Лабораторная работа 1

Изучение принципов получения и обработки информации с датчиков через дискретный и аналоговый интерфейсы

Цель и задачи лабораторной работы

Цель: Получить практические навыки считывания и обработки сигналов с датчиков через дискретный и аналоговый интерфейсы Arduino (микроконтроллера).

Задачи:

1) Ознакомиться с теорией.
2) Изучить способы подключения датчиков к дискретному и аналоговому интерфейсам контроллера.
3) Освоить методику написания программ для получения и обработки сигналов с датчиков через дискретный и аналоговый интерфейсы.
4) Выполнить задание своего варианта.
5) Ответить на контрольные вопросы.
6) Подготовить и защитить отчет по лабораторной работе.

Теория

Аппаратно-программный комплекс Arduino

Arduino – это электронный конструктор и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов. Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Устройство программируется через USB без использования программаторов.

Arduino позволяет компьютеру выйти за рамки виртуального мира в физический и взаимодействовать с ним. Устройства на базе Arduino могут получать информацию об окружающей среде посредством различных датчиков, а также могут управлять различными исполнительными устройствами.

Микроконтроллер на плате программируется при помощи языка Arduino (основан на языке Wiring (C/C++ с дополнительными библиотеками)) и среды разработки Arduino (основана на среде Processing). Проекты устройств, основанные на Arduino, могут работать самостоятельно, либо же взаимодействовать с программным обеспечением на компьютере. Платы могут быть собраны пользователем самостоятельно или куплены в сборе. Программное обеспечение доступно для бесплатного скачивания. Исходные чертежи схем (файлы CAD) являются общедоступными, пользователи могут применять их по своему усмотрению [1].

Микроконтроллер

Микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи. Микроконтроллер отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами [2].

Плата Arduino Uno построена на микроконтроллере с ядром AVR Atmel ATmega328. Микроконтроллеры AVR имеют гарвардскую архитектуру (программа и данные находятся в разных адресных пространствах) и систему команд, близкую к идеологии RISC. Структура микроконтроллера с гарвардской архитектурой показана на рисунке 1.

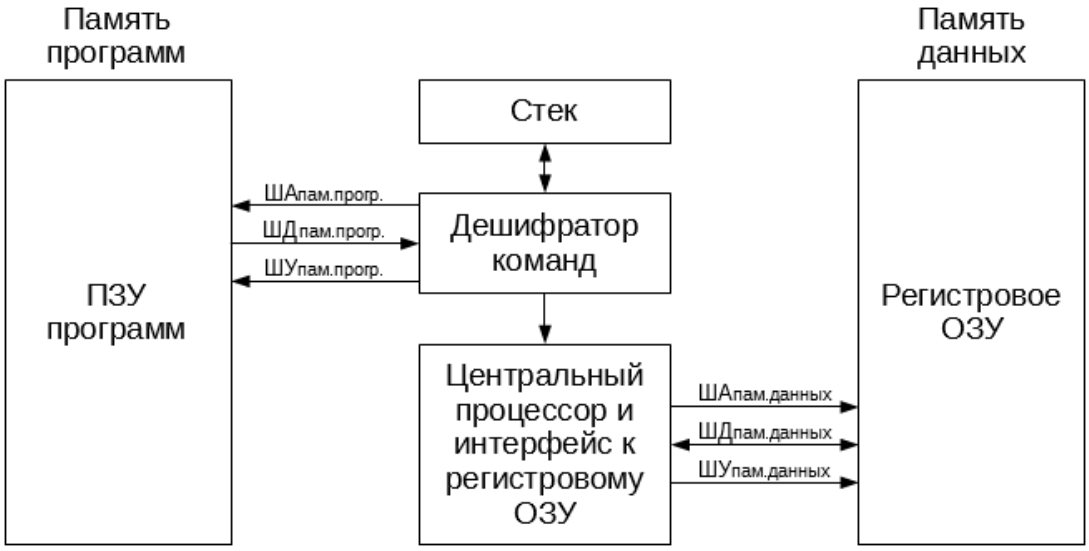


Рисунок 1 – Структура микроконтроллера с гарвардской архитектурой

Гарвардская архитектура обеспечивает потенциально более высокую скорость выполнения программы по сравнению с фон-неймановской за счет возможности реализации параллельных операций. Выборка следующей команды может происходить одновременно с выполнением предыдущей, и нет необходимости останавливать процессор на время выборки команды. Этот метод реализации операций позволяет обеспечивать выполнение различных команд за одинаковое число тактов, что дает возможность более просто определить время выполнения циклов и критичных участков программы [3].

Порты платы Arduino Uno

Для управления портами в их электрической схеме имеется два переключателя, которые можно переключать программно, используя специальные регистры ввода/вывода. Такие переключатели имеются для каждого вывода, что означает возможность управлять любым выводом порта. К примеру, один вывод порта можно настроить на ввод информации, три разряда этого же порта на вывод, а оставшиеся вообще не настраивать, оставить их в третьем логическом (высокоимпедансном) состоянии. Упрощенная схема порта ввода/вывода показана на рисунке 2.

