Материалы ученика

Робот-курьер

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

Уровень итоговой задачи: **Продвинутый**

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

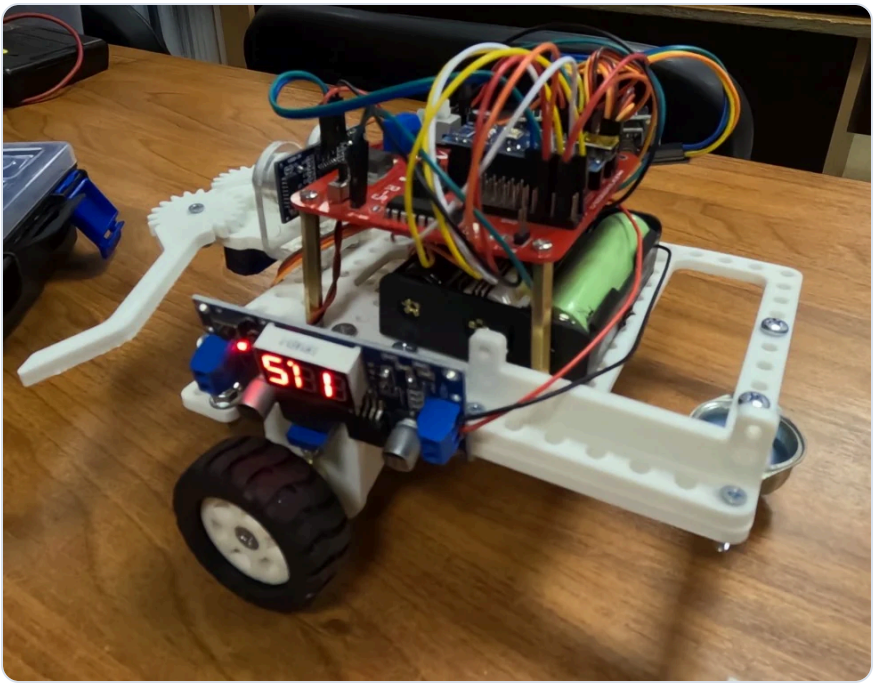
Ничего — это первый урок.

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.

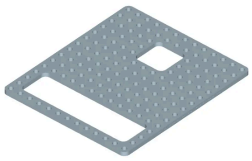
✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

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

2 Разложи детали

7 мин

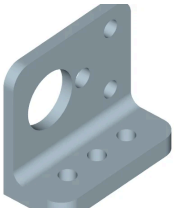
Прежде чем собирать, найди каждую деталь. Так ты не перепутаешь похожие детали и сразу увидишь, если чего-то не хватает.



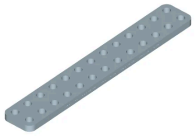
Основа 130 × 150
1 шт.



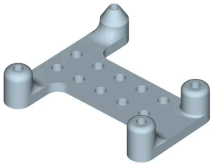
Кронштейн мотора
1 шт.



Кронштейн мотора (зеркальный)
1 шт.



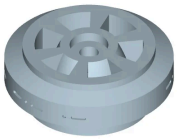
Планка датчиков
1 шт.



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

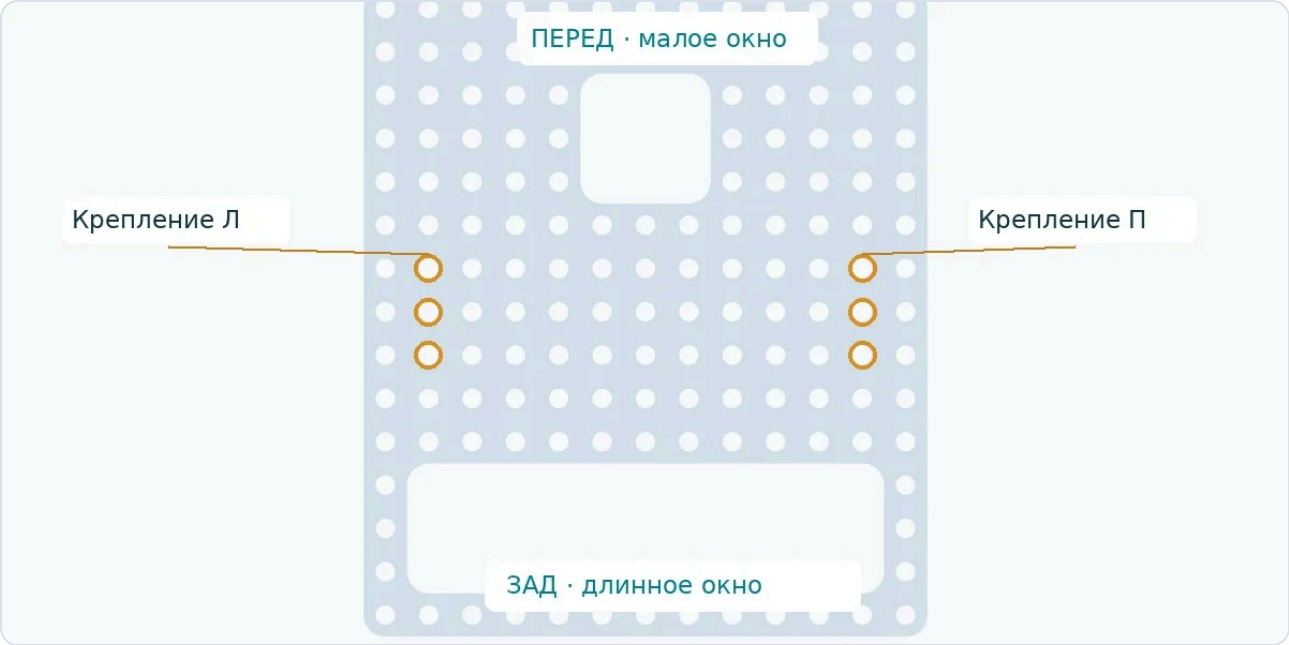
Не заменяй её похожей. Сверь пакет с карточками ещё раз; если детали нет — сообщи взрослому.

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.

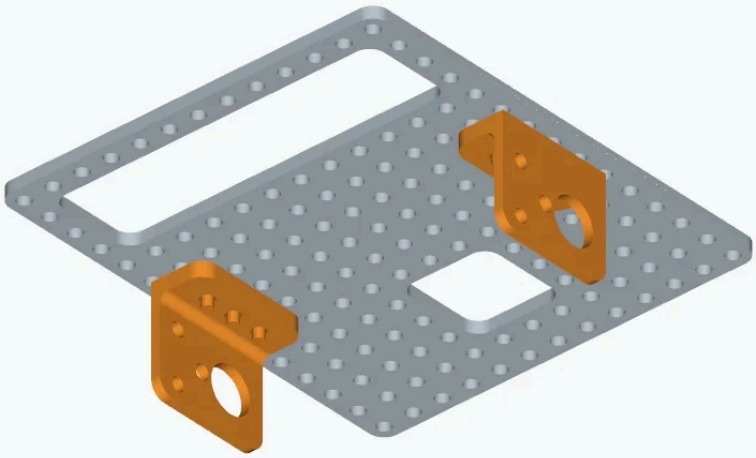
ДОЛЖНО ПОЛУЧИТЬСЯ

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.

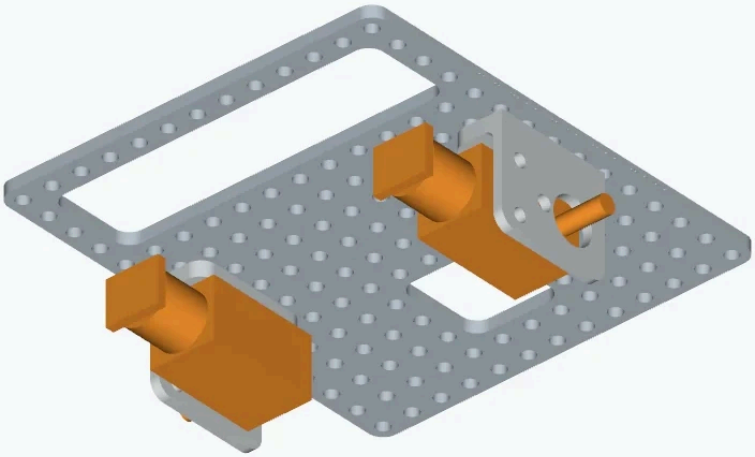
ПРОВЕРЬ СЕБЯ

- [ ] 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.

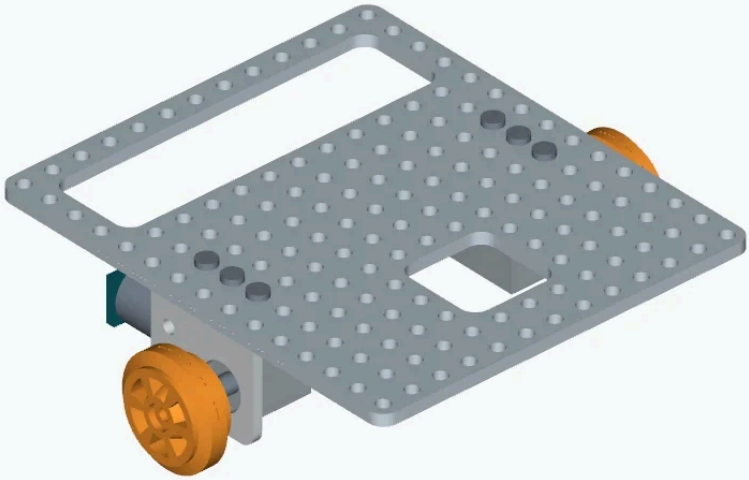
Подсказка: Мотор не встаёт в кронштейн

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.

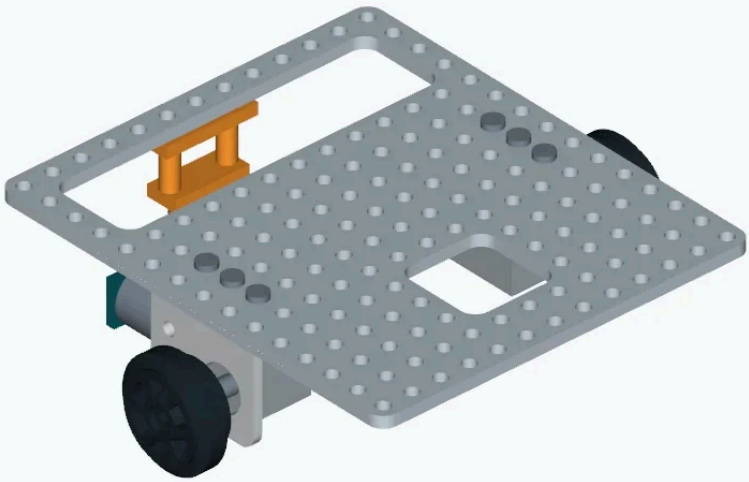
ДОЛЖНО ПОЛУЧИТЬСЯ

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.

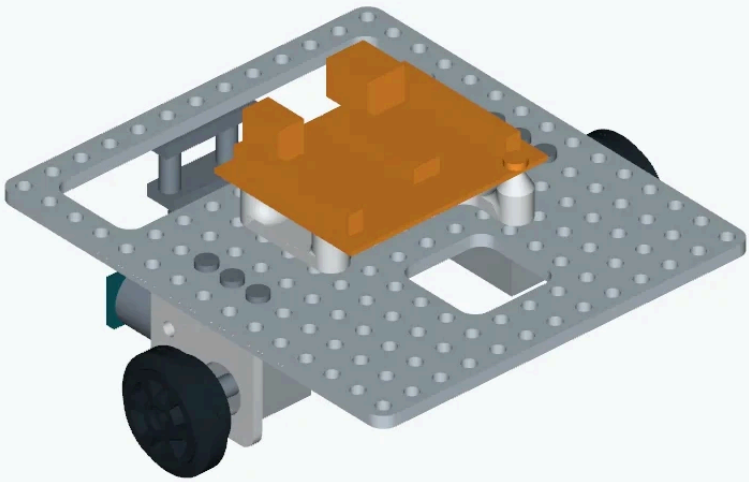
ЗАНЯТИЕ 2 · 45 МИН

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

1. Подключи USB-кабель к Arduino и к компьютеру. Батарейный отсек моторов остаётся отключённым.
2. В MyRobotica открой курс **«Робот-курьер»**, урок **1.1** и его учебный проект. В правой части кнопки **Загрузить на робота** выбери своего настоящего робота. Через **Подробнее о связи** проверь плату **Arduino Uno** и найди её порт.
3. Сохрани программу с пустыми `setup()` и `loop()`, нажми **Проверить код**, а потом **Загрузить на робота**.
4. Запиши в журнал: плату, порт, версию MyRobotica и чем **Проверить код** отличается от **Загрузить на робота** — что делает каждая кнопка и нужна ли для неё плата.

ДОЛЖНО ПОЛУЧИТЬСЯ

Пустой скетч проверен и загружен. В журнале — плата, порт, версия MyRobotica и разница двух кнопок.

Подсказка: Проверить код и Загрузить на робота

Отключи кабель и нажми обе кнопки по очереди. Какая сработала без платы?

11 Проверь загрузку программы

3 мин

Загрузи пустой скетч и прочитай сообщение об успешной загрузке.

Проверка связи с платой L11_Empty.ino

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

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

12 Надень шилд и подключи моторы

7 мин

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

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

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

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

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

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

13 Установи датчики линии

6 мин

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

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

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

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

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

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

14 Проверь все узлы робота

14 мин

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

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

Напиши программу сам. В заготовке есть только номера контактов и выключение моторов.

