УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

по робототехнике и программированию

Для робототехнического
конструктора для обучения
«Избушка»

Грань
Новые Технологии

Содержание

1. ВВЕДЕНИЕ В РОБОТОТЕХНИКУ 5
Лекция: История и развитие робототехники 5
1. Истоки робототехники 5
2. Индустриальная революция и автоматизация 5
3. Роботы XX века и первые шаги к интеллекту 5
4. Современная робототехника и промышленные роботы 5
5. Роботы в повседневной жизни 5
6. Искусственный интеллект и будущее робототехники 6
Лекция: "Современные роботы и их применение". 6
1. Промышленные роботы 6
2. Роботы в медицине 6
3. Сервисные роботы 6
4. Автономные транспортные средства 6
5. Военные и спасательные роботы 6
6. Роботы в космосе и научных исследованиях 7
7. Гуманоидные роботы и их перспективы 7
Практическое занятие: "Обзор робототехнического конструктора Избушка". 7
2. ОСНОВЫ ЭЛЕКТРОНИКИ И МЕХАНИКИ 7
Лекция: "Основные компоненты электроники: резисторы, конденсаторы, транзисторы". 7
Практическое занятие: "Сборка простой электрической цепи". 15
Лекция: "Основы механики: двигатели, редукторы, крепежные элементы". 17
Практическое занятие: "Сборка механической части робота". 18
3. ПРОГРАММИРОВАНИЕ ДЛЯ РОБОТОТЕХНИКИ 19
Лекция: "Основы программирования на языке C для микроконтроллеров". 19
1. Переменные и типы данных 19
2. Ввод и вывод данных 19
3. Условные операторы 20
4. Циклы 20
5. Функции 20
6. Массивы 21
7. Строки 21
8. Указатели 21

9. Структуры 21
10. Работа с файлами 22
11. Препроцессор и директивы 22
12. Циклы и управление потоком выполнения 23
Практическое занятие: "Написание первой программы для Arduino Mega 2560" 23
Лекция: " Основы программирования на языке Python". 25
1. Переменные и типы данных 25
2. Ввод и вывод данных 25
3. Условные операторы 25
4. Циклы 26
5. Функции 26
6. Списки 26
7. Кортежи 27
8. Словари 27
9. Множества 27
10. Обработка исключений 27
11. Работа с файлами 28
12. Модули 28
13. Списковые включения (List Comprehensions) 28
14. Лямбда-функции 28
15. Классы и объекты 28
Практическое занятие: "Создание программы на Python для управления роботом с одноплатным компьютером (Одноплатный комрьютер OrangePI или аналогичный входит в комплект обучающего робототехнического конструктора “Избушка”)". 29
4. СЕНСОРЫ И МОДУЛИ 30
Лекция: Обзор сенсоров: датчики расстояния, температуры, света 30
1. Введение в сенсоры 30
2. Датчики расстояния 30
3. Датчики температуры 30
4. Датчики света 30
5. Принципы работы и применения сенсоров 30
6. Примеры применения сенсоров 31
7. Перспективы развития сенсорных технологий 31
Практическое занятие: "Подключение и программирование датчиков (набор 37 датчиков и сенсоров входит в комплектацию обучающего робототехнического конструктора

“Избушка”) к Arduino Mega 2560 (входит в комплектацию обучающего робототехнического конструктора “Избушка”)" 31
5. РАСПОЗНАВАНИЕ И ОБРАБОТКА ДАННЫХ 34
Лекция: "Основы обработки сигналов и данных". 34
1. Введение в обработку сигналов и данных 34
2. Типы сигналов 34
3. Основные этапы обработки сигналов 34
4. Фильтрация сигналов 34
5. Применение обработки сигналов 35
6. Методы анализа данных 35
7. Заключение 35
Практическое занятие: "Фильтрация данных с сенсоров (набор 37 датчиков и сенсоров входит в комплектацию обучающего робототехнического конструктора “Избушка”) на Orange Pi (входит в комплектацию обучающего робототехнического конструктора “Избушка”) с использованием Python" 36
Лекция: Введение в машинное обучение и нейронные сети 39
1. Определение и значение машинного обучения 39
2. Основные типы машинного обучения 39
3. Введение в нейронные сети 39
4. Архитектуры нейронных сетей 39
5. Обучение нейронных сетей 40
6. Применение машинного обучения и нейронных сетей 40
7. Заключение 40
Практическое занятие: "Создание простой программы с ИИ на основе языка программирования Python для одноплатного компьютера Orange PI (входит в комплектацию обучающего робототехнического конструктора “Избушка”)". 40
6. ПРОЕКТНАЯ РАБОТА 43
7. ЗАКЛЮЧЕНИЕ И ДАЛЬНЕЙШЕЕ РАЗВИТИЕ 43

1. ВВЕДЕНИЕ В РОБОТОТЕХНИКУ

Цель:

- Ознакомить учащихся с основными понятиями и историей робототехники.
- Показать разнообразие роботов и их применение в различных областях.

Содержание:

Лекция: История и развитие робототехники

1. Истоки робототехники

Робототехника как концепция восходит к древним временам. Первые упоминания о механизмах, имитирующих человека, встречаются в античной литературе. Например, греческие мифы описывали автоматов, созданных богами, таких как механический гигант Талос. Уже в средние века Леонардо да Винчи спроектировал «механического рыцаря» — одного из первых гуманоидных роботов. Однако эти проекты оставались на уровне идей и примитивных механизмов.

2. Индустриальная революция и автоматизация

С наступлением индустриальной революции в XVIII-XIX веках начался активный прогресс в области механизмов и автоматов. Были созданы автоматизированные машины для производства и заводов, что стало основой для будущей робототехники. Изобретатели начали разрабатывать механизмы, которые могли выполнять повторяющиеся задачи без человеческого участия.

3. Роботы XX века и первые шаги к интеллекту

В XX веке термин "робот" получил популярность благодаря пьесе Карела Чапека "R.U.R." (1920), где были изображены искусственные люди-роботы. В середине века разработки в области электроники, кибернетики и программирования способствовали созданию первых программируемых роботов. Например, в 1954 году Джордж Девол создал первый промышленный робот Unimate, который работал на конвейере завода General Motors.

4. Современная робототехника и промышленные роботы

С развитием компьютерных технологий в 1970-1980-х годах робототехника получила новый импульс. Промышленные роботы стали использоваться в различных отраслях: от автомобильного производства до сборки электроники. В это же время развивалась идея создания автономных роботов, способных к самообучению и взаимодействию с окружающей средой.

5. Роботы в повседневной жизни

В XXI веке робототехника вышла за пределы производственных процессов и стала частью повседневной жизни. Были разработаны роботы для дома (например, робот-пылесосы), медицины (роботы-хирурги), а также военные и исследовательские роботы. Также началось активное развитие автономных транспортных средств, таких как беспилотные автомобили и дроны.

6. Искусственный интеллект и будущее робототехники

Одним из главных направлений современного развития робототехники является интеграция искусственного интеллекта (ИИ), что позволяет роботам обучаться, принимать решения и адаптироваться к окружающей среде. В ближайшие годы ожидается дальнейший прогресс в разработке гуманоидных роботов, роботов-помощников, а также развитие технологий взаимодействия человека и машины.

Лекция: "Современные роботы и их применение".

1. Промышленные роботы

Промышленные роботы — это основа автоматизации производства. Они используются на заводах для выполнения рутинных, опасных или требующих высокой точности операций. Например, роботы на автомобильных заводах занимаются сборкой, сваркой и покраской машин. Благодаря их применению, увеличивается производительность, снижается риск для работников и повышается качество продукции.

2. Роботы в медицине

Современные медицинские роботы значительно расширяют возможности диагностики и лечения. Роботы-хирурги, такие как система da Vinci, используются для проведения сложных операций с высокой точностью. Также широко применяются роботы для реабилитации пациентов, доставки лекарств в больницах и даже для помощи в уходе за пожилыми людьми.

3. Сервисные роботы

Сервисные роботы находят широкое применение в повседневной жизни. Роботы-пылесосы, такие как Roomba, убирают помещения, а кухонные роботы помогают готовить пищу. В отелях и ресторанах роботы могут выполнять функции консьержей, доставлять еду или регистрировать гостей. Это снижает нагрузку на персонал и увеличивает удобство для пользователей.

4. Автономные транспортные средства

Роботы-транспортные средства, такие как беспилотные автомобили и дроны, уже начинают активно внедряться в нашу жизнь. Беспилотники используются для доставки товаров, мониторинга сельскохозяйственных полей и даже в поисково-спасательных операциях. Автономные автомобили находятся на стадии тестирования и обещают революцию в сфере пассажирских и грузоперевозок.

5. Военные и спасательные роботы

Роботы играют важную роль в оборонной сфере. Военные роботы применяются для разминирования, разведки и даже в боевых действиях. Роботы-сапёры обезвреживают взрывные устройства, снижая риск для людей. В спасательных операциях роботы могут использоваться для поиска пострадавших в зонах стихийных бедствий, работы в опасных условиях, где человеку было бы трудно или опасно.

6. Роботы в космосе и научных исследованиях

Космические исследования невозможны без роботов. Марсоходы, такие как Perseverance, исследуют планеты и доставляют научные данные на Землю. Подводные роботы применяются для изучения океанских глубин, а роботизированные системы помогают исследователям проводить эксперименты в условиях, где присутствие человека ограничено или невозможно.

7. Гуманоидные роботы и их перспективы

Гуманоидные роботы, такие как Atlas от Boston Dynamics или ASIMO от Honda, разрабатываются для взаимодействия с людьми. Они способны выполнять сложные задачи в реальном мире, от помощи в доме до работы в условиях чрезвычайных ситуаций. Их дальнейшее развитие обещает сделать роботов важными помощниками в самых разных сферах человеческой жизни, улучшая качество жизни и повышая эффективность труда.

Практическое занятие: "Обзор робототехнического конструктора Избушка".

2. ОСНОВЫ ЭЛЕКТРОНИКИ И МЕХАНИКИ

Цель:

- Научить основам электроники, необходимым для создания роботов.
- Ознакомить с основными механическими компонентами роботов.