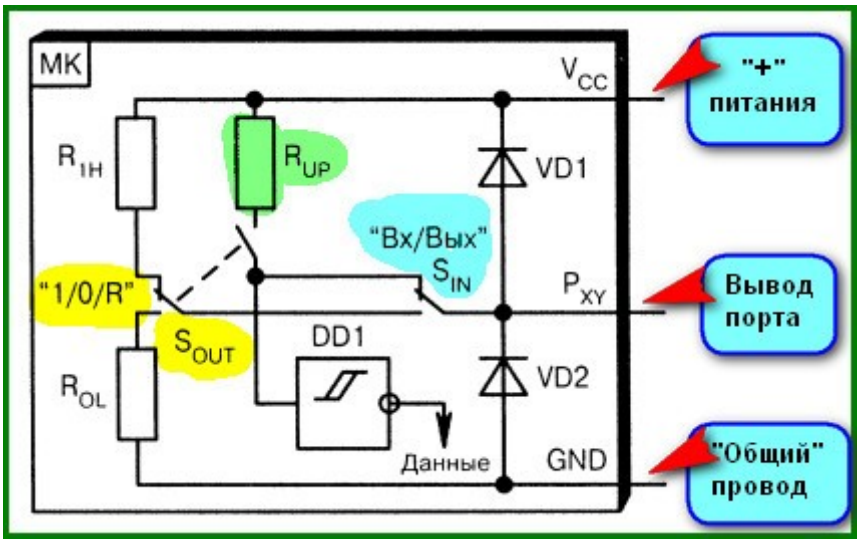


Рисунок 2 – Упрощенная схема порта ввода/вывода микроконтроллера

Обратите внимание на два переключателя – S_{in} и S_{out}, и сопротивление R_{up}. С помощью S_{in} осуществляется переключение вывода порта или для работы на вход, или для работы на выход. Управляется этот переключатель с помощью регистра ввода/вывода DDRx. У каждого порта свой регистр. Каждый разряд регистра управляет соответствующим разрядом порта (нулевой – нулевым, первый – первым и т.д.) При записи в разряд регистра DDRx лог. 1, соответствующий ему разряд порта переключается на вывод информации, а при записи лог. 0 – на ввод информации [4].

Установка и настройка среды Arduino

Скачать среду разработки Arduino можно по адресу https://www.arduino.cc/. После установки среды разработки следует её запустить и подключить плату Arduino, например Arduino Uno, к USB-порту компьютера. Если на плате Arduino установлен приёмопередатчик CH340G, то для него будет необходимо скачать соответствующий драйвер.

Далее необходимо настроить среду Arduino:

1) В пункте меню «Инструменты → Плата» выбрать нужный контроллер, например, «Arduino/Genuino Uno».
2) В пункте меню «Инструменты → Порт» выбрать виртуальный COM-порт, к которому подключён контроллер Arduino. Скорее всего, это будет COM3. Если в списке доступных портов помимо COM3 представлены и другие, то, вероятней всего, это будет порт с самым большим номером.
3) Выбрать пункт меню «Инструменты → Получить информацию о плате», и, если, будет показано окошко с информацией о плате, то всё сделано правильно.

Теперь можно подключать к плате Arduino датчики, и приступать к написанию программы. Проверить программу на присутствие синтаксических ошибок можно выбрав пункт меню «Скетч → Проверить/Компилировать». Если ошибок нет, то можно выполнить загрузку программы в плату, выбрав пункт меню «Скетч → Загрузка».

После успешной загрузки программы можно посмотреть результат её выполнения, при условии, что в программе предусмотрена передача данных с контроллера на компьютер. Для этого можно выбрать пункт меню «Инструменты → Монитор порта»,

тогда на экран будут выводиться данные в текстовом виде, либо «Инструменты → Плоттер по последовательному соединению», и тогда будут строиться графики временной развёртки сигналов.

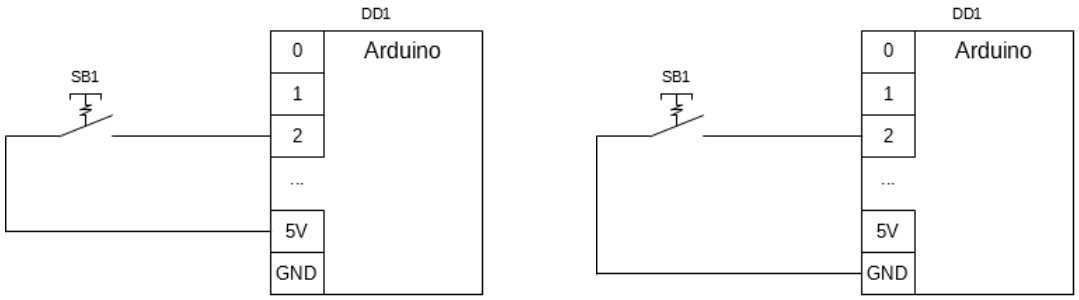
Ход работы

Если лабораторная работа выполняется не в аудитории, то её можно выполнить в симуляторе, например, Virtual Breadboard (http://www.virtualbreadboard.com/), Virtronics Simulator for Arduino (http://www.virtronics.com.au/), Autodesk 123D Circuits (https://circuits.io/), Fritzing (http://fritzing.org/home/) или любом другом.

