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

Институт машиностроения
(наименование института полностью)

Кафедра Промышленная электроника
(наименование)

13.03.02 Электроэнергетика и электротехника
(код и наименование направления подготовки, специальности)

Интеллектуальные энергетические системы
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: Четырехканальный электронный таймер

Студент Н.А.Кошманов
(И.О. Фамилия) (личная подпись)

Руководитель А.К.Кудинов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

на бакалаврскую работу Кошманова Никиты Андреевича

по теме: «Четырехканальный электронный таймер»

Бакалаврская работа состоит из 54 страниц, 3 таблицы, 30 рисунков, списка литературы из 27 источников.

Изучены функции и структура разнообразных таймеров. Разобраны и рассмотрены элементы управления, составляющие элементы этих систем. Произведен обзор и анализ рынка таймеров, сделан выбор наилучших моделей, и произведено сравнение их по характеристикам. Разработано микропроцессорное устройство с функциональностью четырехканального электронного таймера.

Работа состоит из трёх глав, в которых решены упомянутые задачи.

Разработанная установка функционирует на основе платформы быстрого прототипирования Arduino по уникальному алгоритму, написанному на языке программирования C/C++ в среде разработки Arduino IDE. Функциональность устройства проверена в симуляторе Proteus 8.5 pro.

Степень внедрения - установка по разработанной документации является опытным образцом.

Областью применения данной системы является малая автоматизация на производстве предприятий малого и среднего бизнеса, например, для управления электрическими приборами теплицы (освещение, вентиляция, обогрев и полив растений).

Abstract

The title of the graduation work is Four-channel electronic timer.

The senior paper consists of an introduction, three parts, a conclusion, tables, the list of references including foreign sources and the graphic part on 6 A1 sheets.

The key issue of the thesis is the design of a control device for four independent timers. The paper touches on the convenience of managing timers with the continuity of timers.

The purpose of the work is to develop a microprocessor device of a four-channel timer.

We start with the statement of the problem and then logically pass over to its possible solutions.

The senior thesis can be divided into the following logically connected parts which are analysis of existing solutions for controlling the turn-on time of electrical appliances; justification of using Arduino rapid prototyping platform for solving the task; description of the characteristics and features of using components on the Arduino platform; technological and design solutions; development of an algorithm for the operation of a four-channel timer and writing a program using this algorithm in C/C++.

Finally, we carry out analysis of solving the problem of uninterrupted timers' operation in case of power failure.

The work is of interest for wide circle of readers.

Содержание

Введение

Актуальность. Современный мир уже не может обойтись без автоматики, облегчающей человеческий труд как в быту, так и на производстве. Многофункциональные процессорные устройства можно запрограммировать на выполнение множества задач, одной из которых является выполнение периодического включения и выключения разного рода полезных электрических приборов. Для решения этой задачи применяются массово выпускаемые промышленностью таймеры с разной функциональностью. Большинство таких приборов рассчитаны на суточное или недельное использование и, к сожалению, управляют только одним или редко двумя каналами.

Объект исследования: четырехканальный электронный таймер с возможностью задания интервала в пределах суток и с периодичностью включения и выключения в течении 1-30 суток.

Предмет исследования: средства быстрого прототипирования микропроцессорных устройств.

Гипотеза исследования: предполагается, что, используя платы быстрого прототипирования платформы Arduino, можно легко создать многофункциональный электронный таймер, отвечающий поставленным задачам.

Цель исследования: отработать навыки создания микропроцессорной и электронной техники.

Задачи исследования:

- определить основные функции таймеров и по возможности найти функциональные аналоги четырехканального электронного таймера;
- разработать четырехканальный электронный таймер в соответствии с заданием на проектирование;

- произвести расчет себестоимости и сравнить разработанное устройство с имеющимися на рынке функциональными аналогами.

Научная новизна: разработано электронное устройство методом быстрого прототипирования с разработкой уникального алгоритма работы четырехканального таймера с программированием через цифро-графический интерфейс.

Практическая значимость: разработанное устройство может быть использовано в производстве продуктов питания на предприятиях малого и среднего бизнеса, например, для управления электрическими приборами теплицы (освещение, вентиляция, обогрев и полив растений).

1 Состояние вопроса

1.1 Анализ исходных данных

В соответствии с заданием на выполнение бакалаврской работы разрабатываемый четырехканальный электронный таймер питается от однофазной сети с напряжением 220В и частотой переменного тока 50Гц [14] управляет четырьмя независимыми нагрузками с током до 1А и максимальным напряжением до 380В.

Каждый из четырех независимых каналов таймера может быть установлен на срабатывание в интервале от 1 минуты до 24 часов (1сутки=24ч*60мин=1440мин.) [17] с периодичностью от 1 до 30 суток. Исходя из практической значимости, в разрабатываемом устройстве будем задавать время старта (включение нагрузки) и завершения работы таймеров (выключение нагрузки) независимо в каждом канале.

Точность отсчета времени не задана, поэтому возьмем допустимую ошибку точности хода часов в 1 минуту за максимальный период 30 суток, т.е. не более 1мин./30сут. = 2 секунд/сутки [21].

1.2 Обзор известных решений

Таймер – это функциональное устройство, отсчитывающее заданный период времени. Существуют два принципиально различных типа таймеров – механические и электронные.

Механические используют пружинную, синхронную (синхронно частоте питающей сети) или кварцевую технологию привода часового механизма [5]. Они крайне просты в использовании и стоят дешевле электронных аналогов. Индикация режимов работы таймера производится на механической шкале или поворотных барабанах. Пример механического

промышленного таймера с креплением на DIN-рейку Vemer AIKO-DW приведен на рис.1.

Рисунок 1 – механический промышленный таймер Vemer

Электронные таймеры также можно разделить на два типа – построенные на счетчиках и микропроцессорные. Первые имеют задающий генератор и электронные счетчики на отдельных микросхемах малой степени интеграции, отсчитывающие заданный интервал времени. Для индикации режимов работы используются одиночные и семи сегментные индикаторы. В таблице 1 приведены примеры одноканальных микропроцессорных таймеров в промышленном исполнении [11].

Микропроцессорные таймеры наиболее многофункциональные и более компактные, чем другие типы таймеров, т.к. выполнены на специализированных больших интегральных микросхемах (БИС), выполняющих функции множества простых схем. В зависимости от точности отсчета времени можно выделить два вида микропроцессорных таймеров – с модулем реального времени (RTC) с прецизионной кварцевой стабилизацией частоты задающего генератора или встроенным в микропроцессор тактовым генератором средней точности [23]. Микропроцессорные таймеры можно использовать для управления несколькими независимыми нагрузками по не зависимым или зависимым друг от друга программам включения и выключения таймеров с заданной периодичностью [3]. Для индикации

режимов работы таймеров можно использовать различные виды индикаторов и графических экранов.

Таблица 1 – сравнительная таблица одноканальных электронных таймеров

Параметры				
Наименование	3SHC8A	3SHC18A	KG316T	TS-GE2
Аналог	ТЭ-02	ТЭ-15	ТЭ-16	
Число программ, шт	10	8	6	8
Ток коммутации, А	16	16	25	16
Ширина, мм	50	35	74	35
Высота, мм	100	85	120	85
Глубина, мм	75	65	52	65
Вес, гр	160	150	430	140
Способ крепления	din-рейка	din-рейка	на монтажную плоскость	din-рейка

Управление нагрузкой может быть осуществлено механическими контактами, электромеханическим реле или электронным (твердотельным) реле. В зависимости от характеристик нагрузки (ток и напряжение коммутации) выбирается тип управления [13]. Механические контакты менее долговечны, но обладают большей перегрузочной способностью по току в замкнутом состоянии чем электронные.

Электронные и электромеханические таймеры необходимо питать от источника тока. Механические же таймеры работают от завода пружины. Для гарантированного срабатывания таймера необходимо беспрерывное питание электроники на время, заданное таймером [18]. В задании на выполнение бакалаврской работы не указано требование по обеспечению гарантированного питания устройства. Будем считать, что в месте

эксплуатации электронного таймера не будет проблем с беспрерывностью питания электроники на протяжении 30 суток.

По типу корпуса и его креплению можно разделить таймеры на настольные, настенные, стоечные и интегрируемые . Стоечные таймеры монтируются в стойку с оборудованием. Интегрируемые таймеры монтируются в корпус другого прибора, например, на DIN-рейку.

В качестве четырехканального программируемого таймера можно задействовать контроллер «умного дома» ZONT C2000+ (рис.2) [9], который обеспечивает дистанционный контроль параметров системы через веб-сервис и мобильное приложение и оповещает при срабатывании аварийных датчиков или срабатывании таймеров. Он имеет шесть релейных выходов и шесть универсальных. [15]

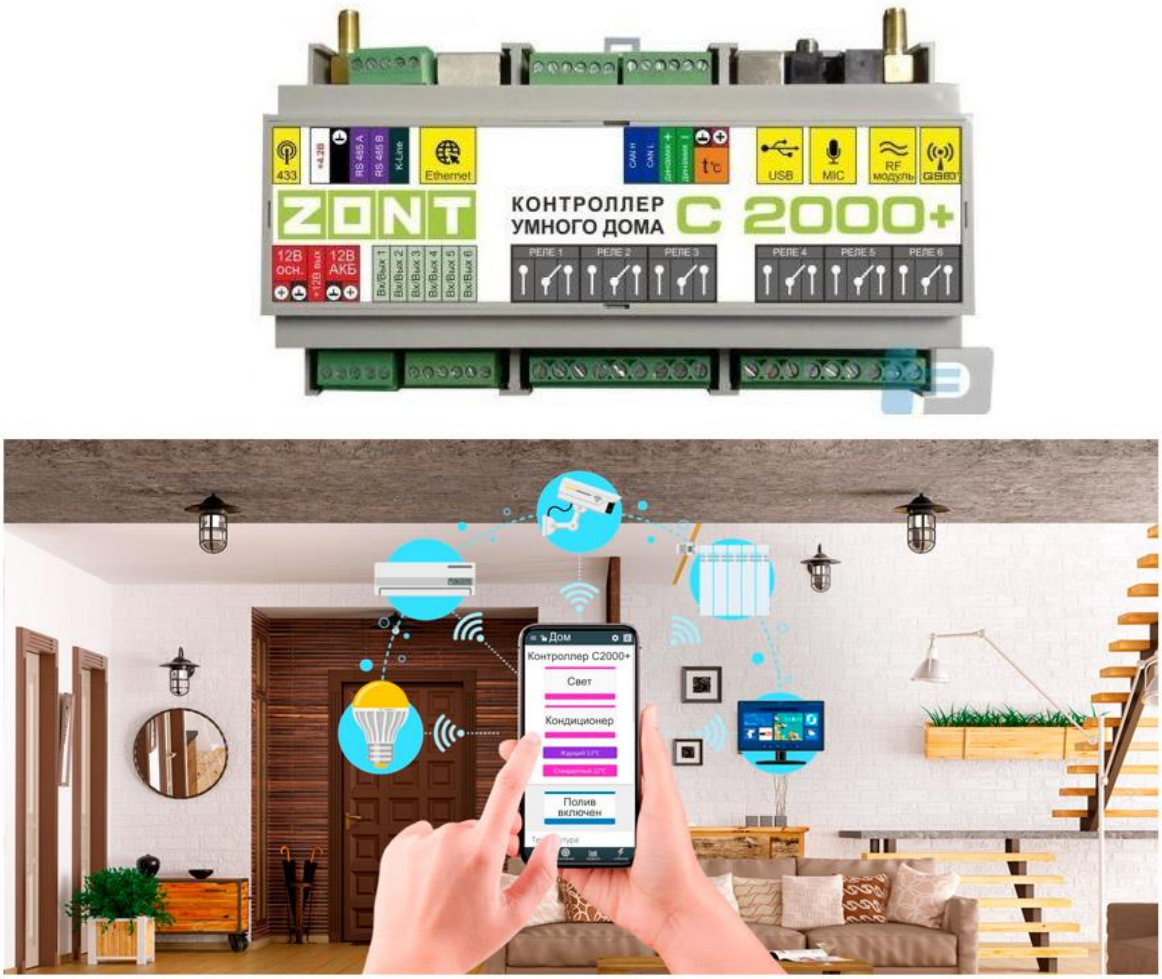

Рисунок 2 – контроллер ZONT C-2000+ и управление со смартфона

При подключении управляемых элементов (сервоприводов, насосов, котлов в релейном режиме, сервоприводов теплого пола, бойлера ГВС и пр.) к устройствам ZONT C-2000+ и панели расширения ZE-66 к универсальным выходам (6 шт.) производитель рекомендует обязательное использование промежуточного реле РЭК78/3. [5]

1.3 Формулировка задач работы

В соответствии с заданием, необходимо разработать микропроцессорный четырехканальный электронный таймер с установкой времени срабатывания по истечении от 1 минуты до 24 часов с периодичностью включения таймера от 1 суток до 30 суток. Исходя из требуемого количества считаемых импульсов в характерных интервалах:

- 1 мин. = 60 с. * 1 000 мс = 60 000 мс;
- 1 сут. = 24 ч. * 60 мин. = 1 440 мин.;
- 30 сут. = 1 440 мин. * 30 = 43 200 мин.,

которые меньше 2^{16} = 65 536, поэтому выбираем шестнадцати битный таймер.

Таким образом, структура разрабатываемого устройства должна состоять из следующих блоков:

- микроконтроллер (МК);
- панель программирования и индикатор МК;
- блок коммутации нагрузок;
- блок питания AC 220В, 50Гц / DC 5В.

Для функционирования 4-х канального электронного таймера необходимо написать программу (sketch), выполняющую следующие функции:

- выбор канала управления;
- установка времени включения таймера в пределах суток;

- установка времени выключения таймера в пределах суток;

- установка периодичности повторения таймера;

- сохранение настроек таймера в энергонезависимую память;

- переход из режима настройки в режим индикации по истечении таймера индикации автоматически или по команде с панели управления вручную;

- индикация состояния всех таймеров на одном экране.

