среднем поле оба датчика видят чёрное. Что должен делать робот по заданию ★?**

- ○ Ехать прямо: оба колеса вперёд
- ○ Остановиться
- ○ Развернуться

2. **Какое условие означает перекрёсток?**

- ○ leftOnLine && rightOnLine
- ○ leftOnLine || rightOnLine
- ○ !leftOnLine && !rightOnLine

3. **У датчика белое — 900, чёрное — 120. Он показывает 300. Что решит условие «ближе к чёрному, чем к белому»?**

- ○ Решить нельзя: у датчика чёрное меньше белого
- ○ Под датчиком линия: до чёрного 180, до белого 600
- ○ Под датчиком белое поле: 300 — маленькое число

4. В программе перепутали LEFT_WHITE и LEFT_BLACK местами. Что будет печатать программа?

- ○ То же, что раньше: abs уберёт разницу
- ○ Программа не загрузится
- ○ «слева линия» над белым полем и ничего над чёрным

5. Почему после остановки левого колеса робот поворачивает налево?

- ○ Правое колесо едет и поворачивает робота вокруг стоящего левого
- ○ Левое колесо начинает крутиться назад
- ○ Робот разворачивается по инерции

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()`?

ЗАНЯТИЕ 1 · 45 МИН

1 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: чуть ушла — чуть поворачиваешь руль, сильно ушла — сильнее. Программа из урока 3.1 так не умеет: она знает только «линия есть» или «нет».

1. Запусти программу из урока 3.1 на дуге 350 мм и посмотри, как едет робот.
2. Запиши в журнал одной фразой: чего не хватает программе, чтобы поворачивать плавно?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Ты можешь объяснить: сила поворота должна зависеть от того, насколько далеко ушла линия.

2 Общая шкала для двух датчиков

15 мин

У датчиков разные числа белого и чёрного, поэтому сравнивать их напрямую нельзя. Нужна общая шкала: **чёрное — 0, белое — 100**. Показание на шкале — это «сколько процентов пути от чёрного до белого» оно прошло.

1. Создай скетч `L32_Error` по шаблону, перенеси измерения из журнала.
2. Напиши функцию `level` сам. Ошибку в `loop()` пока не трогай.

Шаблон: шкала и ошибка L32_Error_Gaps.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

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

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;

// ===== Твоя часть: напиши сам =====

int level(int value, int white, int black) {
  // Шкала: чёрное — 0, белое — 100, между ними — по прямой.
  // Работает и когда у датчика чёрное — большое число, и когда малень
  return 0;
}

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);

  // Ошибка: насколько и в какую сторону линия ушла от середины; линия
  float error = NAN; // Пока ошибка не вычислена, график показывает «н

  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: настройка и защита — не меняй =====

void setup() {
  courseTelemetryBegin();
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    Serial.println("Впиши измерения белого и чёрного");
    while (true) {
      courseTelemetryPoll();
    }
  }
}

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

ПРЕДСКАЖИ

У датчика белое — 120, чёрное — 780. Какое число шкалы получится для показания 450? Посчитай до запуска.

3. Загрузи и проверь на белом, чёрном и краю линии.
4. Подними робота на 3–4 см над столом: датчику почти нечего отразить. Какое «сырое» число получается, что показывает шкала и какая часть функции за это отвечает?
5. Ответь в журнале: какого типа число ты взял для расчёта и почему не обычный `int`?

Подсказка: С чего начать формулу

Путь от чёрного до белого — `white - black`. Пройденная часть — `value - black`. Осталось превратить их отношение в проценты.

Подсказка: Шкала скачет у белого

На Arduino Uno `int` хранит числа только до 32 767. Посчитай, сколько выйдет `(value - black) * 100` для показания 50 и чёрного 780.

Подсказка: Шкала уходит за 0…100

`constrain(x, a, b)` возвращает `x`, но не меньше `a` и не больше `b`.

ДОЛЖНО ПОЛУЧИТЬСЯ

На белом около 100, на чёрном около 0, для показания 450 из примера — 50. Шкала не уходит за 0…100, и ты знаешь почему. В журнале — ответ про тип числа.

3 Считаем ошибку

15 мин

Ошибка показывает, насколько и в какую сторону линия ушла от середины. Считаем её как разницу: левое показание минус правое.

1. Напиши расчёт ошибки в `loop()` сам и загрузи скетч.

ПРЕДСКАЖИ

Линия ушла под левый датчик. Какой знак будет у ошибки? Почему?

2. Проверь прогноз: подвинь линию под левый датчик, потом под правый. Запиши в журнал.

Где линия	leftLevel	rightLevel	Ошибка
Посередине			
Под левым			
Под правым			

3. Ответь в журнале: в какую сторону должен повернуть робот, если ошибка отрицательная?

Подсказка: Знак ошибки

Чёрная линия — 0 на шкале. Если линия под левым датчиком, какое из двух чисел меньше?

ДОЛЖНО ПОЛУЧИТЬСЯ

Знаки в таблице совпали с прогнозом, и ты можешь объяснить, куда поворачивать при каждом знаке.

2

ЗАНЯТИЕ 2 · 45 МИН

П-регулятор на дорожке

4 От ошибки к скоростям

10 мин

Правило П-регулятора: **поправка пропорциональна ошибке**. Чем дальше ушла линия, тем сильнее одно колесо ускоряется, а другое замедляется. Насколько сильно — задаёт `kp`.

1. Создай скетч `L32_P_Line` по шаблону, перенеси измерения и свою функцию `level`.
2. Напиши `loop()` сам: ошибка, поправка, скорости колёс в пределах 0…255 и команда моторам.

Шаблон: П-регулятор L32_P_Line_Gaps.ino

```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

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

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;
float kp = 0.5;     // сила поворота: подбираем на дорожке
int baseSpeed = 100; // скорость на прямой

// ===== Твоя часть: напиши сам =====

int level(int value, int white, int black) {
  // Перенеси свою функцию level из L32_Error.
  return 0;
}

void loop() {
  float observedError = NAN;
  float observedOutput = NAN;
  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);
```

```
40
41    // Ошибка — как в L32_Error.
42    // Поправка пропорциональна ошибке, коэффициент kp.
43    // Робот поворачивает к линии тем сильнее, чем больше ошибка; скоро
44    // Отправь скорости на моторы.
45    // Сохрани вычисленные ошибку и поправку в observedError и observed
46    courseTelemetryReport(leftRaw, rightRaw, observedError, observedOut
47  }
48
49  // ===== Готовое: моторы, настройка и защита — не меняй =====
50
51  void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed)
52    digitalWrite(LEFT_DIR_PIN, leftDir);
53    digitalWrite(RIGHT_DIR_PIN, rightDir);
54    analogWrite(LEFT_SPEED_PIN, leftSpeed);
55    analogWrite(RIGHT_SPEED_PIN, rightSpeed);
56  }
57
58  void setup() {
59    courseTelemetryBegin();
60    pinMode(LEFT_DIR_PIN, OUTPUT);
61    pinMode(LEFT_SPEED_PIN, OUTPUT);
62    pinMode(RIGHT_DIR_PIN, OUTPUT);
63    pinMode(RIGHT_SPEED_PIN, OUTPUT);
64    drive(FORWARD, 0, FORWARD, 0);
65
66    // Без измерений робот не поедет.
67    if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
68      while (true) {
69        courseTelemetryPoll();
70      }
71    }
72    courseWait(2000);
73  }
74
75  // ===== Готовое: передача наблюдений — не меняй =====
76  void courseTelemetryBegin() {
77  #if defined(ROBOT_TELEMETRY_BUILD_ID)
78    using namespace robot_telemetry::wire;
79    courseTelemetry.begin((1UL << Line0) | (1UL << Line1) |
80                          (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);
  }
}
```

3. Посчитай на бумаге: `baseSpeed = 100`, `kp = 0.5`, ошибка −40. Какие скорости получат колёса и куда повернёт робот? На подставке в следующем шаге проверишь это на колёсах.

