Лабораторная работа № 5

Изучение SPI

Цель лабораторной работы: познакомиться с принципами организации SPI, а также научиться использовать SPI на платформе Arduino.

Теоретическая часть

SPI (англ. Serial Peripheral Interface, SPI bus — последовательный периферийный интерфейс, шина SPI) — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии. **SPI** также иногда называют четырёхпроводным (англ. four-wire) интерфейсом.

SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством.

Возможно два вида подключения в интерфейсе SPI: независимое и каскадное. В первом случае при подключении Master обращается к каждому Slave индивидуально, во втором случае подключение происходит по очереди, т.е. каскадно (Рисунок 1).

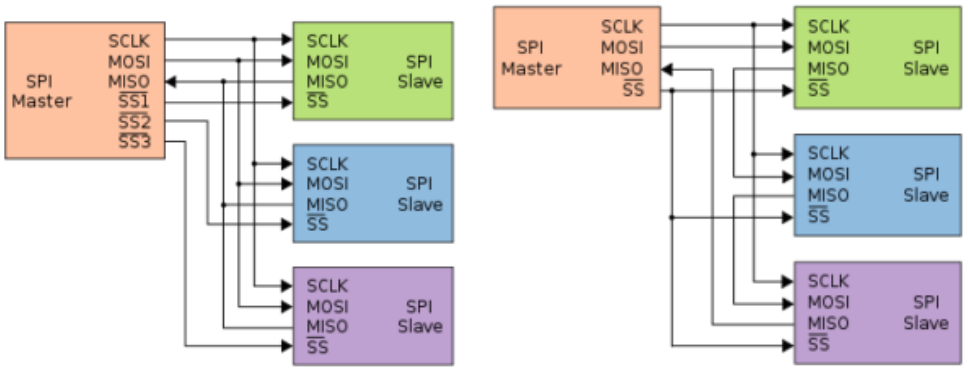

Рисунок 1 – Виды подключения в интерфейсе SPI

Каждое устройство, поддерживающее подключение к шине SPI, имеет четыре линии, по которым осуществляется обмен данными:

MOSI (Master-Out, Slave-In — ведущий посылает, ведомый принимает),

MISO (Master-In, Slave-Out — ведущий принимает, ведомый посылает),

SCK (Serial Clock Line, тактовая линия)

SS или **CS** (Slave Select или Chip Select — выбор ведомого или выбор устройства).

Контакты с D11 по D13 на плате (Рисунок 2) зарезервированы для подключения линий MISO, MOSI и SCK шины SPI, но линию SS можно подключить к любому другому цифровому контакту (часто для этого используется контакт D10, потому что он расположен рядом с контактами, зарезервированными для шины SPI).

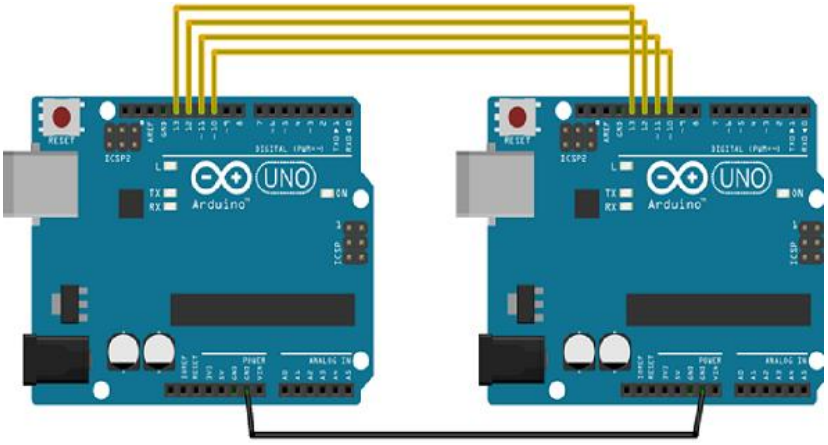

Рисунок 2 – Подключение Arduino to Arduino по SPI

Взаимодействие устройств начинается, когда на выход SS подается низкий уровень сигнала.

Изначально в Ардуино заложено, что данные передаются старшим битом вперед, но перед началом нужно уточнить это в документации.

