МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО «КАБАРДИНО-БАЛКАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. Х.М. БЕРБЕКОВА (КБГУ)»

Институт информатики, электроники и робототехники

Кафедра «Мехатроника и робототехника»

СОГЛАСОВАНО	**УТВЕРЖДАЮ**
Руководитель образовательной программы	И.о. директора ИИЭиР
_____ Х.М. Сенов	_____ Р.Ш. Гешев
«30» 05 2023 г.	«30» 05 2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.О.11.03 «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МЕХАТРОННЫХ СИСТЕМ»

Направление подготовки
15.04.06 Мехатроника и робототехника

Профиль подготовки
Мехатронные системы автоматизации в машиностроении

Квалификация (степень) выпускника
Магистр

Форма обучения
очная

Нальчик 2023

Рабочая программа дисциплины *«Программное обеспечение мехатронных систем» /сост. Ю.В. Болгов* – Нальчик: КБГУ, 2023. - *29* с.

Рабочая программа предназначена для преподавания дисциплины по выбору *блока 1 студентам очной формы обучения по направлению подготовки 15.04.06 Мехатроника и робототехника в 3 семестре.*

Рабочая программа составлена в соответствии с рабочим учебным планом и Федеральным государственным образовательным стандартом высшего образования ФГОС 3++ по направлению подготовки 15.04.06 Мехатроника и робототехника, утвержденным приказом Министерства образования и науки Российской Федерации № 1023 от 14.08.2020.

СОДЕРЖАНИЕ

1 Цели и задачи освоения дисциплины

Целью освоения – обучение студентов принципам разработки программного обеспечения мехатронных систем различного назначения, приобретение навыков объектного программирования.

Задачами дисциплины являются:

- ознакомление студентов с современными подходами к разработке и отладке программного обеспечения мехатронных систем;
- формирование навыков создания программного обеспечения для микроконтроллеров и компьютеров на базе операционной системы Windows;
- формирование навыков объектного программирования на языках высокого уровня (C++, C#) с целью создания управляющих программ вычислительного комплекса мехатронных систем различного назначения.

2 Место дисциплины в структуре ОПОП ВО

Дисциплина Б1.О.11.03. «Программное обеспечение мехатронных систем» входит в перечень дисциплин цикла подготовки магистра.

3 Требования к результатам освоения содержания дисциплины

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций по данному направлению подготовки:

ОПК-4 Способен использовать современные информационные технологии и программные средства при моделировании технологических процессов.

ОПК-4.1 Способен применять современные программные средства и информационные технологии при моделировании мехатронных и робототехнических устройств.

В результате изучения дисциплины студент должен:

Знать:

- синтаксические конструкций современных языков программирования (**З1**);
- шаблоны проектирования высокоуровневого программного обеспечения, применяющиеся в управляющих программах мехатронных систем (**З2**);
- основные алгоритмы управления мехатронными системами (**З3**).

Уметь:

- применять основные методы проектирования сложных систем программного обеспечения с использованием объектно-ориентированного подхода (**У1**);
- создавать высокоуровневые алгоритмы управления сложными мехатронными системами (**У2**).

Владеть:

- навыками применения базовых алгоритмов управления мехатронными системами (**В1**);
- навыками работы в комплексных средах создания программного обеспечения (**В2**);
- навыками написания алгоритмов и программ на языках программирования высокого уровня (**В3**);
- навыками проектирования сложных систем с использованием объектно-ориентированного подхода (**В4**).

4 Содержание и структура дисциплины (модуля)

4.1 Содержание разделов дисциплины

№ разд.	Наименование раздела	Содержание раздела	Форма текущего

			контроля
1.	Основные принципы и методология разработки прикладного программного обеспечения.	Основные принципы и методология разработки прикладного программного обеспечения мехатронных систем на базе алгоритмических языков программирования различного уровня.	Вопросы на экзамене Практические занятия
2.	Языки программирования, классификация. Языки программирования низкого и высокого уровней, как инструмент для реализации управляющих функций в мехатронных и робототехнических системах.	Обзор языков программирования. Язык программирования верхнего уровня «C++», синтаксис языка, обзор основных операторов, примеры реализации различных алгоритмов при написании управляющих программ. Обзор средств разработки и отладки программ - Microsoft Visual Studio 2010, описание интерфейса программ.	Вопросы на экзамене Практические занятия
3.	Подход объектно-ориентированного программирования при разработке ПО для управления мехатронными системами	Объектно-ориентированное программирование, назначение и основные понятия. Инкапсуляция, наследование, полиморфизм. Синтаксис языков программирования C++, C#. Особенности написания программ на языке объектно-ориентированного и объектного программирования.	Вопросы на экзамене Практические занятия Лабораторные занятия
4.	Программное обеспечение для интеллектуальных робототехнических систем, основные подходы к разработке.	Понятие интеллектуальных робототехнических систем. Интеллектуальные датчики, особенности реализации, интерфейсы, особенность написания программного обеспечения при реализации робототехнических систем.	Вопросы на экзамене Практические занятия Лабораторные занятия

4.2 Структура дисциплины

Общая трудоемкость дисциплины составляет 4 зачетные единицы (144 часа)

Вид работы	3 семестр	Всего
Общая трудоемкость	**144**	**144**
Аудиторная работа:	**54**	**54**
Лекции (Л)	9	9
Практические занятия (ПЗ)	9	9
Лабораторные(ЛР)	36	36
Самостоятельная работа:	**63**	**63**
Самостоятельное изучение разделов	40	40
Самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к лабораторным и практическим занятиям,	23	23

коллоквиумам, рубежному контролю и т.д.),		
Контроль (подготовка к сдаче экзамена)	**27**	**27**
Вид промежуточной аттестации	экзамен	экзамен

№ раз-дела	Наименование разделов	Количество часов				
		Всего	Контактная работа			Вне-ауд. работа СР
			Л	ЛР	ПЗ	
1.	Основные принципы и методология разработки прикладного программного обеспечения.	6	1	-	-	5
2.	Языки программирования, классификация. Языки программирования низкого и высокого уровней, как инструмент для реализации управляющих функций в мехатронных и робототехнических системах.	20	2	-	3	15
3.	Подход объектно-ориентированного программирования при разработке ПО для управления мехатронными системами	35	3	6	3	23
4.	Программное обеспечение для интеллектуальных робототехнических систем, основные подходы к разработке.	56	3	30	3	20
Итого:		117	9	36	9	63

4.3 Лабораторные занятия

№	Темы занятий	Кол. часов
1.	Изучение принципа действия цифрового датчика освещенности.	10
2.	Изучение принципа действия цифрового датчика температуры и влажности воздуха.	10
3.	Изучение принципа действия ультразвукового измерителя дальности..	10
4.	Изучение принципа действия систем глобального спутникового позиционирования	6
	ИТОГО	36

4.4 Практические занятия

№	Тема	Кол. часов.
1	Язык программирования верхнего уровня C++, C# синтаксис языков, обзор основных операторов, примеры реализации различных алгоритмов при написании управляющих программ.	10
3	Обзор средств разработки и отладки программ Microsoft Visual Studio 2010, описание интерфейса.	2
4	Интеллектуальные датчики и системы мобильных роботов.	2
ИТОГО		9

4.5 Самостоятельное изучение разделов дисциплины

№	Вопросы, выносимые на самостоятельное изучение	Кол-во часов
1	2	3
1.	Развитие языков программирования, современные тенденции.	5
2.	Синтаксис языков программирования C++, C#.	35
3.	Итого	40

5 Оценочные материалы для текущего контроля успеваемости и промежуточной аттестации

5.1 Оценочные материалы для текущего контроля успеваемости

Темы для рефератов:

За подготовку и защиту реферата студент может набрать 6 баллов (по 2 балла за три контрольные рейтинговые точки). При подготовке реферата студент должен ознакомиться с основной и дополнительной литературой, включая справочные издания, зарубежные источники, конспект основных положений, терминов, сведений, требующих для запоминания и являющихся основополагающими в этой теме. Необходимо составить аннотации к прочитанным литературным источникам. Структуру реферата студент определяет сам. Оценивание проводится с учетом количества обработанных литературных источников, качества оформления реферата, ответа на вопросы по реферату. Тему для реферата студент может предложить сам, либо выбрать из предложенных.

1. Современные языки программирования C++ и C#, Классы, наследование, множественное наследование.
2. Различие в передаче в функцию параметров по значению и по ссылке.
3. Динамическое выделение памяти.
4. Многомерные массивы в языках C+ и C#.
5. Отличие языков программирования C++ и C#.

