

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра «Автоматизация производственных процессов»

ОСНОВЫ МЕХАТРОНИКИ

Методические указания к комплексу лабораторных работ

по дисциплине «Основы мехатроники»

для студентов направлений подготовки

15.03.04 «Автоматизация технологических процессов и производств»

(профиль «Автоматизация технологических процессов и производств
в машиностроении»),

27.03.04 Управление в технических системах (профиль «Системы и
технические средства автоматизации и управления»)

Курган 2018

Кафедра: «Автоматизация производственных процессов»

Дисциплина: «Основы мехатроники»

Составил: канд.техн.наук, доцент Е.К. Карпов

Утверждены на заседании кафедры

21 декабря 2017 г.

Рекомендованы методическим советом университета

12 декабря 2016 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Обзор и тестирование различных типов датчиков, применяемых в мехатронике.....	4
2 Управление пневматическим и электрическим двигателем при помощи микроконтроллера.....	11
3 Последовательный порт, параллельный, UART, передача данных с компьютера на микроконтроллер.....	15
4 Применение циклов, условий, функций и классов при программировании мехатронного устройства.....	18
5 Программный интерфейс и создание собственных библиотек.....	20
6 Диагностика неисправностей микроконтроллерной системы управления.....	21
7 Примеры подключения электронных элементов к микроконтроллеру.....	22

ВВЕДЕНИЕ

Целью освоения дисциплины «Основы мехатроники» является приобретение студентами знаний о содержании, определениях и методах применения мехатроники, мехатронных узлах, компонентном составе мехатронных устройств и особенностях их проектирования, формирование навыков проектирования простых мехатронных устройств на базе микроконтроллеров, их программирования и отладки.

Выполнение заданий данных методических указаний, позволит узнать основные понятия и определения мехатроники, классификацию и характеристики мехатронных устройств, сформировать для себя понятия о структуре мехатронных узлов, изучить принципы действия элементов исполнительной, управляющей и информационной подсистем мехатронных устройств и области их применения.

Перед выполнением лабораторных работ студентам рекомендуется ознакомиться с требованиями, предъявляемым к оформлению отчётов, и справочными материалами, представленными в 8 пункте данных методических указаний.

1 Обзор и тестирование различных типов датчиков, применяемых в мехатронике

Для выполнения проектов первого занятия необходимо понимать структуру представления программ, их основные элементы и особенности языка программирования, на котором они пишутся. Далее дано краткое описание всех команд, которые будут использоваться в первой лабораторной работе для работы с аналоговыми и дискретными датчиками и простым исполнительным устройством – сервоприводом.

Структура программы и базовые функции `setup()` и `loop()`

Любая программа, написанная Вами, должна включать в себя две функции и иметь следующий вид:

```
void setup()
{
  //код программы, выполняемый один раз при включении
}
void loop()
{
  //код, выполняемый постоянно, представляющий собой основную часть
}
```

В функции `setup()` обычно определяются режим работы портов и установка соединения по последовательному порту. В `loop()` записываются все операции чтения и записи данных с портов, математические и логические операции, вызовы других функций и прочие операции работы микроконтроллера, необходимые для выполнения поставленной задачи.

Фигурные скобки {} определяют начало и конец тела функции или блока выражений. На каждую открывающую фигурную скобку в программе должна быть закрывающая скобка.

Создание новых переменных и их типы

Переменные предназначены для хранения значений различных типов и их использования в ходе работы программы. Их типы, способы определения и границы видимости в целом аналогичны изученным на дисциплине “Основы программирования и алгоритмизации”. Стоит отметить, что номера контактов контроллера в больших программах обычно определяются глобальными переменными перед функцией setup(). В таком случае упрощается их перенастройка при замене одних контактов на другие. Пример создания и присваивания переменных:

```
int outPin; // объявление переменной целочисленного типа
outPin = 10; // и присваивание ей значения
float pi = 3.14; // объявление и присваивание – с плавающей точкой
```

В конце каждого выражения и для разделения элементов программ применяется точка с запятой – “;”. Однострочные комментарии начинаются с //.

Определение используемых входов и выходов микроконтроллера

Для того чтобы записать или считать информацию с какого-либо контакта микроконтроллера, необходимо предварительно его определить в функции setup():

```
void setup()
{
  pinMode(12, INPUT); // 12 контакт определяется как дискретный вход
  pinMode(outPin, OUTPUT); // 10 контакт определяется как выход
}
```

Цифровое чтение и цифровая запись сигналов

Функция digitalRead(inputPin); позволяет считать дискретный сигнал с контакта inputPin и получить значение HIGH или LOW (высокий или низкий логический уровень соответственно). Функция digitalWrite(outPin, HIGH); записывает на дискретный выход outPin логический сигнал, который может задаваться из переменной или константой. Используя эти команды можно получать состояния дискретных датчиков (например, кнопок), производить их программный анализ и выводить некоторую информацию на выходы контроллера, к которым подключены дискретные устройства (светодиоды, реле).

Аналоговое чтение и аналоговая запись сигналов

Считывание сигналов с аналоговых входов производится с помощью команды analogRead(A0). В качестве входа могут быть указаны контакты микроконтроллера с A0 до A5, причём считанный сигнал будет с 10-битовым

разрешением (в соответствии с разрядностью аналого-цифрового преобразователя) и будет находиться в диапазоне от 0 до 1023.

При помощи широтно-импульсной модуляции реализуется аналоговый вывод с разрядностью в 8 бит (от 0 до 255). Контакты, которые им оборудованы, обозначены на плате символом “~”. Пример чтения сигнала с А3, его масштабирования и вывода на контакт с ШИМ:

```
int a = analogRead(A3) / 4;  
analogWrite(9, a);
```

Функция задержки

Функция `delay(1000)`; приостанавливает выполнение программы на заданное в миллисекундах время – в данном случае на одну секунду.

Конструкция if, if-else

Данные конструкции предназначены для выполнения некоторого выражения, заключённого в фигурные скобки, в том случае, если соблюдается проверяемое условие. Например:

```
if (a != b) // если a не равно b  
{  
  a = b; // присвоить a значение b  
}  
else // иначе  
{  
  a = 0; // присвоить a  
  b = 0; // и b нулю  
}
```

Вторая часть конструкции `else`, выполняемая в случае не соблюдения условия в скобках после `if`, может быть пропущена, если нет необходимости в альтернативном действии.

В скобках после `if` могут быть использованы следующие операторы сравнения:

```
x == y    // x равно y  
x != y    // x не равно y  
x < y     // x меньше y  
x > y     // x больше y  
x <= y    // x меньше или равно y  
x >= y    // x больше или равно y
```

Для записи нескольких условий, которые должны проверяться одновременно, могут быть использованы логические операторы:

`&&` – логическое “И” – истинно только в том случае, если оба условия выполняются, например:

```
if (x>0 && x<5) // если x больше нуля и меньше пяти
```

`||` – логическое “ИЛИ” – истинно в случае, когда выполняется хотя бы одно из условий:

```
if (x > 0 || x < 0) // истинно, если x не равен нулю
```

Подключение библиотек и работа с ними

Существует множество готовых решений, реализованных как отдельные файлы, содержащие определение внутренних переменных и функций а также внешних функций, через которые осуществляется работа с ними. Их применение значительно упрощает написание программ и работу с отдельным оборудованием, сводя его к двум-трём строчкам кода. В ходе выполнения лабораторных работ вы познакомитесь с несколькими стандартными библиотеками. Подключение библиотек осуществляется при помощи записи вне функций программы следующей конструкции:

```
#include <название_библиотеки.h>
```

Конструкции для работы с конкретными библиотеками сугубо индивидуальны и должны изучаться отдельно при ознакомлении с её примерами или справочными файлами.

Соберите приведённые ниже схемы, напишите программы и проверьте их работоспособность. В качестве подсказок используйте Справочные материалы, находящиеся в конце этого пособия.

