Московский Политехнический Университет

Курсовая работа по дисциплине «Информационные технологии»

Тема: «Прием и обработка данных от удаленных устройств при использовании протоколов стандартов IEEE 802.1 b/g/n (WiFi) на базе Arduino Leonardo и WiFi-модуля ESP8266. Реализация сервера для POSIX-систем на языке C.»

Выполнил: студент гр. 143-311 Дятлов А.А.

Проверил: преподаватель Идиатуллов Т.Т.

Вводная часть.

В данной работе основной акцент сделан на программную часть. В частности разобрана реализация приема-передачи данных с помощью RS-232 и механизма socket в ОС Linux на языке C согласно стандарту POSIX. Созданная программа обладает основным функционалом по приему и обработке данных, реализованном с помощью системных вызовов и библиотек GNU GCC. В ходе написания программы использовался следующий инструментарий: редактор `Vim`, компилятор `gcc-4.8`, утилита сборки `make`, система контроля версий `git`. Возможно дальнейшее совершенствование и расширение серверной части в плане работы с СУБД, графическими клиентами (реализованными с помощью библиотеки Qt), поддержки кроссплатформенности. Исходный код модулей программы представлен на ресурсе github.com ссылка: https://github.com/4lovi4/duino_term.

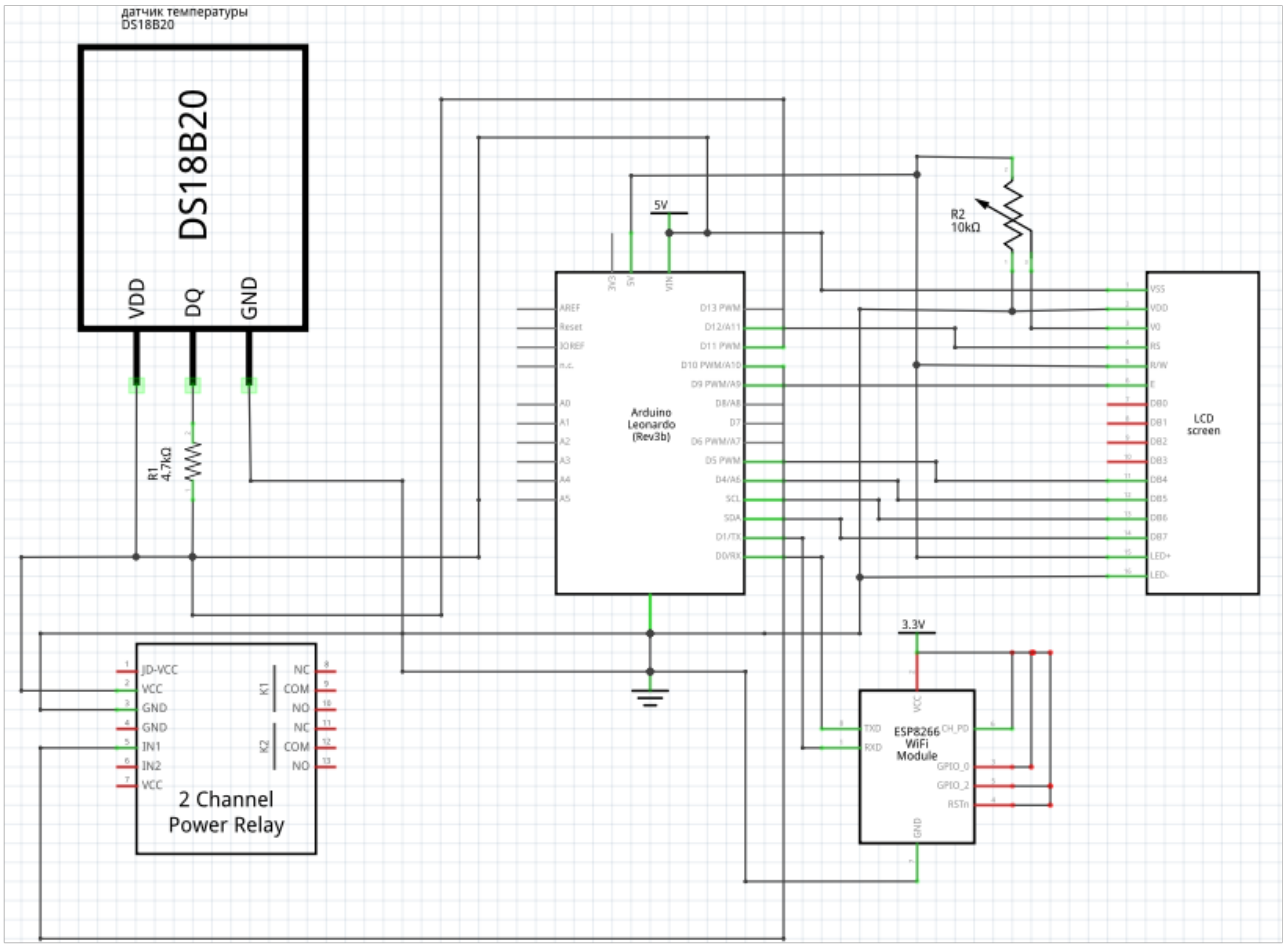
Аппаратная часть на базе отладочной платы Arduino Leonardo реализует контроль и передачу текущей температуры в удаленном помещении, получаемой от датчика DS18B20 и передаваемой по беспроводной сети посредством компактного модуля ESP8266 на серверный ПК, где запущена программа duino_term, мониторящая указанный при запуске последовательный порт и UDP-сокет 7143, переданное значение записывается в создаваемый при открытии программы в текущем каталоге лог-файл. При превышении заданного в аргументах программы порога температуры отправляется сигнал о прерывании реле и отключении электрического нагревателя.