Вопросы к контрольным рейтинговым мероприятиям

I:

S: Язык программирования C++, операция new …
-: служит для объявления массива
-: служит для объявления переменной целого типа
+: позволяет динамически выделить память для объекта, т.е. во время выполнения программы
-: не используется в языке C++
I:
S: Язык программирования C++, класс …
+: это производный структурированный тип, введенный программистом на основе уже существующих типов
-: это массив
-: область оперативной памяти
I:
S: Язык программирования C++, класс, компонентные функции.
-: класс не должен иметь компонентные функции
-: компонентные функции служит для повышения стабильности работы программы
+: компонентные функции класса позволяют обрабатывать данные конкретных объектов класса
I:
S: Язык программирования C++, конструктор класса.
+: это специальная компонентная функция для инициализации объектов класса
-: любая компонентная функция является конструктором класса
-: конструктор класса не применяется в языке C++
I:
S: Язык программирования C++, конструктор класса
-: имя конструктора может быть любым и может не совпадать с именем класса
+: имя конструктора должно совпадать с именем класса
-: у конструктора не должно быть имени
I:
S: Язык программирования C++, для конструктора класса …
+: не определяется тип возвращаемого значения
-: тип возвращаемого значения может быть только int
-: тип возвращаемого значения может быть только любым
-: тип возвращаемого значения может быть только double
I:
S: Язык программирования C++, конструктор класса
-: в классе может быть только один конструктор
-: в классе не может быть конструкторов
+: в классе может быть несколько конструкторов (перегрузка)
I:
S: Язык C#, конструктор класса предназначен для …
-: создания массива
-: создания переменной типа int
-: уничтожения экземпляра класса
+: создания экземпляра класса
I:

S: Язык C#, конструктор класса имеет модификатор доступа …
-: private
+: public
-: protected
-: internal
I:
S: Язык C#, конструктор класса …
-: возвращает значение целого типа
+: не возвращает никакого значения
-: возвращает значение любого типа
-: возвращает массив данных любого типа
I:
S: Язык C#, для задания перегрузки конструктора класса …
+: необходимо написать несколько конструкторов класса, каждый из которых принимает параметры разные по количеству или по типам
-: необходимо написать один конструктор класса, который не принимает параметров
-: перегрузка конструкторов в языке C# запрещена

Лабораторные работы

При выполнении лабораторных работ, студент получает практические навыки программирования. За выполнение лабораторных работ студент может набрать 18 баллов (по 3 балла в каждую рейтинговую точку).

Лабораторная работа № 1. Изучение принципа действия цифрового датчика освещенности
Цель работы: Изучение принципа действия цифрового датчика освещенности.

Лабораторная работа № 2. Изучение принципа действия цифрового датчика температуры и влажности воздуха
Цель работы: Изучение принципа действия цифрового датчика температуры и влажности воздуха.

Лабораторная работа № 3. Изучение принципа действия ультразвукового измерителя дальности
Цель работы: Изучение принципа действия ультразвукового измерителя дальности.

Лабораторная работа № 4. Изучение принципа действия систем глобального спутникового позиционирования
Цель работы: Изучение принципа действия систем глобального спутникового позиционирования.

Образец лабораторной работы представлен ниже.

Изучение принципа действия цифрового датчика освещенности

Методические указания к лабораторной работе

Цель работы: ***Изучение принципа действия цифрового датчика освещенности.***

ОБЩИЕ СВЕДЕНИЯ

Для оценки светового излучения применяются энергетические и светотехнические (визуальные) характеристики. Первые используют, как правило, для излучений, которые лежат за пределами видимого спектра. Вторые служат для описания процессов, протекающих в диапазоне видимого света и воспринимаемых глазом. Такое разделение вызвано тем, что действие видимого света на глаз зависит не только от физических параметров излучения (энергии, частоты, спектрального состава), но и *спектральной чувствительности* $S_c(\lambda)$ глаза (рис.1). Как видно на рисунке, чувствительность глаза к сине-фиолетовому и красно-оранжевому излучению существенно ниже, чем к желто-зеленому. Максимум $S_c(\lambda)$ достигается при λ = 0,5...0,555 мкм (первое значение справедливо для черно-белого, второе — для цветного зрения).

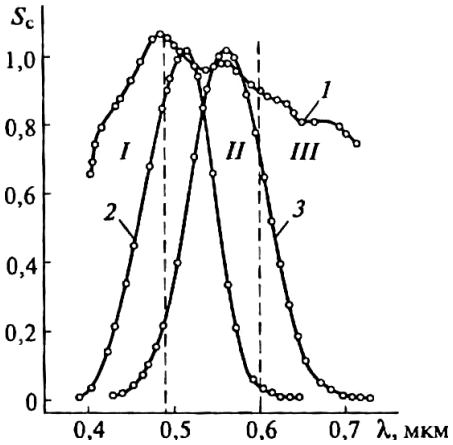


Рис. 1 - Спектральная чувствительность глаза:
I — III — сине-фиолетовая, желто-зеленая и красно-оранжевая области солнечного спектра соответственно, 1 — спектр солнечного света; 2 — черно – белая чувствительность; 3 — цветная чувствительность глаза

Для перевода фотометрических характеристик в энергетические используют *коэффициент видности* k_V, показывающий, как меняется световое ощущение по всему диапазону видимого света. Так как глаз имеет наибольшую видность (световое ощущение) V_{max} к излучению с λ = 0,555 мкм, то

$$k_V = V_\lambda / V_{max}$$

где V_λ — световое ощущение к излучению с длиной волны λ. В диапазоне 0,38...0,77 мкм коэффициент k_V изменяется в 10^5 раз.

Различают следующие основные характеристики светового излучения: энергия излучения W (энергетическая в джоулях и светотехническая в люмен-секундах);

световой поток $\Phi = \frac{dW}{dt}$ (энергетический в Вт и светотехнический в люменах, причем 1 Вт излучения с $\lambda = 5{,}55 \cdot 10^{-7}$ м соответствует 683 лм);

сила света $J = \frac{d\Phi}{d\theta}$, где θ — телесный угол (в канделах);

освещенность $\Im = \frac{d\Phi}{ds}$ (энергетическая в Вт на квадратный метр и светотехническая в люксах);

яркость (интенсивность) $Y = \frac{dJ}{dS_n}$, где dS_n — площадь ортогональной проекции светящегося элемента поверхности dS (светотехническая в канделах на квадратный метр).

В фотометрии для определения светотехнической яркости по известной энергетической используют таблицы видности. Например, энергетическая яркость потока гелий-аргонового лазера с λ = 0,514 мкм, работающего в непрерывном режиме, составляет 10 Вт/м^2. По таблице видности фотометрическая яркость Y= $4 \cdot 10^{15}$ кд/м^2 , что

приблизительно в $2{,}5 \cdot 10^6$ раз больше яркости солнца.

Способность глаза реагировать на изменение яркости в очень большом диапазоне получило название зрительной адаптации. В среднем для человека $\Delta Y = 2(10^{-6}...10^5)$ кд/м^2. Свойство глаза восстанавливать световую чувствительность называется световой или темповой адаптацией. Первая проявляется при резком увеличении освещенности, например при выходе, из темного помещения на свет, и составляет 2—3 мин; вторая существенно продолжительнее и достигает 20—30 мин. Параметры некоторых типовых источников света представлены ниже:

Фотографическая вспышка …… $Y = 7 \cdot 10^{10}$ кд/м^2
Лампа накаливания …… $Y = 6 \cdot 10^6$ кд/м^2
Дневной свет ..... $\Im = 10^4$ лк
Полная луна …….. $Y = 2 \cdot 10^3$ кд/м^2, $\Im = 2 \cdot 10^{-1}$ лк
Звездное небо ..... $Y = 4 \cdot 10^{-4}$ кд/м^2, $\Im = 10^{-1}$ лк
П р и м е ч а н и е . Минимальная видимая яркость $Y = 10^{-5}$ кд/м^2.

Порог чувствительности глаза характеризуется наименьшим количеством энергии, вызывающей его световое раздражение. Пороговое значение светового потока зависит от диаметра зрачка и составляет около $2 \cdot 10^{-14}$ лм при диаметре зрачка 8 мм.