Электрическая схема с кнопкой и диодом

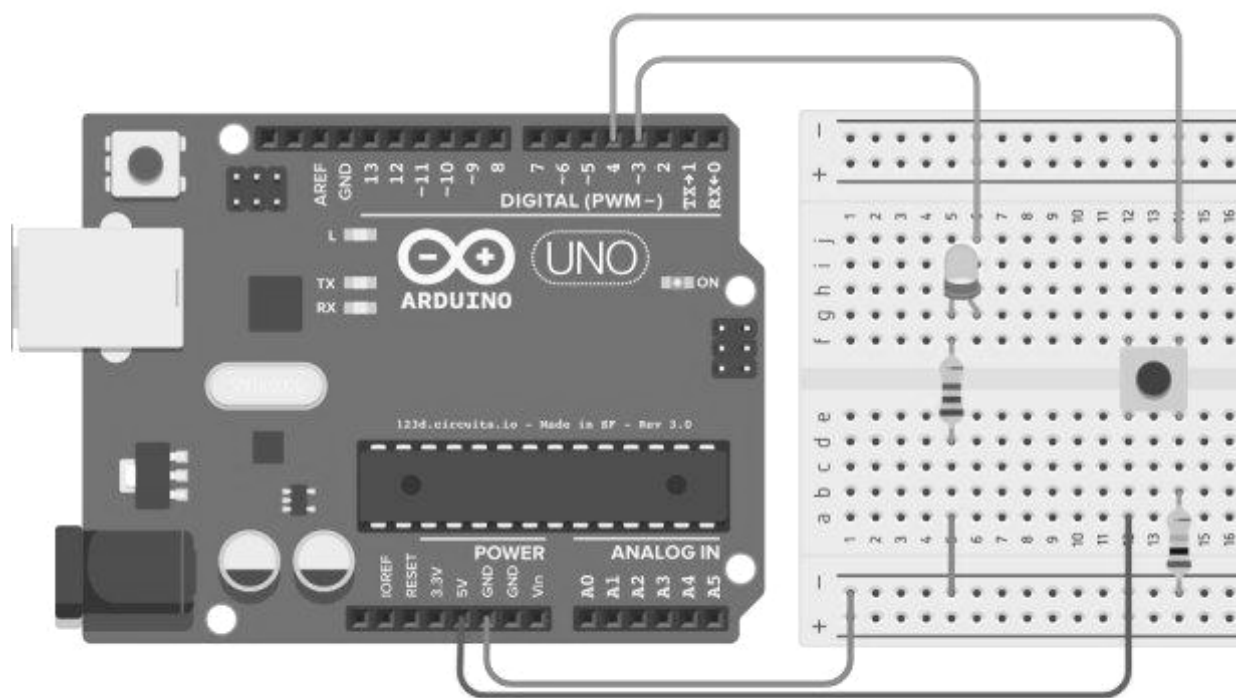


Рисунок 1 – Схема с одной кнопкой и одним диодом

Мигание диодом по нажатию кнопки. Текст программы:

```
void setup() {  
  pinMode(3, OUTPUT); // Третий контакт контроллера – выход (диод)  
  pinMode(4, INPUT); // Четвёртый контакт контроллера – вход (кнопка)
```

```
}
```

```
void loop() {  
  int a = digitalRead(4); // Чтение сигнала с кнопки в переменную  
  digitalWrite(3,a); // Подача значения переменной на диод  
}
```

Два диода и две кнопки. Попробуйте сами собрать электрическую схему:

```
void setup() {  
  pinMode(3, OUTPUT); // Третий контакт контроллера – выход (диод 1)  
  pinMode(5, OUTPUT); // Пятый контакт контроллера – выход (диод 2)  
  pinMode(4, INPUT); // Четвёртый контакт контроллера – вход (кнопка 1)  
  pinMode(6, INPUT); // Шестой контакт контроллера – вход (кнопка 2)  
}
```

```
void loop() {  
  int a = digitalRead(4); // Чтение сигнала с кнопки 1 в переменную a  
  digitalWrite(3,a); // Подача значения переменной на диод 1  
  int b = digitalRead(6); // Чтение сигнала с кнопки 2 в переменную b  
  digitalWrite(5,b); // Подача значения переменной на диод 2  
}
```

Электрическая схема с фоторезистором и диодом

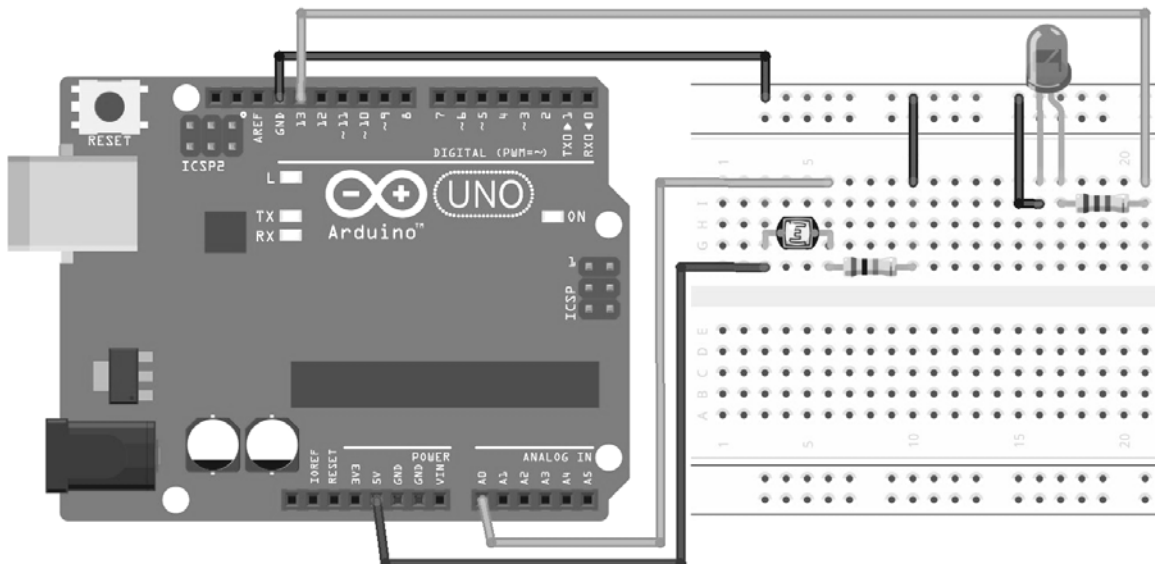


Рисунок 2 – Схема с фоторезистором и одним диодом

Включение диода в темноте и выключение на свету. Текст программы

```
void setup() {  
  pinMode(13, OUTPUT); // тринадцатый контакт – выход (диод)  
}
```



```

void loop() {
  //Если степень освещённости низкая (чтение аналогового порта A0 и
  //сравнение с константой),
  if (analogRead(A0) < 250)
    digitalWrite(13, HIGH); //то включается светодиод

  //Иначе
  else
    digitalWrite(13, LOW); //выключается светодиод
}

```

Потенциометр (переменный резистор) и диод. Электрическая схема

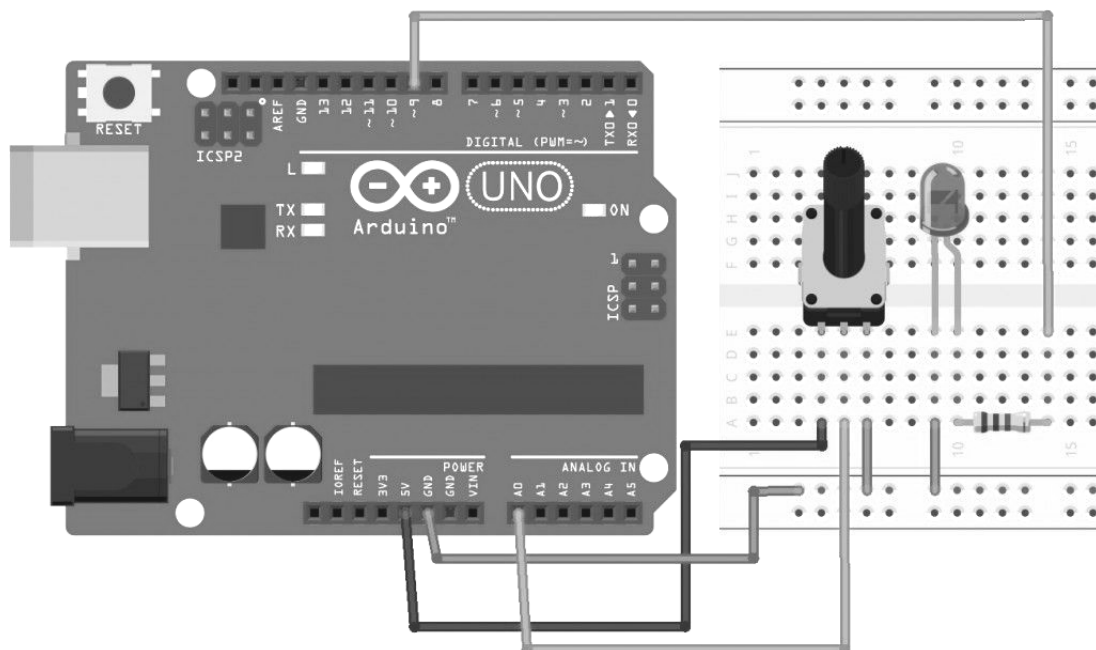


Рисунок 3 – Схема потенциометром и одним диодом

Управление яркостью светодиода от потенциометра. Программа

```

void setup() {
  pinMode(9, OUTPUT); // девятый контакт – выход (диод)
}

void loop() {
  //Чтение напряжения с потенциометра на порте A0 и переход от
  //10-битного значения АЦП (от 0 до 1023) к 8-битному ЦАП
  //(ШИМ от 0 до 255)
  int x = analogRead(A0) / 4;
  analogWrite(9, x); //Запись значения переменной на выход
  delay(50); //Задержка в 50 миллисекунд между выполнениями loop()
}