Подключение кнопки или геркона через дискретный интерфейс

Порт микроконтроллера (Arduino базируется на микроконтроллере), настроенный на ввод, воспринимает лог. 0 при его подключении к «земле» **GND** и лог. 1 – при подключении к питанию **5V**.

Самый простой способ подключения кнопки представлен на рисунке 3, а соответствующий код для платы Arduino – в листинге 1.



а) при нажатии на кнопку фиксируется лог. 1;

б) при нажатии на кнопку фиксируется лог. 0.

Рисунок 3 – Самая простая схема подключения кнопки к Arduino (микроконтроллеру)

Листинг 1 – Простая программа считывания состояния кнопки

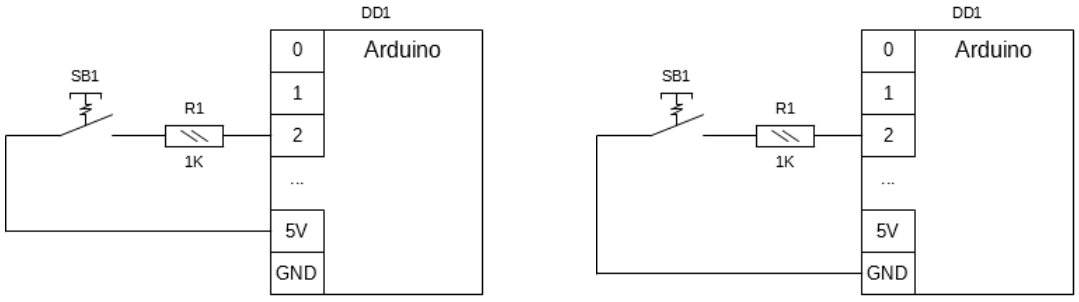
```
#define BTN_PIN 2 //Номер порта для подключения кнопки

bool btnState; //Состояние кнопки (не нажата/нажата)

void setup()
{
  // put your setup code here, to run once:
  pinMode(BTN_PIN, INPUT); //Настроить порт на ввод
  Serial.begin(9600); //Активировать последовательный порт
}

void loop()
{
  // put your main code here, to run repeatedly:
  btnState = digitalRead(BTN_PIN); //Считать состояние кнопки в переменную
  Serial.println(btnState); //Передать значение переменной
  delay(50); //Задержка программы 50 мс
}
```

После загрузки программы в плату Arduino можно посмотреть результат её выполнения с помощью монитора порта или плоттера по последовательному соединению. Что происходит? При нажатии на кнопку получаем стабильный сигнал, в зависимости от варианта подключения, – либо лог. 1, либо лог. 0. Но при отпускании кнопки сигнал может меняться хаотически, либо же не меняться совсем. Это связано с тем, что при размыкании кнопки провод, подключённый к порту ввода, «висит в воздухе», т.е. не подключён ни к питанию **5V**, ни к «земле» **GND**, и является, по сути, антенной.

Следует сказать, что такой способ подключения кнопки к Arduino (микроконтроллеру) не безопасен, и может привести в «выгоранию» порта. Представим себе ситуацию, что кнопка уже подключена, но программа в плату Arduino ещё не загружена. Тогда события могут развиваться по трём сценариям:

1) Если Arduino ещё никто и никогда не программировал, то по умолчанию, все порты настроены на ввод, и ничего страшного при нажатии кнопки не произойдёт, т.к. сопротивление порта в режиме ввода бесконечно большое.
2) Если порт, к которому подключена кнопка кто-то ранее запрограммировал на вывод лог. 0 (т.е. порт через внутренние схемы контроллера соединён с «землёй» **GND**), то при нажатии на кнопку, подключённую по схеме 2а, произойдёт «короткое замыкание» и через порт потечёт ток, превышающий номинальный. При этом порт может выйти из строя, ведь предельный для него ток не должен превышать 20 мА. Если же кнопка подключена по схеме 2б, то ничего страшного не произойдёт.
3) Наоборот, если порт, к которому подключена кнопка кто-то ранее запрограммировал на вывод лог. 1 (т.е. порт через внутренние схемы контроллера соединён с питанием **5V**), то при нажатии на кнопку, подключённую по схеме 2а, ничего страшного не произойдёт. Но произойдёт «короткое замыкание», если кнопка была подключена по схеме 2б.

Чтобы предотвратить возможное «короткое замыкание» нужно ограничить ток, поэтому последовательно с кнопкой устанавливают защитный резистор номиналом, например, 1 кОм (минимум 250 Ом). Тогда максимально возможный ток, протекающий через порт, при самых неблагоприятных условиях составит 5 мА, что не приведёт к повреждению порта. Схема подключения кнопки с токоограничивающим резистором представлена на рисунке 4.