У SPI есть четыре режима передачи (Рисунок 3), которые основаны на комбинации «полярности» тактового сигнала (clock polarity, CPOL) и фазы синхронизации (clock phase, CPHA). CPOL — это уровень на тактовой линии до начала и после окончания передачи. А фаза определяет, на фронте или спаде тактового сигнала передавать биты:

Режим 0: CPOL=0, CPHA=0
Чтение бита происходит на фронте тактового сигнала (переход 0 -> 1), а запись — на спаде (1 -> 0).

Режим 1: CPOL=0, CPHA=1
Чтение — на спаде, запись — на фронте.

Режим 2: CPOL=1, CPHA=0
Чтение — на спаде, запись — на фронте.

Режим 3: CPOL=1, CPHA=1
Чтение — на фронте, запись — на спаде.

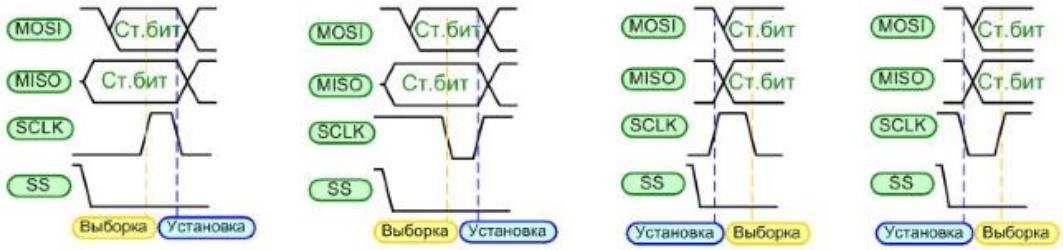

Рисунок 3 – Режимы SPI

Программная реализация SPI на Arduino

```
/* Определяем выводы микроконтроллера для шины SPI
*/

#define CS 8
#define MOSI 9
#define MISO 10
#define CLK 11
#define SPI_TIME 100

/*Настраиваем пины на выход*/

void setup() {
  pinMode(MOSI,OUTPUT);
  pinMode(CLK,OUTPUT);
  pinMode(CS,OUTPUT);
}

/*Основная функция с использованием функций SPI для примера*/
void loop() {
  SPI_Start();
  SPI_Write(147);
  SPI_Stop();
}

/*Функция записи байта*/
void SPI_Write(byte data) {
  for (byte i = 0; i < 8; i++) {
    delayMicroseconds(SPI_TIME / 2);

    if (data & 0x80)
      digitalWrite(MOSI, HIGH);
    else
      digitalWrite(MOSI, LOW);

    data <<= 1;
    delayMicroseconds(SPI_TIME / 2);
    digitalWrite(CLK, HIGH);
    delayMicroseconds(SPI_TIME);
    digitalWrite(CLK, LOW);
  }
}
/*Функция чтения байта*/
byte SPI_Read() {
  byte data = 0;
  for (byte i = 0; i < 8; i++) {
    delayMicroseconds(SPI_TIME);
```

```
    digitalWrite(CLK, HIGH);
    delayMicroseconds(SPI_TIME / 2);

    if (digitalRead(MISO)) data++;
    if (i != 7) data <<= 1;

    delayMicroseconds(SPI_TIME / 2);
    digitalWrite(CLK, LOW);
    }
  return data;
}
/*Инициализация SPI*/
void SPI_Start() {
  delayMicroseconds(SPI_TIME);
  digitalWrite(CLK, LOW);
  digitalWrite(CS, LOW);
  delayMicroseconds(SPI_TIME);
}
/*Остановка SPI*/
void SPI_Stop() {
  delayMicroseconds(SPI_TIME);
  digitalWrite(CS, HIGH);
  delayMicroseconds(SPI_TIME);
}
```

Сдвиговый регистр 74HC595

Сдвиговый регистр - это набор последовательно соединённых триггеров (обычно их 8 штук) (Рисунок 4). В отличии от стандартных регистров, сдвиговые поддерживают функцию сдвига вправо и влево.