```

Схема подключения сервопривода к микроконтроллеру

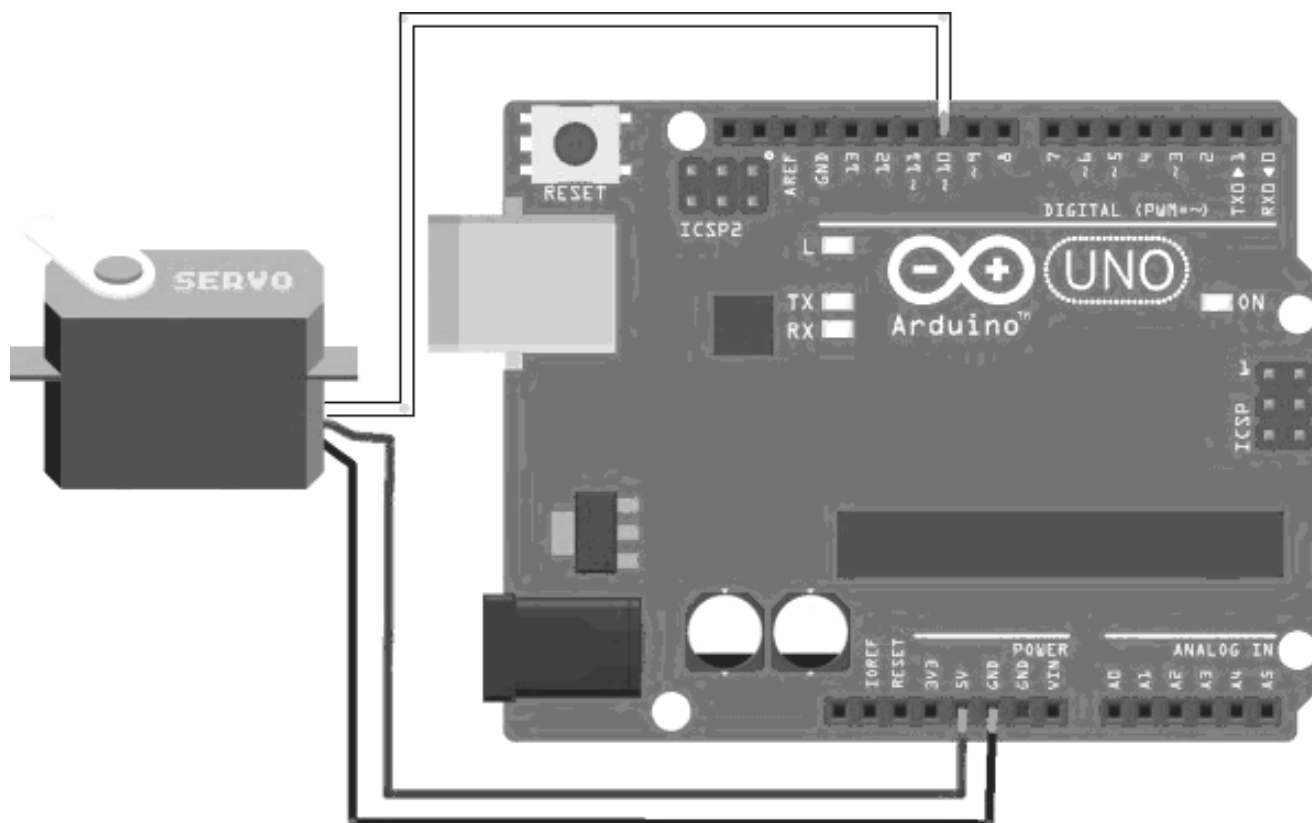


Рисунок 4 – Схема подключения сервопривода

Поворот сервопривода на угол 0 градусов и 180 градусов через каждые 2 секунды

```
#include <Servo.h> //Подключение библиотеки сервоприводов
Servo ServoA; //Создание объекта – сервопривода

void setup()
{
  ServoA.attach(10); //Определение контакта подключения сервопривода
}

void loop()
{
  ServoA.write(0); //Поворот сервопривода на нулевой угол
  delay(2000); //Двухсекундная пауза
  ServoA.write(180); //Поворот сервопривода на 180 градусов
  delay(2000); //Двухсекундная пауза
}
```

Выполните **здания** и сохраните необходимые материалы для отчёта по лабораторным работам – названия схем, которые надо добавить в отчёт, написаны в Справочных материалах в конце этого пособия:

- 1) соберите электрическую схему и напишите программу управления светодиодом по нажатию кнопки;
- 2) соберите схему управления с двумя кнопками и двумя светодиодами;
- 3) соберите схему с фоторезистором и светодиодом – светодиод должен загораться в том случае, если фоторезистор регистрирует низкий уровень освещения;
- 4) соберите схему с потенциометром и светодиодом – управляйте яркостью свечения светодиода, в зависимости от угла поворота потенциометра;
- 5) подключите к микроконтроллеру сервопривод и напишите программу, изменяющую его угол поворота от 0 градусов до 180 и обратно каждые две секунды;
- 6) соберите схему мехатронного устройства с фоторезистором сервоприводом и светодиодом. Если фоторезистор регистрирует низкий уровень освещения, то должен загораться светодиод, в противном случае светодиод должен выключаться, а сервопривод – придать движение. Объясните, какие части полученной модели соответствуют компонентам мехатронного устройства.

2 Управление пневматическим и электрическим двигателем при помощи микроконтроллера

Напрямую от схемы микроконтроллера можно управлять только незначительной нагрузкой, такой, как например светодиоды. Для использования в схемах силовых элементов, характерных для мехатронных устройств, – электроприводов постоянного тока, сервоприводов с напряжением питания более 5В, приводами пневматических клапанов, необходимо использовать силовые ключи, построенные на транзисторах или модулях реле.

Реле подключаются к микроконтроллеру по трёхпроводной схеме: питание, земля, сигнал. Такая схема позволяет снизить нагрузку на управляющие выходы контроллера за счёт отдельной 5В-линии. На силовые линии такого реле подключается нагрузка таким образом, чтобы оно прерывало один из проводов питания. При подаче управляющего сигнала с контроллера, будет происходить замыкание линии питания электропривода или другого исполнительного устройства. В случае управления пневматическими клапанами с электрическими приводами, применение реле позволяет осуществлять программное управление мехатронными устройствами с пневматической силовой составляющей.

Также реле используются для согласования напряжения логических уровней в том случае, когда у управляющих, исполнительных и регистрирующих устройств они различаются.

Далее приведены задания, последовательное построение которых позволит реализовать простое мехатронное устройство управления электроприводом с помощью транзистора с возможностью регулирования его скорости вращения посредством потенциометра. Кнопка, выполняющая функции аварийного останова по прерыванию, и монитор позволяют

расширить функционал устройства и предоставить пользователю информацию о его состоянии.

Соберите схемы второго занятия, напишите программы и проверьте их работоспособность. В качестве подсказок используйте Справочные материалы, находящиеся в конце этого пособия.