ПРЕДСКАЖИ

Что будет делать робот, если поставить `kp = 0`? А если сделать `kp` отрицательным?

Подсказка: Какое колесо ускорять

Линия слева — ошибка отрицательная. Какое колесо должно ехать быстрее, чтобы робот повернул к линии? Согласуй знак поправки с этим.

Подсказка: Границы скорости

Мотор понимает только 0…255. Какая команда из урока не даёт числу выйти за границы?

ДОЛЖНО ПОЛУЧИТЬСЯ

`loop()` написан. Для примера: поправка −20, левое колесо 80, правое 120 — робот поворачивает налево, к линии. Прогноз про `kp = 0` записан — проверишь на подставке.

5 Проверь на подставке

7 мин

1. Поставь робота на подставку, загрузи программу, включи питание моторов.
2. Подведи линию под левый датчик, потом под правый, потом посередине. Запиши, какое колесо ускоряется и какое замедляется.
3. Проверь свой прогноз про `kp = 0`: поставь `kp = 0`, загрузи и подвигай линию. Что делают колёса? Верни свой `kp`.

Где линия	Левое колесо	Правое колесо	Совпало с прогнозом?
Под левым			
Под правым			
Посередине			

ДОЛЖНО ПОЛУЧИТЬСЯ

Колёса меняют скорость плавно. Таблица заполнена, прогноз про `kp = 0` проверен.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Колёса не крутятся	Написан ли `loop()` с командой моторам? Вписаны ли измерения?
Ускоряется не то колесо	Проверь знак ошибки по монитору порта и формулы скоростей

6 Подбери коэффициент

18 мин

Подбери `kp` сам. Правило одно: **меняй одно число за раз**.

1. Начни с `kp = 0.5` при `baseSpeed = 100`. Затем попробуй одно явно меньшее и одно явно большее значение и сузь диапазон.
2. Запиши в журнал не меньше четырёх попыток.

`kp`	`baseSpeed`	Что видно	Что поменять дальше и почему
0.5	100		
	100		
	100		
	100		

3. С лучшим `kp` подними `baseSpeed`. Что пришлось изменить, чтобы робот снова держал дугу?

Подсказка: Как понять, куда менять kp

Робот уходит с дуги наружу — поправка слабая. Робот виляет даже на прямой — поправка сильная.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале четыре попытки с объяснением каждого шага и пара `kp` – `baseSpeed`, на которой робот проходит дугу.

7 Задания по уровням

10 мин

★ **СРЕДНИЙ УРОВЕНЬ · ДУГА НА СКОРОСТИ**

Пройти дугу радиусом 250 мм, как на среднем поле, на скорости не ниже той, что подобрана для прямой. Если не получается — что важнее увеличить: `kp` или уменьшить скорость?

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

★ Средний уровень

1. На дуге радиусом 250 мм робот на скорости съезжает наружу. Что попробовать?

- ○ Немного увеличить kp или уменьшить скорость
- ○ Уменьшить kp и увеличить скорость
- ○ Поменять местами датчики

2. Почему на дуге меньшего радиуса нужна более сильная поправка?

- ○ Линия уходит в сторону быстрее, и ошибка растёт быстрее
- ○ На дуге датчики хуже видят линию
- ○ На дуге колёса крутятся медленнее

3. Линия ушла под левый датчик, ошибка error = leftLevel - rightLevel отрицательная. Какое колесо должно ускориться, чтобы робот вернулся к линии?

- ○ Правое: робот повернёт налево, к линии
- ○ Левое: робот повернёт налево, к линии
- ○ Оба одинаково: так робот быстрее догонит линию

4. Почему в функции level результат считают в long, а не в int?

- ○ Произведение на 100 может не поместиться в int Arduino Uno
- ○ С long программа работает быстрее
- ○ analogRead возвращает только long

5. **Команда поменяла сразу kp и baseSpeed, и робот стал проходить дугу. Что не так с этим опытом?**

- ○ Непонятно, какое из двух изменений помогло
- ○ Всё так: чем больше изменений, тем быстрее подбор
- ○ kp нельзя менять вместе с программой

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? А на изгибе линии?

1. Загрузи скетч со шкалой из урока 3.2 (`L32_Error` или свой шаблон).
2. Медленно провези робота руками по дорожке через изгиб и через перекрёсток. Смотри на монитор порта.
3. Заполни таблицу и ответь: по какому признаку перекрёсток можно отличить от изгиба?

Положение	leftLevel	rightLevel
Линия посередине		
На изгибе		
На перекрёстке		
После перекрёстка		

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Таблица заполнена, признак перекрёстка записан словами.

2 Условие перекрёстка

10 мин

Сформулируй правило сам по своим записям из прошлого шага: **какие показания бывают только на перекрёстке** и никогда — на прямой или изгибе?

1. Запиши правило словами в журнал: что должно быть у левого датчика и что у правого.
2. Выбери порог «чёрного» по своим измерениям. Объясни, почему не 0 и почему не 50.
3. Запиши условие на языке Arduino. Понадобится знак «и»: `&&`.

ПРЕДСКАЖИ

Кто-то написал условие через `||` («или»). Где робот тогда остановится по ошибке?

Подсказка: Как записать «и»

`a < 30 && b < 30` верно, только если верны обе части.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале — правило словами, порог с объяснением, условие и ответ про `||`.

3 Стоп на перекрёстке

20 мин

1. Создай скетч `L41_Crossing_Stop` по шаблону. Перенеси измерения, `kp` и `baseSpeed` из журнала. Функции линии из урока 3.2 уже готовы.
2. Напиши сам `onCrossing()` по своему правилу из прошлого шага, `followLineToCrossing()` и движение в `setup()`. Пока `onCrossing()` не написана, она всегда отвечает «перекрёсток», и робот не трогается.

Шаблон: стоп на перекрёстке L41_Crossing_Stop_Gaps.ino

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

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;

// ===== Твоя часть: напиши сам =====

bool onCrossing() {
  // true — только когда робот на поперечной линии; на прямой и на изг
  return true;
}

void followLineToCrossing() {
  // Ехать по линии с П-регулятором, пока не найдёшь перекрёсток; на к
}

void setup() {
  setupRobot();

  // Доехать по линии до первого перекрёстка и остановиться.
}

// ===== Готовое из уроков 3.1–3.2 — не меняй =====
```

```
40
41 int level(int value, int white, int black) {
42   long result = (long)(value - black) * 100 / (white - black);
43   return constrain(result, 0, 100);
44 }
45
46 void readSensors() {
47   leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
48   rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
49 }
50
51 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
52   digitalWrite(LEFT_DIR_PIN, leftDir);
53   digitalWrite(RIGHT_DIR_PIN, rightDir);
54   analogWrite(LEFT_SPEED_PIN, leftSpeed);
55   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
56 }
57
58 void stopMotors() {
59   analogWrite(LEFT_SPEED_PIN, 0);
60   analogWrite(RIGHT_SPEED_PIN, 0);
61 }
62
63 void followLineStep() {
64   int error = leftLevel - rightLevel;
65   int correction = kp * error;
66   drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
67         FORWARD, constrain(baseSpeed - correction, 0, 255));
68 }
69
70 void setupRobot() {
71   pinMode(LEFT_DIR_PIN, OUTPUT);
72   pinMode(LEFT_SPEED_PIN, OUTPUT);
73   pinMode(RIGHT_DIR_PIN, OUTPUT);
74   pinMode(RIGHT_SPEED_PIN, OUTPUT);
75   stopMotors();
76   // Без измерений робот не поедет.
77   if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
78     while (true) {
79     }
80   }
```