Напиши свою диагностику L11_Diagnostics_Empty.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;

// Напиши диагностику по требованиям урока. При включении моторы стоя
void setup() {
  pinMode(LEFT_PWM, OUTPUT);
  pinMode(RIGHT_PWM, OUTPUT);
  analogWrite(LEFT_PWM, 0);
  analogWrite(RIGHT_PWM, 0);
}

void loop() {
}
```

Что должна уметь программа:

- работать с монитором порта на скорости 9600;
- по букве `d` печатать подписанные числа обоих датчиков;
- по `1` и `2` крутить левое колесо вперёд и назад, по `3` и `4` — правое;
- каждый раз крутить со скоростью 120 не дольше 300 мс, потом выключать оба мотора;
- по букве `s` сразу выключать оба мотора, даже во время вращения;
- после включения и кнопки Reset моторы стоят, пока не придёт новая команда;

- лишние нажатия не копятся в очередь и не продлевают вращение.

> **Подсказка: Как обойтись без delay**
>
> Запомни время начала вращения через `millis()`. В каждом проходе `loop()` проверяй, не пришла ли `s` и не прошло ли 300 мс.

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

★★ Продвинутый уровень

1. Как обеспечить обработку s во время импульса?

- ○ Проверять время без блокирующего ожидания
- ○ Ждать весь импульс в delay
- ○ Отключить последовательный порт

2. Что делает скетч после Reset?

- ○ Ждёт новой команды с выключенными моторами
- ○ Повторяет последний импульс
- ○ Запускает все тесты автоматически

3. Что делать с накопленными командами движения?

- ○ Выполнить все подряд
- ○ Не создавать очередь запусков
- ○ Продлить текущий импульс

4. Можно ли без энкодеров автоматически подтвердить вращение?

- ○ Да, по значению PWM
- ○ Нет, нужна обратная связь или наблюдение человека
- ○ Да, по числу отправленных команд

5. Все программные проверки прошли, но колесо не крутится. Итог?

- ○ Робот исправен
- ○ Достаточно проверить светодиод
- ○ Узел не принят; нужна диагностика и повторная физическая проверка

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

Левый мотор подключён к контактам `7` (направление) и `6` (скорость). Направление «вперёд» на этом комплекте — уровень `HIGH`.

Как это устроено: Справка по командам

Команда	Что делает
`pinMode(pin, OUTPUT)`	Разрешает плате управлять контактом
`digitalWrite(pin, HIGH)` / `LOW`	Направление мотора
`analogWrite(pin, v)`	Сила мотора: `v` от 0 (стоит) до 255
`delay(ms)`	Ждать `ms` миллисекунд

Требование 1. Скетч `L21_Left_Motor`: левое колесо крутится вперёд ровно 2 секунды с силой из именованной переменной и останавливается само. Номера контактов — именованные константы, а не числа в командах.

Приёмка.

1. На подставке напарник засекает по секундомеру, сколько крутится колесо.
2. Замени уровень направления на противоположный. Что сделало колесо? Запиши, какой уровень на твоём роботе — «вперёд».
3. Объясни в журнале, почему колесо остановилось, хотя `loop()` пустой.

Подсказка: Идея (1 из 2)

Один раз при включении: настроить контакты, выбрать направление, включить, подождать, выключить.

Подсказка: Подсказка к пункту 3 (2 из 2)

Где выполнилась команда остановки — в `setup()` или в `loop()`? Сколько раз?

ДОЛЖНО ПОЛУЧИТЬСЯ

Колесо крутится 2 секунды и останавливается. В журнале — время по секундомеру, уровень «вперёд» и объяснение остановки.

3 Включи оба мотора

15 мин

Требование 2. Оба колеса крутятся вперёд ровно одну секунду со скоростью из одной именованной переменной `speed` и останавливаются. Правый мотор подключён к контактам `4` (направление) и `5` (скорость). Начни со своей программы для левого мотора и сохрани её как `L21_Both_Motors`.

Исследование: с какого числа колесо трогается. На подставке ставь `speed` = 20, 40, 60, 80… и запиши для каждого колеса, при каком наименьшем числе оно начинает уверенно крутиться.

speed	Левое крутится?	Правое крутится?
20		
40		
60		

ПРЕДСКАЖИ

Будут ли эти числа одинаковыми у двух колёс? А у тебя и у соседней команды?

Подсказка: Колесо гудит, но не крутится

Мотору не хватает силы сдвинуть редуктор. Такое число — ещё «мёртвая зона».

ДОЛЖНО ПОЛУЧИТЬСЯ

Оба колеса крутятся одну секунду и останавливаются. В журнале — наименьшая скорость трогания для каждого колеса и вывод, какие числа `speed` брать нельзя.

4 Найди ошибку остановки

10 мин

НАЙДИ ОШИБКУ · ПУСТОЙ LOOP(), А РОБОТ ЕДЕТ

У соседей `loop()` пустой, а движение — в `setup()`:

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

void setup() {
  // … настройка контактов
  forward(1000);
}

void loop() {
}
```

Робот должен был проехать одну секунду, а едет, пока его не выключат. Почему, если `loop()` пустой? Запиши ответ в журнал.

Требование. Перепиши программу так, чтобы движение осталось в `loop()`, но робот проехал ровно одну секунду и больше не трогался. `while (true)` использовать нельзя.

1. Проверь на подставке: колёса крутятся одну секунду и стоят, пока не нажмёшь Reset.
2. Объясни в журнале, как твоя программа «помнит», что движение уже было.

Подсказка: Идея

Нужна переменная, которая хранит «уже ехали» — `bool`. Её проверяют в начале `loop()`.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Программа выполняет требование, объяснение записано.

2

ЗАНЯТИЕ 2 · 45 МИН

Функции движения

5 Оформи движения функциями

15 мин

Требование 3. Программа `L21_Moves` построена из функций:

- `drive(leftDir, leftSpeed, rightDir, rightSpeed)` — единственное место, где задаются направление и скорость колёс;
- `forward(ms)` и `backward(ms)` — движение на заданное время, `stopMotors()` — остановка;
- направление «назад» не записано числом, а получено из `FORWARD`: на другом комплекте `FORWARD` может быть другим;
- после включения есть 2 секунды, чтобы убрать руки;
- в `setup()`: вперёд 1 секунду, стоп, пауза полсекунды, назад 1 секунду, стоп.

Подсказка: Идея (1 из 3)

Сначала одна общая функция, которая ставит направление и скорость обоих колёс. Остальные функции только вызывают её с нужными значениями.

Подсказка: План (2 из 3)

1. `drive` — четыре строки из своей программы, но с параметрами вместо чисел.
2. `forward(ms)` — `drive` вперёд и `delay(ms)`.
3. `backward(ms)` — то же назад. `stopMotors()` — скорость 0.

Подсказка: Фрагмент кода (3 из 3)

`const int BACKWARD = (FORWARD == HIGH) ? LOW : HIGH;` — прочитай как «если FORWARD равен HIGH, то LOW, иначе HIGH».

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование 3 и проверяется без ошибок.

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.

Узнай, насколько прямо едет твой робот, и попробуй это исправить **одним изменением**.

1. Проверь программу на подставке, потом поставь робота на пол у начала метровой полосы скотча. Проедь вперёд 2 секунды. Измерь, на сколько сантиметров нос ушёл вбок от полосы. Повтори три раза.
2. Предложи одно изменение, которое уменьшит увод: например, разные скорости левого и правого колеса. Запиши, почему ты ожидаешь улучшения.
3. Сделай изменение и снова три заезда. Стало лучше? На сколько?

	Увод 1, см	Увод 2, см	Увод 3, см
До			
После			

Подсказка: Колёса разные

Если робот уходит вправо, какое колесо крутится быстрее?

ДОЛЖНО ПОЛУЧИТЬСЯ

Две серии по три заезда, одно изменение с обоснованием и вывод по числам.

7 Измени программу

10 мин

Требование 4. Функция `shuttle(times, ms)` делает «челнок» `times` раз: вперёд `ms` миллисекунд, стоп, назад, стоп. Вызовы движений в ней написаны один раз.

Исследование: сколько проезжает робот.

1. Три раза запусти `forward(1000)` при `speed = 100` и измерь путь линейкой в миллиметрах. Повтори при `speed = 200`.
2. Посчитай, сколько миллиметров в секунду проезжает робот при каждой скорости. Во сколько раз отличаются результаты? Вдвое ли, как отличаются числа `speed`?
3. Подбери время `backward` так, чтобы после одного челнока робот вставал не дальше 2 см от старта. Запиши, почему вперёд и назад время получилось разным.

`speed`	Путь 1, мм	Путь 2, мм	Путь 3, мм	мм за секунду
100				
200				

Подсказка: Пути разные при одной программе

Это нормально: колёса проскальзывают, батарея садится. Поэтому меряем три раза и берём среднее.

ДОЛЖНО ПОЛУЧИТЬСЯ

`shuttle` работает. В журнале — таблица путей, скорость в мм/с для двух `speed`, вывод про связь числа `speed` и настоящей скорости и подобранное время возврата.

ПРОВЕРЬ СЕБЯ

- [ ] Я могу объяснить, почему одинаковая программа даёт разный путь.

- [ ] Я знаю, почему без `stopMotors()` робот не остановится.

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · ВОЗВРАТ НА СТАРТ**

Измерь, сколько миллиметров робот проезжает за секунду, и подбери время назад так, чтобы после челнока он вставал не дальше 2 см от старта. Запиши, почему время вперёд и назад разное.

Проверь себя

★★ Продвинутый уровень

1. Робот едет «прямо», но плавно уходит вправо. Что проверить первым?

- ○ Немного уменьшить скорость левого колеса или увеличить скорость правого
- ○ Поменять местами провода обоих моторов
- ○ Увеличить время езды

2. При speed = 30 колесо гудит, но не крутится. Что это значит для программы?

- ○ Ниже порога трогания мотор не сдвигает редуктор — такие значения speed брать нельзя
- ○ Мотор неисправен, его нужно заменить
- ○ analogWrite не принимает чисел меньше 50

3. Почему направление «назад» получают из FORWARD, а не пишут числом?

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

- ○ Так программа работает быстрее
- ○ На другом комплекте FORWARD может быть другим, и «назад» поменяется само
- ○ Константе нельзя присвоить LOW

4. При speed = 100 робот за 1 секунду проехал 180, 200 и 190 мм. Какую скорость записать в журнал?

- ○ 200 мм/с — самый большой путь
- ○ 100 мм/с — как число speed
- ○ 190 мм/с — среднее трёх замеров

5. В чём ошибка функции?

```
void shuttle(int times, int ms) {
  for (int i = 0; i < times; i++) {
    forward(ms);
    backward(ms);
  }
  stopMotors();
}
```

- ○ Между движениями нет остановки: робот резко меняет направление, и путь назад не совпадает с путём вперёд
- ○ Цикл выполнится на один раз больше, чем times
- ○ stopMotors() нельзя вызывать после цикла

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.

Требование. В своей программе `L21_Moves` из урока 2.1 сделай пробу поворота: колёса крутятся в разные стороны с одной скоростью `turnSpeed` в течение `turnMs`, потом робот останавливается. Сохрани как `L22_Turn_Test`.

Исследование: вокруг чего крутится робот.

1. Положи робота на лист бумаги и обведи карандашом оба колеса.
2. Запусти поворот и обведи колёса снова. Найди точку, вокруг которой повернулся робот.
3. Совпадает ли она с серединой между колёсами? Если нет — насколько сантиметров и почему?

Подсказка: Как найти точку поворота

Соедини старое и новое положение каждого колеса. Точка поворота лежит там, где эти положения одинаково далеко.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Проба поворота работает. В журнале — рисунок, точка поворота и объяснение, почему она там.

2 Подбери время поворота на 90°

20 мин

Подбери время поворота **методом** и узнай, насколько ему можно доверять.

1. Найди одно время, при котором робот явно не доворачивает, и одно, при котором явно перекручивает. Пробуй середину между ними и сужай промежуток, пока ошибка не станет меньше 5°.
2. С найденным временем сделай пять поворотов подряд. Для каждого отметь на листе, куда смотрит нос, и измерь транспортиром ошибку в градусах.
3. Повтори пункт 2 при другой `turnSpeed` (например, 160). При какой скорости повороты точнее и одинаковее?
4. Повернёт ли робот направо за то же время? Проверь и запиши.

`turnSpeed`	`turnMs`	Ошибки 5 поворотов, °	Самая большая ошибка, °
100			
160			

Подсказка: Как измерить ошибку без транспортира

Сравни отметку носа с линией 90° на расстоянии 10 см от центра поворота: 1 см в сторону — это примерно 6°.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале — способ поиска, время для двух скоростей, ошибки пяти поворотов и вывод: какую `turnSpeed` брать и одинаково ли время налево и направо.

3 Разворот на 180°

15 мин

1. Предскажи время разворота по своему времени на 90° и проверь на полу. Подбери точное время тем же методом, что и для 90°.
2. Сравни: во сколько раз время на 180° больше времени на 90°? Объясни в журнале, откуда берётся разница. Назови две причины.
3. Выведи правило: сколько миллисекунд нужно на один градус при развороте и при повороте на 90°? Какое из чисел точнее подойдёт для поворота на 45°? Проверь.

ДОЛЖНО ПОЛУЧИТЬСЯ

Время разворота подобрано. В журнале — сравнение с удвоенным поворотом на 90°, две причины разницы и проверка правила на 45°.