Состав и характеристики аппаратной части системы.

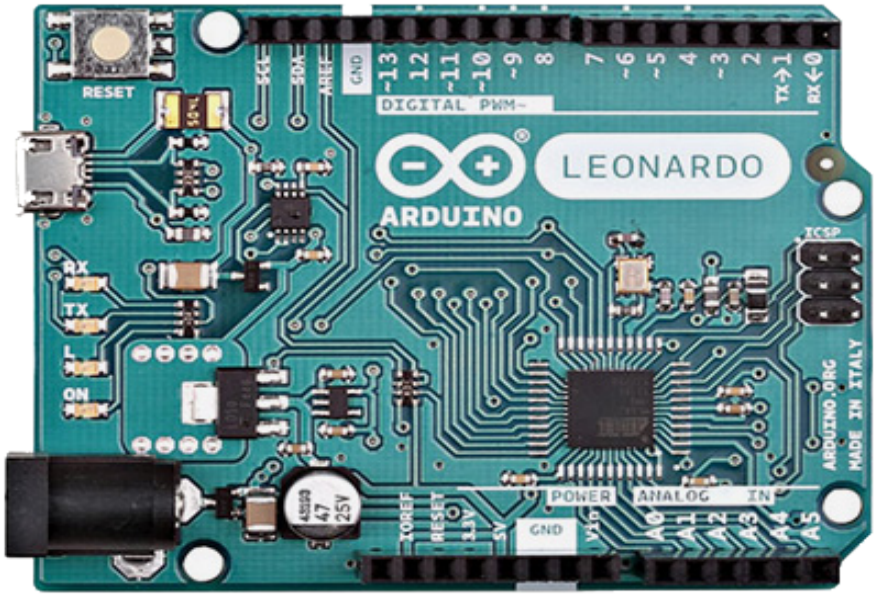
Система состоит из следующих компонентов:

- Отладочная плата Arduino Leonardo
- Компактный WiFi-модуль ESP8266
- Модуль стабилизатор питания 3,3 В на базе микросхемы AMS1127
- LCD Winstar WH-1602B на базе контроллера дисплея HD44780
- Блок реле SRD-05VDC-SL-C ~250В 10А
- Датчик температуры DS18B20

Принципиальная схема системы, созданная в среде программы Fritzing.