Цифровой датчик освещенности BH1750FVI

BH1750FVI - цифровой датчик освещенности реализован в виде микросхемы с цифровым выходом (интерфейс I^{2}C). Датчик может применяться в различных электронных и мехатронных устройствах, для получения данных о внешней освещенности (например, системы управления освещением зданий, для автоматической регулировки яркости экранов ЖК телевизоров и мобильных телефонов и т.д.). Датчик позволяет измерять интенсивность света в широком спектре с высоким разрешением от 1 до 65535 лк. На рис. 2 показан общий вид платы с установленным датчиком.

Рис. 2. Общий вид платы датчика освещенности.

Краткие характеристики датчика.

- Интерфейс I^{2}C.
- Спектральная чувствительность близка к чувствительности человеческого глаза.
- Встроенный 16-и разрядный аналого-цифровой преобразователь.
- Широкий диапазон измерений освещенности (1 - 65535 лк).
- Низкое энергопотребление.
- Функция подавления помех от источников света на частоте: 50Гц - 60Гц.
- Возможность подключения 2-х датчиков к одному контроллеру по интерфейсу I^{2}C (можно выделить два адреса устройства).
- Погрешность измерения (±20%).
- Низкое влияние инфракрасного излучения на результат измерения.
- Максимальная частота обмена данными по интерфейсу I^{2}C – 400 кГц.
- Рабочий диапазон температур -40^0C ÷ $+85^0$C.

Последовательный интерфейс TWI (I^2C)

Интерфейс I^2C требует двухпроводного соединения, но с обязательным объединением "земель", т. к. сигналы в нем абсолютные, а не дифференциальные, и отсчитываются относительно "земли". В интерфейсе I^2C устройства могут работать в режиме "ведущий" (Master) или "ведомый" (Slave). В отличие от большинства других интерфейсов, ведомые устройства с интерфейсом I^2C должны иметь индивидуальный адрес, присваиваемый производителем. Для различения одинаковых устройств, если их более двух на одной линии, в некоторых типах устройств (как в рассматриваемом датчике освещенности) имеются дополнительные адресные линии, выводы для установки индивидуального адреса или входы типа "выбор кристалла". Датчик освещенности имеете адрес **0100011** (если на вывод ADDR датчика подан низкий логический уровень) и адрес **1011100** (если на вывод ADDR датчика подан высокий логический уровень).

I^2C служит для связи между собой микросхем на одной плате или в пределах одного устройства. Однако это медленный интерфейс (максимальное значение скорости обмена для датчика освещенности — 400 кбит/с), и потому применяется там, где не требуется скоростной передачи данных. В интерфейсе есть несколько различающихся состояний ("старт", "стоп", передача от мастера или к мастеру и т. д.). Кроме того, из-за наличия всего одной линии обмена данными (да еще и с поддержкой многих устройств, подключенных к ней – рис. 3), приходится организовывать протокол так, чтобы исключить электрические конфликты.

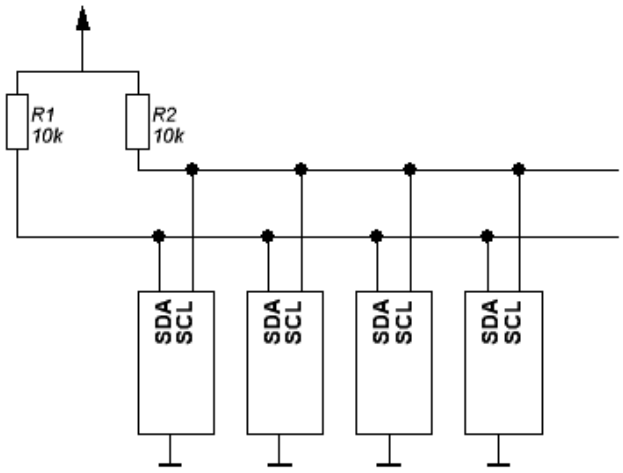


Рис. 3. Схема подключения нескольких устройств к интерфейсу.

Предположим, у нас есть несколько устройств, подключенных параллельно к двум линиям (не считая, естественно, "земли"). По одной из них (SCL) всегда передаются синхронизирующие импульсы, а по второй — собственно данные (SDA). Информация в каждый данный момент времени передается только одним устройством и только в одну сторону. С помощью TWI можно (теоретически) соединить до 128 устройств, так, как показано на рис 3. "Подтягивающие" резисторы должны иметь номинал порядка единиц или десятков КОм (чем выше скорость передачи, тем меньше).

Чтобы различить несколько устройств, каждое из них обязано иметь индивидуальный адрес. Он задается 7-битовым кодом, потому всего таких устройств на одной линии может быть 128.

Типовой вариант обмена информацией по интерфейсу I^2C показан на рис. 4. Кратко расшифруем эту диаграмму. Любой сеанс передачи по протоколу I2C начинается с состояния линии, именуемого Start (когда сигнал на линии SDA меняется с лог. 1 на лог. О

при высоком уровне на линии SCL). Start может выдаваться неоднократно (тогда он называется "повторный старт"). Заканчивается сеанс сигналом Stop (состояние линии SDA меняется с лог. 0 на лог. 1 при высоком уровне на линии SCL). Между этими сигналами линия считается занятой, и только ведущий (тот, который выдал сигнал Start) может управлять линией. Сама информация передается уровнями на линии SDA (в обычной положительной логике, старший разряд первым), причем смена состояний может происходить только при низком уровне на SCL, а при высоком уровне на ней происходит считывание значения бита. Любая смена уровней SDA при высоком уровне SCL будет воспринята как либо Start, либо Stop.

Процесс обмена всегда начинается с передачи ведущим байта, содержащего адрес устройства (также начиная со старшего разряда), который содержится в семи старших битах. Первый (младший!) бит этого байта называется R/W и несет информацию о направлении обмена: если он равен 0, то далее ведущий будет передавать информацию, т. е. писать (W), если равен 1 — читать (R), т. е. ожидать данные от ведомого. Все посылки (и адресные, и содержащие данные) сопровождаются девятым битом, который передается последним и называется битом квитирования. Во время действия этого девятого импульса адресуемое устройство (т. е. ведомый, который имеет нужный адрес после посылки адреса ведущим, или ведущий, если данные направлены к нему, и т. п.) обязан сформировать ответ (ACK) низким уровнем на линии SDA.

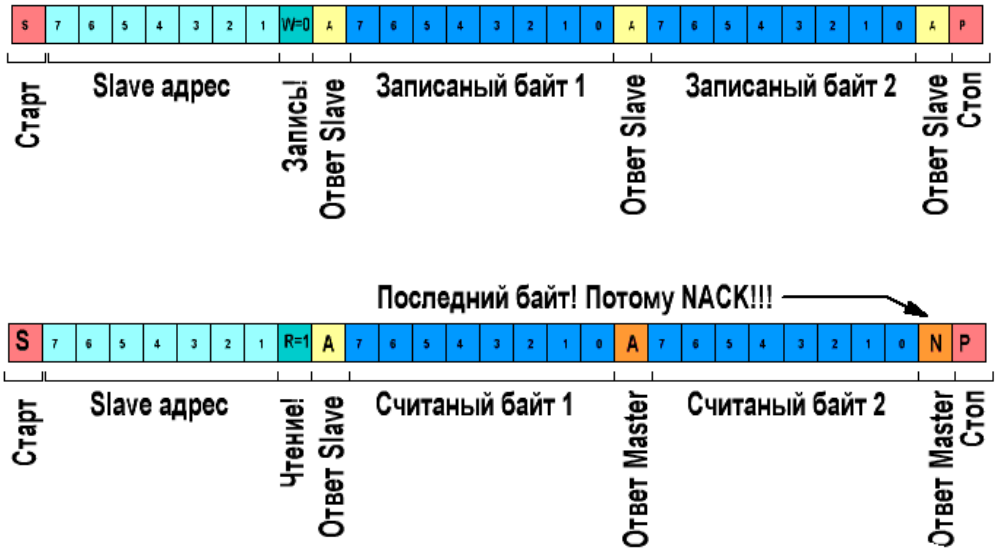


Рис. 4. Процесс обмена данными по интерфейсу.

Если такого ответа нет (NACK), то можно считать, что данные не приняты и фиксировать сбой на линии. Иногда устройства не требуют отсылки бита ACK (или игнорируют его).