Разработанный Sketch записывается в постоянную память МК, через соответствующие программные средства (IDE) с ПК, в соответствии с инструкцией по программированию МК.

2 Основанная часть

2.1 Выбор комплектующих таймера

В качестве МК выбран Arduino MEGA2560, имеющий в своем составе 6 программируемых аппаратных таймеров: 2 восьми битных (макс. отсчитываемое значение 256) и 4 шестнадцати битных (макс. отсчитываемое значение 65 536). [16]

Другие платы Arduino на основе 8-ми битного процессора, и подобные им, содержат в своем составе аппаратный 8-ми битный таймер, за исключением платы Duo, имеющей 32-битный процессор и более высокую стоимость. Другой возможный вариант решения задачи – использование внешних часов реального времени (RTC) с кварцевой стабилизацией и батарейным питанием, например, DS3231. [25]

МК Arduino MEGA2560 построен на основе БИС ATmega2560 с тактовой частотой 16МГц. МК имеет 54 порта ввода/вывода; 15 выходных портов могут работать, как выход с ШИМ сигналом (аналоговый выход); 16 аналоговых входов с АЦП могут обрабатывать сигналы с аналоговых датчиков. Ток выдаваемый каждым портом в нагрузку – не более 40 мА. [21]

Внешнее питание 7÷20В с понижением через встроенный в плату стабилизатор 5В, с током нагрузки до 800мА. Разъем внешнего питания под штекер диаметром 2,1мм с плюсом по середине.

Для хранения программы предусмотрена Flash-память объемом 256 КБ (8 КБ из них используются загрузчиком); оперативная память ОЗУ SRAM объемом 8 КБ; энергонезависимая память EEPROM объемом 4 КБ. [13] Для связи с разными устройствами предусмотрено 4 UART интерфейса, в их роли выступают выводы 0, 1, 14-19. Один из портов направлен на USB через микроконтроллер ATmega16U2. [12] На плате предусмотрены разъемы для установки плат расширения Shield. Размеры платы: 53x110x12,5мм. Внешний вид платы с двух сторон приведен на рисунке 3.

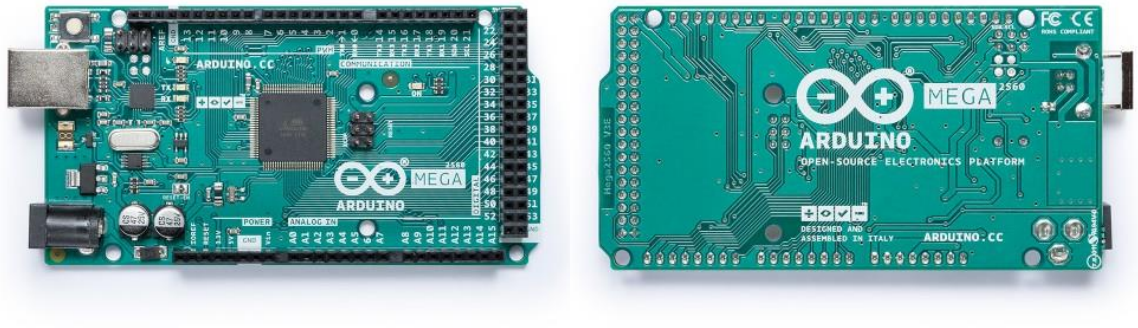

Рисунок 3 – Внешний вид платы Arduino MEGA2560

Панель программирования таймеров и индикатор МК, выбраны с учетом совместимости с МК Arduino MEGA2560 и стоимости реализации, в виде готовой платы расширения Arduino LCD Keypad Shield. [2] Плата содержит индикатор LCD1602, позволяющий отображать два ряда по 16 символов, подключенный к четырем цифровым выходам Arduino D4-D7; шесть кнопок (RIGHT, UP, DOWN, LEFT, SELECT), подключенных к аналоговому входу Arduino A0; кнопка RESET, подключенная к входу RST; резистор настройки контрастности дисплея LCD. Размеры платы 60x83x20мм. Внешний вид платы с двух сторон приведен на рисунке 4.

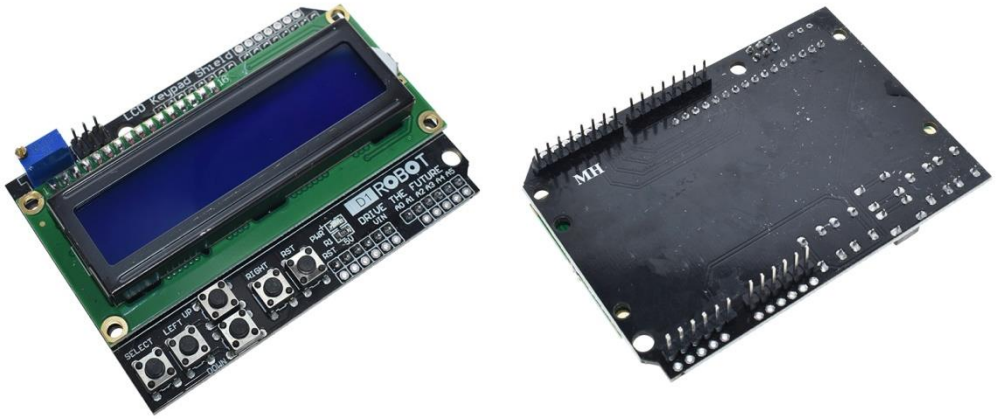

Рисунок 4 – Внешний вид платы расширения LCD Keypad МК

В качестве блока коммутации нагрузок выбрана готовая плата RobotDyn AC Light Dimmer 4 channel, имеющая четыре не зависимых канала регулировки нагрузок переменного напряжения 220/380В с частотой 50/60Гц и током до 16А. На плате размещены 4 симистора BTA16-600V с оптопарами MOC3021S. Есть детектор перехода напряжения питания 220В через ноль. Плата предназначена для плавной регулировки света, но в данном проекте будет использоваться для коммутации нагрузок. [19] Выбор готовой платы сделан исходя из соображений невысокой стоимости и удобства применения данной платы. Размеры платы 66x113x30мм. Внешний вид платы с двух сторон приведен на рисунке 5.

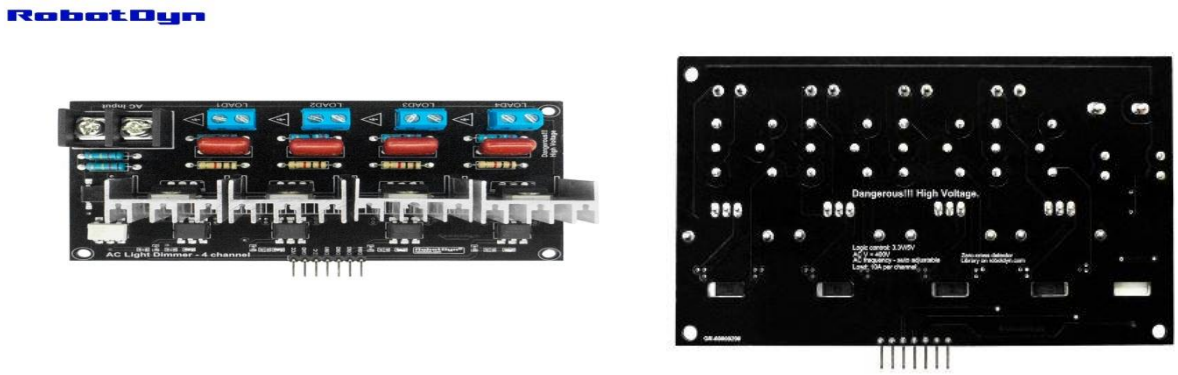

Рисунок 5 – Внешний вид платы коммутации нагрузок

На рисунке 6 изображен соединительный кабель с наконечниками DuPont «папа-мама», используемый для соединения платы МК с платой коммутации нагрузок.

Рисунок 6 – Соединительный кабель с наконечниками DuPont

Блок питания питает низковольтную часть электронного таймера напряжением 9В. Питание поступает на плату МК, где расположен стабилизатор на микросхеме MC33269D-5.0 (аналог LM1117-50) [25], формирующий 5В с током нагрузки до 800мА. Все платы устройства питаются от 5В, при этом токи потребления:

МК Arduino MEGA2560 (CPU+I/O) $I_{потр}$≈50мА;

LCD Keypad Shield (LCD) $I_{потр}$≈1мА + 130мА (Back Light LED);

RobotDyn AC Light Dimmer 4 channel $I_{потр}$≈20мА*4=80мА.

Итого суммарно: $I_{потр}$≈260мА.

Выберем блок питания импульсного типа со стабилизацией AC 220В, 50Гц / DC 9В, 600мА. Блок питания импульсный стабилизированный с ЭМС-фильтром на входе. В этом случае на стабилизаторе 5В будет выделяться мощность порядка 1Вт. [13] Можно уменьшить мощность рассеяния до 0,5Вт, перенеся питание Back Light LED на шину VIN=9В. Размеры платы: 51х24х19мм. Внешний вид платы и кабеля питания платы МК приведен на рисунке 7.

Рисунок 7 – Внешний вид платы блока питания с разъемом DC5.5-2.1

Металлический корпус с розетками для подключения нагрузок разрабатывается для крепления вертикально на стену, для чего по бокам корпуса предусмотрены крепежные уголки. На передней панели

располагаются: расположенные в ряд розетки для включения нагрузок, ниже - экран LCD и кнопки управления электронным таймером. Передняя панель съемная для сервисного обслуживания и доступа к USB МК. Регулировка контрастности LCD и кнопка RESET для сброса МК доступны через технологические отверстия диаметром 3мм рядом с экраном под отвертку. Для подключения общего питания устройства на боковой панели корпуса устройства размещен разъем типа IEK320-C14 в колодке с предохранителем и общим выключателем питания. Передняя панель съемная для сервисного обслуживания и доступа к USB МК. [5]

2.2 Разработка принципиальной схемы таймера

Разрабатываемое устройство полностью собирается из готовых плат для технического творчества (DIY) и прототипирования.

Принципиальная схема основной процессорной части платы Arduino MEGA2560 изображена на рисунке 8.

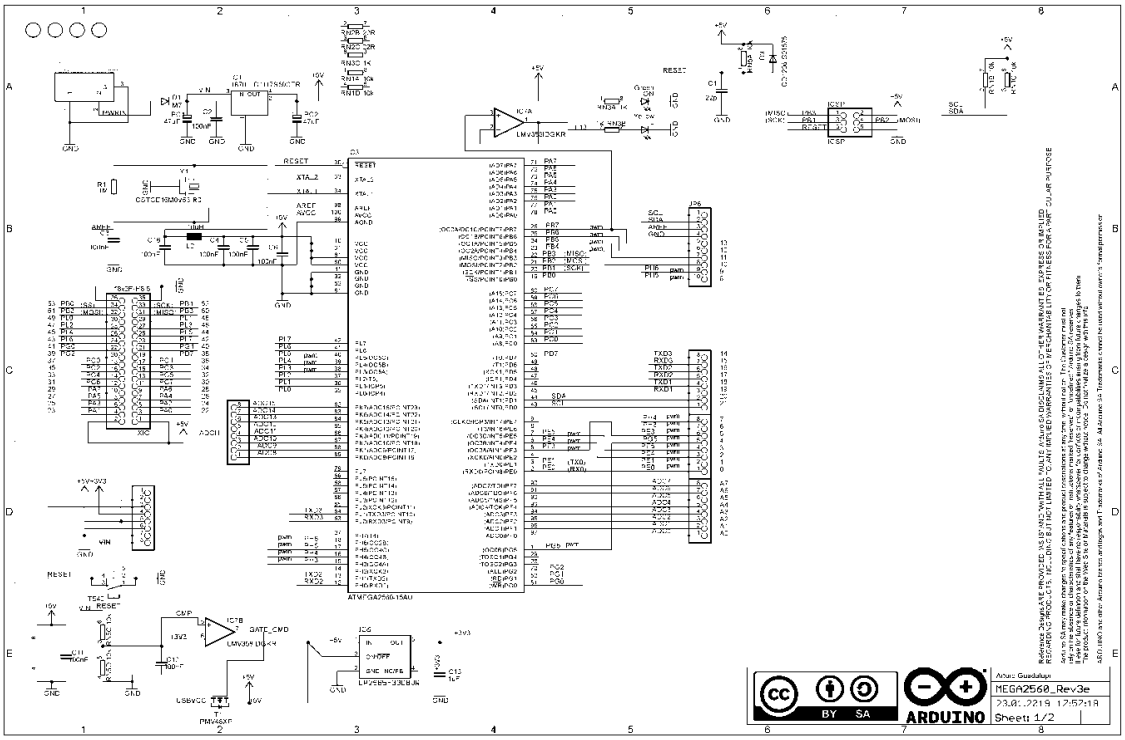

Рисунок 8 – Принципиальная схема основной процессорной части МК

На схеме отображены элементы: микроконтроллер ATmega2560-15AU; кварцевый резонатор на 16МГц, задающий основную тактовую частоту; интегральный стабилизатор на 5В типа LD1117S50 или аналог; интегральный стабилизатор на 3,3В типа LP2985-33 или аналог [11]; индикаторы на светодиодах: зеленый показывает наличие питания МК, желтый светодиод управляется от порта A13 через повторитель на ОУ LM358, второй ОУ этой микросхемы задействован как компаратор, обнаруживающий наличие питания по выводу VIN и дающий команду ключевому транзистору T1 на подачу питания 5В на USB, благодаря защитному обратному диоду этого транзистора возможно питание платы от USB, но не желательно; несколько элементов обвязки микроконтроллера обеспечивающих нормальный режим работы микропроцессора.

