Подключение SD карты памяти

Павел Пронин
C++ разработчик

Проверка связи

Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включён звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти

Поставьте в чат:

\+ если меня видно и слышно

— если нет

Павел Пронин

О спикере:

- Разработчик на C++ более 8-ми лет
- Опыт в разработке беспилотных автомобилей
- С 2022 года разработчик в компании разработки мобильных игр Playrix (компания разрабатывает такие игры как homescapes и gardegscapes)

Вспоминаем прошлое занятие

Вопрос: Какие данные передаются при обмене по протоколу NEC?

Вспоминаем прошлое занятие

Вопрос: Какие данные передаются при обмене по протоколу NEC?

Ответ: Адрес устройства и код устройства

Вспоминаем прошлое занятие

Вопрос: Какие условия срабатывания можно установить для внешнего прерывания?

Вспоминаем прошлое занятие

Вопрос: Какие условия срабатывания можно установить для внешнего прерывания?

Ответ: Срабатывания: по низкому уровню, по фронту сигнала, по спаду сигнала, по любому изменению уровня

Вспоминаем прошлое занятие

Вопрос: Какие библиотечные функции используют таймер T0?

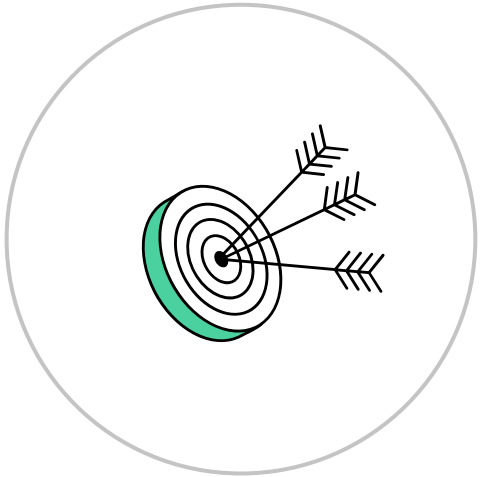
Вспоминаем прошлое занятие

Вопрос: Какие библиотечные функции используют таймер T0?

Ответ: Функции работы с временем (delay() и другие)

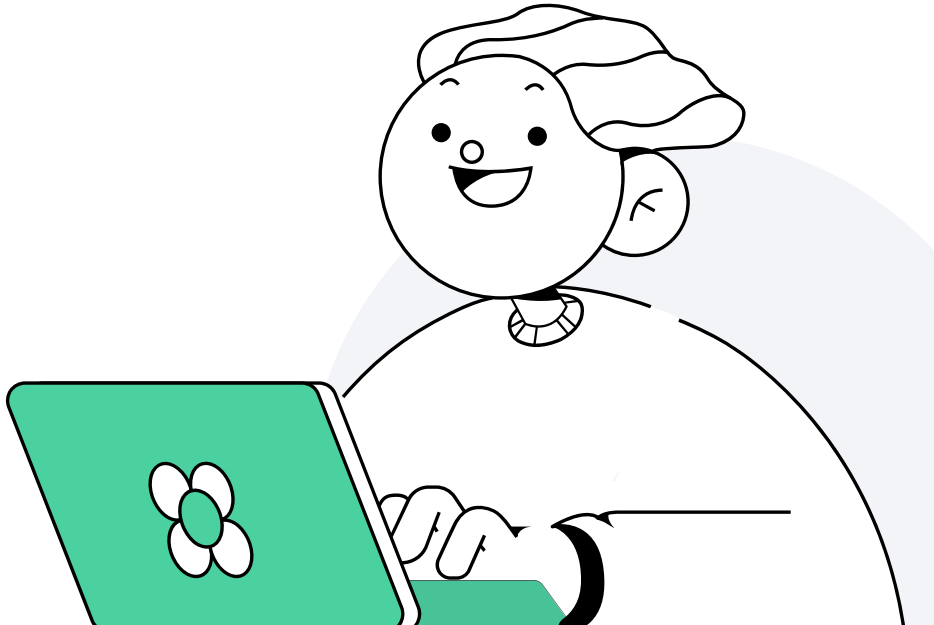
Цели занятия

- Узнаем, как устроен интерфейс SPI
- Познакомимся со сдвиговыми регистрами и научимся их использовать для управления светодиодами
- Познакомимся с SD картами памяти
- Научимся использовать специальную библиотеку для работы с SD картами памяти



План занятия

1. Как использовать интерфейс SPI
2. Как управлять сдвиговым регистром
3. Как использовать SD карту памяти
4. Итоги



Как использовать интерфейс SPI

SPI (Serial Peripheral Interface) – последовательный периферийный интерфейс, является последовательным синхронным интерфейсом передачи данных

В отличие от I2C и UART, SPI требует больше сигналов для работы, но может работать на более высоких скоростях (до 50 Мбит/с).

Параметры интерфейса SPI

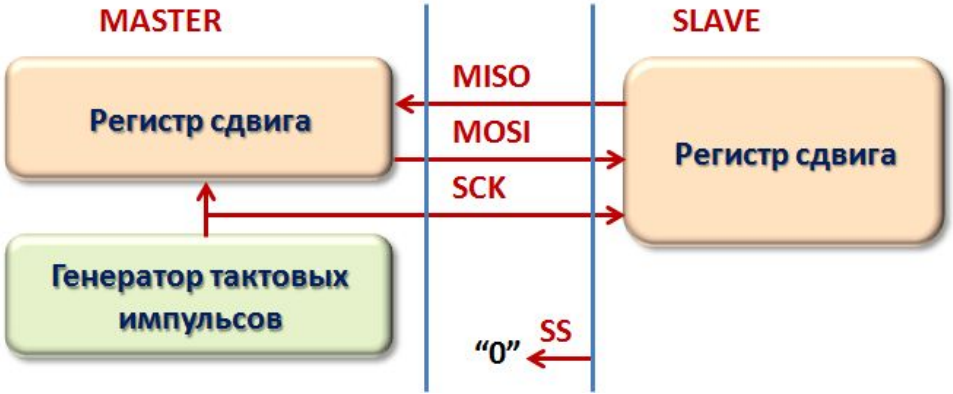
В SPI используются четыре цифровых сигнала:

- **MOSI** — выход ведущего, вход ведомого (Master Out Slave In).
  Служит для передачи данных от ведущего устройства ведомому.
- **MISO** — вход ведущего, выход ведомого (Master In Slave Out).
  Служит для передачи данных от ведомого устройства ведущему.
- **SCLK** или **SCK** — последовательный тактовый сигнал (Serial Clock).
  Служит для передачи тактового сигнала для ведомых устройств.
- **CS** или **SS** — выбор микросхемы, выбор ведомого (Chip Select, Slave Select).

