Лабораторная работа 1. Параллельные порты ввода/вывода

Устройство параллельных портов микроконтроллера

Параллельные порты предназначены для обмена многоразрядной двоичной информацией между микроконтроллером и внешними устройствами, при этом в качестве внешних устройств могут использоваться любые объекты управления или источники информации (различные кнопки, датчики, дополнительная память, исполнительные механизмы, двигатели, реле, микросхемы, другие микроконтроллеры и так далее).

Каждый из портов содержит регистр, имеющий байтовую и битовую адресацию для установки (запись “1”) или сброса (запись “0”) разрядов этого регистра с помощью программного обеспечения. Выходы этих регистров соединены с внешними ножками микросхемы. Разрядность регистров соответствует разрядности микроконтроллера.

С точки зрения внешнего устройства порт представляет собой обычный источник или приемник информации со стандартными цифровыми логическими уровнями (обычно ТТЛ), а с точки зрения микропроцессора — это ячейка памяти, в которую можно записывать данные или в которой «сама собой» появляется информация.

В зависимости от направления передачи данных параллельные порты называются портами ввода, вывода или портами ввода/вывода.

Параллельные порты микроконтроллера ATMega328p

Микроконтроллер ATMega328p имеет три параллельных порта – B, C и D. На рисунке 1 приведена распиновка микроконтроллер ATMega328p.

Схема порта представлена на рисунке 2. Для управления состояниями портов в микроконтроллере ATMega328p имеется три регистра: DDRx, PORTx и PINx, где x – номер порта, B, C или D.

- DDRx – регистр направления порта, если в соответствующий бит записана логическая единица «1» - то этот пин будет работать как выход, а если «0» - то как вход.
- PORTx – регистр управления состоянием вывода. Если пин находится в режиме «Выхода», то 1 и 0 определяют наличие этих же сигналов на выходе. Если же пин находится в режиме «Входа», то «1» подключает подтягивающий резистор, если «0» – высокоимпедансное состояние;
- PINx – регистр чтения. Соответственно в нём находится информация о текущем состоянии выводов порта (логическая единица или ноль).