2

ЗАНЯТИЕ 2 · 45 МИН

Функции поворотов

4 Функции поворотов

12 мин

Требование. Продолжи свою программу `L21_Moves` из урока 2.1 и сохрани как `L22_Turns`:

- `turnLeft()`, `turnRight()`, `turnAround()` — повороты на 90° и разворот по твоим временам, каждый заканчивается остановкой;
- времена — именованные переменные в начале скетча; если в шаге подбора время налево и направо получилось разным, у каждого поворота своя переменная;
- направление и скорость колёс задаёт только `drive`;
- в `setup()` — квадрат для следующего шага: четыре раза «вперёд 1 секунду, налево».

Подсказка: Идея (1 из 2)

Поворот — это `drive` с колёсами в разные стороны, ожидание и остановка.

Подсказка: Фрагмент кода (2 из 2)

`drive(BACKWARD, turnSpeed, FORWARD, turnSpeed);` — так робот крутится налево. Что поменять для поворота направо?

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование и проверяется без ошибок.

5 Квадрат

18 мин

1. Запусти квадрат три раза. Каждый раз измерь, насколько робот промахнулся мимо старта (в сантиметрах) и насколько повёрнут нос (в градусах).
2. Оцени ошибку одного поворота: раздели ошибку направления в конце на четыре. Совпадает ли она с ошибкой, которую ты мерил при подборе времени?
3. Что сильнее портит квадрат: ошибка поворотов или увод на прямой? Как это проверить отдельным опытом? Проведи его и запиши вывод.

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?

ДОЛЖНО ПОЛУЧИТЬСЯ

Замеры трёх квадратов, оценка ошибки одного поворота и опыт, который показал главную причину промаха.

6 Задания по уровням

15 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · ПОВОРОТ НА ЛЮБОЙ УГОЛ**

Напиши функцию `turn(int degrees)`: положительный угол — налево, отрицательный — направо. Время считай как `degrees × msPerDegree`, где `msPerDegree = turn90Ms / 90`. Проверь на 45°, 90° и 180°. Где точность хуже всего и почему?

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

★★ Продвинутый уровень

1. turn90Ms = 600. Сколько миллисекунд займёт turn(45), если msPerDegree = turn90Ms / 90?

- ○ 300
- ○ 450
- ○ 600

2. **Как функция turn(int degrees) понимает, куда поворачивать?**

- ○ По знаку угла: положительный — налево, отрицательный — направо
- ○ По датчику, который видит линию
- ○ Всегда поворачивает налево

3. **Ошибки пяти поворотов: при turnSpeed = 100 — +4°, −6°, +5°, −3°, +2°; при 160 — +1°, −2°, +2°, 0°, −1°. Какую скорость выбрать?**

- ○ Любую: ошибки одинаковые
- ○ 160: самая большая ошибка меньше
- ○ 100: чем медленнее, тем всегда точнее

4. **При 300 мс робот явно не доворачивает, при 500 мс — явно перекручивает. Что попробовать дальше?**

- ○ 600 мс
- ○ Снова 300 мс
- ○ 400 мс — середину между ними

5. turn90Ms = 400. Для 45° по коэффициенту из 90° вышло 200 мс, а робот повернул только на 35°. Почему?

- ○ Разгон занимает одинаковое время в каждом повороте, и в коротком повороте его доля больше
- ○ Коэффициент нельзя делить на 90
- ○ turn() не умеет поворачивать на нечётные углы

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.

Подготовь дорожку для своего уровня и проверь, подходит ли она твоему роботу.

1. Сделай дорожку: прямая, плавный изгиб, крутая дуга и перекрёсток. Подготовь белую и чёрную карточки.
2. Измерь линейкой ширину линии и расстояние между датчиками (между серединами окошек).
3. Поставь робота на линию. Сколько миллиметров остаётся от края линии до каждого датчика? Запиши в журнал: что будет, если линия шире расстояния между датчиками?

ДОЛЖНО ПОЛУЧИТЬСЯ

Дорожка готова, в журнале — ширина линии, расстояние между датчиками, запас с каждой стороны и ответ на вопрос.

2 Посмотри, что видит датчик

12 мин

Датчик светит на пол и меряет, сколько света вернулось. Левый датчик подключён к `A4`, правый — к `A5`.

Как это устроено: Справка по командам

Команда	Что делает
`analogRead(pin)`	Показание датчика: число от 0 до 1023
`Serial.begin(9600)`	Включает связь с компьютером, один раз в `setup()`
`Serial.print(x)` / `Serial.println(x)`	Печатает в монитор порта; `println` — с переходом строки
`millis()`	Сколько миллисекунд прошло с включения платы

Требование. Скетч `L31_Read_Sensors` пять раз в секунду печатает строку `t=… L=… R=…`, где `t` — время в миллисекундах с включения.

Приёмка.

1. По числам `t` в мониторе проверь, что строки идут примерно каждые 200 мс. На сколько миллисекунд отличается промежуток от 200 и почему?
2. Поднеси под левый датчик белую, затем чёрную карточку на рабочей высоте. Запиши, на сколько меняется число.

Подсказка: Почему промежуток не ровно 200

Сколько времени занимают сами чтение и печать, кроме `delay(200)`?

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование. В журнале — реальный промежуток между строками с объяснением и изменение числа на белом и чёрном.

3 Проверь оба датчика

8 мин

Сними **профиль** датчиков: как меняются числа, пока линия проходит под роботом поперёк.

1. Положи робота поперёк линии так, чтобы линия была далеко справа от правого датчика.
2. Сдвигай робота влево шагами по 5 мм, пока линия не уйдёт за левый датчик. На каждом шаге записывай L и R.

Сдвиг, мм	L	R
0		
5		
…		

3. Нарисуй в журнале два графика: L и R от сдвига. Найди по графику, где линия ровно между датчиками и на сколько миллиметров можно уйти в сторону, пока датчик её не заметит.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица и два графика. В журнале — ширина «слепой зоны» между датчиками в миллиметрах.

4 Измерь белое и чёрное

12 мин

Откалибруй датчики и докажи, что калибровке можно доверять.

1. Для каждого датчика запиши по 10 показаний на белом и 10 на чёрном. Найди наименьшее и наибольшее в каждой серии — это **разброс**.
2. Сравни разброс с разницей белого и чёрного. Во сколько раз разница больше разброса?
3. Повтори белое у окна и в тёмном углу класса. Меняются ли числа сильнее разброса?
4. Запиши вывод: где можно доверять калибровке и когда её нужно повторить.

Датчик	Белое: мин–макс	Чёрное: мин–макс	Разница типичных
Левый			
Правый			

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Таблица разбросов, сравнение с разницей и вывод, когда калибровку повторять.

2

ЗАНЯТИЕ 2 · 45 МИН

Робот едет по линии

5 Научи программу замечать линию

10 мин

Требование 1. Функция `onLine(value, white, black)` отвечает `true`, когда под датчиком линия:

- показание ближе к чёрному измерению, чем к белому;
- работает и когда у датчика чёрное — большое число, и когда маленькое;
- внутри нет готовых чисел-порогов: только `value`, `white` и `black`.

ПРЕДСКАЖИ

Почему нельзя просто написать `value > 500`? Назови два случая, когда такое правило ошибётся.

Приёмка. Напиши `onLine` в тестовом скетче, загрузи его и открой монитор порта (9600). Тест проверяет восемь случаев для двух видов датчиков, в том числе показания за пределами твоих измерений.

Тест функции onLine L31_OnLine_Test.ino

```
// Проверка функции onLine. Напиши функцию здесь, загрузи и открой мо
// Когда все строки PASS, перенеси onLine в свою программу.

// Требование 1: true, если показание ближе к чёрному, чем к белому.
bool onLine(int value, int white, int black) {
  return false;
}

int failed = 0;

void expect(int value, int white, int black, bool expected) {
  bool actual = onLine(value, white, black);
  Serial.print(actual == expected ? "PASS " : "FAIL ");
  Serial.print("onLine(");
  Serial.print(value);
  Serial.print(", ");
  Serial.print(white);
  Serial.print(", ");
  Serial.print(black);
  Serial.print(") = ");
  Serial.print(actual ? "true" : "false");
  Serial.print(", ожидаем ");
  Serial.println(expected ? "true" : "false");
  if (actual != expected) {
    failed++;
  }
}

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

  // Датчик, у которого чёрное — большое число, а белое — маленькое.
  expect(780, 120, 780, true);
  expect(120, 120, 780, false);
  expect(600, 120, 780, true);
  expect(300, 120, 780, false);
  // За пределами измерений.
  expect(900, 120, 780, true);
  expect(50, 120, 780, false);
```

```
  // Датчик, у которого наоборот: белое — большое число, чёрное — мале
  expect(300, 900, 120, true);
  expect(700, 900, 120, false);

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

void loop() {
}
```

Когда все строки `PASS`, добавь `onLine` в свою программу из шага «Посмотри, что видит датчик» и сохрани её как `L31_Line_Detect`. Пусть она печатает «слева линия» и «справа линия». Впиши свои измерения и проверь карточкой.

Подсказка: Идея (1 из 3)

Нужны два расстояния: от показания до чёрного и от показания до белого. Какое меньше?

Подсказка: План (2 из 3)

1. Расстояние до чёрного — разница без знака.
2. Расстояние до белого — так же.
3. Вернуть результат сравнения двух расстояний.

Подсказка: Фрагмент кода (3 из 3)

`abs(value - black)` — расстояние до чёрного. Зачем здесь `abs`? Что будет без него у датчика, где чёрное — маленькое число?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Тест показывает «Все проверки пройдены». Твоя программа печатает «слева линия» и «справа линия», только когда под датчиком чёрное.

6 Придумай, как поворачивать

7 мин

Робот едет вперёд, и линия должна оставаться между датчиками. **Составь правила сам**: что делают колёса в каждом из четырёх случаев и почему.

Что видят датчики	Левое колесо	Правое колесо	Куда едет робот	Почему
Оба белое				
Левый — линия				
Правый — линия				
Оба — линия				

1. Проверь каждую строку на выключенном роботе: сдвинь его рукой и покажи, куда он должен поехать.
2. «Оба белое» бывает в двух разных положениях робота. В каких? Может ли программа их различить? Запиши, что тогда случится с роботом.
3. На прямой дорожке «оба — линия» почти не бывает. Где на итоговом поле это встретится и что тогда должен делать робот?

Подсказка: Два случая «оба белое»

Подумай, где может быть линия, когда ни один датчик её не видит.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица из четырёх строк с объяснением. В журнале записано, в каком положении программа не знает, где линия.

7 Запиши правила в программу

10 мин

Возьми каркас: в нём контакты, моторы и функция `drive` из урока 2.1. Перенеси в него свои измерения и функцию `onLine` и сохрани как `L31_First_Line`. Эта программа понадобится тебе в уроке 3.2.

Каркас движения по линии L31_First_Line_Skeleton.ino

```
// Каркас движения по линии. Требования к программе — в тексте урока.
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;

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);
}

// Требование 1: перенеси функцию onLine из тестового скетча.
bool onLine(int value, int white, int black) {
  return false;
}

void setup() {
  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);

  // Требование 3: защита от запуска без измерений и время убрать руки
}

void loop() {
```

```
40    // Требование 2: прочитать датчики, решить по своей таблице правил,
41    drive(FORWARD, 0, FORWARD, 0);
42  }
```

Требование 2. Колёса выполняют твою таблицу правил:

- в каждом повторе программа заново читает оба датчика и решает заново: колесо, остановленное один раз, снова едет, когда его датчик видит белое;
- решение принимает твоя функция `onLine`;
- скорость — одна именованная переменная `speed` в начале скетча.

Требование 3. Робот безопасен:

- без измерений белого и чёрного колёса стоят;
- после включения есть 2 секунды, чтобы убрать руки.

Приёмка — на подставке в следующем шаге: робот выполняет каждую строку твоей таблицы правил, а без измерений колёса не крутятся.

ПРЕДСКАЖИ

Что сделает робот, если скорости обоих колёс задать один раз в `setup()`? Запиши ответ до проверки.

Подсказка: Идея (1 из 3)

Каждый повтор: прочитать датчики → решить → отправить скорости на моторы. Решение начинается с «оба едут», а правила его меняют.

Подсказка: План (2 из 3)

1. Прочитать оба датчика.
2. Спросить `onLine` для каждого.
3. Обе скорости — `speed`.
4. Правила из таблицы меняют скорости.
5. Отправить скорости в `drive`.

Подсказка: Фрагмент кода (3 из 3)

`if (onLine(leftValue, LEFT_WHITE, LEFT_BLACK)) { leftSpeed = 0; }` — а для правого датчика? А строка «оба — линия» из твоей таблицы?

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требования 2 и 3, **Проверить** проходит без ошибок.

8 Проверь на подставке

6 мин

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

Приёмка требований 2–3 на подставке.

1. Загрузи программу с нулями вместо измерений. Колёса не должны крутиться. Впиши измерения и загрузи снова.
2. Проверь каждую строку своей таблицы правил карточками.
3. Засеки, через сколько секунд после включения трогаются колёса.

Проверка	Ждал	Увидел
Без измерений		
Оба белое		
Левый — линия		
Правый — линия		
Оба — линия		
Пауза на старте		

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Все проверки совпали с требованиями. Если нет — исправлено и проверено снова.

9 Первый заезд по линии

8 мин

1. Выключи питание моторов, отключи USB-кабель. Будь готов выключить питание, если робот уходит с дорожки.
2. Найди наибольшую `speed`, при которой робот проходит плавный изгиб **три раза из трёх**. Начни с 80 и поднимай шагами по 20.
3. На наибольшей скорости, которую робот не выдержал, посмотри, где он теряет линию. Объясни в журнале причину через твою таблицу правил.
4. Запиши, что нужно изменить в правилах, чтобы робот держал линию быстрее. Проверишь это в уроке 3.2.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Наибольшая надёжная скорость, место и причина схода, идея улучшения правил.