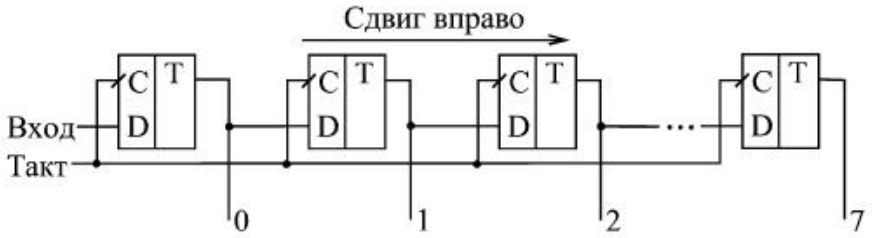

Рисунок 4 – Принципиальная схема регистра сдвига

Сдвиговый регистр 74HC595 (Рисунок 5) работает на интерфейсе SPI:

- выводы DS, ST_CP, SH_CP - это шины управления. Соответственно: шина данных(MOSI), защёлка(SS) и тактовая линия(SCK). Выходы Q0, Q1, ..., Q7 - это выходы регистра (разряды).
- Выход Q7` предназначен для последовательного соединения таких регистров

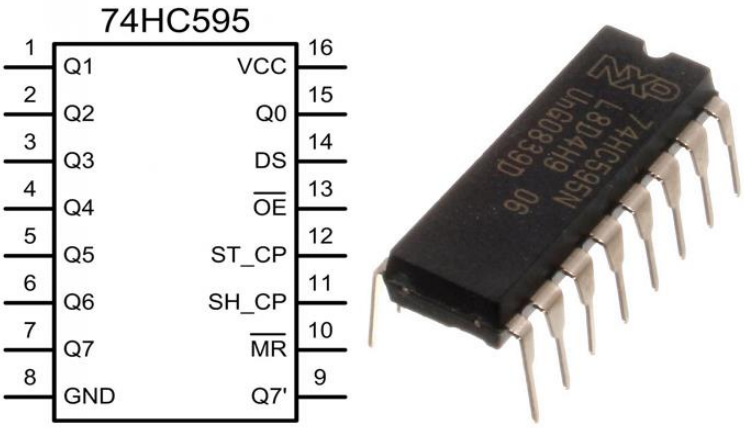

Рисунок 5 – Распиновка и внешний вид регистра сдвига 74HC595

- OE - включение выходов (подключаем к контакту GND).
- VCC и GND - это питание.
- MR - это сброс. Подключаем к +5v (сброс не активен).

Практическая часть

1. Используя программную реализацию SPI передать информацию на сдвиговый регистр 74HC595. Организовать вывод двоичного числа, введенного в консоль. (Схема представлена на рисунке 6)

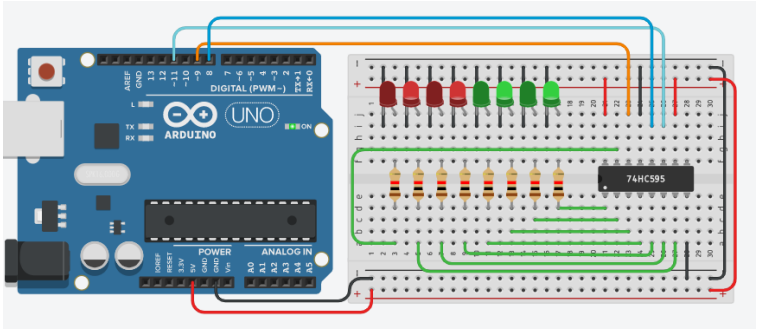

Рисунок 6 – Схема к задаче 1

2. Используя программную реализацию SPI связать две Arduino, передавая данные от одной платы к другой, введённые в консоль.

3. Используя библиотеку SPI для Arduino осуществить связь микроконтроллера со считывателем RFID RC522. Данные, считанные с карты или чипа передать в последовательный порт в виде текста (используя Serial.println())

4. Связать три или более Arduino посредством SPI в поочерёдном режиме. Устройство master должно принимать в последовательный порт информацию и номер абонента slave, для которого эта информация предназначена. Устройства slave принятую информацию выводят в консоль.