Содержание

[Введение 5](#_Toc7801055)

[1 Общетехническое обоснование разработки устройства 6](#_Toc7801056)

[1.1 Анализ исходных данных 6](#_Toc7801057)

[1.1.1 Назначение изделия. 6](#_Toc7801058)

[1.1.2 Основание для разработки 6](#_Toc7801059)

[1.1.3 Технические требования 6](#_Toc7801060)

[1.1.4 Конструктивно-технологические требования 6](#_Toc7801061)

[1.2 Теоретические сведения и принципы функционирования отдельных узлов устройства 6](#_Toc7801062)

[1.2.1 Протокол и структура пакета Modbus RTU 6](#_Toc7801063)

[1.2.2 Перечень и формат команд Modbus RTU 7](#_Toc7801064)

[1.2.3 Структура микроконтроллера семейства *ARMCortex-M4* 7](#_Toc7801065)

[1.2.4 Регистровая модель портов *GPIO* микроконтроллера семейства *ARMCortex-M4* 9](#_Toc7801066)

[1.2.5 Физический, канальный и сеансовый уровни интерфейсов *I2C*, *I2S*, *SPI* и *SDIO* 10](#_Toc7801067)

[1.2.6 Регистровые модели *I2C*, *I2S*, *SPI* и *SDIO* микроконтроллера семейства *ARMCortex-M4* 13](#_Toc7801068)

[1.2.7 Принципы функционирования блока *DMA* прямого доступа к памяти 14](#_Toc7801069)

[1.2.8 Структура и логика функционирования дисплейного модуля *HY32D* на базе видеопроцессора *ILI9341* 15](#_Toc7801070)

[1.2.9 Структура и логика функционирования контроллеров *ADS7846* и *STMPE811* сенсорной панели 15](#_Toc7801071)

[1.2.10 Методика создания пользовательского интерфейса 17](#_Toc7801072)

[1.2.11 Принципиальные основы и схемы зарядки литий-ионных аккумуляторных батарей 18](#_Toc7801073)

[1.2.12 Структура и логика функционирования микросхем *LTC4058* и *BQ24295* зарядки аккумуляторных батарей 18](#_Toc7801074)

[2 Разработка структурной электрической схемы мобильного Ethernet UDP-сервера с поддержкой протокола Modbus RTU 20](#_Toc7801075)

[2.1 Обоснование базовых составляющих структурной схемы 20](#_Toc7801076)

[2.2 Обоснование связей структурной схемы 20](#_Toc7801077)

[3 Разработка принципиальной электрической схемы мобильного Ethernet UDP-сервера с поддержкой протокола Modbus RTU 22](#_Toc7801078)

[3.1 Обоснование выбора САПР для разработки принципиальной схемы 22](#_Toc7801079)

[3.2 Описание используемых библиотечных элементов и процесса их создания 23](#_Toc7801080)

[3.2.1 Блок *PHY-Ethernet LAN8720* 24](#_Toc7801081)

[3.2.2 Соединитель штыревой *PLD-34* 25](#_Toc7801082)

[3.2.3 *MicroUSB* 25](#_Toc7801083)

[3.2.4 Блок зарядки аккумуляторной батареи *BQ24295* 26](#_Toc7801084)

[3.2.5 Аккумуляторная батарея 27](#_Toc7801085)

[3.2.6 Подсветка дисплея *CAT6219* 27](#_Toc7801086)

[3.2.7 Микроконтроллер *ARM Cortex-M4* 28](#_Toc7801087)

[3.2.8 Дисплей *HY32D* на базе микроконтроллера *ILI9341* 29](#_Toc7801088)

[3.3 Обоснование выбора компонентов принципиальной схемы 30](#_Toc7801089)

[3.4 Обоснование связей принципиальной электрической схемы 30](#_Toc7801090)

[3.5 Анализ и обоснование принципиальной электрической схемы зарядки аккумуляторной батареи мобильного устройства 31](#_Toc7801091)

[4 Разработка алгоритма функционирования мобильного Ethernet UDP-сервера с поддержкой протокола Modbus RTU 32](#_Toc7801092)

[5 Разработка конструкции проектируемого изделия 33](#_Toc7801093)

[5.1 Выбор и обоснование элементной базы 33](#_Toc7801094)

[5.2 Выбор и обоснование конструктивных элементов и установочных изделий 34](#_Toc7801095)

[6 Расчёт конструктивно-технологических параметров проектируемого изделия 35](#_Toc7801096)

[6.1 Проектирование печатного модуля 35](#_Toc7801097)

[6.2 Выбор и обоснование материалов конструкции и защитных покрытий, маркировки деталей и сборочных единиц 36](#_Toc7801098)

[7 Применение средств автоматизированного проектирования при разработке устройства 38](#_Toc7801099)

[Заключение 39](#_Toc7801100)

[Список использованных источников 40](#_Toc7801101)

Введение

Целью данного курсового проекта является разработка печатной платы мобильного устройства обмена данными через *Ethernet* по протоколу *MODBUS RTU*. В ходе разработки данной печатной платы предполагается создание программного кода работы устройства, написанного на языке высокого уровня. Необходимой частью разработки печатной платы является также создание чертежей для последующего её производства.

В процессе разработки печатной платы были использованы различные САПР, обоснование выбора и применение которых более детально будет описано далее.

Данная тема является очень актуальной на сегодняшний день, так как общемировые тенденции свидетельствуют о том, что повсеместно труд человека заменяется на машинный. Огромное количество промышленных, производственных и даже бытовых процессов в настоящий момент уже автоматизировано, но некоторая часть пока остается традиционной. Как раз таки протокол передачи данных *MODBUS RTU*используется при построении компьютерных систем автоматизации. Платформа *Ethernet*на сегодняшний день имеет повсеместное распространение и использование. Она отличается простотой использования, а также дешевизной.

Разрабатываемое в данной курсовой работе мобильное устройство базируется на микроконтроллере и имеет потенциальную возможность быть примененным при автоматизации слежения за освещением в жилом доме либо любом другом здании. Данное устройство имеет возможность считывать состояния своих портов, к которым подсоединены датчики света. В свою очередь оператор мобильного устройства может получить все необходимые данные с этого сервера, подключившись к нему через интерфейс *Ethernet*и считав данные по протоколу *MODBUS RTU*.

Задачей данной работы является разработка учебного проекта, который в последствии может быть доработан и произведен для выполнения поставленных задач.

# Общетехническое обоснование разработки устройства

## Анализ исходных данных

### Назначение изделия.

Устройство предназначено для организации Ethernet UDP-сервера с поддержкой протокола Modbus RTU.

### Основание для разработки

Основанием для разработки устройства является задание к курсовому проекту.

### Технические требования

Разрабатываемоемобильное устройство должно иметь:

* основное питание от аккумулятора напряжением 3,6В
* потребляемый ток не более 100 мА.
* возможность зарядки аккумулятора от внешнего источника питания напряжением 5,0В через разъём*microUSB*.

### Конструктивно-технологические требования

Конструкция устройства должна обеспечивать свободный доступ к составным элементам изделия при проведении пуско-наладочных и ремонтных работ.

Материалы и комплектующие изделия должны применяться по действующим стандартам и техническим условиям на них.

Показатели технологичности конструкции изделия должны соответствовать ГОСТ 14.201-73.

Конструктивно, мобильный MODBUSRTUсервер выполнен в виде двух модулей, один из которых содержит электрические радиоэлементы, а другой – экран для вывода результата.

Габариты устройства – не более 100 х 80 x 40 мм.

Масса модуля не более 0,3 кг.

## Теоретические сведения и принципы функционирования отдельных узлов устройства

### Протокол и структура пакета Modbus RTU

Modbus — открытый [коммуникационный протокол](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), основанный на архитектуре [ведущий — ведомый](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B4%D1%83%D1%89%D0%B8%D0%B9_%E2%80%94_%D0%B2%D0%B5%D0%B4%D0%BE%D0%BC%D1%8B%D0%B9) (*master-slave*). Широко применяется в промышленности для организации [связи между электронными устройствами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Может использоваться для передачи данных через последовательные линии связи [*RS-485*](https://ru.wikipedia.org/wiki/RS-485)*,*[*RS-422*](https://ru.wikipedia.org/wiki/EIA-422)*,*[*RS-232*](https://ru.wikipedia.org/wiki/RS-232), и сети [*TCP/IP*](https://ru.wikipedia.org/wiki/TCP/IP)*(Modbus TCP)*. Также существуют нестандартные реализации, использующие [UDP](https://ru.wikipedia.org/wiki/UDP).

Сообщение *Modbus RTU* состоит из адреса устройства *SlaveID*, кода функции, специальных данных в зависимости от кода функции и *CRC* контрольной суммы. Если отбросить *SlaveID* адрес и *CRC* контрольную сумму, то получится *PDU, Protocol Data Unit. SlaveID* – это адрес устройства, может принимать значение от 0 до 247, адреса с 248 до 255 зарезервированы.

На рисунке 1.1 приведен пример запроса *Modbus RTU* для получения значения аналогового выхода.

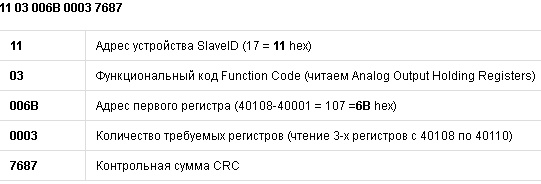


Рисунок 1.1– Пример запроса *Modbus RTU*

### Перечень и формат команд Modbus RTU

На рисунке 1.2 изображен перечень команд *Modbus RTU* с кодами функций чтения и записи регистров[1].



Рисунок 1.2– Перечень команд*Modbus RTU*

### Структура микроконтроллера семейства *ARMCortex-M4*

Общие характеристики семейства:

1. ARM 32-bitCortex-M4 CPU.

2. Частота тактирования 168 МГц.

3. Напряжение питания 1,8…3,6 В.

4. Коммуникационные интерфейсы: I2C, USART (ISO 7816, IrDA), SPI, I2S.

5. USB 2.0 FS/HSOTG.

6. 10/100 Ethernet MAC (IEEE 1588v2, MII/RMII).

7. Контроллер SDIO (карты SD, SDIO, MMC).

