

## Содержание

<b>1</b>	<b>Лекция 1. Вводная</b>	<b>3</b>
<b>2</b>	<b>Лекция 2. О том, как автоматизация пришла в наш мир</b>	<b>4</b>
2.1	Киберфизические системы . . . . .	4
2.2	Прицип цифрового кодирование . . . . .	4
2.3	NC и CNC - компьютер как инструмент управления . . . . .	4
2.4	Управляющее программное обеспечение . . . . .	4
2.5	Простейшие актуаторы и сенсоры . . . . .	5
2.6	Кодирование цифровых элекронных сигналов . . . . .	5
2.7	Кодирование логических значений уровнями напряжения . . . . .	5
2.8	Транзистор (упрощённо для цифровой логики) . . . . .	6
2.8.1	Примечание анонимуса для тех, кто не знает что такое транзистор . . . . .	6
2.9	Основные логические элементы . . . . .	6
2.10	Дополнительные логические элементы . . . . .	7
2.11	Микросхемы цифровой логики . . . . .	8
2.12	Дополнение от автора конспектов . . . . .	8
<b>3</b>	<b>Лекция 3</b>	<b>9</b>
3.1	Пример реализации 7 светодиодов и 7-ми сегментного индикатора . . . . .	9
3.1.1	Описание схемы . . . . .	9
3.1.2	Программа для управления светодиодами . . . . .	9
3.1.3	Программа для управления 7-ми сегментным индикатором . . . . .	9
3.2	Мультиплексированный индикатор. Пример реализации . . . . .	10
3.2.1	Схема . . . . .	10
3.2.2	Программа для управления . . . . .	11
3.3	Мультиплексированный индикатор с декодером . . . . .	12
3.3.1	Схема . . . . .	12
3.4	Линейный двигатель - 1D CNC . . . . .	12
3.5	Серводвигатель . . . . .	12
3.5.1	Пример . . . . .	13
3.5.2	Управление . . . . .	13
3.5.3	Управление сервомотором в Arduino . . . . .	13
3.5.4	Подключение к ардуино . . . . .	14
3.5.5	Пример кода . . . . .	14
<b>4</b>	<b>Лекция 4</b>	<b>16</b>
4.1	Шаговый двигатель (stepper) . . . . .	16
4.1.1	Принцип работы . . . . .	16
4.1.2	Стандарты шаговых двигателей . . . . .	16
4.1.3	Управление сигналами для управления шаговым двигателем . . . . .	17
4.1.4	Пример использования с помощью библиотеки на ардуино . . . . .	17
4.1.5	Пример превращения вращательного движения в поступательное . . . . .	18
4.2	G-code . . . . .	18
4.3	Интерфейсы и протоколы . . . . .	19
4.3.1	Определение и классификация . . . . .	20
4.3.2	Примеры интерфейсов . . . . .	21
4.4	Протокол MQTT . . . . .	22
4.4.1	Основные операции протокола MQTT . . . . .	22
4.4.2	Функция Callback . . . . .	23
4.4.3	Клиенты MQTT-брокера . . . . .	23

<b>5</b>	<b>Лекция 5</b>	<b>24</b>
5.1	Примеры интерфейсов в IoT с низким энергопотреблением . . . . .	24
5.2	Протоколы . . . . .	24
5.3	Протокол Zigbee . . . . .	24
5.4	Zigbee2MQTT - пример шлюза WiFi . . . . .	25
5.5	Zigbee USB stick . . . . .	25
5.6	Интерфейс Zigbee2MQTT . . . . .	26
5.7	Bluetooth и его использование . . . . .	27
5.8	ESPHome . . . . .	27
5.9	NB IoT - Narrow Band Internet of Things . . . . .	27
5.10	Более высокоуровневые протоколы . . . . .	27
5.11	Проблемы с облачными решениями . . . . .	27
5.12	Сборочное программирование в интернете вещей . . . . .	28
5.13	Типовая архитектура систем умного дома . . . . .	28
5.14	Потребительские характеристики умного дома . . . . .	28
5.15	Что нужно для эффективной работы умного дома? . . . . .	29
5.16	Home Assistant - Open Source Smart Home . . . . .	29
5.17	Основные понятия Home Assistant . . . . .	29

# 1. Лекция 1. Вводная

Знакомились друг с другом и со структурой занятий

## 2. Лекция 2. О том, как автоматизация пришла в наш мир

Субтрактивные методы обработки - убирают материал и делают изделие.

Аддитивные методы обработки - наращивают материал и делают изделие.

### 2.1. Киберфизические системы

- Либо станок или робот является киберфизической системой
- Киберфизическая система состоит из сенсоров (датчиков), контроллеров (вычислительных блоков) и актуаторов (исполнительных элементов)
- Примеры киберфизических систем - станки? роботы

### 2.2. Принцип цифрового кодирования

Что может/умеет делать вычислительный блок (компьютер, микроконтроллер и т.д)?

- Считывать закодированный поток цифровых данных от сенсоров
- Обработать полученные данные в соответствии с программной логикой и принимать решения, базируясь на полученных данных
- Передавать закодированный поток цифровых данных на исполнение актуаторам

Это совершенно универсальный принцип - другого не бывает.

Цифровые данные всегда кодируются значениями да/нет

- 0/1
- true/false
- есть отверстие/нет отверстия (перфокарта)
- есть нажатие клавиши/нет нажатия клавиши
- пиксель освещён/пиксель погашен

### 2.3. NC и CNC - компьютер как инструмент управления

- Принцип управления первых механических и электронных станков, использовавших Жаккардов принцип - управления по программе с помощью перфокарт или перфоленты - получил название Numerical Control (NC)
- Использование компьютера для управления, принятый в киберфизических системах принято называть Computer Numerical Control. (CNC)
- Разница - в наличии сенсоров, позволяющих организовать интеллектуальную обратную связь
- Прогресс в микроэлектронике (уменьшение размеров и энергопотребления при одновременном росте вычислительной мощности контроллеров) позволил реализовать весь современный технологический ландшафт

### 2.4. Управляющее программное обеспечение

- Вторым необходимым компонентом современных технологий является программное обеспечение
- Сегодня не существует высокотехнологичных устройств, не содержащих внутри себя микроконтроллеров и программного обеспечения
- Если на устройстве есть дисплей, значит, внутри него есть микроконтроллер и управляющее программное обеспечение

Всё современное программное обеспечение разделилось на несколько классов, для которых используются различные языки и среды программирования

- Инфраструктурное ПО - операционные системы, базы данных и т.д.

- Корпоративное ПО - это весь набор АСУ (ERP, CRM, и т.д.)
- WEB ПО - серверы и HTML 5.0, клиентские программы на Java и JS
- Встроенное (embedded) ПО - АСУТП, именно это программное обеспечение управляет миром

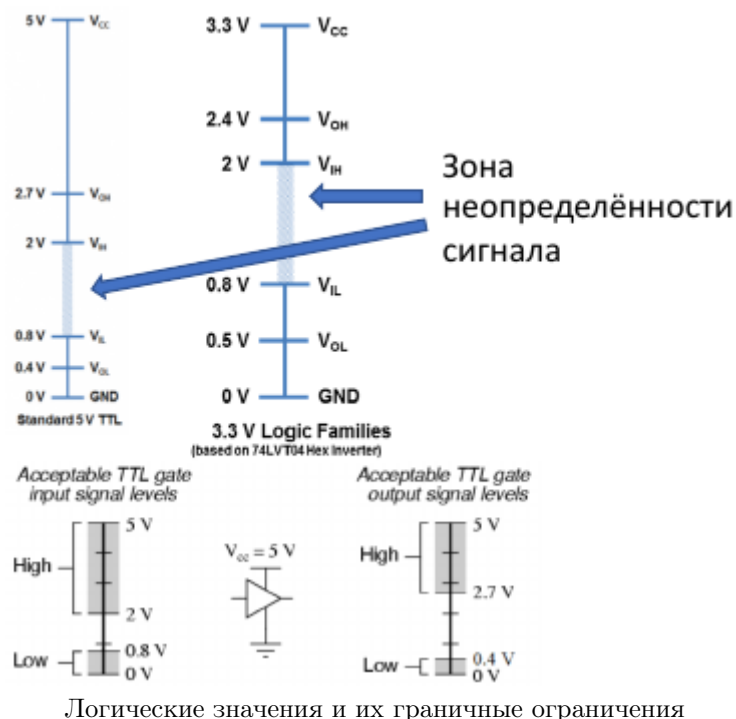
## 2.5. Простейшие актуаторы и сенсоры

- Простейшим сенсором является кнопка (клавиша)
  - есть нажатие клавиши/нет нажатия клавиши
- Простейшим актуатором является светодиод (пиксел)
  - пиксел освещён/пиксел погашен
- Логика работы киберфизической системы определяется программой микроконтроллера

## 2.6. Кодирование цифровых электронных сигналов

Пикча момент

- В электронных устройствах цифровые значения кодируются уровнями напряжений в текущий момент времени
- Существует ряд стандартов, в которых, как правило, логический 0 кодируется напряжением 0 V, а логическая единица - значением +5V для TTL-логики или +3.3 V для CMOS-логики
  - TTL 15-ти минутная лекция
  - CMOS 15-ти минутная лекция