Источник

Принцип работы интерфейса SPI

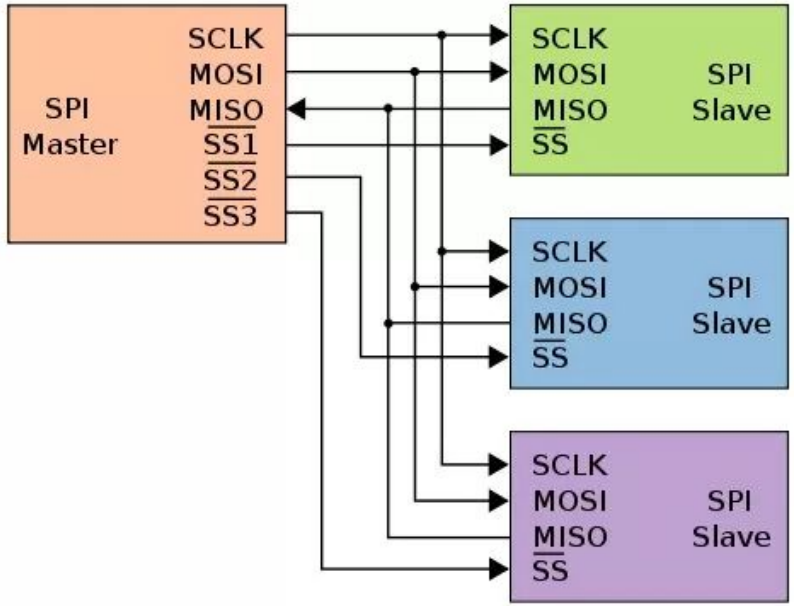
При соединении ведущего и ведомого устройства они формируют единый кольцевой сдвиговый регистр. Ведущее устройство формирует несколько тактовых импульсов (обычно 8), после которых происходит обмен данными между регистрами



Источник

Независимое подключение нескольких ведомых устройств

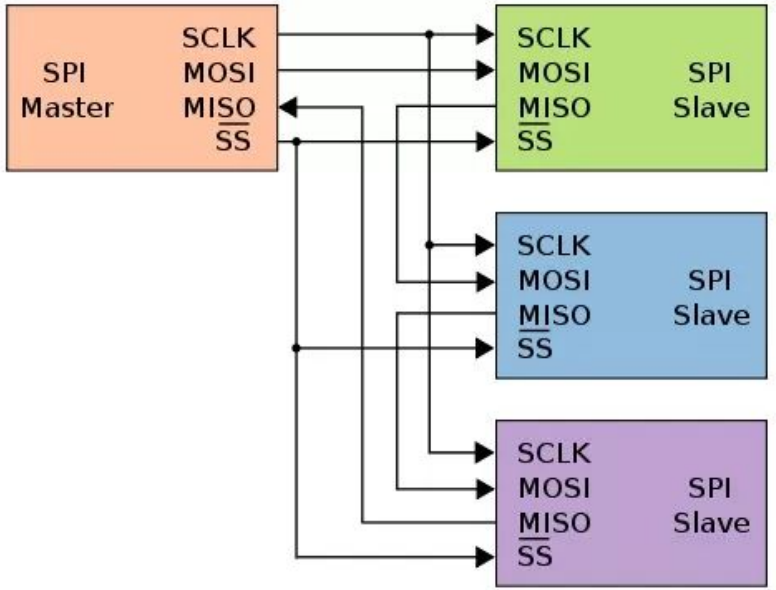
Все сигналы, кроме выбора микросхем, соединены параллельно, а ведущее устройтсво переводом того или иного сигнала SS в низкое состояние задает, с каким подчиненным устройствомон будет обмениваться данными



Источник

Каскадное подключение нескольких ведомых устройств

Нескольких микросхем образуется один большой сдвиговый регистр. Для этого выход передачи данных одной ИС соединяется со входом приема данных другой и т.д. Использование каскадного подключения возможно только в том случае, если оно поддерживается ведомым устройством.

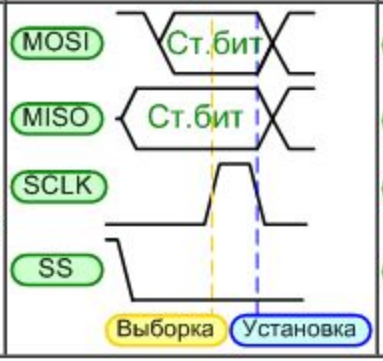
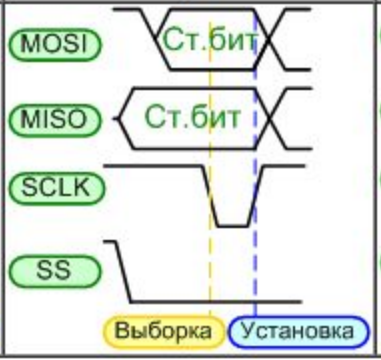
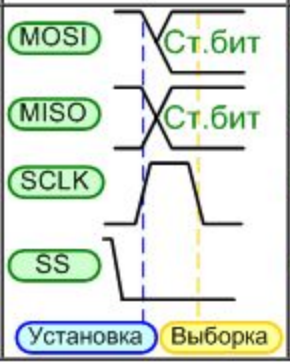
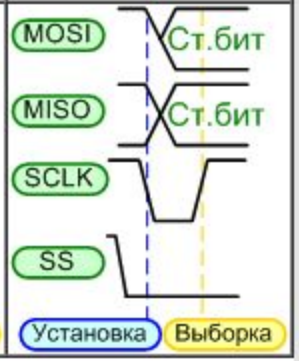


Источник

Режимы работы SPI

Режим работы задается двумя параметрами:

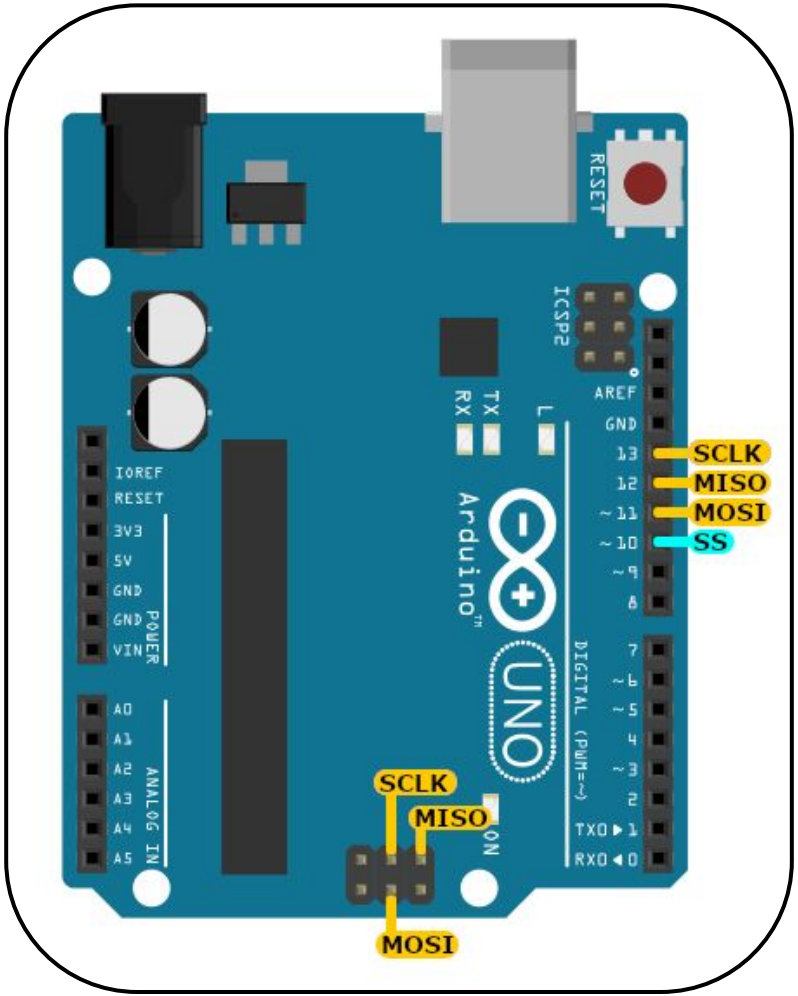
- CPOL - исходный уровень сигнала синхронизации: низкий = 0 или высокий = 1
- CPHA - фаза синхронизации: в какой момент импульса будет выполнена установка и считывание данных. По переднему фронту считывание, по заднему установка = 0, наоборот = 1

Режим SPI	0	1	2	3
CPOL	0	1	0	1
CPHA	0	0	1	1
Временная диаграмма первого цикла синхронизации				

[Источник]

Расположение на плате Arduino UNO

Линии SPI жестко привязаны к выводам платы Arduino UNO: 10 - 13.
Также они дублируются на отдельном разъеме.