8. FSMC-контроллер (Compact Flash, SRAM, PSRAM, NOR, NAND и LCD 8080/6800).

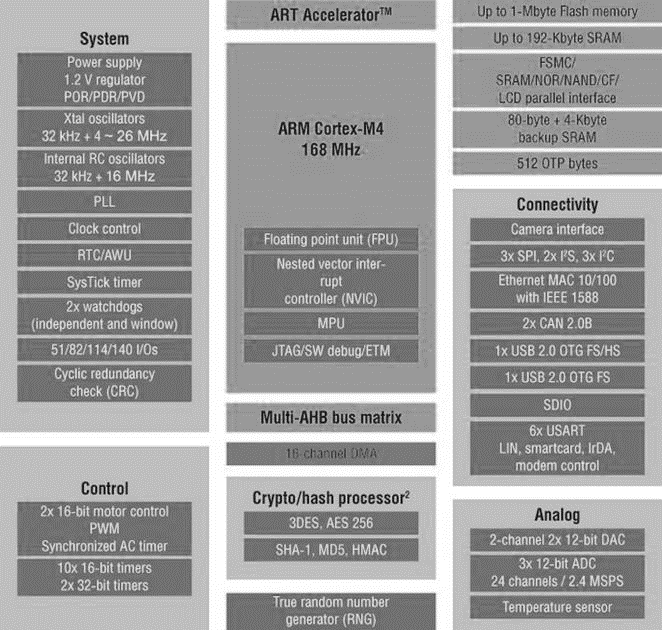


Рисунок 1.3– Структура микроконтроллеров семейства ARM Cortex-M4

Встроенные интерфейсы коммуникации:

**Ethernet*.*** Блок выполнен в строго по стандарту IEEE802.3. Возможна передача данных со скоростью 10/100 Мбит/с. Доступна синхронизация часов для чего протокол IEEE1588 v2 реализован аппаратно.

**USB (UniversalSerialBus)**. На борту присутствует два раздельных блока USB. Первый – USBOTGfull-speed, является полностью аппаратной реализацией и совместим со стандартами USB 2.0, а также OTG 1.0. Работает на скорости до 12 Mбит/с. Второй – USBOTGhigh-speed работает в режиме Host/Device/OTG с высокой скоростью 480 Мбит/с.

**SDIO (SecureDigitalInput/Output)**. Интерфейс позволяет работать с картами SD/SDIO/MMC-картами памяти и дисковыми контроллерами CE-ATA.

**SPI (SerialPeripheralInterface)**. Устройство оснащено тремя блоками SPI, каждый из которых работает в режиме Мaster (Мultimaster) либо в режиме Slave, передавая данные полудуплексно, полнодуплексно либо симплексно.  
**I2C (Inter-IntegratedCircuit)**. На борту МК содержится три блока I2C, поддерживающих работу в режиме Master/Slave (ведущий или ведомый), а также в режиме Мultimaster (режим в котором на шине присутствуют несколько Master-устройств, разделяющих общие ресурсы Slave, либо поочередно изменяющих свое состояние с Master на Slave и обратно).

***FSMC (Flexible Static Memory Controller)***. Блок используется для подключения жидкокристаллических дисплеев либо внешней памяти напрямую.

### Регистровая модель портов *GPIO* микроконтроллера семейства *ARMCortex-M4*

*GPIO* — это порты ввода-вывода общего назначения. Через них микроконтроллер по сути связан с внешним миром.

**Регистры конфигурации**

Эти регистры позволяют задавать режим работы каждой ножки отдельно. Всего их 4 для каждого порта:

*GPIOx*\_*MODER*. Этот регистр задает направление ввода-вывода каждой ножки. Направление может быть: вход, выход, альтернативная ф-я, аналоговый.

*GPIOx*\_*OTYPER*. Этот регистр задает тип выхода: двухтактный или открытый сток. По умолчанию — двухтактный для каждой ножки

*GPIOx*\_*OSPEEDR*. С помощью этого регистра задается частота тактирования каждого вывода. СР описывает частоты как: низкая, средняя, повышенная, высокая.

*GPIOx*\_*PUPDR*. Этот регистр управляет подтяжкой каждой ножки. Нам предлагают на выбор: без подтяжки, подтяжка к питанию, подтяжка к земле.

**Регистры данных**

*GPIOx*\_*IDR*. Регистр ввода.

*GPIOx*\_*ODR*. Регистр вывода.

**Регистр установки/сброса**

*GPIOx*\_*BSRR*. При помощи этого регистра осуществляется запись 0 или 1 в регистр вывода *GPIOx*\_*ODR*.

**Регистр блокировки**

*GPIOx*\_*LCKR*. Этот регистр осуществляет управление механизмом блокировки конфигурации. В заблокированном состоянии доступна работа с регистром установки/сброса(*GPIOx*\_*BSRR*) и регистром вывода(*GPIOx*\_*ODR*).

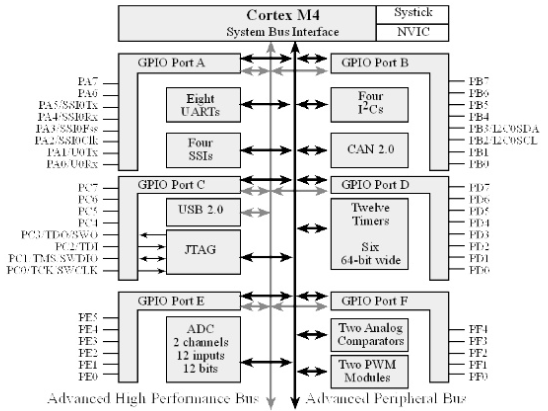


Рисунок 1.4 – Регистровая модель GPIO

### Физический, канальный и сеансовый уровни интерфейсов *I2C*, *I2S*, *SPI* и *SDIO*

При использовании интерфейса*I2C* данные передаются по двум проводам — провод данных и провод тактов. Есть **ведущий**(*master*) и **ведомый**(*slave*), такты генерирует *master*, ведомый лишь поддакивает при приеме байта. Всего на одной двухпроводной шине может быть **до 127 устройств**. Данные шлются пакетами, каждый пакет состоит из девяти бит. 8 данных и 1 бит подтверждения/не подтверждения приема. **Первый пакет** шлется от ведущего к ведомому это **физический адрес устройства и бит направления**.

|  |
| --- |
| http://easyelectronics.ru/img/starters/IIC/IIC-SLA.GIF |
| Рисунок 1.5 – Служебный пакет |
| Сам адрес состоит из**семи бит**,а**восьмой бит** означает, что будет делать ***Slave*** на следующем байте — **принимать или передавать данные**. Девятым битом идет бит подтверждения ***ACK***. Если *Slave* услышал свой адрес и считал полностью, то на девятом такте он придавит линию ***SDA*** в 0, сгенерировав ***ACK*** — то есть «Понял!». Если ***Slave*** не обнаружился, прозевал адрес, неправильно принял байт, сгорел или еще что с ним случилось, то, соответственно, ***SDA*** на девятом такте будет прижать некому и ***ACK*** не получится. Будет ***NACK***. Мастер с горя хлопнет водки и прекратит свои попытки до лучших времен. После адресного пакета идут **пакетысданными** в ту или другую сторону, в зависимости от **бита*RW*** в заголовочном пакете  http://easyelectronics.ru/img/starters/IIC/I2C-2byte.GIF |

Рисунок 1.6 – Запись по интерфейсу I2C

При приеме последнего байта надо дать ведомому понять, что в его услугах больше не нуждаемся и отослать *NACK* на последнем байте. Если отослать *ACK* то после стопа *Master* не отпустит линию — такой уж там конечный автомат. Так что прием двух байтов будет выглядеть так (**R=1**):

|  |
| --- |
| http://easyelectronics.ru/img/starters/IIC/I2C-SAV.GIF |
| Рисунок 1.7 – Чтение по интерфейсу I2C |

Есть еще одно состояние, как **повторныйстарт**.  
Это когда мы не обьявляя ***STOP*** вкатываем на шину еще один ***START***. После него мы можем обратиться к другому устройству не освобождая шину. Но чаще идет обращение к тому же самому устройству и это связано с особенностями организации памяти.

Интерфейс *I2S* электрически представляет собой 4 проводника, которые идут от активного устройства к пассивному, а также 4 сигнала, им соответствующие:

* Тактовый сигнал битовой синхронизации
* Тактовый сигнал фреймовой синхронизации
* Сигнал принимаемых данных
* Сигнал передаваемых данных

Каналы для приема и передачи данных разделены, то есть есть канал для приема данных и канал для передачи данных. Контроллер принимает данные, передаваемые звуковым кодеком, но возможна и обратная ситуация.

В SPI используются четыре цифровых сигнала:

* *MOSI* — выход ведущего, вход ведомого
* *MISO* — вход ведущего, выход ведомого
* *SCLK* — последовательный тактовый сигнал
* *CS* или *SS* — выбор микросхемы, выбор ведомого

Частота следования битовых интервалов в линиях передачи данных определяется синхросигналом *SCK*, который генерирует ведущее устройство, ведомые устройства используют синхросигнал для определения моментов изменения битов на линии данных, при этом ведомые устройства никак не могут влиять на частоту следования битовых интервалов. Как в ведущем устройстве, так и в ведомом устройстве имеется счетчик импульсов синхронизации (битов). Счетчик в ведомом устройстве позволяет последнему определить момент окончания передачи пакета. Счетчик сбрасывается при выключении подсистемы *SPI*, такая возможность всегда имеется в ведущем устройстве. В ведомом устройстве счетчик обычно сбрасывается деактивацией интерфейсного сигнала *SS*.

Передача осуществляется пакетами. Длина пакета, как правило, составляет 1 байт (8 бит), при этом известны реализации *SPI* с иной длиной пакета, например, 4 бита. Ведущее устройство инициирует цикл связи установкой низкого уровня на выводе выбора подчиненного устройства (SS) того устройства, с которым необходимо установить соединение.

Самыми важными линиями в шине интерфейса *SDIO* являются:

* *CLK –*тактированиекарты*.*
* *CMD* - по этой линии передаются команды.
* *DAT0* - линия данных (в случае 4-ех битной шины их будет 4).

Все посылки карточке и обратно есть последовательности битов, строго синхронизированные с тактовым сигналом, передаваемым по линии CLK. Рекомендуемые частоты описаны в спецификации на карту и имеют различное значение, в зависимости от ее типа и класса скорости. Отметимтолько, что для любой карты инициализация проходит на очень малой (по сравнению с передачей данных) частоте. Шина данных может быть 1-битной (работает только *D0*) или 4-битной – это конфигурируется при инициализации. Важно, что для *SD* карт со стороны хоста линии данных и команд должны быть *Push*-*Pull* и быть подтянуты к питанию через резисторы 4.5 – 10 кОм. Тактовую шину тоже нужно подтянуть к питанию.

### Регистровые модели *I2C*, *I2S*, *SPI* и *SDIO* микроконтроллера семейства *ARMCortex-M4*

Под регистровой моделью понимается описание набора регистров процессора, их разрядности, способа организации, методов доступа, основных особенностей и пр. Ниже приведены списки основных регистров для данных интерфейсов:

Регистровая модель *I2C*:

* *I2C\_DR – I2C data register*
* *I2C\_OAR1 – I2C own address register 1*
* *STM32 I2C – control register 1 - I2C\_CR1*
* *STM32 I2C – control register 2 - I2C\_CR2*
* *STM32 I2C – status register 1 - I2C\_SR1*
* *STM32 I2C – status register 2 - I2C\_SR2*
* *I2C clock control register - I2C\_CCR*

Регистровая модель I2S*:*

* Регистры флагов прерывания;
* Регистр разрешения прерываний от того или иного источника (по числу флагов прерывания или по числу разрядов регистра разрешения прерываний)
* Регистр управления, в котором задаются режимы работы контроллера;
* Регистр генератора частоты выборки, в котором можно задавать тактовый сигнал и его частоту для битовой синхронизации — если данные принимаются, то регистр записывает эти данные и те могут быть подсчитаны программно;
* Два регистра приема данных;
* Регистры передачи данных по каналам, в которых могут быть два 32-х битовых регистра, передаваемых последовательно.