Содержание:

Лекция: "Основные компоненты электроники: резисторы, конденсаторы, транзисторы".

1. Резисторы

На вид резисторы - маленькие элементы, несущие на себе разноцветные полоски. Если вы хотите узнать значение сопротивления резистора, можно измерить его мультиметром, либо определить это значение по цветам полосок. Каждому цвету полоски соответствует число (рисунок 1).

Цветовые обозначения резисторов:

цвет	значение	цвет	значение
Черный	0	Синий	6
Коричневый	1	Фиолетовый	7
Красный	2	Серый	8
Оранжевый	3	Белый	9
Желтый	4	Золотистый	1/10
Зеленый	5	Серебристый	1/100

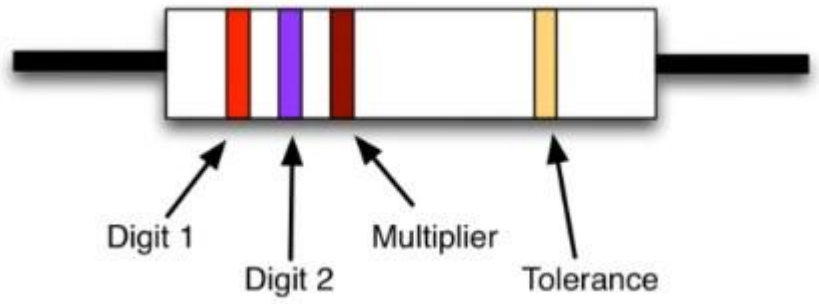

Рисунок 1.- Расшифровка цветовой маркировки резистора

ПРИМЕЧАНИЕ

Золотистый и серебристый цвет указывают не только доли, соответственно. 1/10 и 1/100, но и точность резистора. Так, золотистый цвет означает ±5%, а серебристый - ±10%.

Как правило, на одном конце резистора присутствуют три такие полоски, далее идут пробел и еще одна полоска на другом конце резистора, которая означает точность значения резистора.

На рисунке показано расположение цветных полос. Значение резистора определяется по трем первым полоскам: первая соответствует первой цифре, вторая - второй, а третья - «множитель», указывает, сколько нулей нужно поставить после первых двух цифр.

Исходя из сказанного, резистор, изображенный на рисунке имеет сопротивление 270 Ом: первая цифра - 2 (красная полоска), вторая - 7 (фиолетовая полоска), а множитель - 1 (коричневая полоска), добавляющий 0 после цифр 27. Аналогично, на резисторе сопротивлением 1 кОм будут нанесены коричневая, черная и красная полосы (1, 0. 00).

Резисторы также различаются и по номинальной мощности. Практически все обычные резисторы того типа, что используются в схемах этой книги (для монтажа в сквозные отверстия), обладают мощностью 0,25 Вт. Другие распространенные значения номинальной мощности: 0,5 Вт, 1 Вт и 2 Вт, причем, чем выше номинальная мощность резистора, тем крупнее он сам.

2. Транзисторы

Как правило, любой поставщик комплектующих предлагает немыслимое количество самых разных транзисторов (рисунок 2). Чтобы не усложнять поиск, подобрано всего четыре транзистора, на основе которых можно создавать практически все электронные схемы, способные управлять различными устройствами, подключенными к платам Arduino и Raspberry Pi.

Транзистор из разд. «Эксперимент: управление электродвигателем» это как раз тот самый компонент, при помощи которого слабый ток силой несколько миллиампер позволяет управлять сотнями миллиампер, подаваемыми на двигатель. Хотя транзисторы могут использоваться и для других целей, в этой книге они послужат нам переключателями. Малый ток поступает на базу транзистора и далее на заземляющий вывод (GND) через эмиттер транзистора. Этот ток и будет переключать гораздо более сильный ток, поступающий с коллектора на эмиттер. На рисунке ниже показана подборка транзисторов различных типов, позволяющих работать с разными мощностями.

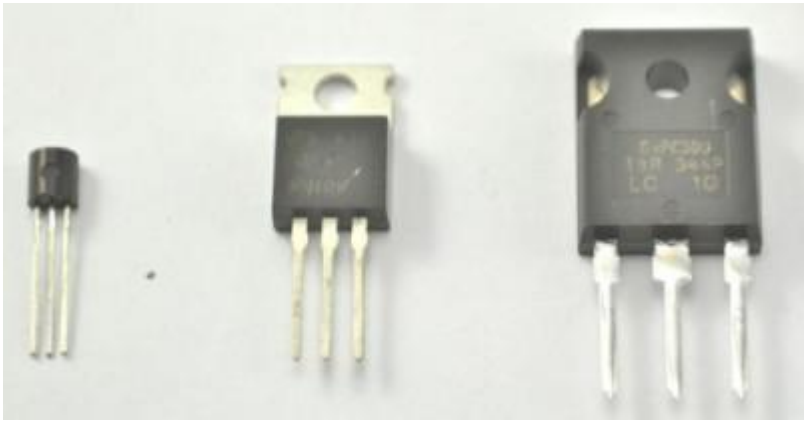

Рисунок 2.- Подборка транзисторов

Существует всего несколько разновидностей корпусов транзисторов. При этом по внешнему виду транзистора определить его свойства невозможно - нужно прочитать, что на нем написано.

Наиболее распространенные варианты корпусов: TO-92 (рис., слева) и TO-220 (рис., в центре). Иногда, для работы на очень сильных токах, могут использоваться транзисторы с более крупным корпусом- как вариант TO-247 показанный на рис., справа. Модели TO-220 и TO-247 предназначены для монтажа на теплоотводах. Впрочем, если вы используете эти транзисторы при токах, которые значительно меньше указанного для них максимума, то устанавливать их на теплоотводах необязательно.

3. Биполярные транзисторы

Транзисторы изготавливаются по разным технологиям, каждой из которых присущи свои достоинства и свои недостатки. Поэтому транзистор может подойти вам в одной ситуации и не пригодиться в другой.

Начиная работать с транзисторами, вы, скорее всего, столкнетесь именно с биполярной моделью. Они практически не изменились со времен зарождения транзисторной отрасли. Преимущество таких транзисторов - в их дешевизне и в том, что их очень легко использовать при малых нагрузочных токах. Есть у них и недостаток - хотя малый ток, проходящий через базу к эмиттеру транзистора, позволить прохождение более сильного тока, идущего от коллектора к эмиттеру, ток коллектора ограничивается множителем тока базы, и этот множитель (называемый коэффициентом усиления, или hFE), как правило, находится в диапазоне от 50 до 200 Соответственно, если Raspberry Pi подает на базу ток силой 2 мА, то через коллектор может идти ток не более 100 мА. Эта величина гораздо меньше той, на которую вы, возможно, рассчитывали, поскольку транзистор вполне может выдерживать и более сильный ток (скажем, 500 мА), но она никогда не достигнет такого предела - ведь тока на базе просто не хватит. Как правило, при работе с Arduino такая проблема не возникает, поскольку Arduino позволяет подавать на базу больший ток (до 40 мА), если вместо резистора сопротивлением 1 кОм применить более слабый резистор. Так, если взять резистор со значением 150 Ом, то ток базы увеличится до:

1-V/R=(5-0,5)/150=30 мА.

И даже в худшем случае если транзистор имеет коэффициент усиления 50 - ток базы силой 30 мА даст на коллекторе ток силой 1,5 А.

Тут надо пояснить: в только что рассмотренном примере напряжение рассчитыва ется как (5 - 0,5), поскольку напряжение между базой и эмиттером биполярного транзистора, когда транзистор включен, составляет около 0,5 В.

В этой книге мы будем работать всего с одной моделью биполярного транзисто ра весьма распространенным 2N3904. Хотя в наличии есть биполярные транзисторы, выдерживающие более сильный ток, при увеличении силы тока лучше работать с более высокотехнологичными транзисторами.

4. Составные транзисторы

Когда вам требуется более значительное усиление если, к примеру, вы управляете небольшим электродвигателем с платы Raspberry Pi, позволяющей подать на базу всего несколько миллиампер, то вместо обычного биполярного транзистора удобно взять составной транзистор (так называемую пару Дарлингтона), который обычно имеет коэффициент усиления не менее 10 000.

Составной транзистор содержит два биполярных транзистора в одном корпусе (рисунок 3), и именно благодаря такой двухкомпонентной структуре составные транзисторы получают столь высокий коэффициент усиления.

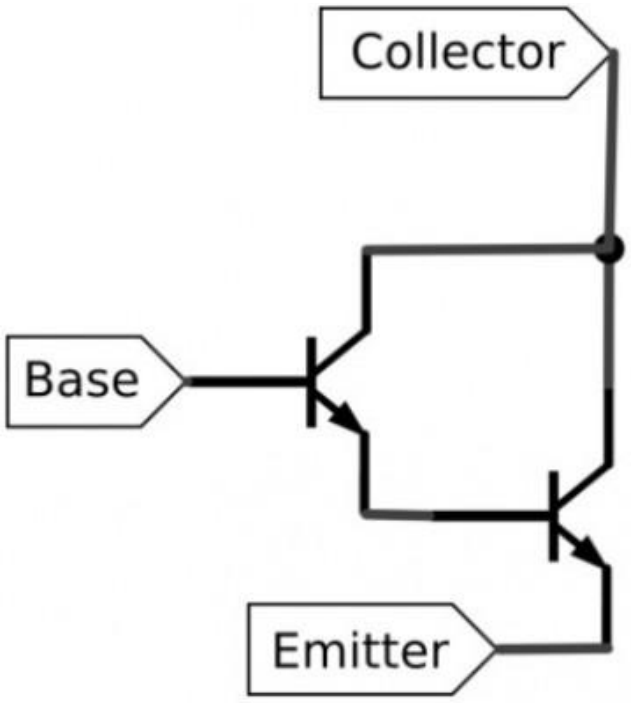

Рисунок 3.- Составной транзистор

Поскольку в составном транзисторе имеются два компонента с базой и эмиттером в каждом, при включенном транзисторе в каждой такой паре перепад напряжения составит как минимум 0,5 В, что дает общий перепад напряжения 1 В, а не 0,5 В. как в обычном биполярном транзисторе. На самом деле, этот перепад касается и напряжения коллектора и увеличивается с повышением нагрузочного тока. Таким образом, когда мы управляем током в 1 А, составной транзистор MPSA14 на практике способен дать нам всего 9 В, питая нагрузку в 12 В. В одних случаях это имеет значение, в других - нет.