Источник

Библиотека SPI для работы с интерфейсом SPI

При подключении библиотеки автоматически вызывается конструктор, который создает объект с именем SPI. Поэтому в пользовательском коде вызывать конструктор не нужно.

void begin() — инициализирует объект SPI и подключается к шине SPI как ведущее устройство

Параметры: нет
Возвращаемое значение: нет

void end() — отключает шину SPI
Параметры: нет
Возвращаемое значение: нет

Библиотека SPI для работы с интерфейсом SPI

void setBitOrder(uint8_t bitOrder) — устанавливает порядок вывода данных в/из шины SPI
Параметры:

- bitOrder: LSBFIRST - наименьший разряд первый, MSBFIRST - старший разряд первый

Возвращаемое значение: нет

void setClockDivider(uint8_t clockDiv) — устанавливает делитель частоты синхронизации SPI относительно частоты микроконтроллера
Параметры:

- clockDiv: значение делителя: 2, 4, 8, 16, 32, 64 или 128. Значение по умолчанию равно 4

Возвращаемое значение: нет

Библиотека SPI для работы с интерфейсом SPI

void setDataMode(uint8_t dataMode) — устанавливает режим работы шины SPI
Параметры:

- dataMode: SPI_MODE0 (CPOL = 0, CPHA = 0); SPI_MODE1 (CPOL = 0, CPHA = 1); SPI_MODE2 (CPOL = 1, CPHA = 0) SPI_MODE3 (CPOL = 1, CPHA = 1)

Возвращаемое значение: нет

uint8_t transfer(uint8_t data) — передает один байт по шине SPI, используется как для прием, так и для передачи данных
Параметры:

- data: байт для передачи через SPI

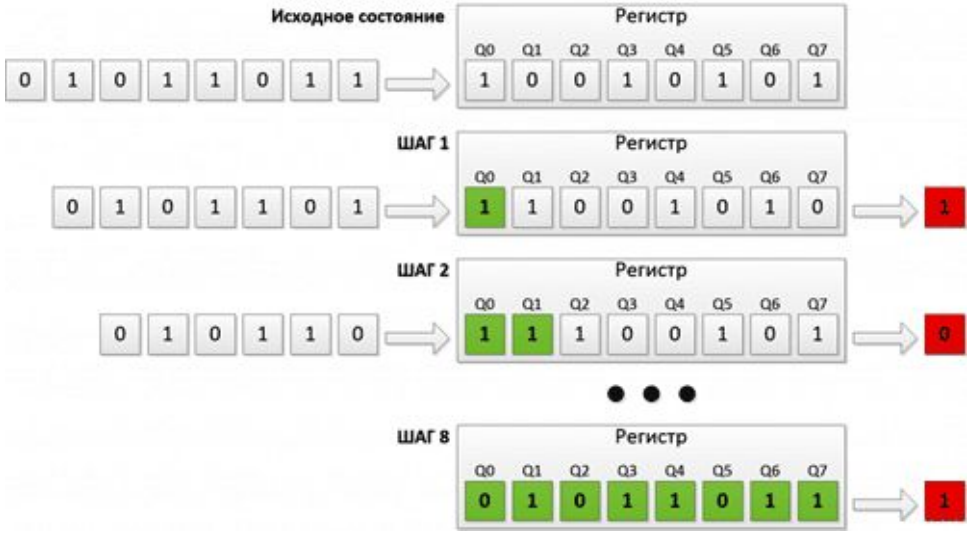
Возвращаемое значение: принятый байт

Как управлять сдвиговым регистром

2

Что такое сдвиговый регистр

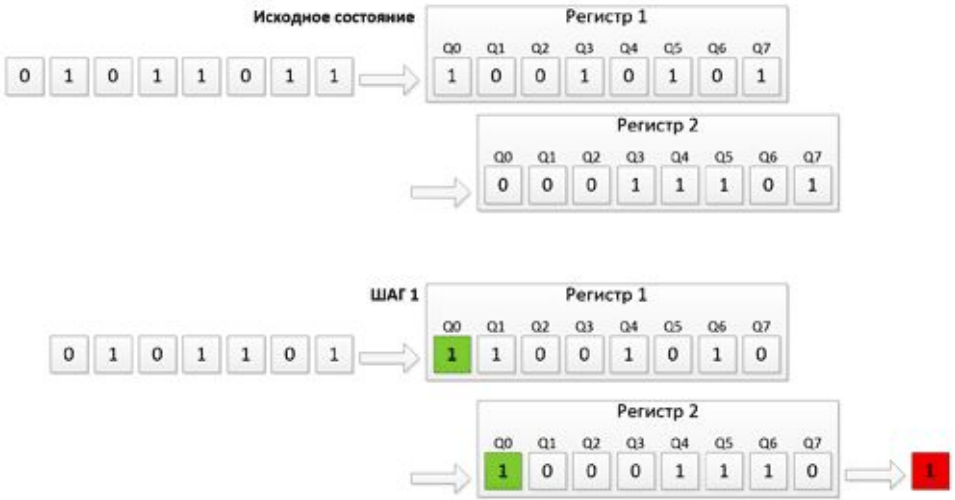
Регистр называется сдвиговым, потому что при добавлении каждого нового бита в него, мы сдвигаем все остальные в сторону.