Регистровая модель *SPI*:

* *SPI Control Register (SPCR)*
* *SPI Status Register (SPSR)*
* *SPI Data Register (SPDR)*

Регистровая модель *SDIO*:

* *OCR Register (32 bits)*
* *CID Register (128 bits)*
* *CSD Register (128 bits)*
* *RCA Register (16 bits)*
* *DSR Register (16 bits, optional)*
* *SCR Register (64 bits)*
* *SD\_CARD\_STATUS (512 bits)*

### Принципы функционирования блока *DMA* прямого доступа к памяти

Независимо от наличия или отсутствия у центрального процессора ввода-вывода, отображаемого на пространство памяти, ему необходимо обращаться к контроллерам устройств, чтобы осуществлять с ними обмен данными. Центральный процессор может запрашивать данные у контроллера ввода-вывода побайтно, но при этом будет нерационально расходоваться его рабочее время, поэтому чаще всего используется другая схема, которая называется прямым доступом к памяти — *DMA* (*DirectMemoryAccess*). Операционная система может использовать *DMA* только при наличии аппаратного *DMA*-контроллера, присутствующего у большинства систем. Иногда этот контроллер встроен в контроллеры дисков и другие контроллеры, но такая конструкция требует отдельного *DMA*-контроллера для каждого устройства. Чаще всего для упорядочения обмена данными с несколькими устройствами, проводимого нередко в параллельном режиме, доступен только один *DMA-*контроллер (размещенный, к примеру, на системной плате).

*DMA*-контроллер ни находился физически, он имеет доступ к системной шине независимо от центрального процессора. В нем имеется несколько регистров, доступных центральному процессору по чтению и записи. В их число входят регистр адреса памяти, регистр счетчика байтов и один или несколько регистров управления. В регистрах управления указывается используемый порт ввода-вывода, направление передачи данных (чтение из устройства ввода-вывода или запись в него), единица передаваемой информации (побайтовая или пословная передача), а также количество байт, передаваемых в одном пакете.

Прямой доступ к памяти осуществляется следующим образом. Сначала центральный процессор программирует *DMA*-контроллер, устанавливая значения его регистров таким образом, чтобы он знал, что и куда нужно передать. Он также выдает команду контроллеру диска на чтение данных с диска во внутренний буфер контроллера и на проверку контрольной суммы. После того как в буфере контроллера окажутся достоверные данные, к работе может приступать *DMA*.

### Структура и логика функционирования дисплейного модуля *HY32D* на базе видеопроцессора *ILI9341*

Дисплей содержит микроконтроллер, на базе которого осуществляется связь дисплея с помощью интерфейса *FSMC*. Кроме того, в дисплее встроен контроллер *TouchScreen*, отвечающий за определение места, где проходит электрический импульс для срабатывания нажатия на дисплей и передающий данные по интерфейсу *SPI*.

В дисплее установлен *TFT*-экран для управления активной матрицей на жидких кристаллах, содержащий 65536 цветов и имеющий подсветку экрана на основе чипа *CAT6219*.

Разрешение дисплея составляет 320 x 240, то есть 76800 пикселей, а размер дисплея 3.2 дюйма.

Логика функционирования заключается в том, что после инициализации интерфейса *FSMC* через него, по интерфейсу *SPI*, осуществляется связь микроконтроллеров *STM32F415VGT6* и *ILI9341*, тем самым сопоставляя передающиеся данные между ними.

### Структура и логика функционирования контроллеров *ADS7846* и *STMPE811* сенсорной панели

*ADS7846* является новой версией промышленного стандарта 4-х проводного контроллера сенсорных экранов. *ADS7846* содержит на кристалле 2.5В источника опорного напряжения (ИОН), который может быть подключен к дополнительному входу, монитору напряжения батареи, или для температурных измерений. ИОН может быть отключен, в случае ненадобности, для сохранения энергии.

Внутренний ИОН может работать при снижении напряжении питания до 2.7 В и непосредственно измерять напряжение батареи от 0 В до 6 В. Минимальное потребление мощности менее 0.5 мВТ при 2.7 В ( ИОН- выключен), на максимальной скорости 125 КГц, и встроенный драйвер экранного сенсора делают *ADS7846*

Идеальным для использования в батарейных системах таких как персональные цифровые органайзеры с резистивным сенсорным экраном, пейджеры, мобильные телефоны, и другое портативное оборудование. *ADS7846* изготавливаются в миниатюрных корпусах *TSSOP*-16, *SSOP*-16, *VFBGA*-48 и аттестованы в диапазоне температур от -40°С до +85°С.

Сенсорные экраны состоят из отображающего экрана, на котором формируется изображение органов управления и другая информация, и прозрачной сенсорной панели, устанавливаемой перед экраном.

Сенсорные панели позволяют определять координаты точки прикосновения в любой области экрана.

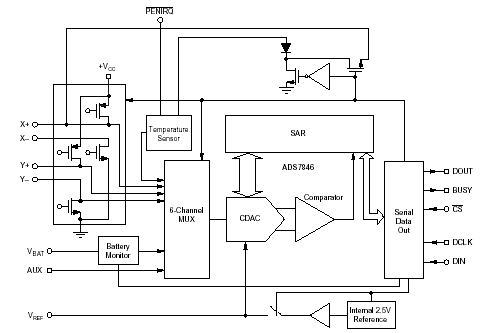


Рисунок 1.8 – Типовая схема включения

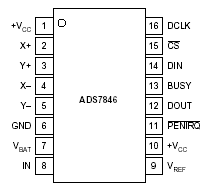


Рисунок 1.9 – Расположение выводов

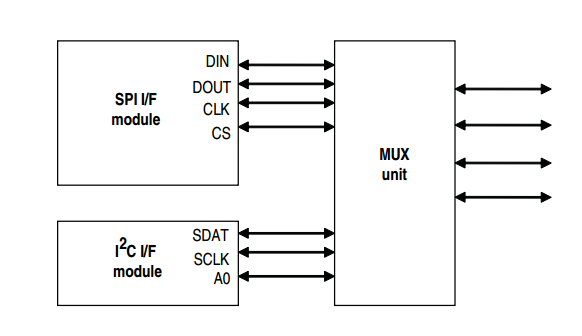


Рисунок 1.10– Интерфейс STMPE811 сенсорной панели

В зависимости от этих координат компьютер принимает решение об активации какого-либо органа управления, изображенного на экране (принцип, аналогичный действию компьютерной мыши).

И на панель, и на мембрану нанесено резистивное покрытие. Пространство между стеклом и мембраной заполнено микроизоляторами, которые равномерно распределены по активной области экрана и надежно изолируют проводящие поверхности. Когда на экран нажимают, панель и мембрана на небольшой окрестности замыкаются, и контроллер с помощью аналогово-цифрового преобразователя регистрирует изменение сопротивления и преобразует его в координаты прикосновения (X и Y).

В общих чертах алгоритм считывания следующий. На верхний электрод подается напряжение питания, а нижний заземляется. Левый и правый электроды соединяются, и на них проверяется напряжение. Это напряжение соответствует Y-координате экрана.

Аналогично на левый и правый электрод подаются питание и «земля», с верхнего и нижнего электродов считывается X-координата.

### Методика создания пользовательского интерфейса

Можно рассматривать два совершенно разных метода создания интерфейсов.

1) Вручную. В таком подходе пишется код, который отвечает за создание элементов интерфейса, обрабатывающие пользовательские события. Это не всегда быстро, но такой способ дает максимальный контроль создания интерфейса и обработки событий.

2) Графической редактор. В данном подходе используется программный продукт, который позволяет нарисовать интерфейс, мгновенно наблюдая то, как он будет отображаться. Данный метод гораздо нагляднее, но почти всегда менее гибок.

Классический подход проектирования пользовательского интерфейса

На основе собранных пользовательских и иных требований проектировщики создают макеты будущего интерфейса в виде так называемых прототипов, которые графически представляют внешний вид интерфейса. Неотъемлемой частью прототипа является описание поведения интерфейса, возникающее в процессе взаимодействия пользователя с продуктом, либо эмуляция поведение продукта.

На основе такой спецификации дизайнеры создают графический стиль продукта, а разработчики его реализуют. У каждого участника разработки имеется собственная зона ответственности и компетенции.

Данный подход отлично работает для обычных веб-приложений и для классических настольных (*GUI*) приложений, где давно устоялись свои модели взаимодействия, построенные на основе фиксированного набора элементов управления.

Следующим подходом к построению пользовательских интерфейсов является концепция *User-Centered Design* - дизайн вокруг пользователя.

Данный подход основан на концепции выстраивания дизайна на основе нужд конечного пользователя. Это позволяет оптимизировать взаимодействие человека и системы, сделать работу более эффективной. Здесь не человек адаптируется к информационной системе, а информационная система - под человека.

### Принципиальные основы и схемы зарядки литий-ионных аккумуляторных батарей

Из официальной документации на микроконтроллер *BQ24295* были подключены основные источники питания, показанные на рисунке 1.11.

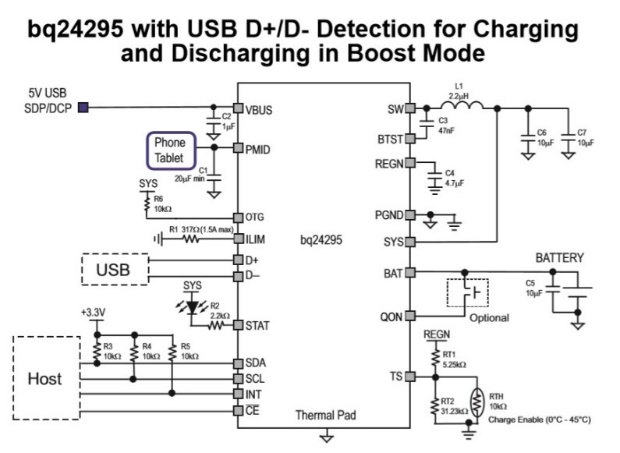


Рисунок 1.11 – Схема подключения аккумуляторной батареи

Подключено USB, от которого будет происходит зарядка устройства, и аккумуляторная батарейка на 6 В, которая будет держать заряд в течении определенного времени. Причем положительный заряд идет на вывод *BAT*, предназначенный для батарейки, а отрицательный заряд идет к земле.