```
  delay(2000);
}

void loop() {
}
```

ПРЕДСКАЖИ

Что сделает робот, если после поиска перекрёстка забыть остановить моторы?

3. Поставь робота на линию перед перекрёстком и запусти.

Подсказка: Ехать, пока…

Функция повторяет одно и то же — прочитать датчики, сделать шаг по линии — и заканчивается, когда найден перекрёсток. Какой цикл повторяет «всегда», и как выйти из функции посреди него?

Подсказка: Как выйти из функции

`return` внутри функции сразу заканчивает её. Программа продолжается со следующей строки в `setup()`.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот едет по линии и останавливается на первом перекрёстке. Прогноз записан.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Робот проезжает перекрёсток	Снизь `baseSpeed` или подними порог
Робот останавливается на изгибе	Порог слишком высокий
Робот не едет	Вписаны ли измерения? Написана ли `onCrossing()`?

ЗАНЯТИЕ 2 · 45 МИН

2 Считаем перекрёстки

4 Найди ошибку счёта

8 мин

НАЙДИ ОШИБКУ · ПАУЗА НЕ ПОМОГЛА

Соседи заметили двойной счёт и добавили паузу после счёта:

```
void loop() {
  readSensors();
  if (onCrossing()) {
    count = count + 1;
    delay(100);
  }
  followLineStep();
}
```

Робот проехал один перекрёсток, а `count` стал 3. Почему пауза не помогла? Чем она ещё опасна для езды по линии?

1. Запиши ответ в журнал.
2. Предложи исправление словами: что должна сделать программа после того, как нашла перекрёсток, прежде чем искать следующий?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ты можешь объяснить, почему пауза не помогает, и предложить исправление.

5 Проехать перекрёсток

10 мин

Чтобы посчитать перекрёсток один раз, его нужно **проехать**: ехать прямо, пока датчики не съедут с чёрного. Напиши функцию `passCrossing()` в журнале:

1. Включи оба мотора вперёд со скоростью `baseSpeed`.
2. Читай датчики, **пока** робот на перекрёстке.
3. После этого подожди 100 мс.

`do { … } while (условие);` — сделать и повторять, пока условие верно.

Подсказка: Почему do, а не while

Датчики нужно прочитать хотя бы один раз, прежде чем проверять условие.

Подсказка: Зачем пауза в конце

Сразу после чёрного датчики видят край поперечной линии. Что будет, если в этот момент снова искать перекрёсток?

ДОЛЖНО ПОЛУЧИТЬСЯ

Функция записана в журнал. Ты можешь объяснить, когда закончится цикл и зачем пауза.

6 Стоп на нужном перекрёстке

17 мин

1. Создай скетч `L41_Count` по шаблону. В `TARGET` укажи, на каком перекрёстке остановиться.
2. Напиши сам `passCrossing()` — по своей записи в журнале — и движение в `setup()`. Пока твоя часть не написана, робот не трогается.

Шаблон: счёт перекрёстков L41_Count_Gaps.ino

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

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;
const int TARGET = 1; // на каком перекрёстке остановиться

// ===== Твоя часть: напиши сам =====

void passCrossing() {
  // Проехать перекрёсток прямо. Функция заканчивается, когда датчики
}

void setup() {
  setupRobot();

  // Перекрёстки до TARGET — найти и проехать прямо; на перекрёстке но
}

// ===== Готовое из уроков 3.2 и 4.1 — не меняй =====

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 loop() {
}
```

ПРЕДСКАЖИ

`TARGET = 3`. Сколько раз робот должен проехать перекрёсток прямо?

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

Подсказка: Повторить нужное число раз

Цикл `for` из урока 1.1 повторяет команды заданное число раз. Сколько перекрёстков нужно проехать перед тем, на котором остановиться?

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот останавливается ровно на перекрёстке с номером `TARGET`.

7 Задания по уровням

10 мин

★ **СРЕДНИЙ УРОВЕНЬ · ТРЕТИЙ ПЕРЕКРЁСТОК**

Остановиться на третьем перекрёстке, проехав два первых прямо. На среднем поле итоговой задачи робот проезжает перекрёстки, не сворачивая.

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

★ Средний уровень

1. Счётчик crossings начинается с 0 и увеличивается на каждом перекрёстке. При каком значении остановиться на третьем перекрёстке?

- ○ 3
- ○ 2
- ○ 4

2. Условие перекрёстка записали через «или»: leftLevel < 30 || rightLevel < 30. Где робот остановится?

- ○ Нигде: условие никогда не выполнится
- ○ На первой дуге или при вилянии: хватает одного тёмного датчика
- ○ Только на перекрёстке

3. Почему порог «чёрного» — 30, а не 0?

- ○ С нулём нельзя сравнивать
- ○ С порогом 0 робот едет медленнее
- ○ Даже на чёрном показание редко ровно 0

4. В шаблоне onCrossing() пока возвращает true. Что делает робот после запуска?

- ○ Сразу останавливается: «перекрёсток» найден в первом же повторе
- ○ Едет по линии без остановки
- ○ Крутится на месте

5. Зачем в passCrossing() цикл do … while, а не while?

- ○ while не умеет вызывать функции
- ○ Датчики нужно прочитать хотя бы один раз, прежде чем проверять условие
- ○ do выполняется быстрее

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?

1

ЗАНЯТИЕ 1 · 45 МИН

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. Поставь робота на центр перекрёстка и медленно поверни руками налево, потом направо.
2. Для каждого поворота запиши: какой датчик первым оказался над новой линией и какой датчик проходит над старой линией в начале поворота.

Поворот	Первым видит новую линию	Сначала проходит над старой
Налево		
Направо		

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена. 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.

ПРЕДСКАЖИ

Робот повернул на месте сразу, как только датчики увидели перекрёсток. Где окажется робот относительно новой линии? Покажи на дорожке руками.

1. Поставь робота датчиками на поперечную линию и поверни его руками на месте на 90°. Где оказались датчики — над новой линией или сбоку?
2. Измерь линейкой, сколько ещё ехать колёсам до центра перекрёстка. Запиши в журнал.
3. Время этого проезда — `centerMs` — подберёшь на следующем шаге. Как ты поймёшь, что время слишком маленькое, а как — что слишком большое? Запиши.

ДОЛЖНО ПОЛУЧИТЬСЯ

Прогноз проверен руками, расстояние записано, признаки «мало» и «много» для `centerMs` записаны.

3 Поворот на перекрёстке

22 мин

1. Создай скетч `L42_Turn_At_Crossing` по шаблону. Перенеси измерения, `kp` и `baseSpeed`. Функции линии и перекрёстков уже готовы.
2. Напиши сам выезд на центр `toCenter()`, поворот до линии `turnUntilLine()` и повороты с перекрёстка налево и направо. `setup()` ниже проверяет их на дорожке.

Шаблон: поворот на перекрёстке L42_Turn_At_Crossing_Gaps.ino

```ino
// Шаблон среднего уровня. Напиши код в разделе «Твоя часть» по целям
// setup() ниже — проверка твоих функций на дорожке.

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

// ===== Твоя часть: напиши сам =====

void toCenter() {
  // Колёса встают над центром перекрёстка: проехать вперёд centerMs
}

void turnUntilLine(bool left) {
  // Крутиться на месте (left = true — налево) до новой линии; в конце
  // Пока датчик ещё над старой линией, она не должна считаться новой
}

void turnLeftAtCrossing() {
  // Повернуть с перекрёстка налево на новую линию.
}
```

```
40
41 void turnRightAtCrossing() {
42   // Повернуть с перекрёстка направо на новую линию.
43 }
44
45 // ===== Готовое из уроков 3.2–4.1 и проверка — не меняй =====
46
47 int level(int value, int white, int black) {
48   long result = (long)(value - black) * 100 / (white - black);
49   return constrain(result, 0, 100);
50 }
51
52 void readSensors() {
53   leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
54   rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
55 }
56
57 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
58   digitalWrite(LEFT_DIR_PIN, leftDir);
59   digitalWrite(RIGHT_DIR_PIN, rightDir);
60   analogWrite(LEFT_SPEED_PIN, leftSpeed);
61   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
62 }
63
64 void stopMotors() {
65   analogWrite(LEFT_SPEED_PIN, 0);
66   analogWrite(RIGHT_SPEED_PIN, 0);
67 }
68
69 void followLineStep() {
70   int error = leftLevel - rightLevel;
71   int correction = kp * error;
72   drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
73         FORWARD, constrain(baseSpeed - correction, 0, 255));
74 }
75
76 void setupRobot() {
77   pinMode(LEFT_DIR_PIN, OUTPUT);
78   pinMode(LEFT_SPEED_PIN, OUTPUT);
79   pinMode(RIGHT_DIR_PIN, OUTPUT);
80   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);
}

// 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() {
}
```

ПРЕДСКАЖИ

Что случится, если начать искать новую линию сразу, не выждав `leaveMs`?