Источник

Каскадирование сдвиговых регистров

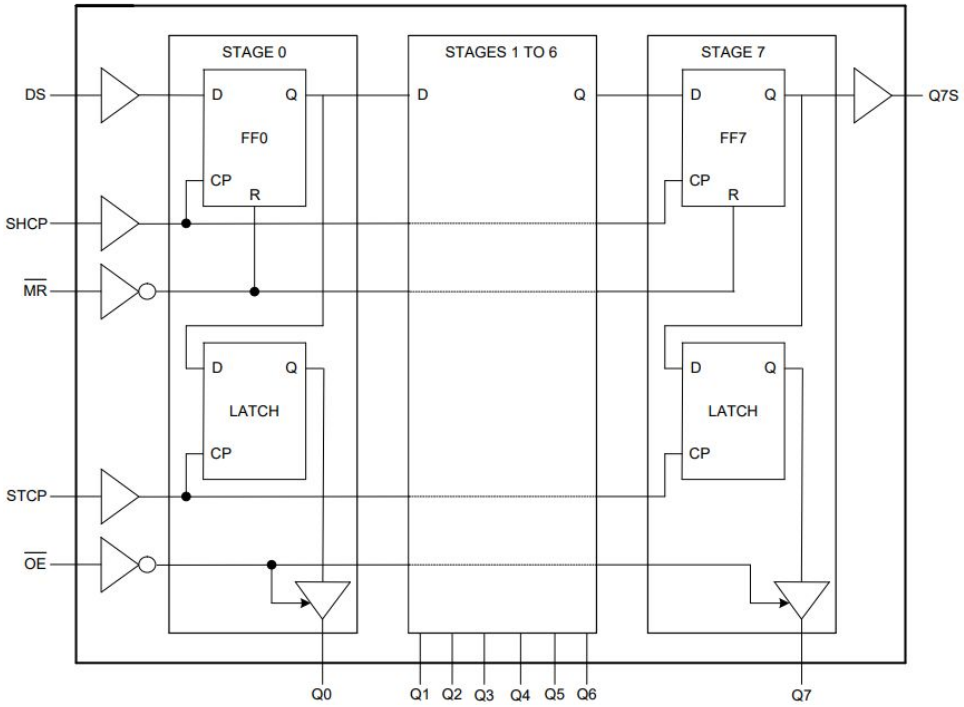
Регистры можно соединять в цепочку. В этом случае вытесненный бит не будет пропадает, а отправляется в начало следующего регистра. При этом увеличивается число доступных выводов.



Источник

Сдвиговый регистр 74HC595

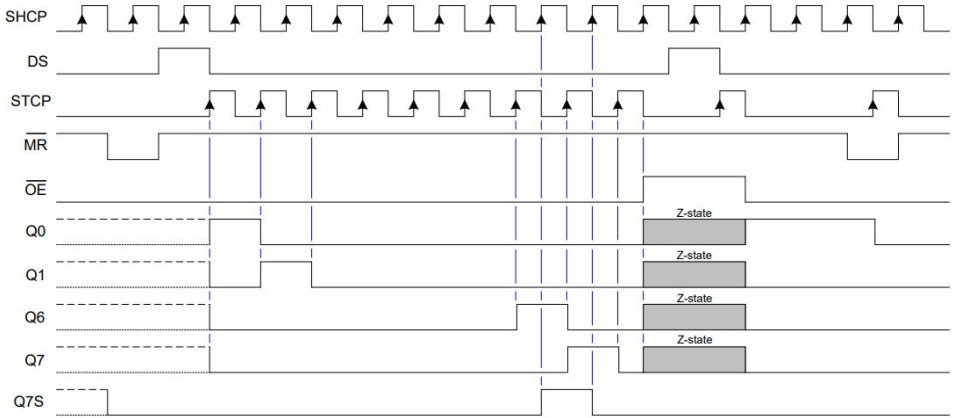
Одним из самых популярных является восьмиразрядный сдвиговый регистр 74HC595 (отечественный аналог КР1564ИР52)



Источник

Временная диаграмма работы 74HC595

Для управления сдвиговым регистром нужно формировать сигналы в соответствии с временной диаграммой



Источник

Управление светодиодами с помощью сдвигового регистра

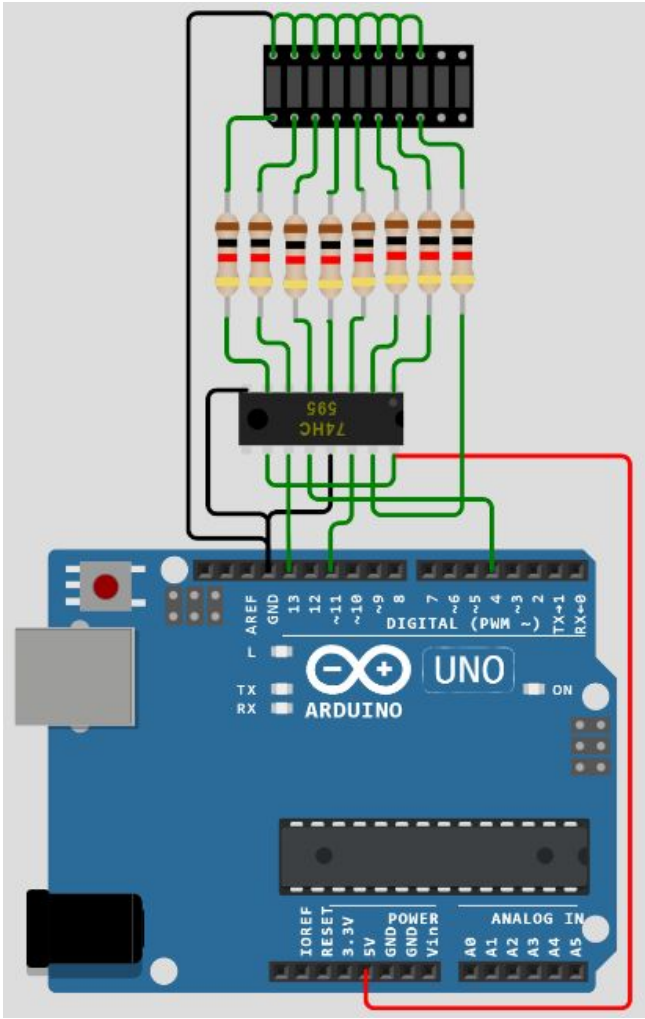
```
#include <SPI.h>

const int STCPPin = 4; // пин, управляющий выходной защёлкой

void setup()
{
  SPI.begin();
  pinMode(STCPPin, OUTPUT);
  digitalWrite(STCPPin, LOW);
  SPI.transfer(0); //сброс всех выходов в ноль
  digitalWrite(STCPPin, HIGH); //формирование фронта на выходную защелку
  digitalWrite(STCPPin, LOW);
}
// Кольцевой сдвиг
void rotateLeft(uint8_t &bits)
{
  uint8_t high_bit = bits & (1 << 7) ? 1 : 0;
  bits = (bits << 1) | high_bit;
}
```

Управление светодиодами с помощью сдвигового регистра

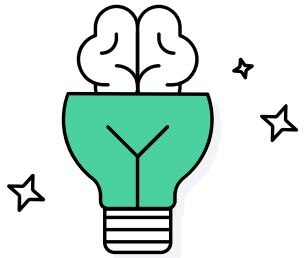
```
void loop()
{
  static uint8_t nomad = 1; //активен только один светодиод
  SPI.transfer(nomad);
  digitalWrite(STCPPin, HIGH); //формирование фронта на выходную защелку
  digitalWrite(STCPPin, LOW);
  rotateLeft(nomad); //сдвиг бита на один разрд
  delay(500);
}
```

Управление светодиодами с помощью сдвигового регистра

Используется только 8 сегментов индикатора



Практическое задание №1



Практика: управление светодиодами с помощью сдвигового регистра

Задание:

1) соберите схему в симуляторе WOKWI, подключив выводы DS и SHCP сдвигового регистра к выводам MOSI и SCLK платы Arduino UNO соответственно;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

Время выполнения: 5 минут

Ваши вопросы?