Как видим, организовать обмен по протоколу I^2C непросто, но это плата за универсальность и простоту электрической схемы. Большинство современных устройств с интерфейсом I^2C могут работать с тактовой частотой до 400 кГц и более, но из-за не слишком высокой помехоустойчивости такой линии максимальные частоты целесообразны только тогда, когда микросхемы установлены на одной плате недалеко друг от друга. При соединении проводами (например, МК с каким-нибудь датчиком) лучше ограничиться частотами до 100 кГц, а при длинных линиях связи (провода в полметра длиной и более), частоту обмена следует снизить до 10-30 кГц.

На рис. 5 представлена спектральная характеристика датчика, а на рис. 6 – диаграмма направленности.

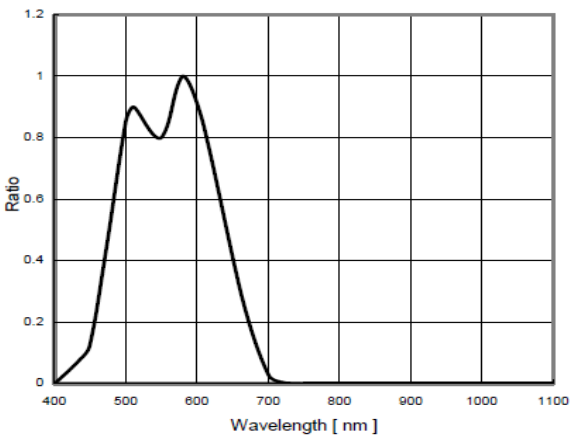


Рис. 5. Спектральная характеристика датчика.

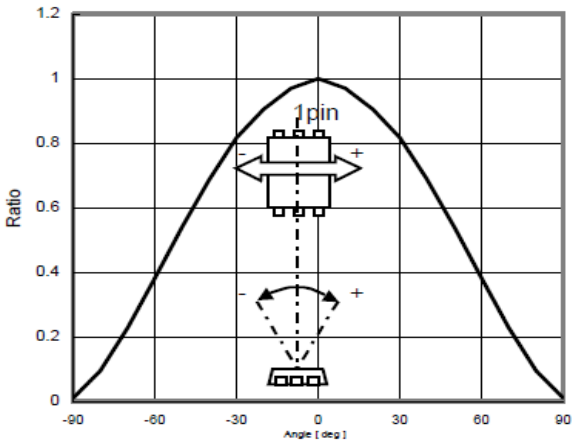


Рис. 6. Диаграмма направленности датчика.

На рис. 7 приведена структурная схема датчика, а в таблице 1 приведены режимы измерения, а в таблице 2 - команды управления режимом работы датчика.

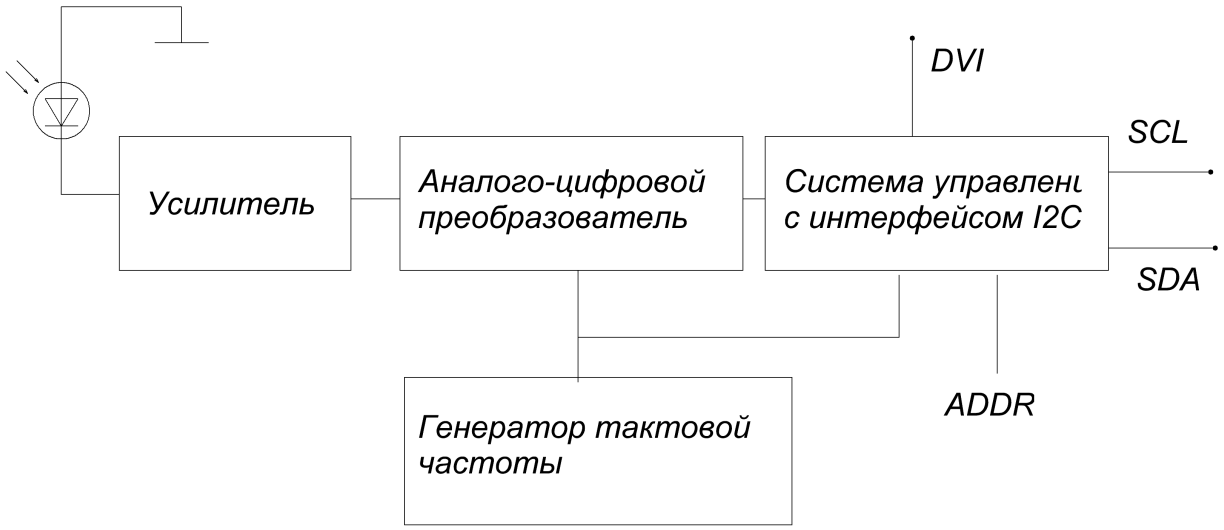


Рис. 7. Структурная схема датчика освещенности.

Таблица 1 – Режимы измерения.

Режим измерения	Время измерения	Разрешение
Режим высокого разрешения (режим 2).	120 ms.	0.5 лк
Режим высокого разрешения (режим 1).	120 ms.	1 лк
Режим низкого разрешения.	16 ms.	4 лк

Таблица 2 - Команды управления режимом работы датчика.

Команда	Код команды	Описание
Отключение датчика	0000_0000	Датчик отключен.
Включение датчика	0000_0001	Датчик включен и ожидает команду на измерение.

Сброс датчика	0000_0111	Сброс настроек датчика. Команда недоступна в режиме отключения датчика.
Непрерывное измерение в режиме высокого разрешения (Режим 1).	0001_0000	Запуск измерений с разрешением 1 лк. Время одного измерения 120 ms.
Непрерывное измерение в режиме высокого разрешения (Режим 2).	0001_0001	Запуск измерений с разрешением 0,5 лк. Время одного измерения 120 ms.
Непрерывное измерение в режиме низкого разрешения.	0001_0011	Запуск измерений с разрешением 4 лк. Время одного измерения 16 ms.
Однократное измерение в режиме высокого разрешения (Режим 1).	0010_0000	Запуск измерений с разрешением 1 лк. Время одного измерения 120 ms. Автоматический перевод датчика в режим отключения после проведения измерения.
Однократное измерение в режиме высокого разрешения (Режим 2).	0010_0001	Запуск измерений с разрешением 0,5 лк. Время одного измерения 120 ms. Автоматический перевод датчика в режим отключения после проведения измерения.
Однократное измерение в режиме низкого разрешения.	0010_0011	Запуск измерений с разрешением 4 лк. Время одного измерения 16ms.Автоматический перевод датчика в режим отключения после проведения измерения

При работе с датчиком рекомендуется применять режим высокого разрешения. Время измерения в режиме высокого разрешения большое из-за необходимости подавления источников помех (например, излучения с частотой 50 – 60 Гц). Режим высокого разрешения (с разрешением 1 лк) позволяет фиксировать небольшой уровень освещенности (менее 10 лк). Режим с разрешением в 0,5 лк позволяет фиксировать излучение практически в полной темноте.

Пример последовательности команд установки режима измерения

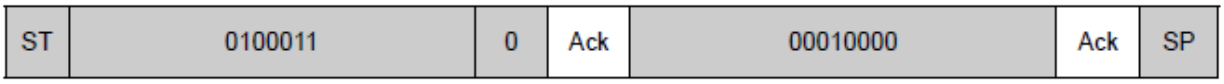
Пример 1.

Непрерывное измерение в режиме высокого разрешения (ADDR = “L”)

■ От ведущего устройства к ведомому.

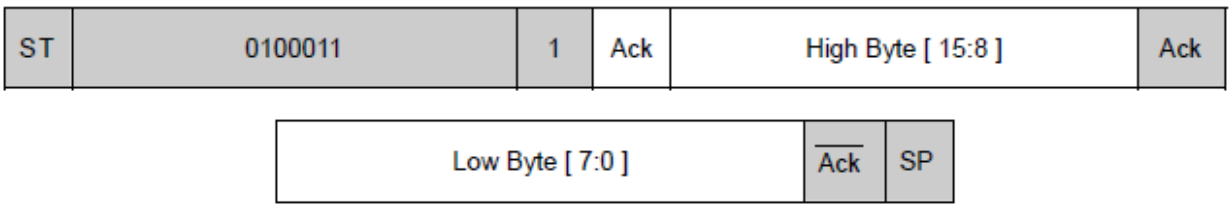
□ От ведомого к ведущему.

1. Запись команды непрерывное измерение в режиме высокого разрешения.



2. Ожидание завершения измерения (максимальное время 180 ms).

3. Чтение результата измерения.