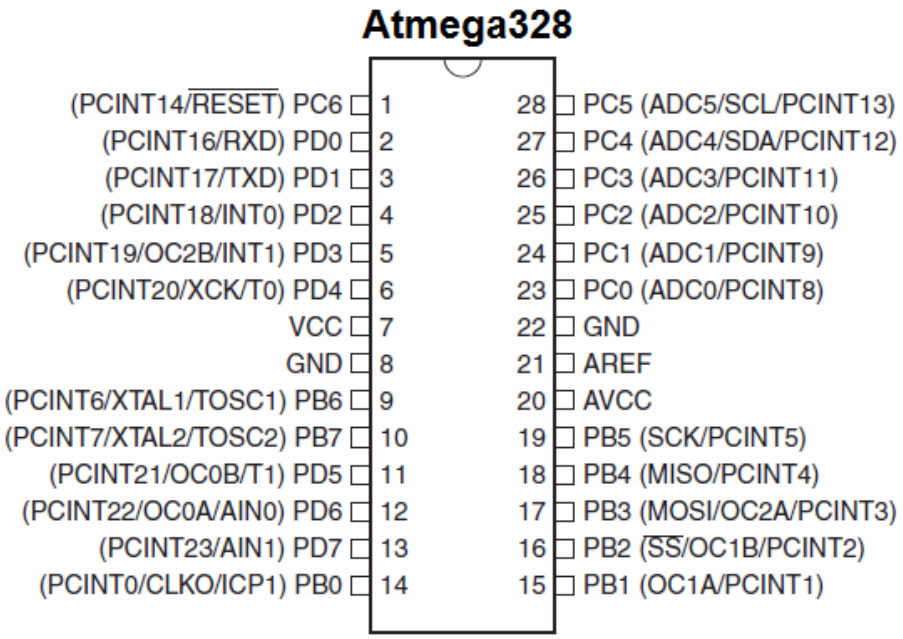


Рисунок 1 – Распиновка микроконтроллера ATMega328p

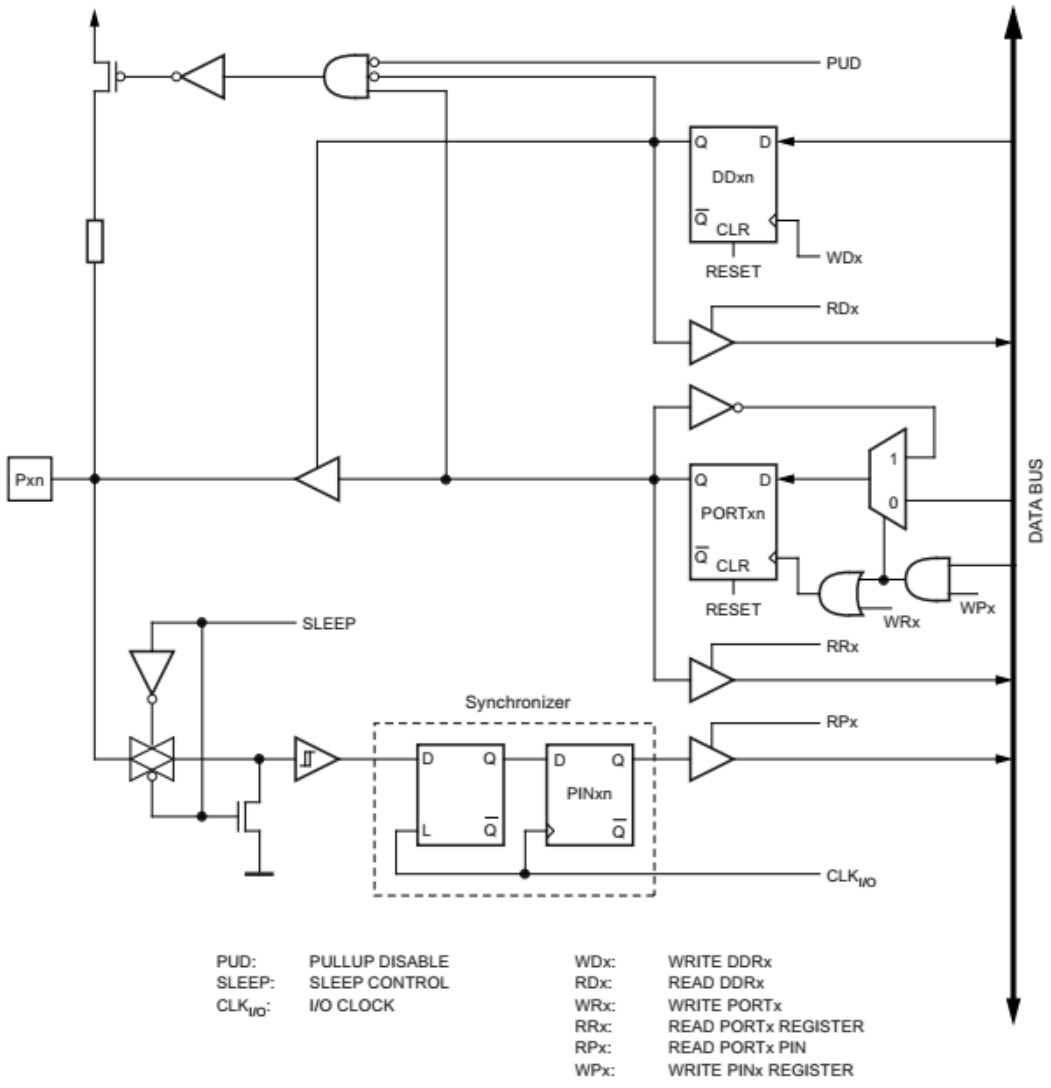


Рисунок 2 –Схема параллельного порта микроконтроллера ATMega328p

Описание регистров параллельного порта B

DDRB – The Port B Data Direction Register (регистр направления порта B)

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PORTB – The Port B Data Register (регистр данных порта B

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PINB – The Port B Input Pins Address (регистр входных данных порта B)