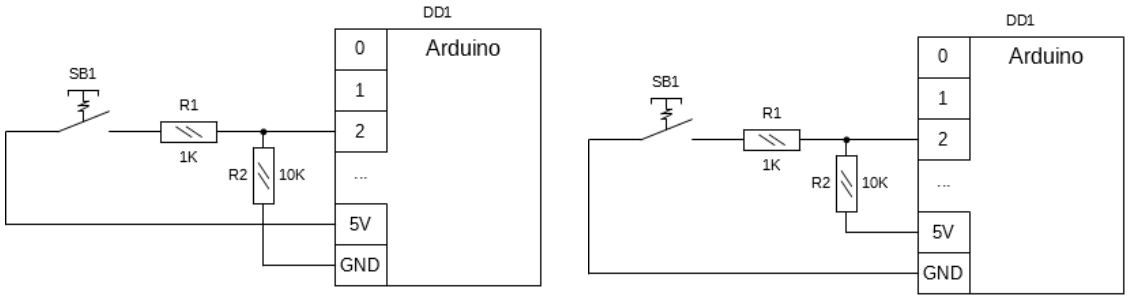
а) при нажатии на кнопку фиксируется лог. 1;

б) при нажатии на кнопку фиксируется лог. 0.

Рисунок 4 – Самая простая и безопасная схема подключения кнопки к Arduino (микроконтроллеру)

Подключив кнопку согласно данной схеме, можно посмотреть результат. И он окажется таким же, как и в первом случае, т.е. получить стабильно работающую кнопку не получится.

Для того чтобы кнопка работала правильно, нужно подключить «подтягивающий резистор» к порту ввода. Подтягивающий резистор может подключаться как к питанию **5V** (наиболее распространённый вариант), так и к «земле» **GND** (рисунок 5). Номинал подтягивающего резистора обычно выбирается из диапазона от 10 до 100 кОм.



а) при нажатии на кнопку фиксируется лог. 1;

б) при нажатии на кнопку фиксируется лог. 0.

Рисунок 5 – Схема подключения кнопки к Arduino (микроконтроллеру) с внешним подтягивающим резистором

Собрать такую схему можно на макетной плате, как показано на рисунке 6.

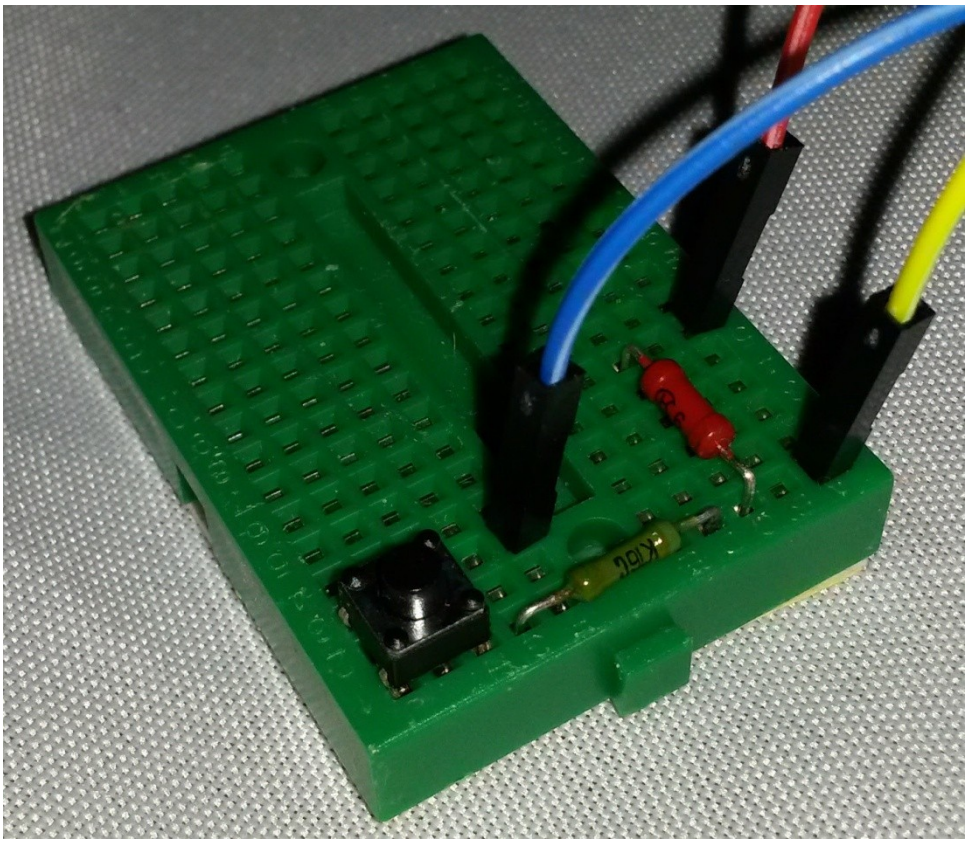


Рисунок 6 – Собранная схема подключения кнопки с токоограничивающим и внешним подтягивающим резистором на макетной плате

Теперь кнопка заработала правильно. Аналогичным образом подключается и геркон. Но нужно иметь в виду, что существует такое явление как «дребезг контактов», с которым борются программными или аппаратными методами.