Управление скоростью мотора с помощью потенциометра

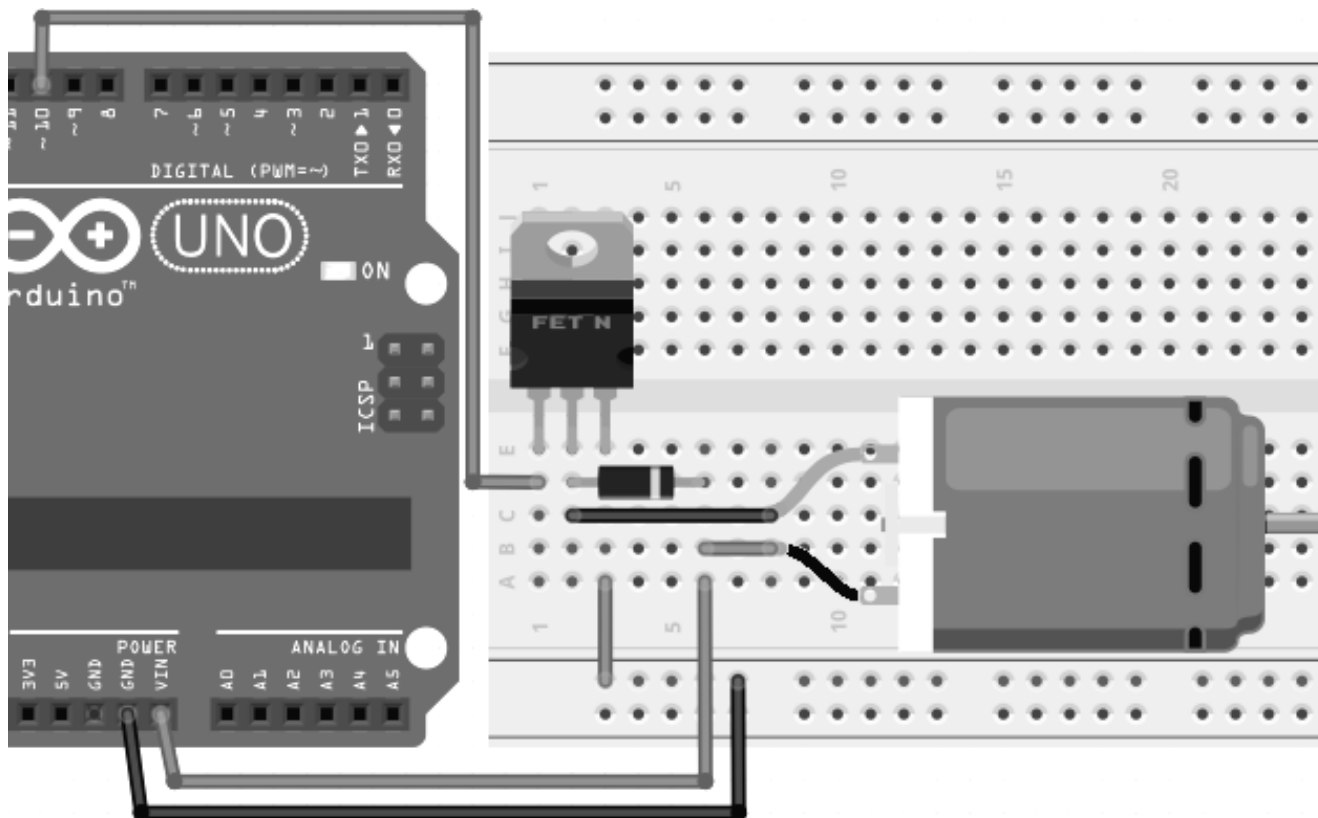


Рисунок 5 – Схема подключения микромотора

Текст программы управления скоростью мотора с помощью потенциометра

```
void setup()
{
  pinMode(10, OUTPUT); //Контакт подключения микромотора
}

void loop()
{
  int val = analogRead(A0) / 4; //Читаем значение потенциометра с A0 в
                                //переменную и согласуем её для АЦП-ЦАП
  analogWrite(10, val); //Вращаем микромотор со скоростью из переменной
}
```

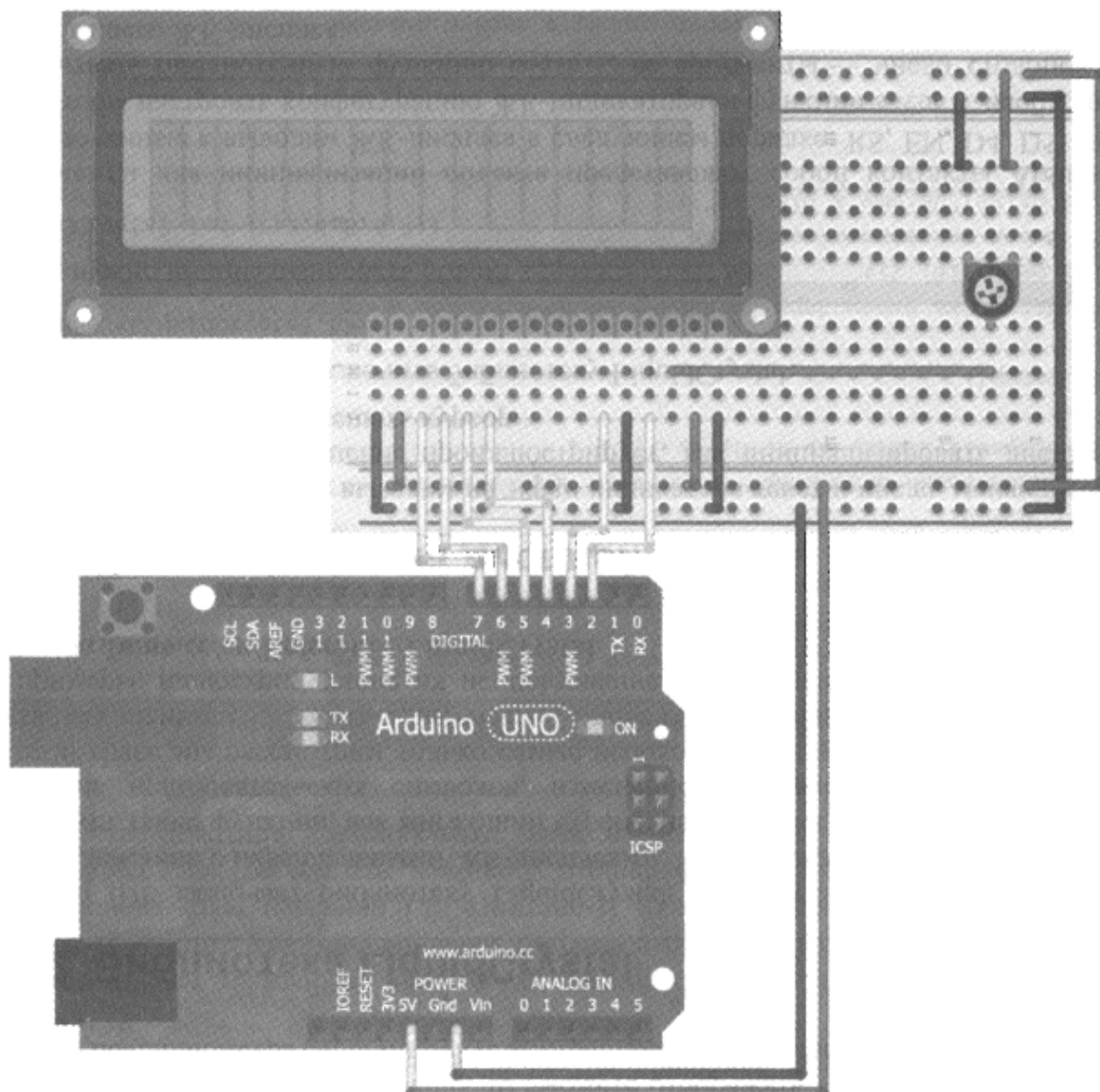


Рисунок 6 – Схема подключения дисплея с потенциометром уровня яркости
Текст программы секундного счётчика

```
#include <LiquidCrystal.h> //Подключение библиотеки мониторов

int time = 0; //Создание переменной счётчика, равного нулю
LiquidCrystal lcd(2,3,4,5,6,7); //Задание контактов подключения дисплея

void setup()
{
  lcd.begin(16,2); //Определение двухстрочного 16-символьного дисплея
  lcd.print("Display test"); //Вывод текста на дисплей
}

void loop()
{
  lcd.setCursor(0,1); //Установка курсора на первый символ второй строки
```

```
lcd.print(time); //Вывод значения переменной
delay(1000); //Задержка 1000 миллисекунд
time++; //Инкрементирование переменной
}
```