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · НЕ СТОП, А ЗАМЕДЛЕНИЕ**

Вместо остановки колеса замедляй его: например, до трети от `speed`. Сравни на одном изгибе два варианта — «стоп» и «замедление». Какой плавнее? При какой скорости каждый вариант ещё держит линию? Запиши результаты в журнал: это первый шаг к П-регулятору, который понадобится на крутых дугах продвинутого поля.

Проверь себя

★★ Продвинутый уровень

1. **speed = 150. По заданию ★★ колесо не останавливается, а замедляется до трети скорости. Какое значение получит колесо?**

- ○ 50
- ○ 0
- ○ 100

2. **Чем замедление колеса лучше остановки?**

- ○ Робот поворачивает плавнее и меньше дёргается
- ○ Робот больше никогда не теряет линию
- ○ Батарея садится вдвое медленнее

3. **Тест onLine не проходит для датчика «наоборот» (white = 900, black = 120). Почему?**

```
bool onLine(int value, int white, int black) {
  return value > (white + black) / 2;
}
```

- ○ bool не может хранить результат сравнения
- ○ Функция считает линией большое число, а у этого датчика чёрное — маленькое
- ○ Деление на 2 даёт дробное число

4. Оба датчика видят белое. В каких двух положениях может быть робот?

- ○ Робот на перекрёстке или на дуге
- ○ Только линия ровно между датчиками
- ○ Линия между датчиками или линия потеряна за обоими датчиками

5. Что будет делать робот с этой программой?

```
void setup() {
  // … настройка моторов
  leftSpeed = speed;
  rightSpeed = speed;
}

void loop() {
  readSensors();
  if (onLine(leftValue, LEFT_WHITE, LEFT_BLACK)) {
    leftSpeed = 0;
  }
  drive(FORWARD, leftSpeed, FORWARD, rightSpeed);
}
```

- ○ Колесо, остановленное один раз, так и стоит — робот кружит вокруг него
- ○ Робот поедет прямо и не будет поворачивать
- ○ Программа не пройдёт проверку

Отвечай на вопросы своего уровня. Нужно верных ответов: Начальный — 4 из 5; Средний — 4 из 5; Продвинутый — 4 из 5. Ответы проверит преподаватель или программа курса.

Итог

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

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

Как подтвердить результат

Для каждого опыта сохраняй условия, версию программы, наблюдение и вывод. Сравнивая попытки, меняй одну причину за раз; неудачные попытки тоже остаются в журнале. Отметка результата относится к уже выполненным действиям, повторять их ради отметки не нужно. Если результат не получен, запиши расхождение и следующую проверяемую гипотезу. Модель, измерения настоящего робота и объяснение — разные виды доказательств.

Словарь урока

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

Тема 3 · Урок 3.2

П-регулятор: плавно и быстро

Робот поворачивает тем сильнее, чем дальше ушла линия, и проходит дуги без рывков.

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

РЕЗУЛЬТАТ УРОКА

Робот плавно едет по линии с П-регулятором и проходит дугу своего уровня. Коэффициент и скорость подобраны и записаны в журнал.

Ты научишься

- Перевести показания двух датчиков в общую шкалу от 0 до 100.
- Посчитать ошибку как разницу показаний и объяснить её знак.
- Объяснить, как поправка меняет скорости колёс.
- Подобрать коэффициент и скорость по наблюдениям на дорожке.

Нужно знать

- 3.1 Датчики линии и первое следование
- 2.1 Моторы: едем и останавливаемся

Понадобится

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

Зачем этот урок

В уроке 3.1 робот ехал по линии рывками: колесо либо крутилось, либо стояло. На крутых дугах итоговой задачи так не проехать. Сегодня робот научится

поворачивать **чуть-чуть**, когда линия ушла немного, и **сильно**, когда ушла далеко. Так работают автопилоты и роботы-курьеры.

РАЗМИНКА · ВСПОМНИ

1. Как программа из урока 3.1 решала, что под датчиком линия?
2. Почему робот в уроке 3.1 ехал рывками?
3. Что делает функция `drive()`?

1

ЗАНЯТИЕ 1 · 45 МИН

Ошибка вместо «да или нет»

1 Плавно — это как?

8 мин

Что должен делать робот. Роботу нужно двигаться по линии плавнее: при маленьком отклонении чуть поправлять курс, а при большом — сильнее. Используем те же датчики отражения и моторы, добавив П-регулятор — правило расчёта поправки.

Что используем и как это работает. Сначала переводим измерения каждого датчика в общую шкалу по его белому и чёрному. Разность уровней называется ошибкой. Коэффициент kp умножает ошибку: чем больше отклонение, тем больше поправка. Одному колесу прибавляем её к базовой команде, у другого вычитаем. Новое измерение позволяет снова скорректировать движение — это обратная связь.

Как связать измерение с действием. Слишком малая поправка не успевает вернуть робота, слишком большая вызывает колебания. Меняем один параметр за раз и сравниваем траекторию и показания на одинаковой трассе. Нулевая ошибка означает равные показания, но не всегда правильное положение: оба датчика могут потерять линию.

Исследуй проблему, прежде чем решать.

1. Запусти свою программу из урока 3.1 на дуге 350 мм, потом на дуге 125 мм. Запиши, как едет робот: рывки, виляние, сход с линии.
2. Объясни в журнале, какую информацию с датчиков программа 3.1 выбрасывает и почему из-за этого робот едет рывками.
3. **До того как читать дальше**, придумай своё правило: как скорости колёс должны зависеть от показаний датчиков. Запиши словами или формулой. В конце урока сравни его с П-регулятором.

Подсказка: Если не знаешь, с чего начать

Программа 3.1 сравнивает показание с порогом и получает «да или нет». А что, если использовать само число?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — наблюдение на двух дугах, объяснение причины рывков и твоё правило управления.

2 Общая шкала для двух датчиков

15 мин

Ты не начинаешь с чистого листа: работай в **своей программе из урока 3.1**. В ней уже есть контакты, измерения белого и чёрного и защита от запуска без измерений. Сохрани её копию как `L32_P_Line` — за урок ты превратишь её в П-регулятор, шаг за шагом, по требованиям.

Подсказка: Нет своей программы из урока 3.1?

Возьми каркас: в нём только контакты, управление моторами и защита. Всё остальное напишешь сам.

Каркас П-регулятора L32_P_Line_Skeleton.ino

```
// Наблюдение в MyRobotica включается сборщиком; обычный Ardui
#include <math.h>
#if defined(ROBOT_TELEMETRY_BUILD_ID)
#include <RobotTelemetry.h>
robot_telemetry::TelemetryStream courseTelemetry;
#endif

// Каркас П-регулятора. Требования к программе — в тексте урок
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;

void drive(int leftDir, int leftSpeed, int rightDir, int right
  digitalWrite(LEFT_DIR_PIN, leftDir);
  digitalWrite(RIGHT_DIR_PIN, rightDir);
  analogWrite(LEFT_SPEED_PIN, leftSpeed);
  analogWrite(RIGHT_SPEED_PIN, rightSpeed);
}

// Требование 1: шкала «чёрное — 0, белое — 100».
int level(int value, int white, int black) {
  return 0;
}

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

```
  pinMode(RIGHT_DIR_PIN, OUTPUT);
  pinMode(RIGHT_SPEED_PIN, OUTPUT);
  drive(FORWARD, 0, FORWARD, 0);

  // Без измерений робот не поедет. Эту защиту не удаляй.
  if (LEFT_WHITE == LEFT_BLACK || RIGHT_WHITE == RIGHT_BLACK) {
    while (true) {
      courseTelemetryPoll();
    }
  }
  courseWait(2000);
}

void loop() {
  float observedError = NAN;
  float observedOutput = NAN;
  // Требования 2 и 3: ошибка, поправка и скорости колёс.
  // Для графиков сохрани свои результаты в observedError и ob
  drive(FORWARD, 0, FORWARD, 0);
  courseTelemetryReport(analogRead(LEFT_SENSOR_PIN), analogRea
                        observedError, observedOutput);
}

// ===== Готовое: передача наблюдений — не меняй =====
void courseTelemetryBegin() {
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  using namespace robot_telemetry::wire;
  courseTelemetry.begin((1UL << Line0) | (1UL << Line1) |
                        (1UL << LineError) | (1UL << ControlOu
#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 по
```

```
void courseTelemetryReport(int leftRaw, int rightRaw, float er
#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);
  }
}
```

Как это устроено: Справка по командам

Команда или тип	Что делает
`analogRead(pin)`	Показание датчика: 0…1023
`analogWrite(pin, v)`	Скорость мотора: `v` от 0 до 255
`digitalWrite(pin, HIGH)` / `LOW`	Направление мотора
`constrain(x, a, b)`	`x`, но не меньше `a` и не больше `b`
`Serial.begin(9600)`	Включает монитор порта, пишут один раз в `setup()`
`Serial.print(x)` / `Serial.println(x)`	Вывод в монитор порта; `println` — с переводом строки
`int` на Arduino Uno	Целые от −32 768 до 32 767
`long` на Arduino Uno	Целые от −2 147 483 648 до 2 147 483 647
`/` с целыми числами	Дробная часть отбрасывается: `7 / 2` даёт `3`

Требование 1. Функция `level(value, white, black)` переводит показание в шкалу 0…100:

- чёрное — 0, белое — 100, между ними — по прямой;
- работает и когда у датчика чёрное — большое число, а белое — маленькое;
- никогда не выходит за 0…100;
- не переполняется на Arduino Uno.

Приёмка. Напиши `level` в тестовом скетче, загрузи его и открой монитор порта. Тест проверяет восемь случаев, в том числе датчик «наоборот» и показания за пределами шкалы. Когда все строки `PASS`, перенеси функцию в свою программу и проверь шкалу обоих датчиков на дорожке.

Тест функции level L32_Level_Test.ino

```
// Проверка функции level. Напиши функцию здесь, загрузи и открой мон
// Когда все строки PASS, перенеси level в свою программу.

// Требование 1: шкала «чёрное — 0, белое — 100».
int level(int value, int white, int black) {
  return 0;
}

int failed = 0;

void expect(int value, int white, int black, int expected) {
  int actual = level(value, white, black);
  Serial.print(actual == expected ? "PASS " : "FAIL ");
  Serial.print("level(");
  Serial.print(value);
  Serial.print(", ");
  Serial.print(white);
  Serial.print(", ");
  Serial.print(black);
  Serial.print(") = ");
  Serial.print(actual);
  Serial.print(", ожидаем ");
  Serial.println(expected);
  if (actual != expected) {
    failed++;
  }
}

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

  // Датчик, у которого чёрное — большое число, а белое — маленькое.
  expect(780, 120, 780, 0);
  expect(120, 120, 780, 100);
  expect(450, 120, 780, 50);
  // За пределами шкалы.
  expect(900, 120, 780, 0);
  expect(50, 120, 780, 100);
  // Датчик, у которого наоборот: белое — большое число, чёрное — мал
```

```
  expect(120, 900, 120, 0);
  expect(510, 900, 120, 50);
  expect(900, 900, 120, 100);