3. Запусти на дорожке с перекрёстком. Подбери `centerMs` по признакам, которые записал в прошлом шаге.

Подсказка: Крутиться до линии

Это похоже на `passCrossing()` из урока 4.1: повторять чтение датчиков, пока условие верно. Какое условие значит «линии под датчиком ещё нет»?

Подсказка: Какой датчик смотреть

Поворачиваем налево — новую линию первым увидит левый датчик.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

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

ЗАНЯТИЕ 2 · 45 МИН

2 Маршрут из решений

4 Разворот на линии

10 мин

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

1. Найди в скетче функцию `turnAroundOnLine()`. Почему в ней нет выезда на центр, как в повороте на перекрёстке? Запиши ответ.
2. Проверь на дорожке три раза: проехать по ответвлению, развернуться, вернуться к перекрёстку.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот три раза разворачивается и возвращается к перекрёстку. Ответ про выезд на центр записан.

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. Начерти в журнале тренировочную дорожку и путь: налево, прямо, направо.
2. Создай скетч `L42_Route` по шаблону и напиши маршрут в `setup()` сам из готовых функций урока.

Шаблон: маршрут L42_Route_Gaps.ino

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

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

// ===== Твоя часть: напиши сам =====

void setup() {
  setupRobot();

  // Маршрут из журнала: на каждом перекрёстке — своё решение по поряд
  // В конце остановиться на следующем перекрёстке.
}

// ===== Готовое из уроков 3.2–4.2 — не меняй =====

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

```
40
41 void readSensors() {
42   leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
43   rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
44 }
45
46 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
47   digitalWrite(LEFT_DIR_PIN, leftDir);
48   digitalWrite(RIGHT_DIR_PIN, rightDir);
49   analogWrite(LEFT_SPEED_PIN, leftSpeed);
50   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
51 }
52
53 void stopMotors() {
54   analogWrite(LEFT_SPEED_PIN, 0);
55   analogWrite(RIGHT_SPEED_PIN, 0);
56 }
57
58 void followLineStep() {
59   int error = leftLevel - rightLevel;
60   int correction = kp * error;
61   drive(FORWARD, constrain(baseSpeed + correction, 0, 255),
62         FORWARD, constrain(baseSpeed - correction, 0, 255));
63 }
64
65 void setupRobot() {
66   pinMode(LEFT_DIR_PIN, OUTPUT);
67   pinMode(LEFT_SPEED_PIN, OUTPUT);
68   pinMode(RIGHT_DIR_PIN, OUTPUT);
69   pinMode(RIGHT_SPEED_PIN, OUTPUT);
70   stopMotors();
71   // Без измерений робот не поедет.
72   if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
73     while (true) {
74     }
75   }
76   delay(2000);
77 }
78
79 // Перекрёсток: оба датчика видят чёрное.
80 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);
}

// 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 loop() {
}
```

ПРЕДСКАЖИ

Нарисуй, где робот остановится на дорожке.

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

Подсказка: Одно решение

Перед каждым решением робот должен доехать до перекрёстка.

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

СРЕДНИЙ УРОВЕНЬ · ЧЕТЫРЕ РЕШЕНИЯ

Составь на тренировочном поле маршрут из четырёх решений, в котором есть налево, направо и прямо. Запиши его в журнал, затем в программу, и пройди без ошибок.

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

★ Средний уровень

1. **Маршрут: налево, прямо, направо, налево. Сколько решений на перекрёстках выполнит робот?**

- ○ 4
- ○ 3
- ○ 2

2. **Зачем маршрут сначала записывают в журнал, а потом в программу?**

- ○ Так проще сверить маршрут со схемой поля до запуска
- ○ Плата читает журнал
- ○ Без журнала программа не соберётся

3. **После поворота робот оказался сбоку от новой линии — со стороны, откуда приехал. Что поменять?**

- ○ Уменьшить centerMs
- ○ Увеличить leaveMs
- ○ Увеличить centerMs: колёса не доехали до центра перекрёстка

4. Для какого поворота подходит этот поиск линии?

```
do {
  readSensors();
} while (leftLevel > 30);
```

- ○ Налево: робот крутится, пока левый датчик не увидит линию
- ○ Направо
- ○ Только для разворота

5. Почему каждое решение маршрута — это две строки: followLineToCrossing() и действие?

- ○ Вторая строка останавливает робота
- ○ Сначала надо доехать до следующего перекрёстка, иначе действие выполнится на том же
- ○ Так требует Arduino

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.

ПРЕДСКАЖИ

Звук пролетает 1 см примерно за 29 микросекунд. Сколько времени пройдёт до возвращения эха, если предмет в 10 см? А в 30 см?

1. Запиши расчёт в журнал.

2. Объясни, почему время нужно умножать на два.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Расчёт для 10 и 30 см записан с объяснением про «туда и обратно».

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` по шаблону и впиши номера контактов вместо `-1`.
3. Напиши функцию `readDistanceCm()` сам. Печать в мониторе порта уже готова. Пока функция не написана, монитор печатает «нет эха».

Команда	Что делает
`digitalWrite(TRIG_PIN, HIGH);` … `LOW`	Щелчок: TRIG в HIGH на 10 мкс, перед этим коротко LOW
`delayMicroseconds(мкс);`	Ждёт указанное число микросекунд
`pulseIn(ECHO_PIN, HIGH, 25000)`	Время эха в микросекундах; 0 — эха нет за 25 000 мкс

Шаблон: расстояние L51_Distance_Gaps.ino

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

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

// ===== Твоя часть: напиши сам =====

int readDistanceCm() {
  // Щёлкнуть датчиком и вернуть расстояние до груза в сантиметрах.
  // Если эхо не вернулось за 25 000 мкс, вернуть -1.
  return -1;
}

// ===== Готовое: настройка датчика и печать — не меняй =====

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);
}
```

ПРЕДСКАЖИ

Эхо вернулось через 1160 мкс. Сколько сантиметров до груза?

4. 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	

Подсказка: На что делить

Звук летит туда и обратно: 29 мкс на сантиметр в одну сторону и 29 — в другую. Сколько микросекунд приходится на один сантиметр расстояния до груза?

Подсказка: Время эха — большое число

Время может быть больше, чем помещается в `int`. Для него берут `unsigned long`.

ДОЛЖНО ПОЛУЧИТЬСЯ

Показания близки к линейке, разница — 1–2 см. Прогноз для 1160 мкс сошёлся.

ЕСЛИ НЕ ПОЛУЧИЛОСЬ

Что видишь	Что сделать
Программа ничего не печатает	Вписаны ли номера TRIG и ECHO вместо −1?
Всегда «нет эха»	Есть ли щелчок перед ожиданием эха? Не перепутаны ли TRIG и ECHO?
Числа прыгают	Груз должен стоять прямо напротив датчика

3 «Нет эха» — не «рядом»

15 мин

1. Проверь датчик в трёх положениях и запиши, что печатает монитор:

Положение	Что печатает
В пустоту	
На мягкую ткань под углом	
Ладонь ближе 2 см	

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?

2. Запиши правильное условие остановки.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена, ошибка соседей объяснена, правильное условие записано.

2

ЗАНЯТИЕ 2 · 45 МИН

Стоп перед грузом

4 Едем к грузу и останавливаемся

17 мин

1. Создай скетч `L51_Stop_At_Can` по шаблону. Перенеси измерения, `kp`, `baseSpeed`, контакты датчика и замени `readDistanceCm` своей функцией.
2. Напиши движение в `setup()` сам: робот едет по линии и останавливается перед грузом. Помни шаг «Нет эха — не рядом».

Шаблон: стоп перед грузом L51_Stop_At_Can_Gaps.ino

```
// Шаблон среднего уровня. Напиши код в разделе «Твоя часть» по целям
// Функцию readDistanceCm ниже замени своей из L51_Distance.