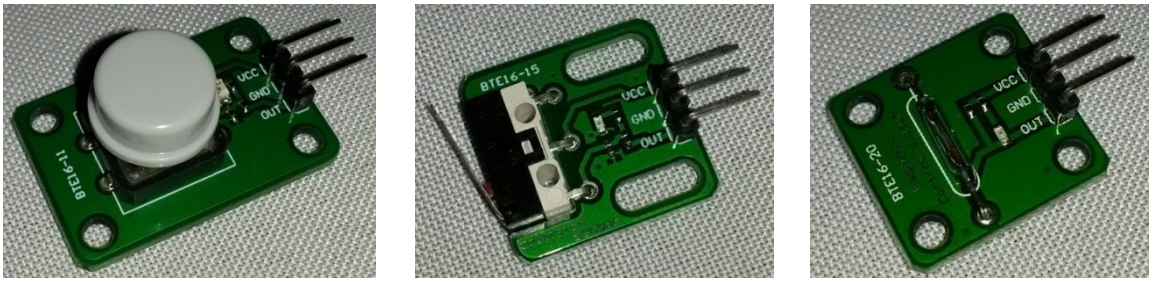
Следует заметить, что можно обойтись и без подключения внешнего подтягивающего резистора, т.к. его можно подключить программно. Тогда подключение кнопки выполняется согласно схеме, представленной на рисунке 4, а в программу из листинга 1 добавляется одна строка. Программа опроса кнопки представлена в листинге 2.

Листинг 2 – Программа считывания состояния кнопки с подключением внутреннего подтягивающего резистора

```
#define BTN_PIN 2 //Номер порта для подключения кнопки

bool btnState; //Состояние кнопки (не нажата/нажата)

void setup()
{
  // put your setup code here, to run once:
  pinMode(BTN_PIN, INPUT); //Настроить порт на ввод
  digitalWrite(BTN_PIN, HIGH); //Подключить внутренний подтягивающий резистор
  Serial.begin(9600); //Активировать последовательный порт
}

void loop()
{
  // put your main code here, to run repeatedly:
  btnState = digitalRead(BTN_PIN); //Считать состояние кнопки в переменную
  Serial.println(btnState); //Передать значение переменной
  delay(50); //Задержка программы 50 мс
}
```

Подключение кнопки с внешним питанием через дискретный интерфейс

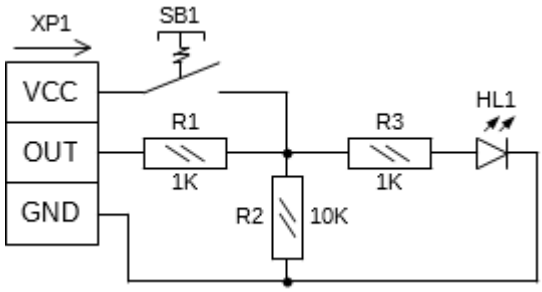
Иногда может возникать необходимость (например, для отладки) подключения кнопок, концевых контактов или герконов (рисунок 7), смонтированных на печатной плате, где уже установлены подтягивающие и токоограничивающие резисторы. Помимо простых кнопок встречаются



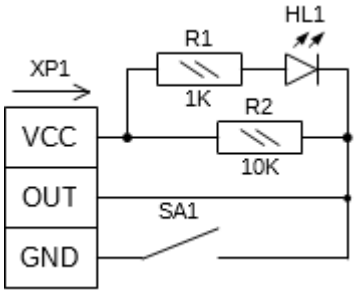
а) при нажатии на кнопку фиксируется лог. 1; б) при замыкании концевого контакта фиксируется лог. 1; в) при срабатывании геркона фиксируется лог. 0.

Рисунок 7 – Варианты исполнения печатных плат с устройствами коммутации и установленными подтягивающими и токоограничивающими резисторами

Электрические принципиальные схемы, представленных печатных плат с устройствами коммутации, приведены на рисунке 8.



а) электрическая принципиальная схема платы с кнопкой или концевым контактом;



б) электрическая принципиальная схема платы с герконом.

Рисунок 8 – Электрические принципиальные схемы для печатных плат с кнопкой, концевым контактом и герконом

Подключение подобных печатных плат с устройствами коммутации к Arduino не составляет никакого труда. Программу для опроса состояния датчиков можно использовать из листинга 1.

Подключение энкодера к дискретным портам ввода

Энкодер является механическим датчиком угла поворота, и преобразует угол поворота вращающегося объекта (например, вала) в электрические сигналы, сдвинутые по фазе на 90° относительно друг друга. На рисунке 9 показан внешний вид энкодера на печатной плате с токоограничивающими резисторами.