Перепад напряжения на резисторе R1 при использовании Raspbertу Pi на самом деле составит не 3,3 В, а 3,3 В 1 В = 2,2 В. Соответственно, Raspberry Pi потребуется подавать ток 2,2 В / 1 кОм 2,2 мА.

Кроме маломощного транзистора MPSA14, который удобен для управления нагрузками до 0,5 В, также рекомендую вам работать с более мощным составным транзистором TIP120, который является стандартным устройством и обязательно должен присутствовать в вашем ящике с комплектующими.

5. МОП-транзисторы

В принципе, биполярный транзистор представляет собой управляемое током устройство: малый базовый ток усиливается и превращается в большой ток коллектора. Однако существует еще один вид транзисторов - так называемые полевые, или МОП-транзисторы (сокращение от «металл-оксид-полупроводник»), требующие для переключения очень малого тока, но остающиеся во включенном состоянии, пока напряжение на их входном затворе превышает некоторое пороговое значение.

На рисунке 4 схематически представлен такой транзистор - как вы можете догадаться, на этой схеме входной его компонент (затвор) электрически не соединен с остальными компонентами транзистора напрямую.

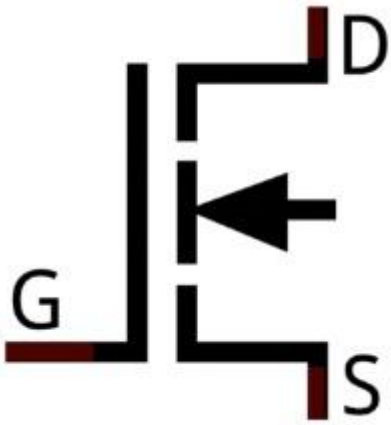

Рисунок 4.- Схема МОП-транзистора

Обратите внимание: в отличие от биполярного транзистора, компоненты которого назывались базой, коллектором и эмиттером, компоненты МОП-транзистора называются затвор (G, от англ. gate), сток (D, от англ. drain) и исток (S, от англ. source). Напрашивается мысль, что исток эквивалентен коллектору, но на самом деле коллектор из биполярного транзистора эквивалентен стоку полевого.

МОП-транзисторы (рисунок 5) при работе с Arduino или Raspberry Pi весьма целесообразно применять для включения/отключения устройств схемы, поскольку в таком случае можно обойтись минимальным током. Нужно просто убедиться, что напряжение на затворе транзистора выше его порогового значения. Пороговым значением затвора является такое, при котором МОП-транзистор включается, пропуская ток от стока к истоку. На рисунке показано, как подключить МОП-транзистор для управления нагрузкой. Как можно видеть, на самом деле подключение точно такое же, что и в случае биполярного транзистора.

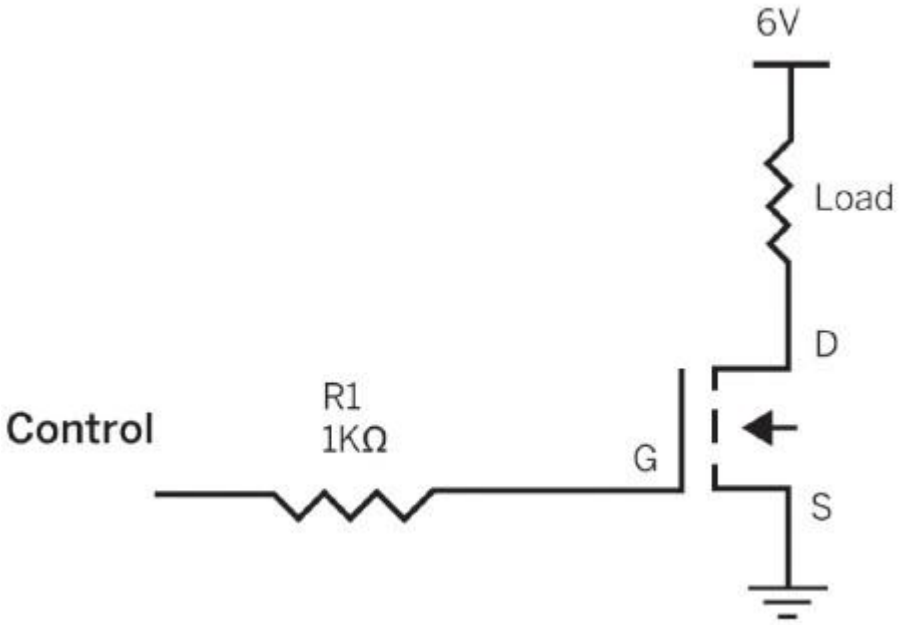

Рисунок 5.- Схема включения МОП-транзистора

В схеме, показанной на рисунке используются два символа, с которыми мы ранее не сталкивались. От истока транзистора (S) отходит линия, несущая три параллельные линии, которые постепенно укорачиваются. Это символ заземления - применяя его на схеме, мы избавляемся от лишних соединительных линий.

Второй символ расположен в верхней части схемы это просто горизонтальная черта, которой отмечены 6 В. Она означает, что напряжение в этой части схемы составит 6 В, что избавляет нас от необходимости рисовать батарею.

Мы и стараемся использовать транзисторы со взаимно совместимыми цоколевками (расположением выводов), это правило не универсально. Не все транзисторы в схожих корпусах имеют одинаковую цоколевку, поэтому перед применением нового транзистора сверяйтесь с его паспортом.

Возможно, после рассмотрения рисунка выше у вас возникнет вопрос - а зачем нам по-прежнему нужен резистор R1, ведь затвор не принимает никакого тока? Дело в том, что иметь этот резистор все равно целесообразно, ведь стоит нам увеличить напряжение на затворе и на долю секунды возникнет молниеносный бросок тока. Резистор гарантирует, что такой бросок не сожжет контакт GPIO.

Сложность работы с МОП-транзисторами в том, что иногда их пороговое значение слишком высоко, и его не удается переключить при помощи напряжения в 3,3 В с Raspberry Pi или 5 В с Arduino. МОП-транзисторы, чье пороговое значение затво а достаточно низкое, чтобы их можно было переключать непосредственно с GPIO-контактов наших плат, называются МОП-транзисторами с логическим уровнем входа (logic-level). Для этой книги выбраны два стандартных МОП-транзистора: на низких мощностях используется 2N7000, а на сравнительно высоких - FOP30N061. Пороговое значение на затворе у обоих гарантированно не превышает 3 В, поэтому их можно использовать как с Arduino, так и с Raspberry Pi

В целом МОП-транзисторы при переключении нагрузок нагреваются не так сильно, как биполярные. Один из основных параметров МОП-транзисторов, на который нужно обращать внимание при покупке компонента, называется сопротивлением открытого канала (on resistance). МОП-транзисторы с очень низким сопротивлением открытого

канала будут переключать большие токи, даже не нагреваясь. Как вы догадываетесь, чем ниже сопротивление открытого канала у МОП-транзистора, тем он дороже.

Мы будем достаточно много работать с МОП-транзисторами. В основном я буду рассказывать о FQP30N06L (токи до 30 А), но для работы на таких токах вам понадобится большой теплоотвод.

Кстати, коллектор, база и эмиттер составного транзистора TIP120 и исток, затвор и сток МОП-транзистора FQP30N06L расположены одинаково, поэтому вы можете просто извлечь из макетной платы TIP120 и поставить на его место FQP30N06L не меняя конфигурации - при этом схема все равно должна работать.

6. PHP-транзисторы и транзисторы с p-каналом

Транзисторы каждого из всех ранее описанных типов на самом деле существуют в двух разновидностях. До сих пор мы рассматривали всего одну их разновидность: отрицательно-положительно-отрицательные транзисторы (NPN, от англ. negative-positive-negative), они же транзисторы с n-каналом (в случае МОП-транзисторов) Такие транзисторы - наиболее распространены, и обычно ими вполне можно обойтись.

Транзисторы другого вида - это положительно-отрицательно-положительные (PNP, от англ. positive-negative-positive) устройства, они же транзисторы с p-каналом. Если устройства N-типа применяются для переключения нагрузки на заземление, то устройства P-типа переключают нагрузку на источник положительного питания. МОП-транзисторы с p-каналом используются в мостовых схемах управления двигателем, рассматриваемых далее.

Как подбирать транзистор?

Подобрать правильный транзистор порой бывает непросто. С помощью таблицы проблема выбора у вас сведена всего к пяти транзисторам.

При работе с Raspberry Pi имеется в виду, что между GPIO-контактом и базой или затвором транзистора стоит резистор сопротивлением 1 кОм. В случае с Arduino предполагается, что сопротивление такого резистора будет 150 Ом. Приведенные значения получены по результатам тестирования реальных устройств, а максимальные значения напряжения взяты из спецификаций изделий.

Таблица. Полезный набор транзисторов

Transistor	Type	Package	Max. current (Pi 3.3V)	Max. current (Arduino 5V)	Max. volts
2N3904	Bipolar	TO-92	100mA	200mA	40V
2N7000	MOSFET	TO-92	200mA	200mA	60V
MPSA14	Darlington	TO-92	1A	1A	30V

Полезный набор транзисторов

TIP120	Darlington	TO-220	5A	5A	60V
FQP30N06L	MOSFET	TO-220	30A	30A	60V

Полезный набор транзисторов

Убедитесь, что FQP30N061 это МОП-транзистор версии L (логический) название модели оканчивается именно на L, в противном случае пороговое напряжение затвора может быть слишком высоким.

Транзистор MPSA14 является практически универсальным для работы с токами до 1 А. хотя при таких токах перепад напряжения составляет почти 3 В, и транзистор разогревается до 120°С! При токе 500 мА перепад напряжения уже не такой страшный всего 1,8 В. и температура транзистора 60°С.