Логические значения и их граничные ограничения

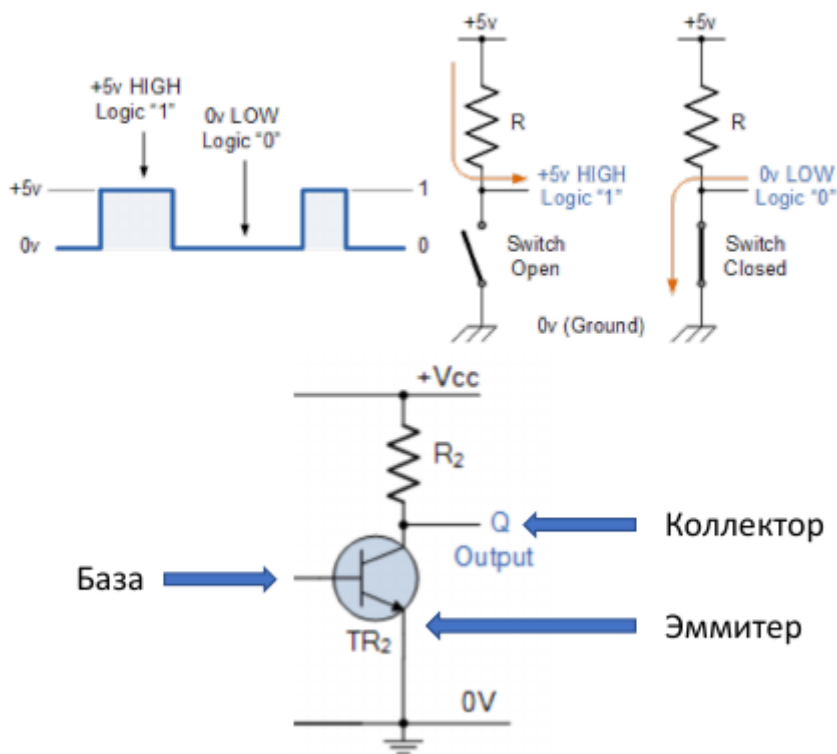
## 2.7. Кодирование логических значений уровнями напряжения

- Последовательности цифровых (двоичных) данных кодируются уровнями напряжений 0-5 V с определённой частотой дискретизации (квант времени или такт)
- Принято называть эти уровни HIGH для наличия сигнала (2-5 V) и LOW для его отсутствия (0-0.8V)
- В случае, если уровень меняется в течение такта, то сигнал может быть не определён

## 2.8. Транзистор (упрощённо для цифровой логики)

Какой-то видос на ютубе на 4 минуты И ещё с физикой и немного внутренностями

- В цифровой TTL-логике транзистор может упрощённо считаться управляемым выключателем (ключом)
- Если на базу транзистора подан сигнал HIGH, выключатель включён
- Если на базу транзистора подан сигнал LOW, выключатель выключен



Логика работы транзистора

### 2.8.1. Примечание анонимуса для тех, кто не знает что такое транзистор

Когда-то будет пикрелейтед

И вот можно если сравнить с трубой и вентиляем

- коллектор - откуда бежит вода
- база - кран
- эмиттер - куда бежит вода

В случае транзистора - вода это ток.

На коллектор и базу подаётся ток. В зависимости от того, сколько подали тока на базу - зависит то, сколько пропустим тока от коллектора к эмиттеру

## 2.9. Основные логические элементы

- Существует три основных логических элемента цифровой логики, соответствующие операциям булевой алгебры.
- С их помощью можно реализовать любую логическую схему (см. теорему Поста)
- Для удобства проектирования реализации принято выделять ещё некоторые элементы.

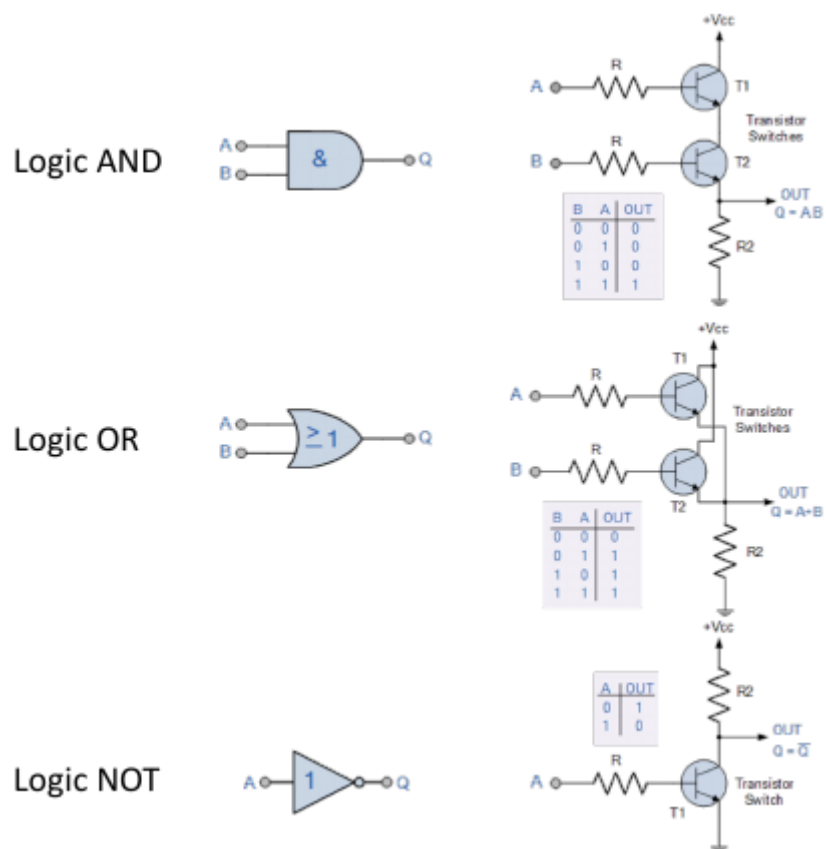


Схема основных логических элементов

## 2.10. Дополнительные логические элементы

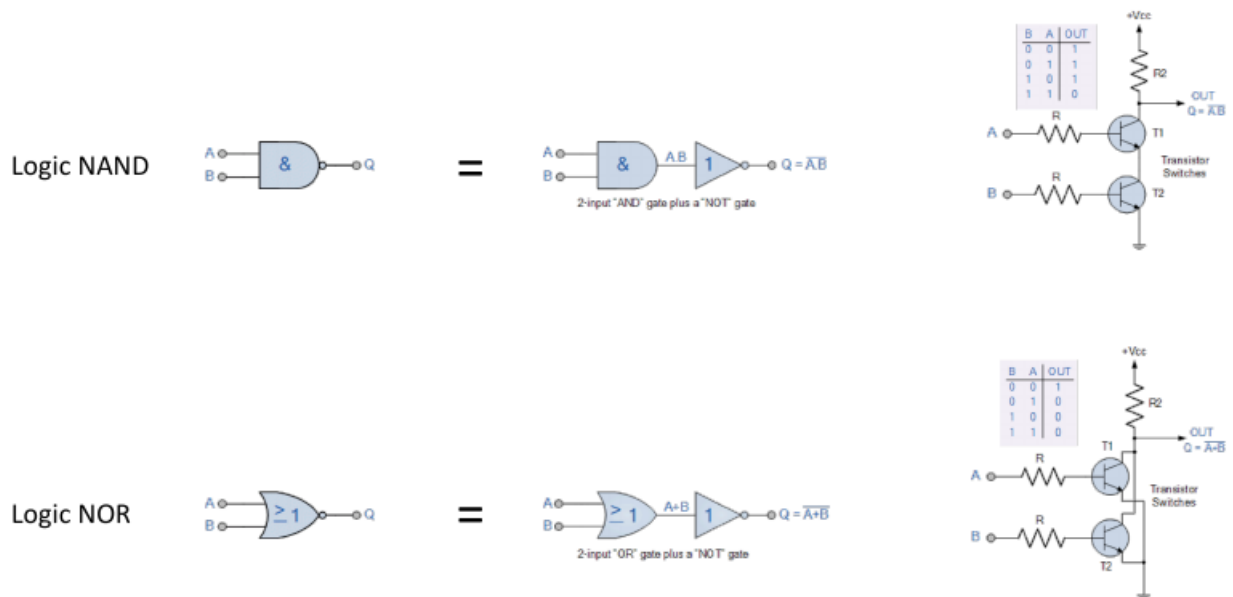
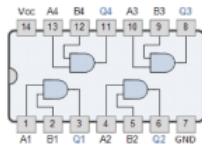


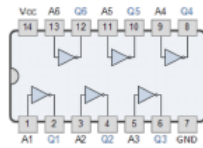
Схема дополнительных логических элементов