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

void loop() {
}
```

Подсказка: Идея (1 из 3)

Шкала — это доля пути от чёрного до белого. Весь путь — разница белого и чёрного.

Подсказка: План (2 из 3)

1. Насколько показание ушло от чёрного?
2. Раздели на длину всего пути и умножь на 100.
3. Обрежь результат границами шкалы.

Подсказка: Фрагмент кода (3 из 3)

`long result = (long)(value - black) * 100 / (white - black);` — почему здесь `long` и почему умножение стоит раньше деления?

ДОЛЖНО ПОЛУЧИТЬСЯ

Тест показывает «Все проверки пройдены». На дорожке шкала показывает около 0 на чёрном и около 100 на белом.

3 Считаем ошибку

15 мин

Требование 2. Программа считает ошибку — число со знаком: насколько и в какую сторону линия ушла от середины. Знак выбираешь ты, но потом поправка должна поворачивать робота **к линии**.

1. Добавь расчёт ошибки и временный вывод `leftLevel`, `rightLevel`, ошибки в монитор порта.
2. Заполни таблицу и запиши в журнал своё правило знака одной фразой.

Где линия	leftLevel	rightLevel	Ошибка
Посередине			
Под левым			
Под правым			

3. Разберись с краем: линия ушла **за пределы обоих датчиков**. Что покажет ошибка и что тогда сделает робот? Запиши, чем это опасно на крутой дуге.

Подсказка: Идея (1 из 2)

Если линия ровно посередине, оба датчика видят одно и то же — ошибка должна быть ноль.

Подсказка: Проверка знака (2 из 2)

Выбери: `left - right` или `right - left`. Запиши, какое колесо должно ускориться, когда линия слева, — это понадобится в требовании 3.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена, правило знака записано. Ты знаешь, что видит регулятор, когда линия ушла за оба датчика.

2

ЗАНЯТИЕ 2 · 45 МИН

П-регулятор на дорожке

4 От ошибки к скоростям

10 мин

Требование 3. Робот едет вперёд со скоростью `baseSpeed` и поворачивает к линии тем сильнее, чем больше ошибка:

- поправка пропорциональна ошибке, коэффициент `kp`;
- `kp` и `baseSpeed` — именованные переменные в начале скетча, их подбирают на дорожке;
- скорости колёс всегда в пределах 0…255;
- без измерений белого и чёрного колёса стоят (защита из урока 3.1 остаётся).

Приёмка на бумаге. Посчитай скорости колёс для своей программы:

`baseSpeed`	`kp`	Ошибка	Левое	Правое
100	0.5	−40		
200	0.8	60		
200	1.0	100		

В последней строке одно колесо упирается в границу. Насколько слабее робот повернёт, чем «хотел»? Запиши в журнал, что можно с этим сделать.

Подсказка: Идея (1 из 3)

Одно колесо получает базовую скорость плюс поправку, другое — минус поправку.

Подсказка: План (2 из 3)

1. Ошибка из требования 2.
2. Поправка = коэффициент × ошибка.
3. Левое и правое колесо: база ± поправка, каждое — в границах 0…255.

4. Отправить скорости на моторы.

Подсказка: Фрагмент кода (3 из 3)

`int leftSpeed = constrain(baseSpeed + correction, 0, 255);` — а что для правого? Проверь знак по своему правилу из требования 2.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование 3, таблица посчитана, ограничение записано в журнал.

5 Проверь на подставке

7 мин

Приёмка требования 3 на подставке.

1. Временно выведи в монитор порта ошибку, поправку и обе скорости колёс.
2. Подводи линию под левый датчик, правый и посередине. Совпадает ли направление поправки с твоим правилом знака из требования 2?
3. Найди положение линии, при котором одно колесо упирается в 0 или 255. Сравни скорости в мониторе с тем, что ты посчитал в таблице требования 3.
4. Отключи измерения (нули) — колёса стоят? Верни измерения.

ДОЛЖНО ПОЛУЧИТЬСЯ

Направление поправки, упор в границу и защита проверены по монитору порта. Расхождения с таблицей объяснены.

6 Подбери коэффициент

18 мин

Подбери `kp` и скорость **методом**, а не наугад, и докажи выбор записями.

1. При `baseSpeed = 100` начни с `kp = 0.1` и удваивай: 0.1, 0.2, 0.4, 0.8… Найди наименьший `kp`, при котором робот начинает вилять **на прямой**.
2. Рабочий `kp` возьми примерно вдвое меньше. Проверь его на дуге своего уровня.
3. Поднимай `baseSpeed` шагами по 20. Для каждой скорости — три заезда. Запомни последнюю скорость, на которой робот прошёл все три.
4. Запиши в журнал таблицу и вывод: почему именно эта пара, что ограничивает скорость.
5. Вернись к правилу, которое ты придумал в первом шаге. Чем оно похоже на П-регулятор и чем отличается? Что бы ты в нём изменил теперь?

`kp`	`baseSpeed`	Заезды (из 3)	Что видно

Подсказка: Робот виляет при любом kp

Проверь, не упирается ли колесо в 255 или 0 на дуге — вспомни последнюю строку таблицы из требования 3.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица с методом подбора, пара `kp – baseSpeed`, проверенная тремя заездами, вывод с обоснованием и сравнение своего правила с П-регулятором.

7 Задания по уровням

10 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · КРУТАЯ S-СВЯЗКА**

Пройти две дуги радиусом 125 мм подряд в разные стороны, как на продвинутом поле. Начни с пары `kp — скорость`, выбранной в шаге подбора. Если она не удерживает S-связку, измени один параметр и дополни существующую таблицу результатом; заново переписывать удачные и неудачные попытки не нужно.

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

★★ Продвинутый уровень

1. На чёрном и у края линии шкала верна, но для показания 50 при white = 120 и black = 780 функция возвращает неверное число. В чём ошибка?

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

- ○ (50 − 780) × 100 = −73 000 не помещается в int Arduino Uno
- ○ constrain не работает с отрицательными числами
- ○ Нельзя делить на отрицательное число

2. Что вернёт эта функция для показания 450 при white = 120 и black = 780?

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

- ○ 50
- ○ 0
- ○ 100

3. Ошибку посчитали как rightLevel - leftLevel, а поправку по-прежнему прибавляют к левому колесу. Линия ушла под левый датчик. Что сделает робот?

- ○ Повернёт от линии и потеряет её
- ○ Повернёт к линии, как обычно
- ○ Остановится

4. На крутой дуге линия ушла за пределы обоих датчиков. Что сделает П-регулятор из урока?

- ○ Оба датчика видят белое, ошибка около нуля — робот поедет прямо и потеряет линию
- ○ Ошибка станет максимальной — робот резко развернётся к линии
- ○ Сработает защита — робот остановится

5. baseSpeed = 200, kp = 1.0, ошибка = 100, скорости ограничены 0…255. Что будет с поворотом?

- ○ Поворот слабее расчётного: левое колесо упёрлось в 255, разница колёс 155 вместо 200
- ○ Поворот такой же, как рассчитан: constrain на него не влияет
- ○ Робот повернёт в другую сторону

Отвечай на вопросы своего уровня. Нужно верных ответов: Начальный — 4 из 5; Средний — 4 из 5; Продвинутый — 4 из 5. Ответы проверит преподаватель или программа курса.

Итог

Робот поворачивает 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.

Сколько показаний робот успевает получить, пока едет по поперечной линии?

1. Добавь в свою программу из урока 3.2 счётчик повторов `loop()` и раз в секунду печатай, сколько повторов прошло. Запиши число повторов в секунду на рабочей скорости.
2. Возьми свою скорость в мм/с из урока 2.1 или измерь её заново.
3. Посчитай: сколько повторов придётся на поперечную линию шириной 19 мм?
4. Провези робота руками через перекрёсток и сравни показания со своей таблицей положений.

Подсказка: Как посчитать

Время на линии — ширина линии, делённая на скорость. Повторов — это время, умноженное на число повторов в секунду.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — повторы в секунду, скорость, расчёт числа повторов на линии и показания на перекрёстке.

2 Условие перекрёстка

10 мин

Требование 1. Функция `onCrossing()` отвечает `true` на поперечной линии и `false` во всех остальных положениях робота.

Проверь условие на бумаге, прежде чем писать код. Для каждого положения запиши показания левого и правого датчика по шкале 0–100 и ответ функции:

Положение	leftLevel	rightLevel	`onCrossing()`
Линия посередине			
Линия под одним датчиком на крутой дуге 125 мм			
Линия ушла за оба датчика			
Поперечная линия			

1. Выбери порог по своим измерениям и обоснуй его: какой запас до самого тёмного показания вне перекрёстка?
2. Найди положение, в котором твоё условие может ошибиться. Что тогда сделает робот?

Подсказка: Где искать ошибку

Провези робота руками по дуге 125 мм из урока 3.2: насколько тёмным становится «край» линии под датчиком?

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица заполнена по измерениям, порог обоснован запасом, опасное положение и его последствие записаны.

3 Стоп на перекрёстке

20 мин

Требование 2. Продолжи свою программу `L32_P_Line` из урока 3.2 и сохрани как `L41_Crossing_Stop`:

- шаг П-регулятора вынесен в функцию `followLineStep()`, чтение датчиков — в `readSensors()`; П-регулятор при этом работает как раньше;
- функция `followLineToCrossing()` ведёт робота по линии и заканчивается, как только `onCrossing()` вернул `true`;
- в `setup()` робот доезжает до первого перекрёстка и останавливается;
- защита без измерений и пауза на старте остаются.

Приёмка. Робот останавливается на перекрёстке пять раз из пяти и ни разу на дуге своего уровня.

Подсказка: Идея (1 из 3)

Функция «едет, пока…» — это бесконечный цикл, из которого есть выход по условию.

Подсказка: План (2 из 3)

1. Повторять всегда: прочитать датчики.
2. Перекрёсток — выйти из функции.
3. Иначе — шаг П-регулятора.

Подсказка: Фрагмент кода (3 из 3)

```
while (true) { readSensors(); if (onCrossing()) { return; } … }
```

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требования 1–2, пять остановок из пяти записаны в журнал.

2

ЗАНЯТИЕ 2 · 45 МИН

Считаем перекрёстки

4 Найди ошибку счёта

8 мин

НАЙДИ ОШИБКУ · СЧЁТ ПО МОМЕНТУ ПОЯВЛЕНИЯ, А ВСЁ РАВНО 40

Соседи решили считать только момент, когда перекрёсток появился:

```
bool was = false;

void loop() {
  readSensors();
  bool now = onCrossing();
  if (now && !was) {
    count = count + 1;
  }
  followLineStep();
}
```

Робот проехал один перекрёсток, а `count` всё равно стал 40. Где ошибка?

1. Найди ошибку и запиши исправленный код в журнал.
2. Сравни 40 со своим расчётом числа повторов на линии из прошлого шага. Похоже?
3. Чем счёт по моменту появления хуже или лучше проезда перекрёстка, который будет в следующем шаге?

Подсказка: Что хранит was

Когда в этой программе меняется значение `was`?

ДОЛЖНО ПОЛУЧИТЬСЯ

Ошибка найдена, исправление записано кодом, число 40 объяснено расчётом, сравнение двух способов записано.

5 Проехать перекрёсток

10 мин

Требование 3. Функция `passCrossing()` проезжает перекрёсток прямо и заканчивается, когда датчики уже не на поперечной линии и не на её краю.

1. Напиши функцию в своей программе.
2. Исследуй паузу после схода с чёрного: 0, 50, 100, 200 мс. Для каждой запусти робота через два перекрёстка подряд (в `setup()` — `followLineToCrossing()`, `passCrossing()`, `followLineToCrossing()`, стоп) и отметь, на каком перекрёстке он остановился.
3. Запиши, какая пауза нужна и от чего она зависит. Что будет, если поднять `baseSpeed`?

Пауза, мс	Где остановился	Что видно
0		
50		
100		
200		

Подсказка: Идея (1 из 2)

Ехать прямо и повторять чтение датчиков, пока условие перекрёстка верно.

Подсказка: Фрагмент кода (2 из 2)

`do { readSensors(); } while (onCrossing());` — почему здесь `do`, а не `while`?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

`passCrossing()` работает, таблица паузы заполнена, вывод о связи паузы и скорости записан.

6 Стоп на нужном перекрёстке

17 мин

Требование 4. Функция `goToCrossing(n)` ведёт робота по линии, проезжает прямо перекрёстки до `n`-го и останавливается на `n`-м. Сохрани программу как `L41_Count`.

Приёмка. Дорожка с тремя перекрёстками, `n` = 1, 2, 3, по три запуска на каждое.

`n`	Запуск 1	Запуск 2	Запуск 3
1			
2			
3			

ПРЕДСКАЖИ

Что должна сделать функция при `n = 0`? Реши сам, запиши и проверь, что программа делает именно это.

Подсказка: План (1 из 2)

`n - 1` раз: найти перекрёсток и проехать его. Потом найти последний и остановиться.

Подсказка: Фрагмент кода (2 из 2)

`for (int i = 1; i < n; i++) { … }` — сколько раз выполнится цикл при `n = 1`?

ДОЛЖНО ПОЛУЧИТЬСЯ

Девять запусков без ошибки в таблице, поведение при `n = 0` выбрано и проверено.

7 Задания по уровням

10 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · БЕЗ ОСТАНОВОК**

Сделай так, чтобы робот проезжал все перекрёстки без остановки и не терял линию после них. Подсказка: в `loop()` — если перекрёсток, `passCrossing()`, иначе шаг П-регулятора.

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

★★ Продвинутый уровень

1. Что должна делать функция passCrossing()?

- ○ Проехать перекрёсток прямо и съехать с чёрной области без остановки
- ○ Остановить робота
- ○ Повернуть налево

2. Что помогает не потерять линию сразу после перекрёстка? Выбери все верные ответы.

- [ ] Проезжать перекрёсток прямо с одинаковой скоростью колёс
- [ ] Сразу после перекрёстка вернуться к шагу П-регулятора
- [ ] Выключить датчики линии на всю поездку

3. Без паузы после схода с поперечной линии робот иногда останавливается сразу за перекрёстком. Почему?

- ( ) passCrossing() выключает моторы
- ( ) Счётчик перекрёстков сбрасывается
- ( ) Датчики ещё видят край поперечной линии, и он считается новым перекрёстком

4. Вызвали goToCrossing(2). На каком перекрёстке остановится робот?

