Лабораторная работа № 1

Цифровой ввод/вывод в Arduino

Цель лабораторной работы: познакомиться с принципами элементарного ввода/вывода информации в цифровом (битовом) виде. Научиться настраивать выводы микроконтроллера на соответствующий задаче режим работы.

Теоретическая часть

Основные понятия

Любой микроконтроллер взаимодействует с окружающим миром посредством **GPIO** (с англ. General Purpose Input-Output) - это входы-выходы общего назначения. **Pin (Пин)** – это вывод микроконтроллера микроконтроллера, имеющий свой номер, по которому к нему можно обратиться. **Порт** – это совокупность пинов.

Микроконтроллер спроектирован так, чтобы обеспечить хорошее быстродействие, поэтому например пины объединены в порты, в одном порте обычно до 8 пинов (потому, что в байте 8 бит). Работая с микроконтроллером напрямую, мы можем за одно действие установить состояние для всех пинов одного порта. Это очень быстро.

На рисунке 1 представлена распиновка платформы Arduino Uno. Здесь можно увидеть, какие из пинов являются цифровыми.

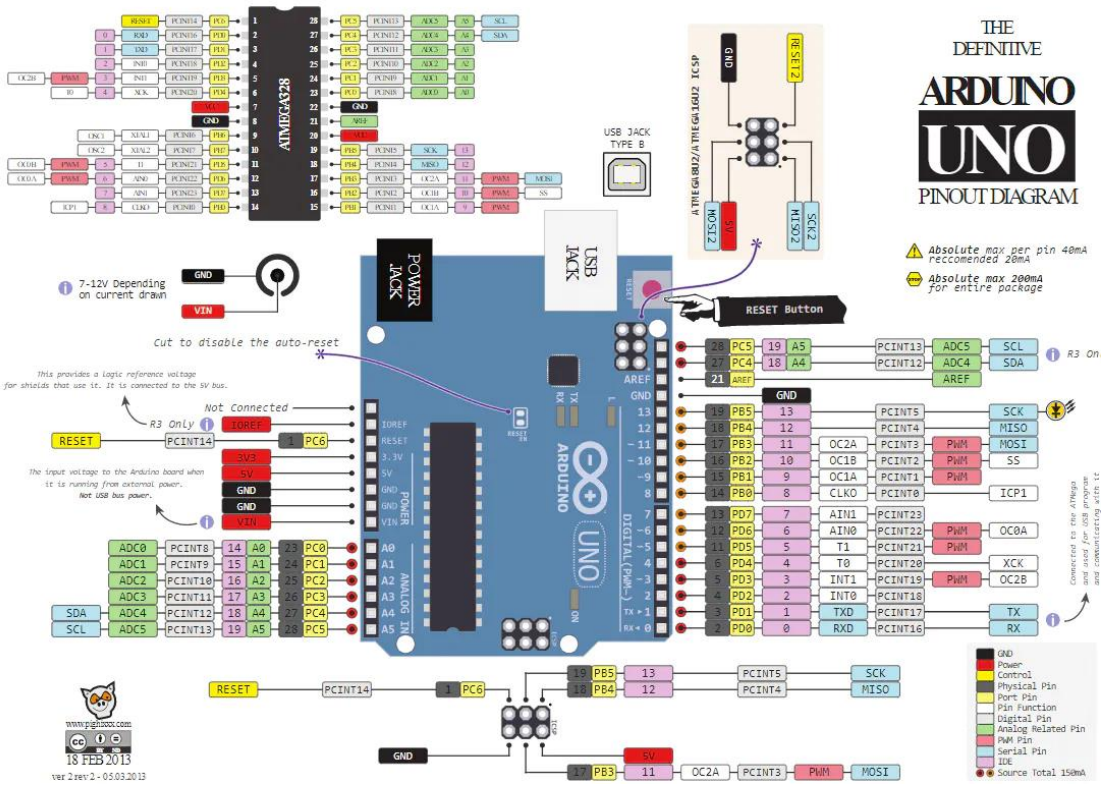

Рисунок 1 - Распиновка платы Arduino Uno

Все пины, имеющие название PD*, PC*, PB*, являются GPIO. D, C и B указывает на номер порта, которому принадлежит пин. На плате пины подписаны по-другому, просто по порядку. Таким образом видно, что все пины с D0-D13 и A0-A5 являются GPIO, то есть цифровыми входами-выходами. В некоторых источниках информации пины A0–A7 называют аналоговыми, на самом деле это цифровые пины с дополнительной функцией чтения аналогового сигнала.

Пины на плате Arduino Uno промаркерованы как цифровые D* и аналоговые A* . К цифровым пинам из программы обращаются по их номеру (т.е. D3 это просто 3). Обращение к пинам A* возможно двумя способами:

- Обращение через букву с номером (например: A0, A3...
- Номером по порядку после цифровых, так например у Arduino Uno последний цифровой – D13, следующий за ним "аналоговый" A0 будет иметь номер 14, и так далее до A5, который имеет номер 19, по которому к нему тоже можно обратиться.

Режимы работы пинов

Цифровой пин может находиться в двух состояниях: вход и выход. В режиме входа пин может считывать логические уровни напряжения, а в режиме выхода – выдавать такие же уровни. Режим работы выбирается при помощи функции

pinMode(pin, mode),

где **pin** это номер пина, а **mode** это режим:

Режим работы (**mode**) может быть следующим:

- **INPUT** – вход
- **OUTPUT** – выход
- **INPUT_PULLUP** – подтянутый к питанию вход

Режим **INPUT_PULLUP** подразумевает использование внутреннего подтягивающего резистора (pull-up). В режиме входа пин микроконтроллера никуда не подключен и на нём наводятся помехи, которые могут иметь различный логический уровень. Для задания пину "состояния по умолчанию" используют подтяжку резистором к земле или питанию. Таким образом,

режим INPUT_PULLUP включает встроенную в микроконтроллер подтяжку пина к питанию (Рисунок 2).

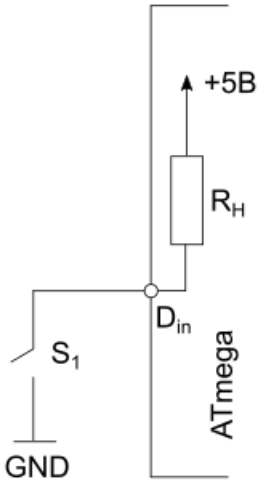

Рисунок 2 - Внутренний подтягивающий резистор, подключаемый программно.

Важно помнить, что по умолчанию все пины сконфигурированы как входы (INPUT). Поэтому, если есть пин должен работать на вход всегда, то необязательно настраивать его явным образом (с использованием функции pinMode).

Для настройки группы пинов на соответствующий режим работы удобно использовать циклический перебор:

```
for (byte i = 2; i <= 19; i++) {
pinMode(i, OUTPUT);
}
```

Цифровой вывод

Цифровой пин в режиме выхода (OUTPUT) может генерировать цифровой сигнал, т.е. выдавать сигнал низкого или высокого уровня. Сигнал низкого уровня это 0 Вольт, грубо говоря в этом состоянии пин подключается к GND микроконтроллера. Сигнал высокого уровня подключает пин к VCC микроконтроллера, то есть к питанию.

Микроконтроллер – это логическое устройство, которое создано для управления другими устройствами при помощи логических (цифровых) сигналов. Под словом логическое подразумевается не силовое, то есть напрямую запитывать внешние устройства от микроконтроллера нельзя. Максимальное значение тока, протекающего через пин должно быть не более 40 мА, а рекомендуемое производителем не более 20 мА. В других микроконтроллерах ограничение по току на пин может составлять 5-10 мА.

Также есть общее ограничение на ток с цифровых пинов – 200 мА. При превышении допустимых значений токов микроконтроллер выйдет из строя, поэтому ничего мощнее светодиода или небольшого звукового излучателя к микроконтроллеру подключать нельзя. Никаких моторчиков, лампочек, нагревателей, мощных радио-модулей и прочего питать от цифровых пинов нельзя. Цифровые пины служат для подачи команд другим устройствам, например транзисторам или реле, для коммутации нагрузок.

Для подачи логического уровня на пин Arduino существует функция:

digitalWrite(pin, value), где

- **pin** – номер цифрового пина микроконтроллера, подписанного на плате как D . Для A0-A5 номера 14-19.
- **value** – уровень сигнала: HIGH высокий, LOW низкий. Также можно использовать цифры 0 и 1

Пин, настроенный как OUTPUT, по умолчанию имеет сигнал LOW

Пример, в котором пины инициализируются как выходы, и на них формируется логический уровень:

```
void setup() {
  pinMode(11, OUTPUT);  // D10 как выход
  pinMode(A3, OUTPUT);  // A3 как выход
  pinMode(19, OUTPUT);  // A5 как выход
  digitalWrite(11, HIGH); // высокий сигнал на D10
  digitalWrite(A3, 1);    // высокий сигнал на A3
  digitalWrite(19, 1);    // высокий сигнал на A5
}
void loop() {}
```

Цифровой ввод

Цифровой пин может в режиме входа работает с сигналом низкого уровня - LOW (от 0 до ~2 В) или с сигналом высокого уровня - HIGH (от ~2 В до VCC). Таким образом микроконтроллер может работать с логическими устройствами, которые шлют ему высокий сигнал с напряжением 3.3V, он такой сигнал примет как HIGH. Уровни сигнала, выходящие за диапазон 0-5В (VCC) повредят микроконтроллер, поэтому крайне необходимо правильно реализовать сопряжение микроконтроллера с внешними устройствами.

Для чтения уровня сигнала на пине используется функция

digitalRead(pin), где

- pin – номер пина на плате. Это пины, подписанные как D, а также пины A0-A5 у Arduino Uno.

Данная функция возвращает 0, если сигнал низкого уровня, и 1 – если высокого.

Пример:

```
byte button;

void setup() {

}

void loop() {

  button = digitalRead(5);

}
```

Данный код будет сохранять в переменную button состояние кнопки, подключенной к пину D5. Если подключить пин проводом к VCC – в button получим 1, если к GND – в button получим 0.

Практическая часть

Упражнение 1

1. Разработать программу бегущего огня из пяти светодиодов с использованием функции задержки delay(time), где time - время задержки в миллисекундах. Время задержки между переключениями 100 мс.

2. Разработать программу "Светофор", которая моделирует работу реального светофора.

3. Разработать программу, отображающую на светодиодах в двоичном коде числа от 0 до 15. (* с использованием побитовых операций) с задержкой в 0.5 с.

Упражнение 2

1. Написать программу зажигания светодиода по нажатию кнопки. При каждом нажатии на кнопку состояние пина светодиода должно меняться на противоположное. Необходимо разработать программную фиксацию состояния светодиода, а также использовать алгоритм антидребезга кнопки.

2. Разработать программу гирлянды из пяти светодиодов с возможностью смены режимов (больше трёх) при помощи кнопки. Отдельные режимы оформить в виде функций.

3. Разработать игру "Кнопочные ковбои". Игра состоит в следующем: в устройстве должно быть 2 кнопки (2 игрока) и 3 светодиода (сигнальный и два светодиода участников). Задача двух игроков - быстрее соперника нажать на свою кнопку при зажигании сигнального светодиода. При выигрыше зажечь светодиод победителя.