Принципиальная схема дополнительной части платы Arduino MEGA2560, выполняющей работу с USB, изображена на рис. 9.

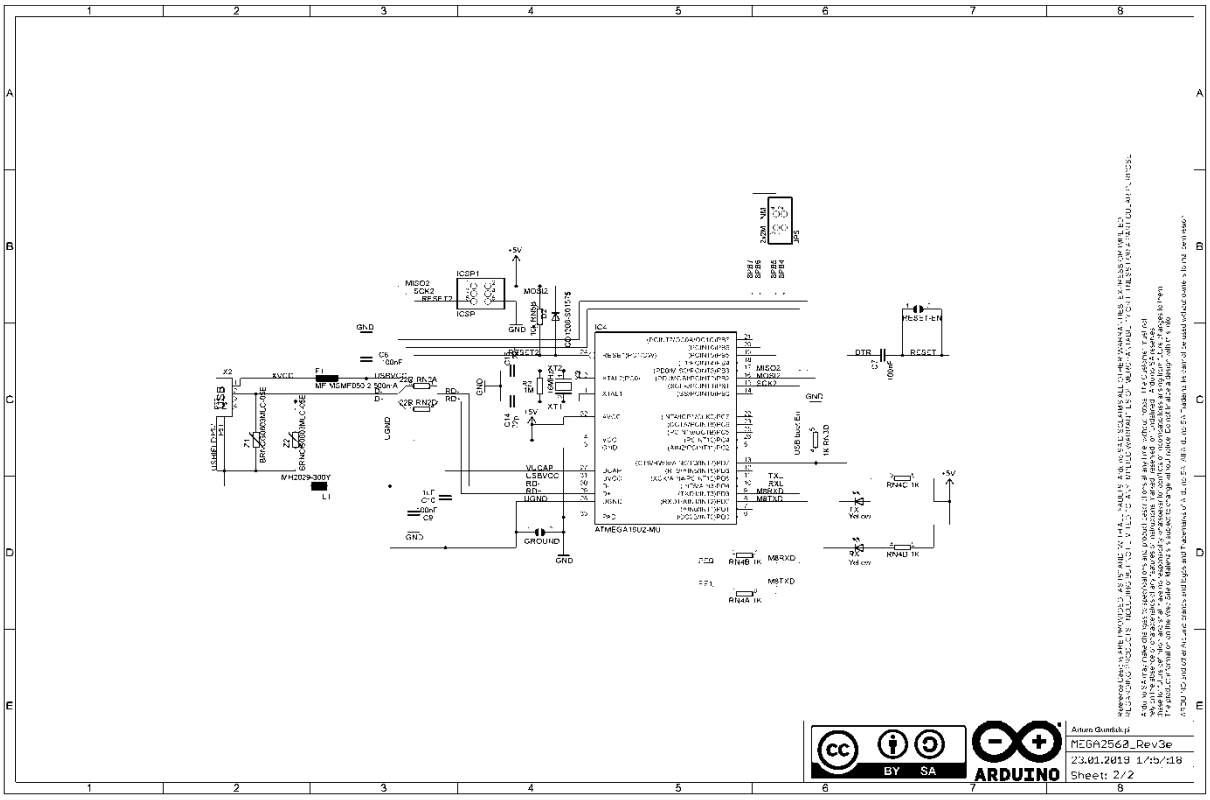

Рисунок 9 – Принципиальная схема контролера USB МК

На схеме изображены элементы: микроконтроллер ATmega16U2-MU; кварцевый резонатор на 16МГц, задающий основную тактовую частоту этого контроллера; USB-порт с элементами обвязки и защиты от перенапряжения; светодиоды, показывающие режимы работы микроконтроллера и несколько элементов обвязки микроконтроллера обеспечивающих нормальный режим работы микропроцессора.

На рисунке 10 приведена принципиальная схема платы управления и индикации Keypad LCD Shield [7]. На схеме изображены: разъемы платы расширения J1-J7 и ICSP; светодиод индикации питания VCC; кнопка RST, подключённая к выводу RST платы Arduino, необходимая для сброса микроконтроллера, т.к. плата расширения закрывает кнопку RESET на плате Arduino; кнопки RIGHT, UP, DOWN, LEFT и SELECT с резисторной матрицей подключаются к аналоговому входу AD0 платы Arduino [22]; элементы индикатора LCD1602: резистор регулировки контрастности изображения RP1, транзистор коммутации подсветки Q1 по выводу D10 платы Arduino.

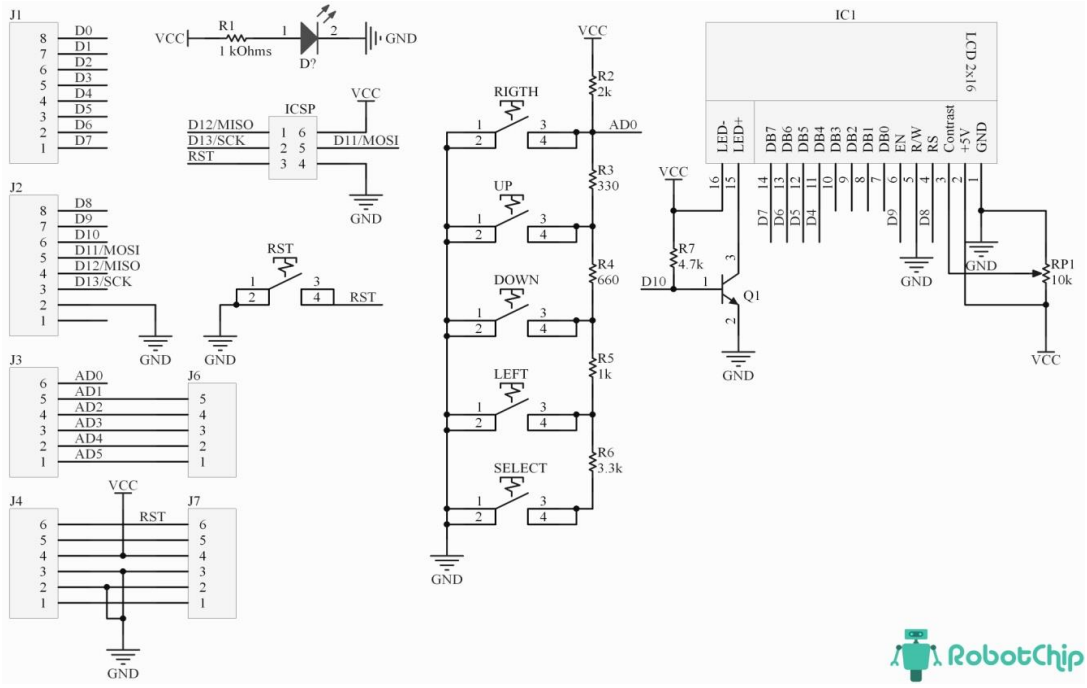

Рисунок 10 – Принципиальная схема платы индикации и управления

На рисунке 11 изображена часть схемы AC Light Dimmer Module. Схема выполнена на 4 симмисторах Q1 типа BTA16-600V с оптической развязкой D3 на динисторных оптопарах MOC3021S и индикацией состояния входа на светодиоде D4 [20]. Резисторы R4-R6 служат для ограничения тока через полупроводниковые элементы и задания рабочих режимов. На оптопаре D2, диодном мосте D1 и ограничительных резисторах R1-R3 построена схема детектора нуля, необходимая для правильного переключения симмистора во время перехода сетевого напряжения питания нагрузки через нуль.

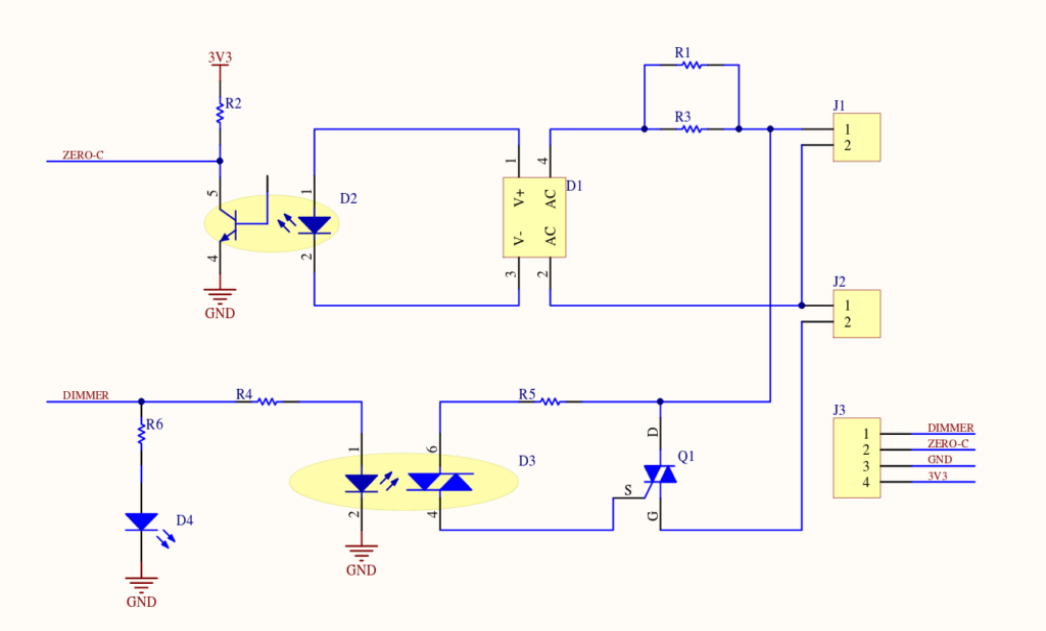

Рисунок 11 – Часть принципиальной схемы платы коммутации нагрузок

На рисунке 12 приведена принципиальная схема импульсного блока питания Coreset SM-PLG06A-09 [3], собранного по классической трансформаторной схеме однотактного ИППН на специализированной интегральной микросхеме с входным фильтром ЭМИ, уменьшающим проникновение высокочастотных пульсаций в питающую сеть, регулировкой выходного напряжения через оптопару и интегральный стабилизатор типа TL431.