Итак, если вам требуется переключать токи всего лишь около 100 мА, то вам вполне хватит транзистора 2N3904. Если нужен 1 А, используйте транзистор MPSA14. При более сильных токах наилучший вариант, пожалуй, это транзистор FQP30N06L, если, конечно, цена вас не смущает, поскольку транзистор TIP120 существенно дешевле.

7. Диоды

Дело в том, что электродвигатели создают скачки напряжения и всевозможные электрические помехи, которые могут устроить настоящий хаос в такой тонкой электронике, как Raspberry Pi или Arduino. Диод гарантирует, что всплески тока, спровоцированные двигателем, не приведут к мгновенному изменению направления тока, что может сразу сжечь транзистор. Суть в том, что диод пропускает ток только в одном направлении - в этом он похож на обратный клапан. Ток может идти через диод лишь в том направлении, куда указывает сам диод, по своей форме напоминающий стрелку.

Из-за таких свойств диодов их довольно часто ставят на выводах двигателя. Обычно ток идет через двигатель в направлении, обратном тому, которое допускает диод, но если случится отрицательный скачок напряжения, то диод вступает в дело, проводит ток и обнуляет его, фактически гася тем самым короткое замыкание.

8. Светодиоды

Как вы уже догадываетесь, светодиод работает как самый обычный диод, однако когда через него проходит ток, он излучает свет. Схемный символ светодиода точно такой же, что и у обычного диода, но добавленные справа стрелки означают световое излучение (рисунок 6).

Рисунок 6.- Символ диода

Светодиоды бывают самых разных цветов и размеров. Ими можно управлять непосредственно с GPIO-контакта Arduino или Raspberry Pi, однако, как и при работе с транзистором, необходимо использовать резистор для ограничения силы тока. О том, как это делается, рассказано позднее.

9. Конденсаторы

Можно сказать, что конденсатор предназначен для временного запасания электричества примерно, как очень малоемкие батарейки, сохраняющие небольшой резервный заряд. Конденсаторы мы станем использовать в проектах книги в различных

качествах - с их помощью мы будем подавлять электрические помехи, а также запасать небольшие резервы электроэнергии, чтобы при резкой необходимости добавить энергию, ее можно было взять с конденсатора.

На рисунке 7 показаны символы конденсаторов. Если конденсатор имеет очень высокое значение емкости, то он обычно поляризуется. Малоемкие конденсаторы не имеют положительного и отрицательного полюса.

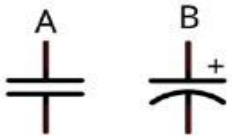

Рисунок 7.- Символы конденсатора: неполяризованный (А) и поляризованный (Б)

Иногда встречаются немного иные, но тем не менее узнаваемые символы конденсатора: в них положительный полюс поляризованного конденсатора обозначается пустой рамкой, а отрицательный - закрашенной. В этой книге конденсаторы обозначаются символикой, принятой в США.

10. Интегральные схемы

Интегральные схемы (ИС), часто именуемые просто чипами или микросхемами, состоят из множества транзисторов, диодов и других электронных компонентов, размещенных на кремниевом кристалле и соединенных между собой в сложные схемы. На печатных платах и Raspberry Pi и Arduino расположено множество таких микросхем и прочих электронных элементов.

Существуют интегральные микросхемы специального назначения, предназначенные для работы в любых типах электронных устройств, с которыми бы вам только захотелось иметь дело. Но в этой книге нам особенно важны микросхемы, которые позволяют управлять электронными устройствами. В корпусе таких микросхем часто объединяются сильноточные транзисторы и логические схемы управления.

Практическое занятие: "Сборка простой электрической цепи".

Цель занятия:

Научиться собирать простую электрическую цепь на основе робототехнического конструктора для обучения “Избушка” и понимать принципы её работы, включая использование источника питания, проводников, переключателей и нагрузки.

Оборудование и материалы:

1) Батарейка (источник питания) — от 1,5 В и до 9 В
2) Провода с зажимами (проводники)
3) Лампочка (нагрузка) — с соответствующим напряжением
4) Переключатель или кнопка
5) Резистор (по необходимости) — для ограничения тока
6) Монтажная плата или зажимы для соединения элементов (опционально)
7) Мультиметр — для измерения напряжения и тока (опционально)

Этапы выполнения задания:

1. Подготовка элементов цепи

2. Подготовьте все элементы цепи, убедитесь, что они исправны. Проверьте состояние батарейки, лампочки и проводов.
3. Определите полярность батарейки (плюс и минус) — это необходимо для правильного подключения цепи.
4. Подключение источника питания
5. Подключите один провод от положительного полюса батарейки (или блока питания) к одной из сторон лампочки.
6. Другой провод подключите от отрицательного полюса батарейки к переключателю или напрямую к лампочке, если не используете переключатель.
    a. Добавление переключателя (если используется)
7. Переключатель устанавливается между отрицательным полюсом батарейки и лампочкой. Это позволит включать и выключать цепь.
8. Соедините один провод от лампочки к одной клемме переключателя, а второй провод от другой клеммы переключателя к отрицательному полюсу батарейки.
    a. Замыкание цепи
9. Проверьте все соединения. Когда переключатель находится в положении "включено", цепь должна замкнуться, и лампочка загорится.
10. Если лампочка не загорается, проверьте полярность батарейки, целостность проводов и надёжность соединений.
    a. Измерение параметров цепи (опционально)
11. Используя мультиметр, измерьте напряжение на батарейке и ток, протекающий через лампочку. Это поможет понять, как меняются параметры цепи в зависимости от напряжения и сопротивления нагрузки.
    a. Добавление резистора (по необходимости)
12. Если в цепи используется мощный источник питания или чувствительная лампочка, добавьте резистор для ограничения тока. Подключите его последовательно с лампочкой.

Вопросы для обсуждения:

1. Какие элементы являются обязательными для функционирования электрической цепи?

Источник питания, проводники и нагрузка (лампочка в нашем случае).

2. Что произойдет, если в цепь добавить больше нагрузки, например, подключить вторую лампочку?

Это может изменить ток в цепи, лампочки могут светить слабее или перестать работать, в зависимости от конфигурации цепи (последовательное или параллельное подключение).

3. Какая роль переключателя в электрической цепи?

Переключатель замыкает и размыкает цепь, управляя её работой (включает и выключает ток).

Ошибки и их устранение:

Лампочка не горит: Проверьте полярность подключения батарейки и целостность проводов. Убедитесь, что переключатель замкнут, и напряжение от батарейки поступает на лампочку.

Лампочка тускло горит или горит слишком ярко: Возможно, батарейка не даёт достаточного напряжения или ток слишком большой. Измерьте напряжение на батарейке и, при необходимости, добавьте резистор для ограничения тока.

Заключение:

Сборка простой электрической цепи позволяет на практике изучить ключевые элементы электротехники. Основные компоненты — источник питания, проводники и нагрузка — должны быть правильно соединены, чтобы обеспечить протекание тока и нормальное функционирование цепи.

Лекция: "Основы механики: двигатели, редукторы, крепежные элементы".

1. Двигатели: назначение и виды

Двигатель — это устройство, преобразующее различные виды энергии (электрическую, химическую, тепловую) в механическую работу. Существует множество типов двигателей, включая электрические, внутреннего сгорания и паровые. Электродвигатели, такие как коллекторные и бесколлекторные, широко применяются в робототехнике и промышленности благодаря их высокой эффективности и точности управления.

2. Принципы работы двигателей

Электрические двигатели работают на основе взаимодействия магнитных полей. Вращательное движение создается за счет тока, проходящего через катушки (в роторе или статоре), что вызывает вращение. Преобразование энергии в механическое движение позволяет применять двигатели в различных системах автоматизации, транспортных средствах и механизмах.

3. Редукторы: роль и применение

Редуктор — это механизм, изменяющий скорость и крутящий момент, передаваемый от двигателя на конечный механизм. Он состоит из системы шестерен, которые изменяют передаточное число. Редукторы необходимы для снижения скорости вращения двигателя и увеличения крутящего момента, что важно для управления тяжелыми или требующими высокой точности системами.

4. Виды редукторов

Существуют различные виды редукторов: планетарные, цилиндрические, червячные и конические. Планетарные редукторы обеспечивают высокую точность и эффективность, а червячные — компактность и возможность самоблокировки. Выбор редуктора зависит от требований к системе, таких как мощность, размеры и необходимость регулирования скорости.

5. Крепежные элементы: назначение и типы

Крепежные элементы — это детали, обеспечивающие фиксацию и соединение компонентов механической системы. Основные виды крепежа включают болты, гайки, винты, шайбы и шпильки. Правильное применение крепежа обеспечивает надежность конструкции и безопасность работы механизмов. Материалы крепежных элементов выбираются в зависимости от условий эксплуатации (нагрузки, вибрации, коррозионные факторы).

6. Принципы выбора крепежа

При выборе крепежа важно учитывать не только его размеры, но и механические свойства материалов, из которых он изготовлен. Например, для высоконагруженных систем рекомендуется использовать болты и гайки из высокопрочной стали. Также необходимо следить за правильностью затяжки крепежных элементов, чтобы избежать разболтанности или деформации.

7. Взаимодействие компонентов: двигатели, редукторы и крепеж

В любой механической системе двигатели, редукторы и крепежные элементы работают совместно. Двигатель приводит в движение механизм через редуктор, который регулирует крутящий момент. Крепежные элементы обеспечивают жесткость и стабильность всей конструкции, предотвращая разрывы и деформации. Комплексный подход к выбору и монтажу этих компонентов гарантирует надежность и долговечность механической системы.

Практическое занятие: "Сборка механической части робота".

Цель занятия:

Научиться собирать механические соединения роботов на основе обучающего робототехнического конструктора “Избушка” и понимать принципы работы сервоприводов и электродвигателей с редуктором.

Этапы выполнения сборки:

1) Сборка крыши
2) Сборка корпуса
3) Сборка рук
4) Сборка гусениц
5) Соединение основных частей

Сборка выполняется согласно пользовательской инструкции по сборке (входит в комплект конструктора)

3. ПРОГРАММИРОВАНИЕ ДЛЯ РОБОТОТЕХНИКИ

Цель:

- Обучить основам программирования, необходимым для управления роботами.
- Показать, как писать программы для микроконтроллеров.