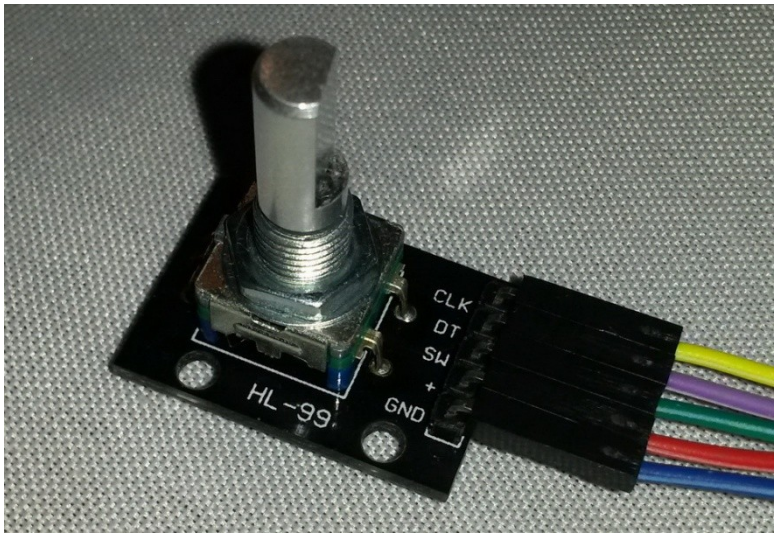


Рисунок 9 – Внешний вид печатной платы с энкодером и токоограничивающими резисторами

Данный модуль имеет пять выводов: три сигнальных – **`CLK`**, **`DT`**, **`SW`**, и два питания – питание «**+**» и «земля» **`GND`**. С энкодера считываются два сигнала (A и B), которые противоположны по фазе, с линий **`CLK`** и **`DT`**. Данный энкодер имеет 20 положений на один оборот (каждое положение 18°). На рисунке 10 показано, как зависят выходы A (**`CLK`**) и B (**`DT`**) друг от друга при вращении энкодера по часовой или против часовой стрелки.

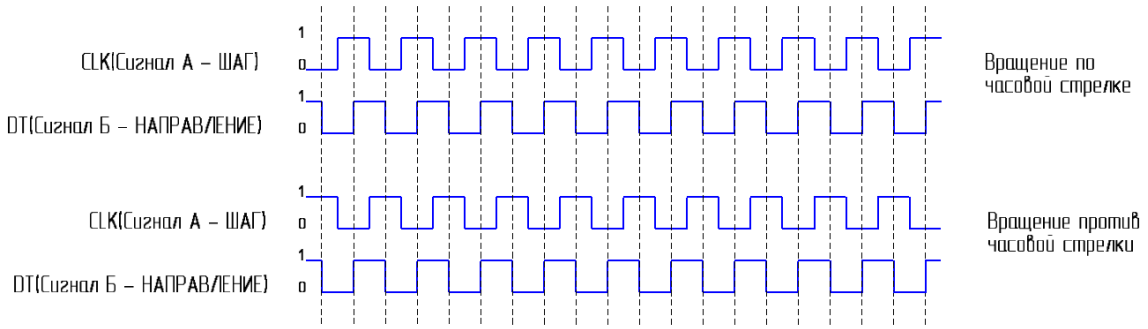


Рисунок 10 – Зависимость выходов A (CLK) и B (DT) друг от друга при вращении энкодера по часовой или против часовой стрелки

Вывод энкодера **SW** используется для получения состояния центральной оси энкодера, которая работает как кнопка.

Энкодер подключается к Arduino по схеме, представленной на рисунке 11. Программа представлена в листинге 3.

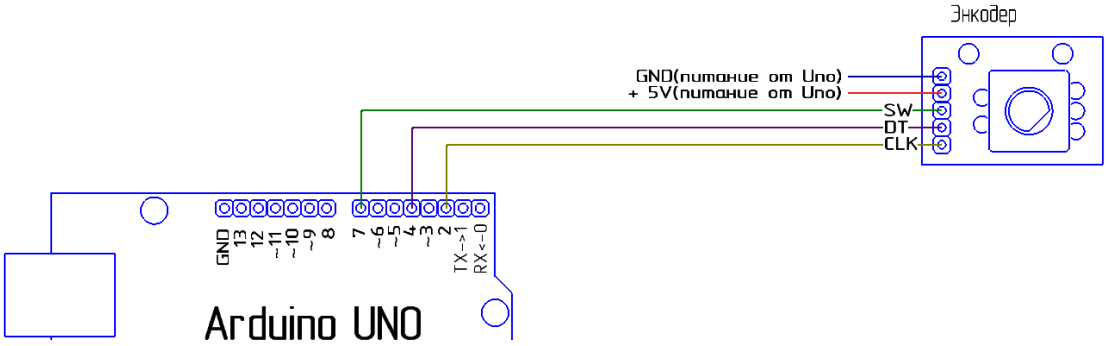


Рисунок 11 – Подключение энкодера на печатной плате к Arduino Uno

Листинг 3 – Программа считывания состояния энкодера

```
#define SW_PIN 7 //Кнопка SW
#define CLK_A_PIN 2 //Сигнал CLK (A)
#define DAT_B_PIN 4 //Сигнал DT (B)

int swState = 0;
int encPos = 0;
bool clkAPre = HIGH;
bool clkACur = LOW;

void setup()
{
  // put your setup code here, to run once:
  pinMode(SW_PIN, INPUT);
  digitalWrite(SW_PIN, HIGH);
  pinMode(CLK_A_PIN, INPUT);
  pinMode(DAT_B_PIN, INPUT);
  Serial.begin(9600);
}

void loop()
{
  // put your main code here, to run repeatedly:
  if (digitalRead(SW_PIN) == LOW)
  {
    while (digitalRead(SW_PIN) == LOW)
```

```
    {
      //Ждём отпускания кнопки
    }
    swState++;
    Serial.print("BtnClick = ");
    Serial.println(swState);
  }

  clkACur = digitalRead(CLK_A_PIN);
  if ((clkAPre == LOW) && (clkACur == HIGH))
  {
    if (digitalRead(DAT_B_PIN) == LOW)
    {
      encPos++;
    } else
    {
      encPos--;
    }
    Serial.print("EncoderPos = ");
    Serial.println(encPos);
  }
  clkAPre = clkACur;
}
```

Подключение кнопок через аналоговый интерфейс

Здесь вскоре может появиться текст.