## 2.11. Микросхемы цифровой логики

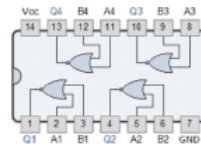
7408 Quad 2-input AND Gate



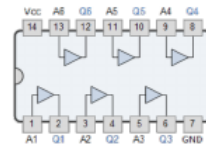
7404 NOT Gate or Inverter



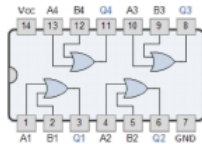
7402 Quad 2-input NOR Gate



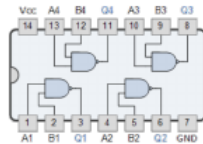
74LS07 Digital Buffer



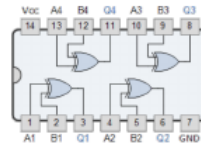
7432 Quad 2-input Logic OR Gate



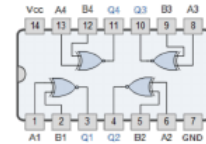
7400 Quad 2-input Logic NAND Gate



7486 Quad 2-input Exclusive-OR Gate



74266 Quad 2-input Ex-NOR Gate



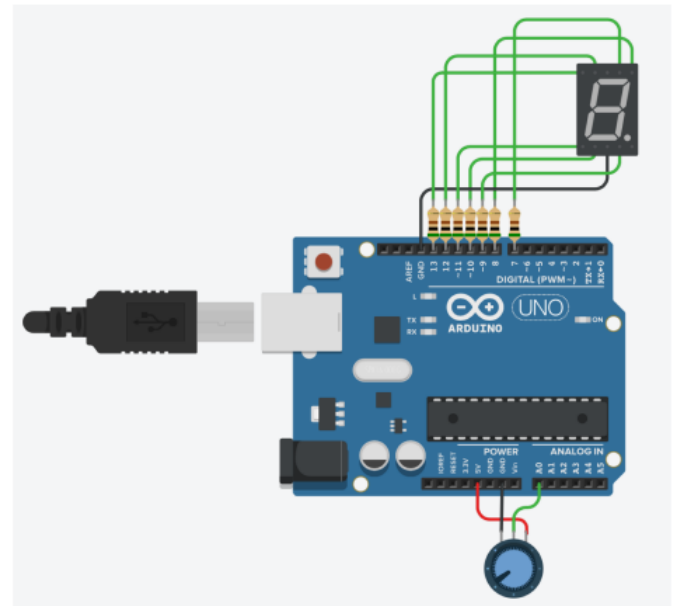
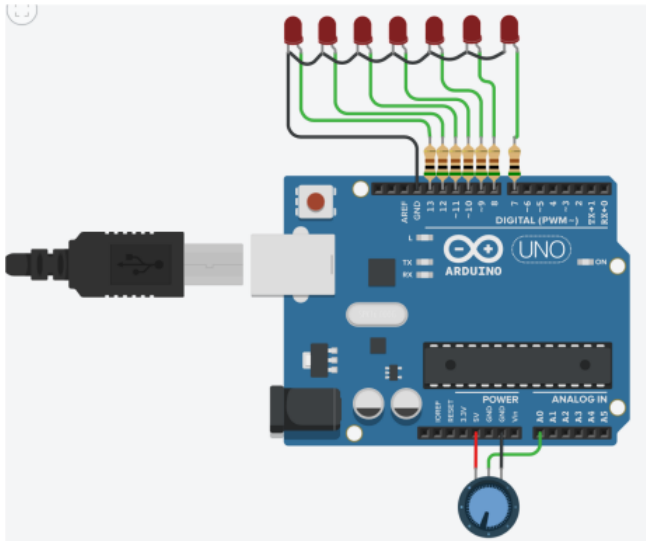
## 2.12. Дополнение от автора конспектов

Чтобы получить минимальную функцию, выраженную через  $\wedge$ ,  $\vee$ ,  $\neg$ , нужно привести выпписать матрицу значений  $\forall x \in X : f(x)$ , записать в виде КНФ или ДНФ и редуцировать



### 3. Лекция 3

#### 3.1. Пример реализации 7 светодиодов и 7-ми сегментного индикатора



##### 3.1.1. Описание схемы

Потенциометр подключается для того, чтобы управлять - какое значение сейчас отображать на светодиодах/регистре

##### 3.1.2. Программа для управления светодиодами

```
int sensorValue = 0;

void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  for (int i = 7; i < 14; ++i) {
    digitalWrite(i, LOW);
  }
}

void loop() {
  sensorValue = analogRead(A0) / 147;
  digitalWrite(sensorValue + 7, HIGH);
  delay(500);
  digitalWrite(sensorValue + 7, LOW);
}
```

##### 3.1.3. Программа для управления 7-ми сегментным индикатором

```
int sensorValue = 0;

int matrix[10][7] = {
  1, 1, 1, 1, 1, 1, 0,
```

```

0, 1, 1, 0, 0, 0, 0,
1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 1, 0, 0, 1,
0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 0, 0, 0,
1, 1, 1, 1, 0, 1, 1
};

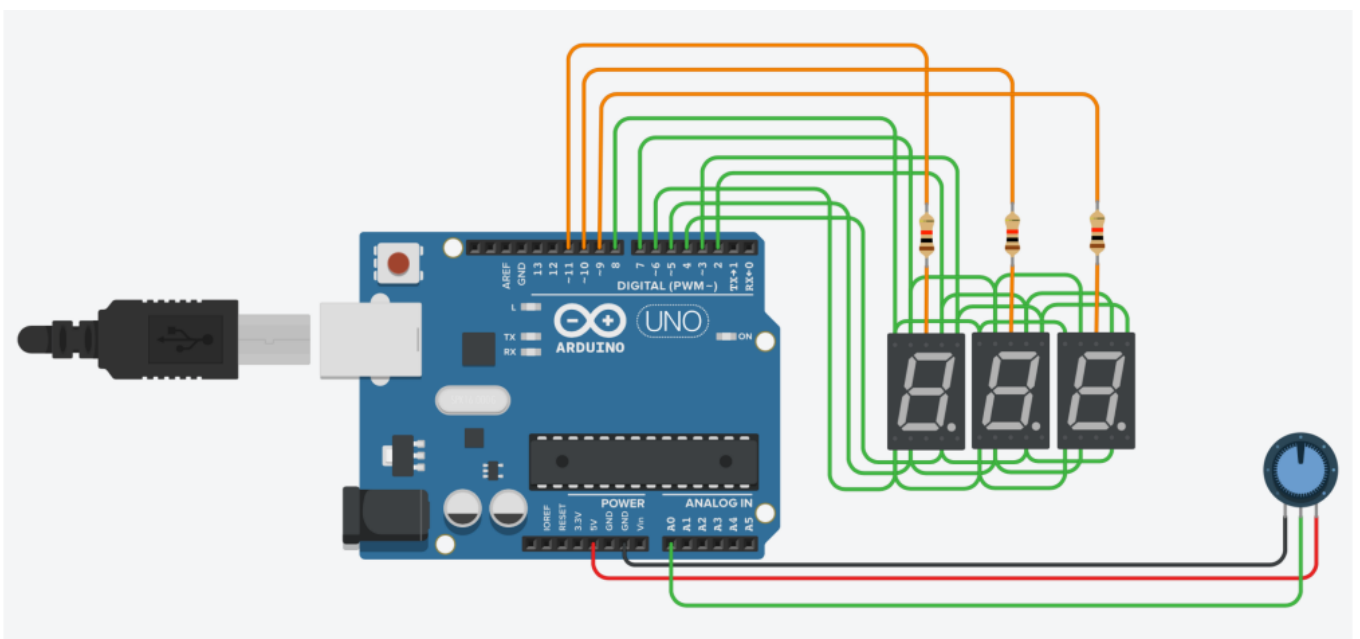
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  for (int i = 7; i < 14; ++i) {
    digitalWrite(i, LOW);
  }
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(A0) / 102.4;
  Serial.print(sensorValue);
  Serial.print("\n");
  for (int i = 7; i < 14; ++i) {
    digitalWrite(i, matrix[sensorValue][i - 7]);
    Serial.print(matrix[sensorValue][i - 7]);
    Serial.print("\n");
  }
  Serial.println();
}