```
void goToCrossing(int n) {
  for (int i = 0; i < n; i++) {
    followLineToCrossing();
    passCrossing();
  }
  followLineToCrossing();
  stopMotors();
}
```

- ○ На третьем
- ○ На втором
- ○ На первом

5. На крутой дуге робот иногда останавливается, будто на перекрёстке. Что проверить первым?

- ○ Заряд батареи
- ○ Показания обоих датчиков на дуге: не опускаются ли оба ниже порога
- ○ Коэффициент kp

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.

Проверь идею «крутиться до новой линии» измерением, прежде чем её программировать.

1. Положи под робота лист с транспортиром или линиями через 10°. Поставь робота на центр перекрёстка.
2. Медленно поворачивай руками налево. Запиши угол, на котором левый датчик **впервые** видит новую линию, и угол, на котором робот смотрит точно вдоль неё.
3. Повтори направо.
4. Совпадают ли углы? Что тогда сделает робот, который останавливается в момент, когда датчик увидел линию? Как это исправить?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — два угла для каждого поворота и вывод о том, где остановится робот.

2 Выезд на центр перекрёстка

12 мин

Датчики стоят **впереди** колёс, поэтому перед поворотом робот выезжает на центр перекрёстка. Рассчитай время этого выезда, а не подбирай наугад.

1. Измерь расстояние от датчиков до оси колёс в миллиметрах.
2. Возьми из журнала урока 2.1 свою скорость в миллиметрах в секунду при близкой `baseSpeed`. Нет таблицы — измерь: проезд 1 секунду, путь линейкой, три раза.
3. Посчитай `centerMs` и запиши расчёт. На следующем шаге сравни его с подобранным на дорожке.
4. Запиши, почему выезд по времени — слабое место и от чего его точность зависит.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — расстояние, скорость, расчёт `centerMs` и две причины, по которым выезд по времени может ошибаться.

3 Поворот на перекрёстке

22 мин

Требование 1. Продолжи свою программу `L41_Count` и сохрани как `L42_Turn_At_Crossing`. Функция `turnUntilLine(left)` поворачивает робота на месте до новой линии:

- сначала робот крутится, не глядя на датчики, пока датчик не уйдёт со старой линии;
- потом — пока датчик со стороны поворота не увидит линию;
- если линия не нашлась за 3 секунды, робот останавливается: на поле без защиты он будет крутиться бесконечно.

Требование 2. `turnLeftAtCrossing()` и `turnRightAtCrossing()`: выезд на центр по твоему `centerMs`, затем поворот до линии.

Приёмка. По пять поворотов налево и направо: после каждого робот уезжает по новой линии. Отдельно проверь защиту: подними робота над столом во время поворота.

Подсказка: Идея (1 из 3)

Поиск линии — цикл «читать датчики, пока линии нет». Для защиты запомни время начала поиска.

Подсказка: План (2 из 3)

1. Включить поворот, подождать `leaveMs`.
2. Запомнить `millis()`.
3. Повторять чтение, пока линии нет и прошло меньше 3 секунд.
4. Остановить моторы.

Подсказка: Фрагмент кода (3 из 3)

`while ((left ? leftLevel : rightLevel) > 30 && millis() - start < 3000)` — прочитай условие словами.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требования 1–2, десять поворотов и проверка защиты записаны. Подобранный `centerMs` сравнён с расчётом.

ЗАНЯТИЕ 2 · 45 МИН

2 Маршрут из решений

4 Разворот на линии

10 мин

Требование 3. `turnAroundOnLine()` разворачивает робота на линии и останавливается, когда линия снова под датчиком. Защита 3 секунды из требования 1 работает и здесь.

Приёмка. Пять разворотов на конце ответвления: после каждого робот возвращается к перекрёстку. Для каждого измерь, насколько нос смотрит мимо линии сразу после разворота.

Разворот	Вернулся?	Нос мимо линии, мм
1		

Если нос каждый раз смотрит в одну сторону, предложи и проверь одно исправление.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Пять разворотов в таблице, исправление систематической ошибки проверено.

5 Маршрут по порядку

22 мин

Требование 4. Маршрут планируется до программы и проверяется по частям.

1. Выбери на тренировочном поле путь из четырёх решений или больше, в котором есть налево, направо и прямо. Если в классе разложено поле итоговой задачи, возьми путь от старта до одной зоны доставки. Запиши его таблицей:

№	Перекрёсток	Решение	Куда смотрит робот после
1			

2. Запиши маршрут в программу `L42_Route` из своих функций.
3. Если маршрут сломался, не переписывай всё: запусти только первые решения, потом половину, и найди первое решение, на котором робот ошибается. Запиши, как искал.

ПРЕДСКАЖИ

На каком решении маршрут скорее всего сломается и почему? Запиши до первого запуска.

Подсказка: Как проверять по частям

Временно поставь `stopMotors();` и `return;` после нужного решения в `setup()`.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица маршрута, программа по ней, прогноз и запись поиска ошибки. Робот проходит маршрут на тренировочной дорожке.

6 Задания по уровням

13 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · МАРШРУТ МАССИВОМ**

Запиши решения массивом: `char route[] = {'L', 'S', 'R', 'L'};`. Напиши функцию `decide(d)`, которая выполняет одно решение, и цикл, который проходит весь массив. Составь маршрут по схеме поля так, чтобы для нового маршрута менялась только строка с массивом.

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

★★ Продвинутый уровень

1. Какое решение выполнится третьим?

```
1 char route[] = {'L', 'S', 'R', 'L'};
```

- ○ Направо
- ○ Прямо
- ○ Налево

2. Что нужно изменить в программе с массивом, чтобы пройти новый маршрут?

- ○ Только строку с массивом route
- ○ Функцию decide()
- ○ Весь цикл обхода маршрута

3. От датчиков до оси колёс 60 мм, скорость робота 200 мм/с. Каким должен быть centerMs по расчёту?

- ○ 120 мс
- ○ 3000 мс
- ○ 300 мс

4. Робота подняли над столом во время поворота. Что произойдёт?

```
unsigned long start = millis();
do {
  readSensors();
} while ((left ? leftLevel : rightLevel) > 30 && millis() - start < 3
stopMotors();
```

- ○ Через 3 секунды моторы выключатся, хотя линия не найдена
- ○ Робот будет крутиться бесконечно
- ○ Моторы выключатся сразу

5. Маршрут из восьми решений ломается. Как быстрее найти сломанное решение?

- ○ Переписать маршрут заново
- ○ Запустить первую половину маршрута, потом половину оставшейся части
- ○ Каждый раз запускать весь маршрут и смотреть

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. По времени полёта считают расстояние.

1. Звук пролетает 1 см примерно за 29 мкс. Выведи формулу: расстояние в сантиметрах по времени эха в микросекундах.
2. Скорость звука зависит от температуры. При 0 °C звук пролетает 1 см примерно за 30 мкс. Груз стоит в 50 см. Сколько покажет датчик зимой в

холодном коридоре, если программа считает по 29 мкс? Важна ли эта ошибка для остановки в 5 см?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Формула и расчёт ошибки записаны. Ты можешь объяснить, когда ошибкой можно пренебречь.

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.

Требование 1. Функция `readDistanceCm()` возвращает расстояние в сантиметрах или −1, если эха нет. Как работает датчик:

- на TRIG подаётся импульс HIGH длиной 10 мкс (перед ним — короткий LOW);
- датчик держит ECHO в HIGH столько микросекунд, сколько звук летел туда и обратно;
- звук пролетает 1 см примерно за 29 мкс;
- ждать эхо дольше 25 000 мкс не нужно: команда `pulseIn(pin, HIGH, timeoutUs)` вернёт 0.

Без номеров контактов программа не должна работать с датчиком: пусть она остановится. Напиши скетч `L51_Distance`, который печатает расстояние или «нет эха».

Приёмка. Таблица по линейке и ответы в журнале.

По линейке	Показание	Ошибка, см
5 см		
10 см		
20 см		
30 см		

2. Какое самое большое расстояние может измерить функция при ожидании 25 000 мкс?
3. Растёт ли ошибка с расстоянием? Чем ты это объяснишь?

Подсказка: Идея (1 из 3)

Щёлкнуть, измерить длину импульса эха, перевести микросекунды в сантиметры.

Подсказка: План (2 из 3)

1. TRIG: LOW 2 мкс, HIGH 10 мкс, LOW.
2. `pulseIn` на ECHO с ожиданием 25 000 мкс.
3. Ноль — нет эха. Иначе — время делить на 58.

Подсказка: Фрагмент кода (3 из 3)

`unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000);` — почему `unsigned long`, а не `int`?

ДОЛЖНО ПОЛУЧИТЬСЯ

Функция выполняет требование 1, таблица с ошибками заполнена, ответы записаны.

3 «Нет эха» — не «рядом»

15 мин

Найди границы, за которыми датчику нельзя верить.

1. Поставь груз в 20 см и поворачивай его боком к датчику. С какого угла пропадает эхо?
2. Придвигай груз к датчику. С какого расстояния показания перестают совпадать с линейкой?
3. Направь датчик на мягкую ткань. Что печатает монитор?

НАЙДИ ОШИБКУ · РОБОТ ЗАКРЫЛ ЗАХВАТ В ПУСТОТЕ

Соседи написали остановку так: `if (readDistanceCm() <= 5) { stopMotors(); }`. Робот иногда останавливается посреди пустой дороги. Почему?

4. Запиши, какие из найденных границ опасны для доставки и как программа должна себя вести в каждом случае.

ДОЛЖНО ПОЛУЧИТЬСЯ

В журнале — угол потери эха, ближняя граница, поведение на ткани и правила для программы.

ЗАНЯТИЕ 2 · 45 МИН

2 Стоп перед грузом

4 Едем к грузу и останавливаемся

17 мин

Требование 2. Продолжи свою программу `L42_Turn_At_Crossing` и сохрани как `L51_Stop_At_Can`. Функция `followLineToCan(stopCm)` ведёт робота по линии и останавливает его, когда груз не дальше `stopCm`:

- «нет эха» никогда не считается «груз рядом»;
- после остановки моторы выключены;
- функции линии и перекрёстков из прошлых уроков не меняются.

Приёмка. Груз в конце линии — робот останавливается, не касаясь груза. Груза нет — робот доезжает до конца линии, не останавливаясь посреди дороги (будь готов выключить питание).

Подсказка: Идея (1 из 2)

Это `followLineToCrossing()` из урока 4.1, только выход — по расстоянию.

Подсказка: Фрагмент кода (2 из 2)

`if (distance > 0 && distance <= stopCm) { return; }` — почему здесь `return`, а не `break`?

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование 2, обе проверки приёмки записаны.

5 Точная остановка

15 мин

Добейся остановки на нужном расстоянии **±1 см три раза из трёх** на своей рабочей скорости — той, на которой робот проходит линию.

1. Измерь выбег: при `baseSpeed` 80, 120 и своей рабочей поставь `STOP_CM = 15` и измерь, на сколько сантиметров робот прокатился после команды стоп.
2. Выбери способ точной остановки и обоснуй его таблицей: поправить `STOP_CM` на выбег или ехать медленнее в последние сантиметры перед грузом.
3. Проверь три раза.

`baseSpeed`	`STOP_CM`	Где остановился, см	Выбег, см
80	15		
120	15		
	15		

Подсказка: Медленно в конце

Два порога: дальше первого — рабочая скорость, между первым и вторым — медленная, у второго — стоп.

ДОЛЖНО ПОЛУЧИТЬСЯ

Таблица выбега, выбранный способ с обоснованием и три остановки ±1 см записаны.

6 Задания по уровням

13 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · МЕДИАНА ИЗ ТРЁХ**

Делай три измерения и бери среднее по величине — медиану. Одно случайное неверное число тогда не испортит остановку. Сравни в мониторе порта сырые измерения и медиану.

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

★★ Продвинутый уровень

1. pulseIn ждёт эхо не дольше 25 000 мкс. Какое наибольшее расстояние может вернуть функция?

- ○ Около 862 см
- ○ 250 см
- ○ Около 431 см

2. Почему для остановки берут медиану трёх измерений, а не среднее арифметическое?

- ○ Одно случайное неверное число не сдвигает результат
- ○ Медиана считается быстрее всего
- ○ Медиана всегда равна первому измерению

3. Где ошибка, опасная для робота?

```
int readDistanceCm() {
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000);
  return echoUs / 58;
}
```

- ○ Если эха нет, функция вернёт 0 — это похоже на «груз вплотную»
- ○ Делить нужно на 29
- ○ pulseIn нельзя вызывать с ожиданием

4. **Выбег при рабочей скорости — 3 см. Роботу нужно стоять в 5 см от груза. Какой STOP_CM поставить?**

- ○ 5
- ○ 8
- ○ 2

5. **Сравнение с линейкой: 5 → 6, 10 → 11, 20 → 21, 30 → 31 см. Что это значит?**

- ○ Ошибка растёт с расстоянием
- ○ Датчик неисправен
- ○ Постоянный сдвиг на 1 см: его учитывают одной поправкой

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.

Сервопривод поворачивается на заданный угол и держит его: внутри мотор, шестерёнки и датчик положения.

1. Посмотри контакт сервопривода на карточке у робота и впиши его в скетч пробы угла.
2. Сделай опыт: `gripper.write(40)`, пауза, `gripper.write(90)`, пауза. Пробуй паузы 100, 300 и 600 мс. При какой паузе захват успевает дойти до угла?
3. Запиши, сколько примерно времени нужно сервоприводу на 50° и почему программа должна ждать после команды.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

В журнале — опыт с паузами и наименьшая пауза, при которой захват доходит до угла.

2 Найди углы «открыто» и «закрыто»

22 мин

Посмотри контакт сервопривода на карточке у робота.

Требование 1. Скетч `L52_Servo_Test` для подбора углов: угол вводится числом в монитор порта, и захват поворачивается сразу, без новой загрузки программы.

- Принимаются только числа от 0 до 180; на другое число программа отвечает, что так нельзя, и захват не двигается.
- После каждого поворота программа печатает, какой угол сейчас.
- Без номера контакта программа не трогает сервопривод.

Как это устроено: Справка по командам

Команда	Что делает
`#include <Servo.h>`, `Servo gripper;`	Подключить библиотеку и создать сервопривод
`gripper.attach(pin)`	Связать сервопривод с контактом
`gripper.write(angle)`	Повернуть на угол 0...180
`Serial.available()`	Сколько символов пришло из монитора порта
`Serial.parseInt()`	Прочитать из пришедших символов целое число

Требование 2. Углы «открыто» и «закрыто» найдены с запасом и проверены:

- «открыто» — груз входит в захват, даже если робот остановился на 1 см левее или правее середины;
- «закрыто» — найди угол, где захват упирается в груз, и отступи от него так, чтобы сервопривод не гудел; запиши, на сколько градусов отступил;
- груз не выпадает, когда робота с грузом наклоняют на 30° вперёд и назад.

Требование 3. Функция `moveGripper(target)` поворачивает захват к углу плавно — шагами по 2° — а не одним рывком. Сравни, как ведёт себя груз при резком и плавном закрытии.

Подсказка: Идея (1 из 2)

Помни текущий угол в переменной. Пока он не равен цели, сдвигай его на шаг в сторону цели и делай короткую паузу.

Подсказка: Фрагмент кода (2 из 2)

`currentAngle += (target > currentAngle) ? 2 : -2;` — что будет, если цель отличается на нечётное число градусов?

Подсказка: В мониторе порта ничего не происходит

Проверь скорость 9600 и что монитор отправляет строку по Enter.

ДОЛЖНО ПОЛУЧИТЬСЯ

Скетч подбора углов выполняет требование 1. В журнале — углы, запас «закрыто» в градусах, проверка наклоном и сравнение резкого и плавного закрытия. `moveGripper` работает.

3 Порядок действий важен

15 мин

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?

1. Запиши правила порядка для «взять» и «поставить».
2. Найди опытом наименьшую паузу после открытия, при которой груз остаётся на месте: 100, 200, 400, 600 мс, по три раза каждую.
3. Какую паузу ты возьмёшь в программу и почему не самую короткую из удачных?

ДОЛЖНО ПОЛУЧИТЬСЯ

Правила записаны, таблица опыта и выбранная пауза с обоснованием.

2

ЗАНЯТИЕ 2 · 45 МИН

Перевозка груза

4 Взять и поставить

20 мин

Требование 4. Продолжи свою программу `L51_Stop_At_Can` и сохрани как `L52_Pick_Place`. Робот выполняет доставку по линии:

- `pickUp()` и `putDown()` соблюдают порядок: остановиться — закрыть; открыть — подождать — отъехать;
- захват двигается твоей функцией `moveGripper`;
- без контакта сервопривода программа не начинает движение;
- в `setup()`: подъехать к грузу, взять, развернуться на линии, проехать с грузом, поставить.

Приёмка. Груз в конце линии, робот — на другом конце: доставка без помощи рук.

Подсказка: Идея (1 из 2)

Каждое действие доставки — уже готовая функция из прошлых уроков или этого урока. Осталось соединить их по порядку.

Подсказка: План (2 из 2)

1. `followLineToCan(STOP_CM)`.
2. `pickUp()`.
3. `turnAroundOnLine()`.
4. Проехать по линии `CARRY_MS`.
5. `putDown()`.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требования 1–4, доставка проходит без помощи рук.

5 Надёжность

12 мин

Измерь надёжность доставки и улучши её **по данным**.

1. Сделай 10 доставок подряд и отметь для каждой, на каком этапе она сорвалась: подъезд, захват, разворот, перевозка, постановка.
2. Посчитай, сколько доставок из 10 прошли. Какой этап даёт больше всего срывов?
3. Исправь только этот этап и снова сделай 10 доставок. Стало лучше? На сколько?

Серия	Успешно из 10	Подъезд	Захват	Разворот	Перевозка	Постанови
До						
После						

Подсказка: Если срывов нет

Усложни условия: поставь груз на 1 см в сторону от линии или подними `baseSpeed`. Где сорвётся первым?

ДОЛЖНО ПОЛУЧИТЬСЯ

Две серии по 10 доставок, этап с больше всего срывов, что изменено и насколько выросла надёжность.

6 Задания по уровням

13 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · ФУНКЦИЯ DELIVER**

Напиши функцию `deliver(маршрут до груза, маршрут до места, время до места)`, которая выполняет всю доставку. Маршруты передавай строками решений, например `"LS"`. Тогда в итоговой задаче доставка будет одной строкой программы.

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

★★ Продвинутый уровень

1. Что в этом вызове означает строка "LS"?