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;

// Контакты ультразвукового датчика: впиши номера из таблицы проводки
const int TRIG_PIN = -1;
const int ECHO_PIN = -1;
const int STOP_CM = 5; // на каком расстоянии от груза остановиться

// ===== Твоя часть: напиши сам =====

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

  // Ехать по линии и остановиться, когда груз не дальше STOP_CM.
  // «Нет эха» — не «груз рядом»: тогда робот едет дальше.
}

// ===== Готовое из уроков 3.2 и 5.1 — не меняй =====

```

```
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);
```

```
}

// 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 loop() {
}
```

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

Подсказка: Ехать, пока…

Как в `followLineToCrossing()` урока 4.1: в цикле измерить расстояние, решить, пора ли останавливаться, иначе сделать шаг по линии.

Подсказка: Две проверки

Остановиться можно, только если измерение есть и груз близко.

Какое число вернёт твоя функция, если эха нет?

Как это устроено: Как выйти из цикла

`break` немедленно выходит из цикла `while`. Программа продолжается после цикла.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

5 Точная остановка

15 мин

Подбери `STOP_CM` и `baseSpeed` так, чтобы робот останавливался там, где удобно захвату. Меняй одно число за раз.

1. Реши, на каком расстоянии от груза роботу нужно стоять. Если захват уже стоит, измерь у него; если нет — выбери расстояние, при котором робот не касается груза, и уточни его на уроке 5.2. Запиши.
2. Запусти, измерь линейкой, где робот остановился. Запиши в таблицу и реши, что менять.

`STOP_CM`	`baseSpeed`	Где остановился, см	Что поменять и почему

3. Когда получилось, повтори три раза.

Подсказка: Робот останавливается ближе, чем STOP_CM

Робот не останавливается мгновенно: он ещё немного катится. Чем выше скорость, тем дальше.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот три раза подряд останавливается на нужном расстоянии. В журнале — таблица попыток с объяснением.

6 Задания по уровням

13 мин

★ **СРЕДНИЙ УРОВЕНЬ · ПОВТОР ПРИ ОТСУТСТВИИ ЭХА**

Если эха нет, измерь ещё раз. Если эха нет пять раз подряд — останови робота: что-то не так с датчиком. Так робот не уедет с поля, потеряв груз.

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

★ Средний уровень

1. Эха нет пять раз подряд. Что делает робот по заданию ★?

- ○ Останавливается: возможно, с датчиком что-то не так
- ○ Едет дальше на полной скорости
- ○ Считает, что груз в 5 см

2. **Измерение вернуло «нет эха». Что программа делает сначала?**

- ○ Измеряет ещё раз
- ○ Закрывает захват
- ○ Поворачивает

3. **Эхо вернулось через 1160 мкс. Сколько сантиметров до груза?**

- ○ 40
- ○ 58
- ○ 20

4. **Почему время эха делят на 58, а не на 29?**

- ○ Звук летит туда и обратно: по 29 мкс на сантиметр в каждую сторону
- ○ Датчик удваивает время
- ○ 58 — скорость звука в сантиметрах

5. На высокой скорости робот останавливается ближе к грузу, чем на низкой. Почему?

- ○ STOP_CM зависит от скорости
- ○ После команды стоп робот ещё катится, и тем дальше, чем выше скорость
- ○ На скорости датчик хуже видит груз

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.

1

ЗАНЯТИЕ 1 · 45 МИН

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. Командуем углом в градусах: `gripper.write(90)`.

ПРЕДСКАЖИ

Что сделает сервопривод после `gripper.write(90)`, а потом `gripper.write(40)`? А мотор колеса после `analogWrite(…, 90)`?

1. Запиши в журнал: чем команда сервоприводу отличается от команды мотору колеса — что задаёт число и когда движение заканчивается.

ДОЛЖНО ПОЛУЧИТЬСЯ

Сравнение сервопривода и мотора колеса записано.

2 Найди углы «открыто» и «закрыто»

22 мин

У каждого захвата свои углы. Найди их опытом — способ выбери сам, но меняй только `angle` и записывай каждую попытку.

Команда	Что делает
`gripper.attach(контакт);`	Связывает сервопривод с контактом
`gripper.write(угол);`	Поворачивает на угол 0...180 и держит его

1. Создай скетч `L52_Servo_Test` по шаблону. Посмотри контакт сервопривода на карточке у робота и впиши его вместо `-1`.
2. Допиши `setup()` сам: захват должен повернуться на `angle`.

Шаблон: проба угла L52_Servo_Test_Gaps.ino

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

#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;

// ===== Твоя часть: напиши сам =====

void setup() {
  if (SERVO_PIN < 0) {
    while (true) {
    }
  }

  // Захват поворачивается на угол angle и держит его.
}

void loop() {
}
```

3. Найди угол «открыто»: груз свободно входит в захват.
4. Найди угол «закрыто»: груз держится крепко, а сервопривод не гудит.
5. Проверь «закрыто»: подними робота с грузом и слегка наклони. Груз не выпадает?

Попытка	`angle`	Что видно	Что поменять и почему
1	90		
2			
3			
4			

Подсказка: Сервопривод гудит или дрожит

Захват упёрся и не может дойти до угла. Что нужно сделать с углом?

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале — попытки с объяснением, углы «открыто» и «закрыто» и проверка наклоном.

3 Порядок действий важен

15 мин

1. До того как смотреть код, запиши в журнал по шагам: как роботу правильно **взять** груз и как **поставить**. Что должно случиться раньше: остановка или захват? Открытие или отъезд?
2. Сравни свой порядок с правилом:

- **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

Твой порядок записан и сверен с правилом, ошибка соседей объяснена.

2

ЗАНЯТИЕ 2 · 45 МИН

Перевозка груза

4 Взять и поставить

20 мин

1. Создай скетч `L52_Pick_Place` по шаблону. Перенеси измерения, `kp`, `baseSpeed`, контакты датчика и сервопривода, свои углы. Езда по линии, перекрёстки и подъезд к грузу уже готовы.
2. Напиши сам подготовку захвата `setupGripper()`, `pickUp()`, `putDown()` и доставку в `setup()` — по своим правилам порядка из прошлого шага.