### Структура и логика функционирования микросхем *LTC4058* и *BQ24295* зарядки аккумуляторных батарей

*BQ24295* - однокамерное USB-зарядное устройство с регулируемым напряжением. Его низкоимпедансный путь мощности оптимизирует работу в режиме переключения, сокращает время зарядки аккумулятора и продлевает срок службы батареи во время фазы разрядки. Последовательный интерфейс *I2C* с зарядными и системными настройками делает устройство по-настоящему гибким. Устройство поддерживает входные источники *USB* с напряжением от 3,9 до 6,2 В, включая стандартный *USB*-порт и *USB*-порт для зарядки, с защитой от перенапряжения напряжением 6,4 В. *BQ24295* соответствует спецификациям *USB* 2.0 и *USB* 3.0 с регулировкой входного тока и напряжения. Чтобы установить предел входного тока по умолчанию, bq24295 обнаруживает входной источник через детектор D + / D-, следуя спецификации зарядки аккумулятора *USB* 1.2. Кроме того, он обнаруживает нестандартные адаптеры 2A / 1A. Управление трактом питания регулирует систему немного выше напряжения батареи, но не опускается ниже минимального напряжения системы на 3,5 В. (программируется).

# Разработка структурной электрической схемы мобильного Ethernet UDP-сервера с поддержкой протокола Modbus RTU

## Обоснование базовых составляющих структурной схемы

Основными составляющими структурной схемы являются:

1. *PHY-Ethernetмодуль*.Этот блок обеспечивает работу протокола *Ethernet*на физическом уровне модели *OSI*. При помощи именно этого модуля, содержащего разъем подключения *RJ-45*, будет осуществляться взаимодействие мобильного устройства и компьютера, к которому оно будет подключено. Согласно протокола *ModbusRTU,*разработанное мобильное устройство будет являться *slave*-устройством, которое принимает запрос определенного формата от компьютера. После получения запроса устройство обязано отправить через этот же модуль ответ *master*-устройству, т.е. компьютеру.
2. *MicroUSB*. С его помощью будет происходить зарядка аккумуляторной батареи через блок зарядки аккумуляторной батареи *BQ24295*.
3. *Блок зарядки аккумуляторной батареи*. Этот элемент отвечает за саму зарядку аккумуляторной батареи. Кроме этого, блок зарядки содержит светодиод, показывающий состояние заряда батареи (заряжается – светодиод горит, не заряжается – соответственно, не горит).
4. *Микроконтроллер*. Он отвечает за логику работы всего устройства в целом и определяет предельно допустимые параметры работы с устройством.
5. *Массив светодиодов*. Этот блок является своего рода абстракцией и упрощением для реальной нагрузки устройства. Каждый светодиод симулирует освещение в определенном помещении. В данном учебном проекте микроконтроллер будет управлять включением и выключением светодиодов, что в последующей доработке проекта может быть преобразовано в управление освещением реальных помещений.
6. *Дисплей*. Он отвечает за итоговый вывод информации о том, какие из светодиодов в определенный момент времени включены, а какие неактивны.
7. *Аккумуляторная батарея*. Она осуществляет накопления заряда вследствие ее зарядки для более длительного пребывания устройства без блока зарядки.
8. *Подсветка дисплея*. Отвечает за удобное использование устройства в светлые или темные времена суток.

## Обоснование связей структурной схемы

Слот карты памяти подключается напрямую к микроконтроллеру *STM32F407VGT6* на базе *ARMCortex-M4* через шину посредством специальных регистров, находящихся в нем, тем самым обеспечивая быстрый доступ к нужным данным.

*MicroUSB* подключено напрямую к блоку зарядки аккумуляторной батареи, поскольку именно этот блок отвечает за функционирование процесса зарядки устройства. В процессе подключения зарядного устройства, сигнал идет на микроконтроллер, где он обрабатывается, и после обработки и проверки правильности подключения к микроконтроллеру, *microUSB*получает ответ в виде разрешенного соединения и начинается процесс зарядки аккумуляторной батареи.

К блоку зарядки аккумуляторной батареи идет два соединения, одно из которых получение состояния о подключении к *microUSB* зарядного устройства, а другое – сама аккумуляторная батарея, которая в случае подключенного зарядного устройства накапливает заряд для работы от батареи, а в другом случае, при отсутствии зарядного устройства, работает от самой батареи, учитывая степень ее зарядки.

Микроконтроллер является входной точкой к практически всем элементам на структурной схеме, поскольку именно он отвечает за логику функционирования составных частей устройства, обрабатывая, анализируя и определяя их разрешения на тое или иное событие.

К дисплею подключены основные составляющие для его работы: микроконтроллер по интерфейсу FSMC, от которого после обработки данных дисплей будет получать данные о светодиодах, подсветка дисплея, подключенная также по интерфейсу FSMC, обеспечивающая удобное использование дисплея, а также аккумуляторная батарея, так как дисплей без питания не сможет функционировать.

Аккумуляторная батарея подключена ко всем элементам на структурной схеме, поскольку она обеспечивает «запитывание» всего устройства в целом для его полноценного функционирования.

Подсветка дисплея подключается к самому дисплею посредством интерфейса *FSMC*.

# Разработка принципиальной электрической схемы мобильного Ethernet UDP-сервера с поддержкой протокола Modbus RTU

## Обоснование выбора САПР для разработки принципиальной схемы

*Altium Designer* – комплексная [система](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) автоматизированного проектирования ([САПР](https://ru.wikipedia.org/wiki/%D0%A1%D0%90%D0%9F%D0%A0)) радиоэлектронных средств, разработанная австралийской компанией [*Altium*](https://ru.wikipedia.org/wiki/Altium). Ранее эта же фирма разрабатывала САПР [P-CAD](https://ru.wikipedia.org/wiki/P-CAD), который приобрёл необычайную популярность среди российских разработчиков электроники.Программа в большей степени рассчитана на профессионалов, чем на радиолюбителей.

Состав программного пакета *Altium Designer* включает весь необходимый набор инструментов для создания, редактирования и правки работ на основе электрических и программируемых интегральных схем. Редактор схем позволяет работать с проектами любого размера и сложности, преобразовывая их в простейшие подблоки. Цифро-аналоговое моделирование учитывает почти все реальные параметры и предоставляет в распоряжение конструктора огромное количество различных анализов, включая анализы переходных процессов, частотный, шумов, передаточных функций, Фурье, методом *Monte-Carlo*, с изменением значений температуры. На схемотехническом уровне проверяются и устраняются различные импедансы и перекрестные отражения. Редактор печатных плат программы содержит уникальные средства для автоматического (программы *StatisticalPlacer, Cluster Placer*) и интерактивного размещения компонентов. Топологический трассировщик Situs использует полностью настраиваемый алгоритм для решения задач разводки печатных плат с большой плотностью установки элементов. Он может работать по неортогональным направлениям и с самостоятельным выбором слоев. Постоянно обновляемые библиотеки программы хранят более 90 тысяч компонентов. Многие из них имеют модели посадочных мест, *IBIS* и *SPICE*-модели, а также *3D*-модели. Каждую из них можно создать в программе самостоятельно с минимальными затратами времени путем последовательного ввода сведений о компоненте.

Основу системы *Altium Designer* составляет программная оболочка *Design Explorer*, которая интегрирует в себе различные модули, выполняющие определенные функции проектирования, например, редактор принципиальных схем, редактор печатных плат, автотрассировщик, программу моделирования, интерфейсы импорта и экспорта, *CAM* средства.

Проекты бывают четырех типов: проекты печатных плат (*PCB*), программируемой логики (*FPGA*), *VHDL* описания (*Embedded*) и интегрированные библиотеки компонентов (*Integrated Library*).

В редакторе принципиальных схем применяется несколько видов иерархии, причем один из них ранее применялся только в «тяжелых» САПР для построения многоканальных проектов. Подобные функции дают возможность пользователям избавиться от необходимости копировать подчиненные листы по числу одинаковых каналов. Достаточно нарисовать схему канала один раз и правильно связать ее с вышестоящим листом. При моделировании или передаче проекта в редактор печатных плат система автоматически размножит описанные каналы, присвоит компонентам уникальные позиционные обозначения и добавит необходимые связи. Многоканальная структура проекта сохранится и в редакторе печатных плат: все компоненты определенного канала будут автоматически привязаны к так называемой «комнате» размещения (*Room*), что облегчит их последующее размещение и трассировку связей, благодаря уникальной функции *Copy Room Format*.

Редактор схем системы *Altium Designer* работает как в дюймовой, так и метрической системах измерения. Это полностью снимает ограничения, связанные с использованием метрической сетки в более ранних версиях системы *Protel* для оформления схем согласно требованиям ЕСКД.

Редактор символов элементов является не автономным, как в *P-CAD*, приложением, а составной частью редактора схем. Этим обеспечивается его простота в работе, а также возможность «на лету» редактировать имеющиеся библиотеки. Система *Altium Designer* имеет очень удобную функцию, позволяющую извлекать информацию о компонентах из проекта и формировать на ее основе собственные библиотеки. Данная функция особенно полезна при работе с проектами, полученными от других разработчиков, использующих собственные библиотеки компонентов.

## Описание используемых библиотечных элементов и процесса их создания

Базовые элементы, присутствующие на схеме будут следующие:

* блок*PHY-EthernetLAN8720*cразъёмом*RJ-45*;
* соединитель штырьковый *PLD-34*;
* *microUSB*;
* блок зарядки аккумуляторной батареи *BQ24295*;
* аккумуляторная батарея;
* подсветка дисплея *CAT6219*;
* микроконтроллер *ARMCortex-M4*;
* дисплей *HY32D* на базе микроконтроллера *ILI9341.*

Процесс создания библиотечных элементов проходит по общему алгоритму для каждого компонента, но различаться лишь размерами корпуса, существующих и представленных в официальном документе условных графических обозначений, а также количеством выводов.

### Блок *PHY-Ethernet LAN8720*

Особенности данного устройства [2]:

* гибкая архитектура питания (1,6 В … 3,6 В);
* может быть использован с одиночным питанием 3.3В;
* высокоэффективный*10/100 Ethernet*передатчик;
* два светодиода для отображения статуса;
* диапазон рабочей температуры от -40°C до +85°C;

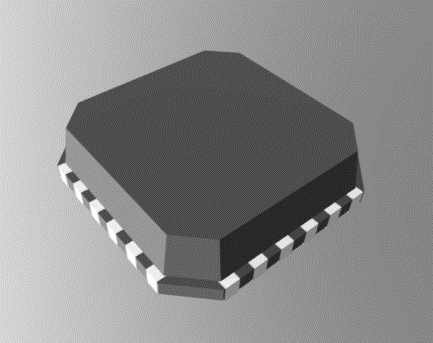


Рисунок 3.1 – Блок *PHY-Ethernet LAN8720*

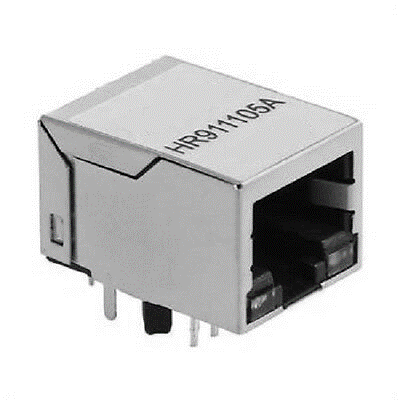


Рисунок 3.2 – Разъем *RJ-45*

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

### Соединитель штыревой *PLD-34*

Особенности данного устройства [3]:

* монтаж: *DIP*;
* шаг ножки: 2.54 мм;
* размер ножки: 0.64 мм;
* материал контактов: латунь;
* покрытие контактов: золото;
* Номинальный ток: 2.5 А;
* Номинальное напряжение: 250 В;
* Предельное напряжение: 1500 В;
* Сопротивление изолятора: 1000 MΩ;
* Сопротивление контактов: 30 mΩ;
* Рабочая температура: -25 ... +85 °C.