```

## 3.2. Мультиплексированный индикатор. Пример реализации

### 3.2.1. Схема



### 3.2.2. Программа для управления

```
byte digitPins[] = {11, 10, 9};           // MSB to LSB
byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8}; // seg A to G

byte digits[3];

byte bits[] = {
    B00000001,
    B01001111,
    B00010010,
    B00000110,
    B01001100,
    B00100100,
    B00100000,
    B00001111,
    B00000000,
    000000100
};

int val;

void setup() {
    for (int i = 0; i < 3; ++i) {
        pinMode(digitalPins[i], OUTPUT);
        digitalWrite(digitalPins[i], LOW);
    }
    for (int i = 0; i < 7; ++i) {
        pinMode(segmentPins[i], OUTPUT);
        digitalWrite(segmentPins[i], LOW);
    }
}

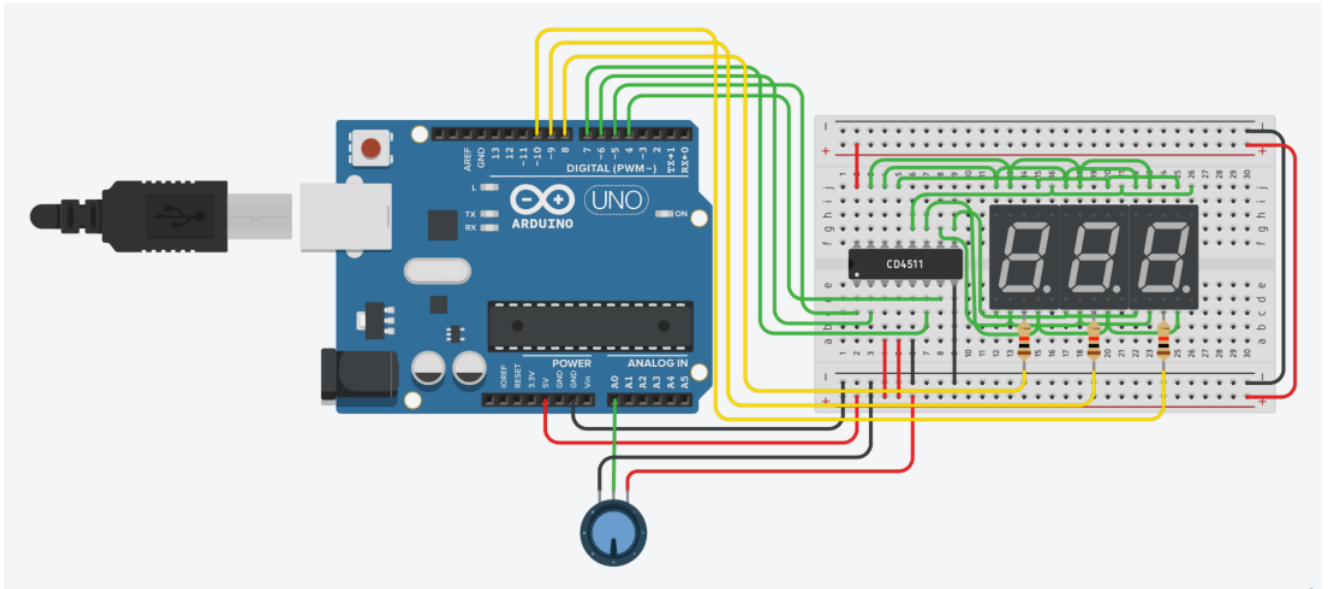
void loop() {
    int val = map(analogRead(A0), 0, 1023, 0, 999);
    digits[0] = val / 100;
    digits[1] = (val / 10) % 10;
    digits[2] = val % 10;

    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 7; ++j) {
            digitalWrite(segmentPins[j], bitRead(bits[digits[i]], 6-j));
        }
        digitalWrite(digitalPins[i], 1);
        delay(1);
        digitalWrite(digitalPins[i], 0);
    }
}
```

Тем не менее, есть специализированные решения для управления трёх индикаторов

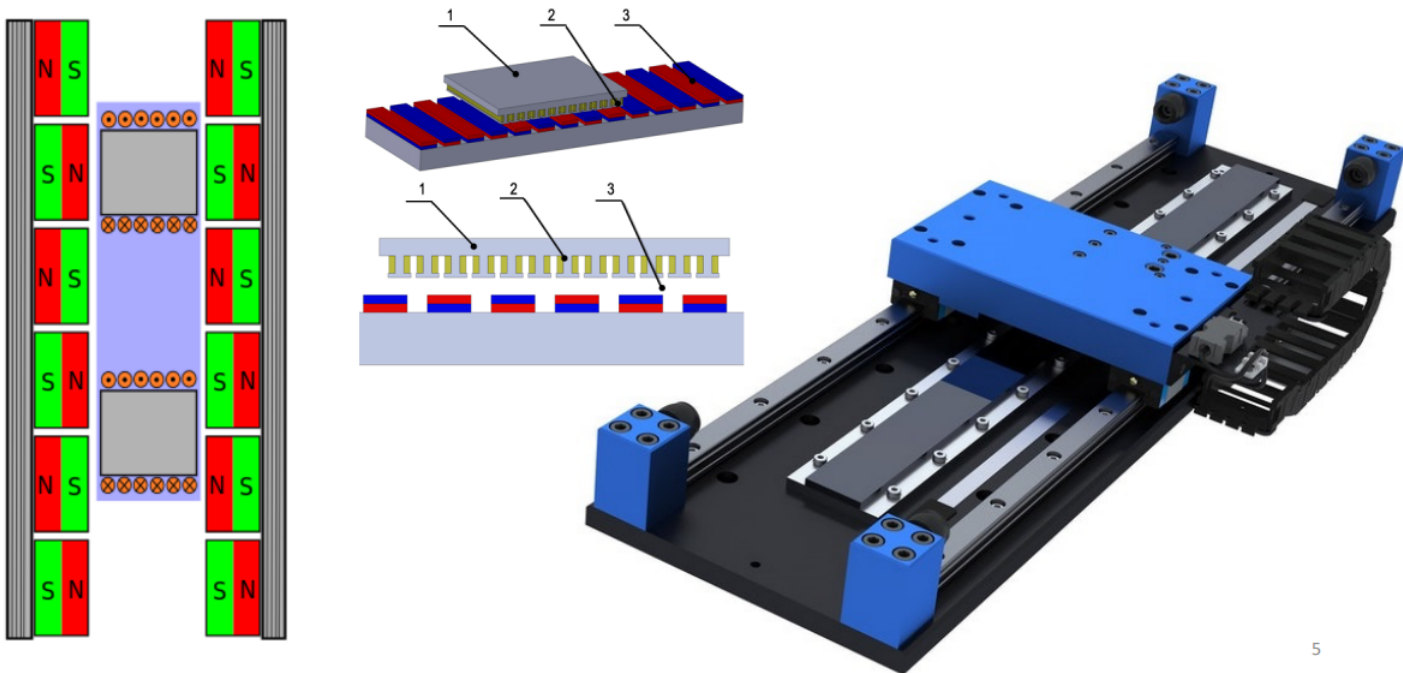
### 3.3. Мультиплексированный индикатор с декодером

#### 3.3.1. Схема



### 3.4. Линейный двигатель - 1D CNC

Один из примеров электродвигателей. Двигается за счёт магнитов и подачи тока.



5

Сложен в производстве, есть куда более простые модели двигателей, которые могут дать нужную точность.

### 3.5. Серводвигатель

Имеет не линейное, а вращательное движение.

### 3.5.1. Пример



### 3.5.2. Управление

Управление происходит с помощью импульсно-частотной модуляцией

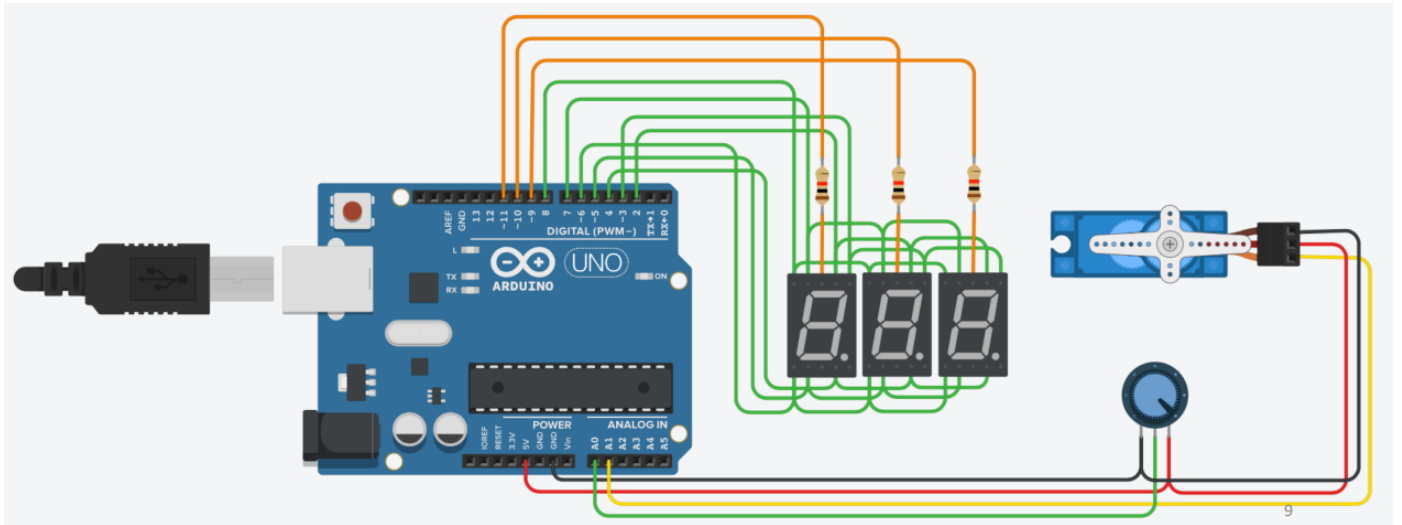
### 3.5.3. Управление сервомотором в Arduino

```
#include <Servo.h>           // include library for servo
/*
 something code
 */
Servo myServo;               // var for servo

void setup() {
  myServo.attach(A1); // attach pin to servo
  /*
   ...
  */
}

void loop() {
  double angel = getAngle(); // get angle of servo
  myServo.write(angel);      // move servo to angel 'angel';
  /*
   ...
  */
}
```