```
1  deliver("LS", "R", 1500);
```

- ○ Маршрут до груза: на первом перекрёстке налево, на втором прямо
- ○ Маршрут до места доставки
- ○ Скорость моторов

2. Зачем оформлять доставку функцией deliver()?

- ○ В итоговой задаче доставка будет одной строкой, а исправлять её нужно в одном месте
- ○ Без функции захват не закроется
- ○ Функция ускоряет моторы

3. currentAngle = 90, target = 41. Что произойдёт?

```
1 void moveGripper(int target) {
2   while (currentAngle != target) {
3     currentAngle += (target > currentAngle) ? 2 : -2;
4     gripper.write(currentAngle);
5     delay(15);
6   }
7 }
```

- ○ Цикл не закончится: угол перескакивает 41 (42 → 40 → 42 …)
- ○ Захват остановится на 42
- ○ Захват дойдёт ровно до 41

4. **Серия из 10 доставок: срывов на подъезде — 0, на захвате — 1, на развороте — 4, при перевозке — 1, на постановке — 0. Что улучшать первым?**

- ○ Всё сразу
- ○ Разворот: там больше всего срывов
- ○ Захват

5. **После исправления разворота успешных доставок стало 7 из 10 вместо 4. Как убедиться, что помогло именно исправление?**

- ○ Сделать ещё одну доставку
- ○ Никак: 7 больше 4, этого достаточно
- ○ Сравнить срывы по этапам: на развороте их должно стать меньше, на остальных — столько же

Отвечай на вопросы своего уровня. Нужно верных ответов: Начальный — 4 из 5; Средний — 4 из 5; Продвинутый — 4 из 5. Ответы проверит преподаватель или программа курса.

Итог

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

Как подтвердить результат

Результат опыта связывает условия, версию программы, наблюдение и вывод. Сравнение попыток с одним изменением помогает проверить причину, включая

неудачные попытки. Отметка относится к уже выполненной практике; повтор ради отметки не нужен. Расхождение с прогнозом задаёт следующую проверяемую гипотезу. Модель, измерения настоящего робота и объяснение — разные виды доказательств.

Словарь урока

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

Тема 6 · Урок 6.1

Робот-курьер: план и программа

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

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

РЕЗУЛЬТАТ УРОКА

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

Ты научишься

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

Нужно знать

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

Понадобится

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

Твоя задача

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

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

РАЗМИНКА · ВСПОМНИ

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

1

ЗАНЯТИЕ 1 · 45 МИН

Уровень и план

1 Выбери уровень

6 мин

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

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

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

Задача есть в трёх уровнях. Правила и критерии оценки у них общие, а поля разные.

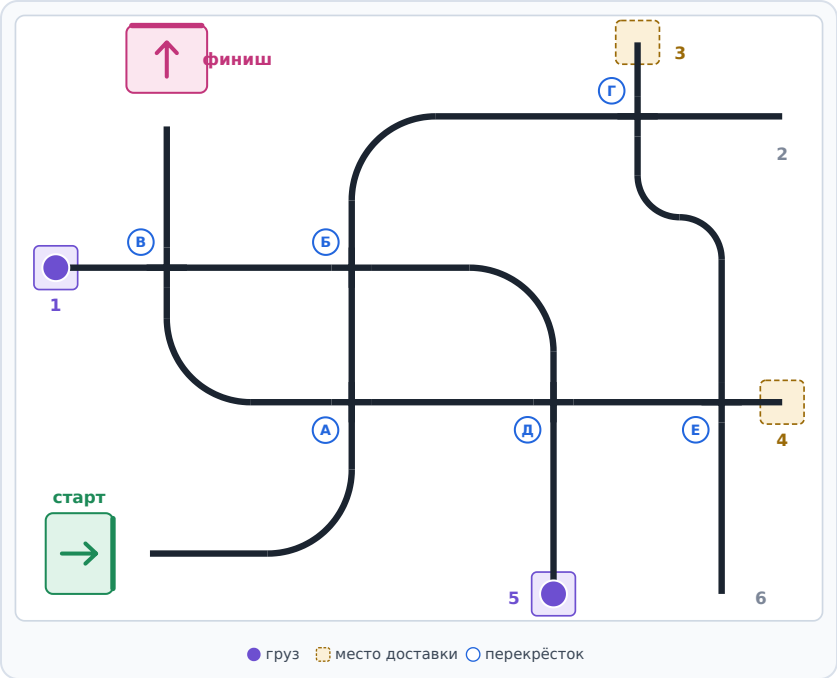
Уровень итоговой задачи	Какие задания уроков нужны
Начальный S-образная линия, два перекрёстка, порядок действий известен заранее.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 4.2 5.1 5.2
Средний Сеть дорог, четыре перекрёстка, проезд прямо и повторные встречи с перекрёстками.	● Основное 1.1 2.1 2.2 3.1 3.2 4.1 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. Выпиши из журнала задания ★★, которые у тебя получились, и числа надёжности из уроков: ошибку поворота, сколько раз робот прошёл крутую дугу, точность остановки у груза, надёжность доставки из урока 5.2.
2. Для продвинутого поля назови два самых слабых умения по этим числам.
3. Выбери поле сам и обоснуй выбор числами. Запиши, что ты сделаешь со слабыми умениями до зачёта.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
Поле выбрано и обосновано числами из журнала, два слабых умения и план по ним записаны.

2 Разберись в задании своего уровня

8 мин



Продвинутое поле. Перекрёстки A–E, конечные позиции 1–6. На схеме — расстановка для карточки 1.

Банки стоят в позициях 1 и 5. Перед планированием вытяни карточку из стопки: на ней написано, в каком порядке и куда их перевезти.

Карточка	Первая доставка	Вторая доставка
1	1 → 4	5 → 3
2	1 → 3	5 → 4
3	5 → 3	1 → 4
4	5 → 4	1 → 3

Готового маршрута нет: путь по перекрёсткам ты выбираешь сам, до запуска. Правильных путей несколько. Справа на поле — крутая S-образная связка.

Проверь, что твой робот проходит S-образную связку с грузом.

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.

Проверь умения **по своим числам**: для каждого запиши, какой результат ты считаешь надёжным, и проверь пять раз.

Умение	Мой критерий	Прошло из 5
Прямо: увод на 1 м	не больше … см	
Поворот 90°	ошибка не больше …°	
Крутая дуга своего поля	без схода	
Перекрёсток и поворот с него	на новую линию	
Остановка у груза	… ± 1 см	
Захват	груз не выпадает при развороте	

Умения, которые не прошли пять раз из пяти, отметь — с них начнёшь работу.

> ✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**
> Критерии записаны числами, у каждого умения — результат из пяти проб.

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.

План составлен по своей карточке. Для каждой доставки ты выбрал путь по буквам перекрёстков и можешь объяснить, почему именно его. Маршрут записан так, чтобы для другой карточки менялись только решения на перекрёстках.

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

Требование 1. Программа собрана из твоих проверенных программ уроков 4.2–5.2, а не из заготовки:

- действия — функции: линия до перекрёстка, решения на перекрёстке, подъезд к грузу, взять, поставить, разворот;
- маршрут — **данные**, отдельные от действий: для другой карточки меняются только строки маршрута, функции не трогаются;
- у каждой функции, которая чего-то ждёт, есть защита от бесконечного ожидания.

Своей программы из урока 5.2 нет — возьми у преподавателя решение урока 5.2 как основу; маршрут и защиты пишешь сам.

1. Запиши в журнал список функций своей программы: что делает каждая и в каком уроке проверена.
2. Отметь функции без защиты от бесконечного ожидания и добавь защиту.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Список функций с уроком проверки, маршрут отделён от действий, защиты добавлены.

★★ **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

Приёмка части 1. Робот от старта до груза 1: три раза из трёх, в том числе когда его поставили в зоне старта на 2 см левее или правее и чуть повёрнутым.

1. Запиши маршрут первой части в данные программы и проверь.
2. Каждую неудачу запиши: какая функция не сработала и почему.
3. Работает — сохрани «рабочую версию 1».

Запуск	Как стоял на старте	Результат	Что сорвалось
1	Ровно		
2	2 см левее		
3	2 см правее, повёрнут		

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Три запуска из трёх с разной постановкой на старте, неудачи разобраны.

7 Часть 2: первая доставка

12 мин

Приёмка части 2. Первая доставка три раза из трёх, в том числе когда груз сдвинут на 1 см в сторону от линии.

1. Допиши в данные маршрута первую доставку и проверь её отдельно: ставь робота перед грузом 1.
2. Как робот узнаёт место отпускания? Запиши, чем это место определяется в твоей программе и насколько точно.
3. Работает — сохрани «рабочую версию 2».

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Три доставки из трёх, включая сдвинутый груз; способ найти место отпускания и его точность записаны.

8 Часть 3: вторая доставка и финиш

15 мин

1. Допиши вторую доставку и путь к финишу. После финиша робот стоит.
2. Запусти весь маршрут от старта. Сохрани «рабочую версию 3».
3. **Проверь отделение маршрута от действий:** вытяни другую карточку, составь для неё маршрут и поменяй **только данные маршрута**. Робот проходит новый маршрут хотя бы один раз.
4. Запиши, какие части маршрута ненадёжны и что ты проверишь в уроке 6.2.

ДОЛЖНО ПОЛУЧИТЬСЯ

Маршрут своей карточки пройден целиком, маршрут второй карточки пройден после замены только данных. Список ненадёжных частей записан.

ПРОВЕРЬ СЕБЯ

- [ ] Для новой карточки я меняю только данные маршрута.
- [ ] 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

★★ 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. Проведи два пробных заезда подряд без исправлений между ними.
2. Для каждого запиши: на каком этапе первый срыв (выезд, линия, перекрёсток, подъезд, захват, разворот, постановка, финиш) и сколько секунд длился заезд.
3. Срыв в одном месте в обоих заездах или в разных? Что это говорит о причине?

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Два заезда в журнале с этапом срыва и временем, вывод о том, случайная ошибка или постоянная.

2 Найди и исправь ошибку

25 мин

Отлаживай по **доказательствам**, а не по догадке.

1. Добавь в программу диагностику: номер выполняемого решения маршрута и показания датчиков в момент решения выводятся в монитор порта. Подключи USB-кабель и запусти робота с ним, придерживая кабель; или зажигай светодиод L на каждом решении.
2. Для своей ошибки запиши **гипотезу** и **опыт, который может её опровергнуть**, — до того, как менять код.
3. Проведи опыт. Гипотеза не подтвердилась — запиши новую. В журнале должно быть не меньше двух кругов, если первый не дал ответа.
4. Исправление засчитывается, когда участок проходит пять раз из пяти.

Гипотеза	Опыт	Что показал	Вывод

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Диагностика работает, в журнале — гипотезы, опыты и выводы, участок проходит пять раз из пяти.

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.

Попроси напарника выдать тебе другую карточку и пройди её, поменяв только маршрут в программе. Если для этого нужно переписывать функции, программа ещё не готова к зачёту.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

2 ЗАНЯТИЕ 2 · 45 МИН
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. Напиши короткий инженерный отчёт в журнал: - надёжность по своим данным: сколько проб каждого участка прошло; - главная ошибка, гипотезы и опыты, которыми ты её нашёл; - самый большой оставшийся риск и одно изменение, которое бы его уменьшило, — и как ты проверил бы, что оно помогло.
4. Расскажи отчёт классу за две минуты.
5. Сохрани итоговый скетч, лист оценки и отчёт.

ПРОВЕРЬ СЕБЯ

- [ ] Я подтверждаю выводы о надёжности своими данными.
- [ ] Я знаю главный риск своей программы и как проверить его исправление.

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

★★ 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. Измерь пути «по времени» при скорости 100, 150 и 200. Построй в журнале таблицу «скорость — путь».
2. Оцени: можно ли по этой таблице пересчитать время для 50 см на скорости 120? Насколько ты доверяешь такому пересчёту завтра, на другом заряде батарей? Объясни.
3. Запиши, что нужно **измерять** во время езды, чтобы путь не зависел от этих причин.

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Таблица, оценка пересчёта и ответ — в журнале.

2 Как устроен энкодер

12 мин

1. Посчитай прорези, запиши контакты энкодеров из таблицы проводки.
2. Оцени частоту импульсов: колесо 45 мм, 20 прорезей, робот едет 0,3 м/с. Сколько импульсов в секунду даёт каждый энкодер?
3. Объясни в журнале, почему счёт делают прерыванием и почему на Arduino Uno для этого подходят только контакты 2 и 3.

БЕЗОПАСНОСТЬ

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

Подсказка: Частота импульсов

Сначала — сколько оборотов в секунду делает колесо при 300 мм/с.

ДОЛЖНО ПОЛУЧИТЬСЯ

Оценка частоты и объяснение — в журнале.

3 Импульсов на один оборот

22 мин

Требование 1. Программа считает импульсы каждого колеса прерыванием и печатает оба счётчика в монитор порта; счётчики читаются так, что прерывание не портит число. Без вписанных контактов программа не идёт дальше.

1. Напиши программу сам.

Как это устроено: Справка по командам

Команда	Что делает
`attachInterrupt(digitalPinToInterrupt(pin), f, RISING)`	Вызывать функцию `f` на каждый фронт импульса на контакте `pin`
`volatile long x`	Переменная, которую меняет прерывание
`noInterrupts()` / `interrupts()`	Выключить и снова включить прерывания

2. Измерь импульсы на оборот: прокрути колесо 10 оборотов по метке, потом **20 оборотов**. Насколько различаются два результата? Какой точнее и почему?
3. Найди положение колеса, при котором счёт растёт без движения. Объясни в журнале, что происходит в датчике.

Подсказка: Идея (1 из 2)

Функции прерывания должны быть короткими: только увеличить счётчик.

Подсказка: Счёт без движения (2 из 2)

На краю прорези свет попадает на приёмник «наполовину», и сигнал дрожит около порога.

ДОЛЖНО ПОЛУЧИТЬСЯ

Программа выполняет требование 1, импульсы на оборот измерены дважды, объяснение дрожания записано.

2

ЗАНЯТИЕ 2 · 45 МИН

Точное расстояние

4 Миллиметры в импульсы

15 мин

Требование 2. Функция `long ticksFor(int mm, int ticksPerRev, int diameterMm)` переводит миллиметры в импульсы с округлением до ближайшего целого и не переполняется на Arduino Uno.

Приёмка. Напиши функцию в тестовом скетче. Когда все строки `PASS`, перенеси её в свою программу.

Тест функции ticksFor LF1_Ticks_Test.ino