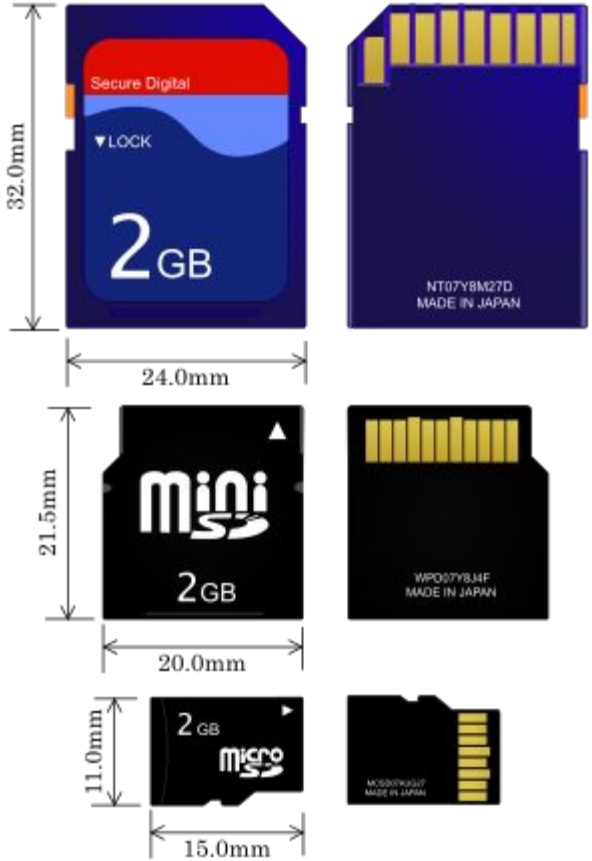
Как использовать SD карту памяти

3

Понятие SD карты

Secure Digital Memory Card (SD) — формат карт памяти (флеш-память), разработанный SD Association (SDA) для использования в портативных устройствах. Существует пять поколений карт памяти данного формата, различающиеся возможным объемом данных (совместимы сверху вниз):

- SD 1.0 — от 8 МБ до 2 ГБ;
- SD 1.1 — до 4 ГБ;
- SDHC — до 32 ГБ;
- SDXC — до 2 ТБ;
- SDUC — до 128 ТБ.

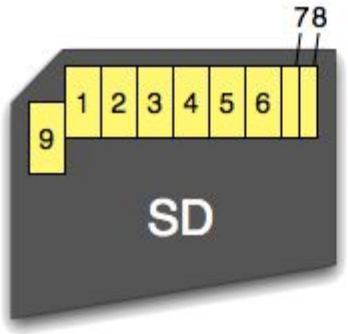


Источник

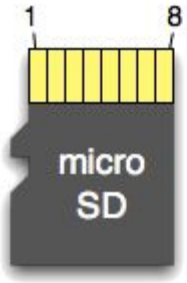
Распиновка SD карты

Карты могут поддерживать различные сочетания следующих типов шин и режимов передачи:

- режим шины SPI, подключение по требованиям интерфейса SPI;
- однобитовый режим шины SD: отдельная шина для команды и каналов передачи данных;
- четырёхбитовый режим шины SD: использует дополнительные контакты, переназначены некоторые контакты.



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X



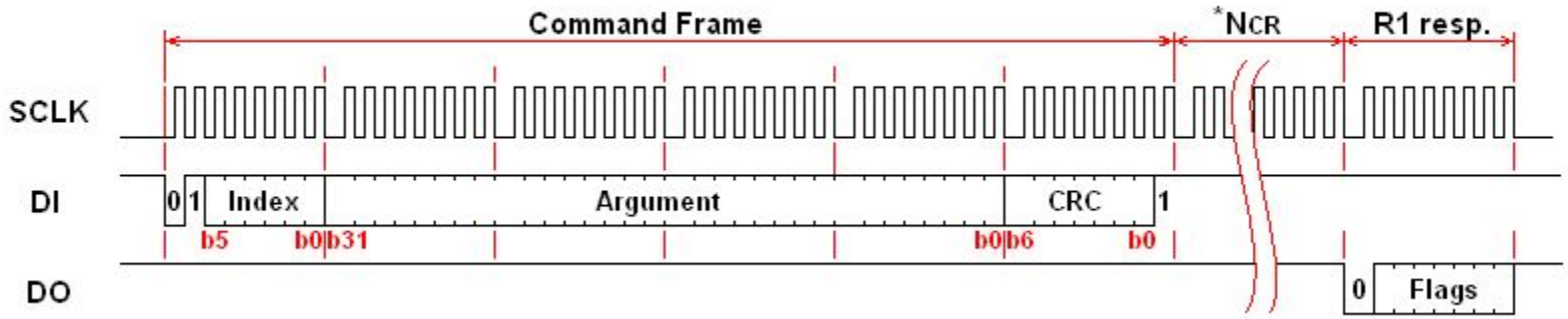
Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

Источник

Формат команды в режиме SPI

Команды, передаваемые по интерфейсу SPI, имеют размер 48 бит. Формат команд:

- Бит 47 (крайний слева) всегда содержит значение 0
- Бит 46 всегда содержит значение 1
- Биты 45..40 содержат индекс команды
- Биты 39..8 содержат аргументы команды
- Биты 7..1 содержат CRC от предыдущих бит
- Бит 0 всегда содержит значение 1

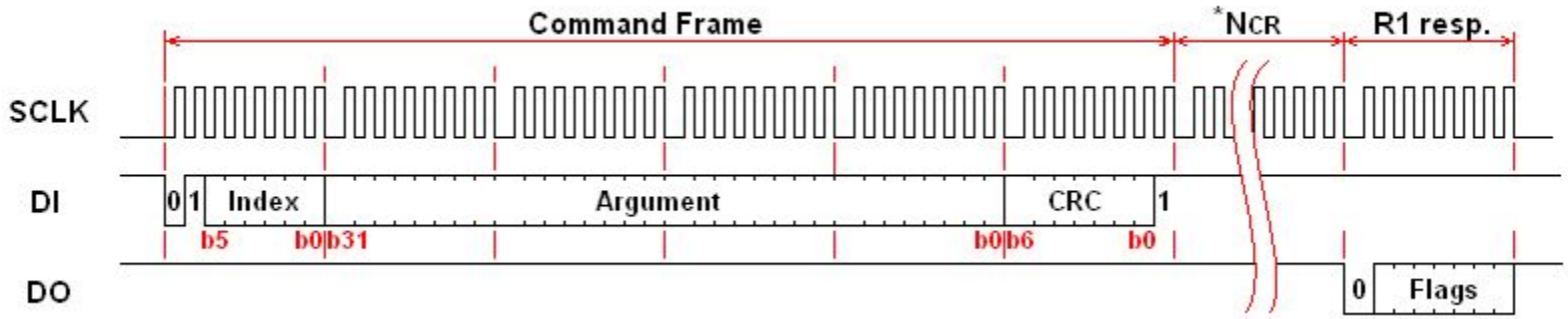


Источник

Формат команды в режиме SPI

Команды, передаваемые по интерфейсу SPI, имеют размер 48 бит. Формат команд:

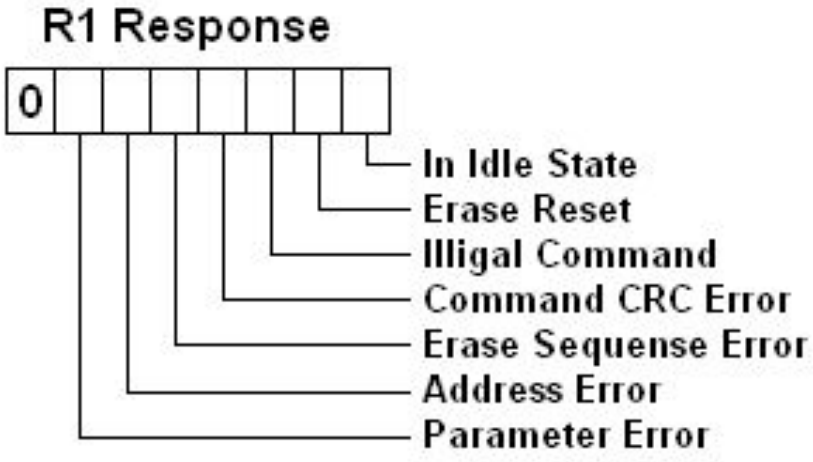
- Бит 47 (крайний слева) всегда содержит значение 0
- Бит 46 всегда содержит значение 1
- Биты 45..40 содержат индекс команды
- Биты 39..8 содержат аргументы команды
- Биты 7..1 содержат CRC от предыдущих бит
- Бит 0 всегда содержит значение 1

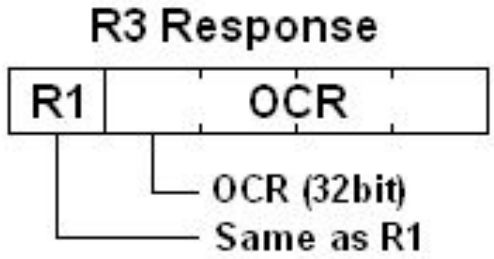


Источник

Формат ответа в режиме SPI

На команды, используемые при работе с SD-картой, будут поступать ответы типа R1, R3, R7. Все команды и ответы на них подробно расписаны в спецификации Physical Layer Simplified Specification

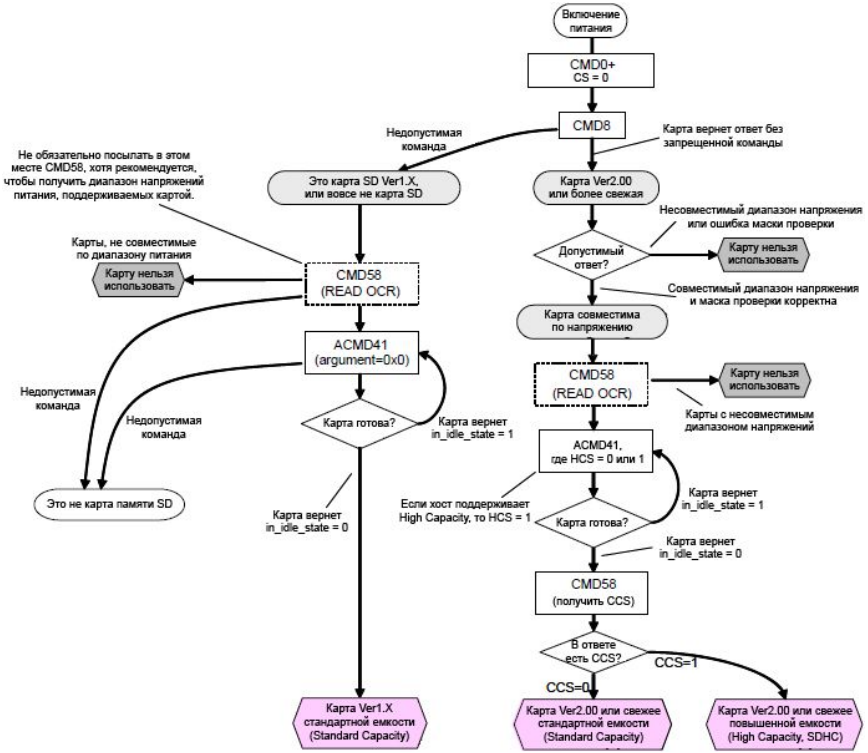




Источник

Алгоритм инициализации SD карты

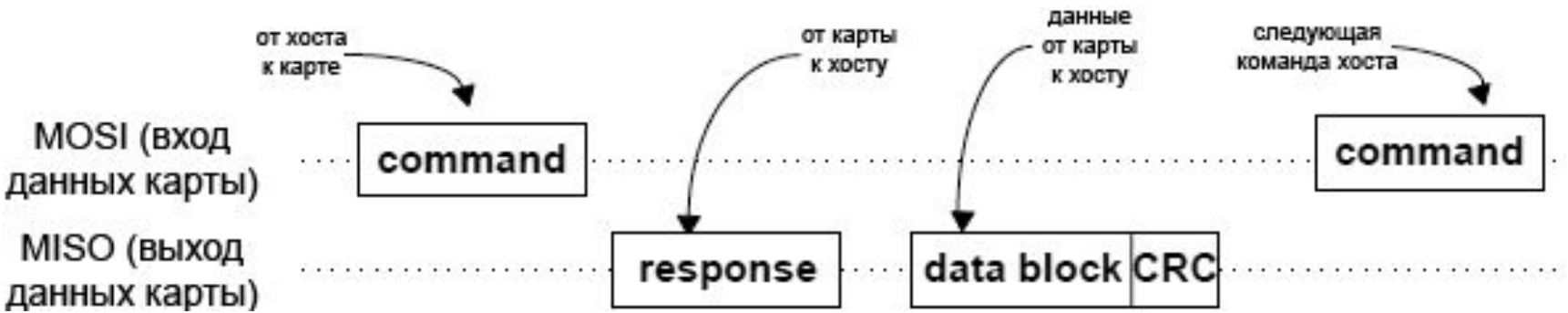
Алгоритм инициализации состоит в последовательной передаче набора команд и реакции на ответ от SD карты



Источник

Чтение данных из SD карты

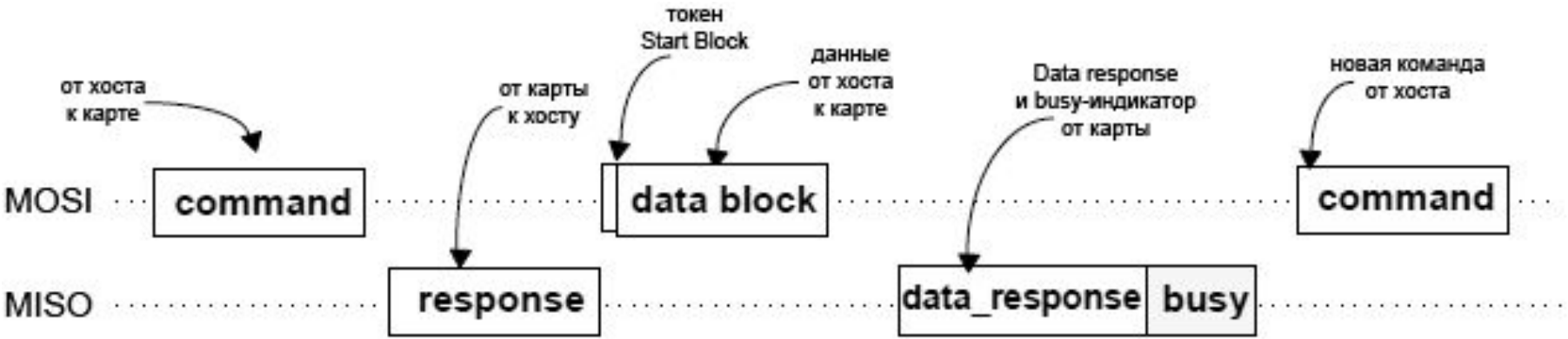
Режим SPI поддерживает одиночное чтение блока (Single Block Read) и множественное чтение блоков (Multiple Block Read), команды CMD17 (на рисунке) или CMD18 соответственно из традиционного протокола SD Memory Card