### 3.5.4. Подключение к ардуино



### 3.5.5. Пример кода

```
#include <Servo.h>

int lastVal = 0;
byte digitPins[] = {11, 10, 9};           // MSB to LSB
byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8}; // seg A to G

byte digits[3];

byte bits[] = {
    B00000001,
    B01001111,
    B00010010,
    B00000110,
    B01001100,
    B00100100,
    B00100000,
    B00001111,
    B00000000,
    000000100
};

Servo myServo;
int val;

void setup() {
    pinMode(A1, OUTPUT);
    myServo.attach(A1);
    for (int i = 0; i < 3; ++i) {
        pinMode(digitPins[i], OUTPUT);
        digitalWrite(digitPins[i], LOW);
    }
    for (int i = 0; i < 7; ++i) {
        pinMode(segmentPins[i], OUTPUT);
        digitalWrite(segmentPins[i], LOW);
    }
}

void loop() {
    int val = map(analogRead(A0), 0, 1023, 0, 179);
```

```

    if (val != lastVal) {
        lastVal = val;
        myServo.write(179 - val);
        Serial.println(val);
    }

    digits[0] = val / 100;
    digits[1] = (val / 10) % 10;
    digits[2] = val % 10;

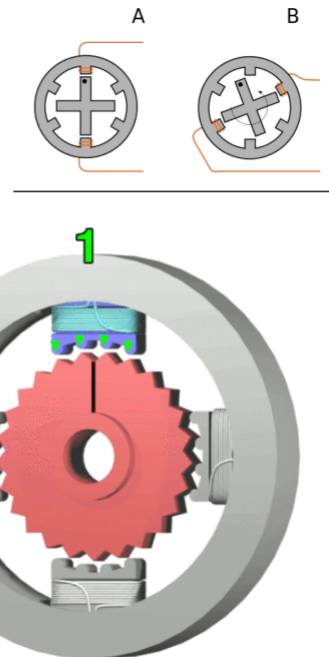
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 7; ++j) {
            digitalWrite(segmentPins[j], bitRead(bits[digits[i]], 6-j));
        }
        digitalWrite(digitalPins[i], 1);
        delay(1);
        digitalWrite(digitalPins[i], 0);
    }
}

```

## 4. Лекция 4

### 4.1. Шаговый двигатель (stepper)

#### Шаговый двигатель (stepper)



#### 4.1.1. Принцип работы

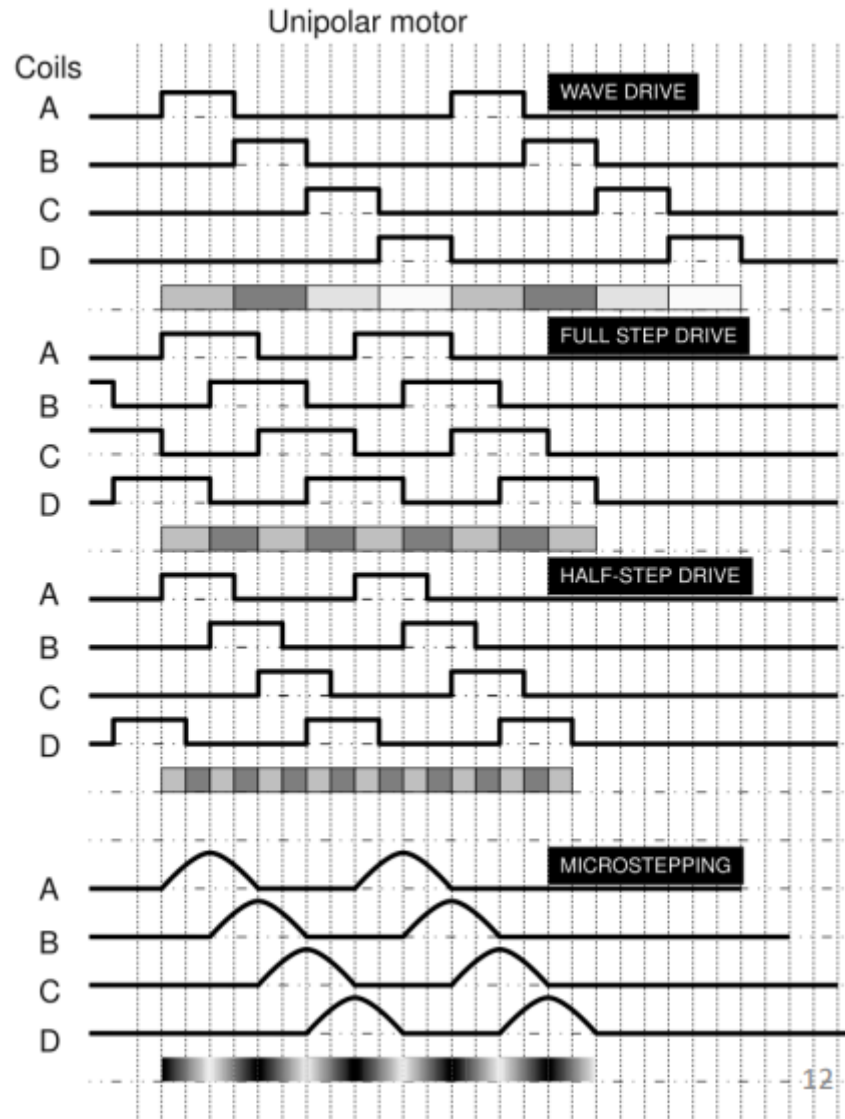
На зубчиках - магниты. Подачей и снятием тока - поворачиваем шестерёнки. Как это происходит - переход из состояния А в В на рисунке.

#### 4.1.2. Стандарты шаговых двигателей

- Шаговые двигатели стандартизированы национальной ассоциацией производителей электрооборудования (NEMA) по посадочным размерам и размеру фланца: NEMA 17, NEMA 23, NEMA 34, - размер фланца 42мм, 57мм, 86мм соответственно
- Шаговые электродвигатели NEMA 23 могут создавать крутящий момент до 30 кгс\*см, NEMA 34 до 120 кгс\*см



#### 4.1.3. Управление сигналами для управления шаговым двигателем



#### 4.1.4. Пример использования с помощью библиотеки на ардуино

```
#include <Stepper.h>

#define STEPS_PER_MOTOR_REVOLUTION 32 // steps for one full turn 360
#define STEPS_PER_OUTPUT_REVOLUTION 32*64
#define STEPS_PER_MILLIMETER 32*32

stepper smallStepper(STEPS_PER_MOTOR_REVOLUTION, 8, 10, 9, 11);

void Move(float mm) {
    long steps = STEPS_PER_MILLIMETER * mm;
    int megaSteps = steps / 16384;
    int smallSteps = steps % 16384;
    for (int i = 0; i < abs(megaSteps); ++i) {
        if (steps > 0) {
            small_stepper.step(16384);
        } else {
            small_stepper.step(-16384);
        }
    }
}
```

```

    }
    small_stepper.step(smallSteps);
}

void setup() {
    small_stepper.setSpeed(5);
    Move(+10);
    Move(-10);
}

```

#### 4.1.5. Пример превращения вращательного движения в поступательное















14

Слева на изображении - передача винт-гайка, справа - шарико-винтовая передача, которая является более точной. ШВП - шариковый подшипник линейного направления действия. В ШВП шарики в подшипники по специальному каналу при достижении одной из стенок перемещаются к противоположной. В отличие от винт-гайки соединения - практически не имеет люфтов.

## 4.2. G-code

Язык программирования перемещения в пространстве

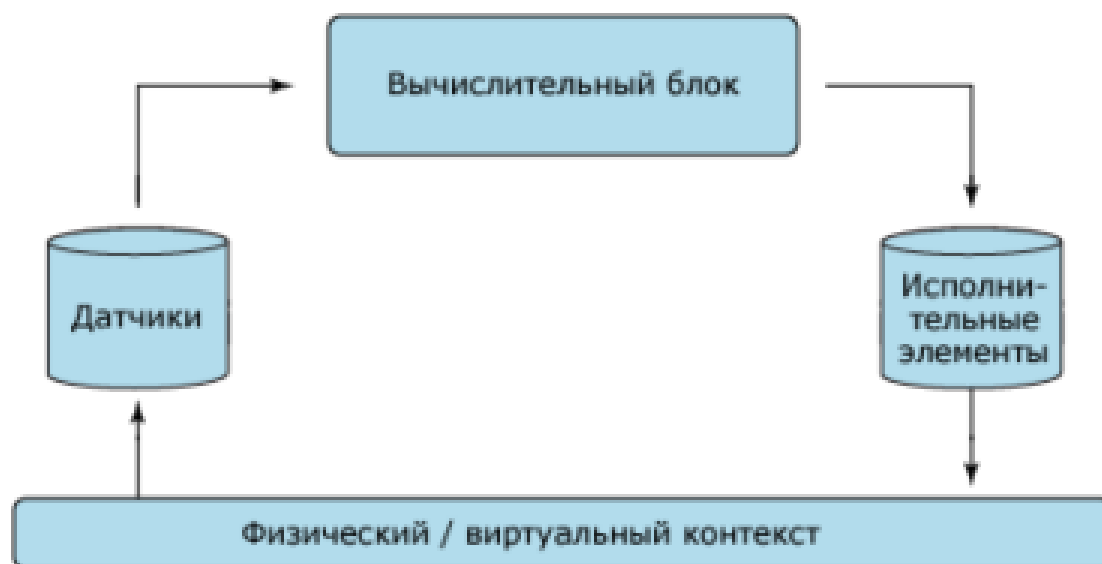
G-code	Explanation	Format	Example	Machine type
<b>G00</b> 	Go rapidly (with maximum traverse rate) to the X/Y/Z position. This code is used for positioning and not for actual machining.	G00 [X#][Y#][Z#]	G00 Z100	All machines
<b>G01</b> 	Travel in a straight line using the programmed feed rate (F). This code is used for machining.	G01 [X#][Y#][Z#][F#]	G01 X2.5 Y4.1 F200	All machines
<b>G02</b> XY-machines  Lathe 	Circular/Helical Interpolation clockwise. It causes a clockwise circular movement at the programmed feed rate (F). The motion can be 2-dimensional (flat) or 3-dimensional (helical). The default plane of the circular movement is the XY-plane (G17) but other planes can be used as well (see G17-G19). The center of the arc or circle is programmed using the I, J and K letters (R can also be used).	G02[X#][Y#][Z#] [I#][J#][K#][R#][F#]	G02 X10 Y10 I10 J0 F200	All machines
<b>G03</b> XY-machines  Lathe 	Exactly like G02 but the circular motion is going counterclockwise.	G03[X#][Y#][Z#] [I#][J#][K#][R#][F#]	G03 X10 Y10 I10 J0 F200	All machines