```
// Проверка функции ticksFor. Напиши функцию здесь, загрузи и открой м
// Когда все строки PASS, перенеси ticksFor в свою программу.

// Требование 2: сколько импульсов нужно, чтобы проехать mm миллиметро
long ticksFor(int mm, int ticksPerRev, int diameterMm) {
  return 0;
}

int failed = 0;

void expect(int mm, int ticksPerRev, int diameterMm, long expected) {
  long actual = ticksFor(mm, ticksPerRev, diameterMm);
  Serial.print(actual == expected ? "PASS " : "FAIL ");
  Serial.print("ticksFor(");
  Serial.print(mm);
  Serial.print(", ");
  Serial.print(ticksPerRev);
  Serial.print(", ");
  Serial.print(diameterMm);
  Serial.print(") = ");
  Serial.print(actual);
  Serial.print(", ожидаем ");
  Serial.println(expected);
  if (actual != expected) {
    failed++;
  }
}

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

  expect(0, 20, 45, 0);
  expect(141, 20, 45, 20);   // почти ровно один оборот
  expect(500, 20, 45, 71);   // 70.7 округляется вверх
  expect(1000, 20, 65, 98);  // другое колесо
  expect(1000, 40, 45, 283);
  expect(2000, 40, 45, 566); // 2000 × 40 не помещается в int

  if (failed == 0) {
```

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

void loop() {
}
```

Требование 3. Программа едет вперёд, пока **среднее** двух счётчиков не достигнет цели, останавливается и печатает цель и насчитанное после остановки. Без вписанного числа импульсов на оборот робот стоит.

1. Выполни требования 2 и 3, проедь 500 мм от метки и измерь путь.

Подсказка: Идея (1 из 2)

Импульсы = обороты × импульсы на оборот, обороты = путь / длина окружности.

Подсказка: Тест на 2000 мм не проходит (2 из 2)

`2000 * 40` — 80 000, больше, чем вмещает `int`. Считай в `float` или `long` до умножения.

ДОЛЖНО ПОЛУЧИТЬСЯ

Тест пройден, поездка на 500 мм сделана и измерена.

5 Почему робот проезжает дальше

15 мин

Требование 4. За `SLOW_TICKS` импульсов до цели робот сбавляет скорость, чтобы меньше катиться после остановки.

1. Выполни требование 4 и подбери `SLOW_TICKS` и пониженную скорость.
2. Сравни три заезда без замедления и три с замедлением: среднюю ошибку и разброс.
3. Объясни в журнале, чем замедление лучше поправки «остановиться раньше на N импульсов».

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

Требование 4 выполнено, сравнение двух способов и вывод — в журнале.

6 Задания по уровням

12 мин

★★ **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

★★ 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.

Сначала спланируй опыт, потом пиши код.

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

Подсказка: Какое число брать из трёх заездов

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

✓ **ДОЛЖНО ПОЛУЧИТЬСЯ**

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

2 Робот измеряет сам

15 мин

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

Каркас: измерение заезда LF2_Compare_Skeleton.ino

```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

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

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

// Измерения белого и чёрного из журнала (урок 3.1).
const int LEFT_WHITE = 0;
const int LEFT_BLACK = 0;
const int RIGHT_WHITE = 0;
const int RIGHT_BLACK = 0;

const int LEFT_DIR_PIN = 7;
const int LEFT_SPEED_PIN = 6;
const int RIGHT_DIR_PIN = 4;
const int RIGHT_SPEED_PIN = 5;
const int FORWARD = HIGH;

// Регулятор заезда: 0 — релейный, 1 — П, 2 — ПД, 3 — ПИД.
const int MODE = 1;
int baseSpeed = 100; // одна скорость для всех регуляторов
float kp = 0.5;      // твой kp из урока 3.2
float kd = 0;        // Д: реакция на то, как быстро уходит линия
float ki = 0;        // И: реакция на ошибку, которая копится
const long SUM_LIMIT = 3000;       // предел накопленной ошибки
const unsigned long RUN_MS = 6000; // длина заезда: робот должен 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 // Требование 2: здесь будет int controller(int mode, int error).
55
56 // Перенеси свою функцию level из урока 3.2.
57 int level(int value, int white, int black) {
58   return 0;
59 }
60
61 void drive(int leftDir, int leftSpeed, int rightDir, int rightSpeed)
62   digitalWrite(LEFT_DIR_PIN, leftDir);
63   digitalWrite(RIGHT_DIR_PIN, rightDir);
64   analogWrite(LEFT_SPEED_PIN, leftSpeed);
65   analogWrite(RIGHT_SPEED_PIN, rightSpeed);
66 }
67
68 // ===== Измерение заезда =====
69
70 const int MARKER = 2610; // признак того, что в памяти записан заезд
71 Result run;
72 int lastSide = 0;
73 unsigned long startTime;
74 unsigned long nextStep;
75
76 // Добавляет ошибку шага к итогам заезда.
77 void record(int error) {
78   run.steps++;
79   run.sumAbs += abs(error);
80   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, LEI
    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;
}
```

```
163
164 void loop() {
165   if (millis() - startTime >= RUN_MS) {
166     drive(FORWARD, 0, FORWARD, 0);
167     EEPROM.put(0, run);
168     Serial.println("Заезд окончен:");
169     printResult(run);
170     courseTelemetryReport(analogRead(LEFT_SENSOR_PIN), analogRead(RIGI
171     while (true) {
172       courseTelemetryPoll();
173     }
174   }
175   while (millis() < nextStep) {
176     courseTelemetryPoll();
177   }
178   nextStep += STEP_MS;
179
180   float observedError = NAN;
181   float observedOutput = NAN;
182   // Требования 1–3: ошибка, record(error), регулятор по MODE и скоро
183   // Для графиков сохрани свои результаты в observedError и observedOu
184   drive(FORWARD, 0, FORWARD, 0);
185   courseTelemetryReport(analogRead(LEFT_SENSOR_PIN), analogRead(RIGHT_
186                         observedError, observedOutput);
187 }
188
189 // ===== Готовое: передача наблюдений — не меняй =====
190 void courseTelemetryBegin() {
191 #if defined(ROBOT_TELEMETRY_BUILD_ID)
192   using namespace robot_telemetry::wire;
193   courseTelemetry.begin((1UL << Line0) | (1UL << Line1) |
194                         (1UL << LineError) | (1UL << ControlOutput));
195 #else
196   Serial.begin(9600);
197 #endif
198 }
199
200 void courseTelemetryPoll() {
201 #if defined(ROBOT_TELEMETRY_BUILD_ID)
202   courseTelemetry.update();
203 #endif
```

```
}

// NAN означает «ещё не вычислено», а не нулевую ошибку или поправку.
void courseTelemetryReport(int leftRaw, int rightRaw, float error, flo
#if defined(ROBOT_TELEMETRY_BUILD_ID)
  using namespace robot_telemetry::wire;
  courseTelemetry.setUnsigned(Line0, leftRaw);
  courseTelemetry.setUnsigned(Line1, rightRaw);
  if (isnan(error)) courseTelemetry.setMissing(LineError);
  else courseTelemetry.setFloat(LineError, error);
  if (isnan(output)) courseTelemetry.setMissing(ControlOutput);
  else courseTelemetry.setFloat(ControlOutput, output);
  courseTelemetry.update();
#endif
}

void courseWait(unsigned long durationMs) {
  unsigned long started = millis();
  while (millis() - started < durationMs) {
    courseTelemetryPoll();
    delay(1);
  }
}
```

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

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

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

1. Перенеси `level` и ошибку из урока 3.2.
2. Выполни требование 1 для режимов 0 и 1. Подбери `RUN_MS` так, чтобы робот останавливался до конца дорожки.

3. Найди в `record` порог 10. Запиши в журнал, что изменится в числе переходов, если порог убрать, и почему.

БЕЗОПАСНОСТЬ

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

Подсказка: Робот не трогается со старта

`waitForStart` ждёт белого по шкале `level`. Пока `level` возвращает 0, робот не стартует никогда — это защита каркаса.

ДОЛЖНО ПОЛУЧИТЬСЯ

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

3 Релейный против П

20 мин

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

ДОЛЖНО ПОЛУЧИТЬСЯ

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

2

ЗАНЯТИЕ 2 · 45 МИН

ПД, ПИД и общий итог

4 ПД: тормозим поворот заранее

14 мин

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

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

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

Тест функции controller LF2_Controller_Test.ino

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

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

int lastError = 0;
long errorSum = 0;

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

int failed = 0;

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

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

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

```

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

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

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

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

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

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

```

```
81 void loop() {
82 }
```

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

Подсказка: Идея (1 из 3)

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

Подсказка: План (2 из 3)

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

Подсказка: Фрагмент кода (3 из 3)

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

ДОЛЖНО ПОЛУЧИТЬСЯ

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

5 ПИД: что даёт накопленная ошибка

11 мин

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

Подсказка: Когда И-часть полезна

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

ДОЛЖНО ПОЛУЧИТЬСЯ

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

6 Сравнение всех четырёх

12 мин

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

ДОЛЖНО ПОЛУЧИТЬСЯ

Итоговая таблица и вывод из трёх частей — в журнале.

7 Задания по уровням

8 мин

★★ **ПРОДВИНУТЫЙ УРОВЕНЬ · ПРЕДЕЛЬНАЯ СКОРОСТЬ**

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

ПРОВЕРЬ СЕБЯ

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

Проверь себя

★★ Продвинутый уровень

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

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

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

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

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

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

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

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

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

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

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

Отвечай на вопросы своего уровня. Нужно верных ответов: Начальный — 4 из 5; Средний — 4 из 5; Продвинутый — 4 из 5. Ответы проверит преподаватель или программа курса.

Итог

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

Как подтвердить результат

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

Словарь урока

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

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:

_____