Содержание:

Лекция: "Основы программирования на языке С для микроконтроллеров".

С — это один из наиболее популярных и широко используемых языков программирования. Он применяется для разработки системного ПО, встраиваемых систем и других проектов, требующих высокой производительности. Эта методичка содержит основные базовые конструкции языка С, которые помогут вам начать работу с этим языком.

1. Переменные и типы данных

В С необходимо явно указывать тип данных переменной при её объявлении. Некоторые основные типы данных:

int — целые числа
float — числа с плавающей точкой
double — числа с плавающей точкой двойной точности
char — один символ
_Bool — логический тип (вместо bool в стандартном С)

Пример:

```
int a = 10;
float b = 3.14;
char c = 'A';
```

Для вывода значений переменных можно использовать функцию printf():

```
printf("Целое число: %d\n", a);
printf("Вещественное число: %f\n", b);
printf("Символ: %c\n", c);
```

2. Ввод и вывод данных

Для работы с вводом и выводом в С используются функции стандартной библиотеки, такие как printf() для вывода и scanf() для ввода.
Пример вывода:

```
printf("Привет, мир!\n");
```

Пример ввода:

```
int number;
printf("Введите число: ");
scanf("%d", &number);
printf("Вы ввели: %d\n", number);
```

Знак & перед переменной нужен для передачи адреса переменной, чтобы функция scanf() могла изменить её значение.

3. Условные операторы

В C условные операторы if, else if, и else позволяют выполнять определённые блоки кода в зависимости от условий.

Пример:

```
int x = 10;

if (x > 5) {
   printf("x больше 5\n");
} else if (x == 5) {
   printf("x равно 5\n");
} else {
   printf("x меньше 5\n");
}
```

4. Циклы

C поддерживает два основных типа циклов: for и while.

1) Цикл for

Цикл for часто используется для выполнения кода фиксированное количество раз.

```
for (int i = 0; i < 5; i++) {
   printf("%d\n", i);
}
```

2) Цикл while

Цикл while выполняется до тех пор, пока условие истинно.

```
int i = 0;
while (i < 5) {
   printf("%d\n", i);
   i++;
}
```

5. Функции

Функции в C используются для структурирования кода. Каждая программа на C должна содержать как минимум одну функцию — main(), которая является точкой входа.

Определение функции:

```
int add(int a, int b) {
   return a + b;
}

int main() {
   int result = add(5, 10);
   printf("Результат: %d\n", result);
   return 0;
}
```

Функции могут принимать аргументы и возвращать значение с помощью ключевого слова return.

6. Массивы

Массивы в C — это структуры данных, хранящие несколько значений одного типа.

Пример создания массива:

```
int numbers[5] = {1, 2, 3, 4, 5};

for (int i = 0; i < 5; i++) {
   printf("%d\n", numbers[i]);
}
```

Массивы имеют фиксированную длину, которая указывается при их объявлении.

7. Строки

В C строки представляют собой массивы символов, оканчивающиеся символом \0 (нулевой символ).

Пример:

```
char str[] = "Hello";
printf("Строка: %s\n", str);
```

Для работы со строками можно использовать функции из библиотеки string.h, например, strlen() для определения длины строки или strcpy() для копирования строк.

8. Указатели

Указатели — это переменные, хранящие адреса других переменных. С помощью указателей можно работать с памятью напрямую.

Пример:

```
int x = 10;
int *p = &x;  // p хранит адрес переменной x
printf("Значение x: %d\n", x);
printf("Адрес x: %p\n", p);
printf("Значение через указатель: %d\n", *p);
```

Звездочка (*) перед именем указателя используется для разыменования, то есть получения значения по адресу.

9. Структуры

Структуры позволяют группировать несколько переменных разных типов под одним именем.

Пример:

```
struct Person {
   char name[50];
   int age;
};
int main() {
   struct Person person1;
```

```
    // Присваиваем значения полям структуры
    strcpy(person1.name, "Иван");
    person1.age = 30;

    printf("Имя: %s\n", person1.name);
    printf("Возраст: %d\n", person1.age);

    return 0;
}
```

10. Работа с файлами

С позволяет работать с файлами с использованием функций стандартной библиотеки.

Открытие и закрытие файла:

```
FILE *file;
file = fopen("example.txt", "r");  // Открытие файла для чтения

if (file == NULL) {
    printf("Ошибка при открытии файла\n");
    return 1;
}

// Закрытие файла
fclose(file);
```

Чтение и запись в файл:

```
// Запись в файл
file = fopen("example.txt", "w");
fprintf(file, "Привет, мир!\n");
fclose(file);

// Чтение из файла
file = fopen("example.txt", "r");
char line[100];
fgets(line, 100, file);
printf("Прочитано из файла: %s", line);
fclose(file);
```

11. Препроцессор и директивы

С поддерживает использование препроцессорных директив, таких как #include, #define, и условные компиляции.

Пример:

```
#include <stdio.h>
#define PI 3.14

int main() {
    printf("Значение PI: %f\n", PI);
    return 0;
}
```

Условная компиляция:

```
#ifdef DEBUG
   printf("Режим отладки включен\n");
#endif
```

12. Циклы и управление потоком выполнения

Помимо стандартных циклов for и while, C поддерживает циклы do-while, который выполняет блок кода хотя бы один раз:

Пример:

```
int i = 0;
do {
   printf("i: %d\n", i);
   i++;
} while (i < 5);
```

Также используются операторы break и continue для управления выполнением циклов.

Заключение

Этот вводный курс охватывает базовые концепции языка C, которые помогут вам начать программирование и понять основные принципы работы с памятью, вводом-выводом, а также управления потоком выполнения. C является низкоуровневым языком, что требует понимания управления памятью, указателей и других тонкостей работы с системой, но одновременно дает широкие возможности и высокую производительность.

Практическое занятие: "Написание первой программы для Arduino Mega 2560"

Цель занятия:

Научиться создавать и загружать простую программу (скетч) для платформы Arduino Mega 2560. Ознакомиться с основными функциями работы с цифровыми пинами и созданием простого алгоритма для управления светодиодом.

Оборудование и материалы:

1) Arduino Mega 2560
2) USB-кабель для подключения Arduino к компьютеру
3) Компьютер с установленной Arduino IDE
4) Светодиод (LED)
5) Резистор (220 Ом)
6) Соединительные провода
7) Макетная плата (опционально)

Этапы выполнения задания:

1) Установка Arduino IDE

Загрузите и установите Arduino IDE с официального сайта arduino.cc.
Подключите Arduino Mega 2560 к компьютеру через USB-кабель.

2) Настройка Arduino IDE

Откройте Arduino IDE.
Перейдите в меню Инструменты → Плата → Arduino Mega 2560.

В меню Инструменты → Порт выберите COM-порт, к которому подключена ваша плата (он появится в списке после подключения устройства).

3) Подключение компонентов

Подключите длинную ножку светодиода (анод) к пину 13 Arduino Mega 2560, а короткую ножку (катод) — к GND через резистор.

Если используется макетная плата, вставьте светодиод и резистор на неё и подключите их с помощью соединительных проводов.

4) Написание первой программы (скетча)

Откройте Arduino IDE и создайте новый скетч с кодом:

```
// Программа мигает светодиодом, подключённым к пину 13

void setup() {
  // Инициализация пина 13 как выходного
  pinMode(13, OUTPUT);
}

void loop() {
  // Включение светодиода
  digitalWrite(13, HIGH);
  delay(1000); // Задержка 1 секунда

  // Выключение светодиода
  digitalWrite(13, LOW);
  delay(1000); // Задержка 1 секунда
}
```

В этом коде функция setup() выполняется один раз при старте программы и инициализирует пин 13 как выходной, а функция loop() будет выполняться циклически, включая и выключая светодиод с интервалом в 1 секунду.

5) Загрузка программы на плату

Подключите Arduino Mega 2560 к компьютеру.
Нажмите кнопку Загрузить (Upload) в верхней части интерфейса Arduino IDE. Программа скомпилируется и загрузится на плату.

После успешной загрузки вы увидите, как светодиод на пине 13 начнёт мигать с интервалом в одну секунду.

6) Модификация программы

Измените задержку в функции delay() (например, на 500 миллисекунд) и повторно загрузите программу, чтобы наблюдать изменения в частоте мигания.
Попробуйте использовать другой пин (например, пин 12) для подключения светодиода. Для этого замените pinMode(13, OUTPUT) на pinMode(12, OUTPUT) и аналогично измените вызовы digitalWrite(13, HIGH) и digitalWrite(13, LOW) на digitalWrite(12, HIGH) и digitalWrite(12, LOW) соответственно.

Вопросы для обсуждения:

Для чего используется функция pinMode()?

Она определяет, как будет использоваться конкретный пин: как вход или как выход. В данном примере пин 13 используется как выход для управления светодиодом.

Что делает функция digitalWrite()?

Эта функция устанавливает высокое (HIGH) или низкое (LOW) состояние на заданном пине, что позволяет включать или выключать светодиод.

Что произойдет, если изменить значение задержки в функции delay()?

Уменьшение или увеличение значения задержки изменяет интервал времени, через который светодиод включается и выключается, что влияет на частоту мигания.

Заключение:

Это практическое занятие позволяет понять основные принципы работы с платформой Arduino Mega 2560, освоить процесс написания простых программ для управления устройствами и научиться загружать программы на плату через Arduino IDE.

Лекция: " Основы программирования на языке Python".

Python — это высокоуровневый язык программирования с простым синтаксисом, который делает его удобным для начинающих. В данной методичке будут рассмотрены основные функции и конструкции, которые помогут вам освоить Python.

1. Переменные и типы данных

Переменные в Python используются для хранения данных. Они не требуют явного указания типа — он определяется автоматически.