M-code	Explanation	Format	Example	Machine type
<b>M00</b> 	Unconditional stop (pause). The execution of the CNC program will pause. Click start or play to continue.	M00	M00	All machines
<b>M01</b> 	Optional stop (pause). Will pause the execution of the CNC program if the "Op.Stop" switch on the virtual CNC Controller is activated. Click start or play to continue.	M01	M01	All machines
<b>M02</b> 	Program end. The simulation will stop.	M02	M02	All machines
<b>M03</b> 	Start of spindle clockwise. The rotation speed is controlled by the S code letter.	M03	M03 S2500	Milling and turning machines only
<b>M04</b> 	Start of spindle counterclockwise. The rotation speed is controlled by the S code letter.	M04	M04 S2500	Milling and turning machines only
<b>M05</b> 	Spindle stop.	M05	M05	Milling and turning machines only

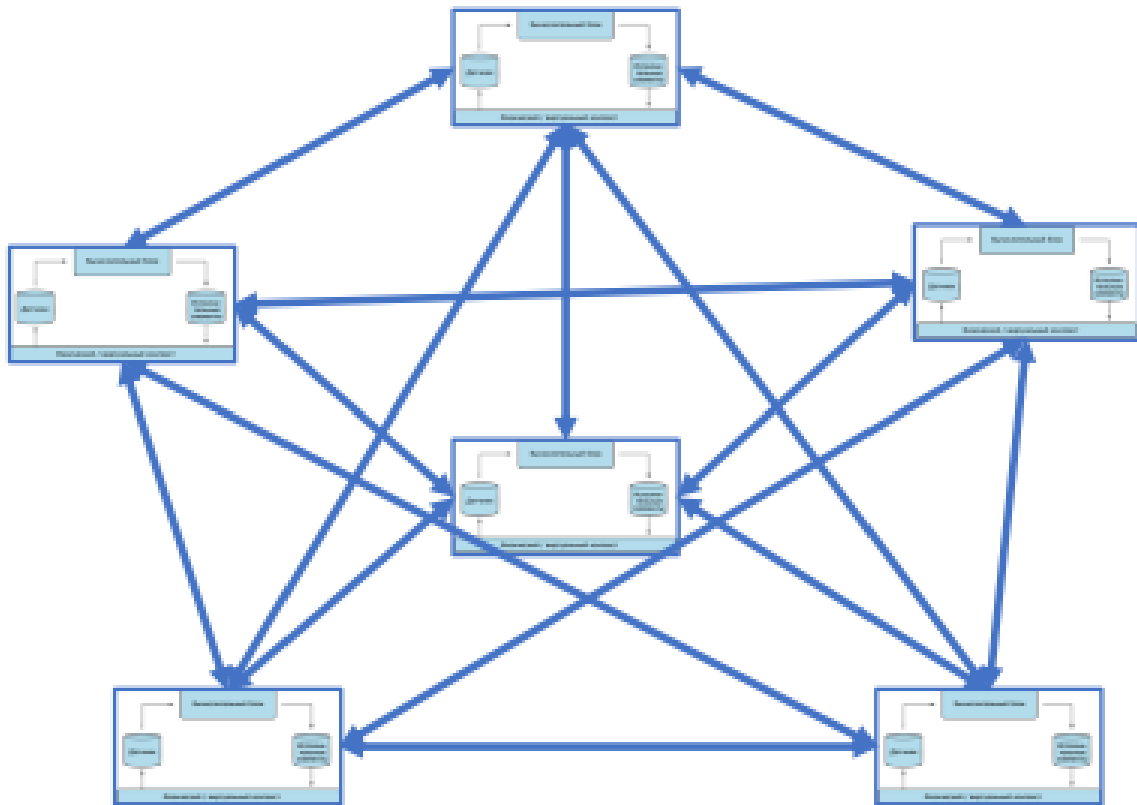
Пример g-кодов и m-кодов

### 4.3. Интерфейсы и протоколы

- До сих пор мы рассматривали киберфизические системы как одиночные объекты
- В реальных киберфизических системах, как правило, имеется множество систем, взаимодействующих между собой
- Естественно, взаимодействие организуется между контроллерами



Одна какая-то компонента системы



Множество систем общаются друг с другом

Уточнение, что стрелки, которые отображают общение систем между собой - должны идти в контроллер

#### 4.3.1. Определение и классификация

- Интерфейсом называется физический механизм передачи информации между
  - Контроллером и сенсором
  - Контроллером и актуатором
  - Двумя контроллерами
 с формализованным протоколом взаимодействия
- Например, провод, соединяющий датчик температуры с контроллером, может считаться интерфейсом, если определено каким образом по нему передаётся информация:
  - Аналоговым
  - Цифровым в соответствии с каким-то протоколом
- Существует несколько категорий интерфейсов:
  - Ближнего действия (внутри платы, например)
  - Среднего действия (внутри устройства)
  - Дальнего действия (между устройствами)
- Ортогональная классификация интерфейсов:
  - Проводные интерфейсы
  - Беспроводные интерфейсы

### 4.3.2. Примеры интерфейсов

- Интерфейс I2C
  - Ближний/средний проводной интерфейс
  - Шина I2C использует два провода для передачи информации (с общей землёй)
    - ❑ serial data (SDA)
    - ❑ serial clock (SCL)
  - Устройства на интерфейсе всегда являются Master или Slave
    - ❑ Master инициирует передачу информации и генерирует пульс (clock)
    - ❑ Slave отвечает на команды на шине
    - ❑ Каждый Slave на шине имеет уникальный адрес
    - ❑ На шине может быть несколько Master-устройств, и в этом случае необходим арбитраж
  - Передача команд ведётся в старт-стопном режиме
- Интерфейс 1Wire
  - Ближний/средний проводной интерфейс
  - Шина 1Wire использует 1 провод для передачи информации информации (с общей землёй)
  - Устройства на интерфейсе всегда являются Master или Slave
  - На шине возможен только один Master
  - Устройства подключаются к шине по схеме с открытым стоком и подтягивающим резистором
  - У каждого устройства 1Wire есть 64-разрядный идентификатор (ID)
  - Интерфейс 1Wire требует более интеллектуальных Slave-устройств, чем интерфейс
  - Для интерфейса 1Wire существует библиотека <OneWire.h>
- Интерфейс SPI (Serial Peripheral Interface) - синхронный последовательный дуплексный интерфейс
  - Интерфейс ближнего действия
  - Разработан компанией Motorola в середине 1980-х годов и стал промышленным стандартом
  - Типичные применения - работа с SD-картами и LED-дисплеями
  - Устройства на интерфейсе всегда являются Master или Slave
  - На шине возможен только один Master
  - Особенностью интерфейса является адресация устройств по выделенной линии (SS)
    - ❑ Кроме того, для обеспечения дуплекса используются 2 линии (MISO (Master Input Slave Output) и MOSI (Master Output Slave Input))
    - ❑ Третья общая линия (CS) используется для тактирования мастером
  - Таким образом, в SPI используется всегда 3+N линий, где N - количество устройств на шине
  - Обычно контроллер SPI встраивается в систему на кристалле микроконтроллера
  - В некоторых случаях применяется для внутрисхемного программирования Flash-памяти микроконтроллеров
- Интерфейс Serial (или же UART - Universal Asynchronous Receiver-Transmitter)
  - Проводной интерфейс среднего/дальнего действия
  - Исторически происходит от стандарта RS232
  - В микроконтроллерных технологиях может использовать всего два сигнальных провода (с общей землёй) - TXD и RXD
  - В оригинальном стандарте используется 9 сигнальных линий
  - Реализуется программно и аппаратно
    - ❑ Аппаратные решения - как правило, пины 0 и 1 микроконтроллера
    - ❑ Программное решение - библиотека <SoftwareSerial.h>
  - Используется как виртуальный интерфейс внутри USB-канала

- Существует промышленный интерфейс дальнего действия RS485, в котором, обычно, используется протокол Serial-интерфейса
- Serial-интерфейс требует весьма интеллектуальных устройств (буферизация и т.д.), и обычно используется между контроллерами
- Интернет-протоколы UDP и TCP/IP
  - Для работы интернет-протоколов необходим доступ контроллера к сети
    - ❑ Ethernet
    - ❑ WiFi
  - Для различных контроллеров возможны либо платы расширения, либо встроенный интерфейс WiFi
    - ❑ ESP8266 и ESP32 имеют встроенный WiFi
    - ❑ Raspberry PI имеет Ethernet-порт и встроенный WiFi начиная с версии 3
    - ❑ Существуют варианты Arduino с встроенными WiFi
  - При наличии доступа в сеть возможна работа со всеми стандартами интернета, включая HTTP, SSL/TLS и т.д.
  - Существуют специальные протоколы для интернета вещей
    - ❑ MQTT - message queuing telemetry transport