Подключение джойстика через аналоговый интерфейс

Для плат Arduino существуют модули аналоговых джойстиков. Джойстик, как правило, имеет две оси – X и Y, а также кнопку. Кнопка подключается к дискретному интерфейсу ввода. Джойстик позволяет плавно отслеживать степень отклонения от нулевой точки. По направлениям отклонения VRX (X) и VRY (Y), на плате джойстика установлены два потенциометра. Через аналоговый интерфейс ввода **`A0`** и **`A1`** Arduino получает сигналы, находящиеся в диапазоне от 0 до 1023 (10-битный АЦП), тем самым отслеживая отклонения джойстика влево, вправо, вверх и вниз. Внешний вид джойстика на печатной плате представлен на рисунке 12.

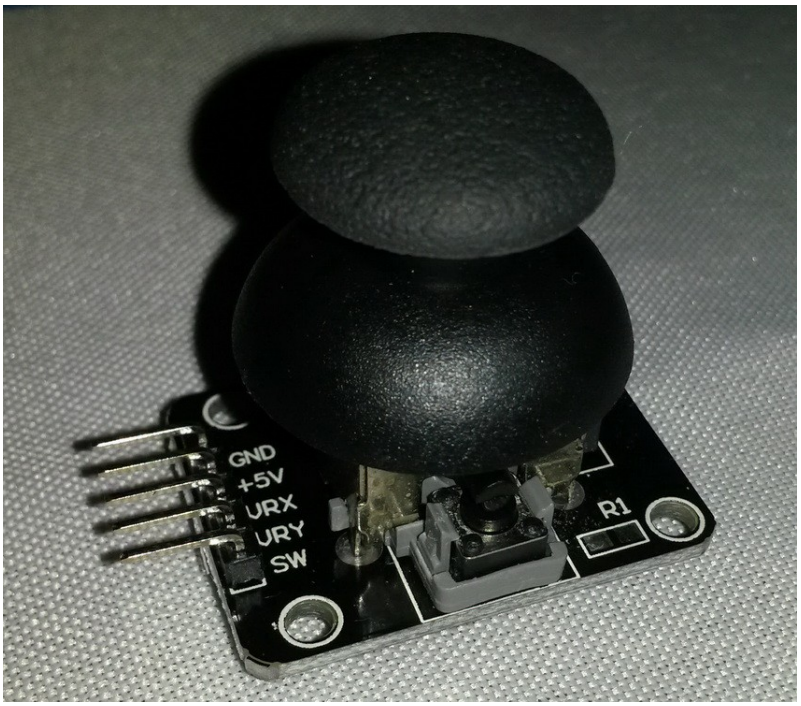


Рисунок 12 – Внешний вид двухосевого джойстика на печатной плате

Схема подключения джойстика к Arduino представлена на рисунке 13, а программа – в листинге 4.