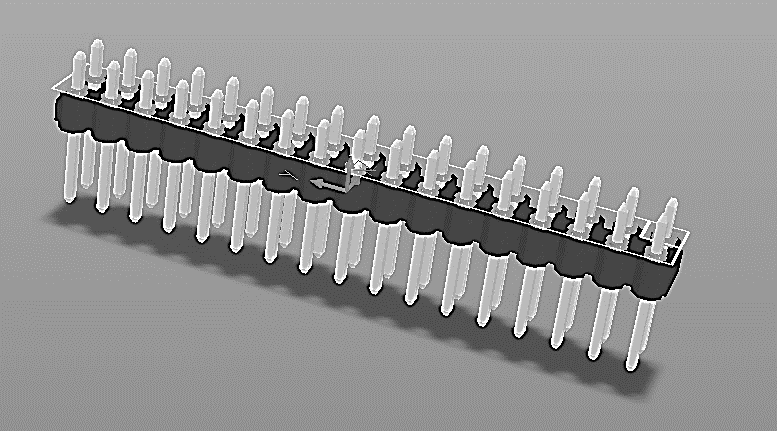


Рисунок 3.3 – Соединитель штыревой *PLD-34*

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

### *MicroUSB*

Основные особенности данного устройства [4]:

* основан на стандарте *USB* 2.0;
* максимальная скорость передачи данных достигает 480 Мбит/с.

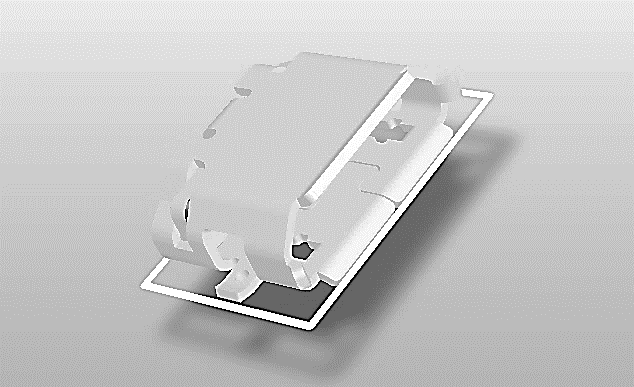


Рисунок 3.4 – *MicroUSB*

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

### Блок зарядки аккумуляторной батареи *BQ24295*

Особенности данного устройства [7]:

* входное напряжение от 3.9 В до 6.2 В;
* переключаемая частот 1.5 МГц для низкопрофильной катушки индуктивности;
* наличие порта *I2C* для оптимальной производительности системы;
* автономная зарядка батареи;
* высокая точность;
* безопасность;
* определение статуса заряда с помощью светодиода;
* возможность отслеживания максимального входного напряжения;
* корпус *VQFN-24*, размером 4 x 4 мм.

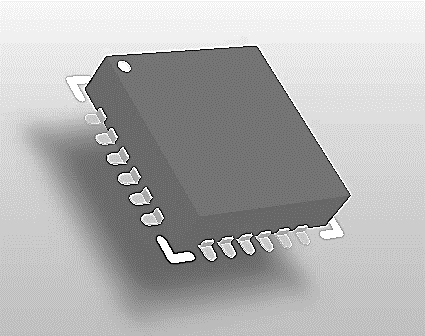


Рисунок 3.5 – Блок зарядки аккумуляторной батареи *BQ24295*

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

### Аккумуляторная батарея

На плате данная батарея будет представляться в виде штыревой вилки *PLS-2R.*

Особенности данного элемента[8]:

* шаг контактов: 2.54 мм;
* материал изолятора: полимер, усиленный стекловолокном;
* сопротивление изолятора не менее 500 Мом;
* материал контактов: фосфористая бронза;
* покрытие контактов: золото;
* сопротивление контактов не более 0.01 Ом;
* рабочий ток: 1А;
* предельное напряжение не менее 1500В;
* рабочее напряжение: 500В;
* рабочая температура: -25 ... +105 °C.

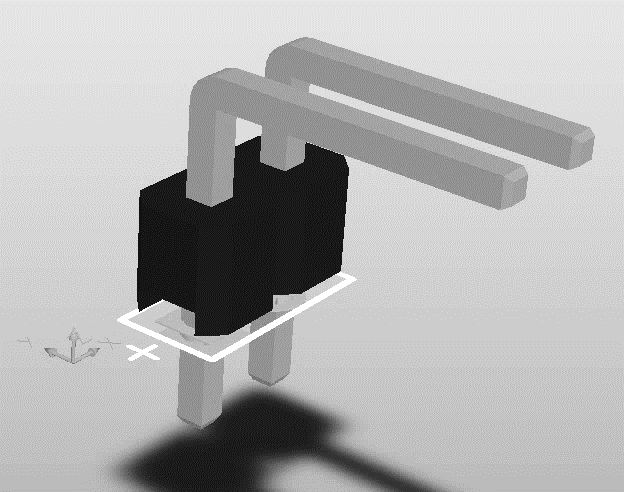


Рисунок 3.6 – Аккумуляторная батарея

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

### Подсветка дисплея *CAT6219*

Особенности данного устройства [9]:

* гарантированный пиковый выходной ток 500 мА;
* малое падение напряжения до 300 мВ при токе в 500 мА;
* внешний шунтирующий конденсатор для более низкого уровня шума;
* быстрый старт;
* тепловая защита;
* корпус *TSOT-23* с пятью ножками;

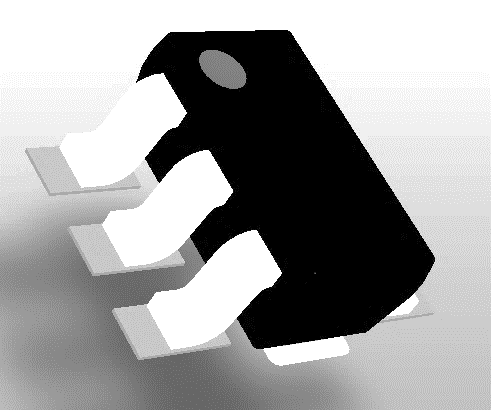


Рисунок 3.7 – Подсветка дисплея *CAT6219*

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

### Микроконтроллер *ARM Cortex-M4*

Вариант микроконтроллерного ядра *Cortex-M4* это мультистадийный *RISC*-процессор. Данное ядро основано на архитектуре *ARMv7-M* и полностью реализует наборы команд *Thumb* и *Thumb2*. Из особенностей следует упомянуть аппаратное умножение 32-разрядных чисел за 1 цикл, а также деление чисел подобной разрядности (от 2 до 12 циклов). Производительность процессора составляет 1.25 МГц. Энергопотребление примерно в два раза выше, чем у варианта M0. Количество физических прерываний увеличено до 240. В ядре предусмотрен механизм защиты памяти. Кроме того, он дополнительно оснащен *DSP*-инструкциями. Наличие последних существенно ускоряет обработку потоковых данных, что в свою очередь делает *M4* весьма привлекательным для использования в системах управления и обработки информации.

Возможности *DSP*, входящего в состав *M4*, позволяют параллельно выполнять четыре операции сложения/вычитания для 8-ми разрядных чисел или две операции сложения/вычитания с 16-ти разрядными операндами. Также реализовано умножение за один цикл, при этом для 16-ти разрядных чисел возможно параллельное исполнение двух операций [7].

В серии *M4* есть еще один вариант, под обозначением *Cortex-M4F*. В нем, дополнительно к *DSP*, установлен блок операций для чисел с плавающей точкой – *FPU*.

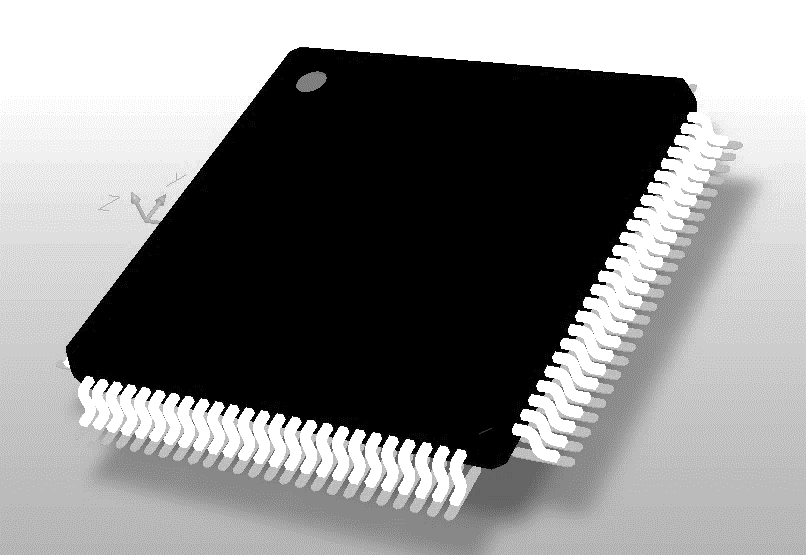


Рисунок 3.8 – Микроконтроллер *ARM Cortex-M4*

Создание данного библиотечного элемента производилось с помощью внешней библиотеки, показанной справа на рисунке.

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step* [8].

### Дисплей *HY32D* на базе микроконтроллера *ILI9341*

Особенности данного устройства:

* в дисплее встроен контроллер *TouchScreen*, отвечающий за определение места, где проходит электрический импульс для срабатывания нажатия на дисплей и передающий данные по интерфейсу *SPI*;
* в дисплее установлен *TFT*-экран для управления активной матрицей на жидких кристаллах, содержащий 65536 цветов и имеющий подсветку экрана на основе чипа CAT6219.
* разрешение дисплея составляет 320 x 240пикселей;
* размер дисплея 3.2 дюйма.