Источник

Запись данных в SD карту

Режим SPI поддерживает команды одиночной записи блока (Single Block Write) и множественной записи блоков (Multiple Block Write). На приеме команды записи CMD24 (на рисунке) или CMD25 протокола SD Memory Card, карта будет отвечать токеном response, и будет ждать блока, отправленного хостом



Источник

Библиотека SD для работы с SD картой

При подключении библиотеки автоматически вызывается конструктор, который создает объект с именем SD. Поэтому в пользовательском коде вызывать конструктор не нужно.

boolean begin(uint8_t csPin = SD_CHIP_SELECT_PIN) — инициализирует библиотеку и SD-карту памяти. Также активизируется шина SPI и вывод CS, в качестве которого по умолчанию выступает линия SS.

Параметры:

- csPin: (необязательный параметр) номер вывода, подключенного к линии CS SD-карты памяти; по умолчанию равен номеру вывода SS аппаратной шины SPI

Возвращаемое значение: при успешном выполнении true, в противном случае - false

Библиотека SD для работы с SD картой

boolean exists(const char *filepath) — проверяет существование на SD-карте указанного файла или директории
Параметры:

- filepath: имя файла, существование которого необходимо проверить. Имя может включать в себя директории, разделенные прямым слешем "/"

Возвращаемое значение: если указанный файл или директория существует - true, в противном случае - false

boolean mkdir(const char *filepath) — создает на SD-карте указанную директорию. Если путь к указанной директории содержит несуществующие папки - функция также их создает
Параметры:

- filepath: имя создаваемой директории. Имя может включать в себя вложенные папки, разделенные прямым слешем "/"

Возвращаемое значение: если директория успешно создана - true, в противном случае - false

Библиотека SD для работы с SD картой

File open(const char *filename, uint8_t mode = FILE_READ) — функция открывает файл на SD-карте памяти. Если указанный файл не существует, то при открытии для записи он будет создан автоматически. Директория, содержащая файл, заведомо должна существовать

Параметры:

- filepath: имя открываемого файла. Может содержать директории, разделенные прямым слешем "/"
- mode: (не обязательный параметр) режим открытия файла, по умолчанию - FILE_READ. Может принимать одно из следующих значений: FILE_READ - открыть для чтения с начала файла, FILE_WRITE - открыть для чтения или записи в конец файла

Возвращаемое значение: объект File, ссылающийся на открытый файл; если файл открыть не удалось, возвращаемый объект будет эквивалентен значению false (в логическом смысле)

Библиотека SD для работы с SD картой

boolean remove(const char *filepath) — удаляет файл с SD-карты

Параметры:

- filepath: имя удаляемого файла. Может содержать директории, разделенные прямым слешем "/"

Возвращаемое значение: если файл был успешно удален - true, в противном случае - false

boolean rmdir(const char *filepath) — удаляет директорию с SD-карты. Указываемая директория должна быть пустой

Параметры:

- filepath: имя удаляемой директории. Может содержать директории, разделенные прямым слешем "/"

Возвращаемое значение: если директория была успешно удалена - true, в противном случае - false

Библиотека SD для работы с SD картой

Библиотека также содержит класс File, который предназначен для чтения и записи данных в отдельные файлы на SD-карте памяти. Он является наследником класса Stream.

File(void) — конструктор, который создает объект класса File.

Параметры: нет
Возвращаемое значение: нет

int available() — проверяет наличие в файле байт, доступных для чтения

Параметры: нет
Возвращаемое значение: количество доступных для чтения байт

Библиотека SD для работы с SD картой

void close() — закрывает файл, удостоверившись, что все записанные в него данные физически сохранены на SD-карту памяти
Параметры: нет
Возвращаемое значение: нет

void flush() — физически сохраняет все записанные данные на SD-карту памяти
Параметры: нет
Возвращаемое значение: нет

int peek() — считывает из файла байт данных, не перемещая указатель текущей позиции файла на следующий символ
Параметры: нет
Возвращаемое значение: байт данных (или символ), либо -1, если таковой отсутствует

Библиотека SD для работы с SD картой

uint32_t position() — позволяет узнать текущее положение указателя внутри открытого файла
Параметры: нет
Возвращаемое значение: номер текущей позиции указателя внутри файла

size_t print(void data, int BASE) — выводит данные в файл, предварительно открытый для записи. При этом числа выводятся как последовательность цифр, каждая из которых является ASCII-символом
Параметры:

- data: данные, которые необходимо вывести (char, byte, int, long или string)
- BASE: (не обязательный параметр) система счисления, в которой необходимо выводить числа: BIN для двоичной системы (основание 2), DEC - для десятичной (основание 10), OCT - для восьмеричной (основание 8), HEX - для шестнадцатеричной (основание 16)

Возвращаемое значение: количество записанных байт

Библиотека SD для работы с SD картой

size_t println(void data, int BASE) — выводит данные в файл, заканчивающиеся символом перевода каретки и пустой строкой. Файл должен быть предварительно открыт для записи. При этом числа выводятся как последовательность цифр, каждая из которых является ASCII-символом

Параметры:

- data: (не обязательный параметр) данные, которые необходимо вывести (char, byte, int, long или string)
- BASE: (не обязательный параметр) система счисления, в которой необходимо выводить числа: BIN для двоичной системы (основание 2), DEC - для десятичной (основание 10), OCT - для восьмеричной (основание 8), HEX - для шестнадцатеричной (основание 16)

Возвращаемое значение: количество записанных байт

Библиотека SD для работы с SD картой

boolean seek(uint32_t pos) — изменяет положение текущей позиции внутри файла. Задаваемая позиция должна лежать в пределах от 0 до текущего размера файла (включительно)
Параметры:

- pos: позиция, в которую необходимо переместить указатель

Возвращаемое значение: в случае успешного выполнения - true, в противном случае - false

uint32_t size() — возвращает размер файла
Параметры: нет
Возвращаемое значение: размер файла в байтах

Библиотека SD для работы с SD картой

int read() — считывает байт данных из открытого файла
Параметры: нет
Возвращаемое значение: байт данных (или символ), либо -1, если таковых нет

size_t write(uint8_t buf)
size_t write(const uint8_t *buf, size_t size) — записывает данные в файл
Параметры:

- buf: записываемый байт данных, символ или строка
- size: количество элементов массива (при записи массива данных)

Возвращаемое значение: количество записанных байт

Библиотека SD для работы с SD картой

boolean isDirectory(void) — директории технически являются особым видом файлов. Данная функция позволяет проверить, является ли указанный файл директорией или нет.
Параметры: нет
Возвращаемое значение: true - директория, false - файл