Шаблон: взять и поставить L52_Pick_Place_Gaps.ino

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

#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;

// Контакты ультразвукового датчика: впиши номера из таблицы проводки
const int TRIG_PIN = -1;
const int ECHO_PIN = -1;
int turnSpeed = 100;
int centerMs = 250; // подбираем: проехать от датчиков до центра пере
int leaveMs = 200;  // время, чтобы датчик ушёл со старой линии

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

// ===== Твоя часть: напиши сам =====
```

```
40
41 void setupGripper() {
42   // Без контакта сервопривода программа дальше не идёт.
43   // Захват подключён и открыт; дай ему время раскрыться.
44 }
45
46 void pickUp() {
47   // Взять груз: груз оказывается в захвате и не выталкивается.
48 }
49
50 void putDown() {
51   // Поставить груз: груз остаётся на месте, когда робот отъезжает на
52 }
53
54 void setup() {
55   setupSonar();
56   setupGripper();
57   setupRobot();
58
59   // Доставка: подъехать к грузу, взять, развернуться на линии, проеха
60 }
61
62 // ===== Готовое из уроков 3.2–5.1 — не меняй =====
63
64 int level(int value, int white, int black) {
65   long result = (long)(value - black) * 100 / (white - black);
66   return constrain(result, 0, 100);
67 }
68
69 void readSensors() {
70   leftLevel = level(analogRead(LEFT_SENSOR_PIN), LEFT_WHITE, LEFT_BLA
71   rightLevel = level(analogRead(RIGHT_SENSOR_PIN), RIGHT_WHITE, RIGHT_
72 }
73
74 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
75   digitalWrite(LEFT_DIR_PIN, leftDir);
76   digitalWrite(RIGHT_DIR_PIN, rightDir);
77   analogWrite(LEFT_SPEED_PIN, leftSpeed);
78   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
79 }
80
```

```
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);
}

// 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);
}

// 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();
}

// 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();
}

void loop() {
}
```

ПРЕДСКАЖИ

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

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

Подсказка: Сколько ждать

Захвату нужно время, чтобы дойти до угла. Попробуй паузу 300 и 600 мс: успевает ли он?

Подсказка: Без контакта

Защиту без контакта ты уже видел в скетче пробы угла.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

5 Надёжность

12 мин

1. Повтори перевозку пять раз подряд. Каждую неудачу запиши: на каком этапе и что случилось.

Заезд	Подъезд	Захват	Перевозка	Постановка	Что случилось
1					
2					
3					
4					
5					

2. Найди этап, где ошибок больше всего. Какое одно число поменять, чтобы это исправить? Поменяй и повтори три раза.

Подсказка: Какие числа есть

`STOP_CM` — где робот встаёт перед грузом, `CARRY_MS` — где ставит, углы захвата — как держит.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица из пяти заездов, исправленный этап и три перевозки подряд без помощи рук.

6 Задания по уровням

13 мин

★ **СРЕДНИЙ УРОВЕНЬ · ЧЕРЕЗ ПЕРЕКРЁСТОК**

Груз стоит на одной ветке, место доставки — на другой. Перевези груз через перекрёсток, используя функции урока 4.2.

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

★ Средний уровень

1. Груз стоит на одной ветке, место доставки — на другой. Что нужно добавить к перевозке по линии?

- ○ Решение на перекрёстке из урока 4.2
- ○ Второй захват
- ○ Медиану измерений

2. В каком порядке выполняется доставка?

- ○ Подъехать к грузу, закрыть захват, доехать до места, открыть захват
- ○ Закрыть захват, подъехать к грузу, открыть захват
- ○ Открыть захват на месте, потом подъехать к грузу

3. Функция pickUp() осталась пустой. Что сделает робот у груза?

- ○ Развернётся и уедет без груза
- ○ Будет стоять у груза
- ○ Возьмёт груз как обычно

4. При проверке наклоном груз выпадает из захвата. Что поменять?

- ○ CARRY_MS
- ○ Угол «закрыто»: захват держит слишком слабо
- ○ STOP_CM

5. **Три заезда из пяти сорвались на постановке: груз ставится мимо места. Какое одно число менять первым?**

- ○ OPEN_ANGLE
- ○ kp
- ○ CARRY_MS — сколько робот едет с грузом

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. Поле выбирают по тому, какие задания уроков ты уже выполнил.

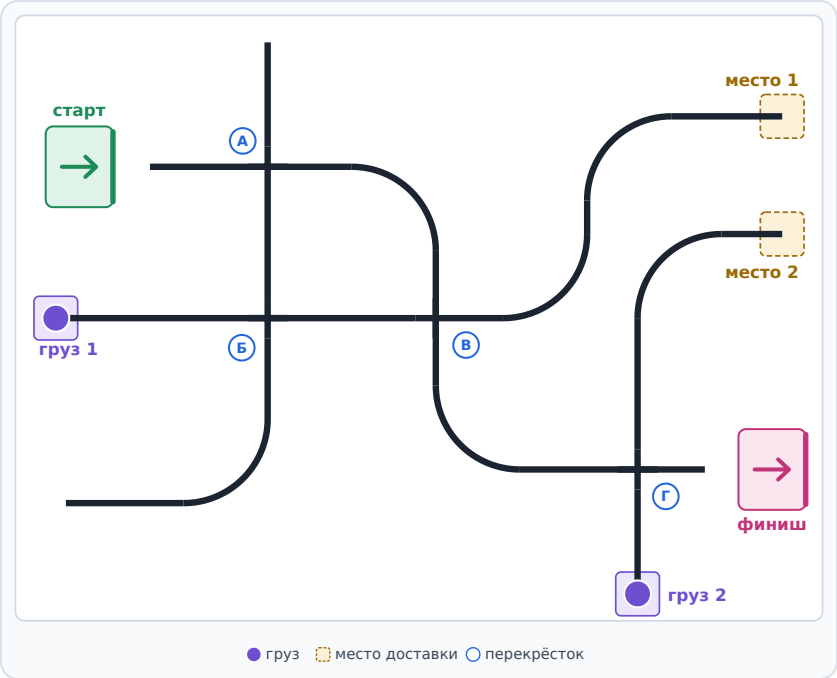
Уровень итоговой задачи	Какие задания уроков нужны
Начальный S-образная линия, два перекрёстка, порядок действий известен заранее.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2
Средний Сеть дорог, четыре перекрёстка, проезд прямо и повторные встречи с перекрёстками.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 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. Выпиши, какие задания ★ у тебя получились и в каких уроках.
2. Сравни со списком, который нужен для среднего поля. Чего не хватает?
3. Выбери поле сам. Если выбираешь среднее, но какого-то задания ★ нет, запиши, как и когда ты его доделаешь.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
Поле выбрано, в журнале — список выполненных заданий ★ и план для недостающих.

2 Разберись в задании своего уровня

8 мин



Среднее поле. Перекрёстки отмечены буквами А, Б, В, Г.

Груз 1 → место 1, груз 2 → место 2, затем финиш. Маршрут задан по перекрёсткам:

Участок	Через перекрёстки
Старт → груз 1	А → В → Б
Груз 1 → место 1	Б → В
Место 1 → груз 2	В → Г
Груз 2 → место 2	Г
Место 2 → финиш	Г

Куда поворачивать на каждом перекрёстке — налево, направо или прямо, — определяешь ты. Через некоторые перекрёстки робот проезжает несколько раз. Ветки, которых нет в маршруте, не используются.

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 **три раза** коротким опытом.

Умение	Урок	Прошло из 3
Колёса вперёд, назад и стоп	2.1	
Поворот 90° и разворот 180°	2.2	
Линия и дуги своего поля	3.2	
Перекрёсток и поворот с него	4.1, 4.2	
Остановка перед грузом	5.1	
Захват берёт и отпускает	5.2	

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена. Умения, которые прошли меньше трёх раз из трёх, ты чинишь первыми.

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 есть повороты налево, направо и проезд прямо. Каждая встреча с перекрёстком записана отдельной строкой — даже если это тот же перекрёсток. Программа не видит букв: она знает только, какое по счёту решение сейчас выполняет.

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. Когда маршрут записан вызовами функций, его легко сверить с планом.

Заготовка: маршрут по плану L61_Courier_Plan.ino

```
// Робот-курьер, средний уровень: маршрут по плану из журнала.
// Функций здесь нет: перенеси свои проверенные функции из уроков 2.1-
// Маршрут в setup() разбит на участки плана. В каждый участок впиши к

// --- Контакты, измерения и настройки из своих скетчей ---

// --- Мои функции (перенеси и проверь каждую отдельно) ---
// Нужны: остановка моторов, прямой проезд, линия до перекрёстка, прое
// прямо, поворот налево и направо на перекрёстке, разворот на линии,
// взять груз, поставить груз.

void setup() {
  // Подготовь выводы, датчики и захват. Моторы стоят.