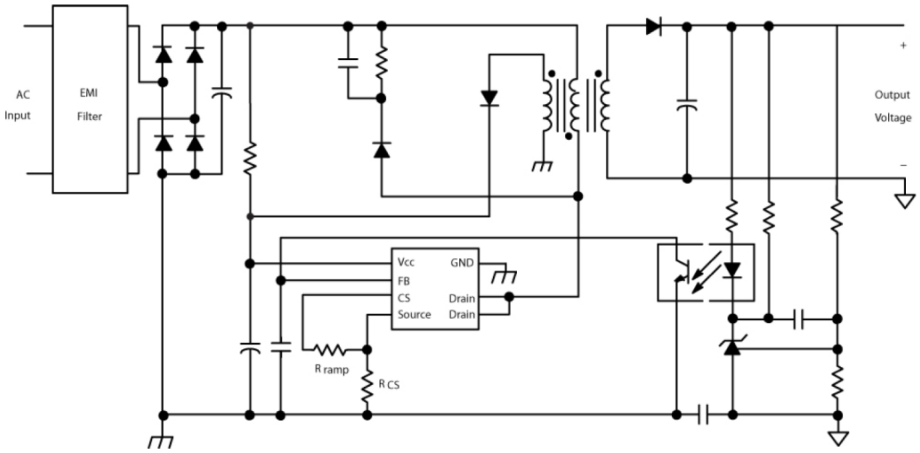

Рисунок 12 – Принципиальная схема импульсного блока питания

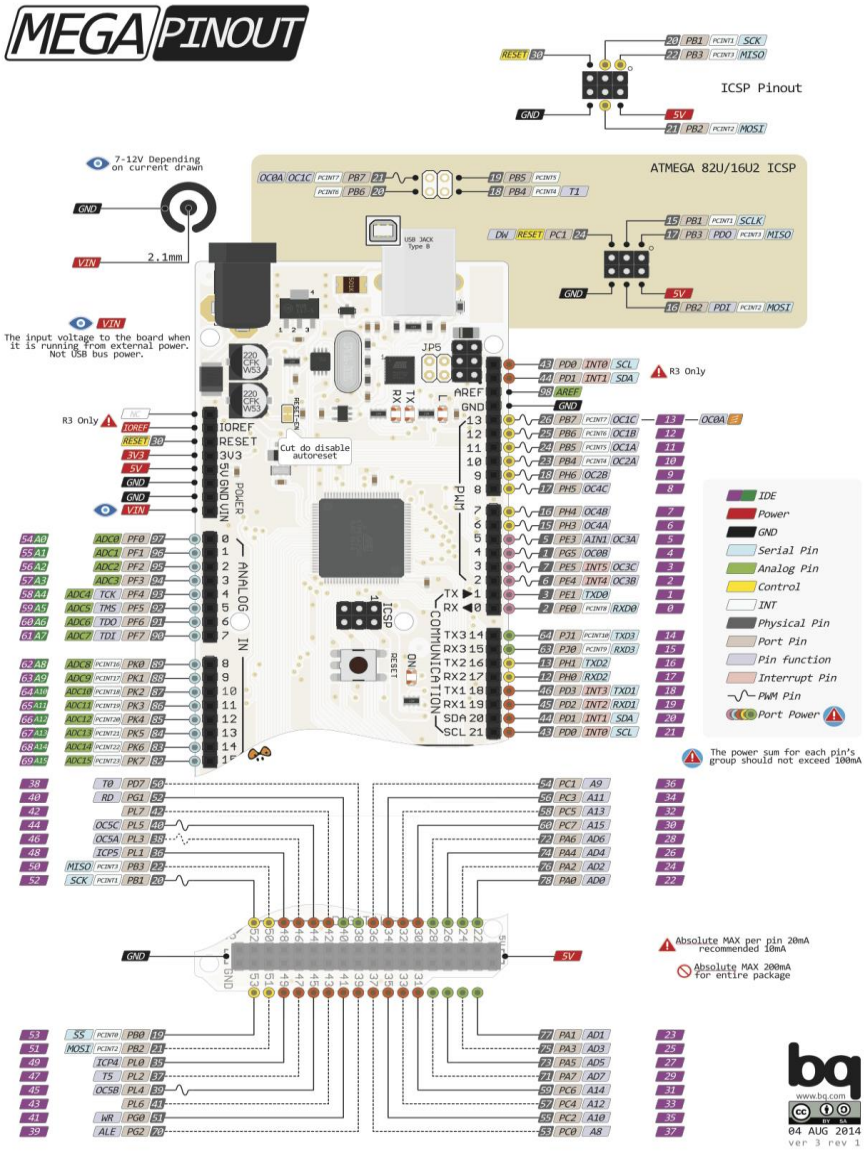

Рисунок 13 – Данные по назначению контактов на плате МК

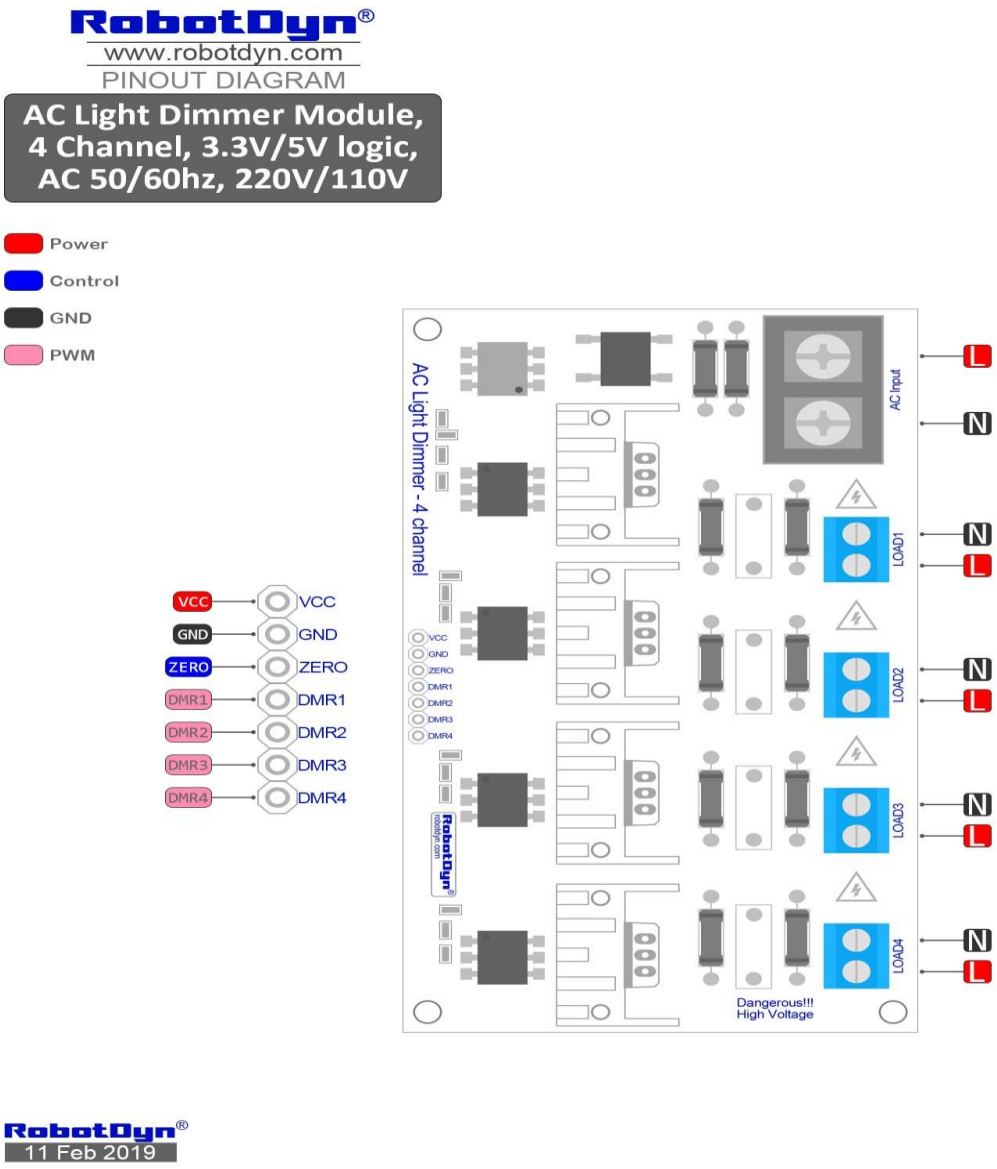

Рисунок 14 – Назначение контактов платы коммутации нагрузок

На рисунке 13 приведены данные по назначению контактов на плате МК Arduino MEGA2560. На рисунке 14 приведено назначение контактов платы коммутации нагрузок RobotDyn AC Light Dimmer 4 channel [23].

2.3 Разработка конструкции системы

Для устройства выбран вертикальный корпус размерами 150x200x50мм. Передняя панель выполнена с вырезами под розетки, общий

выключатель питания, индикатор и подпружиненные толкатели кнопок. Применение металлической передней панели с металлическими толкателями кнопок управления уменьшает возможность статического пробоя МК при программировании таймеров. Часть передней панели открывается дверкой для предоставления доступа к USB порту МК для программирования с внешнего персонального компьютера.

Для подключения нагрузок применим 4 панельные скрытые розетки промышленного типа с крышками (рис.15). Размер розеток 50х50мм, с учетом шарнира крышки 65мм, глубина установки 36мм, высота крышки 15мм. Диаметр отверстия в панели под установку розетки 43мм. Розетки крепятся к панели 4-мя болтами М4, через 4 отверстия диаметром 5мм, шарнирами крышек к индикатору МК.

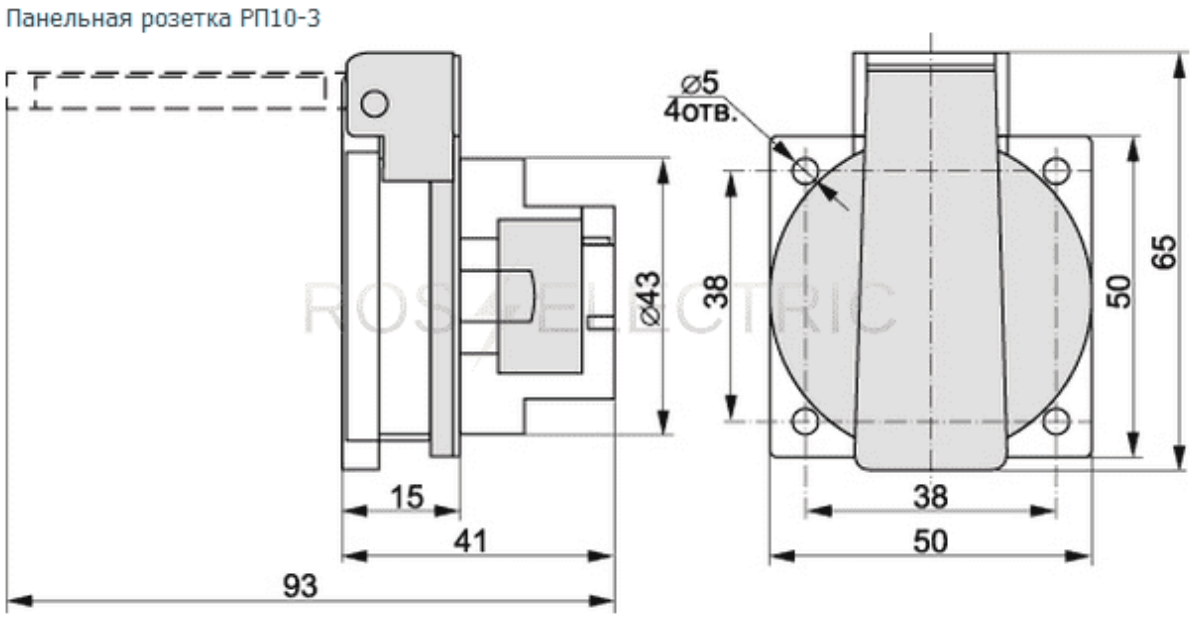

Рисунок 15 – Панельная промышленная розетка 220В/16А

Все платы внутри корпуса располагаются вертикально на монтажных стойках, причем, платы индикатора и МК крепятся внутри корпуса к передней панели корпуса. Плата коммутации нагрузок и блок питания крепятся на задней крышке корпуса электронного таймера. Плата индикации устанавливается в разъемы платы МК (рис.16) как плата расширения (Shield).

Соединение МК с платой коммутации нагрузок осуществляется плоским 7-ми проводным кабелем с наконечниками DuPont «папа-мама» (рис. 6). Питание 9В от блока питания подключается к МК проводом с штекером DC5.5*2,1 (рис. 3, 7, 13).

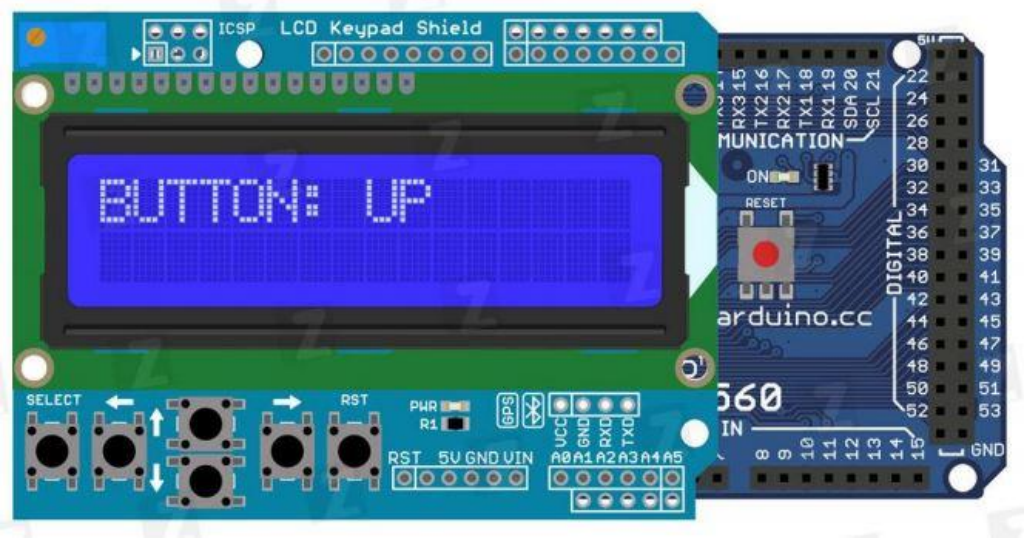

Рисунок 16 – Вид платы индикации подключенной к плате МК

Кабель питания электронного таймера подключается через 3-х контактный разъем типа IEC320-C14 с выключателем и колодкой предохранителя модели Multicomp JR-101-1-FRSG-03 (рис.17) [13]. На боковой панели под него делается прямоугольное отверстие размерами 28мм на 48мм.

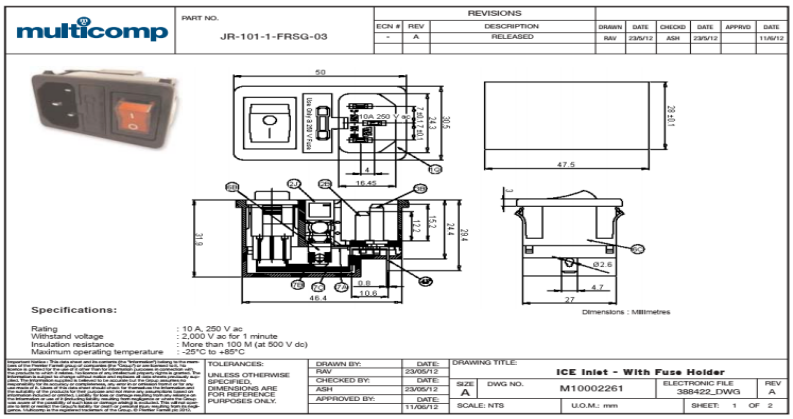

Рисунок 17 – Модуль подключения кабеля питания

Размеры основных плат, с отметками крепежных отверстий диаметром 3,2мм под винт М3, приведены на рисунках 18-21.