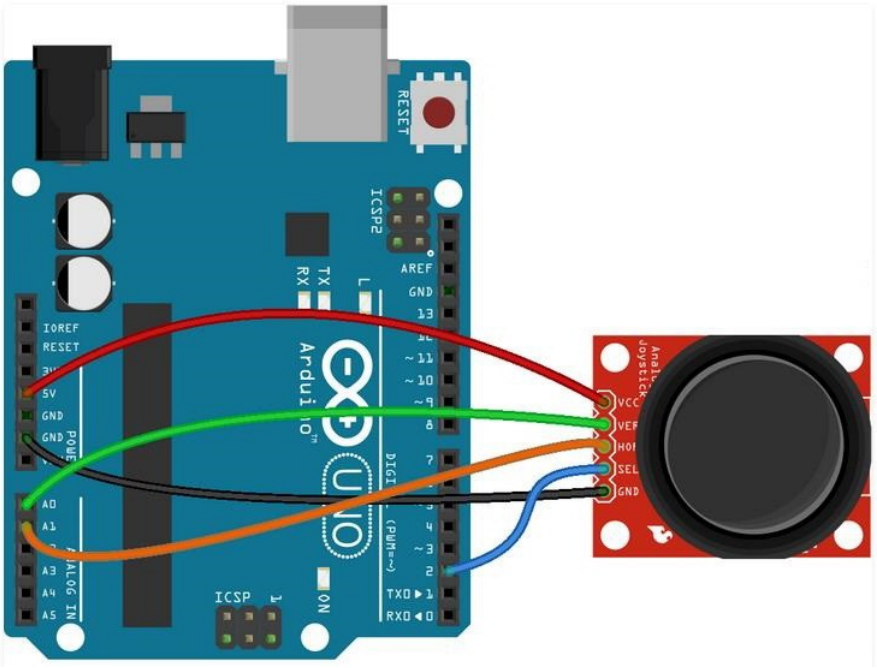


Рисунок 13 – Подключение джойстика на печатной плате к Arduino Uno

Листинг 4 – Программа считывания состояния джойстика

```
#define AX A0 //Ось X
#define AY A1 //Ось Y
#define BTN 2 //Кнопка

void setup()
{
  // put your setup code here, to run once:
  pinMode(BTN, INPUT); //Настроить порт на ввод
```

```
  digitalWrite(BTN, HIGH); //Подключить внутренний подтягивающий резистор
  Serial.begin(9600);
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.print(analogRead(AX));           //Такой способ вывода результатов
  Serial.print(" ");                      //используется для корректной
  Serial.print(analogRead(AY));           //работы «Плоттера по
  Serial.print(" ");                      //последовательному соединению»,
  Serial.println(digitalRead(BTN) * 500); //т.е. будут строится 3 графика
}
```

Задания по вариантам

1) В лабораторной работе был рассмотрен способ получения информации о состоянии кнопки по времени, т.е. через каждые 50 мс. Напишите программу, где бы состояние кнопки считывалось по событию, т.е. информация на выходе появляется только тогда, когда кнопка нажата или отпущена.
2) Напишите программу для Arduino, где кнопка после нажатия и последующего отпускания переключает выходной сигнал.
3) Подключите 4 кнопки (например, сенсорные) к Arduino и напишите программу, где при наборе последовательности цифр 1234 к переменной добавляется 1, а последовательности 4321 отнимается 1, результат выводится на экран.
4) Подключите 4 кнопки (например, сенсорные) к Arduino и напишите программу, где при наборе правильной последовательности выводится результат.
5) Подключите 4 кнопки (например, сенсорные) к Arduino и напишите программу, где при вводе простого числа (от 1 до 100) выводится лог. 1, а иначе – лог. 0.
6) Напишите программу для Arduino, где поворотом энкодера устанавливается первое слагаемое, после нажатия на кнопку устанавливается второе слагаемое, после нажатия – сумма.
7) Напишите программу для Arduino, где поворотом энкодера устанавливается уменьшаемое, после нажатия на кнопку устанавливается вычитаемое, после нажатия – разность.
8) Напишите программу для Arduino, где поворотом энкодера устанавливается первый множитель, после нажатия на кнопку устанавливается второй множитель, после нажатия – произведение.
9) Напишите программу для Arduino, где поворотом энкодера устанавливается делимое, после нажатия на кнопку устанавливается делитель, после нажатия – частное.
10) Напишите программу для Arduino, где после оборота джойстика по часовой стрелке на выходе формируется логическая единица.
11) Напишите программу для Arduino, где после оборота джойстика против часовой стрелке на выходе формируется логическая единица.

Требования к содержанию отчета

Отчет по лабораторной работе должен содержать следующие пункты:

1) Цель и задачи лабораторной работы.
2) Краткие теоретические сведения.
3) Подключение кнопки/геркона через дискретный интерфейс.
4) Подключение кнопки с внешним питанием через дискретный интерфейс.
5) Подключение энкодера к дискретным портам ввода.
6) Подключение кнопок через аналоговый интерфейс.
7) Подключение джойстика через аналоговый интерфейс.
8) Задание согласно своему варианту.
9) Ответы на контрольные вопросы.
10) Вывод по лабораторной работе.

Контрольные вопросы

1) Если задержка программы равна 50 мс, то с какой частотой будет опрашиваться кнопка. А что, если события (нажатия на кнопку) будут происходить с большей частотой?
2) Что такое «дребезг контактов»? Почему это явление возникает? Как с ним борются?
3) Что будет, если подключить и внешний и внутренний подтягивающий резистор одновременно для считывания состояния кнопки?
4) Каковы недостатки самого простого способа подключения кнопки? Что нужно сделать, чтобы кнопка заработала правильно?

Список использованных источников

1) Аппаратная платформа Arduino [Электронный ресурс]. URL: http://arduino.ru/ (дата обращения: 01.09.2017)
2) Исследование универсального USB-программатора - Современные наукоемкие технологии (научный журнал) [Электронный ресурс]. URL: https://www.top-technologies.ru/ru/article/view?id=33695 (дата обращения: 01.09.2017)
3) НОУ ИНТУИТ. Лекция. Процессорное ядро и память микроконтроллеров [Электронный ресурс]. URL: http://www.intuit.ru/studies/courses/3/3/lecture/72?page=2 (дата обращения: 01.09.2017)
4) Порты ввода-вывода микроконтроллера [Электронный ресурс]. URL: http://radio-stv.ru/mikrokontrolleri/ustroystvo-i-programmirovanie-mikrokontrollerov-dlya-nachinayushhih/portyi-vvoda-vyivoda-mikrokontrollera (дата обращения: 01.09.2017)