Пример:

```
a = 10        # Целое число (int)
b = 3.14      # Вещественное число (float)
c = "Hello"   # Строка (str)
d = True      # Логическое значение (bool)
```

Тип переменной можно узнать с помощью функции type():

```
print(type(a))  # <class 'int'>
```

2. Ввод и вывод данных

Для вывода данных на экран используется функция print():

```
print("Привет, мир!")  # Привет, мир!
```

Для ввода данных от пользователя используется функция input():

```
name = input("Введите ваше имя: ")
print("Привет,", name)
```

3. Условные операторы

Python поддерживает конструкции условных операторов, которые позволяют выполнять код в зависимости от выполнения условия.

Пример:

```
x = 10
if x > 5:
    print("x больше 5")
```

```
elif x == 5:
   print("x равно 5")
else:
   print("x меньше 5")
```

4. Циклы

Python поддерживает два основных типа циклов: for и while.

1) Цикл for

Используется для перебора элементов последовательности (списка, строки, диапазона и т.д.).

```
for i in range(5):
   print(i)  # 0, 1, 2, 3, 4
```

2) Цикл while

Повторяет выполнение блока кода, пока условие истинно.

```
i = 0
while i < 5:
   print(i)
   i += 1  # Увеличение переменной на 1
```

5. Функции

Функции в Python используются для структурирования кода и многократного использования блоков.

Определение функции:

```
def greet(name):
   print("Привет,", name)

greet("Анна")  # Привет, Анна
```

Функции могут возвращать значения:

```
def add(a, b):
   return a + b

result = add(5, 10)
print(result)  # 15
```

6. Списки

Список — это изменяемый упорядоченный набор элементов.

Пример создания и работы со списком:

```
numbers = [1, 2, 3, 4, 5]

# Добавление элемента
numbers.append(6)

# Удаление элемента
numbers.remove(3)
```

```
# Обращение к элементу по индексу
print(numbers[0])  # 1

# Цикл по элементам списка
for num in numbers:
    print(num)
```

7. Кортежи

Кортежи похожи на списки, но они неизменяемы.

Пример:

```
coordinates = (10, 20)

# Обращение по индексу
print(coordinates[0])  # 10
```

8. Словари

Словарь — это структура данных, которая хранит пары "ключ-значение".

Пример:

```
person = {
    "name": "Иван",
    "age": 30,
    "city": "Москва"
}

# Доступ по ключу
print(person["name"])  # Иван

# Добавление нового ключа
person["job"] = "Программист"
```

9. Множества

Множества — это коллекции уникальных элементов.

Пример:

```
numbers = {1, 2, 3, 4, 5}

# Добавление элемента
numbers.add(6)

# Удаление элемента
numbers.remove(2)

# Проверка наличия элемента
if 3 in numbers:
    print("3 есть в множестве")
```

10. Обработка исключений

Для обработки ошибок в Python используются блоки try и except.

Пример:

```
try:
    x = int(input("Введите число: "))
    print(10 / x)
except ZeroDivisionError:
    print("Деление на ноль невозможно!")
except ValueError:
    print("Неверный формат ввода!")
```

11. Работа с файлами

Python позволяет работать с файлами для чтения и записи данных.

Чтение файла:

```
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

Запись в файл:

```
with open('example.txt', 'w') as file:
    file.write("Привет, мир!")
```

12. Модули

Модули позволяют использовать уже готовые функции и классы, которые находятся в других файлах.

Пример импорта стандартного модуля:

```
import math

# Использование функции из модуля
print(math.sqrt(16))  # 4.0
```

13. Списковые включения (List Comprehensions)

Списковые включения позволяют быстро создавать списки с применением выражений.

Пример:

```
squares = [x**2 for x in range(10)]
print(squares)  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

14. Лямбда-функции

Лямбда-функции — это анонимные функции, которые могут быть определены "на лету".

Пример:

```
add = lambda x, y: x + y
print(add(3, 5))  # 8
```

15. Классы и объекты

Python поддерживает объектно-ориентированное программирование. Основные элементы — это классы и объекты.

Пример определения класса:

```
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        print(f"{self.name} говорит: Гав!")

# Создание объекта класса
my_dog = Dog("Бобик")
my_dog.bark()  # Бобик говорит: Гав!
```

Заключение

Эта методичка охватывает базовые аспекты языка Python, которые позволят вам начать программировать и разрабатывать простые приложения. Python прост в изучении и предлагает мощные инструменты для решения самых разнообразных задач.

Практическое занятие: "Создание программы на Python для управления роботом с одноплатным компьютером (Одноплатный комрьютер OrangePI или аналогичный входит в комплект обучающего робототехнического конструктора “Избушка”)".

Цель занятия:

Научиться создавать и загружать простую программу для операционной системы Linux, установленной на одноплатном компьютере. Ознакомиться с основными функциями работы в операционной системе и созданием простых алгоритмов.

Весь необходимый исходный код и инструменты для написания программы управления доступны на форуме ios.ru (ios.ru/forum) в разделе “Робототехнический набор "Избушка". Инструкция”

4. СЕНСОРЫ И МОДУЛИ

Цель:

- Ознакомить с основными сенсорами и модулями, используемыми в робототехнике.
- Научить подключать и использовать сенсоры и модули.

Содержание:

- Лекция: "Обзор сенсоров: датчики расстояния, температуры, света".

Лекция: Обзор сенсоров: датчики расстояния, температуры, света

1. Введение в сенсоры

Сенсоры — это устройства, которые преобразуют физические параметры окружающей среды (температуру, свет, расстояние и др.) в электрические сигналы, которые могут быть обработаны компьютером или микроконтроллером. В современных робототехнических и автоматизированных системах сенсоры играют важную роль, обеспечивая сбор данных для принятия решений и управления устройствами.

2. Датчики расстояния

Датчики расстояния измеряют расстояние до объекта с помощью различных технологий, таких как ультразвук, инфракрасное излучение или лазерные лучи. Популярные ультразвуковые датчики, такие как **HC-SR04**, используют отражение звуковой волны для определения расстояния. Они часто применяются в роботах для навигации и обнаружения препятствий. Лазерные и инфракрасные датчики дают более точные измерения на больших расстояниях и могут использоваться для точного позиционирования.

3. Датчики температуры

Датчики температуры преобразуют температуру в электрический сигнал. Примеры таких сенсоров включают термисторы, термопары и цифровые сенсоры (например, **DS18B20**). Термисторы изменяют сопротивление в зависимости от температуры, в то время как термопары генерируют напряжение. Цифровые сенсоры температуры передают данные напрямую в микроконтроллер, что упрощает их интеграцию в системы мониторинга климата, бытовой техники или промышленных установок.

4. Датчики света

Датчики света (фоторезисторы, фотодиоды и фототранзисторы) измеряют уровень освещённости. Фоторезисторы изменяют своё сопротивление в зависимости от уровня света, а фотодиоды и фототранзисторы генерируют ток под воздействием света. Такие сенсоры применяются в системах управления освещением, в камерах для автоматической регулировки яркости или в устройствах обнаружения движения.

5. Принципы работы и применения сенсоров

Сенсоры работают на основе преобразования физических характеристик среды в электрические сигналы. Эти сигналы могут быть аналоговыми (например, изменяющееся напряжение) или цифровыми (например, передача данных в двоичном коде). В системах

автоматизации, такие сенсоры позволяют адаптировать работу оборудования в зависимости от условий окружающей среды, что повышает эффективность и безопасность систем.

6. Примеры применения сенсоров

Датчики широко используются в различных областях. Датчики расстояния применяются в автомобилях для парковочных систем и беспилотного вождения. Датчики температуры используются в системах климат-контроля и для защиты от перегрева. Датчики света можно найти в умных домах, где они управляют освещением, а также в портативных устройствах, которые регулируют яркость экранов.

7. Перспективы развития сенсорных технологий

Современные сенсоры становятся всё более точными, миниатюрными и энергоэффективными. Они играют ключевую роль в развитии интернета вещей (IoT), автономных транспортных систем и умных городов. Развитие технологий интеграции сенсоров с микроконтроллерами и системами анализа данных позволяет создавать более умные устройства и системы, которые адаптируются к изменяющимся условиям среды в режиме реального времени.

Практическое занятие: "Подключение и программирование датчиков (набор 37 датчиков и сенсоров входит в комплектацию обучающего робототехнического конструктора “Избушка”) к Arduino Mega 2560 (входит в комплектацию обучающего робототехнического конструктора “Избушка”)"

Цель занятия:

Научиться подключать и программировать различные датчики (датчик температуры, датчик расстояния и датчик света) к платформе Arduino Mega 2560. Изучить, как считывать данные с датчиков и выводить их на последовательный монитор.

Оборудование и материалы:

1) Arduino Mega 2560
2) USB-кабель для подключения Arduino к компьютеру
3) Компьютер с установленной Arduino IDE
4) Датчик температуры (например, DHT11)
5) Датчик расстояния (например, HC-SR04)
6) Датчик света (например, фоторезистор)
7) Резисторы (10 кОм для фоторезистора, 220 Ом для других)
8) Соединительные провода
9) Макетная плата

Этапы выполнения задания:

1) Установка Arduino IDE

Убедитесь, что Arduino IDE установлена на вашем компьютере. Загрузите её с arduino.cc при необходимости.

2) Подключение датчиков

Датчик температуры DHT11:

- Подключите вывод VCC к 5V на Arduino.
- Подключите вывод GND к GND на Arduino.
- Подключите вывод DATA к пину 7 на Arduino.

Датчик расстояния HC-SR04:

- Подключите вывод VCC к 5V на Arduino.
- Подключите вывод GND к GND на Arduino.
- Подключите вывод TRIG к пину 9 на Arduino.
- Подключите вывод ECHO к пину 10 на Arduino.

Датчик света (фоторезистор):

- Подключите один вывод фоторезистора к 5V на Arduino.
- Другой вывод соедините с одной стороной резистора (10 кОм), а другую сторону резистора подключите к GND.
- Соедините среднюю точку между фоторезистором и резистором с аналоговым пином A0 на Arduino.

3) Установка библиотек
   - Для работы с датчиком DHT11 необходимо установить библиотеку:
   - Откройте Arduino IDE и перейдите в Скетч → Подключить библиотеку → Установить библиотеку.
   - Найдите и установите библиотеку DHT sensor library от Adafruit.