Пример обработчика прерываний

Соберите схему с кнопкой и светодиодом (для диода используйте сопротивление 100 Ом, для кнопки – 10 кОм).

```
volatile int state = LOW; //квалификатор перед переменной используется
                          //чтобы её можно было изменить из обработчика
                          //прерывания

void setup()
{
  pinMode(13, OUTPUT);
  attachInterrupt(0, blink, RISING); //настройка обработчика прерывания на
  //нулевую линию (второй контакт); вызываемая прерыванием
  // функция – blink; прерывание срабатывает при переходе сигнала
  // на контакте с LOW на HIGH
}

void loop()
{
  digitalWrite(13, state); //запись значения переменной state на светодиод
}

void blink() //функция, вызываемая прерыванием
{
  state = !state; //инверсия значения переменной
}
```

Задания:

- 1) соберите схему с мотором и потенциометром. Напишите программу для регулирования скорости вращения мотора в зависимости от угла поворота потенциометра;
- 2) дополните схему двухстрочным шестнадцатисимвольным LCD дисплеем, выводите на него значение, считанное с потенциометра. Для случая, когда мотор остановлен, на экран должно выводиться слово “STOP”;
- 3) дополните схему кнопкой. При нажатой кнопке мотор должен останавливаться. На экран монитора должно выводиться уведомление об этом;
- 4) перепишите программу таким образом, чтобы по нажатию кнопки происходило прерывание. Функция прерывания должна запускать мотор на вращение с заданной потенциометром скоростью, если он был остановлен и – останавливать, если он вращался. Информация о скорости вращения и срабатывании прерывания должна отображаться на дисплее.

3 Последовательный порт, параллельный, UART, передача данных с компьютера на микроконтроллер

Обмен информацией между различными устройствами управления в автоматизированной системе является важным фактором для построения распределённых систем и организации их иерархической структуры. В первом случае становится возможна установка отдельных устройств, содержащих датчики или управляющие элементы, на большом расстоянии друг от друга. Во втором случае организуется система с несколькими подчинёнными устройствами, каждое из которых решает локальную задачу, и глобальным устройством управления верхнего уровня, которое задаёт общий вектор управления.

Соберите схемы для проверки написанных ниже программ, проверьте их работоспособность и выполните задания. В качестве подсказок используйте Справочные материалы, находящиеся в конце этого пособия.

Управление контроллером от компьютера через последовательный порт

```
int val = 0; //создаём переменную для хранения информации с компьютера
int state=LOW; //состояние светодиода (выкл/вкл)
```

```
void setup()
{
  pinMode(13,OUTPUT); //13 ножка - выход(светодиод)
  Serial.begin(9600); //устанавливаем последовательное соединение
}
```

```
void loop()
{
  val = Serial.read(); //читаем информацию с компьютера
  if (val == '1') //если считали с компьютера цифру 1, то
  {
    state = !state; //меняем состояние диода на противоположное
                  //(HIGH->LOW или LOW->HIGH)
    Serial.println("Diode switched!"); //выводим сообщение
                                     //"диод переключился"
  }
  //проверяем статус и включаем или выключаем светодиод
  if (state == HIGH)
    digitalWrite(13,HIGH);
  else
    digitalWrite(13,LOW);
}
```

Связь по последовательному порту называемая как универсальный асинхронный прием/передача (UART). Как правило, она используется для программирования и отладки Arduino через порт USB. Существуют разные датчики и приборы, которые используют UART в качестве основной связи, и иногда нам нужно объединять два и больше Arduino между собой для обмена информацией.

Обмен данными между двумя Arduino при помощи программного UART

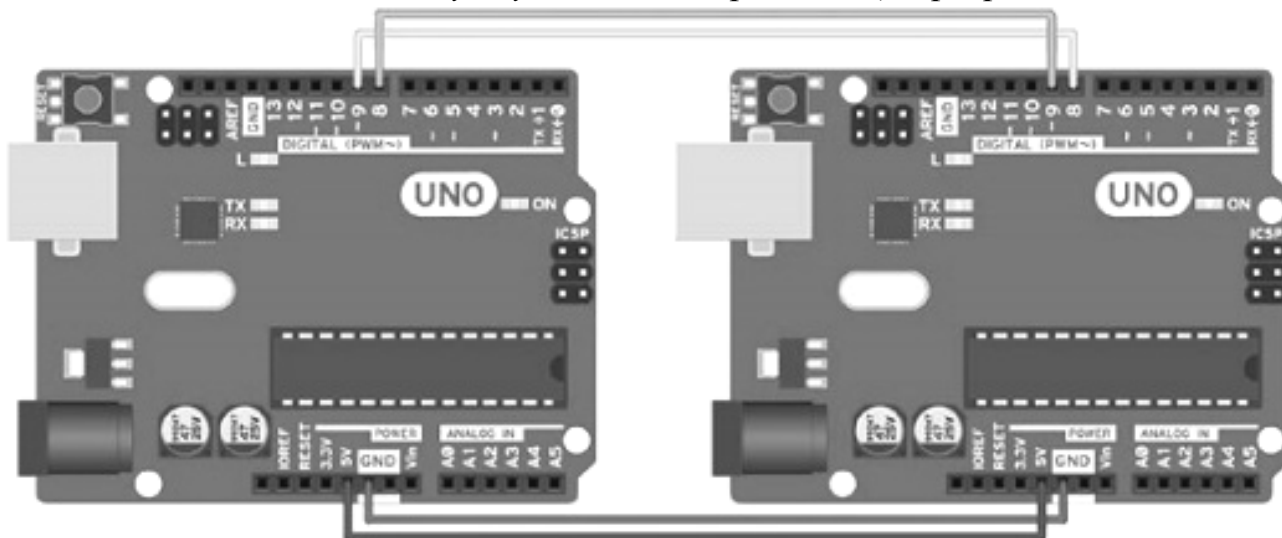


Рисунок 7 – Объединение двух контроллеров посредством UART

Текст программы для управляющего микроконтроллера:

```
// Подключение библиотеки Software Serial
#include <SoftwareSerial.h>

//Объявление дискретных каналы контроллера для связи
SoftwareSerial softSerial(8, 9); // RX, TX

void setup()
{
  Serial.begin(9600); // Задание скорости передачи данных
  softSerial.begin(9600); // Инициализация программного
                        // последовательного порта
}
void loop()
{
  if (Serial.available()) // Проверка получения команд от компьютера
  {
    softSerial.write(Serial.read()); //Отправка команды на UART
  }
}
```

Текст программы для управляемого микроконтроллера:

```
#include <SoftwareSerial.h> // Подключение библиотеки Software Serial
// Назначение задействованных дискретных каналов
SoftwareSerial softSerial(8, 9); // RX, TX
int LED = 13;

void setup()
{
  softSerial.begin(9600);
  pinMode(LED, OUTPUT); // Определение выхода светодиода
}

void loop()
{
  if (softSerial.available()) //Проверка наличия информации в буфере
  {
    // Чтение одного символа из буфера и запись его в переменную
    int com = softSerial.read();
    if (com == 'x')
    {
      digitalWrite(LED, LOW); // Выключение светодиода
    }
    else if (com == 'a')
    {
      digitalWrite(LED, HIGH); // Включение светодиода
    }
  }
}
```

Задания:

1) соберите схему с микроприводом, позволяющую задавать скорость его вращения с компьютера через последовательный порт. Если значение задаваемой скорости больше 255 или меньше 0, то микроконтроллер должен возвращать на компьютер соответствующее предупреждение;

2) соберите схему соединения двух микроконтроллеров посредством интерфейса UART и проверьте его работу;

3) соберите схему из двух микроконтроллеров, один из которых подключён к компьютеру. К одному из них должен подключаться потенциометр, а ко второму – микропривод, скорость которого должна определяться положением потенциометра и передаваться от одного контроллера к другому. Также значение скорости должно выводиться на монитор порта. Должна быть реализована возможность останова микропривода командой с компьютера, переданной по последовательному порту.