Для реализации проекта была выбрана в качестве ядра системы плата Arduino Leonardo, исходя из доступности устройства и простоты реализации программы, а также наличия встроенной поддержки USB, что упрощает процесс отладки.



Характеристики и особенности платы. В основе микроконтроллер ATmega32u4. Она имеет 20 цифровых входных/выходных выводов (7 из которых могут использоваться в качестве ШИМ выходов и 16 в качестве аналоговых входов), кварцевый резонатор 16 МГц, подключение USB, разъем питания, разъем ICSP и кнопку перезагрузки. Она содержит всё необходимое для работы с микроконтроллером; для того, чтобы начать работу с ней, просто подключите ее к компьютеру с помощью USB кабеля или подайте питание от блока питания AC/DC или от батареи.

ATmega32u4 имеет встроенные средства для связи через USB, устраняющие необходимость в дополнительном процессоре. Это позволяет в дополнение к виртуальному (CDC) последовательному COM-порту подключать Arduino Leonardo к компьютеру, например как мышь или клавиатуру, или в других применениях.

Сводная таблица технических характеристик.

Микроконтроллер	ATMega32u4
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7-12 В

Входное напряжение (предельное)	6-20 В
Цифровые входные/выходные выводы	20
ШИМ каналы	7
Аналоговые входные выводы	12
Постоянный ток через входные/выходные выводы	40 мА
Постоянный ток через вывод 3,3 В	50 мА
Флеш-память	32 Кб (ATmega32u4), из которых 4 Кб используются загрузчиком
Оперативная память SRAM	2,5 Кб
Энергонезависимая память EEPROM	1 Кб
Тактовая частота	16 МГц
Длина	68,6 мм
Ширина	53,3 мм
Вес	20 г

Питание платы осуществляется от внешнего источника 5В постоянного тока подключенного к пинам Vin и GND. Программирование платы происходит по USB-интерфейсу, отладка и связь с ПК идет через порт `Serial`, тогда как ESP8266 подключен к `Serial1`, для чего задействованы пины 0 и 1. Пин 10 включен параллельно через резистор 4,7КОм с пином 2 (Data) датчика температуры DS18B20, питание которого производится по пинам 1 и 3 и берется с общей шина +5В на макетной плате. Блок реле запитывается также от +5В, его задача, при получении высокого уровня сигнала с цифрового вывода 11 разорвать цепь линию переменного тока L и обеспечить отключение электронагревателя в помещении.

Дисплей Winstar WH-1602B размерности 16x02 (т.е. по 16 символов в двух строках), на базе контроллера HD44780, являются одними из самых простых, доступных и востребованных дисплеев для разработки различных электронных устройств. Режим самотестирования для проверки и настройки контрастности и яркости дисплея включается при подключении устройства следующим образом: запитывается контроллер на выводах VSS и VDD, подсветка (пины A и K), регулирующий контрастность пин V0 подключен через третий вывод потенциометра на 10 кОм, на крайние ножки которого подается +5В и GND**.** После подачи питания на схему необходимо добиться правильного контраста, если он будет настроен не верно, то на экране ничего не будет отображаться. Для настройки контраста следует поиграться с потенциометром. При правильной сборке схемы и правильной настройке контраста, на экране должна заполниться прямоугольниками верхняя строка.

Отображение кириллицы на дисплее возможно благодаря реализации библиотеки LiquidCrystalRus https://github.com/mk90/LiquidCrystalRus.