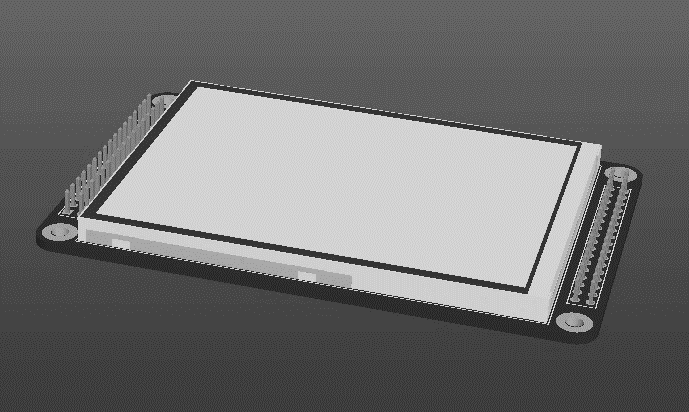


Рисунок 3.9 – Дисплей *HY32D*

*3D*-модель была скачана с открытого источника и отдельно подключена с помощью файла с расширением *\*.step*

## Обоснование выбора компонентов принципиальной схемы

Компоненты на принципиальной схеме будут следующие:

1. *Блок PHY-EthernetLAN8720*. Данный компонент предназначен для соединения с сетью *Ethernet*, приема и передачи информации.
2. *MicroUSB*. С его помощью будет происходить зарядка устройства.
3. *Блок зарядки аккумуляторной батареи BQ24295*. Данный компонент выбрал вследствие совместимости с микроконтроллером семейства *STM32F4*, а также в целом для данного устройства ввиду его особенностей функционирования и подключения на принципиальной схеме.
4. *Микроконтроллер STM32F407VGT6 на базе ARMCortex-M4*. Данный микроконтроллер выбран в корпусе, содержащем 100 выводов. Это связано с тем, что в нашем устройстве присутствует интерфейс *FSMC*, к которому подключается дисплей, выводы которого, по документации, содержатся не менее чем в 100-контактной площадке. Такой микроконтроллер выбран оптимальным по соотношению «цена-производительность».
5. *Дисплей HY32D*. Такой дисплейный модуль выбран в связи с хорошей совместимостью с микроконтроллером семейства *HY32D*, а также, в настоящее время, с наиболее приемлемыми параметрами.
6. *Подсветка дисплея CAT6219*. Данный чип выбран вследствие полной совместимости с дисплеем *HY32D*.

## Обоснование связей принципиальной электрической схемы

Блок зарядки аккумуляторной батареи имеет три вывода, специально отведенные под подключение *USB*, тем самым позволяя через него проводить зарядку аккумуляторной батареи. Далее, сигнал идет по шине через интерфейс *I2C* к микроконтроллеру.

К блоку зарядки подключен светодиод, который показывает состояние зарядки аккумуляторной батареи. Также, в этой цепи имеется подключенный последовательно резистор для того, чтобы повысить напряжение до момента, который не превышает пиковое напряжение, то есть ослаблять источник питания для того, чтобы он не дал большее напряжение и светодиод не сгорел, Кроме этого, сам ток по цепи протекал бы быстрее.

В этой схеме также содержится терморезистор, который сопровождают два постоянных резистора, меняющий свое электрическое сопротивление в зависимости от его температуры. Один резистор подключен параллельно для выпрямления тока в цепи, а другой последовательно для увеличения напряжения в цепи.

Подсветка дисплея подключена напрямую к интерфейсу *FSMC*. Для увеличения переходной характеристики, в цепи подключены конденсаторы, а резисторы играют роль выпрямителей напряжения.

В общих целях, к микрофоны подключены два конденсатора, функционирующие как фильтр, и два резистора, который не дадут конденсатором выйти из строя. Помимо этого, в микрофоне имеются два вывода, реализованные посредством интерфейса *SPI*, идущие к микроконтроллеру и подключенные по ключевому слову «*SPI*» исходя из документации на микроконтроллер.

Подключение регистров интерфейса *SDIO* и микроконтроллера осуществлялось с помощью официальной документации *(datasheet)* на микроконтроллер семейства *STM32F4* с помощью поиска по ключевому фрагменту слова «*SDIO*». Кроме этого, на каждом из выводов, идущих на микроконтроллер, присутствуют резисторы-перемычки.

Все выводы, помимо тех, которые предназначаются для подсветки дисплея и сенсорного экрана, подключены напрямую к микроконтроллеру семейства *STM32F4*. Это обусловлено тем, что данный микроконтроллер на 100 выводов содержит регистры, предназначенные для полноценной инициализации интерфейса *FSMC*на данном устройстве. Подключение происходило непосредственно с помощью документации на микроконтроллер с помощью поиска по ключевому фрагменту слова «*FSMC\_*».

## Анализ и обоснование принципиальной электрической схемы зарядки аккумуляторной батареи мобильного устройства

За соединение батарейки с блоком заряда аккумуляторной батарейки отвечает вывод *BAT*. Помимо него, к этому выводу подключен вывод *QON*, отвечающий за контроль соединения батареи с блоком зарядки.

Параллельно батарейке подключен конденсатор. С помощью него зарядка батарейки будет проходить примерно в два раза быстрее, чем без него.

После этого, и конденсатор, и батарейка идут на сигнальную землю, с помощью которого измеряют или оцифровывают напряжение сигнальный проводов. При этом питание на приборы могут подаваться из разных источников.

# Разработка алгоритма функционирования мобильного Ethernet UDP-сервера с поддержкой протокола Modbus RTU

Согласно архитектуры построения сети, поддерживающей работу протокола *MODBUS RTU*, все устройства, подключенные к этой сети, подразделяются на два типа, а именно *slave*-устройства и *master*-устройства. Следуя главной идее данного курсового проекта, разрабатываемое в нём мобильное устройство является *slave*-устройством. Это связано с тем, что данное устройство должно принимать и отвечать на запросы master-устройства, которым является компьютер, соединенный с мобильным устройством посредством *Ethernet* кабеля.

Определение данного аспекта является ключевым при построении схемы алгоритма, так как определяет всю суть и принцип функционирования устройства.

Итак, после включения устройства проводится инициализация всех компонентов печатной платы. Успешное окончание инициализации означает то, что все её составные элементы соединены правильно и готовы к работе.

После инициализации устройство готово выполнять свою функцию сервера, а именно отвечать на запросы master-устройства. Мобильный сервер прослушивает порт *Ethernet* на наличие поступления какого-либо сигнала. При получении пакета протокола *MODBUS RTU* с запросом серверу необходимо проверить, кому предназначается данный запрос. Ведь в сети может находиться огромное количество серверов.

После проверки и совпадения адреса целевого устройства, указанного в пакете, и адреса сервера он анализирует полученные данные дальше. Следующей ячейкой пакета является команда, которую серверу необходимо выполнить. В нашем случае это может быть либо команда чтения дискретного выхода микроконтроллера, либо команда записи.

Если получена команда чтения, то микроконтроллеру необходимо проверить значения указанных далее в пакете адресов дискретных выводов. Далее формируется ответ для *master*-устройства. В случае же получения команды на запись микроконтроллер записывает полученную информацию в свои дискретные выходы. После окончательной обработки полученного пакета данных и выполнения необходимых команд мобильный сервер отправляет пакет, содержащий ответ.

На этом цикл работы данного устройства окончен. Оно готово к получению новых пакетов, их анализу и выполнению.

# Разработка конструкции проектируемого изделия

## Выбор и обоснование элементной базы

В настоящее время происходит минимизация всего оборудования, всех элементов, уменьшение размеров технологического процесса вплоть до 10 нм. Тем самым необходимо использовать и соответственно маленькие элементы. Поскольку разрабатываемое устройство является мобильным, то по умолчанию при его проектировании и производстве должны использоваться максимально возможно малые элементы. Для таких целей подходят резисторы и конденсаторы в *SMD*-корпусе, обладающие такими же, а в некоторых случаях и лучшими характеристиками, чем обычные МЛТ-резисторы, но имеющие гораздо меньшие габаритные размеры. Также элементы в *SMD*-корпусе менее подвержены воздействию внешних механических повреждений, что является весомым преимуществом для мобильного устройства.

В нашем случае используется корпус *SMD* 0805.

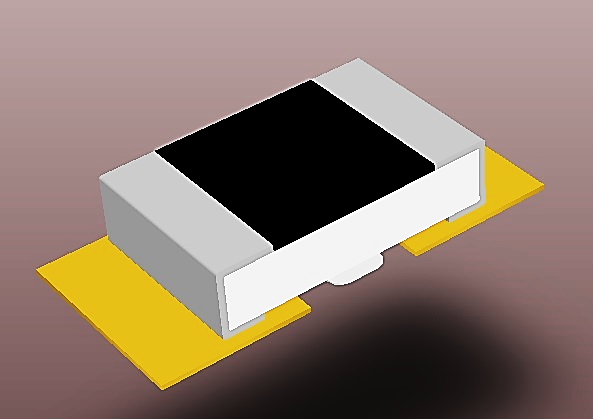


Рисунок 5.1 – Резистор в *AltiumDesigner*

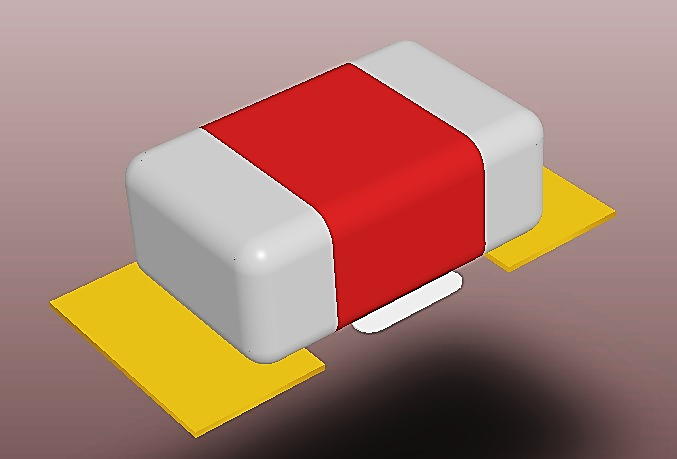


Рисунок 5.2 – Конденсатор в *AltiumDesigner*

Корпус 0805 в обоих элементах был выбран потому, что, во-первых, он считается наиболее распространенным и часто используемым, а, во-вторых, его основных характеристик для данного устройства будет более, чем достаточно. Также немаловажным фактором является то, что цена на данные элементы не столь высока.

## Выбор и обоснование конструктивных элементов и установочных изделий

Основные конструктивные элементы печатной платы – основание (подложка) и проводники. Эти элементы необходимы и достаточны для того, чтобы печатная плата функционировала. Круг второстепенных элементов несколько шире: контактные площадки, переходные металлизированные и монтажные отверстия, ламели для контактирования с разъемами, участки для осуществления теплоотвода и прочее.

В отличие от навесного монтажа, на печатной плате электропроводящий рисунок выполнен из фольги, целиком расположенной на твердой изолирующей основе. Печатная плата содержит монтажные отверстия и контактные площадки для монтажа выводных или планарных компонентов. Кроме того, в печатных платах имеются переходные отверстия для электрического соединения участков фольги, расположенных на разных слоях платы. С внешних сторон на плату обычно нанесены защитное покрытие («паяльная маска») и маркировка (вспомогательный рисунок и текст согласно конструкторской документации).

Основой печатной платы служит диэлектрик, наиболее часто используются такие материалы, как стеклотекстолит.