File openNextFile(uint8_t mode = FILE_READ) — возвращает имя следующего файла или папки в пределах директории
Параметры:

- mode: (не обязательный параметр) режим открытия файла, по умолчанию - FILE_READ. Может принимать одно из следующих значений: FILE_READ - открыть для чтения с начала файла, FILE_WRITE - открыть для чтения или записи в конец файла

Возвращаемое значение: объект File, ссылающийся на следующий файл в пределах директории

Библиотека SD для работы с SD картой

char * name() — считывает имя файла
Параметры: нет
Возвращаемое значение: строка с именем файла (с расширением)

Считывание данных с SD карты

```
#include <SD.h>

const int CSPin = 10;
File root;

void setup()
{
  Serial.begin(115200);
  Serial.print("Initializing SD card... ");
  if (!SD.begin(CSPin)) //Если ошибка инициализации, то блокирование работы
  {
    Serial.println("Card initialization failed!");
    while (1);
  }

  Serial.println("initialization done.");
  Serial.println("Files in the card:"); //считывание имен всех файлов на карте
  root = SD.open("/");
  printDirectory(root); //вывод всех имен файлов в директории
  Serial.println("");
```

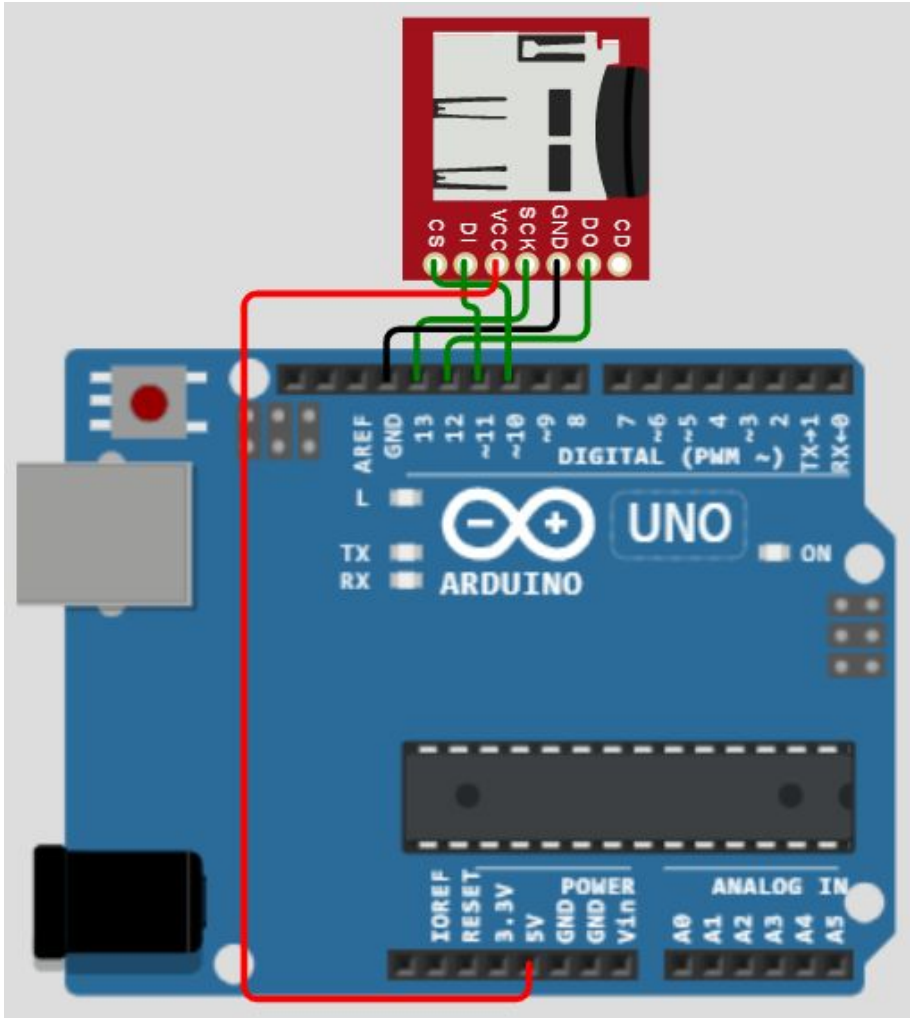
```
  //Пример чтения из файла
  File textFile = SD.open("wokwi.txt");
  if (textFile)
  {
    Serial.print("wokwi.txt: ");
    while (textFile.available())
    {
      Serial.write(textFile.read());
    }
    textFile.close();
  } else {
    Serial.println("error opening
wokwi.txt!");
  }
}
```

Считывание данных с SD карты

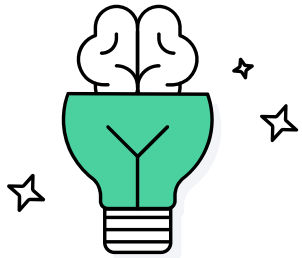
```
void loop()
{
}
//функция чтения имен файлов
void printDirectory(File dir)
{
  while (1)
  {
    File entry =  dir.openNextFile(); //переход к следующему файлу
    if (! entry) //выход, если больше нет файлов
    {
      break;
    }
    Serial.print(entry.name()); //вывод имени
    if (entry.isDirectory()) //если это директория, то ее размера нет
    {
      Serial.println("/");
    } else
    {
      Serial.print("\t\t");
      Serial.println(entry.size(), DEC);
    }
    entry.close();
  }
}
```

Считывание данных с SD карты

Открытая версия симулятора позволяет только считывать файлы. Запись в файл не поддерживается



Практическое задание №2



Практика: считывание данных с модуля MPU6050 с помощью библиотеки MPU6050

Задание:

1) соберите схему в симуляторе WOKWI, подключив SD карту ко выводам модуля SPI платы Arduino UNO;
2) создайте скетч с текстом, приведенным выше;
3) проведите моделирование работы

Как выполнять: напишите в чат об удачной работе схемы

Время выполнения: 5 минут

Итоги

4

Итоги занятия

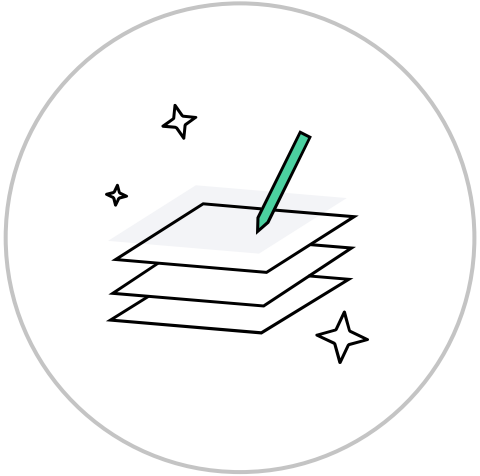
Сегодня мы

1. Узнали особенности интерфейса SPI
2. Научились использовать сдвиговый регистр для управления светодиодами
3. Узнали особенности использования SD карт памяти
4. Научились применять специальную библиотеку для работы с SD картами памяти

Домашнее задание

Давайте посмотрим ваше домашнее задание.

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Задавайте вопросы и пишите отзыв о лекции

Павел Пронин
C++ разработчик