#### 4.4. Протокол MQTT

- Упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщений между устройствами по принципу издатель-подписчик
- Первая версия - 1999 год
- Спецификация MQTT 3.1.1 была стандартизирована консорциумом OASIS в 2014 году
- Возможно использование как в локальной сети, так и в интернете
- Наиболее известный брокер - Mosquitto, может работать на Raspberry PI
- Множество доступных серверов в сети - например этот сервер
- Множество инструментов для клиентских тестовых систем - например MQTT fx
- Полностью Open-Source-решение

##### 4.4.1. Основные операции протокола MQTT

- Connect - установка соединения с брокером
  - `MQTT::Connect con(CLIENT_ID);`
  - `con.set_auth(USER, PASSWORD);`
  - `con.set_will(WILLTOPIC, "Offline");`
- Subscribe - подписка на какой-то топик
  - `client.set_callback(callback);`
  - `client.subscribe(CMNDTOPIC);`
- Publish - опубликовать какое-то значение в топике
  - `client.publish(MQTT::Publish(WILLTOPIC, "Online").set_qos(1));` - напомнить, что мы живы
  - `client.publish(MQTT::Publish(STATTOPIC, "ON").set_qos(1));` - сообщить о своём состоянии
- Примеры из библиотеки `<PubSubClient.h>`

#### 4.4.2. Функция Callback

- При подписке на topic задаётся функция, которая будет вызвана, когда кто-то опубликует сообщение в этом топике, например:

```
void callback(const MQTT::Publish& pub) {  
    state = (pub.payload_string() == "on");  
    digitalWrite(relay, state);  
}
```

- Топики устроены как дерево файлов, например типичное имя топика "stat/monitor/POWER/0"
- При публикации топика указывается его значение (payload) - это всегда просто строка

#### 4.4.3. Клиенты MQTT-брокера

- Клиентами могут быть программы на:
  - Микроконтроллерах
  - Компьютерах
  - Смартфонах
  - Любых других интеллектуальных устройствах
- MQTT позволяет:
  - Без особых затрат обмениваться информацией между устройствами, входящими в киберфизический проект
  - Осуществлять синхронизацию и оркестрацию работы комплекса устройств
  - Реализовывать сложные проекты интернета вещей класса "Умный дом" или "Интеллектуальная транспортная система"

## 5. Лекция 5

### 5.1. Примеры интерфейсов в IoT с низким энергопотреблением

- Особенностью решений интернета вещей является в ряде случаев требование низкого энергопотребления
  - Особенно актуально для беспроводных устройств с батарейным питанием
  - Стандартные беспроводные протоколы (из группы WiFi) не являются энергоэффективными
- Примером такого интерфейса и группы протоколов является RF433
- Другим примером интерфейса и протокола является Bluetooth
- Однако основным стандартом для использования в решениях интернета вещей стал протокол Zigbee
- Для работы энергоэффективных протоколов в стандартных сетях требуются шлюзы
  - примером такого шлюза является Zigbee2MQTT

### 5.2. Протоколы

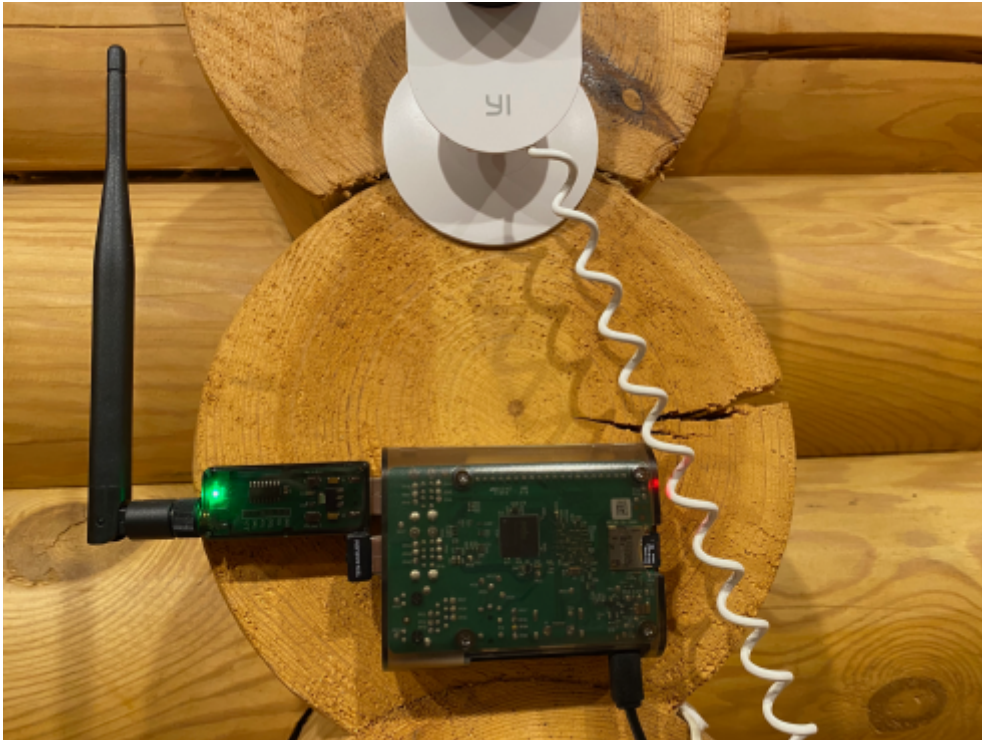
- RF433
  - Очень низко потребительный в плане энергии
  - Без обратной связи
- Bluetooth
  - Первые версии были очень сырыми
  - Сейчас этот протокол работает хорошо и используется много где
  - Одним из первых устройств использовавших bluetooth - беспроводные наушники
  - Обычно работают точка-точка: к одному устройству можно подключить только одно

### 5.3. Протокол Zigbee

- Zigbee, так же как и Bluetooth, является беспроводным протоколом для создания PAN(Personal Area Network)
  - Радиус действия не меньше 10 и не больше 100 метров
- Базируется на стандарте IEEE 802.15.4
- В отличие от Bluetooth позволяет создавать WANET(Wireless Ad hoc NETwork) - децентрализованные беспроводные сети
  - В Zigbee сетях существуют три типа устройства - единственный координатор, роутеры и конечные устройства
  - Именно роутеры умеют договариваться между собой для организации Mesh-сети
- Разработан Zigbee Alliance (Connectivity Standards Alliance) - включает более 500 компаний, специализируется в решениях для интернета вещей
- Устройства на базе Zigbee отличаются миниатюрным размером и низким энергопотреблением, позволяющим обеспечить 1-3 года эксплуатации на стандартном элементе питания 3V типа 2032
- Zigbee2MQTT - Open Source шлюз, позволяющий Zigbee-устройствам работать в MQTT-сетях, требует наличия аппаратного координатора Zigbee и интерфейса Ethernet/WiFi, может работать на Raspberry Pi



#### 5.4. Zigbee2MQTT - пример шлюза WiFi



- Шлюз реализован на Raspberry Pi 2 Model B Rev 1.1
- В этой модели Raspberry Pi нет встроенного WiFi, поэтому использован USB-модуль
- В другом слоте USB установлен Zigbee-координатор






































#### 5.5. Zigbee USB stick

- Большинство устройств Zigbee USB реализуются на чипсете CC2531
- Этот чипсет сам по себе является программируемым микроконтроллером
- Конкретная версия и конфигурация протокола задаётся прошивкой flash-памяти чипсета