4 Применение циклов, условий, функций и классов при программировании мехатронного устройства

Помимо правильного подключения различных исполнительных элементов и датчиков к микроконтроллеру, необходимо написать программу, способную правильно обрабатывать полученную информацию и формировать управляющие воздействия. Существуют стандартные языковые конструкции для программирования контроллера, которые позволяют реализовать различные логические операции на нём. Будут рассмотрены наиболее распространённые из них – встречающиеся во многих языках программирования: оператор `if`, оператор `for`, цикл `while`.

Оператор `if` используется в сочетании с операторами сравнения и позволяет выполнять определённые действия при истинности условия.

Например:

```
if (a != b) // если a не равно b
{
    a = b; // присвоить a значение b
}
else // иначе
{
    a = 0; // присвоить a
    b = 0; // и b нулю
}
```

Вторая часть конструкции `else`, выполняемая в случае не соблюдения условия в скобках после `if`, может быть пропущена, если нет необходимости в альтернативном действии.

В скобках после `if` могут быть использованы следующие операторы сравнения:

```
x == y    // x равно y
x != y    // x не равно y
x < y     // x меньше y
x > y     // x больше y
x <= y    // x меньше или равно y
x >= y    // x больше или равно y
```

Для записи нескольких условий, которые должны проверяться одновременно, могут быть использованы логические операторы:

`&&` – логическое “И” – истинно только в том случае, если оба условия выполняются, например:

```
if (x>0 && x<5) // если x больше нуля и меньше пяти
```

`||` – логическое “ИЛИ” – истинно в случае, когда выполняется хотя бы одно из условий:

```
if (x > 0 || x < 0) // истинно, если x не равен нулю
```

Оператор `for` используется для повторения блока операторов, заключённых в фигурные скобки. Счетчик повторений обычно используется

для приращения и завершения цикла. Оператор `for` подходит для любых повторяющихся действий и используется в сочетании с массивами данных или выходов.

Наращивание яркости горения светодиода на аналоговом выходе

```
void setup()
{
  pinMode(10, OUTPUT);
}

void loop()
{
  for (int i=0; i <= 255; i++)
  {
    analogWrite(10, i);
    delay(10);
  }
}
```

Программа работает следующим образом: переменная `i` создаётся в начале выполнения оператора `for` один раз и приравнивается нулю, после этого, пока `i` меньше или равно 255, её значение подаётся на аналоговый выход контроллера. Каждое выполнение блока операторов сопровождается инкрементированием `i` на единицу. По достижении 256, оператор завершает своё выполнение и контроллер начинает выполнять функцию `loop()` сначала.

Цикл `while`(“условие”) будет выполнять блок операторов до тех пор, пока “условие” в скобках не примет значение логического нуля. Например:

```
int var = 0;          //созданная целочисленная переменная равна нулю
while(var < 200)     //пока значение переменной меньше 200, выполнять
{                   //следующий за while блок операторов
  var++;            //инкрементируем значение переменной, чтобы цикл
}                   //имел возможность завершения
```

Следует отметить, что конструкция `for` применяется в тех случаях, когда нам заранее известно количество повторений блока операторов, а цикл `while` может использоваться в ситуации, когда число итераций цикла заранее неизвестно.

Задание: Соберите схему, состоящую из микроконтроллера, LCD-монитора и двух кнопок, подключённых к линиям прерывания. Напишите программу, которая будет угадывать число от одного до ста за семь вопросов следующего типа: “Задуманное число больше 50?”. Пользователь должен отвечать на вопросы “Да” либо “Нет”, нажимая на одну из двух

соответствующих кнопок. Начните с моделирования процесса работы программы, записав последовательности вопросов и ответы на них на бумаге. Составьте блок-схему работы программы и включите её в отчёт в соответствии с требованиями к его оформлению, приведённым в справочных материалах данных методических указаний.

5 Программный интерфейс и создание собственных библиотек

Любое автоматизированное устройство, алгоритм работы которого необходимо настраивать человеку-оператору, должно иметь в своём составе интерфейс для взаимодействия с ним. В учебных проектах он будет ограничен небольшим количеством кнопок, светодиодов индикации определённых состояний процесса настройки и LCD-монитора, на который будет выводиться значащая информация о текущем состоянии устройства и его компонентов.

Довольно часто отдельные фрагменты кода необходимо многократно использовать в различных проектах. Для того, чтобы не переписывать его каждый раз, его помещают в отдельные файлы – библиотеки. Некоторые из них уже использовались в проектах ранее: библиотека монитора, сервопривода, связи микроконтроллеров.

В качестве примера приводится процесс создания библиотеки, содержащей всего одну функцию – вычисления площади круга.

Сначала в папке программы Arduino IDE\libraries\ создаётся папка библиотеки, название которой может состоять только из латинских букв и цифр, при этом не может начинаться с цифры. Назовём её “circleRound”. Внутри папки при помощи текстового редактора создаются два файла:

- circleRound.h;
- circleRound.cpp.

Файл circleRound.h будет содержать следующий код:

```
#include <inttypes.h> // Необходимо для использования числовых типов
#define PI 3.14 // Определяем Пи с точностью в 2 знака после запятой
```

```
float circleRound(float radius); //Единственная библиотечная функция
```

Файл circleRound.cpp будет содержать следующий код:

```
#include <inttypes.h> // Необходимо для использования числовых типов
#include <Arduino.h> // Подключение функции pow()
#include <circleRound.h> // в заголовочном файле
// находится PI и объявление функции

// Собственно библиотечная функция
float circleRound (float radius)
{
    return PI*pow(radius, 2);
}
```

Чтобы библиотека заработала и подключилась к программе, Arduino IDE необходимо перезапустить.

В других файлах библиотеку можно применить следующим образом:

```
#include <circleRound.h> // Подключение библиотеки
```

```
float a = circleRound(5.21); // присвоить созданной переменной a  
//значение площади круга с радиусом 5.21
```

Задание: соберите автоматическое устройство и напишите программу к нему, реализующую управление некоторым объектом по информации с двух-четырёх датчиков. Исполнительными устройствами могут быть сервоприводы, микроприводы или реле. Снабдите устройство интерфейсом управления, состоящим из монитора и нескольких кнопок или работающего от сигналов, поступающих через последовательное соединение. Уточните вариант своего задания у преподавателя.

6 Диагностика неисправностей микроконтроллерной системы управления

При построении автоматизированных систем, начиная с самых простых, возникает задача автоматического отслеживания различных нештатных ситуаций, которые могут с ними произойти и произведения адекватной реакции на них без вмешательства человека-оператора. Типичными неполадками для проектов данных методических указаний (помимо ошибок кода программ и неправильных подключений отдельных элементов) являются:

- обрыв канала связи датчика, при котором наблюдаются предельные значения, считываемые контроллером в служебные переменные. Простой анализ этих значений позволит выявить ошибку и исключить переход системы в нештатный режим;

- некорректное значение выходного сигнала контроллера на исполнительное устройство, выходящее за допустимый диапазон, например, при задании угла поворота сервопривода или скорости вращения микропривода;

- превышение частоты управляющих воздействий от датчиков, обусловленное дребезгом контактов, в системах с обработчиками прерываний. Дребезг может быть устранён программно или аппаратно.

Задание: Модифицируйте проект из предыдущего занятия таким образом, чтобы он реализовывался на нескольких микроконтроллерах а также анализировал информацию с датчиков и управляющие воздействия на наличие ошибок (обрыв, выход из диапазона допустимых значений) и выводил информацию о них на монитор.

7 СПРАВОЧНЫЕ МАТЕРИАЛЫ

7.1 Образец оформления титульного листа отчёта по лабораторным работам

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курганский государственный университет»

Кафедра «Автоматизация производственных процессов»

ОСНОВЫ МЕХАТРОНИКИ

Отчёт по лабораторным работам

Выполнили:
студенты группы Т-*****

Ф.И.О.

Ф.И.О.

Ф.И.О.

Проверил:
к.т.н., доц. Карпов Е.К.