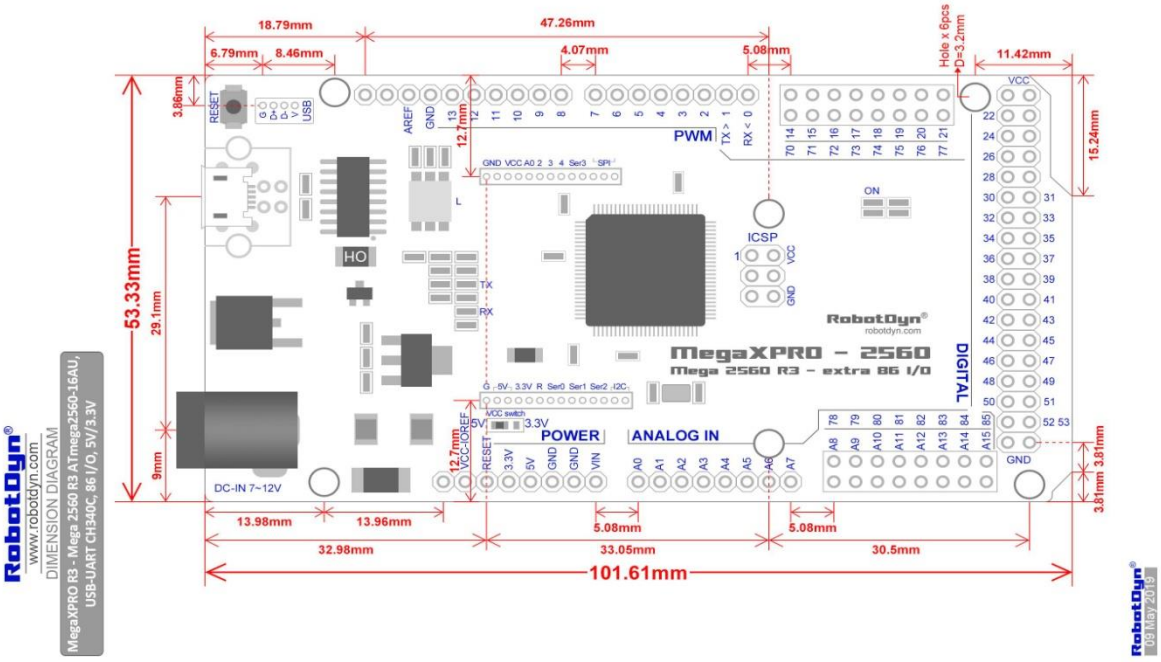

Рисунок 18 – Модуль микроконтроллера

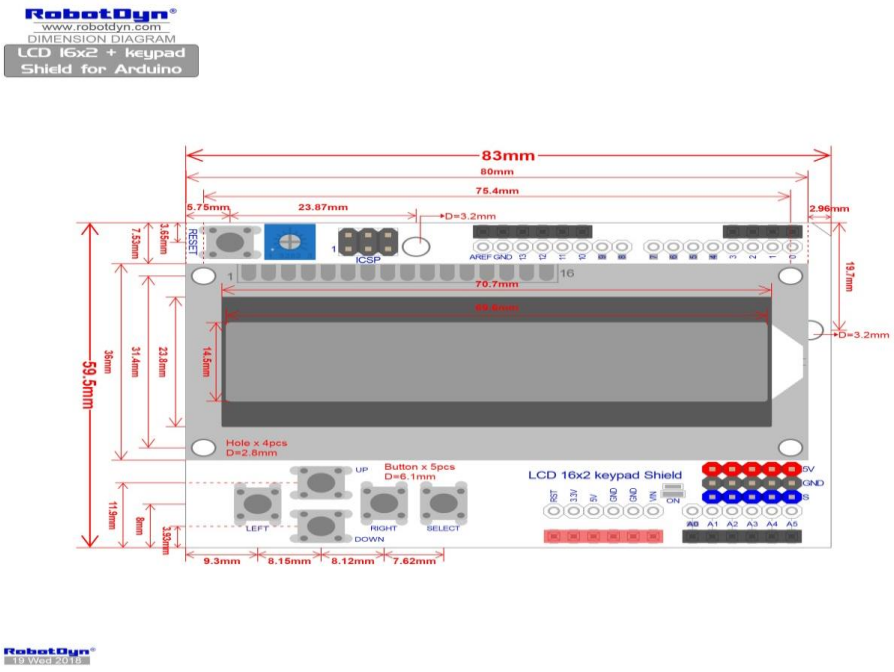

Рисунок 19 – Модуль индикатора и панели управления

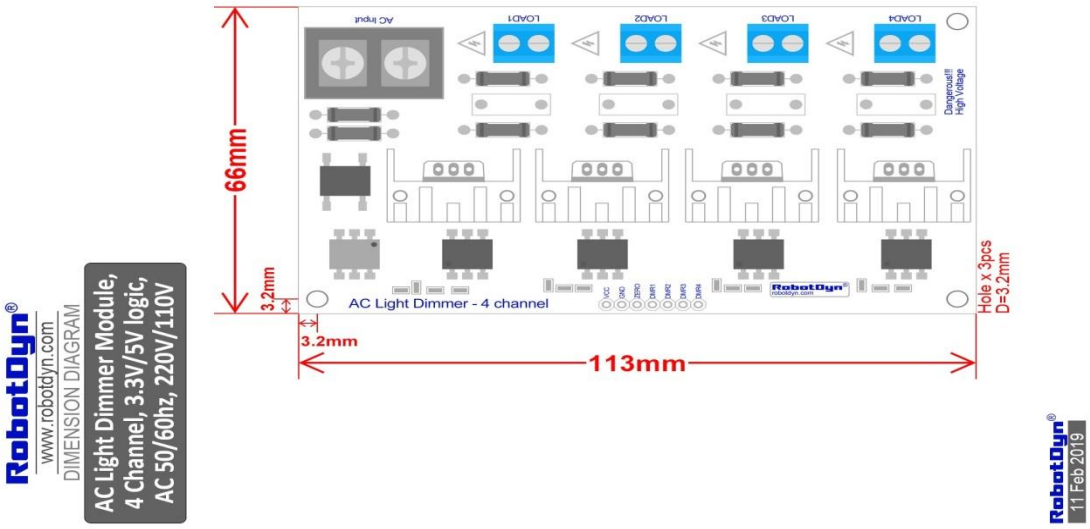

Рисунок 20 – Модуль коммутатора нагрузок

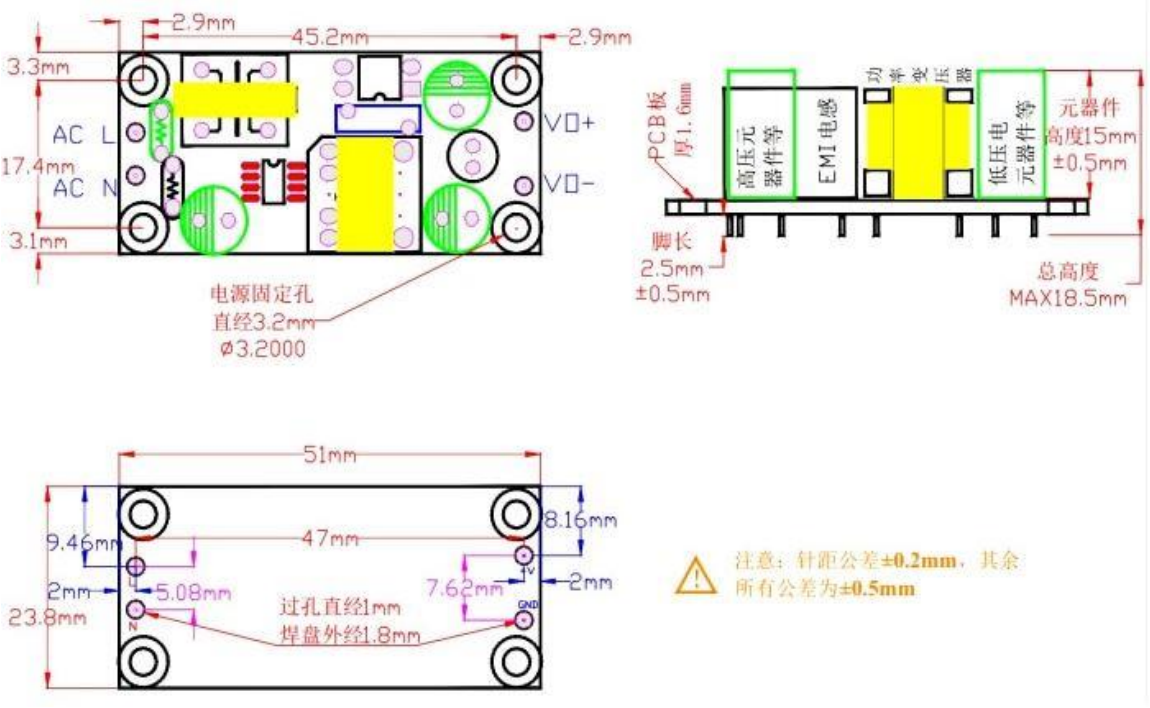

Рисунок 21 – Модуль питания

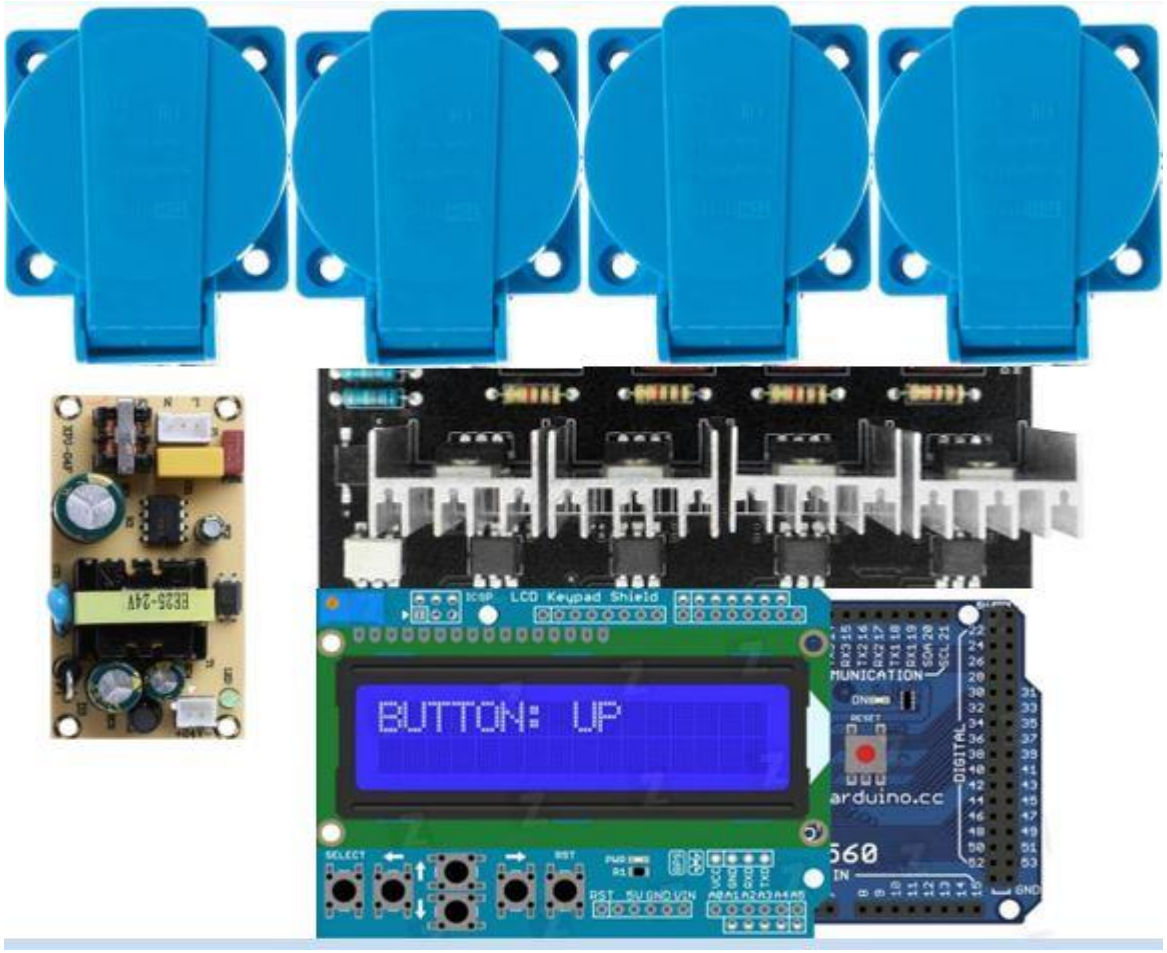

Рисунок 22 – Взаимное расположение плат в собранном устройстве

На рисунке 22 изображено взаимное расположение плат устройства. Вверху четыре розетки, закрепленные на лицевой поверхности корпуса устройства, чуть ниже слева блок питания МК, справа – плата коммутации нагрузок, крепящиеся к задней стенке корпуса устройства, а ниже по рисунку, над платой коммутации нагрузок, плата Arduino MEGA2560 с платой индикации Key LCD Shield закрепленные на лицевой части корпуса устройства [1].

2.4 Разработка алгоритма программы таймера

Для функционирования 4-х канального электронного таймера необходимо написать программу выполняющую следующие функции:

- выбор канала управления;

- установка времени включения таймера в пределах суток;

- установка времени выключения таймера в пределах суток;

- установка периодичности повторения таймера;

- сохранение настроек таймера в энергонезависимую память;

- переход из режима настройки в режим индикации по истечении таймера индикации автоматически или по команде с панели управления вручную;

- индикация состояния всех таймеров на одном экране.

Для МК Arduino пишется программа (sketch, прошивка) на языке программирования C/C++ в среде программирования Arduino IDE [6], состоящая из:

- программный модуль (функция) формирования минутных импульсов, в котором задействуется аппаратный 16 битный таймер Atmega 2560, через функцию millis() [8], который отсчитывает миллисекунды и, по достижении 60 секунд, записывает соответствующее значение в переменную минут, перезапускает малый цикл программы каждую минуту;

- программный модуль (функция) формирования суточных импульсов, который отсчитывает минуты из данных первого таймера и по достижении 60 минут записывает соответствующее значение в переменную часов, по достижении значения 1440 регистрируется суточный интервал, а по достижении значения установленного в переменную повторения суточного цикла запускается соответствующий таймер;

- программный модуль вычисления текущего времени из переменных часов и минут с учетом начальной установки часов;

- 4 программных модуля (функции) счетчиков электронных таймеров с формированием сигналов управления блоком коммутации нагрузок, сравнивают текущее значение переменных минут и суток с заданным и дают команды на включение и выключение нагрузок;

- программный модуль (функция) формирования информации на

индикаторе МК, обрабатывает формирование отображения символов в навигации по меню настройки устройства;

- программный модуль (функция) кнопок панели программирования МК, обрабатывает аналоговый сигнал с панели кнопок и обеспечивает управление навигацией по меню настройки устройства;

- программный модуль (функция) записи/воспроизведения значений переменных в энергонезависимую память.