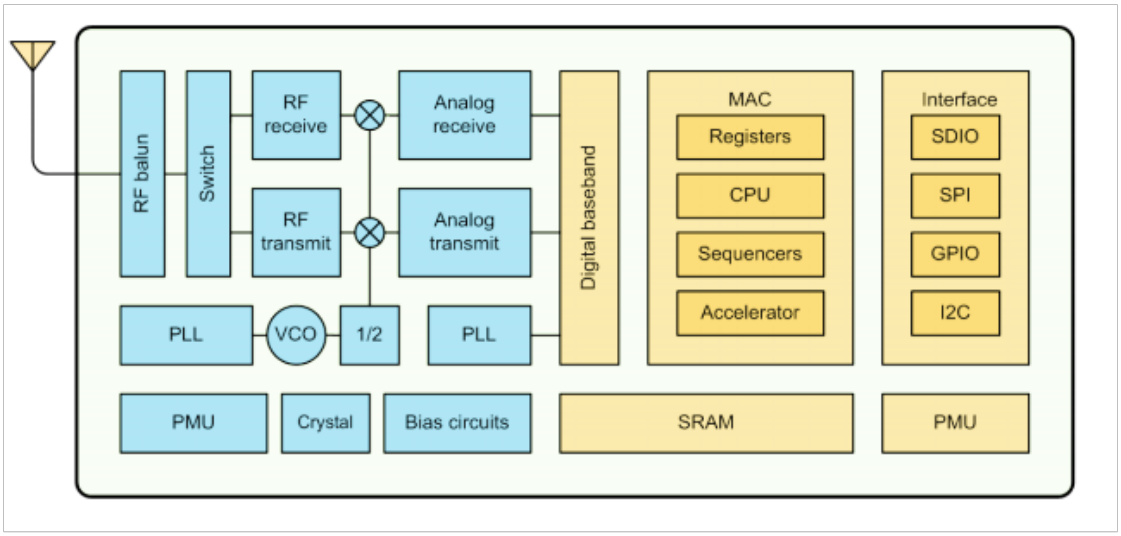
Подключение дисплейного модуля WH1602B.

- 1 — **Vss**, земля ⇨ GND
- 2 — **Vdd**, питание ⇨ +5 В
- 3 — **Vo**, управление контрастностью напряжением ⇨ выход потенциометра
- 4 — **RS**, выбор регистра ⇨ пин 12 Arduino
- 5 — **R/W**, чтение/запись ⇨ GND (режим записи)
- 6 — **E**, он же Enable, строб по спаду ⇨ пин 9 Arduino
- 7-10 — **DB0-DB3**, младшие биты 8-битного интерфейса; не подключены
- 11-14 — **DB4-DB7**, старшие биты интерфейса ⇨ пины 5-2 Arduino
- 15 — **A**, питание для подсветки ⇨ +5 В
- 16 — **K**, земля для подсветки ⇨ GND

Модуль ESP8266.

Является простой и дешевой альтернативой полноразмерному WiFi Shield от Arduino. Представлена самая распространенная и компактная модификация ESP-01. В чипе интегрированы антенные разъемы, согласующее устройство, малошумный приемный усилитель, фильтры, модули питания. Это в совокупности позволяет обходится минимумом внешней обвязки и меньшими габаритами. Данное устройство обладает широкими возможностями и функционалом, не смотря на скромные размеры и цену, так как это полноценный микроконтроллер с поддержкой беспроводных интерфейсов, он может обеспечить работу веб-интерфейса и `JSON API`, что весьма интересно в контексте реализации идей т.н. «интернета вещей». Коммуникация с ним осуществляется по UART, для чего задействованы пины 0 и 1 платы Leonardo.

Принципиальная схема ИС ESP8266.