  // Участок 1. Старт → линия:

  // Участок 2. Линия → груз 1:

  // Участок 3. Груз 1 → место 1:

  // Участок 4. Место 1 → груз 2:

  // Участок 5. Груз 2 → место 2:

  // Участок 6. Место 2 → финиш, стоп:
}

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

1. Открой заготовку и сохрани её под своим именем. Готовых функций в ней нет.
2. Перенеси свои проверенные функции из уроков 2.1–5.2. Проверь по списку в заготовке, что ни одной не забыл.
3. Перед каждой строкой плана в журнале запиши, какая функция её выполняет. Если для строки нет функции — напиши её и проверь отдельно.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Все нужные функции перенесены и проверены, у каждой строки плана есть своя функция.

★★ **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. Запиши в журнал вызовы функций первого участка по плану: выезд, линия, перекрёстки, повороты, подъезд к грузу. Потом перенеси их в `setup()`.
2. Проверь на поле только эту часть **три раза**. Робот должен каждый раз остановиться перед грузом 1.
3. Работает — сохрани копию: «рабочая версия 1».

Подсказка: Где искать ошибку

Сравни, что сделал робот, со строкой плана. На какой строке разошлось? Проверь эту одну функцию отдельно.

ДОЛЖНО ПОЛУЧИТЬСЯ

Робот три раза подряд выезжает со старта и останавливается перед грузом 1.

7 Часть 2: первая доставка

12 мин

1. Запиши в журнал вызовы функций первой доставки: захват, разворот, путь к месту 1, отпускание, возвращение к дороге. Потом допиши их в программу.
2. Проверь только эту часть три раза: ставь робота перед грузом 1 и запускай.
3. Работает — сохрани «рабочую версию 2».

Подсказка: Порядок с захватом

Вспомни правило урока 5.2: остановиться — закрыть; открыть — подождать — отъехать.

ДОЛЖНО ПОЛУЧИТЬСЯ

Три раза подряд груз 1 стоит в месте 1, 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 по плану. Груз 2 стоит в другом месте, поэтому повороты будут другими. Каждую встречу с перекрёстком — отдельной строкой.
2. В конце — прямой въезд в финиш и остановка. После финиша робот должен стоять.
3. Запусти весь маршрут от старта. Если сорвалось — запусти с места срыва только оставшуюся часть. Сохрани «рабочую версию 3».

Подсказка: Как понять, что пора въезжать в финиш

Чёрной метки финиша на поле нет. Используй последний перекрёсток и проверенное время прямого движения.

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. Отметь в своём плане из урока 6.1 каждую строку: выполнена или нет. Первая невыполненная строка — участок, с которого начнёшь отладку.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

План размечен по пробному заезду, участок для отладки выбран.

2 Найди и исправь ошибку

25 мин

Отладка идёт по кругу: **наблюдаю → причина → одно изменение → проверка**. Меняй одну вещь за раз, иначе не поймёшь, что помогло.

Наблюдаю	Причина	Одно изменение	Проверка

1. Заполни таблицу в журнале для своей ошибки. Если первая причина не подтвердилась, добавь строку со следующей.
2. Проверяй исправление только на своём участке: поставь робота перед ним.
3. Когда участок работает три раза подряд, проверь весь маршрут и сохрани новую рабочую версию.

Подсказка: Не знаю, на каком участке ошибка

Запусти первую половину маршрута. Сработала — ищи во второй. Так за два-три запуска найдёшь участок.

ДОЛЖНО ПОЛУЧИТЬСЯ

Ошибка найдена и исправлена, в журнале — строки круга отладки.

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.

Все решения на перекрёстках верны в двух заездах подряд.

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, по строкам круга отладки.
4. Объясни одно своё решение: почему на каком-то перекрёстке или участке ты сделал именно так. Что было бы при другом решении?
5. Запиши в журнал: какое умение у твоего робота работает лучше всего, а какое ты бы улучшил дальше и как.
6. Сохрани итоговый скетч и лист оценки.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, как устроена моя программа, по своему плану.
- [ ] Я могу обосновать одно своё решение и сравнить его с другим.

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. Запусти программу «по времени» из урока 2.1 трижды: скорость 100, 150 и 200 при одном времени. Измерь пути линейкой.
2. Ответь в журнале: что, кроме скорости в программе, меняет путь за одно и то же время? Назови не меньше трёх причин.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Три пути измерены, причины записаны.

2 Как устроен энкодер

12 мин

Энкодер — диск с прорезями на оси колеса и оптический датчик щели: прорезь проходит мимо датчика — датчик даёт **импульс**. Считать импульсы будет **прерывание**: Arduino сама вызывает короткую функцию на каждый импульс, что бы ни делала основная программа.

1. Посчитай прорези на диске, запиши номера контактов энкодеров из таблицы проводки.

ПРЕДСКАЖИ

Колесо диаметром 45 мм, на диске 20 прорезей. Сколько миллиметров проезжает колесо между двумя импульсами? Посчитай до запуска.

2. Ответь в журнале: почему счётчик импульсов нельзя увеличивать в `loop()` после `delay(200)`?

БЕЗОПАСНОСТЬ

Не подключай энкодеры при включённом питании. Контакты не указаны в таблице — подключает преподаватель.

Подсказка: Сколько проезжает колесо за оборот

За один оборот колесо проходит длину своей окружности: π × диаметр.

ДОЛЖНО ПОЛУЧИТЬСЯ

Число прорезей, контакты, прогноз «мм на импульс» и ответ про `delay` — в журнале.

3 Импульсов на один оборот

22 мин

1. Создай скетч `LF1_Count` по шаблону. Впиши контакты, напиши свою часть.

Шаблон: счёт импульсов LF1_Count_Gaps.ino

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

// 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;

// ===== Твоя часть: напиши сам =====

void countLeft() {
  // Каждый импульс левого энкодера увеличивает его счётчик на 1.
}

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

void loop() {
  long left = 0;
  long right = 0;
  // Скопируй оба счётчика в left и right так, чтобы прерывание не изм