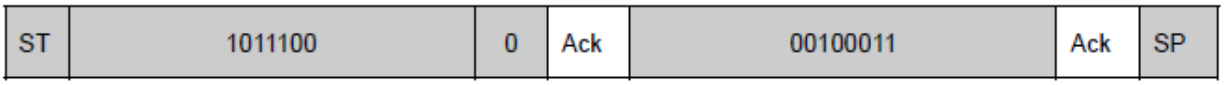
Пример расчета результата измерения: Старший байт – «10000011», младший байт «10010000». $(2^{15}+2^9+2^8+2^7+2^4)/1{,}2 = 28067$ лк.

Результаты измерений будут непрерывно обновляться (120 ms для режима высокого разрешения и 16 ms для низкого разрешения).

Пример 2.

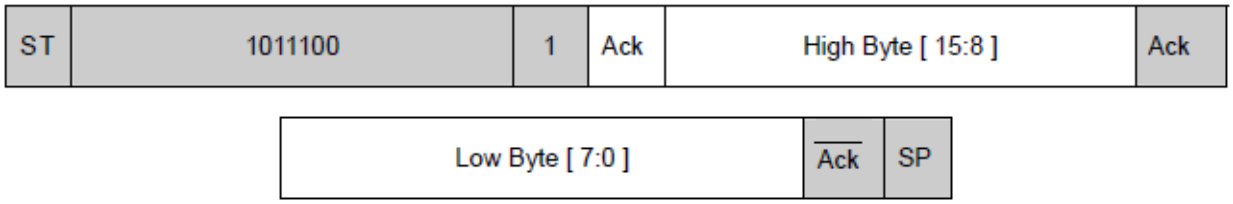
Однократное измерение в режиме низкого разрешения (ADDR = “H”)

1. Запись команды однократного измерения в режиме низкого разрешения.



2. Ожидание завершения измерения в режиме низкого разрешения (максимальное время 24 ms).

3. Чтение результатов измерения.



Пример расчета результата измерения: Старший байт – «00000001», младший байт «00010000». $(2^8+2^4)/1{,}2 = 227$ лк.

После однократного измерения датчик перейдет в режим отключения. Для повторного измерения необходимо заново записать команды.