• Стандрты WiFi 802.11 b/g/n

• Интегрированный 32-битный процессор

• Встроеный 10-битный АЦП

• Реализация стэка TCP/IP

• Интегрированный радиоприемные модули

• Интегрированные системы питания

• Поддержка различных антенн

• WiFi 2.4 GHz, support WPA/WPA2

• Поддержка STA/AP/STA+AP режимов

• Поддержка Link Function для Android и iOS устройств

• SDIO 2.0, (H) SPI, UART, I2C, I2S, IR Remote Control, PWM, GPIO

• STBC, 1x1 MIMO, 2x1 MIMO

• A-MPDU & A-MSDU

• Потребление в режиме сна <10uA, Утечка при отключении питания < 5uA

• Время реакции < 2ms

• Потребление < 1.0mW (DTIM3)

• +20 dBm выходная мощность 802.11b mode

• Рабочий диапазон температур -40C ~ 125C

• FCC, CE, TELEC, WiFi Alliance, и SRRC сертификаты.

Реализация программы для обработки информации от датчика температуры и беспроводной передачи данных на удаленный сервер в IDE Arduino.

В качестве инструмента разработки на данном этапе используется среда arduino-1.6.7. Для загрузки исходного кода в выпадающем меню выбирается пункт «Сервис - Плата - Arduino Leonardo». Для работы с устройствами датчиком температуры DS1820B по протоколу OneWire и компактным WiFi-модулем ESP8266 были взяты внешние библиотеки, содержащие встроенные функции, реализующие соответственно опрос устройств по OneWire (https://github.com/PaulStoffregen/OneWire) и пересылку AT-команд в беспроводной модуль по последовательному порту (https://github.com/itead/ITEADLIB_Arduino_WeeESP8266), что существенно ускоряет отработку взаимодействия с поддерживаемыми устройствами.

Структурно скетч разделен по функциона ильности на несколько процедур, отвечающих за работу с устройствами:

`temp_sensor()` – Функция, отвечающая за обработку показаний датчика температуры с помощью функционала библиотеки `OneWire`, а также вызывающая процедуры отображения текущего значения температуры с точностью до пол градуса;

`wireless_utp(float)` – Вызывается из предыдущей и принимает значение температуры, далее посредством функций recv() и send() библиотеки `ESP8266` организуется сетевое взаимодействие с удаленным хостом, после начальной инициализации подключения в процедуре setup() ;

`lcd_temp(float)` – Принимает дробное значение параметра и отображает его как объект класса `String` на экране дисплея с помощью функции `LiquidCrystal.print()` из библиотеки `LiquidCrystalRus`, поддерживающей кириллические символы;

`setup()` – инициализация основных объектов используемых классов;

`loop()` – здесь производится вызов данных функций в бесконечном цикле, оставлены отладочные сообщения передаваемые на COM-порт и обрабатываемые сервером;

`relay(int)` – управляет по изменению уровня сигнала на пине 11 открытием закрытием реле, получая на вход в качестве аргумента состояние 0 или 1, соответственно включить или выключить устройство.