7.2 Состав отчёта

Отчёт, выполняемый группой студентов из двух-трёх человек, должен быть пронумерован, набран 12-14 шрифтом на компьютере или разборчиво написан от руки и должен состоять из следующих частей:

- титульный лист, сделанный по образцу;
- оглавление с указанием номеров страниц, на которых начинаются отдельные задания, представленные ниже;
- схема с двумя кнопками и двумя светодиодами;
- схема с фоторезистором и светодиодом;
- схема с потенциометром и светодиодом;
- схема с сервоприводом, светодиодом и фоторезистором;
- схема с микроприводом и потенциометром;
- схема с микроприводом, потенциометром, кнопкой останова и монитором;
- схема с микроприводом, потенциометром, кнопкой останова, работающей через прерывание, и монитором;
- схема управления микроприводом от компьютера через последовательный порт;
- схема соединения двух микроконтроллеров посредством интерфейса UART;
- схема распределённого управления скоростью микропривода от потенциометра с возможностью его останова, состоящая из двух микроконтроллеров и компьютера;
- задача нахождения неизвестного числа, реализованная на схеме из микроконтроллера, монитора и двух кнопок, подключённым к прерываниям;
- индивидуальное задание на простой проект автоматизации с четырьмя датчиками и двумя исполнительными элементами и одним микроконтроллером;
- комплексное индивидуальное задание на проект автоматизации с интерфейсом взаимодействия с пользователем посредством монитора/компьютера и одним-двумя микроконтроллерами.

Каждая задача в отчёте должна содержать название, схему (если она изначально есть в методических указаниях), программный код с комментариями и блок-схему работы.

7.3 Принципы построения блок-схем

Блок-схема – распространенный тип схем, описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности. Правила выполнения регламентируются ГОСТ 19.701-90.

Основные элементы блок-схем представлены на рисунке 5.

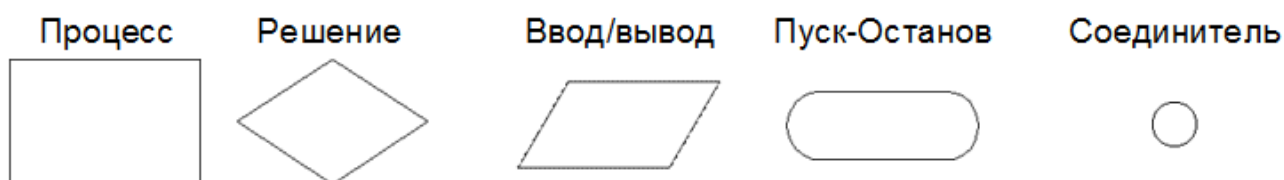


Рисунок 8 – Элементы блок-схем

Процесс – выполнение операции или группы операций, в результате чего изменяется значение, форма представления или расположения данных. Внутри символа или же в виде комментария на естественном языке или в виде формулы записываются действия, которые производятся при выполнении операции или группы операций.

Решение – выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий.

Ввод/вывод – ввод или вывод данных вне зависимости от физического носителя.

Пуск – Останов – начало, конец, прерывание процесса обработки данных или выполнения программы.

Соединитель – указание связи между прерванными линиями потока, связывающими символы. Если блок-схема состоит из нескольких частей, расположенных на одной странице, то линия потока одной части заканчивается символом Соединителя, а линия потока на продолжении блок-схемы начинается с этого же символа. Внутри символов Соединителя ставятся одинаковые порядковые номера, соответствующие разорванной линии потока.

7.4 Перечень основных команд с их описанием

Таблица 1 – Основные команды с их описанием

Действие	Программный код	Замечания
Базовые функции setup() и loop()	<pre>void setup() { //код программы, выполняемый один //раз при включении } void loop() { //код, выполняемый постоянно, //представляющий собой //основную часть }</pre>	<p>Фигурные скобки {} определяют начало и конец тела функции или блока выражений. На каждую открывающую фигурную скобку в программе должна быть закрывающая скобка.</p> <p>В конце каждого выражения и для разделения элементов программ применяется точка с запятой.</p> <p>Однострочные комментарии начинаются с //.</p>
Создание новых переменных и их типы	<pre>int outPin; outPin = 10; float pi = 3.14;</pre>	<p>// объявление переменной целочисленного типа</p> <p>// и присваивание ей значения</p> <p>// объявление и присваивание – с плавающей точкой</p>
Определение используемых входов и выходов	<pre>pinMode(12, INPUT); pinMode(outPin, OUTPUT);</pre>	<p>// 12 контакт определяется как дискретный вход</p> <p>// 10 контакт определяется как выход</p>
Цифровое чтение и цифровая запись сигналов	<pre>int a = digitalRead(inputPin); digitalWrite(outPin, HIGH);</pre>	<p>// чтение сигнала с 12 контакта в переменную a</p> <p>// запись высокого уровня на контакт outPin</p>
Аналоговое чтение и аналоговая запись сигналов	<pre>analogRead(A0); analogWrite(9, a);</pre>	<p>// чтение сигнала с аналогового входа A0</p> <p>// запись сигнала a на аналоговый выход 9</p>
Функция задержки	<pre>delay(1000);</pre>	<p>// останов выполнения программы на 1 секунду</p>

Продолжение таблицы 1

if, if-else	<pre>if (a != b) // если a не равно b { a = b; // присвоить a значение b } else // иначе { a = 0; // присвоить a b = 0; // и b нулю } if (x>0 && x<5) // если x больше нуля и //меньше пяти if (x > 0 x < 0) // истинно, если x не //равен нулю</pre>	<p>Вторая часть конструкции else, выполняемая в случае не соблюдения условия в скобках после if, может быть пропущена, если нет необходимости в альтернативном действии.</p> <p>Операторы сравнения: $x == y$ // x равно y $x != y$ // x не равно y $x < y$ // x меньше y $x > y$ // x больше y $x <= y$ // x меньше или равно y $x >= y$ // x больше или равно y</p> <p>&& – логическое “И” – истинно только в том случае, если оба условия выполняются. – логическое “ИЛИ” – истинно в случае, когда выполняется хотя бы одно из условий.</p>
Процедура подключения библиотеки	#include <название_библиотеки.h>	Конструкции для работы с конкретными библиотеками сугубо индивидуальны и должны изучаться отдельно при ознакомлении с её примерами или справочными файлами.
Операции инкремента и декремента	a++; b--;	// увеличение переменной a на единицу // уменьшение переменной b на единицу
Объявление переменной, изменяемой прерывании	volatile int state = LOW;	квалификатор перед переменной используется чтобы её можно было изменить из обработчика прерывания
Обработчик прерывания	attachInterrupt(0, функ, RISING);	У применяемых микроконтроллеров есть две аппаратные линии прерываний: 0 и 1, находящиеся соответственно на 2 и 3 контактах. функ – функция, вызываемая при срабатывании прерывания. Режим обработки прерывания определяет, когда оно должно срабатывать, и может быть настроен следующим образом: LOW - вызывает прерывание, когда на контакте LOW; CHANGE - прерывание вызывается при смене значения, с LOW на HIGH и наоборот; RISING - прерывание вызывается только при смене значения с LOW на HIGH; FALLING - прерывание вызывается только при смене значения с HIGH на LOW
Установка последовательного соединения с компьютером	Serial.begin(9600);	Скорость соединения должна совпадать с установленной в настройках оборудования. По умолчанию – 9600 бод.
Проверка наличия связи по последовательному порту	Serial.available();	Возвращает HIGH, если связь есть, иначе возвращает LOW.
Чтение информации	val = Serial.read();	Чтение информации с порта в переменную
Передача информации на компьютер	Serial.println("info"); Serial.println(a);	// передача текста info // передача значения переменной a

7.5 Процедура записи программы на микроконтроллер

Для того чтобы написать программу и записать её на микроконтроллер, необходимо выполнить следующую последовательность действий:

1) включить на компьютере исполнительный файл `arduino.exe` – это среда предназначена для разработки программного обеспечения и взаимодействия с контроллером посредством последовательного соединения;

2) выбрать в выпадающем меню “Файл”->“Новый” или нажать комбинацию клавиш `Ctrl+N` для создания нового проекта;

3) написать программный код, который предполагается исполнять на микроконтроллере, одного из заданий или разработанный самостоятельно;

4) проверить и скомпилировать программу, нажав на кнопку “Проверить” или используя комбинацию клавиш `Ctrl+R`. Во время первой проверки программа предложит сохранить файл программы. Если проверка прошла успешно, то можно переходить к следующему пункту действий, в противном случае необходимо проверить синтаксис и правильность написания логики программы ещё раз;

5) прежде чем загружать программу в микроконтроллер, необходимо выбрать порт его подключения к компьютеру, тип платы и процессора, если это необходимо. Все эти операции производятся в подпунктах выпадающего меню “Инструменты” на верхней панели программы (в нашем случае это будет `Arduino Uno` без выбора типа процессора);

6) если все предыдущие действия были выполнены без ошибок, то можно произвести загрузку программы на контроллер, нажав кнопку “Загрузка” или используя комбинацию клавиш `Ctrl+U`.