Ниже приведен листинг тестовой программы получение данных от модуля.

```
/*
Чтения данных датчика освещенности
Проверено на м/к AVR Atmega2560
 */
#include <avr/io.h>
#include <util/delay.h>     /* Для delay */
#include <avr/wdt.h>
#include <stdlib.h>
#include <avr/interrupt.h>


unsigned int delay_counter=0;
int count;        // Счетчик байт (получаем 2 байта)
unsigned char  datadht[2]; // массив принятых байт
int flag;
#define DHT_PORT       PORTB // порт
#define DHT_DDR        DDRB
#define DHT_PIN        PINB
#define DHT_BIT        4 // БИТ порта

// Для индикатора используется порт A микроконтроллера
#define  LCDPORT                        PORTA
#define  LCDDDR                         DDRA
#define LCD_PIN_RS              2    // 2 - пин порта A
#define LCD_PIN_E               3
#define LCD_PIN_D4              4
#define LCD_PIN_D5              5
```

```
#define LCD_PIN_D6              6
#define LCD_PIN_D7              7

// Команды управления ЖК-дисплеем
#define LCD_CLEAR               0x01  //
#define LCD_HOME                0x02
#define LCD_ON                          0x0C    // 0x0C - курсор невидим
//#define LCD_ON                        0x0E    // 0x0E - курсор видим
#define LCD_OFF                         0x08

// Вспомогательные определения
#define  COMMAND                        0
#define DATA                    1
#define sbi(sfr, bit)           (sfr|=(1<<bit))
#define cbi(sfr, bit)           (sfr&=~(1<<bit))

// Прототипы функций
void Pin_Init();
void Timer_Init();
void Pa3_On();
void Pa3_Off();
void TWI_Stop();

unsigned int h;
//================================================
// Инициализация интерфейса TWI
void TWI_MasterInit()
{
// Настройка скорости передатчика
// 200 кГц
TWBR = 0x20;
TWSR &= ~(1<<0) & ~(1<<1); // 00 - можно не ставить
}

//============================================================
// Запуск TWI
void TWI_Start()
{
            // Установим Старт - 1 в TWSTA
            // Разрешим работу модуля - 1 в TWEN
            // Сбросим флаг TWINT - новый цикл обмена
            TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
    wdt_reset(); // Сброс сторожевого таймера
            // Ждем формирования сигнала Старт - установки разряда TWINT
            while (!(TWCR & (1<<TWINT)));
    wdt_reset(); // Сброс сторожевого таймера
// Записываем адрес устройства, последний бит 0 - запись
                        TWDR = 0b01000110;
// передаем
                        TWCR = (1<<TWINT) | (1<<TWEN);
//Ждем конца передачи и сигнала ASK (адресный пакет передан)
                        while(!(TWCR & (1<<TWINT)))
                        {};
      wdt_reset(); // Сброс сторожевого таймера
// Записываем команду Power On
                        TWDR = 0b00000001;
// передаем
                        TWCR = (1<<TWINT) | (1<<TWEN);
//Ждем конца передачи и сигнала ASK (команда передана)
                        while(!(TWCR & (1<<TWINT)))
                        {};
     wdt_reset(); // Сброс сторожевого таймера
                        // Формируем сигнал Стоп
```

```
    TWI_Stop();
}

//=================================================
// Установим измерение в режим высокого разрешения
void Datchik_H_mode()
{
             wdt_reset(); // Сброс сторожевого таймера
            // Установим Старт - 1 в TWSTA
            // Разрешим работу модуля - 1 в TWEN
            // Сбросим флаг TWINT - новый цикл обмена
            TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
            // Ждем формирования сигнала Старт - установки разряда TWINT
            while (!(TWCR & (1<<TWINT)));
    wdt_reset(); // Сброс сторожевого таймера
// Записываем адрес устройства, последний бит 0 - запись
                        TWDR = 0b01000110;
// передаем
                        TWCR = (1<<TWINT) | (1<<TWEN);

//Ждем конца передачи и сигнала ASK (адресный пакет передан)
                        while(!(TWCR & (1<<TWINT)))
                        {};
      wdt_reset(); // Сброс сторожевого таймера
// Записываем команду One Time H - Resolution Mode
                        TWDR = 0b00100001;
// передаем
                        TWCR = (1<<TWINT) | (1<<TWEN);

//Ждем конца передачи и сигнала ASK (команда передана)
                        while(!(TWCR & (1<<TWINT)))
                        {};
    wdt_reset(); // Сброс сторожевого таймера
                        // Формируем сигнал Стоп
     TWI_Stop();

// Задержка 180 мС
            _delay_ms(180);
}

//=================================================
// Измеряем
void Datchik_R()
{
 wdt_reset(); // Сброс сторожевого таймера
            // Установим Старт - 1 в TWSTA
            // Разрешим работу модуля - 1 в TWEN
            // Сбросим флаг TWINT - новый цикл обмена
            TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
            // Ждем формирования сигнала Старт - установки разряда TWINT
            while (!(TWCR & (1<<TWINT)));
 wdt_reset(); // Сброс сторожевого таймера
// Записываем адрес устройства, последний бит 1 - чтение
                        TWDR = 0b01000111;
// передаем
                        TWCR = (1<<TWINT) | (1<<TWEN);

//Ждем конца передачи и сигнала ASK (адресный пакет передан)
                        while(!(TWCR & (1<<TWINT)))
                        {};
 wdt_reset(); // Сброс сторожевого таймера
// Считываем первый байт (старший)
```

```
		TWCR = (1<<TWEA) | (1<<TWINT) | (1<<TWEN);

                        while (!(TWCR & (1<<TWINT)))
                        {};
 wdt_reset(); // Сброс сторожевого таймера
                        datadht[0] = TWDR; // Записываем старший байт
		TWCR = (1<<TWINT) | (1<<TWEN);
                        while (!(TWCR & (1<<TWINT)));
 wdt_reset(); // Сброс сторожевого таймера
                        datadht[1] = TWDR; // Записываем младший байт
                        // Формируем сигнал Стоп
    TWI_Stop();
        h = (datadht[0] * 256 + datadht[1])/1.2; // получаем освещенность

}

//============================================
// Остановка TWI
void TWI_Stop()
{
// Установим состояние СТОП - 1 в TWSTO
// Модуль работает в состоянии стоп
TWCR = (1<<TWINT) | (1<<TWSTO) | (1<<TWEN);
}

//=========================================================
// обработка прерывания при совпадении А таймера 1
ISR(TIMER1_COMPA_vect)
{
        if(~PINA & (1<<3))  // Если PA3 = 0
        {
//                      Pa3_On();
        }
        else
        {
//      Pa3_Off();
        }

        // Сброс таймера 1 в начальное значение
  TCNT1 = 0x0000 ;   // Старший и младший байт 16-разрядного регистра TCNT1

}

//=================================================================
// Настройка таймеров
void Timer_Init()
{
//-------------------------------------------------------------
// Настройка таймера 1
// Установим делитель для таймера 1
                                // 0 - таймер остановлен
                                                        // 1 - коэффициент пределителя частоты
таймера - 1
                                                        // 2 - 8
                                                        // 3 - 64
                                                        // 4 - 256
                                                        // 5 - 1024
        TCCR1B = 5;
TCNT1 = 0x0000;   // Старший и младший байт 16-разрядного регистра TCNT1


// Значение регистра OCR1A, непрерывно сравнивается с
// текущим значением счетного регистра TCNT1
```

```
OCR1A = 0xffff;

//---------------------------------------------------------------
// Прерывание по совпадению A таймера 1 разрешено
	TIMSK1 |= (1<<OCIE1A);
}
//-------------------------------------------------------
// Отлючение светодиода
void Lite_Off()
{
PORTB &= ~(1<<7); // Подаем логический ноль
}
//-------------------------------------------------------
// Включение светодиода
void Lite_On()
{
PORTB |= (1<<7); // Подаем логическую единицу
}
//-----------------------------------------------------------------------------------------
// Выдача байта в контроллер ЖК-модуля
void lcd_send(unsigned char type, unsigned char c)
{
	if (type==COMMAND)
		cbi(LCDPORT, LCD_PIN_RS); // RS=0: последует команда
	else
		sbi(LCDPORT, LCD_PIN_RS); // RS=1: последуют данные
	// Передача старшего полубайта
	if (bit_is_set(c, 7)) sbi(LCDPORT, LCD_PIN_D7); else cbi(LCDPORT, LCD_PIN_D7);
	if (bit_is_set(c, 6)) sbi(LCDPORT, LCD_PIN_D6); else cbi(LCDPORT, LCD_PIN_D6);
	if (bit_is_set(c, 5)) sbi(LCDPORT, LCD_PIN_D5); else cbi(LCDPORT, LCD_PIN_D5);
	if (bit_is_set(c, 4)) sbi(LCDPORT, LCD_PIN_D4); else cbi(LCDPORT, LCD_PIN_D4);