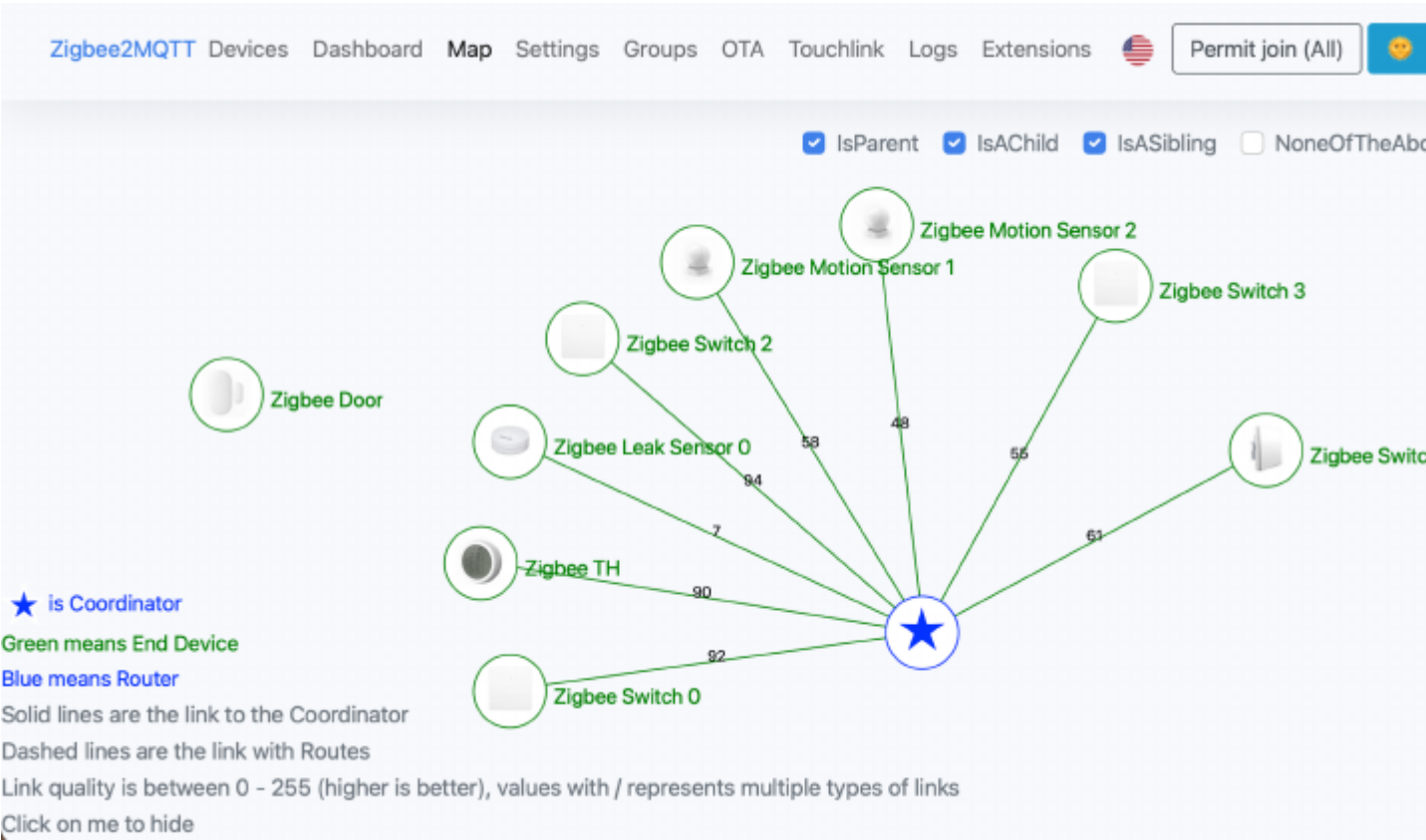


Zigbee USB stick

5.6. Интерфейс Zigbee2MQTT

Zigbee2MQTT									
Devices Dashboard Map Settings Groups OTA Touchlink Logs Extensions  <div>Permit join (All)</div> 									
3		Zigbee Switch 2	0x804b50fffe0bcaef (0xC42D)	TuYa	TS0011	93		  	
4		Zigbee Leak Sensor 0	0x00158d000292d665 (0x9C9A)	Perenio	PECLS01	6		  	
5		Zigbee Door	0xbc33acfffe4e7c76 (0x7EA9)	TuYa	TS0203	69		  	
6		Zigbee Switch 0	0xec1bbdffffe908773 (0x46C3)	TuYa	TS0011	93		  	
7		Zigbee Switch 3	0x804b50fffe0bcaed (0x3D09)	TuYa	TS0011	63		  	
8		Zigbee Switch 1	0x00158d0004515dc6 (0x1690)	Xiaomi	QBKG04LM	51		  	
9		Zigbee Motion Sensor 1	0x00124b0005ae8d31 (0x00BA)	TuYa	TS0202	60		  	

Как в интерфейсе видно устройства



Как в интерфейсе видно соединения между устройствами

## 5.7. Bluetooth и его использование

- Bluetooth имеет ограниченное применение в системах IoT с небольшим радиусом действия и моделью "точка-точка"
- 

## 5.8. ESPHome

ESPHome - семейство устройств для умного дждома на базе ESP8266 и ESP32

## 5.9. NB IoT - Narrow Band Internet of Things

- NB-IoT (Narrow Band Internet of Things) - стандарт сотовой связи для устройств телеметрии с низкими объёмами обмена данными
- Разработан консорциум 3GPP в рамках работ над стандартами сотовых сетей нового поколения
- Первая рабочая версия спецификации представлена в июне 2016 года
- Предназначен для подключения к цифровым сетям связи широкого спектра автономных устройств
- Среди достоинств NB-IoT:
  - Гибкое управление энергопотреблением устройств (вплоть до 10 лет в сети от батареи емкостью 5 Вт\*ч)
  - Огромная ёмкость сети (десятки-сотни тысяч подключённых устройств на одну базовую станцию)
  - Низкая стоимость устройств
  - Оптимизированная для улучшения чувствительности модуляции сигнала

## 5.10. Более высокоуровневые протоколы

- Иногда в киберфизических системах и решениях интернета вещей используются более высокоуровневые протоколы, характерные скорее для DevOps-решений
  - Например, AMQP (Advanced Message Queuing Protocol) с использованием RabbitMQно это скорее исключение, чем правило
- Зато в случае взаимодействия киберфизических систем с облачными сервисами, широкое распространение получили высокоуровневые протоколы на базе HTTP
  - Например, Яндекс в своих решениях для умного дома (Яндекс.Диалоги), позволяющих интегрировать IoT с Алисой, применяет взаимодействие по HTTP

## 5.11. Проблемы с облачными решениями

- Киберфизические системы и решения интернета вещей могут пользоваться облачными сервисами
  - Соответствующий инструментарий доступен
  - В ряде случаев облачные сервисы предоставляют возможности, которые трудно или невозможно обеспечить на уровне микроконтроллеровно должны обеспечивать эффективное базовое функционирование в случае недоступности облака
  - Например, для умного дома при недоступности Алисы или другого голосового помощника
  - Роботы и беспилотники не могут использовать облачные сервисы при принятии решений в силу требований реального времени
- Собственно, именно в силу этого IoT-решения должны рассматриваться как модель Edge-вычислений

### 5.12. Сборочное программирование в интернете вещей

- Сборочное программирование в интернете вещей - это **построение системы из программных и аппаратных интерфейсов**
- Такое определение является обобщением понятия сборочного программирования в традиционных вычислительных системах
- Технологии сопровождения и развития киберфизических систем в значительной мере определяют стоимость жизненного цикла таких систем
- На сегодняшний день нельзя говорить, что в этой области сформировались стандарты, в отличие от традиционной программной инженерии
- Существуют разнонаправленные тренды
  - Коммерческие поставщики IoT решений, как правило, стремятся замкнуть потребителей в своей экосистеме
  - Одновременно существует сильный тренд на интеграцию решений от различных поставщиков на Open Source платформе

### 5.13. Типовая архитектура систем умного дома

- Под термином "умный дом" обычно понимают интеграцию в единую систему управления зданием следующих систем:
  - Систему отопления, вентиляции и кондиционирования
  - Охранно-пожарную сигнализацию, систему контроля доступа в помещения, контроль протечек воды, утечек газа
  - Систему видеонаблюдения и обеспечения безопасности
  - Сети связи (в том числе телефон и локальная сеть здания), с выходом в глобальные сети
  - Систему освещения, включая автоматическое и автоматизированное управление освещением
  - Систему электропитания здания (в том числе с использованием тех или иных источников бесперебойного питания)
  - Механизацию здания (открытие/закрытие ворот, шлагбаумов и т.п.)
  - Централизованное управление аудио-, видеотехникой, домашним кинотеатром, мультимедиа
  - Телеметрия и мониторинг - удалённое слежение за системами и удалённое информирование об инцидентах в доме (квартире, офисе, объекте) и управление системами дома через сети связи
  - Удалённое управление электроприборами, приводами механизмов и всеми системами автоматизации

### 5.14. Потребительские характеристики умного дома

- Безопасность, включая физическую и инженерную безопасность
- Энергосбережение, включая пассивную энергоэффективность и активную оптимизацию использования энергоресурсов
- Комфорт проживания и использования, включая удобство централизованного управления инженерными системами и доступ к информационным ресурсам различных типов
- Мультимедиа как компонент комфортности проживания
- Возможность интеллектуального и голосового управления

### 5.15. Что нужно для эффективной работы умного дома?

- Хорошее внутридомовое покрытие беспроводной сетью
  - Обычно одной точки доступа WiFi недостаточно
  - Можно рекомендовать MESH-системы
  - Zigbee по определению является MESH-сетью, но в последнее время появляются решения и для MESH-WiFi сетей
    - \* Пример - TP-Link Deco
- Сервер управления и автоматизации
  - Именно на нём осуществляется координация систем
  - Требования - невысокие энергопотребление и постоянная работа
  - Стандартные решения на базе Intel не подходят (требуют охлаждения)
  - Как правило, сервер реализуется на Raspberry Pi, но в последнее время появляются специализированные решения

### 5.16. Home Assistant - Open Source Smart Home

- Открытые API для интеграции устройств
- Более 2000 готовых интеграций
- Гибкий конструктор интерфейсов
- Может работать как в облачном, так и в автономном режиме
- Возможность интеграции с голосовыми помощниками
- Более 8000 участников проекта (разработчиков) по всему миру
- Сотни тысяч инсталляций - и нет двух одинаковых
  - Не бывает двух одинаковых домов или квартир - значит, не может быть и двух одинаковых систем умного дома

### 5.17. Основные понятия Home Assistant

- Integrations - интеграции - настраиваемые модули связи с устройствами (Devices), могут работать с несколькими однотипными устройствами
- Devices - устройства - логические представления физических устройств, как группы сущностей (Entities)
- Helpers - вспомогательные сущности - виртуальные сущности, не имеющие физического представления
- Scripts - сценарии - простые последовательности действий
- Automations - автоматизации - сложные последовательности действий, запускаемые автоматически по наступлению тех или иных условий
- Services - сервисы - операции, выполняемые над устройствами и сущностями
- Areas - места - зоны размещения сущностей и сервисов
- Zones - зоны - зоны размещения объектов Home Assistant
- Dashboards - дашборды - интерфейсные блоки Home Assistant
- Tabs - вкладки - страницы дашбордов