4) Написание программы (скетча)
   Откройте Arduino IDE и введите следующий код:

```
#include <DHT.h>

// Определение пинов
#define DHTPIN 7        // Пин, к которому подключен DHT11
#define DHTTYPE DHT11    // Тип датчика DHT11

DHT dht(DHTPIN, DHTTYPE); // Создание объекта DHT

// Пины для датчика расстояния
const int trigPin = 9;
const int echoPin = 10;

void setup() {
  Serial.begin(9600);  // Инициализация последовательного порта
  dht.begin();         // Инициализация датчика DHT11

  // Инициализация пинов для датчика расстояния
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Чтение данных с датчика DHT11
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Проверка на ошибки при чтении
  if (isnan(h) || isnan(t)) {
    Serial.println("Ошибка при чтении DHT!");
```

```
    return;
  }

  // Вывод данных о температуре и влажности
  Serial.print("Температура: ");
  Serial.print(t);
  Serial.print(" °C, Влажность: ");
  Serial.print(h);
  Serial.println(" %");

  // Чтение данных с датчика расстояния
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  float distance = (duration * 0.0343) / 2; // Преобразование времени в расстояние

  // Вывод расстояния
  Serial.print("Расстояние: ");
  Serial.print(distance);
  Serial.println(" см");

  // Чтение уровня света с фоторезистора
  int lightLevel = analogRead(A0);
  Serial.print("Уровень света: ");
  Serial.println(lightLevel);

  delay(2000); // Задержка перед следующим чтением
}
```

5) Загрузка программы на плату

Подключите Arduino Mega 2560 к компьютеру.

Нажмите кнопку Загрузить (Upload) в верхней части Arduino IDE. Программа скомпилируется и загрузится на плату.

6) Проверка работы датчиков

Откройте Последовательный монитор в Arduino IDE (Ctrl + Shift + M) для просмотра выводимых данных.

Вы должны увидеть данные о температуре, влажности, расстоянии и уровне света, которые будут обновляться каждые 2 секунды.

Вопросы для обсуждения:

1) Как работает датчик температуры DHT11 и какие данные он передаёт?
   DHT11 измеряет температуру и влажность, передавая данные в цифровом формате на Arduino.
2) Каков принцип работы датчика расстояния HC-SR04?
   Датчик использует ультразвуковые волны для измерения расстояния до объекта, вычисляя время, которое требуется сигналу, чтобы вернуться после отражения от объекта.
3) Для чего используется фоторезистор, и как можно интерпретировать его показания?
4) Фоторезистор измеряет уровень освещенности. Чем выше уровень света, тем ниже его сопротивление, что отражается в значении, считываемом с аналогового пина.

Заключение:

Это практическое занятие позволяет понять основные принципы подключения и работы с различными датчиками на платформе Arduino Mega 2560. Вы научились считывать данные с датчиков и обрабатывать их в программе, что является важным шагом в создании умных систем и автоматизации.

5. РАСПОЗНАВАНИЕ И ОБРАБОТКА ДАННЫХ

Цель:

- Научить распознавать и обрабатывать данные, получаемые с сенсоров и камер.
- Обучить основам машинного обучения и нейронных сетей.

Лекция: "Основы обработки сигналов и данных".

1. Введение в обработку сигналов и данных

Обработка сигналов и данных — это область, занимающаяся анализом, преобразованием и интерпретацией различных сигналов, получаемых из окружающей среды или систем. Сигналы могут быть аналоговыми или цифровыми и могут представлять разнообразные физические параметры, такие как звук, температура, свет и другие. Обработка данных включает в себя использование алгоритмов для извлечения информации, улучшения качества сигналов и автоматизации анализа.

2. Типы сигналов

Сигналы делятся на два основных типа: аналоговые и цифровые. Аналоговые сигналы — это непрерывные величины, которые могут принимать любые значения в заданном диапазоне. Примеры включают звуковые волны и электрические напряжения. Цифровые сигналы представляют собой дискретные значения, которые могут быть представлены в виде двоичного кода. Преобразование аналоговых сигналов в цифровые (и наоборот) осуществляется с помощью аналогово-цифровых (ADC) и цифрово-аналоговых (DAC) преобразователей.

3. Основные этапы обработки сигналов

Обработка сигналов обычно включает несколько ключевых этапов:

Сбор данных: получение сигналов из различных источников, таких как сенсоры или микрофоны.

Преобразование: применение методов для преобразования сигналов в более удобный для анализа вид (например, быстрое преобразование Фурье для анализа частот).

Фильтрация: удаление шумов и ненужных составляющих сигнала для улучшения качества и точности данных.

Анализ: применение алгоритмов для извлечения полезной информации, выявления паттернов и принятий решений на основе данных.

4. Фильтрация сигналов

Фильтрация — это процесс, используемый для улучшения качества сигналов и удаления нежелательных шумов. Существует несколько типов фильтров:

Фильтры нижних частот пропускают сигналы с частотами ниже заданного значения и attenuate высокие частоты.

Фильтры верхних частот работают наоборот, пропуская высокие частоты и attenuating низкие.

Полосовые фильтры позволяют проходить только сигнал в заданном диапазоне частот. Применение фильтров широко используется в аудиообработке, радиосвязи и изображениях.

5. Применение обработки сигналов

Обработка сигналов находит применение в различных областях, включая:

Аудио и видео: улучшение качества звука и изображения, кодирование и сжатие данных.

Телcommunications: передача информации по различным каналам связи с минимальными потерями.

Медицинские технологии: анализ сигналов ЭКГ, ЭЭГ и других биомедицинских данных для диагностики и мониторинга состояния пациентов.

Автоматизация и управление: применение алгоритмов обработки сигналов в робототехнике и системах управления для улучшения взаимодействия с окружающей средой.

6. Методы анализа данных

Современные методы анализа данных включают статистические методы, машинное обучение и искусственный интеллект. Эти подходы позволяют автоматически выявлять закономерности и строить прогнозы на основе больших объемов данных. Использование этих технологий в сочетании с обработкой сигналов открывает новые возможности для разработки интеллектуальных систем, способных адаптироваться к изменениям в окружающей среде.

7. Заключение

Обработка сигналов и данных — это важная область, играющая ключевую роль в современных технологиях. С её помощью можно улучшить качество информации, повысить эффективность систем и развивать новые технологии. Понимание основ обработки сигналов и данных является необходимым для специалистов в области инженерии, информатики, медицины и других дисциплин, связанных с анализом и интерпретацией данных.

Практическое занятие: "Фильтрация данных с сенсоров (набор 37 датчиков и сенсоров входит в комплектацию обучающего робототехнического конструктора “Избушка”) на Orange Pi (входит в комплектацию обучающего робототехнического конструктора “Избушка”) с использованием Python"

Цель занятия:

Научиться подключать сенсоры к плате Orange Pi, считывать данные с них и применять методы фильтрации для улучшения качества получаемой информации.

Оборудование и материалы:

1) Плата Orange Pi
2) USB-кабель для подключения к компьютеру
3) Датчик температуры и влажности (например, DHT11 или DHT22)
4) Датчик расстояния (например, HC-SR04)
5) Провода для подключения (dupont)
6) Компьютер с установленной операционной системой (например, Armbian)
7) Python и необходимые библиотеки:
8) Adafruit_DHT для работы с датчиками DHT
9) numpy для фильтрации данных

Этапы выполнения задания:

1) Установка необходимых библиотек

Откройте терминал на вашей Orange Pi и выполните следующие команды для установки необходимых библиотек:

```
sudo apt update
sudo apt install python3-pip
pip3 install Adafruit_DHT numpy
```

2) Подключение датчиков

Датчик DHT11:

- Подключите вывод VCC к 5V на Orange Pi.
- Подключите вывод GND к GND на Orange Pi.
- Подключите вывод DATA к любому доступному GPIO пину (например, GPIO 17).

Датчик расстояния HC-SR04:

- Подключите вывод VCC к 5V на Orange Pi.
- Подключите вывод GND к GND на Orange Pi.
- Подключите вывод TRIG к GPIO 27.
- Подключите вывод ECHO к GPIO 22.

3) Написание программы (скрипта)

Откройте текстовый редактор и создайте новый файл sensor_filter.py. Введите следующий код:

```
import Adafruit_DHT
import RPi.GPIO as GPIO
import time
```

```
import numpy as np

# Константы
DHT_SENSOR = Adafruit_DHT.DHT11
DHT_PIN = 17  # GPIO pin для DHT11

TRIG_PIN = 27  # GPIO pin для TRIG
ECHO_PIN = 22  # GPIO pin для ECHO

# Настройка GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)

def get_distance():
    # Измерение расстояния с HC-SR04
    GPIO.output(TRIG_PIN, True)
    time.sleep(0.00001)  # 10 мкс
    GPIO.output(TRIG_PIN, False)

    start_time = time.time()
    stop_time = time.time()

    while GPIO.input(ECHO_PIN) == 0:
        start_time = time.time()

    while GPIO.input(ECHO_PIN) == 1:
        stop_time = time.time()

    # Расчет расстояния
 elapsed_time = stop_time - start_time
 distance = (elapsed_time * 34300) / 2  # Время * скорость звука / 2
 return distance
def get_temperature_humidity():
    # Считывание температуры и влажности
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
    return temperature, humidity

def moving_average(data, window_size):
    # Фильтрация с помощью скользящего среднего
    if len(data) < window_size:
        return None
    return np.mean(data[-window_size:])

if __name__ == "__main__":
    temperature_data = []
    humidity_data = []

    try:
        while True:
            # Получение данных от датчиков
            distance = get_distance()
            temperature, humidity = get_temperature_humidity()

            if temperature is not None and humidity is not None:
                temperature_data.append(temperature)
                humidity_data.append(humidity)
```

```
            # Применение фильтрации
            filtered_temperature = moving_average(temperature_data, 5)
            filtered_humidity = moving_average(humidity_data, 5)

            print(f"Расстояние: {distance:.2f} см")
            print(f"Температура: {temperature:.2f} °C, Влажность: {humidity:.2f} %")
            if filtered_temperature is not None:
                print(f"Отфильтрованная температура: {filtered_temperature:.2f} °C")
            if filtered_humidity is not None:
                print(f"Отфильтрованная влажность: {filtered_humidity:.2f} %")
            print("--------------------------")

        time.sleep(2)  # Задержка между измерениями

except KeyboardInterrupt:
    print("Завершение работы...")
finally:
    GPIO.cleanup()
```

4) Запуск программы

Сохраните файл и запустите его в терминале с помощью следующей команды:

Копировать код

```
python3 sensor_filter.py
```

5) Проверка работы

В терминале должны выводиться данные о расстоянии, температуре и влажности, а также отфильтрованные значения температуры и влажности на основе скользящего среднего. Если всё работает правильно, вы увидите обновления каждые 2 секунды.