	// Формирование фронта для приема данных
	sbi(LCDPORT, LCD_PIN_E);
	cbi(LCDPORT, LCD_PIN_E);
	// Передача младшего полубайта
	if (bit_is_set(c, 3)) sbi(LCDPORT, LCD_PIN_D7); else cbi(LCDPORT, LCD_PIN_D7);
	if (bit_is_set(c, 2)) sbi(LCDPORT, LCD_PIN_D6); else cbi(LCDPORT, LCD_PIN_D6);
	if (bit_is_set(c, 1)) sbi(LCDPORT, LCD_PIN_D5); else cbi(LCDPORT, LCD_PIN_D5);
	if (bit_is_set(c, 0)) sbi(LCDPORT, LCD_PIN_D4); else cbi(LCDPORT, LCD_PIN_D4);
	// Формирование фронта для приема данных
	sbi(LCDPORT, LCD_PIN_E);
	cbi(LCDPORT, LCD_PIN_E);
	// Ожидание контроллера ЖК-модуля
	_delay_ms(5);
}

// Отображение строки символов
void lcd_write(char *t)
{
	unsigned char i;
	for (i=0;i<255;i++)
	{
		if (t[i]==0) // Конец строки?
			return;
		else
			lcd_send(DATA, t[i]);
	}
}

// Инициализация ЖК-модуля
void lcd_init()
{
	// Переключение порта на вывод
```

```
	LCDPORT = 0x00;
	LCDDDR		= 0xFF;
	_delay_ms(50); // Ожидание готовности ЖК-модуля
	// Конфигурирование четырехразрядного режима
	sbi(LCDPORT, LCD_PIN_D5);
	cbi(LCDPORT, LCD_PIN_D4);
	// Активизация четырехразрядного режима
	sbi(LCDPORT, LCD_PIN_E);
	cbi(LCDPORT, LCD_PIN_E);
	_delay_ms(5);
	// 2 строки, передача по четыре разряда
	lcd_send(COMMAND, 0x28);
//lcd_send(COMMAND, 0b00100100);
	lcd_send(COMMAND, LCD_OFF);
	lcd_send(COMMAND, LCD_CLEAR);
	lcd_send(COMMAND, 0x06);
	lcd_send(COMMAND, LCD_ON);
}
// Функция, реализующая ожидание i секунд
void seconds(char i)
{
	unsigned char k;
	for (;i>0;i--)
	{
		for (k=0;k<10;k++)
			_delay_ms(100);
	}
}

//-------------------------------------------------
//	Main программа
//-------------------------------------------------
int main(void)
{
// Начальные установки программы
	wdt_enable(WDTO_4S); // Включение сторожевого таймера на 4 с
count = 0;
flag = 0;
// Светодиод индикации на плате Arduino (PB7)
		DDRB |= (1<<7); // Разряд 7 (PB7) на вывод
		PORTB &= ~(1<<7); // Подаем логический 0 (гасим светодиод)
	 sei(); // Разрешим прерывания
		int radix=10; // основание
	 char buffer[7]; // Буфер
	 char *p1;
	lcd_init(); // настройка индикатора
	lcd_write("Starting..."); // вывод на экран строки

	_delay_ms(1000);	// Задержка 1 с
	lcd_send(COMMAND, LCD_CLEAR);

   for(;;)
	{		/* цикл программы */
	wdt_reset(); // Сброс сторожевого таймера
	_delay_ms(100);
wdt_reset(); // Сброс сторожевого таймера
      TWI_Start();
		Datchik_H_mode();
		Datchik_R();
		TWI_Stop();
 wdt_reset(); // Сброс сторожевого таймера
	Lite_Off();
	wdt_reset(); // Сброс сторожевого таймера
```

```
{

// выводим на экран
                                                lcd_send(COMMAND, LCD_HOME);
                                                 lcd_write("Osv = ");
                                                p1=itoa(h, buffer, radix);
                                                lcd_write(p1);
                                                lcd_write(" ");
                                                 lcd_write("L               ");
}
     wdt_reset(); // Сброс сторожевого таймера
       }}
//*********************************************************************
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Соберите устройство для измерения освещенности (рис. 8). Принципиальная электрическая схема показана на рис. 9.

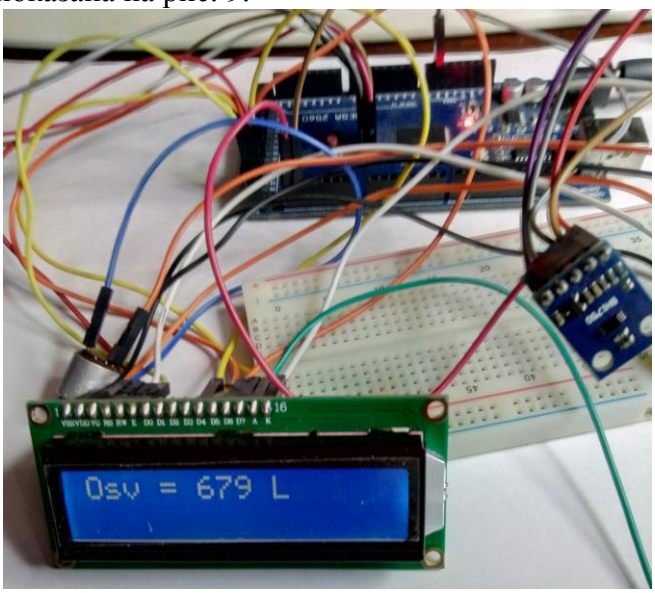


Рис. 8. Собранное устройство измерения освещенности.

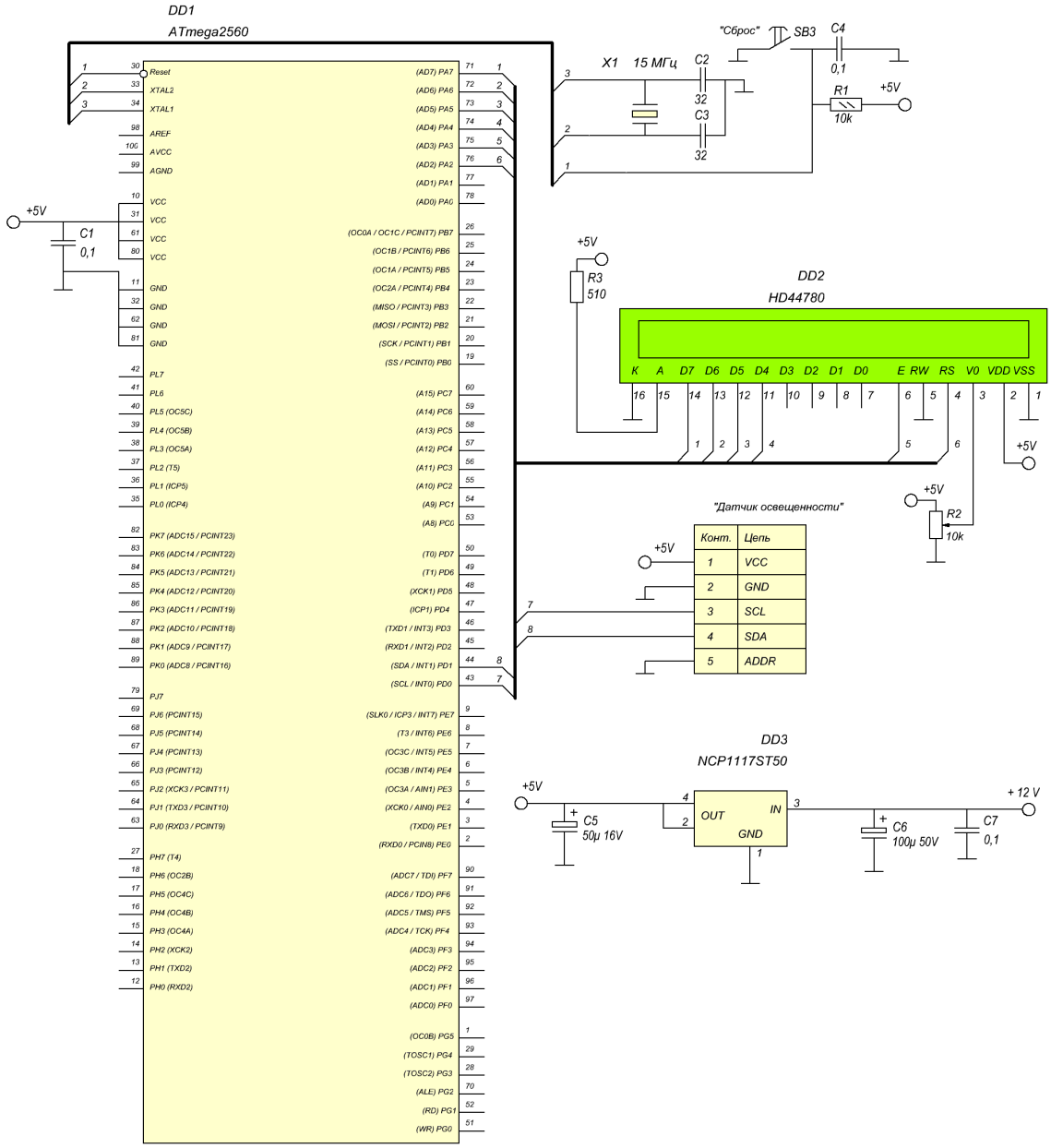


Рис. 9. Схема подключения датчика освещенности к микроконтроллеру и индикатору.

2. Запишите в память микроконтроллера управляющую программу. Убедитесь, что система работает.
3. Измените код программы из условия – измерений с разрешением 0,5 лк.

СОДЕРЖАНИЕ ОТЧЕТА

1. Цель выполнения работы.
2. Характеристики и область применения цифрового датчика освещенности.
3. Режимы работы датчика и процедура получения информации.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Цель выполнения работы?

2. Принцип действия цифрового измерителя освещенности.
3. Опишите возможные режимы измерения.
4. Опишите интерфейс подключения датчика к микроконтроллеру.
5. Область практического применения цифрового датчика освещенности.

ЛИТЕРАТУРА

1. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательский дом «Додека-XXI», 2006. -288 с.
2. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих. –М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. -240 с.
3. Ревич Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. – СПб.: БХВ-Петербург, 2008. -384 с.
4. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров. –К.: «МК-Пресс», 2006. -400 с.
5. Воротников С.А. Информационные устройства робототехнических систем. –М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. -384 с.
6. Распопов В. Я. Микромеханические приборы. –М.: Машиностроение, 2007. -400с.

Вопросы к экзамену

1.	Языки программирования низкого и высокого уровня. Общий обзор.
2.	Объектно-ориентированное программирование, назначение и основные понятия.
3.	Понятие класса в объектно-ориентированном программировании.
4.	Понятия – инкапсуляция, наследование, полиморфизм.
5.	Язык С++, основные типы переменных, преобразование типов, явное и неявное.
6.	Язык С++, операторы управления потоком выполнения программы – if – else, switch.
7.	Язык С++, операторы цикла.
8.	Язык С++, указатели и ссылки, назначение и применение.
9.	Язык С++, массивы и указатели.
10.	Язык С++, операторы new и delete, их назначение.
11.	Язык С++, многомерные массивы, динамические массивы.
12.	Язык С++, определение, описание и вызов функций, рекурсивные функции.
13.	Язык С++, перегрузка функций.
14.	Язык С++, структуры и объединения.
15.	Язык С++, понятие класса, методы и члены класса.
16.	Язык С++, класс, конструктор и деструктор, доступность компонентов класса.
17.	Язык С++,С#. Классы, наследование, множественное наследование.
18.	Язык С++,С#. Обработка исключений.
19.	Язык С#, перегрузка функций.
20.	Язык С#, перегрузка конструктора класса.
21.	Язык С#, операторы цикла.
22.	Язык С#, структуры и объединения.
23.	Язык С#, основные типы переменных, преобразование типов, явное и неявное.
24.	Язык С#, класс, конструктор и деструктор, доступность компонентов класса.

25.	Интерфейс RS232, назначение и область применения.
26.	Интерфейс RS485, назначение и область применения.
27.	Использование интерфейса USART микроконтроллера для подключения к COM порту ПК
28.	Организация обмена данными по интерфейсу I2C.

6 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

6.1 Результаты освоения учебной дисциплины, подлежащие проверке

Контролируемые компетенции (часть компетенций)	**Результаты обучения (объекты оценивания)**	**Основные показатели оценки результатов**	**Оценочные средства**
способностью **ОПК-4.1** Способен применять современные программные средства и информационные технологии при моделировании мехатронных и робототехнических устройств	**З1** Знать синтаксические конструкций современных языков программирования	Знание синтаксических конструкций современных языков программирования C, C++, C#	лекция, практическое занятие, лабораторная работа, экзамен
	З2 Знать шаблоны проектирования высокоуровневого программного обеспечения, применяющиеся в управляющих программах мехатронных систем	Шаблоны проектирования высокоуровневого программного обеспечения, применяющиеся в управляющих программах мехатронных систем	лекция, практическое занятие, лабораторная работа
	З3 Знать основные алгоритмы управления мехатронными системами	Основные алгоритмы управления мехатронными системами	лекция, практическое занятие
	У1 Уметь применять основные методы проектирования сложных систем программного обеспечения с использованием объектно-ориентированного подхода	Использование методов проектирования сложных систем программного обеспечения с использованием объектно-ориентированного подхода	лекция, практическое занятие
	У2 Умение создавать высокоуровневые алгоритмы управления сложными мехатронными системами	Создание высокоуровневых алгоритмов управления сложными мехатронными системами	практическое занятие
способностью разрабатывать экспериментальные макеты управляющих, информационных и исполнительных модулей мехатронных и робототехнических систем и проводить их исследование с применением современных информационных	**В1** Владеть навыками применения базовых алгоритмов управления мехатронными системами	Применение базовых алгоритмов управления мехатронными системами	практическое занятие
	В2 Владеть навыками работы в комплексных средах создания программного обеспечения	Работа в комплексных средах создания программного обеспечения	лекция, практическое занятие, лабораторная работа, экзамен