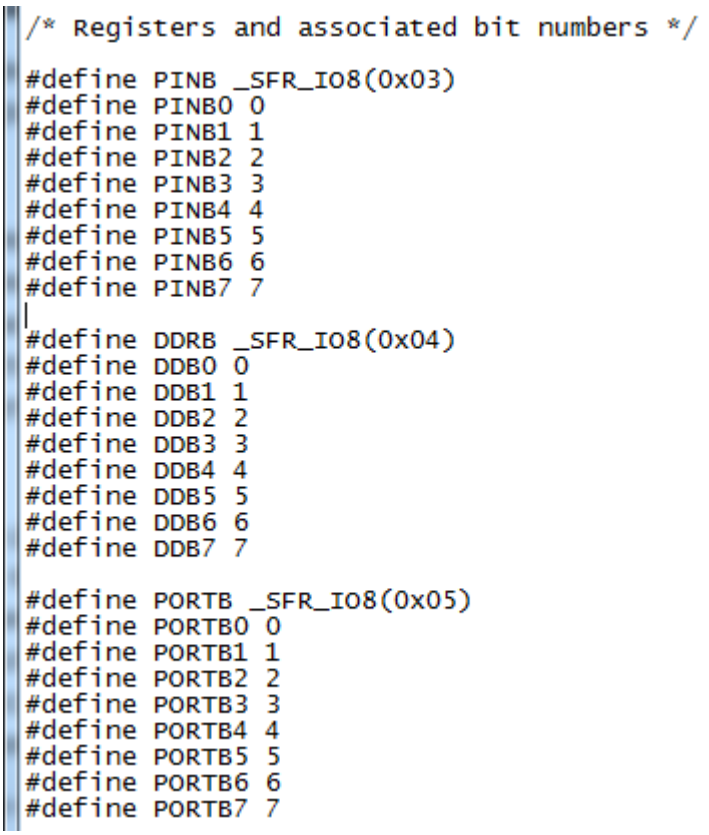
Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Программирование микроконтроллера

Для написания программы будет использоваться язык C для микроконтроллеров. Среда программирования – Arduino IDE.

В программе регистр микроконтроллера представляется как регистровая переменная. Для каждого микроконтроллера существует свой заголовочный файл, в котором описаны физические адреса регистров. В Arduino IDE такой файл подключается автоматически, в других средах, например, Atmel Studio, его нужно будет подключить явно.

На рисунке 3 приведена часть заголовочного файла для микроконтроллера ATMega328p. Макрос _SFR_IO8(…) определяет адрес регистра.



```
/* Registers and associated bit numbers */

#define PINB _SFR_IO8(0x03)
#define PINB0 0
#define PINB1 1
#define PINB2 2
#define PINB3 3
#define PINB4 4
#define PINB5 5
#define PINB6 6
#define PINB7 7

#define DDRB _SFR_IO8(0x04)
#define DDB0 0
#define DDB1 1
#define DDB2 2
#define DDB3 3
#define DDB4 4
#define DDB5 5
#define DDB6 6
#define DDB7 7

#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
#define PORTB1 1
#define PORTB2 2
#define PORTB3 3
#define PORTB4 4
#define PORTB5 5
#define PORTB6 6
#define PORTB7 7
```

Рисунок 3 – Часть заголовочного файла iom328p.h

Программа для микроконтроллера всегда должна выполняться в бесконечном цикле, т.к. в противном случае при завершении выполнения «полезного» кода счётчик команд может и далее увеличиваться. Это приведёт к тому, что будут выполняться «случайные» команды из непроинициализированной памяти («мусор»). А это, в свою очередь, может привести к плохим последствиям, вплоть до выхода из строя оборудования и даже катастроф.

На рисунке 4 приведён внешний вид среды Arduino IDE с шаблоном для программы.

Компилирование программы производится с помощью кнопки «проверить», а загрузка с помощью кнопки «загрузка»

Перед загрузкой программы в плату необходимо в меню инструменты выбрать порт, к которому она подключена. На рисунке 5 приведён пример соответствующей настройки.