Индикатор показывает время в часах и минутах, разделяющий их знак «:» мигает с периодичностью в 1 секунду, что говорит об основном режиме работы индикатора. В режиме настройки времени знак «:» не мигает, но мигает цифра или знак, которые в данный момент изменяем.

В основном режиме работы индикатора возможны несколько видов экрана:

1) основной экран - индикатор устройства показывает время всех четырех таймеров (T1-T4) в часах и минутах «hh:mm» и текущее состояние каждого таймера (включен «*» или выключен « ») - рис.22. Нажатием кнопок LEFT/RIGHT можно переключаться между видами экрана состояния таймеров устройства по кругу. Нажатием кнопок UP/DOWN можно перейти к экрану текущего времени. Нажатием кнопки SELECT можно перейти к настройкам таймеров по 4 каналам устройства [10];

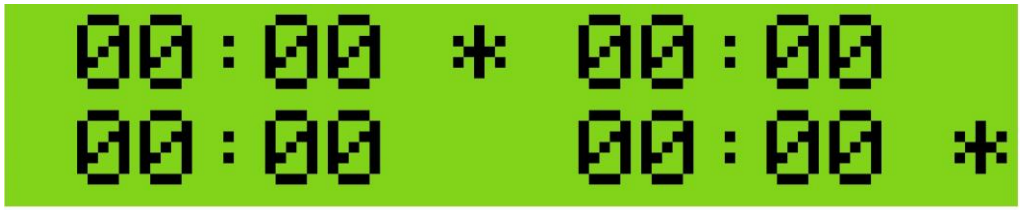

Рисунок 22 – Основной экран

2) экран текущего времени - индикатор показывает текущее время «Now time» в часах и минутах «hh:mm» и время, прошедшее с

момента включения устройства в днях «uptime ** day» - рис.23 [17]. Нажатием кнопки SELECT можно перейти к настройкам текущего времени, где нажатием кнопок LEFT/RIGHT можно переключаться между цифрами, а нажатием кнопок UP/DOWN можно изменять их. Отображение изменяемой в данный момент времени цифры будет мигать с периодичностью в 1 секунду, знак «:» не мигает. Нажатием кнопки SELECT установка текущего времени записывается в энергонезависимую память;

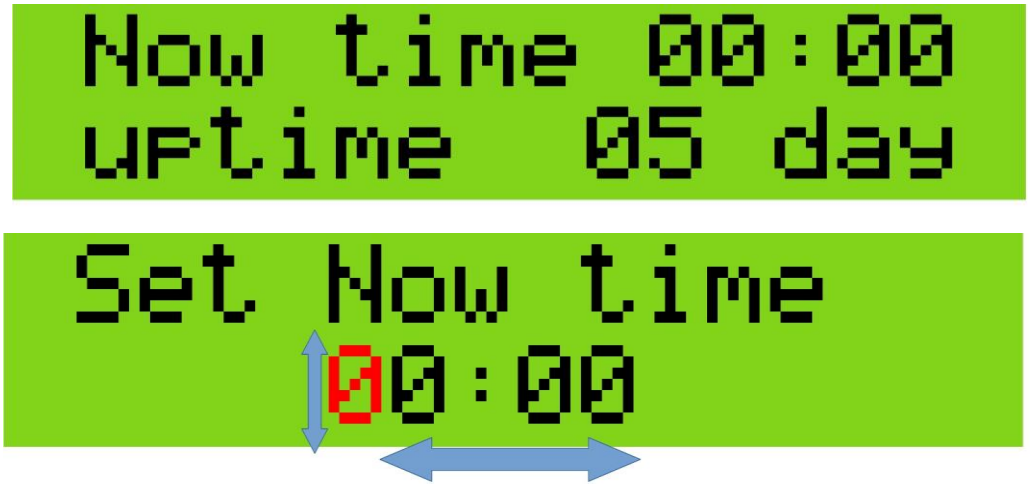

Рисунок 23 – Экран текущего времени

3) индикатор устройства показывает состояние (включен «ON» или выключен «OFF») одного текущего таймера «T*» в минутах «****m», его и периодичность включения таймера «Period ** day» - рис.24 или о прошедшем времени после последнего изменения состояния «PASTtime ****m» на рис.25 или об оставшемся времени до следующего изменения состояния таймера «END time ****m» на рис.26 [4];

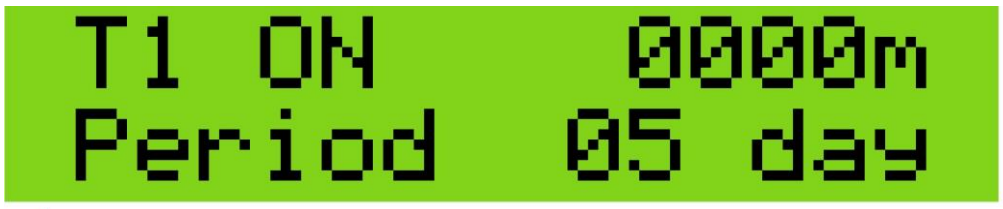

Рисунок 24 – Экран длительности и периодичности включения текущего таймера

Рисунок 25 – Экран прошедшего времени с момента включения текущего таймера

Рисунок 26 – Экран оставшегося времени работы текущего таймера

Нажатием кнопки SELECT можно перейти к настройкам таймеров устройства (рис.27), где нажатием кнопок LEFT/RIGHT можно переключаться между таймерами, а нажатием кнопок UP/DOWN можно изменять их настройки. Отображение изменяемого в данный момент времени числа будет мигать с периодичностью в 1 секунду. Повторное нажатие кнопки SELECT переключает экран настройки таймера между длительностью и периодичностью включения таймера. Нажатием кнопки SELECT настройки таймера записываются в энергонезависимую память и устройство переходит к отображению основного экрана (рис.22);

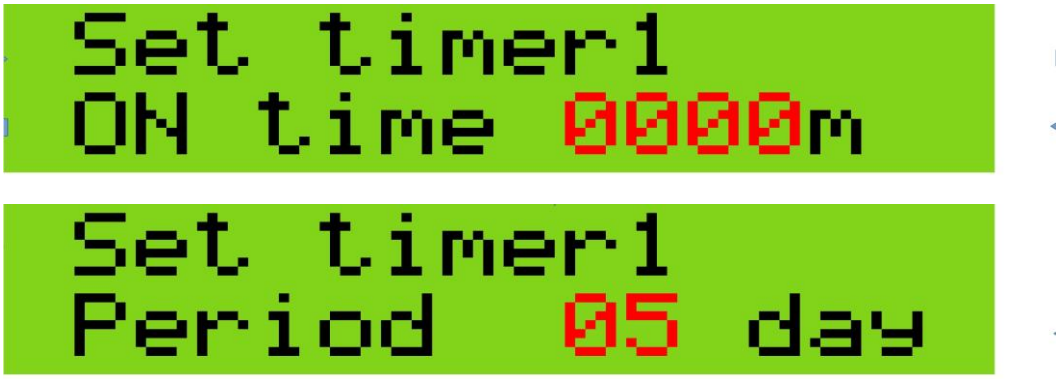

Рисунок 27 – Экраны настройки таймеров

На рис.28 отображен алгоритм работы экранного меню в упрощенном виде – отображение состояния и настроек 2 и 3 таймеров аналогично 1 и 4 таймеров, поэтому для наглядности они изображены серым прямоугольником.

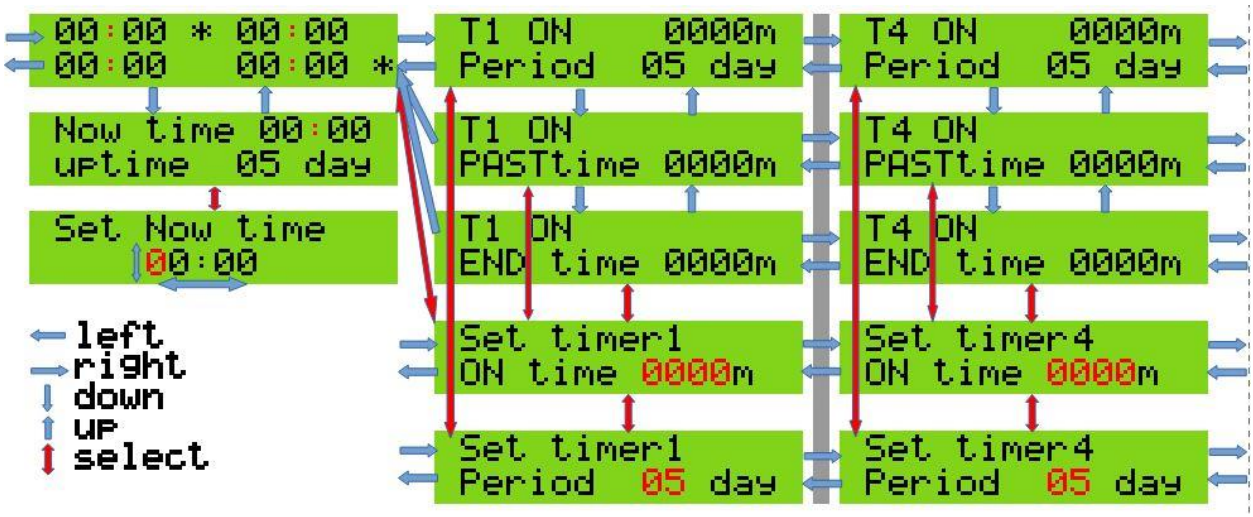

Рисунок 28 – Алгоритм работы экранного меню

Синими стрелками изображены переходы между экранами при нажатии кнопок LEFT/RIGHT и UP/DOWN. Красными стрелками изображен переход между экранами при нажатии кнопки SELECT. Красным шрифтом на экранах изображены области, мигающие с частотой 1 Гц.

2.5 Составление программы по разработанному алгоритму

По разработанному уникальному алгоритму произведено составление программы в интегрированной среде разработки Arduino IDE на языке программирования C/C++ [24].

Подключаем библиотеки дисплея, энергонезависимой памяти и математических формул:

```
#include <LiquidCrystal.h>
#include <EEPROM.h>
#include <MATH.h>
```

Активируем и назначаем выводы МК для подключения дисплея:

```
LiquidCrystal lcd (12, 11, 9, 8, 7, 6, 5, 4, 3, 2);
```

Задаем переменные необходимые для работы программы устройства и резервируем под них память МК:

```
int rtcM=0, rtcH=0, rtcS=0;          //начальные значения переменных времени
int p1set=0, p2set=0, p3set=0, p4set=0, t1set=0, t2set=0, t3set=0, t4set=0; //начальные значения установок таймеров
int timer1=0, timer2=0, timer3=0, timer4=0, period1=0, period2=0, period3=0, period4=0;  //счетчики таймеров
int D=0, Dig=1, Sel=0, Dx=0;    //начальный экран Display(0…4), первая цифра
bool b=0;                       //переменная отображения информации на экране
bool t1=0, t2=0, t3=0, t4=0;          //переменные состояния таймеров
```

Раздел основных настроек МК:

```
void setup() {
  lcd.begin(16, 2);          // инициализируем дисплей: 2 строки по 16 символов
  readSet();                        // считываем ранее сохраненные настройки
  pinMode(22, OUTPUT);                        // назначение порта 22 на выход управление нагрузкой таймера 1
  pinMode(24, OUTPUT);                        // назначение порта 24 на выход управление нагрузкой таймера 2
  pinMode(26, OUTPUT);       // назначение порта 26 на выход управление нагрузкой таймера 3
  pinMode(28, OUTPUT);    // назначение порта 28 на выход управление нагрузкой таймера 4
}
```

Раздел основного цикла программы устройства:

```
void loop() {
  xKey();                          //запуск обработки нажатий клавиш
  timersWork();                    //запуск обработки таймеров
```

```
 setOut();                          //установка уровней выходов таймеров
}
```

Создаем функцию обработки нажатия кнопок:

```
void xKey(){
 int uKey= analogRead (0);       // Создаем переменную uKey и опрашиваем аналоговый порт A0, к которому подключены кнопки
 if (uKey > 50){                  // отсекаем шум, если нажата кнопка
  if (uKey < 100) {                  // Если uKey меньше 100 нажата кнопка "Вправо" Right
    if(Sel<1) {                  // Переход к следующему таймеру
     if(D<5) D++; else D=0;
     }
    else {                       // Переход к следующей цифре
     if(Dig<5) Dig++; else Dig=1;
     }
    }
  else if (uKey < 200) {                  // Если uKey меньше 200 нажата кнопка "Вверх" Up, то в зависимости от вида представления
    if(D==0){                    // основное окно, отображаются 4 таймера, переход к отображению текущего времени
      if(Sel>0) {
       if(Dx<10) Dx++; else Dx=0;
       }
      else if(Dx<1) Dx++; else Dx=0;}
    else {                       // таймер 1-4
     if(Sel==0) {
       if(Dx<3) Dx++; else Dx=0;
       }
       else if(Sel==1) {
```

```
        if(Dx<1441) Dx++; else Dx=1;
        }
        else {
          if(Dx<31) Dx++; else Dx=1;
          }
       }
    }
  else if (uKey < 400){                    // Если uKey меньше 400 нажата кнопка
"Вниз" Down, то в зависимости от вида представления
    if(D==0){                              // основное окно, отображаются 4 таймера,
переход к отображению текущего времени
      if(Sel>0) {
        if(Dx>0) Dx--; else Dx=9;
        }
        else if(Dx>0) Dx--; else Dx=1;
        }
     else {                                // таймер 1-4
      if(Sel==0) {
        if(Dx>0) Dx--; else Dx=2;
        }
        else if(Sel==1) {
          if(Dx>1) Dx--; else Dx=1440;}
          else {if(Dx>1) Dx--; else Dx=30;
          }
      }
    }
  else if (uKey < 600){  // Если uKey меньше 600 нажата кнопка "Влево" Left
    if(D>0) D--; else D=4;  // Переход к предыдующему таймеру или цифре
    }
```

```
   else if (uKey < 800){   // Если uKey меньше 800 нажата кнопка "Выбор" SELECT
     if(D>0){ // Переключение режима отображения информации при каждом нажатии на кнопку по кругу
       if(Sel<5) Sel++; else Sel=0;
       }
       else {
         if(Sel<2) Sel++; else Sel=0;
         }
     saveSet();  // Сохранение введенных данных по нажатию кнопки SELECT
     }
  }
  setPlay();                          // Вывод информации на экран
}
```

Создаем функцию обработки счетчиков времени:

```
void timersWork(){
  unsigned long tX = 0;                    //переменная текущего времени
  if(millis()-tX>=60000){                      //если количество милисекунд станет больше 60сек, то
    b=!b;          //инверсия отображения значения переменной отображения
    if(rtcM<60){                        //пока количество минут еще не 60
      rtcM++;                           //увеличиваем минуты на единицу
     } else {                          //а иначе
       rtcM=0;                          //обнуляем количество минут
       if(rtcH<24){                      //пока количество часов еще не 24
         rtcH++;                        //увеличиваем часы на единицу
         } else {
           rtcH=0;                      //а иначе обнуляем количество часов
           if(rtcS<31){                  //пока количество еще не 30
```

```
          rtcS++;                    //увеличиваем сутки на единицу
          if(period1 <= p1set) period1++; else period1=0; //счетчик периода таймера 1
          if(period2 <= p2set) period2++; else period2=0; //счетчик периода таймера 2
          if(period3 <= p3set) period3++; else period3=0; //счетчик периода таймера 3
          if(period4 <= p4set) period4++; else period4=0; //счетчик периода таймера 4
         } else {
          rtcS=0;                    //а иначе обнуляем сутки
         }
        }
       }
   if(timer1 <= t1set) timer1++; else timer1=0;      //счетчик таймера 1
   if(timer2 <= t2set) timer2++; else timer2=0;      //счетчик таймера 2
   if(timer3 <= t3set) timer3++; else timer3=0;      //счетчик таймера 3
   if(timer4 <= t4set) timer4++; else timer4=0;      //счетчик таймера 4
   do{   //в цикле защиты от переполнения таймеров микроконтроллера
    tX+=60000;     //увеличиваем переменную времени на 60 секунд
    if(tX<60000) break;  //если переменная станет меньше 60 секунд, т.е. счетчик переполнится, выходим из цикла
   }while(tX<millis()-60000);    //до тех пор, пока переменная времени меньше чем разница между текущим значением функции времени и 60 секундами
  }
}
```

Создаем функцию управления переключенем экранов:

```
void setPlay(){
 if(Sel<1){                        // основные экраны Sel=0
```

```
  if(D<1){                          // D=0
    if(Dx<1) screen1();             // основной экран Dx=0
    else screenW();                 // часы Dx=1
  }
  else if(Dx<1) screen2();          // Таймеры D=1-4, Dx=0
       else if(Dx>1) screen4();     // Dx=2
            else screen3();         // Dx=1
 }                                  // экраны установок Sel=1,2
 else if(D<1) screenSet1();         // установка часов D=0
      else if(Sel>1) screenSet3();  // установка длительности таймера Sel=2
           else screenSet2();       // установка периодичности таймера Sel=1
}
```

Формируем экраны через функции экранов. Всего формируем 4 основных типа экранов и три экрана настройки времени и таймеров. Первый основной экран отображает время в часах и минутах всех четырех таймеров с указанием маркером работы таймера.

```
void screen1(){                                 //основной экран с 4 таймерами
 lcd.clear();                                   //очистка буфера экрана
 lcd.setCursor(0, 0); //установить курсор на первую строку и первый символ
 String tr = ":";    //определить отображение разделителя часов и минут
 if(b) tr = " ";   //определить мигание переменной - стирание отображения на текущий проход
 int rtcH1 = timer1 / 60;               //определить количество часов таймера 1
 int rtcH2 = timer2 / 60;               //определить количество часов таймера 2
 int rtcH3 = timer3 / 60;               //определить количество часов таймера 3
 int rtcH4 = timer4 / 60;               //определить количество часов таймера 4
 int rtcM1 = timer1 - rtcH1*60;    //определить количество минут таймера 1
 int rtcM2 = timer1 - rtcH1*60;    //определить количество минут таймера 2
 int rtcM3 = timer1 - rtcH1*60;    //определить количество минут таймера 3
```

```
  int rtcM4 = timer1 - rtcH1*60;   //определить количество минут таймера 4
  String rtcHs1 = String(rtcH1);    //определить отображение переменной часов таймера 1
  String rtcMs1 = String(rtcM1);       //определить отображение переменной минут таймера 1
  String rtcHs2 = String(rtcH2);          //определить отображение переменной часов таймера 2
  String rtcMs2 = String(rtcM2);          //определить отображение переменной минут таймера 2
  String rtcHs3 = String(rtcH3);          //определить отображение переменной часов таймера 3
  String rtcMs3 = String(rtcM3);          //определить отображение переменной минут таймера 3
  String rtcHs4 = String(rtcH4);          //определить отображение переменной часов таймера 4
  String rtcMs4 = String(rtcM4);          //определить отображение переменной минут таймера 4
  String tmNs1 = String(t1?"*":" ");                    //определить состояние таймера 1 0=false (OFF), 1=true (ON)
  String tmNs2 = String(t2?"*":" ");                    //определить состояние таймера 2 0=false (OFF), 1=true (ON)
  String tmNs3 = String(t3?"*":" ");                    //определить состояние таймера 3 0=false (OFF), 1=true (ON)
  String tmNs4 = String(t4?"*":" ");                    //определить состояние таймера 4 0=false (OFF), 1=true (ON)
  String Firststr = String(" " + rtcHs1 + ":" + rtcMs1 + " " + tmNs1 + " " + rtcHs3 + ":" + rtcMs3 + " " + tmNs3);                  //формирование первой строки
  lcd.println(Firststr);                                 //вывод на экран первой строки
  String Secondstr = String(" " + rtcHs2 + ":" + rtcMs2 + " " + tmNs2 + " " +
```

```
rtcHs4 + ":" + rtcMs4 + " " + tmNs4);          //формирование второй строки
  lcd.println(Secondstr);                      //вывод на экран второй строки
}

void screenW(){     //экран отображения текущего времени Watch
  lcd.clear();                                 //очистка буфера экрана
  lcd.setCursor(0, 0);   //установить курсор на первую строку и первый символ
  String tr = ":";   //определить отображение разделителя часов и минут
  if(b) tr = " ";     //определить мигание переменной - стирание отображения
на текущий проход
  String rtcHs = String(rtcH);     //определить отображение переменной часов
  String rtcMs = String(rtcM);     //определить отображение переменной минут
  String Firststr = String(" Now time " + rtcHs + tr + rtcMs);     //формирование
первой строки
  lcd.println(Firststr);                       //вывод на экран первой строки
  String Secondstr = String(" Uptime  " + String(rtcS) + " day"); //формирование
второй строки
  lcd.println(Secondstr);                      //вывод на экран второй строки
}
void screen2(){                                //экран таймера
  lcd.clear();                                 // очистка буфера экрана
  lcd.setCursor(0, 0);   //установить курсор на первую строку и первый символ
  String nT = String(D);     //номер таймера равный номеру дисплея 1..4
  String tmNs = "";  //определить отображение переменной часов таймера
  switch(D){        //определить состояние таймера 0=false (OFF), 1=true (ON)
    case 1:      tmNs = String(t1?"ON":"OFF"); break;
    case 2:      tmNs = String(t2?"ON":"OFF"); break;
    case 3:      tmNs = String(t3?"ON":"OFF"); break;
    case 4:      tmNs = String(t4?"ON":"OFF"); break;
```

```
  }
  String Firststr = String(" T" + nT + " " + tmNs + "m");          //формирование первой строки из номера и текущего времени таймера
  lcd.println(Firststr);                                           //вывод на экран первой строки
  String Secondstr = String(" Period  " + String(p1set) + " day"); //формирование второй строки с установленной величиной периодичности включения таймера
  lcd.println(Secondstr);                  //вывод на экран второй строки
}
void screen3(){      //экран прошедшего времени таймера
  lcd.clear();                                          //очистка буфера экрана
  lcd.setCursor(0, 0);   //установить курсор на первую строку и первый символ
  String tr = ":";       //определить отображение разделителя часов и минут
  int tmH = 0, tmM = 0, tmE = 0; //определить переменные часов и минут таймера
  switch(D){    //вычислить переменные часов и минут в зависимости от номера таймера
    case 1:      tmE = timer1; break;
    case 2:      tmE = timer2; break;
    case 3:      tmE = timer3; break;
    case 4:     tmE = timer4; break;
  }
  tmH = tmE / 60;
  tmM = tmE - tmH*60;
  String tmHs = String(tmH);     //определить отображение переменной часов таймера
  String tmMs = String(tmM);    //определить отображение переменной минут таймера
  if(b) tr = " ";        //определить мигание переменной - стирание отображения
```

```
на текущий проход
  String tmNs = "";     //определить отображение переменной часов таймера
  switch(D){        //определить состояние таймера 0=false (OFF), 1=true (ON)
    case 1:      tmNs = String(t1?"ON":"OFF"); break;
    case 2:      tmNs = String(t2?"ON":"OFF"); break;
    case 3:      tmNs = String(t3?"ON":"OFF"); break;
    case 4:      tmNs = String(t4?"ON":"OFF"); break;
  }
  String Firststr = String(" T" + String(D) + " " + tmNs);          //формирование
первой строки из номера таймера и его состояния
  lcd.println(Firststr);                                  //вывод на экран первой строки
  String Secondstr = String(" PASTtime    " + tmHs + tr + tmMs);
//формирование второй строки из времени прошедшего с момента
включения таймера
  lcd.println(Secondstr);                //вывод на экран второй строки
}

void screen4(){            //экран оставшегося времени таймера
  String tr = ":";      //определить отображение разделителя часов и минут
  int tmH = 0, tmM = 0, tmE = 0; //определить переменные часов и минут
таймера
  switch(D){      //вычислить переменные часов и минут в зависимости от
номера таймера
    case 1:      tmE = t1set - timer1; break;
    case 2:      tmE = t2set - timer2; break;
    case 3:      tmE = t3set - timer3; break;
    case 4:      tmE = t4set - timer4; break;
  }
  tmH = tmE / 60;
```

```
  tmM = tmE - tmH*60;
  String tmHs = String(tmH);              //определить отображение переменной часов таймера
  String tmMs = String(tmM);              //определить отображение переменной минут таймера
  if(b) tr = " ";                         //определить мигание переменной - стирание отображения на текущий проход
  String tmNs = "";       //определить отображение переменной часов таймера
  switch(D){   //определить состояние таймера 0=false (OFF), 1=true (ON)
    case 1:
      tmNs = String(t1?"ON":"OFF");
      break;
    case 2:
      tmNs = String(t2?"ON":"OFF");
      break;
    case 3:
      tmNs = String(t3?"ON":"OFF");
      break;
    case 4:
      tmNs = String(t4?"ON":"OFF");
      break;
  }
  String Firststr = String(" T" + String(D) + " " + tmNs);          //формирование первой строки из номера таймера и его состояния
  lcd.println(Firststr);                                  //вывод на экран первой строки
  String Secondstr = String(" PASTtime   " + tmHs + tr + tmMs);  //формирование второй строки из времени прошедшего с момента включения таймера
  lcd.println(Secondstr);                                 //вывод на экран второй строки
```

```
}
void screenSet1(){                    //экран установки текущего времени Watch
  lcd.clear();                                    //очистка буфера экрана
  lcd.setCursor(0, 0);    //установить курсор на первую строку и первый символ
  String Firststr = String(" Set now time");        //формирование первой строки
  lcd.println(Firststr);                          //вывод на экран первой строки
  String Secondstr, tmStr;                  //определить переменные второй строки
  int tmDH = 0, tmEH = 0, tmDM = 0, tmEM = 0;                    //определить
переменные цифр часов и минут
  tmDH = rtcH / 10;
  tmEH = rtcH - tmDH * 10;
  tmDM = rtcM / 10;
  tmEM = rtcM - tmDM * 10;
  //определить мигание переменной - стирание отображения на текущий
проход
  switch(D){
     case 1:
       if(b) tmStr = " "; else tmStr = String(tmDH);
       Secondstr = tmStr + String(tmEH) + ":" + String(tmDM) + String(tmEM);
       break;
     case 2:
       if(b) tmStr = " "; else tmStr = String(tmEH);
       Secondstr = String(tmDH) + tmStr + ":" + String(tmDM) + String(tmEM);
       break;
     case 3:
       if(b) tmStr = " "; else tmStr = String(tmDM);
       Secondstr = String(tmDH) + String(tmEH) + ":" + tmStr + String(tmEM);
       break;
     case 4:
```

```
      if(b) tmStr = " "; else tmStr = String(tmEM);
      Secondstr = String(tmDH) + String(tmEH) + ":" + String(tmDM) + tmStr;
      break;
   }
 lcd.println(Secondstr);                    //вывод на экран второй строки
 rtcH = tmDH * 10 + tmEH;    //сохранение измененных данных в переменную часов
 rtcM = tmDM * 10 + tmEM;  //сохранение измененных данных в переменную минут
 }

void screenSet2(){  //экран настройки длительности включения таймера
 lcd.clear();                              //очистка буфера экрана
 lcd.setCursor(0, 0);    //установить курсор на первую строку и первый символ
 String Firststr = String(" Set timer" + String(D));//формирование первой строки
 lcd.println(Firststr);                    //вывод на экран первой строки
 String Secondstr, tmStr;                  //определить переменные второй строки
 if(b) tmStr = " "; else tmStr = String(Dx);          //определить мигание переменной - стирание отображения на текущий проход
 Secondstr = " ON time " + tmStr + " m";              //формирование второй строки
 lcd.println(Secondstr);                    //вывод на экран второй строки
 //сохранение измененных данных в переменную соответствующего таймера
 switch(D){
   case 1:     timer1 = Dx; break;
   case 2:     timer2 = Dx; break;
   case 3:     timer3 = Dx; break;
   case 4:     timer4 = Dx; break;
```

```
  }
}

void screenSet3(){       //экран настройки периодичности включения таймера
  lcd.clear();                                    //очистка буфера экрана
  lcd.setCursor(0, 0);                            //установить курсор на первую строку и первый символ
  String Firststr = String(" Set timer" + String(D));          //формирование первой строки
  lcd.println(Firststr);                          //вывод на экран первой строки
  String Secondstr, tmStr;                 //определить переменные второй строки
  if(b) tmStr = " "; else tmStr = String(Dx);                  //определить мигание переменной - стирание отображения на текущий проход
  Secondstr = " Period " + tmStr + " day";         //формирование второй строки
  lcd.println(Secondstr);                         //вывод на экран второй строки
  //сохранение измененных данных в переменную соответствующего таймера
  switch(D){
    case 1:     period1 = Dx; break;
    case 2:     period2 = Dx; break;
    case 3:     period3 = Dx; break;
    case 4:     period4 = Dx; break;
  }
}
```

Управление нагрузками осуществляется через плату расширения на которую приходит цифровой сигнал о включении или выключении соответствующего выхода. Формируем функцию управления состоянием выходов МК и устанавливаем в соответствующее состояние булевые переменные таймеров.

```
void setOut(){                          //установка уровня выхода
 if(timer1>0) {digitalWrite(22, HIGH); t1=1;}        //если время таймера 1 больше 0 (включен) включить выход по 22 порту
   else {digitalWrite(22, LOW); t1=0;}               //если время таймера 1 равно 0 (выключен) выключить выход по 22 порту
 if(timer2>0) {digitalWrite(24, HIGH); t2=1;}        //если время таймера 2 больше 0 (включен) включить выход по 24 порту
   else {digitalWrite(24, LOW); t2=0;}               //если время таймера 2 равно 0 (выключен) выключить выход по 24 порту
 if(timer3>0) {digitalWrite(26, HIGH); t3=1;}        //если время таймера 3 больше 0 (включен) включить выход по 26 порту
   else {digitalWrite(26, LOW); t3=0;}               //если время таймера 3 равно 0 (выключен) выключить выход по 26 порту
 if(timer4>0) {digitalWrite(28, HIGH); t4=1;}        //если время таймера 4 больше 0 (включен) включить выход по 28 порту
   else {digitalWrite(28, LOW); t4=0;}               //если время таймера 4 равно 0 (выключен) выключить выход по 28 порту
}
```

Для предотвращения сброса настроек при случайном пропадании питания применены функции сохранения основных настроек таймеров и часов в энергонезависимую память. Но лучшим решением, конечно же, будет обеспечение бесперебойного питания платы МК с подключением алкалинового аккумулятора напряжением 7,4В с цепью подзарядки от сетевого блока питания.

```
void saveSet(){
 EEPROM.put(0, t1set); //Обновление дл-ти таймера 1 (int занимает 2 байта)
 EEPROM.put(3, t2set);          // Обновление длительности таймера 2
 EEPROM.put(5, t3set);          // Обновление длительности таймера 3
 EEPROM.put(7, t4set);          // Обновление длительности таймера 4
```

```
  EEPROM.put(9, p1set);              // Обновление периодичности таймера 1
  EEPROM.put(11, p2set);             // Обновление периодичности таймера 2
  EEPROM.put(13, p3set);             // Обновление периодичности таймера 3
  EEPROM.put(15, p4set);             // Обновление периодичности таймера 4
}
void readSet(){
  EEPROM.get(0, t1set);          // Чтение длительности таймера 1
  EEPROM.get(3, t2set);          // Чтение длительности таймера 2
  EEPROM.get(5, t3set);          // Чтение длительности таймера 3
  EEPROM.get(7, t4set);          // Чтение длительности таймера 4
  EEPROM.get(9, p1set);          // Чтение периодичности таймера 1
  EEPROM.get(11, p2set);         // Чтение периодичности таймера 2
  EEPROM.get(13, p3set);         // Чтение периодичности таймера 3
  EEPROM.get(15, p4set);         // Чтение периодичности таймера 4
}
```

Составленная программа откомпилирована в Arduino IDE ver.2.0, т.е. не содержит синтаксических ошибок.

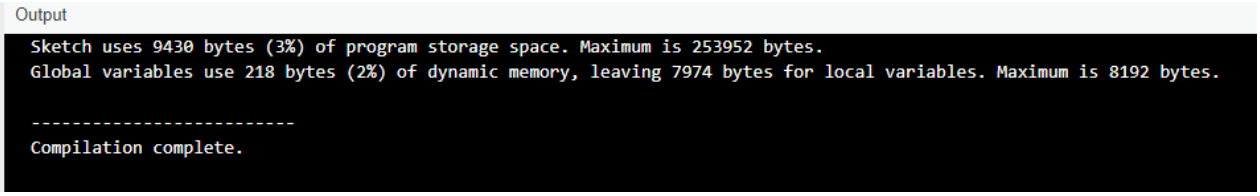

Рисунок 29 – Отчет о компиляции программы

Составленная программа загружена в симулятор PROTEUS и проверена в работе. Все функции четырехканального таймера работают в соответствии с заданием на проектирование.

3 Оценочная часть

По окончании разработки была оценена себестоимость разработанного устройства. Устройство собирается и упаковывается вручную монтажником за 12 часов рабочего времени. В таблицу 2 внесены все позиции из Перечня элементов устройства для расчета суммы затрат. Итоговая себестоимость устройства составляет 7015 рублей.

Таблица 2 – расчёт себестоимости устройства

№ п/п	Наименование	Кол-во, шт.	Стоимость, руб.
1	RobotDyn Arduino MEGA2560	1	1360
2	RobotDyn Keypad LCD Shield	1	333
3	RobotDyn AC Light Dimmer 4 channel	1	422
4	Coreset SM-PLG06A-09	1	210
5	Multicomp JR-101-1-FRSG-03	1	152
6	Розетка TDM РП10-3 панельная скрытая с крышкой IP44 16А 2P+PE 220В	4	59
7	Шлейф проводов 28AWG для Arduino с разъемами DuPont «Папа-мама»	1	160
8	Кабель питания с штекером DC5.5*2,1 для Arduino	1	14
9	Стойка крепления для платы М3х10мм	10	6
10	Стойка крепления для платы М3х20мм	4	7
11	Винт М3*6	30	3
12	Винт М4*20	18	5
13	Шайба d3	18	3
14	Гайка М3	18	4
15	Корпус прибора металлический GAINTA G2119 (200х150х55)	1	1040
16	Кабель питания	1	149
17	Инструкция по эксплуатации	1	56
18	Упаковка из картона (210х160х100)	1	99
19	Нормо-час монтажника	12	200
		Сумма	7015

Для сравнения приведенные в таблице 1 одноканальные электронные многопрограммные таймеры, с возможностью установки времени в течении суток, которые можно рассматривать в качестве ограниченных функциональных аналогов разработанного устройства, имеют стоимость около 1500 руб. (ТЭ-08А), и даже меньше 1000 руб. (ТЭ-16А и KG-316T). Контроллер «умного дома» ZONT C2000+ (рис.2), который выполняет множество функций помимо таймеров и имеет 12 каналов управления нагрузками, можно купить за 14810руб., что дороже разработанного устройства.

Можно предположить, что 4-х канальный функциональный аналог с возможностью повторения заданного интервала с заданной периодичностью будет стоить около 6000 рублей, что дешевле разработанного устройства.

Анализируя конструкцию разработанного 4-х канального электронного таймера можно с уверенностью сказать, что есть возможность удешевить устройство за счет разработки одноплатной конструкции, применения пластикового корпуса и массового производства. Примененная в данной работе конструкция, собранная из плат Arduino, позволяет быстро собрать прототип - минимально жизнеспособный продукт (minimum viable product, MVP) и оценить работоспособность устройства, детально проработав алгоритм работы микропроцессорного устройства.

Заключение

Данная бакалаврская работа выполнена в соответствии с заданием на разработку четырехканального электронного таймера. Особенностью данной работы является сборка многофункционального программируемого устройства на платформе быстрого прототипирования Arduino с программированием уникального алгоритма работы устройства на языке программирования C/C++ в интегрированной среде разработки Arduino IDE с последующей симуляцией работы устройства в программной среде Proteus 8.5 pro.

Поставленные задачи были выполнены в полном объеме:

- определены основные функции таймеров и найдены функциональные одноканальные аналоги электронного таймера с ограниченным периодом повторения (сутки, неделя);

- разработан четырехканальный электронный таймер в соответствии с заданием на проектирование;

- произведен расчет себестоимости и сравнено разработанное устройство с имеющимися на рынке функциональными аналогами.

Полученная себестоимость устройства оказалась выше средней цены четырех отдельно взятых электронных таймеров, массово выпускаемых промышленностью, но плюсом данной разработки является то, что она выполнена с возможностью установки большего интервала повторения таймера до 30 суток против 1-7 дней в массовых моделях электронных таймеров.

Разработанное устройство может быть использовано для малой автоматизации на производстве в малом и среднем бизнесе, например, для управления электрическими приборами теплицы (освещение, вентиляция, обогрев и полив растений). Путем изменения программы устройства можно

создать разную функциональность средств промышленной автоматизации, в том числе для управления устройствами умного дома.

Список используемой литературы

1. Аливерти, П. Электроника для начинающих / П. Аливерти [пер. с ит. И.В.Потрясиловой] – М.: Эксмо, 2018. – 368с.
2. Бейктал, Дж. Конструируем роботов на Arduino. Первые шаги [Электронный ресурс] / Дж. Бейктал [пер. с англ. О. А. Трефиловой]. — Эл. изд. — Электрон. текстовые дан. (1 файл pdf : 323 с.). — М.: Лаборатория знаний, 2016.
3. Белов, А.В. Arduino: от азов программирования до создания практических устройств / А.В. Белов, - СПб.: Наука и Техника, 2018. – 480с.
4. Бокселл, Дж. Изучаем Arduino. 65 проектов своими руками. / Дж. Бокселл, [пер. с англ.]. — СПб.: Питер, 2017. — 400 с.
5. Быстрый старт. Первые шаги по освоению Arduino. Набор конструктор начинающего изобретателя MaxKit.Ru / Creative commons, San Francisco, USA, 2020 – 80с.
6. Геддес, М. 25 крутых проектов с Arduino / Марк Геддес [пер. с англ. М.А. Райтмана]. — Москва: Эксмо, 2019. — 272 с.
7. Жимарши, Ф. Сборка и программирование мобильных роботов в домашних условиях / Ф. Жимарши [перев. с фр. М.А.Комаров] – М.: ИТ Пресс, 2007.- 288с.
8. Колмаков, С. Дело в программировании. Пособие / С. Колмаков, - Курган, 2017. – 101с.
9. Макаров, С.Л. Arduino Uno и Raspberry Pi 3: от схемотехники к интернету вещей. - М.: ДМКПресс, 2018. - 204 с.
10. Мамичев, Д. Программирование на Ардуино. От простого к сложному / Д.Мамичев, — М.: СОЛОН-Пресс, - 2018. - 244 с.
11. Момот, М.В. Мобильные роботы на базе Arduino. / М.В. Момот, — СПб.: БХВ-Петербург, 2017. — 288 с.
12. Монк, С. Электроника. Сборник рецептов: готовые решения на базе Arduino и Raspberry Pi / С. Монк [Пер.с англ.] - СПб.: ООО "Диалектика': 2019. - 480с.
13. Петин, В.А. Arduino и Raspberry Pi в проектах Internet of Things / В.А.Петин

– СПб. БХВ-Петербург, 2019. – 432с.

14. Петин, В.А. Практическая энциклопедия Arduino / В.А. Петин, А.А. Биняклвский, - М.: ДМК Пресс, 2017 – 152с.
15. Петин, В.А. Проекты с использованием контроллера Arduino / В.А. Петин, - СПб.: БХВ-Петербург, 2019. -496с.
16. Платт, Ч. Электроника для начинающих / Ч. Платт, [Перев. с англ. М.Райтмана] – СПб.: БХВ-Петербург, 2017. – 416с.
17. Редкар М. Создание роботов вместе с Arduino / М. Редкар – 2020. 24с.
18. Хуанг, Б. Arduino для изобретателей на 10 занимательных проектах / Б.Хуанг, Д.Ранберг [пер. с англ.]. — СПб.: БХВ-Петербург, 2019. — 288 с.
19. Юревич, Е.И. Основы робототехники: учебное пособие / Е.И. Юревич, - СПб.: БХВ-Петербург, 2010. – 368с.
20. Яценков, В.С. От Arduino до Omega: платформы для мейкеров шаг за шагом / В.С. Яценков, - СПб.: БХВ-Петербург, 2018 – 304с.
21. Evans, B.W. Arduino Programming Notebook / B.W.Evans, - Creative Commons: San Francisco, 2007. – 40p.
22. Margolis, M. Arduino Cookbook, Second Edition / Michael Margolis - O'Reilly Media, Inc., 2012. – 724p.
23. Schmidt, M. Arduino: A Quick-Start Guide / M. Schmidt - The Pragmatic Bookshelf – Dallas, USA: 2011. – 284p.
24. Schwartz M. Интернет вещей с Arduino Yún / M. Schwartz – 100с.
25. Wiley, J. Exploring Arduino: Tools and Techniques for Engineering Wizardry / J. Wiley, - Indianapolis, Canada, 2020 – 491p.