Также основой печатных плат может служить металлическое основание, покрытое диэлектриком (например, анодированный алюминий), поверх диэлектрика наносится медная фольга дорожек. Такие печатные платы применяются в силовой электронике для эффективного теплоотвода от электронных компонентов. При этом металлическое основание платы крепится к радиатору.

Проводники на печатной плате соответствуют классу точности 4, о чем будет рассказано далее. Это связано с тем, что некоторые элементы на плате имеют довольно маленькую ширину контактных площадок, что при ширине проводников в более, чем 0.25 мм, невозможно будет провести трассировку печатной платы.

Ширина проводников, связанных с питанием, нескольким больше, чем обычные проводники, поскольку через них течет больший ток, по сравнению с обычными проводниками.

При трассировки печатной платы были использованыпереходные металлизированные отверстия. Это связано с отсутствием или невозможностью провести проводник на одной стороне печатной платы.

# Расчёт конструктивно-технологических параметров проектируемого изделия

## Проектирование печатного модуля

При проектировании мобильного устройства за основу была взята двусторонняя печатная плата. Двусторонней является печатная плата, на обеих сторонах которой могут располагаться дорожки электропроводящего материала, соединяющие радиоэлементы между собой. Это позволяет наиболее плотно скомпоновать электрические элементы на плате, повысив процент плотности её монтажа. Двусторонние печатные платы используются в измерительной технике, системах управления и других областях. Данный тип печатной платы выбран по причине того, что некоторые высокие элементы, такие как *microUSB*, не поместятся под дисплеем, который будет располагаться на противоположной стороне печатной платы.

Изготовление печатных плат определенного класса точности (ГОСТ 23751-86) обеспечивают, применяя различные техническое оснащение и вспомогательные материалы. Печатные платы 1 и 2 классов точности наиболее просты в исполнении, надежны в эксплуатации и имеют минимальную стоимость; 3 класса – требуют использования высококачественных материалов, более точного инструмента и оборудования; 4 и 5 классов – специальных материалов, прецизионного оборудования, особых условий для изготовления.

Наименьшие номинальные значения основных размеров элементов конструкции печатной платы для узкого места в зависимости от классов точности приведены в таблице 6.1. В качестве условных обозначений ширины печатного проводника, расстояния между краями соседних элементов проводящего рисунка, гарантийного пояска и отношения значения диаметра наименьшего металлизированного отверстия к толщине печатной платы используются соответственно буквы латинского алфавита *t, S, b, g\**.

Таблица 6.1 – Классы точности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Условное обозначение | Номинальное значение основных размеров для класса точности | | | | |
| 1 | 2 | 3 | 4 | 5 |
| t, мм | 0,75 | 0,45 | 0,25 | 0,15 | 0,10 |
| S, мм | 0,75 | 0,45 | 0,25 | 0,15 | 0,10 |
| b, мм | 0,30 | 0,20 | 0,10 | 0,05 | 0,025 |
| g\* | 0,40 | 0,40 | 0,33 | 0,25 | 0,20 |

Исходя из чертежа печатной платы, выбран класс точности 3.

В соответствии с ГОСТ 10317-79, основной шаг координатной сети должен составлять 2.5 мм.

При изготовлении печатной платы был выбран аддитивный метод. Этот метод заключается в создании проводящего рисунка посредством металлизации достаточно толстым слоем химической меди (25 – 35 мкм), что позволяет исключить применение гальванических операций и операций травления.

Выбор на этот метод изготовления пал потому, что в нем могут присутствовать металлизированные отверстия, он подходит для двусторонних печатных плат, обладает рядом преимуществ и малым количеством недостатков, а самое главное – минимальная ширина проводника должна составлять 0.1 мм, что подходит для данной печатной платы.

Основные параметры полученного устройства:

* габаритные размеры: 100 x 80 x 10 мм, поскольку практически всю одну сторону печатной платы занимает дисплейный модуль;
* толщина печатной платы: 1.5 мм;
* расположение элементов проводящего рисунка: 0.15 мм.

Таким образом, были описаны основные параметры печатной платы для того, чтобы передать ее в производство.

## Выбор и обоснование материалов конструкции и защитных покрытий, маркировки деталей и сборочных единиц

Печатная плата должна быть изготовлена из стеклотекстолита. На сегодняшний день этот материал повсеместно используется для производства печатных плат, так как хорошо зарекомендовал себя, благодаря большому количеству преимуществ по сравнению с другими материалами. Стеклотекстолит – это электроизоляционный слоистый материал, который используется при производстве электрооборудования и электротехники, деталей и изделий сложной конфигурации.

Стеклотекстолит является разновидностью текстолита, который используют при производстве подшипников, шестерней, пластин и прочих деталей, от которых требуется повышенная прочность. Одно из главных достоинств данного материала – возможность обрабатывать его любым механическим способом, что является причиной выбора данного материала.

Основное назначение таких покрытий – защитить медную поверхность контактных площадок и обеспечить качественную пайку электронных компонентов на печатные платы.

По составу защитные покрытия в основном представляют собой металлы и их сплавы. Исключением является т.н. *OSP* покрытие, которое стоит особняком не только потому, что является органическим, но и потому, что оно разлагается при пайке, тогда как металлические покрытия растворяются в сплаве, образующем паяное соединение.

Среди методов нанесения металлических покрытий наибольшее распространение получили лужение, химическое и гальваническое осаждение. Лужение – достаточно простой способ, однако он теряет свою актуальность из-за сложности получения поверхностей с высокой плоскостностью, а также высоких температур процесса, отрицательно влияющих на надежность. Гальваническое осаждение – быстрый и хорошо контролируемый процесс, однако он требует наличия электрического контакта между всеми поверхностями, на которые наносится покрытие, что в случае рисунка печатной платы бывает достаточно сложно обеспечить. Для этого применяются технологические перемычки, соединяющие различные проводники и области в единое целое, которые потом удаляются, однако это в любом случае снижает технологичность процесса. В силу этих причин наибольшее распространение среди металлических покрытий получили т.н. иммерсионные покрытия (от англ. *immersion* – погружение), получаемые путем химической реакции замещения в растворе. Этот метод химического осаждения обеспечивает достаточно тонкие и однородные покрытия именно тех участков, где имеется открытая медь, т.е. благодаря реакции замещения процесс является самоуправляемым.

Среди наиболее распространенных защитных покрытий находится*HASL (Hot Air Solder Leveling)* – покрытие припоем с выравниванием. Процесс заключается в погружении платы в емкость с расплавом олова и свинца с последующим удалением излишков пасты при помощи «воздушных ножей», которые обдувают горячим воздухом поверхность платы. Одно из очевидных преимуществ технологии *HASL* заключается в том, что плата подвергается температурному воздействию в 265°C, которое выявит такую проблему как расслоение, до того, как компоненты будут напаяны на плату. Однако, медь и олово обладают химическим сродством, неизбежно произойдет диффузия одного металла в другой, что непосредственно отразится на сроке хранения и общей производственной пригодности покрытии.

Основная маркировка, наносимая на печатную плату, обязательнодолжна содержать:

* обозначение ПП или ее условный шрифт;
* дату изготовления (год, месяц);
* буквенно-цифровое обозначение слоя многослойных печатных плат.
* Дополнительная маркировка наносится при необходимости и может содержать:
* порядковый или заводской номер печатной платы;
* позиционное обозначение навесных электрорадиоэдементов;
* изображение контуров навесных электрорадиоэдементов.

# Применение средств автоматизированного проектирования при разработке устройства

В данном курсовом проекте было использовано три основных САПР: *AltiumDesigner*, *SolidWorks* и *AutoCAD* и каждый из них выполнял свою роль.

С помощью САПР *AltiumDesigner* были спроектированы библиотечные элементы для каждого компонента на принципиальной схеме, которые были найдены предварительно, далее созданы посадочные места, содержащие в себе контактные площадки либо отверстия, в зависимости от компонента, а также подключенные *STEP*-модели в *3D*. После сопоставления библиотечных элементов и посадочных мест переходим к созданию электрической принципиальной схемы в данном САПР. Воссоздав схему в данном САПР, компилируем ее и проверяем на наличие ошибок, и в случае их отсутствия извлекаем элементы на печатную плату, предварительно указав длину, ширину и толщину самой печатной платы. Результат конечной печатной платы показан на рисунках Д.1 и Д.2.

САПР *SolidWorks* использовался для импорта модели печатной платы *STEP 3D* из *AltiumDesigner* путем создания сборки печатной платы из деталей (электрорадиоэлементов). После чего создавался сборочный чертеж на основе полученной сборки. Кроме этого, данный САПР использовался для создания видов и разрезов электрорадиоэлементов, находящихся на плате. Следует отметить, что *SolidWorks*, *AltiumDesigner* и *AutoCAD* являются плохо совместимыми САПР, поскольку логика функционирования у каждого своя, таким образом получаем, что экспорт любого чертежа, схемы и так далее не будет даже примерно соответствовать тем требованиям, по которым делается данный курсовой проект.

И третий САПР *AutoCAD* использовался для окончательного изменения и редактирования всех чертежей, поскольку, как было сказано выше, экспорт проходит таким образом, что изменять чертежи приходится практически в полном объеме.

Заключение

В результате выполнения данного курсового проекта был разработан мобильный *Ethernet UDP-*сервер протокола *ModbusRTU*. В ходе работы были пройдены основные этапы проектирования.

Первым этапом была разработка структурной и принципиальной схем устройства, поиск необходимых компонентов и создание библиотеки элементов принципиальной схемы.

Второй этап – разработка печатной платы. На этом этапе была создана библиотека футпринтов используемых компонентов, произведено портирование компонентов платы с принципиальной схемы на схему печатной платы и трассировка электропроводящих дорожек.

Вышеуказанные этапы были полностью проделаны в САПР *AltiumDesigner.*

После этого была разработана вся необходимая документация по устройству: чертеж принципиальной схемы, чертеж печатной платы, сборочный чертеж, алгоритм работы устройства. Все эти чертежи были выполнены в среде *AutoCad* и *SolidWorks.*

В результате была разработана печатная плата размером 100x80 мм.