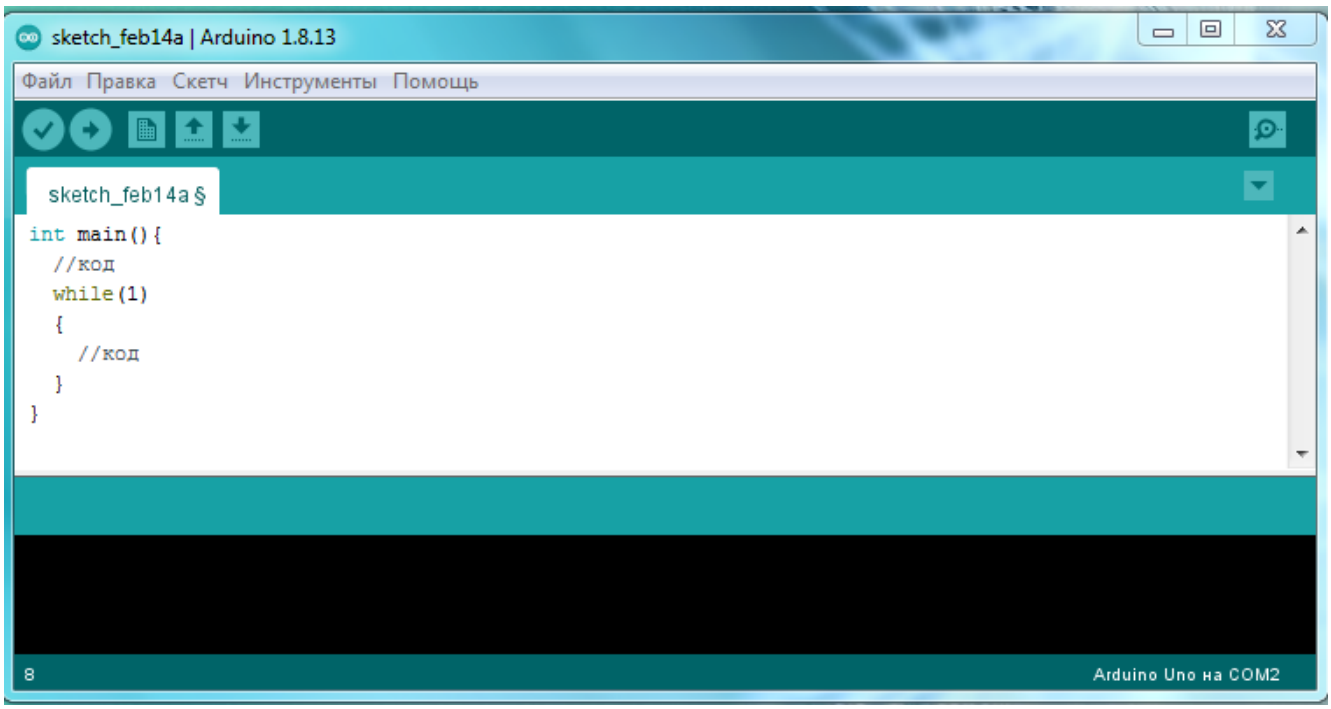


Рисунок 4– Внешний вид среды Arduino IDE

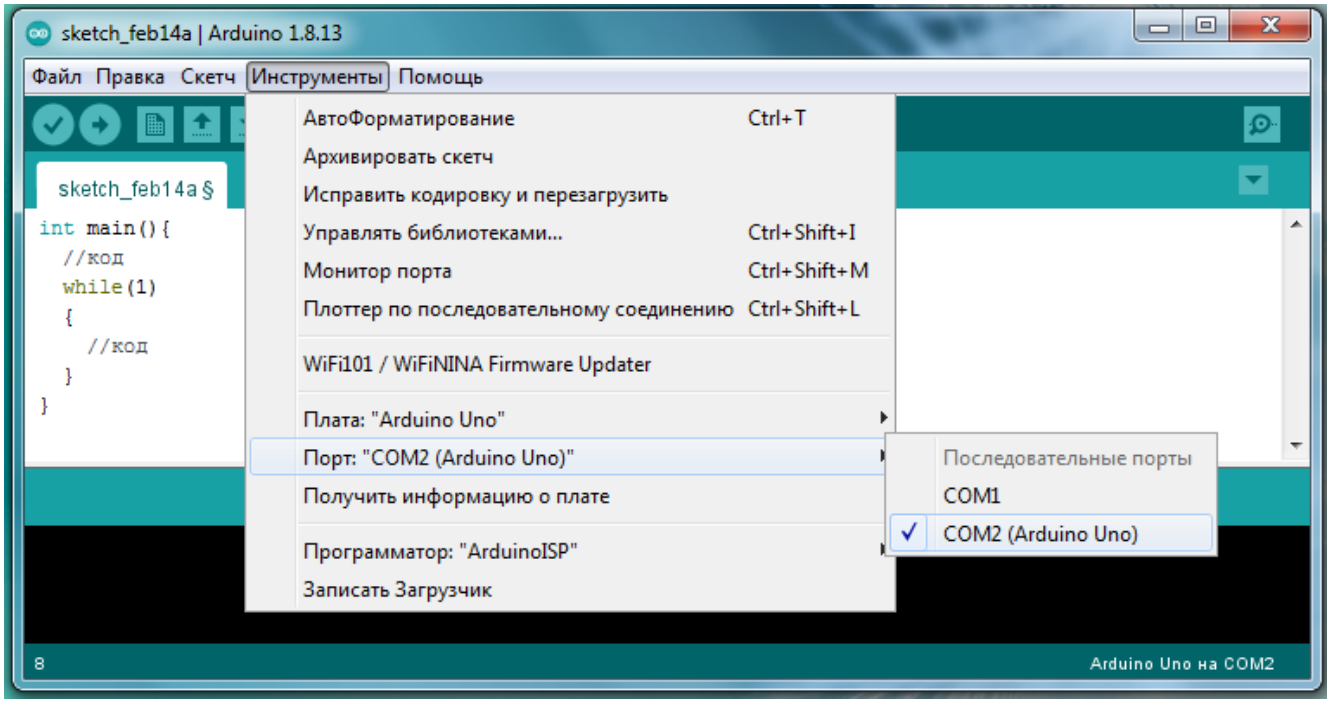


Рисунок 5 – Выбор порта

Подключение внешнего светодиода и кнопки

Для подключения внешнего светодиода к плате Arduino можно использовать цифровые порты (на плате обозначены как digital) со 2 по 12.

Схемы подключения светодиода представлены на рисунке 6. Подключить можно Для уменьшения тока понадобится ограничивающий резистор номиналом около 0,5 кОм.

На рисунке 7 приведена схема подключения кнопки. Подтягивающий резистор можно выбрать любого номинала.

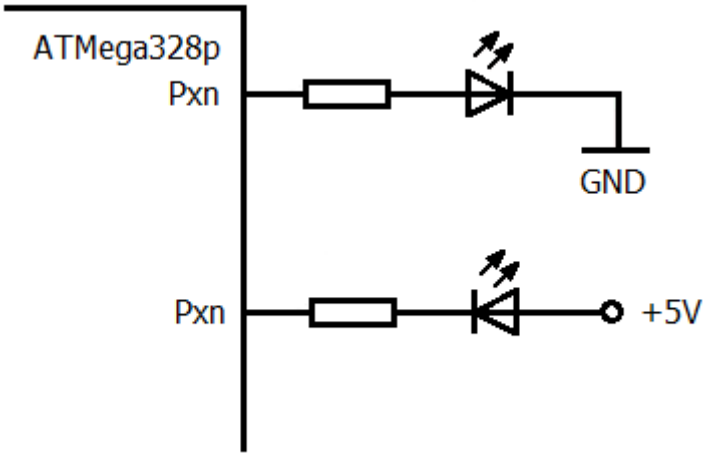


Рисунок 6 – Подключение внешнего светодиода (два способа)

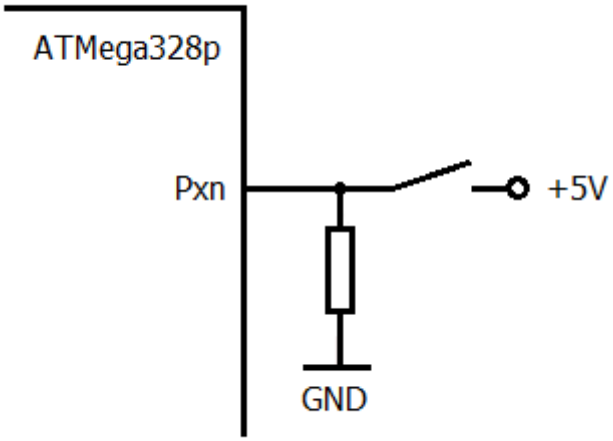


Рисунок 7 – Подключение кнопки

Для определения соответствия порта микроконтроллера портам Arduino можно посмотреть по рисунку 8.

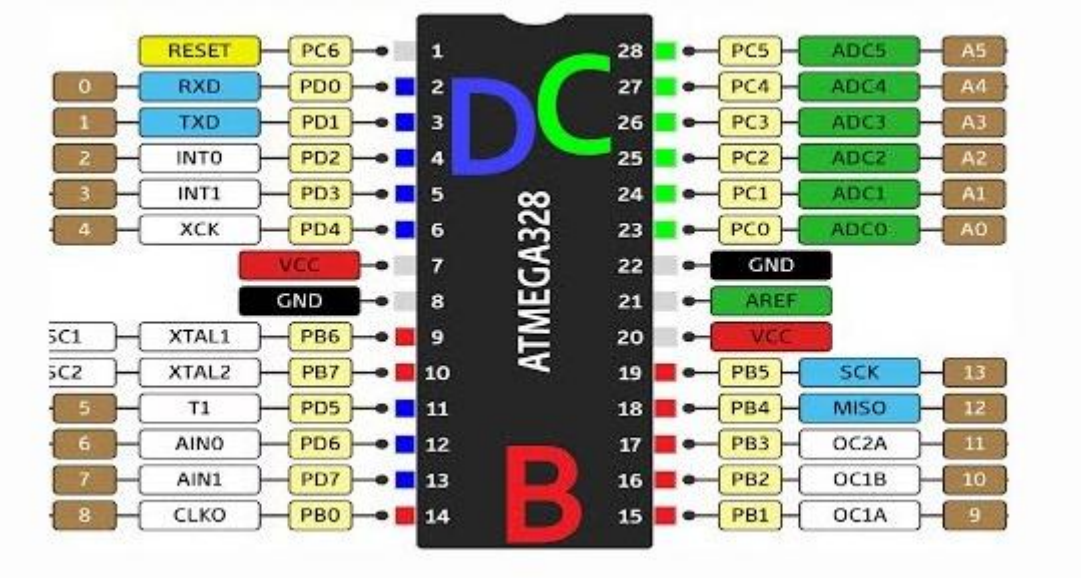


Рисунок 8 – Распиновка ATMega328p с указанием портов Arduino

Программирование портов при помощи библиотек Arduino IDE

Используя стандартные библиотеки Arduino IDE установить режим работы порта можно с помощью функции pinMode. В ней указывается номер порта и его роль:

pinMode (nomer_porta, naznachenie)

Есть три режима в которых может работать порт:

INPUT – вход, в этом режиме происходит считывание данных с датчиков, состояния кнопок, аналогового и цифрового сигнала. Порт находится в т.н. высокоимпедансном состоянии, простыми словами – у входа высокое сопротивление. Устанавливается это значение, на примере 13 пина платы, при необходимости так:
pinMode (13, INPUT);

OUTPUT – выход, в зависимости от команды прописанной в коде порт принимает значение единицы или нуля. Выход становится своего рода управляемым источником питания и выдаёт максимальный ток (в нашем случае 20 мА и 40 мА в пике) в нагрузку к нему подключенную. Чтобы назначить порт как выход на Arduino нужно ввести:
pinMode (13, OUTPUT);

INPUT_PULLUP – порт работает как вход, но к нему подключается т.н. подтягивающий резистор в 20 кОм.
Условную внутреннюю схему порта в таком состоянии вы видите ниже. Особенностью этого входа является то, что входной сигнал воспринимается как проинвертированный («единица» на входе воспринимается микроконтроллером как «ноль»). В таком режиме можно не использовать внешние подтягивающие резисторы при работе с кнопками.

pinMode (13, INPUT_PULLUP);

Данные принимаются с портов и передаются на них с помощью функций:

- digitalWrite(пин, значение) – переводит выходной пин в логическую 1 или 0, соответственно на выходе появляется или исчезает напряжение 5В, например digitalWrite (13, HIGH) – подаёт 5 вольт (логическую единицу) на 13 пин, а digitalWrite (13, low) – переводит 13 пин в состояние логического ноля (0 вольт);
- digitalRead(пин) – считывает значение со входа, пример digitalRead (10), считывает сигнал с 10 пина;
- analogRead(пин) – считывает аналоговый сигнал с аналогового порта, вы получаете значение в диапазоне от 0 до 1023 (в пределах 10-битного АЦП), пример analogRead (3).

Задание

Задание к работе:

1. По схеме определите к какому пину подсоединён светодиод L.
2. Напишите программу, используя регистры, которая будет зажигать и гасить светодиод через некоторые промежутки времени (для задержки на данном этапе можно использовать функцию _delay_ms (int ms)).
3. Используя макетную плату, провода и резистор подключите светодиод на другой вход Arduino. Определите по схеме номер порта и пина. Модифицируйте программу для работы с данным светодиодом.
4. Подключите к плате Arduino кнопку. Напишите программу, определяющую нажатие. Чтобы определить нажатие визуально, можно по нажатию останавливать мигание, гасить или зажигать светодиод.