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

// ===== Готовое: подключение энкодеров — не меняй =====

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();
}
```

2. Прокрути каждое колесо рукой ровно 10 оборотов по метке и посчитай импульсы на оборот.
3. Сравни с числом прорезей. Совпало? Если нет — найди причину.

Подсказка: Как не испортить число при чтении

Счётчик — `long`, это 4 байта, а Arduino читает их по одному. Прерывание посередине даст смесь старого и нового числа. Есть пара команд, которая на время выключает прерывания.

ДОЛЖНО ПОЛУЧИТЬСЯ

Счёт работает, импульсы на оборот обоих колёс записаны и сравнены с числом прорезей.

2

ЗАНЯТИЕ 2 · 45 МИН

Точное расстояние

4 Миллиметры в импульсы

15 мин

1. Создай скетч `LF1_Distance` по шаблону, впиши контакты и импульсы на оборот.
2. Напиши `ticksFor` и `averageTicks` по целям.

Шаблон: точное расстояние LF1_Distance_Gaps.ino

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

// 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;

// ===== Твоя часть: напиши сам =====

long ticksFor(int mm, int ticksPerRev, int diameterMm) {
  // Сколько оборотов колеса нужно на mm миллиметров: один оборот — д
  // Сколько это импульсов, округлённо до целого. Число PI в Arduino
  return 0;
}

long averageTicks() {
  // Среднее двух счётчиков. Читай их так, чтобы прерывание не измени
  return 0;
}

// ===== Готовое: энкодеры, моторы и поездка — не меняй =====

void countLeft() {
  leftTicks++;
}
```

```
40
41 void countRight() {
42   rightTicks++;
43 }
44
45 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
46   digitalWrite(LEFT_DIR_PIN, leftDir);
47   digitalWrite(RIGHT_DIR_PIN, rightDir);
48   analogWrite(LEFT_SPEED_PIN, leftSpeed);
49   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
50 }
51
52 void setupEncoders() {
53   // Без контактов энкодеров программа дальше не пойдёт.
54   if (LEFT_ENCODER_PIN < 0 || RIGHT_ENCODER_PIN < 0) {
55     Serial.println("Впиши контакты энкодеров");
56     while (true) {
57     }
58   }
59   pinMode(LEFT_ENCODER_PIN, INPUT);
60   pinMode(RIGHT_ENCODER_PIN, INPUT);
61   attachInterrupt(digitalPinToInterrupt(LEFT_ENCODER_PIN), countLeft,
62   attachInterrupt(digitalPinToInterrupt(RIGHT_ENCODER_PIN), countRigh
63 }
64
65 void setup() {
66   Serial.begin(9600);
67   pinMode(LEFT_DIR_PIN, OUTPUT);
68   pinMode(LEFT_SPEED_PIN, OUTPUT);
69   pinMode(RIGHT_DIR_PIN, OUTPUT);
70   pinMode(RIGHT_SPEED_PIN, OUTPUT);
71   drive(FORWARD, 0, FORWARD, 0);
72   setupEncoders();
73
74   // Без числа импульсов на оборот робот не поедет.
75   if (TICKS_PER_REV <= 0) {
76     Serial.println("Впиши число импульсов на оборот");
77     while (true) {
78     }
79   }
80   delay(2000); // время убрать руки от робота
```

```
81
82    long target = ticksFor(DISTANCE_MM, TICKS_PER_REV, WHEEL_DIAMETER_MM
83    drive(FORWARD, speed, FORWARD, speed);
84    while (averageTicks() < target) {
85    }
86    drive(FORWARD, 0, FORWARD, 0);
87    delay(1000); // робот докатывается
88
89    Serial.print("Цель, импульсов: ");
90    Serial.println(target);
91    Serial.print("Насчитано после остановки: ");
92    Serial.println(averageTicks());
93  }
94
95  void loop() {
96  }
```

ПРЕДСКАЖИ

Сколько импульсов вернёт `ticksFor(500, 20, 45)` ? Посчитай до запуска.

3. Запусти поездку от метки, измерь путь линейкой. Сравни с монитором порта: насколько «насчитано после остановки» больше цели?

Подсказка: Округление до целого

Если прибавить 0.5 и отбросить дробную часть, получится округление до ближайшего целого.

ДОЛЖНО ПОЛУЧИТЬСЯ

Прогноз совпал: 71. Робот проехал около 50 см, перекат после остановки виден в мониторе порта.

5 Почему робот проезжает дальше

15 мин

1. Сделай три заезда на 500 мм. Для каждого запиши путь линейкой и «перекат» в импульсах из монитора порта.
2. Добавь в программу поправку: останавливайся на `OVERSHOOT` импульсов раньше цели, где `OVERSHOOT` — средний перекат из трёх заездов. Сделай ещё три заезда.
3. Ответь в журнале: будет ли эта поправка верна на другой скорости? Почему?

ДОЛЖНО ПОЛУЧИТЬСЯ

Шесть заездов в журнале, поправка проверена, ответ про скорость записан.

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.

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

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

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

ПРЕДСКАЖИ

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

ДОЛЖНО ПОЛУЧИТЬСЯ

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

2 Робот измеряет сам

15 мин

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

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

Шаблон: функция регулятора LF2_Compare_Gaps.ino

```
// Наблюдение в MyRobotica включается сборщиком; обычный Arduino 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

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

#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;
```

```
40
41 // Итоги заезда: хранятся в памяти Arduino и после выключения.
42 struct Result {
43   int marker;
44   int mode;
45   int baseSpeed;
46   float kp;
47   float kd;
48   float ki;
49   long steps;
50   long sumAbs;
51   int maxAbs;
52   int crossings;
53 };
54
55 // ===== Твоя часть: напиши сам =====
56
57 // Поправка П-, ПД- или ПИД-регулятора по ошибке этого шага.
58 int controller(int mode, int error) {
59   // Занятие 1. П: поправка пропорциональна ошибке, коэффициент kp.
60   // Занятие 2. Д, режимы 2 и 3: прибавь kd, умноженный на то, насколь
61   // Занятие 2. И, только режим 3: копи ошибку в errorSum, не выпуска
62   // В конце запомни ошибку этого шага в lastError: она нужна Д на сле
63   return 0;
64 }
65
66 // ===== Готовое: шкала, моторы, измерение заезда и запуск — не меняй
67
68 // Переводит показание в шкалу: чёрное — 0, белое — 100.
69 int level(int value, int white, int black) {
70   long result = (long)(value - black) * 100 / (white - black);
71   return constrain(result, 0, 100);
72 }
73
74 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed) {
75   digitalWrite(LEFT_DIR_PIN, leftDir);
76   digitalWrite(RIGHT_DIR_PIN, rightDir);
77   analogWrite(LEFT_SPEED_PIN, leftSpeed);
78   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
79 }
80
```

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

// Добавляет ошибку шага к итогам заезда.
void record(int error) {
  run.steps++;
  run.sumAbs += abs(error);
  if (abs(error) > run.maxAbs) {
    run.maxAbs = abs(error);
  }
  int side = 0;
  if (error > 10) {
    side = 1;
  }
  if (error < -10) {
    side = -1;
  }
  if (side != 0 && lastSide != 0 && side != lastSide) {
    run.crossings++;
  }
  if (side != 0) {
    lastSide = side;
  }
}

void printResult(Result r) {
  const char *names[] = {"релейный", "П", "ПД", "ПИД"};
  Serial.print("Регулятор: ");
  Serial.print(names[r.mode]);
  Serial.print("  скорость=");
  Serial.print(r.baseSpeed);
  Serial.print("  kp=");
  Serial.print(r.kp);
  Serial.print("  kd=");
  Serial.print(r.kd);
  Serial.print("  ki=");
  Serial.println(r.ki, 3);
  Serial.print("Средняя ошибка: ");
```

```
  Serial.println(r.steps > 0 ? (float)r.sumAbs / r.steps : 0, 1);
  Serial.print("Наибольшая ошибка: ");
  Serial.println(r.maxAbs);
  Serial.print("Переходов через линию: ");
  Serial.println(r.crossings);
}

// Ждёт, пока робот стоит на старте: оба датчика видят белое 1 секунду
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
}
```

```
245
246 void courseWait(unsigned long durationMs) {
247   unsigned long started = millis();
248   while (millis() - started < durationMs) {
249     courseTelemetryPoll();
250     delay(1);
251   }
252 }
```

3. Поставь `MODE = 1`, загрузи, поставь робота на старт. Подбери `RUN_MS` так, чтобы робот останавливался до конца дорожки, и больше его не меняй.
4. Найди в готовой части функцию `record` и ответь в журнале: когда она добавляет переход через линию и зачем там порог 10?

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

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

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

3 Релейный против П

20 мин

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

Подсказка: Почему средняя по величине

Если один заезд случайно вышел плохим, среднее арифметическое изменится сильно, а средняя по величине — нет.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица для двух регуляторов заполнена, выбраны средние по величине, объяснение переходов записано.

ЗАНЯТИЕ 2 · 45 МИН

2 ПД, ПИД и общий итог

4 ПД: тормозим поворот заранее

14 мин

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

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

ПРЕДСКАЖИ

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

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

Подсказка: Где взять прошлую ошибку

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

Подсказка: Робот дёргается даже при маленьком kd

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

ДОЛЖНО ПОЛУЧИТЬСЯ

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

5 ПИД: что даёт накопленная ошибка

11 мин

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

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

Подсказка: Как не выпустить сумму за предел

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

ДОЛЖНО ПОЛУЧИТЬСЯ

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

6 Сравнение всех четырёх

12 мин

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

ДОЛЖНО ПОЛУЧИТЬСЯ

Итоговая таблица, обоснованный вывод и ответы — в журнале.

7 Задания по уровням

8 мин

● **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:
