Направление подготовки: 140400 Электроэнергетика и электротехника

Профиль подготовки: Электрический транспорт

Квалификация (степень) выпускника: магистр

Форма обучения: очная

Дисциплина: "Системы и устройства автоматического управления электроподвижным составом"

Комаров В.Г. Лекция 4 10.03.2026 г.

Тема: Этапы разработки систем управления ЭПС

1. Структурирование по уровням управления

Уровни: глобальный, диспетчерский, линейный, поездной, вагонный, агрегатный, узловой, локальный, элементный.

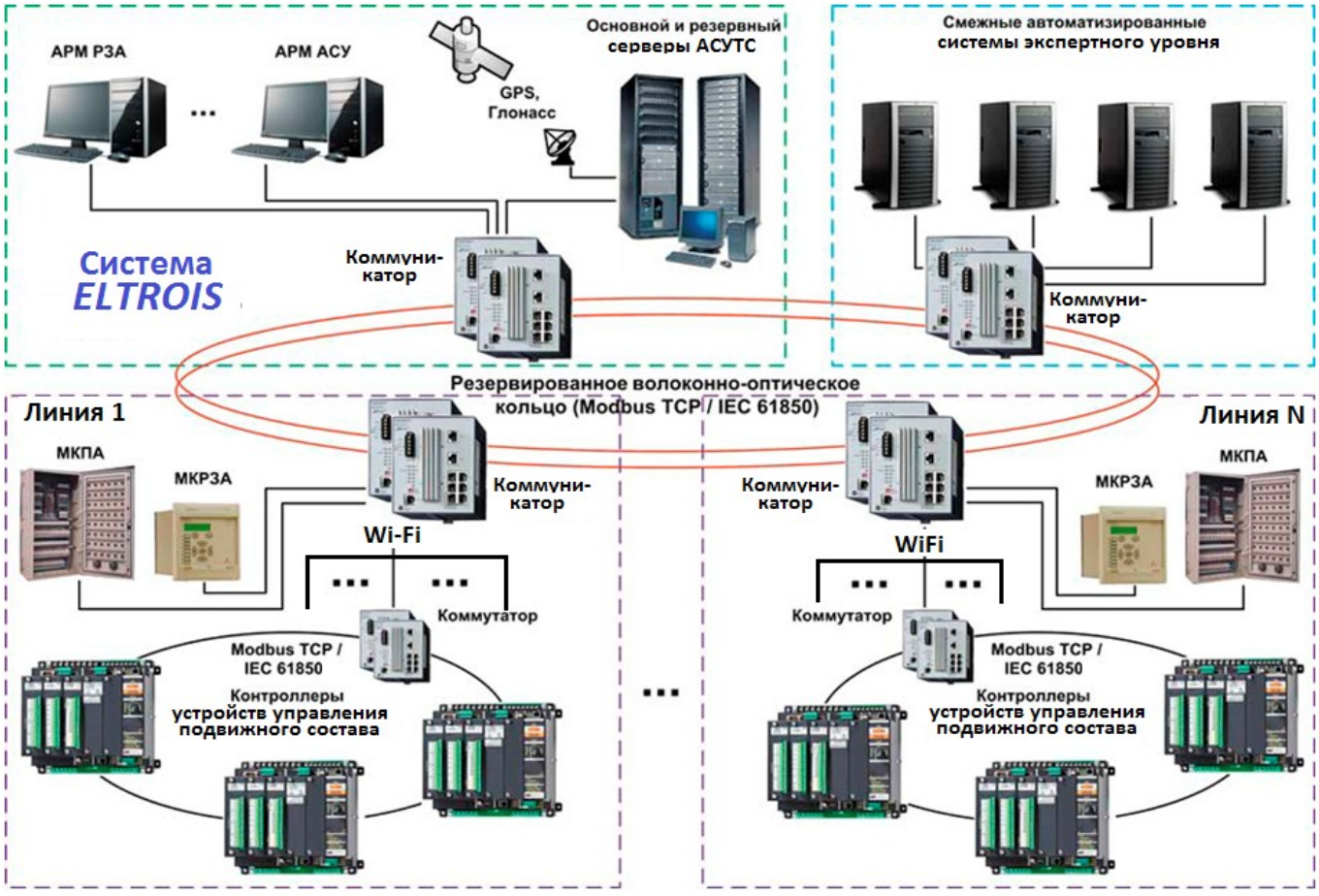


Рис. 1. Распределённая система диспетчерского управления ELTROis интегрированная с ГИС

Рис. 2. Диспетчерский терминал системы управления

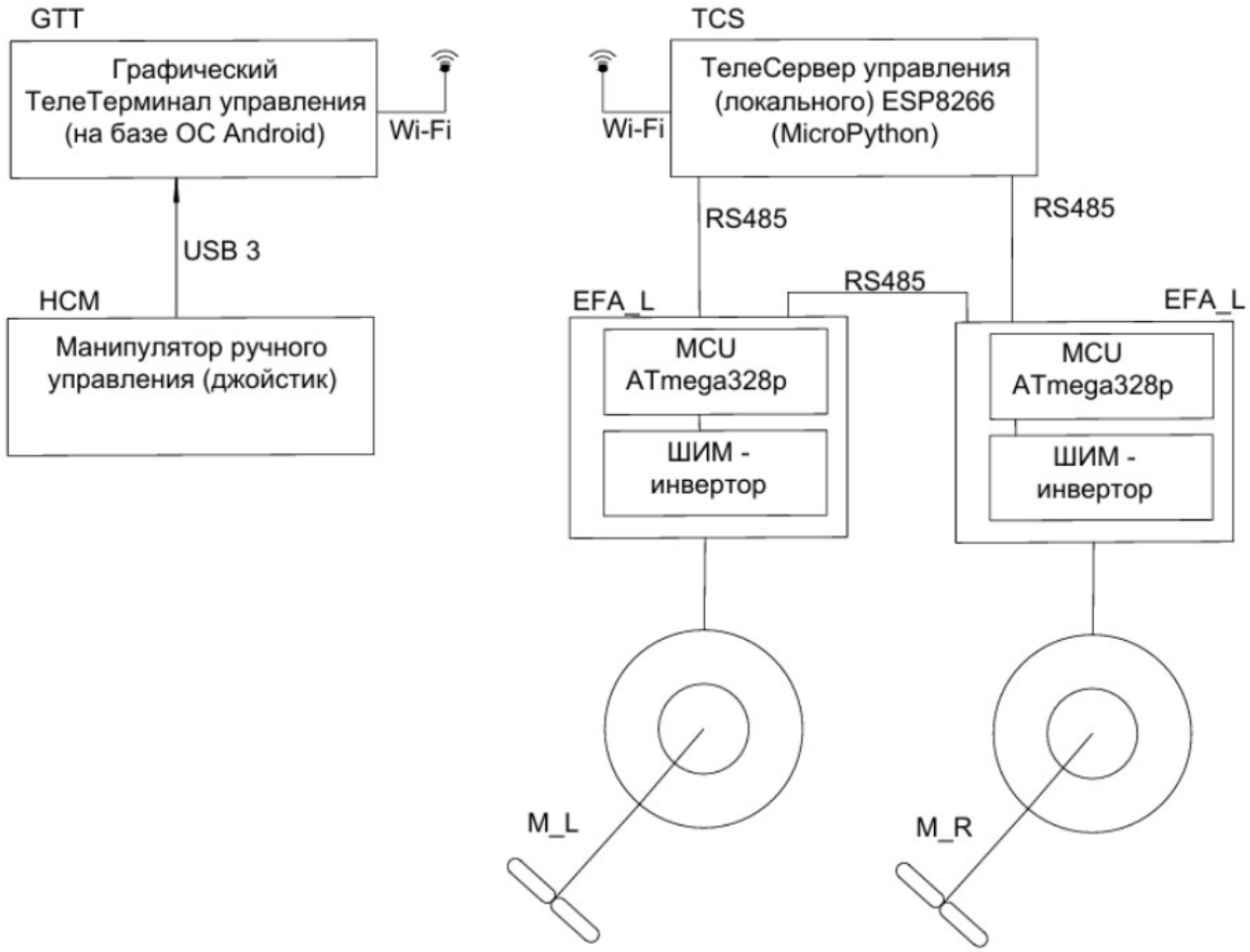


Рис. 3. Структурная схема унифицированного управления ЭПС

HCM – Hand Control Manipulator, GTT – Graphical TeleTerminal, TCS - Tele Control Server, EFA – Electric Force Aggregate, MCU - Micro Control Unit.

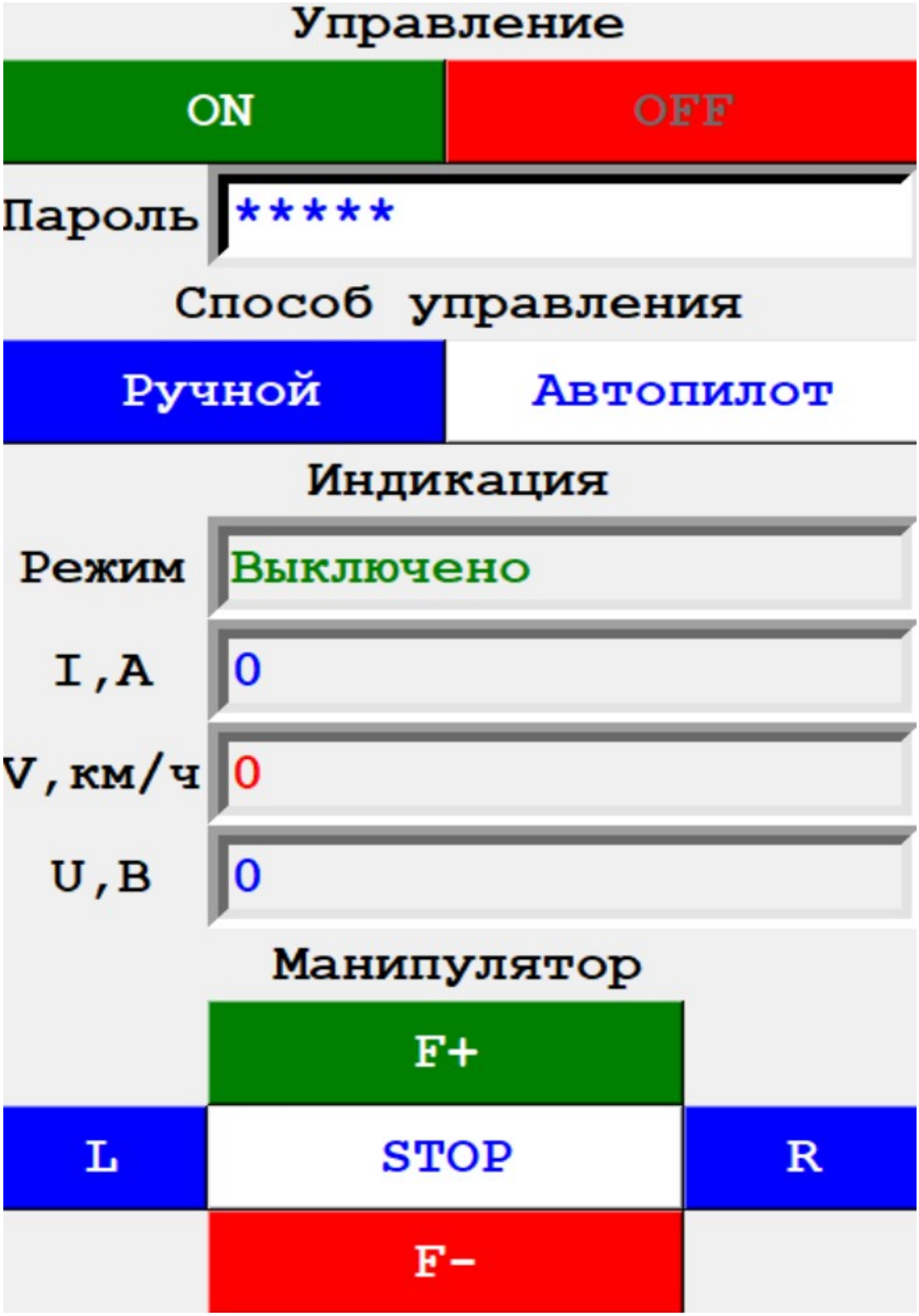
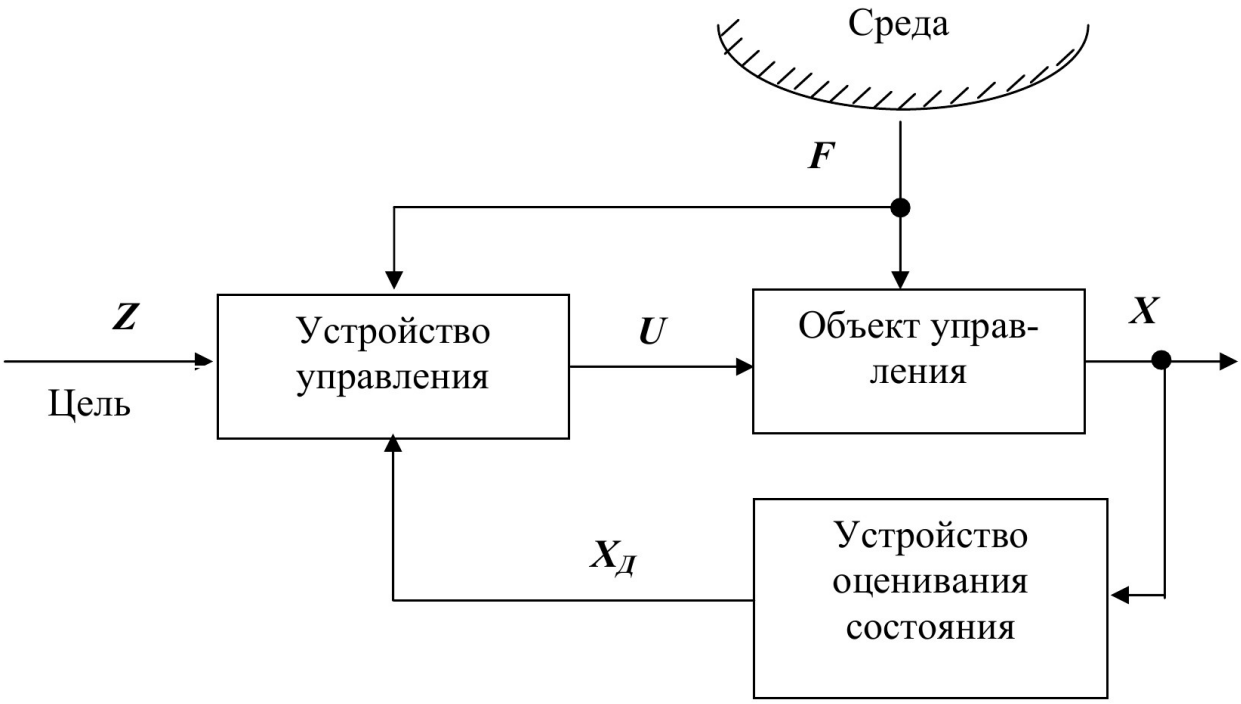


Рис. 4. Вид универсального графического терминала управления

Рис. 5. Вид универсального ручного манипулятора управления

2. Структурирование по принципам управления

Управление.



Обобщенная функциональная схема системы управления

Логическое и алгоритмическое управление представляет из себя событийное управление по качественным показателям.

Программным управлением принято называть управление, реализующее изменение координат состояния объекта или процесса в соответствии с заранее заданным алгоритмом. Одной из разновидностей программного управления является программно-логическое управление (ПЛ-управление). Требуемый алгоритм управления реализуется в зависимости от состояния и событий в объекте управления, которые задаются в форме ряда условий, определяющих последовательность выполнения операций управления.

Синтез алгоритма системы ПЛ-управления производится в несколько этапов.

1 этап. Составляется таблица состояний, в которых может находиться объект и формализованное их описание в виде соответствующих логических условий.

2 этап. На основе таблицы состояний определяются возможные переходы из одного состояния в другое. Составляется направленный граф состояний и переходов, а также логические уравнения, описывающие условия реализации переходов.

3 этап. Определяется необходимое информационное обеспечение работы системы, а также требуемые по технологии временные задержки и блокировки.

4 этап. На основании полученных результатов составляется формализованное описание алгоритма реализации ПЛ-управления средствами вычислительной техники.

В качестве примера ПЛ-управления ЭПС создадим модель логического автомата (конечного автомата) с блоком комбинационной логики с заданной таблицей истинности. Блок использует три входных логических сигнала:
X0 – сигнал T (TRACTION, ТЯГА), формируемый командоаппаратом водителя;
X1 – сигнал B (BRACKE, ТОРМОЗ), также формируемый командоаппаратом водителя;
X2 – сигнал V (VELOCITY, СКОРОСТЬ), формируемый датчиком минимальной скорости (<0,5 м/с), при которой включается стояночный тормоз (при v<0,5 м/с X2=0).

В зависимости от этих сигналов логическим устройством формируются четыре режима работы электрического транспортного средства, которые могут рассматриваться, как состояния логического автомата и характеризоваться кодовой переменной Y:
S - СТОП – режим остановки подвижного состава, в котором отключены тяговые электродвигатели и включён механический стояночный тормоз. Этот режим должен наступать при скорости меньше минимальной во всех случаях, кроме задания режима ТЯГА X0=1.
T – TRAC (ТЯГА) – режим с включёнными тяговыми двигателями в режим тяги (MOTORING).
B – BRAK (ТОРМОЗ) – режим с включёнными тяговыми двигателями в режим электричекого торможения (GENERATING).
F – FREE (ВЫБЕГ) – режим с выключенными тяговыми двигателями со свободным движением (FREE).

На выходе логического устройства формируютя коды режима (состояния) Y (два разряда Y0,Y1) и коды индикации Z (четыре разряда Z0-СТОП, Z1-ТЯГА , Z2-ТОРМ , Z3-ВЫБЕГ). Коды Y выдаются в магистраль управления поездом, а Z на индикатор режима движения. Для компактности и наглядности таблицы и графа переходов все двоичные коды можно заменить символами любого алфавита. В качестве символов можно, например, использовать стандартный американский код обмена информацией ASCII (American Standard Code Information Interchange). Однако более целесообразно в качестве символов использовать соответствующие им целые числа в десятичном или шестнадцатиричном представлении, что мы и будем использовать.

Каждому из режимов поставим в соответствие одноимённое устойчивое состояние автомата в соответствии с табл. 1.

Таблица 1. Таблица режимов-состояний Y логического автомата

Наименование режима (состояния)	Обозначение состояния	Код состояния		
		uint8	Y1	Y0
STOP (СТОП)	S	0	0	0
TRAC (ТЯГА)	T	1	0	1
BRAK (ТОРМОЗ)	B	2	1	0
FREE (ВЫБЕГ)	F	3	1	1

Далее из содержательного описания алгоритма работы логического автомата при управлении транспортным средством можно построить граф логического автомата Мура (рис. 6). В вершинах графа указаны состояния в соответствии со значениями кода Y, на исходящих началах дуг указаны коды X входных сигналов, вызывающие переход в следующее соответствующее состояние. При переходах из одного состояния в другое часть входных сигналов X являются избыточными и их в принципе можно исключить.

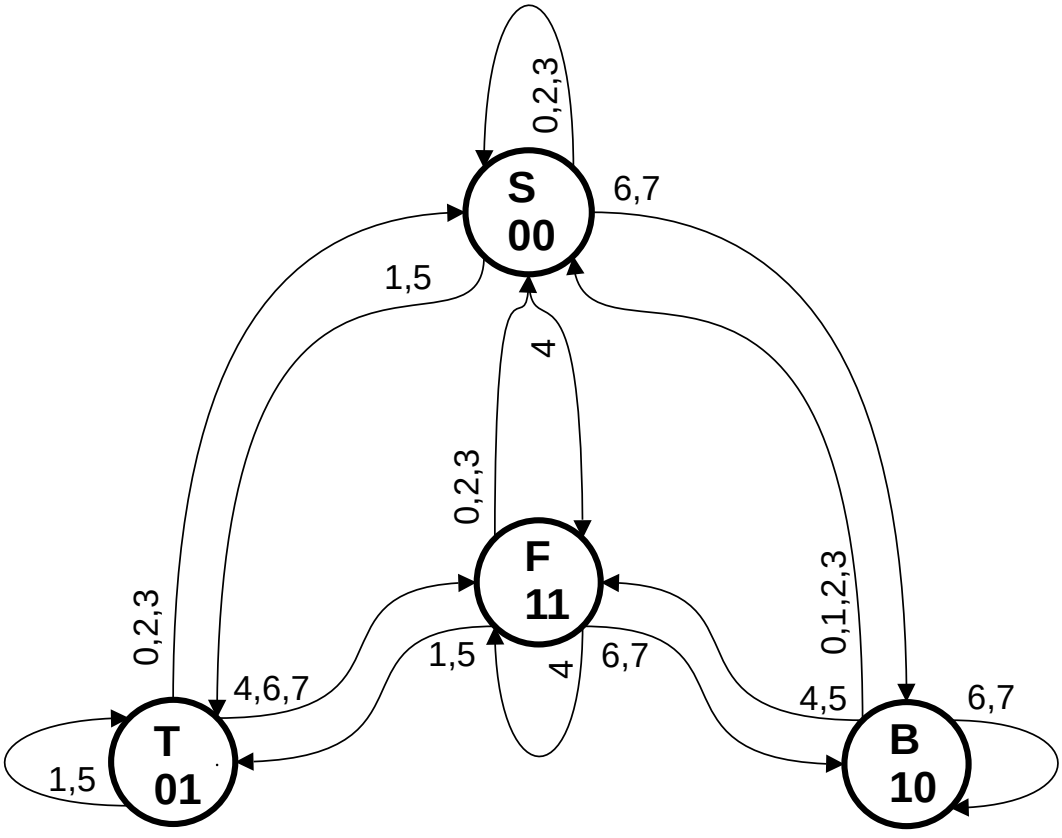


Рис. 6.

Так при переходе из состояния S в состояние T входной сигнал X2 является избыточным, т.к. указанный переход происходит независимо от его значения (и при X2=0 и при X2=1). Поэтому его при этом переходе можно не учитывать. Аналогично при переходе из состояния S в состояние B можно не учитывать входной сигнал X0 и т.д.

Для моделирования логического устройства воспользуемся оператором логического выбора (ветвления) if...else if.. языка Си. Общий синтаксис конструкции имеет вид:

```
 if (y == 0) {
     x = 2;
 }
 else if (y == 1) {
     x = 4;
 }
 else if (y == 2) {
     x = 8;
 . . .
 else
     return 1;
 }
```

При первом же совпадении выполняются операторы этой ветви. Если ни одного совпадения не оказалось, то выполняются операторы ветви else. Это позволяет легко написать программу логического автомата прямо по графу логического автомата Мура.

Ниже приведён пример написания части полной программы логического автомата, для логических переходов из состояния S в состония T и B.

```
//Программа логического управления режимами работы транспортного средства
"DriveModeLogic - dml"

#include <stdio.h> // Описания стандартного ввода-вывода

int main() {
    int x=0, y=0, z=0;
    printf("Введите код Y исходного режима работы:\n");
    scanf("%d", &y);
    printf("y = %d\n", y);
    printf("Введите задающий код X:\n");
    scanf("%d", &x);
    printf("x = %d\n", x);
        if(y==0) {
            if(x==0) {
                  y=0,z=1,printf("СТОП\n");
                                }
            else if(x==2) {
                  y=0,z=1,printf("СТОП\n");
                                }
            else if(x==3) {
                  y=0,z=1,printf("СТОП\n");
                                }
            else if(x==4) {
                  y=3,z=8,printf("ВЫБЕГ\n");
                                }
            else if(x==1) {
                  y=1,z=2,printf("ТЯГА\n");
                                }
            else if(x==5) {
                  y=1,z=2,printf("ТЯГА\n");
                                }
            else if(x==6) {
                  y=2,z=4,printf("ТОРМОЗ\n");
```

```
                        }
        else if(x==7) {
              y=2,z=4,printf("ТОРМОЗ\n");
                        }
        else {
              printf("ОШИБКА_ЗАДАНИЯ\n");
        return 1;
              }
        }
    else {
        printf("ОШИБКА_РЕЖИМА\n");
        return 2;
        }
      printf("ИНДИКАТОР_РЕЖИМА = %d\n", z);
}
```

Сохранив программу в текущем каталоге под именем dml.cpp, её можно скомпилировать, используя компилятор g++ в файл под основным именем, используя ключ -o в команде g++

```
user1@eltron-host1:~$ g++ -o DriveModeLogik dml.cpp
```

Запустив на выполнение функцию DriveModeLogic она будет выполнять логическое управление с разными кодами входных переменных X автомата при разных кодах Y режимов работы транспортного средства.

Регулирование.

Регулирование представляет из себя управление по количественным показателем с целью стабилизации параметров или обеспечения заданных законов изменения этих параметров.

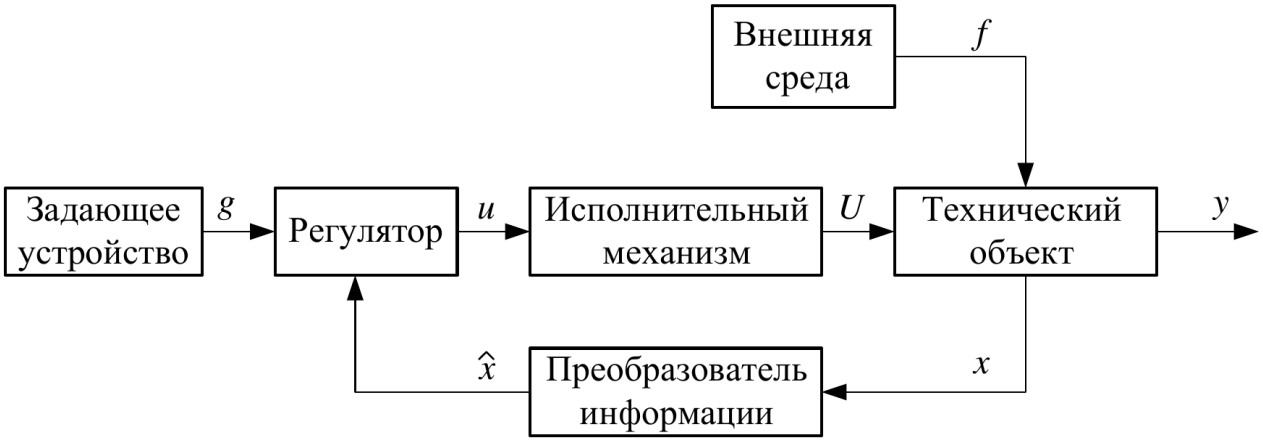


Рис. 7. Обобщенная функциональная схема системы регулирования

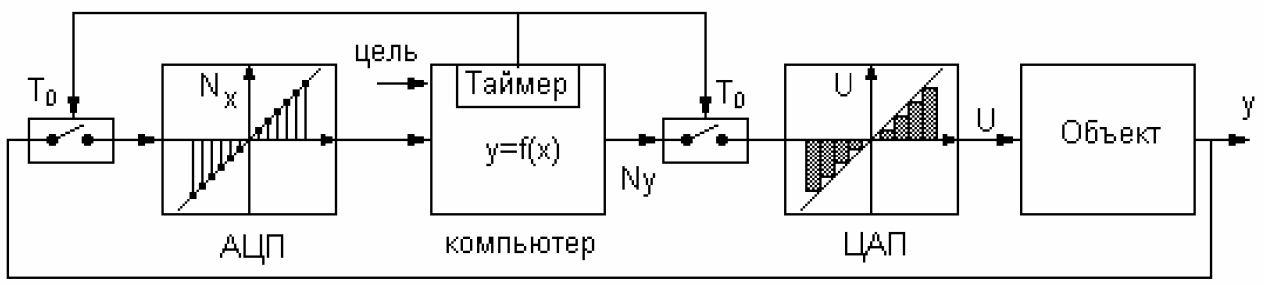


Рис. 8. Структурная схема системы прямого цифрового (компьютерного) регулирования объектом непрерывного типа.

3. Выбор интерфейсов межуровнего и межэлементного взаимодействия

Способ передачи данных, способ классификации параметров, способ адресации и т.п.

Интерфейс - граница между двумя функциональными объектами, требования к которой определяются стандартом. Кроме того интерфейс это совокупность средств, методов и правил взаимодействия: управления, контроля и т. д. между элементами системы. Иерархическая структура интерфейсов отображается сетевой моделью.

Согласно стандарту ISO 24765 сетевая модель OSI это сетевая модель стека сетевых протоколов OSI/ISO, посредством которой разные сетевые устройства могут взаимодействовать друг с другом. Модель состоит из семи уровней, представленных на рисунке 9.

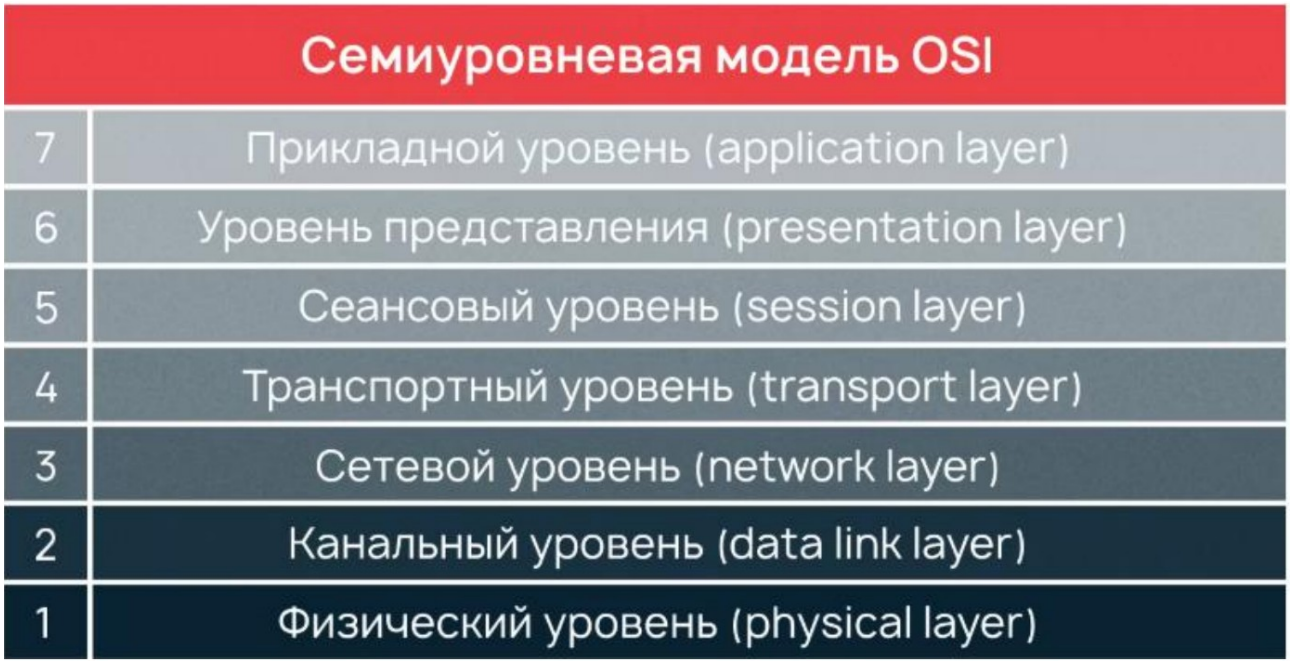


Рис. 9. Модель OSI

Первый (физический) уровень отвечает за обмен физическими сигналами между физическими устройствами, «железом». Устройства физического уровня оперируют битами: набором нулей и единиц, который передаются по кабелям или дистанционно. Второй (канальный) уровень превращает биты в кадры («фреймы») с адресом отправителя и получателя, после чего отправляет их по сети. На этом уровне используются коммутаторы, передающие по MAC-адресам сформированные кадры. На третьем (сетевом) уровне используются маршрутизаторы (роутеры), которые получают MAC-адрес и строят маршрут от между устройствами. Задачей четвертого (транспортного) уровня является транспортировка пакетов. Пятый (сеансовый) уровень отвечает за поддержку сессии связи. Этот уровень управляет взаимодействием между приложениями, обменом информации и завершением сеанса. Уровень шесть (представления) отвечает за преобразование протоколов и кодирование/декодирование данных. Седьмой (прикладной) уровень – это графический интерфейс, чья функция используя свои протоколы, донести до пользователя данные в понятном для него формате.

В настоящее время в основном используются последовательные интерфейсы. В последовательных интерфейсах используются одна линия связи, что означает прием/передачу только одного бита, поэтому используются последовательные коды, которые передают биты один за другим последовательно во времени. Эти коды группируются в пакеты различного назначения. Последовательная магистраль проще и дешевле, чем параллельная.

Modbus - открытый коммуникационный протокол для обмена данными между промышленными устройствами. Modbus использует архитектуру Ведущий-Ведомый. Согласно этому в сети выделяется ведущее устройство, которое отправляет запросы на ведомые устройства с целью чтения или записи их параметров. К основным достоинствам Modbus можно отнести простоту в реализации и использовании, возможность работы на разных скоростях передачи данных, что делает его гибким для различных задач. Для протокола Modbus не нужны специальные интерфейсные контроллеры, требующие для своей реализации заказных микросхем. Он обладает простотой программной реализации, а также имеет совместимость с большим количеством оборудования. Также Modbus позволяет унифицировать команды обмена благодаря стандартизации адресов регистров.
Протокол Modbus имеет несколько режимов передачи: RTU (remote terminal unit) и ASCII. Режим RTU в протоколе Modbus должен присутствовать обязательно, а режим ASCII является опциональным. Стандарт предусматривает применение физического интерфейса RS-485, RS-422 или RS-232. Наиболее распространенным является 2-проводной интерфейс RS-485. В протоколе Modbus RTU сообщение начинает восприниматься как новое после паузы длительностью не менее времени передачи 3,5 байт, величина паузы по времени зависит от скорости передачи.

Таблица 2 - Модель OSI для Modbus

Номер уровня	Название уровня	Реализация
7	Прикладной	*Modbus application protocol*
6	Уровень представления	-
5	Сеансовый	-
4	Транспортный	-
3	Сетевой	-
2	Канальный	Протокол «Ведущий–Ведомый». Режимы *RTU* и *ASCII*
1	Физический	*RS*-485, *RS*-422 или *RS-232*

Формат кадра показан на рисунке 10. Поле адреса всегда содержит только адрес ведомого устройства. Благодаря этому ведущее устройство знает, от какого модуля пришёл ответ.

Рис. 10. Формат кадра протокола Modbus RTU

Каждый кадр Modbus RTU состоит из 4 полей: адрес, код функции, данные, контрольная сумма. По коду функции устройство понимает, какой тип данных с него спрашивают. Поле данных может быть от 0 до 255 байтов. Контрольная сумма содержит CRC длиной 2 байта.

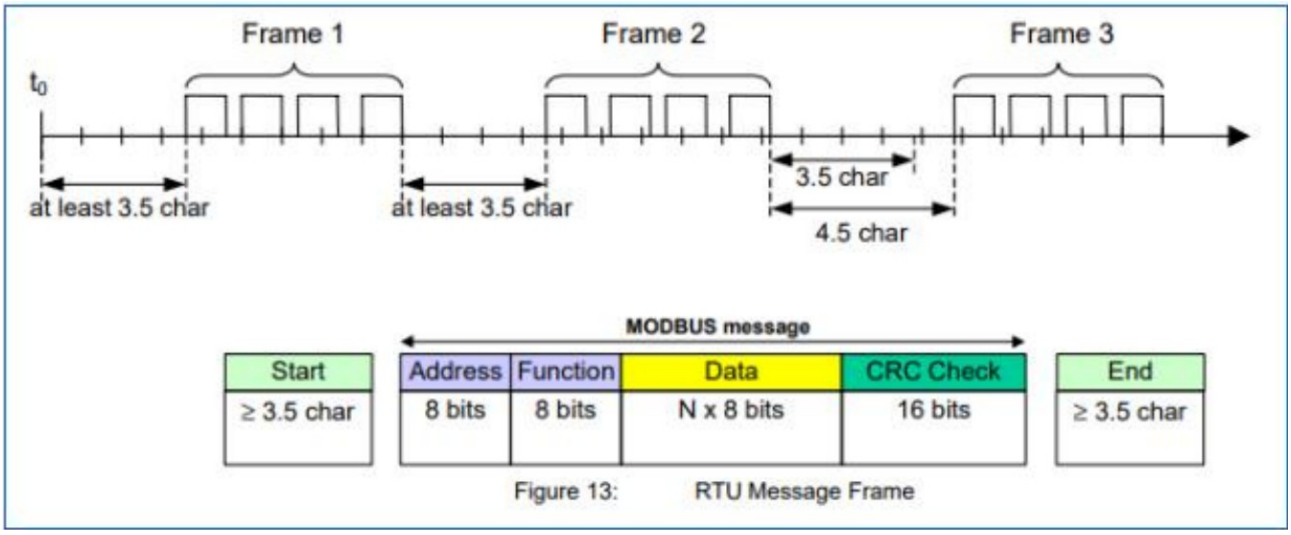


Рис. 11. Формат потока данных протокола Modbus RTU

Для отправки кадра нужно задать адрес, номер функции, данные и контрольную сумму. Список функций приведен на рисунке 12.

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

Рис.12. Список функций протокола Modbus RTU

На физическом уровне для протокола Modbus можно использовать практически любой как проводной, так и беспроводной интерфейс, например, RS-485, Ethernet, WiFi и т.д. Интерфейс RS-485 (EIA/TIA-485) - стандарт последовательной асинхронной передачи данных. Является полудуплексным интерфейсом, т.е. данные не могут одновременно передаваться в обоих направлениях, только в одном. В основе интерфейса RS-485 лежит принцип дифференциальной передачи данных, когда сигнал передается по двум проводам. Причем, если по одному проводу (условно A) идет оригинальный сигнал, то по-другому (условно B) - его инверсная копия. Таким образом, между проводами всегда присутствует разность потенциалов: при "1" она положительна, при "0" - отрицательна. Принцип приведен на рисунке 13. Этой разностью потенциалов и передается значение сигнала. Дифференциальной способ передачи

обеспечивает высокую устойчивость к синфазной помехе, которая одинаково действует на оба провода линии.

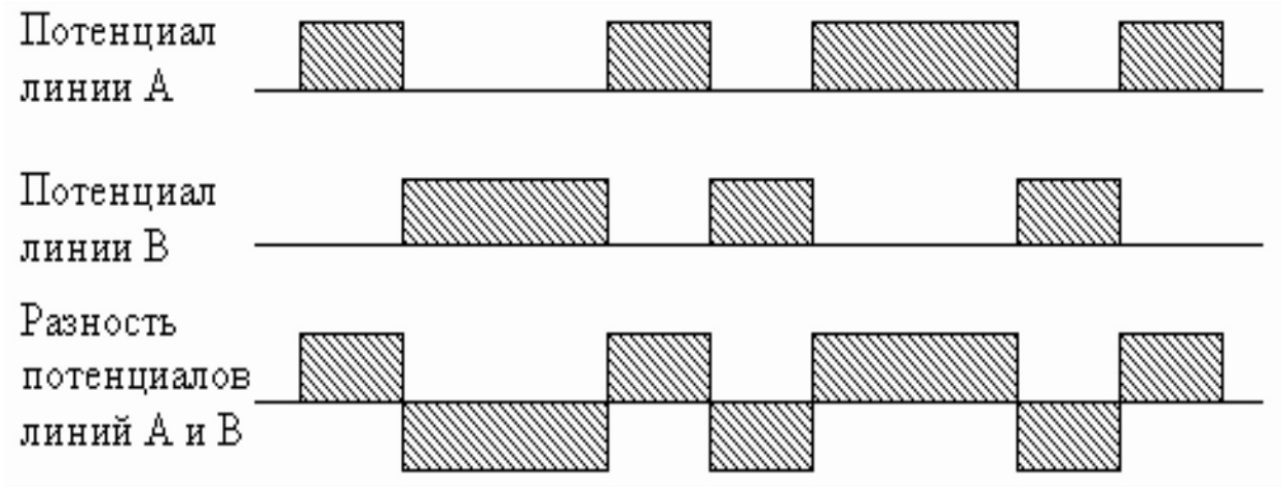


Рис. 13. Принцип дифференциальной передачи данных

Для передачи данных на подвижной состав используется беспроводная связь. На данный момент управление через беспроводную сеть Wi-Fi является одним из самых перспективных направлений. Данная технология создаёт локальную сеть в которой устройства могут свободно обмениваться данными, не выходя в глобальный интернет, что обеспечивает безопасность соединение между приёмником и передатчиком. Сети Wi-Fi имеют два основных частотных диапазона: 2.4 ГГц и 5 ГГц. Частота 5 ГГц используется при большом количестве связываемых устройств, чтобы не перекрывать сигналы друг друга. Современные микроконтроллеры имеют на кристалле устройства, поддерживающие WiFi, например микроконтроллеры ESP-32 (рис.14) , основанные на архитектуре RISC-V.

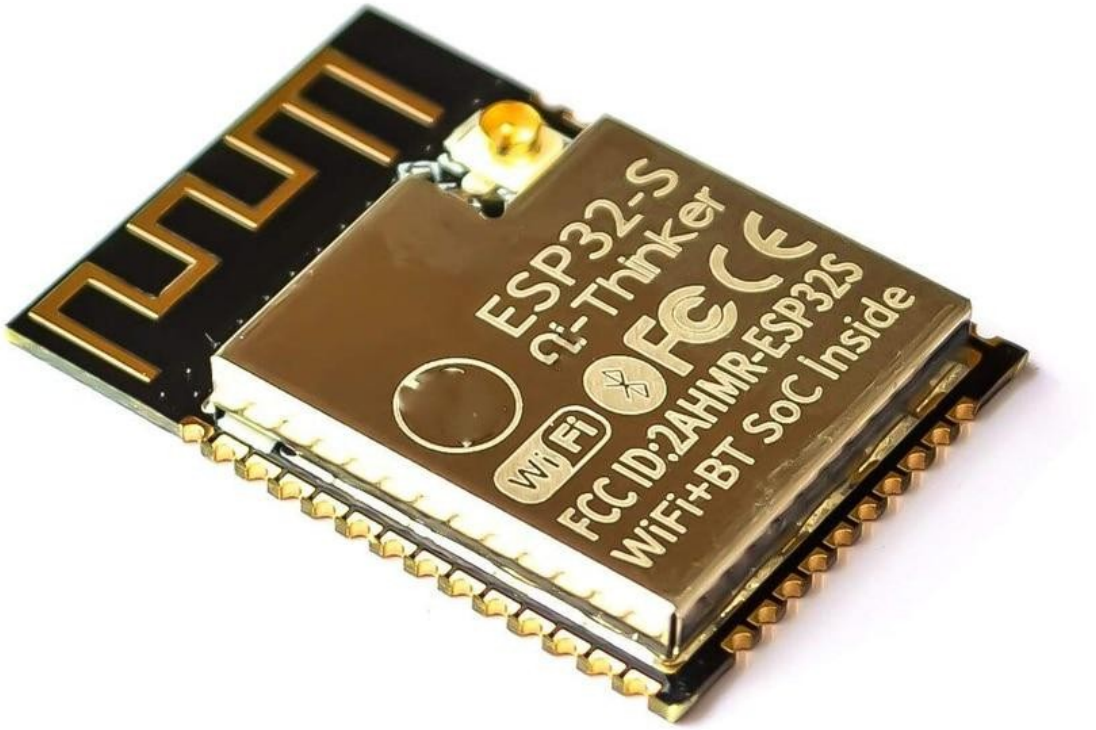


Рис. 14. Микроконтроллер архитектуры RISC-V.

4. Анализ функциональных связей и построение функциональной схемы управления

Определение и кодирование списка величин, характеризующих управляющие воздействия, внешние воздействия и контролируемые параметры.

5. Анализ объекта управления и создание его модели

6. Анализ устойчивости и качества процессов регулирования

7. Конструктивная реализация блоков и устройств СУ