Вопросы для обсуждения:

1) Как работает функция скользящего среднего и зачем она используется?
   Скользящее среднее помогает сгладить колебания в данных, обеспечивая более стабильные значения. Это особенно полезно при наличии шумов в измерениях.
2) Как можно изменить алгоритм фильтрации?
   Можно использовать другие методы фильтрации, такие как медианный фильтр или экспоненциальное сглаживание, в зависимости от конкретных требований и характеристик данных.
3) Какие еще сенсоры можно подключить к Orange Pi?
   К Orange Pi можно подключать множество других сенсоров, таких как датчики света, давления, газа и многие другие, в зависимости от задач проекта.

Заключение:

В этом практическом занятии вы научились подключать сенсоры к Orange Pi, считывать данные и применять методы фильтрации для улучшения качества получаемой информации. Это важный шаг в разработке проектов, связанных с мониторингом и автоматизацией на основе данных из сенсоров.

Лекция: Введение в машинное обучение и нейронные сети

1. Определение и значение машинного обучения

Машинное обучение (МЛ) — это подмножество искусственного интеллекта (ИИ), которое позволяет компьютерам обучаться на основе данных и делать предсказания или принимать решения без явного программирования. МЛ стало основой многих современных технологий, таких как рекомендательные системы, распознавание изображений и обработка естественного языка, что делает его важной областью исследования и применения.

2. Основные типы машинного обучения

Существует три основных типа машинного обучения:

Обучение с учителем: модель обучается на размеченных данных, где известны входные и соответствующие выходные значения. Примеры включают классификацию и регрессию.

Обучение без учителя: модель работает с неразмеченными данными, выявляя скрытые структуры или паттерны. Кластеризация и уменьшение размерности — примеры таких подходов.

Обучение с частичным обучением: сочетает элементы обоих предыдущих типов, использует как размеченные, так и неразмеченные данные для обучения, что помогает повысить эффективность модели.

3. Введение в нейронные сети

Нейронные сети (НН) — это модели машинного обучения, вдохновленные структурой и функцией человеческого мозга. Они состоят из множества связанных между собой "нейронов", организованных в слои: входной слой, один или несколько скрытых слоев и выходной слой. Нейронные сети способны обрабатывать сложные данные, такие как изображения, текст и аудио, что делает их особенно мощным инструментом для решения различных задач.

4. Архитектуры нейронных сетей

Существует множество архитектур нейронных сетей, каждая из которых предназначена для решения определённых задач:

Полносвязные сети (Fully Connected Networks): каждый нейрон в одном слое связан со всеми нейронами следующего слоя. Применяются для базовых задач классификации и регрессии.

Сверточные нейронные сети (Convolutional Neural Networks, CNN): оптимизированы для обработки изображений и видеоданных, используют свёрточные операции для извлечения признаков из данных.

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN): применяются для обработки последовательных данных, таких как текст или временные ряды, и способны запоминать информацию о предыдущих входах.

5. Обучение нейронных сетей

Процесс обучения нейронной сети включает в себя оптимизацию весов и смещений нейронов с помощью алгоритма обратного распространения ошибки. Модель проходит через множество итераций, минимизируя функцию потерь, которая измеряет разницу между предсказанными и фактическими значениями. Эффективное обучение требует большого объема данных и может занять значительное время.

6. Применение машинного обучения и нейронных сетей

Машинное обучение и нейронные сети находят широкое применение в различных областях, включая:

Здравоохранение: диагностика заболеваний, анализ медицинских изображений.

Автономные транспортные средства: распознавание объектов и принятие решений на основе данных от сенсоров.

Финансовые технологии: алгоритмическая торговля, оценка кредитных рисков.

Рекомендательные системы: персонализированные предложения товаров и услуг на основе анализа пользовательских данных.

7. Заключение

Введение в машинное обучение и нейронные сети открывает двери к пониманию и разработке интеллектуальных систем. С учетом постоянно растущего объема данных и вычислительных мощностей, эти технологии продолжают развиваться и находить новые области применения, что делает их неотъемлемой частью будущего технологий и науки.

Практическое занятие: "Создание простой программы с ИИ на основе языка программирования Python для одноплатного компьютера Orange PI (входит в комплектацию обучающего робототехнического конструктора “Избушка”)".

Для того чтобы запустить простой ИИ на **Orange Pi** с использованием языка программирования **Python**, можно воспользоваться библиотеками для машинного обучения, такими как **TensorFlow**, **scikit-learn** или **Keras**. Ниже представлена методичка, которая описывает основные шаги по установке необходимых пакетов, разработке простого ИИ и запуску на Orange Pi.

1. Настройка среды на Orange Pi

Шаг 1: Установка ОС и Python

Установите операционную систему (например, Armbian) на Orange Pi. Это можно сделать, загрузив образ с официального сайта и записав его на SD-карту.

Включите Orange Pi и выполните начальные настройки сети и пользователя.

Проверьте установленную версию Python с помощью команды:

```
python3 --version
```

Если Python не установлен, его можно установить следующей командой:

```
sudo apt update
sudo apt install python3 python3-pip
```

Шаг 2: Установка виртуального окружения

Используйте виртуальные окружения для изоляции зависимостей проекта:

```
sudo apt install python3-venv
python3 -m venv myenv
source myenv/bin/activate
```

Теперь у вас настроено виртуальное окружение.

2. Установка библиотек для машинного обучения

Orange Pi не обладает мощностью ПК, поэтому для простых проектов лучше использовать легковесные библиотеки.

Шаг 1: Установка scikit-learn

```
pip install scikit-learn
```

Если планируете использовать TensorFlow (легковесную версию для процессоров без GPU), выполните:

```
pip install tensorflow
```

3. Разработка простого ИИ

В качестве примера разработаем простую модель классификации с использованием библиотеки **scikit-learn**.

Шаг 1: Пример программы

Создадим Python-скрипт для классификации данных с использованием алгоритма логистической регрессии.

```
# Импортируем необходимые библиотеки
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Загружаем датасет "Ирисы"
iris = load_iris()
X = iris.data
y = iris.target

# Разделяем данные на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Создаем и обучаем модель логистической регрессии
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Делаем предсказание на тестовых данных
y_pred = model.predict(X_test)
```

```
# Выводим точность модели
accuracy = accuracy_score(y_test, y_pred)
print(f"Точность модели: {accuracy * 100:.2f}%")
```

Шаг 2: Запуск программы

Сохраните код в файл, например, simple_ai.py.

Убедитесь, что вы находитесь в активированном виртуальном окружении.

Запустите программу:

python3 simple_ai.py

Программа выведет точность работы модели на тестовых данных.

4. Оптимизация для Orange Pi

Orange Pi обладает ограниченными вычислительными ресурсами, поэтому некоторые оптимизации могут быть полезны:

Используйте легковесные модели (например, линейные модели, деревья решений).

Снижайте размерность данных, если это возможно (например, с помощью PCA).

Выполняйте обучение на более мощной машине (ПК) и перенесите обученную модель на Orange Pi для выполнения предсказаний.

5. Развертывание модели

После того как модель обучена, можно сохранить её и загружать для предсказаний в реальном времени:

```
import pickle

# Сохранение модели в файл
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)

# Загрузка модели из файла
with open('model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)

# Использование загруженной модели для предсказаний
y_pred_loaded = loaded_model.predict(X_test)
```

6. Пример использования на реальных данных

Для демонстрации работы модели можно подключить Orange Pi к сенсорам, или использовать внешние источники данных для получения информации, которую будет обрабатывать ИИ.

Заключение

Запуск ИИ на Orange Pi с использованием Python не требует сложных конфигураций, но нужно помнить о необходимости оптимизации модели для работы на устройстве с ограниченными ресурсами.

6. ПРОЕКТНАЯ РАБОТА

Цель:

- Закрепить полученные знания и навыки.
- Создать собственный проект на основе робототехнического конструктора.

Содержание:

- Самостоятельный обзор обучающимися возможных проектов: мобильные роботы, манипуляторы, беспилотные устройства.
- Индивидуальная работа: разработка и реализация собственного проекта.
- Презентация и обсуждение проектов.

7. ЗАКЛЮЧЕНИЕ И ДАЛЬНЕЙШЕЕ РАЗВИТИЕ

Цель:

- Подвести итоги курса.
- Показать возможности для дальнейшего обучения и развития в области робототехники и программирования.

Содержание:

- Лекция: "Современные тенденции в робототехнике".
- Обзор онлайн-курсов, литературы и сообществ по робототехнике и программированию.
- Планы на будущее и возможности для участия в соревнованиях и проектах.

Дополнительные материалы

Учебники и пособия:

- "Программирование для начинающих на Arduino".
- "Основы робототехники: теория и практика".
- "Python для робототехники: от простого к сложному".

Онлайн-ресурсы:

- Видеоуроки на YouTube по сборке и программированию роботов.
- Форумы и сообщества для обмена опытом и решения проблем.
- Онлайн-курсы и вебинары по робототехнике и программированию.

Оборудование:

- Робототехнический конструктор для обучения "Избушка".
- Набор инструментов для сборки и пайки.
- Дополнительные модули и сенсоры для расширения возможностей конструктора.

Этот учебно-методический комплекс поможет учащимся получить всестороннее образование в области робототехники и программирования, начиная с базовых понятий и заканчивая созданием собственных проектов.