	В3 Владеть навыками написания алгоритмов и программ на языках программирования	Написание алгоритмов и программ на языках программирования высокого уровня C++, C#	лекция, практическое занятие, лабораторная работа, экзамен

технологий (ПК-3).	высокого уровня		
	В4 Владеть навыками проектирования сложных систем с использованием объектно-ориентированного подхода	Проектирование сложных систем с использованием объектно-ориентированного подхода	лекция, практическое занятие, лабораторная работа, экзамен

6.2 Шкала оценивания планируемых результатов обучения

6.2.1 Текущий и рубежный контроль

В рамках текущего и рубежного контроля по дисциплине студент может набрать до 70 баллов

Семестр	Шкала оценивания			
	0-35 баллов	36-50 баллов	51-60 баллов	61-70 баллов
3	Частичное посещение аудиторных занятий. Неудовлетворительное выполнение лабораторных и практических работ. Плохая подготовка к балльно-рейтинговым мероприятиям. Студент не допускается к промежуточной аттестации	Полное или частичное посещение аудиторных занятий. Частичное выполнение и защита лабораторных и практических работ. Выполнение контрольных работ, тестовых заданий на оценки «удовлетворительно».	Полное или частичное посещение аудиторных занятий. Полное выполнение и защита лабораторных и практических работ. Выполнение контрольных работ, тестовых заданий на оценки «хорошо».	Полное посещение аудиторных занятий. Полное выполнение и защита лабораторных и практических занятий. Выполнение контрольных работ, тестовых заданий на оценки «отлично».

6.2.2 Промежуточная аттестация

Оценка результатов освоения учебной дисциплины в 6 семестре проводится по шкале, используемой на экзамене:

Семестр	Шкала оценивания			
	Неудовлетворительно (36-60 баллов)	Удовлетворительно (61-80 баллов)	Хорошо (81-90 баллов)	Отлично (91-100 баллов)
3	Студент имеет 36-60 баллов по итогам текущего и рубежного контроля, на экзамене не дал полного ответа ни на один вопрос. Студент имеет 36-45 баллов по итогам текущего и	Студент имеет 36-50 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос и частично (полностью)	Студент имеет 51-60 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос и частично (полностью)	Студент имеет 61-70 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос и частично

	рубежного контроля, на экзамене дал полный ответ только на один вопрос	ответил на второй. Студент имеет 46-60 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос или частично ответил на оба вопроса. Студент имеет по итогам текущего и рубежного контроля 61-70 баллов на экзамене не дал полного ответа ни на один вопрос.	ответил на второй. Студент имеет 61 – 65 баллов по итогам текущего и рубежного контроля, на экзамене дал полный ответ на один вопрос и частично ответил на второй. Студент имеет 66-70 баллов по итогам текущего и рубежного контроля, на экзамене) дал полный ответ только на один вопрос.	(полностью) ответил на второй.

7 Учебно-методическое обеспечение дисциплины (модуля)

7.1 Основная литература

1. Шилдт Г. Полный справочник по C#. –М.: «Вильямс», 2004. -752 с.
1. Подбельский В.В. Язык C++. –М.: Финансы и статистика, 2007. -560 с.
2. Брагин В.Б., Войлов Ю.Г., Жаботинский Ю.Д., и др. Системы очувствления и адаптивные промышленные роботы. –М.: Машиностроение, 1985. -256 с.
3. Куафе Ф. Взаимодействие робота с внешней средой: Пер. с франц. –М.: Мир, 1985. -285 с.

7.2 Дополнительная литература

1. Яценков В.С. Основы спутниковой навигации. Системы GPS NAVSTAR и ГЛОНАС. – М.: Телеком, 2005. – 272 с.
2. Жданов А.А. Автономный искусственный интеллект/ -М.: Бином. 2008. -359с.
3. Интеллектуальные роботы: учебное пособие для вузов/ под ред. Е.И. Юревича. –М.: Машиностроение, 2007 -360с.
4. Робототехнические системы и комплексы: Учеб. Пособие для вузов/ Мачульский И.И, Запятой В.П., Майоров Ю.П. и др. М.: Транспорт 1999. 446 с.

7.3 Интернет-ресурсы

1. Wikipedia – свободная энциклопедия. - http://ru.wikipedia.org/.
2. http://www.atmel.com
3. https://msdn.microsoft.com
4. http://www.iprbookshop.ru/

7.4 Методические указания к лабораторным занятиям

Комплект учебного оборудования «Микропроцессорные системы управления электроприводов» ПО1033 Методические указания к выполнению лабораторных работ. ООО ТД «ПрофОбразование» 2015.

7.5 Программное обеспечение современных информационно-коммуникационных технологий. Программное обеспечение

1. Microsoft Windows.
2. Пакет Microsoft Office.
3. Программные продукты: Atmel Studio.
4. Программный продукт: Microsoft Visual Studio 2010

8 Материально-техническое обеспечение дисциплины

Требования к условиям реализации дисциплины:

№ п/п	Вид аудиторного фонда	Требования
1.	Лекционная аудитория	Оснащение специализированной учебной мебелью. Оснащение техническими средствами обучения: настенный экран с дистанционным управлением, мультимедийное оборудование.
2.	Кабинет для практических занятий	Оснащение специализированной учебной мебелью. Оснащение техническими средствами обучения: подвижная маркерная доска, считывающее устройство для передачи информации в компьютер; настенный экран с дистанционным управлением, мультимедийное оборудование.
3.	Компьютерные классы	Оснащение специализированной учебной мебелью. Оснащение техническими средствами обучения: ПК с возможностью подключения к локальным сетям и Интернету. Наличие ВТ из расчета один ПК на два студента.

Перечень материально-технического обеспечения дисциплины:

№ п/п	Вид и наименование оборудования	Вид занятий	Краткая характеристика
1.	IBM PC - совместимые персональные компьютеры.	Практические занятия.	Процессор серии не ниже Pentium IV. Оперативная память не менее 512 Мбайт. ПК должны быть объединены локальной сетью с выходом в Интернет.
2.	Мультимедийные средства.	Лекционные и практические занятия.	Демонстрация с ПК электронных презентаций, документов Word, электронных таблиц, графических изображений.

№ работ	Материальное обеспечение лабораторных занятий
1	2
	1. Комплект учебного оборудования «Микропроцессорные системы управления электроприводов». 2. Микроконтроллеры фирмы Atmel. 3. Программаторы для микроконтроллеров.

	4. Макетная плата для микроконтроллера, датчики различных параметров, средства индикации, шаговые электродвигатели и электродвигатели постоянного тока.

9 Особенности реализации дисциплины для инвалидов и лиц с ограниченными возможностями здоровья

Для студентов с ограниченными возможностями здоровья созданы специальные условия для получения образования. В целях доступности получения высшего образования по образовательным программам инвалидами и лицами с ограниченными возможностями здоровья университетом обеспечивается:

1. Альтернативная версия официального сайта в сети «Интернет» для слабовидящих;

2. Для инвалидов с нарушениями зрения (слабовидящие, слепые)

- присутствие ассистента, оказывающего обучающемуся необходимую помощь, дублирование вслух справочной информации о расписании учебных занятий; наличие средств для усиления остаточного зрения, брайлевской компьютерной техники, видеоувеличителей, программ невизуального доступа к информации, программ-синтезаторов речи и других технических средств приема-передачи учебной информации в доступных формах для студентов с нарушениями зрения;
- задания для выполнения на экзамене зачитываются ассистентом;
- письменные задания выполняются на бумаге, надиктовываются ассистенту обучающимся;

3.Для инвалидов и лиц с ограниченными возможностями здоровья по слуху (слабослышащие, глухие):

- на зачете/экзамене присутствует ассистент, оказывающий студенту необходимую техническую помощь с учетом индивидуальных особенностей (он помогает занять рабочее место, передвигаться, прочитать и оформить задание, в том числе записывая под диктовку);
- зачет/экзамен проводится в письменной форме;

4. Для инвалидов и лиц с ограниченными возможностями здоровья, имеющих нарушения опорно-двигательного аппарата, созданы материально-технические условия обеспечивающие возможность беспрепятственного доступа обучающихся в учебные помещения, объекту питания, туалетные и другие помещения университета, а также пребывания в указанных помещениях (наличие расширенных дверных проемов, поручней и других приспособлений).

- письменные задания выполняются на компьютере со специализированным программным обеспечением или надиктовываются ассистенту;
- по желанию студента экзамен проводится в устной форме.

Обучающиеся из числа лиц с ограниченными возможностями здоровья обеспечены электронными образовательными ресурсами в формах, адаптированных к ограничениям их здоровья.