Таким образом, устройство отвечает техническим требованиям, полученным в начале курсового проектирования. На выходе получен готовый проект устройства, готового для сборки и выпуска в массовое производство

Список использованных источников

1. Протокол *ModbusRTU*[Электронный ресурс]. ­– Режим доступа:https://ipc2u.ru/articles/prostye-resheniya/modbus-rtu/
2. *LAN8720A/LAN8720Ai*[Электронный ресурс]. ­– Режим доступа:http://ww1.microchip.com/downloads/en/devicedoc/00002165b.pdf
3. Трехмерная модель соединителя штыревого PLD-34 [Электронный ресурс]. ­– Режим доступа: http://www.3dcontentcentral.com/secure/download-model.aspx?catalogid=171&id=227431.
4. USB and Mini USB [Электронныйресурс]: Datasheet / Tyco Electronics. ­– Режим доступа: http://www.farnell.com/datasheets/110335.pdf.
5. Трехмерная модель microUSB [Электронный ресурс]. ­– Режим доступа: http://www.3dcontentcentral.com/secure/Download-Partner-Model.aspx?partner=TraceParts&name=Connector%2cmicroUSB
6. I2C Controlled 3A Single Cell USB Charger with Adjustable Voltage and 1.5A OTG (Rev. A) [Электронныйресурс]: Datasheet / Texas Instruments. ­– Режим доступа:http://www.ti.com/lit/ds/symlink/bq24295.pdf.
7. Трехмерная модель BQ24295 [Электронный ресурс]. ­– Режим доступа : http://www.3dcontentcentral.com/secure/download-model.aspx?  
   catalogid=171&id=445118.
8. Трехмерная модель аккумуляторной батареи [Электронный ресурс]. ­– Режим доступа: http://www.3dcontentcentral.com/secure/download-model.aspx?catalogid=171&id=354787.
9. CAT6219.rev14 [Электронныйресурс]: Datasheet / ON Semiconductor. ­– Режим доступа : http://www.onsemi.com/pub\_link/Collateral/  
   CAT6219-D.PDF.
10. Трехмерная модель CAT6219[Электронный ресурс]. ­– Режим доступа: http://www.3dcontentcentral.com/secure/download-model.aspx?  
    catalogid=171&id=186243.
11. Классы точности печатных плат [Электронный ресурс]. ­– Режим доступа:http://www.circuitry.ru/files/article\_pdf/2/article\_2242\_556.pdf

Приложение Д  
(**обязательное**)  
Трехмерная модель печатной платы устройства

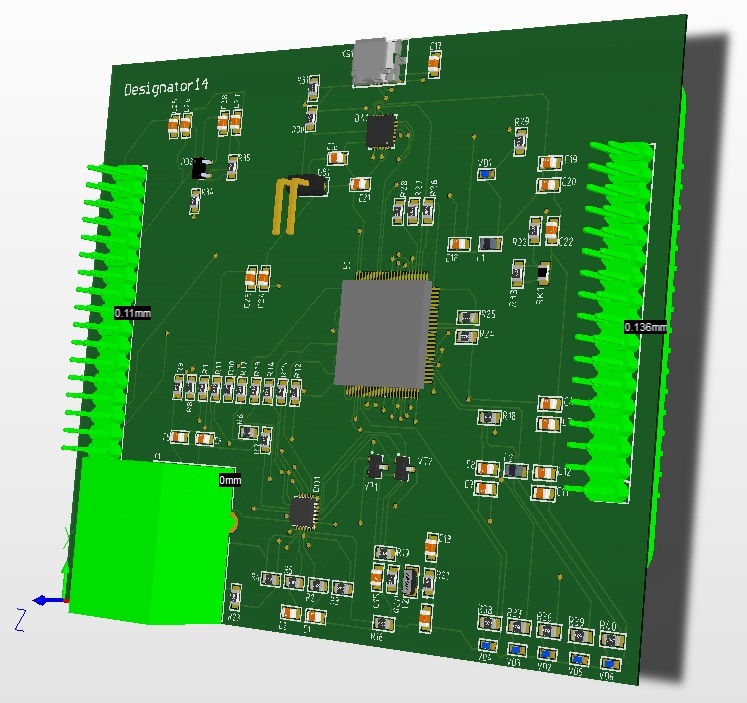


Рисунок Д.1 – Лицевая сторона печатной платы

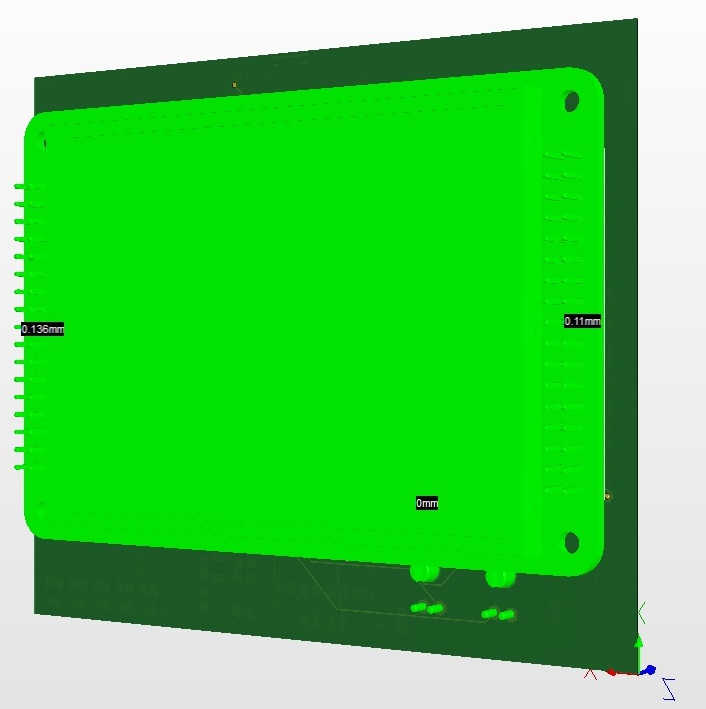


Рисунок Д.2 – Тыльная сторона печатной платы

Приложение Е  
(**обязательное**)  
Текст программы

**Main.c**

#include <mb.h>

#include <mbport.h>

#include "ModbusSlaveSoftwareSerial.h"

#include "SoftwareSerial.h"

#define BUFFER\_SIZE 128

unsigned char frame[BUFFER\_SIZE];

unsigned int holdingRegsSize;

unsigned char broadcastFlag;

unsigned char slaveID;

unsigned char function;

unsigned char TxEnablePin;

unsigned int errorCount;

unsigned int T1\_5;

unsigned int T3\_5;

void exceptionResponse(unsigned char exception);

unsigned int calculateCRC(unsigned char bufferSize);

void sendPacket(unsigned char bufferSize);

SoftwareSerial mySerial(0, 1);

unsigned int modbus\_update(unsigned int \*holdingRegs)

{

unsigned char buffer = 0;

unsigned char overflow = 0;

while (mySerial.available())

{

if (overflow)

mySerial.read();

else

{

if (buffer == BUFFER\_SIZE)

overflow = 1;

frame[buffer] = mySerial.read();

buffer++;

}

delayMicroseconds(T1\_5);

}

if (overflow)

return errorCount++;

if (buffer > 6)

{

unsigned char id = frame[0];

broadcastFlag = 0;

if (id == 0)

broadcastFlag = 1;

if (id == slaveID || broadcastFlag)

{

unsigned int crc = ((frame[buffer - 2] << 8) | frame[buffer - 1]);

if (calculateCRC(buffer - 2) == crc)

{

function = frame[1];

unsigned int startingAddress = ((frame[2] << 8) | frame[3]);

unsigned int no\_of\_registers = ((frame[4] << 8) | frame[5]);

unsigned int maxData = startingAddress + no\_of\_registers;

unsigned char index;

unsigned char address;

unsigned int crc16;

if (!broadcastFlag && (function == 3))

{

if (startingAddress < holdingRegsSize)

{

if (maxData <= holdingRegsSize)

{

unsigned char noOfBytes = no\_of\_registers \* 2;

unsigned char responseFrameSize = 5 + noOfBytes;

frame[0] = slaveID;

frame[1] = function;

frame[2] = noOfBytes;

address = 3;

unsigned int temp;

for (index = startingAddress; index < maxData; index++)

{

temp = holdingRegs[index];

frame[address] = temp >> 8;

address++;

frame[address] = temp & 0xFF;

address++;

}

crc16 = calculateCRC(responseFrameSize - 2);

frame[responseFrameSize - 2] = crc16 >> 8;

frame[responseFrameSize - 1] = crc16 & 0xFF;

sendPacket(responseFrameSize);

}

else

exceptionResponse(3);

}

else

exceptionResponse(2);

}

else if (function == 6)

{

if (startingAddress < holdingRegsSize) {

unsigned int startingAddress = ((frame[2] << 8) | frame[3]);

unsigned int regStatus = ((frame[4] << 8) | frame[5]);

unsigned char responseFrameSize = 8;

holdingRegs[startingAddress] = regStatus;

crc16 = calculateCRC(responseFrameSize - 2);

frame[responseFrameSize - 2] = crc16 >> 8;

frame[responseFrameSize - 1] = crc16 & 0xFF;

sendPacket(responseFrameSize);

}

else

exceptionResponse(2);

}

else if (function == 16) {

if (frame[6] == (buffer - 9)) {

if (startingAddress < holdingRegsSize) {

if (maxData <= holdingRegsSize) {

address = 7;

for (index = startingAddress; index < maxData; index++) {

holdingRegs[index] = ((frame[address] << 8) | frame[address + 1]);

address += 2;

}

crc16 = calculateCRC(6);

frame[6] = crc16 >> 8;

frame[7] = crc16 & 0xFF;

if (!broadcastFlag)

sendPacket(8);

}

else

exceptionResponse(3);

}

else

exceptionResponse(2);

}

else

errorCount++;

}

else

exceptionResponse(1);

}

else

errorCount++;

}

}

else if (buffer > 0 && buffer < 8)

errorCount++;

return errorCount;

}

void exceptionResponse(unsigned char exception)

{

errorCount++;

if (!broadcastFlag) {

frame[0] = slaveID;

frame[1] = (function | 0x80);

frame[2] = exception;

unsigned int crc16 = calculateCRC(3);

frame[3] = crc16 >> 8;

frame[4] = crc16 & 0xFF;

sendPacket(5);

}

}

void modbus\_configure(long baud, unsigned char \_slaveID, unsigned char \_TxEnablePin, unsigned int \_holdingRegsSize)

{

slaveID = \_slaveID;

mySerial.begin(baud);

if (\_TxEnablePin > 1)

{

TxEnablePin = \_TxEnablePin;

pinMode(TxEnablePin, OUTPUT);

digitalWrite(TxEnablePin, LOW);

}

if (baud > 19200) {

T1\_5 = 150;

T3\_5 = 350;

}

else {

T1\_5 = 15000000/baud;

T3\_5 = 35000000/baud;

}

holdingRegsSize = \_holdingRegsSize;

errorCount = 0;

}

unsigned int calculateCRC(byte bufferSize)

{

unsigned int temp, temp2, flag;

temp = 0xFFFF;

for (unsigned char i = 0; i < bufferSize; i++) {

temp = temp ^ frame[i];

for (unsigned char j = 1; j <= 8; j++)

{

flag = temp & 0x0001;

temp >>= 1;

if (flag)

temp ^= 0xA001;

}

}

temp2 = temp >> 8;

temp = (temp << 8) | temp2;

temp &= 0xFFFF;

return temp;

}

void sendPacket(unsigned char bufferSize)

{

if (TxEnablePin > 1)

digitalWrite(TxEnablePin, HIGH);

for (unsigned char i = 0; i < bufferSize; i++)

mySerial.write(frame[i]);

mySerial.flush();

delayMicroseconds(T3\_5);

if (TxEnablePin > 1)

digitalWrite(TxEnablePin, LOW);

}