7.6 Процедура установления связи между компьютером и микроконтроллером посредством последовательного соединения

Для установления связи необходимо выполнить следующую последовательность действий:

1) включить на компьютере исполнительный файл `arduino.exe` – это среда предназначена для разработки программного обеспечения и взаимодействия с контроллером посредством последовательного соединения;

2) выбрать порт подключения микроконтроллера к компьютеру, тип платы и процессора, если это необходимо. Все эти операции производятся в подпунктах выпадающего меню “Инструменты” на верхней панели программы (в нашем случае это будет `Arduino Uno` без выбора типа процессора);

3) в выпадающем меню “Инструменты” выбрать пункт “Монитор порта” или использовать комбинацию клавиш `Ctrl+Shift+M`;

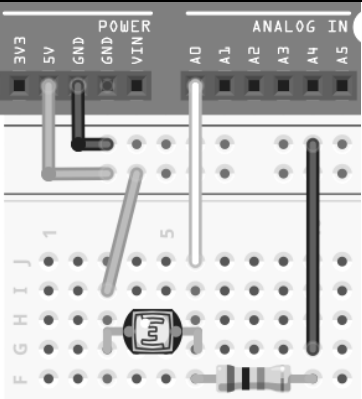
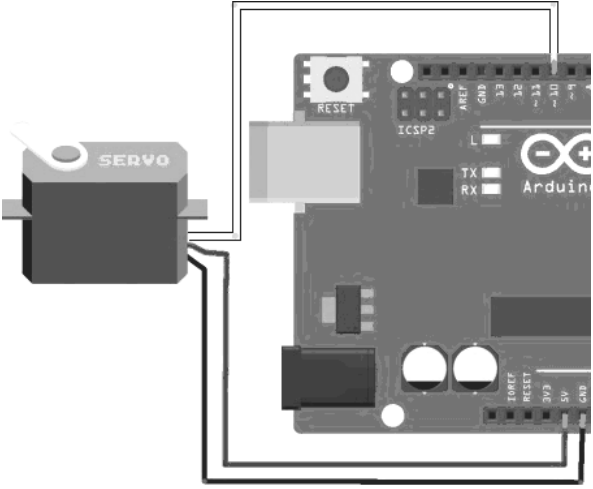
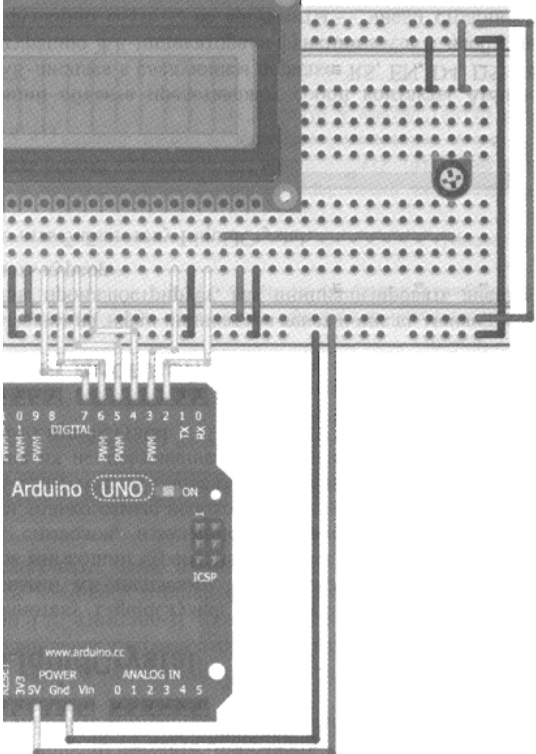
4) в нижнем поле появившегося окна будет выводиться вся информация, передаваемая от микроконтроллера, чтобы передать информацию на него, необходимо ввести её в верхнее поле окна и нажать клавишу `Enter` или кнопку “Отправить”. Скорость соединения, включение автоматической прокрутки и выбор символа конца строки можно настроить в процессе соединения.

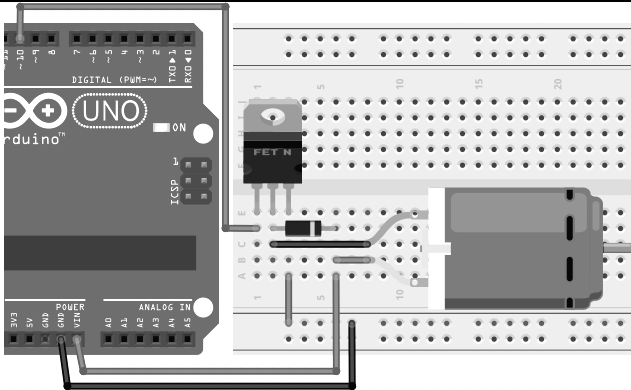
В случае переподключения платы необходимо закрыть окно “Монитор порта” и открыть его заново.

7.7 Примеры подключения электронных компонентов к микроконтроллеру

Таблица 2 – Подключение электронных компонентов

№	Поясняющий рисунок	Фрагмент Программного кода для работы с элементом	Замечания
Светодиод			
1		<pre>void setup() { //Определение ножки подключения //светодиода как выхода контроллера pinMode(13, OUTPUT); } void loop() { //Включение светодиода, //подачей высокого уровня сигнала //на ножку его подключения digitalWrite(13, HIGH); }</pre>	<p>Соблюдайте полярность светодиода</p> <p>Номинал резистора >100 Ом</p>
Кнопка			
2		<pre>void setup() { //Определение ножки подключения //кнопки как входа контроллера pinMode(3, INPUT); } void loop() { //Считывание уровня сигнала //кнопки с ножки 3 в переменную x int x = digitalRead(3); }</pre>	<p>Номинал резистора =10 кОм</p>
Потенциометр			
3		<pre>void setup() { } void loop() { //Считывание значения напряжения //потенциометра с ножки A0 в //переменную x int x = analogRead(A0); }</pre>	

Фоторезистор			
4		<pre>void setup() { } void loop() { //Считывание значения напряжения //фоторезистора с ножки A0 в //переменную x int x = analogRead(A0); }</pre>	<p>Номинал резистора =10 кОм</p>
Сервопривод			
5		<pre>//Подключение библиотеки #include <Servo.h> //Определение объекта сервопривода Servo Myservo; void setup() { //Определение ножки подключения //сервопривода Myservo.attach(10); } void loop() { //Задание угла положения //сервопривода Myservo.write(90); }</pre>	<p>Провода: Коричневый – земля Красный – 5 Вольт Жёлтый - сигнал</p>
LCD монитор 1602			
6		<pre>//Подключение библиотеки #include <LiquidCrystal.h> //Определение объекта монитора LiquidCrystal lcd(2,3,4,5,6,7); void setup() { //Определение типа монитора (16x2) lcd.begin(16,2); } void loop() { //Установка курсора в 3 символ //второй строки lcd.setCursor(2,1); //Вывод текста на экран lcd.print("Text"); //Вывод значения переменной //на экран int a=5; lcd.print(a); }</pre>	<p>Потенциометр необходим для регулирования яркости монитора</p>

Мотор с редуктором			
7		<pre>void setup() { //Определение ножки подключения //мотора как выхода контроллера pinMode(10, OUTPUT); } void loop() { //Вращение мотора с половиной //от максимальной скорости analogWrite(10, 128); }</pre>	Соблюдайте полярность диода

Карпов Егор Константинович

ОСНОВЫ МЕХАТРОНИКИ

Методические указания к комплексу лабораторных работ

по дисциплине «Основы мехатроники»

для студентов направлений подготовки

15.03.04 Автоматизация технологических процессов и производств
(профиль «Автоматизация технологических процессов и производств
в машиностроении),

27.03.04 Управление в технических системах (профиль «Системы и
технические средства автоматизации и управления»)

В авторской редакции

Подписано в печать 05.04.18	Формат 60x84 1/16	Бумага 65 г/м ²
Печать цифровая	Усл. печ. л. 2, 00	Уч.-изд. л. 2,00
Заказ №67	Тираж 25	Не для продажи

БИЦ Курганского государственного университета.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.