Исходный текст скетча.

```
#include <LiquidCrystalRus.h>
#include <ESP8266.h>
#include <OneWire.h>
#define MY_SSID     "MGTS_GPON"
#define PASSWORD    "PASSWORD"
#define HOST_NAME   "192.168.1.67"
#define HOST_PORT   (7143)

ESP8266 wifi(Serial1);
//***//
int led_pin = 13,
    rel_pin = 11,
    answer_ping = 0;
OneWire  ds(10);  // датчик DS18B20 подключен на 10 пин
                  // в параллель с 4.7K резистором и VCC

LiquidCrystalRus lcd(12, 9, 5, 4, 3, 2);

void serial_ping(void)
{
  digitalWrite(led_pin, HIGH);
  Serial.println("PING?");
  delay(2000);
  digitalWrite(led_pin, LOW);
  delay(2000);
}

void led_stable(void)
{
```

```
 digitalWrite(led_pin, HIGH);
}
void relay(int rel_status)
{
  int i = 0;
  switch (rel_status) {
  case 2:
        for (i = 0; i < 3; i++) {
            digitalWrite(rel_pin, LOW);
            delay(5000);
            digitalWrite(rel_pin, HIGH);
            delay(5000);
            digitalWrite(rel_pin, LOW);
        }
        break;
  case 1:                                  // 1 - вкл выхода реле
        if (digitalRead(rel_pin) == HIGH)
            digitalWrite(rel_pin, LOW);
        break;
  case 0:                                  // 0 - выкл выхода реле
        if (digitalRead(rel_pin) == LOW)
            digitalWrite(rel_pin, HIGH);
        break;
  }
}


void temp_sensor(void) // Функция получения и обработки данных от
датчика температуры
{
  byte i;
  byte present = 0;
  byte type_s;
```

```
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }
  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }
  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return;
  }
  Serial.println();
    switch (addr[0]) {
    case 0x10:
      Serial.println("  Chip = DS18S20");  // определение типа датчика
DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
```

```
      Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return;
  }
  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);
  delay(1000);
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);
  Serial.print("  Data = ");
  Serial.print(present, HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print(OneWire::crc8(data, 8), HEX);
  Serial.println();
  // Преобразование данных в текущее значение
  // Резульатат - 16 битное  целое со знаком
  // сохраняется в переменной типа "int16_t", котрая всегда занимает 16
бит
  // даже прикомпиляции на 32 битном процессоре.
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
```

```
    raw = raw << 3;
    if (data[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// Результат занимает 12 бит, 750 мс время обработки
  }
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;
  Serial.print("  Temperature = ");
  Serial.print(celsius);
  Serial.print(" Celsius, ");
  Serial.print(fahrenheit);
  Serial.println(" Fahrenheit");

  lcd_temp(celsius);        // Отображение температуры на LCD
  wireless_utp(celsius);    // Отправка значения температуры по udp
}

void lcd_temp(float t)  // Функция вывода значения температуры на LCD
{
  String deg = String(t, 2);
  lcd.clear();
  lcd.print("Температура:");
  lcd.setCursor(0,1);
  lcd.print("t=");
  lcd.print(deg);
  lcd.print("\377C");
```

```
}
void wireless_utp(float t) // Передача данных по беспроводной сети
{
    char buf[128] = {0};
    if (wifi.registerUDP(HOST_NAME, HOST_PORT)) {
        Serial.print("register udp ok\r\n");
    } else {
        Serial.print("register udp err\r\n");
    }
    String temp_str = String("T=");
    String celsius_str = String(t);
    temp_str += celsius_str;
    temp_str.toCharArray(buf, temp_str.length());
    wifi.send((const uint8_t*)buf, (uint32_t)strlen(buf));
    memset(buf, 0, sizeof(buf)/sizeof(buf[0]));
    uint32_t len = wifi.recv((uint8_t *)buf, sizeof(buf), 10000);
    if (len > 0) {
        Serial.print("Received:[");
        for(uint32_t i = 0; i < len; i++) {
            Serial.print((char)buf[i]);
        }
        Serial.print("]\r\n");
    }
    if (wifi.unregisterUDP()) {
        Serial.print("unregister udp ok\r\n");
    } else {
        Serial.print("unregister udp err\r\n");
    }
    delay(5000);
}
void setup(void)  // Установка начальных параметров
{
```

```
  pinMode(led_pin, OUTPUT);
  pinMode(rel_pin, OUTPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.print("Старт...");
  relay(0);                    // Производим начальное выключение
//нагревателя
  delay(3000);
  Serial.print("WiFi setup begin\r\n");
  Serial.print("FW Version:");
  Serial.println(wifi.getVersion().c_str());

  if (wifi.setOprToStationSoftAP()) {
    Serial.print("to station + softap ok\r\n");
  }
  else {
      Serial.print("to station + softap err\r\n");
  }

  if (wifi.joinAP(MY_SSID, PASSWORD)) {
    Serial.print("Join AP success\r\n");
    Serial.print("IP: ");
    Serial.println(wifi.getLocalIP().c_str());
  }
  else {
    Serial.print("Join AP failure\r\n");
  }

  if (wifi.disableMUX()) {
    Serial.print("single ok\r\n");
  }
  else {
```

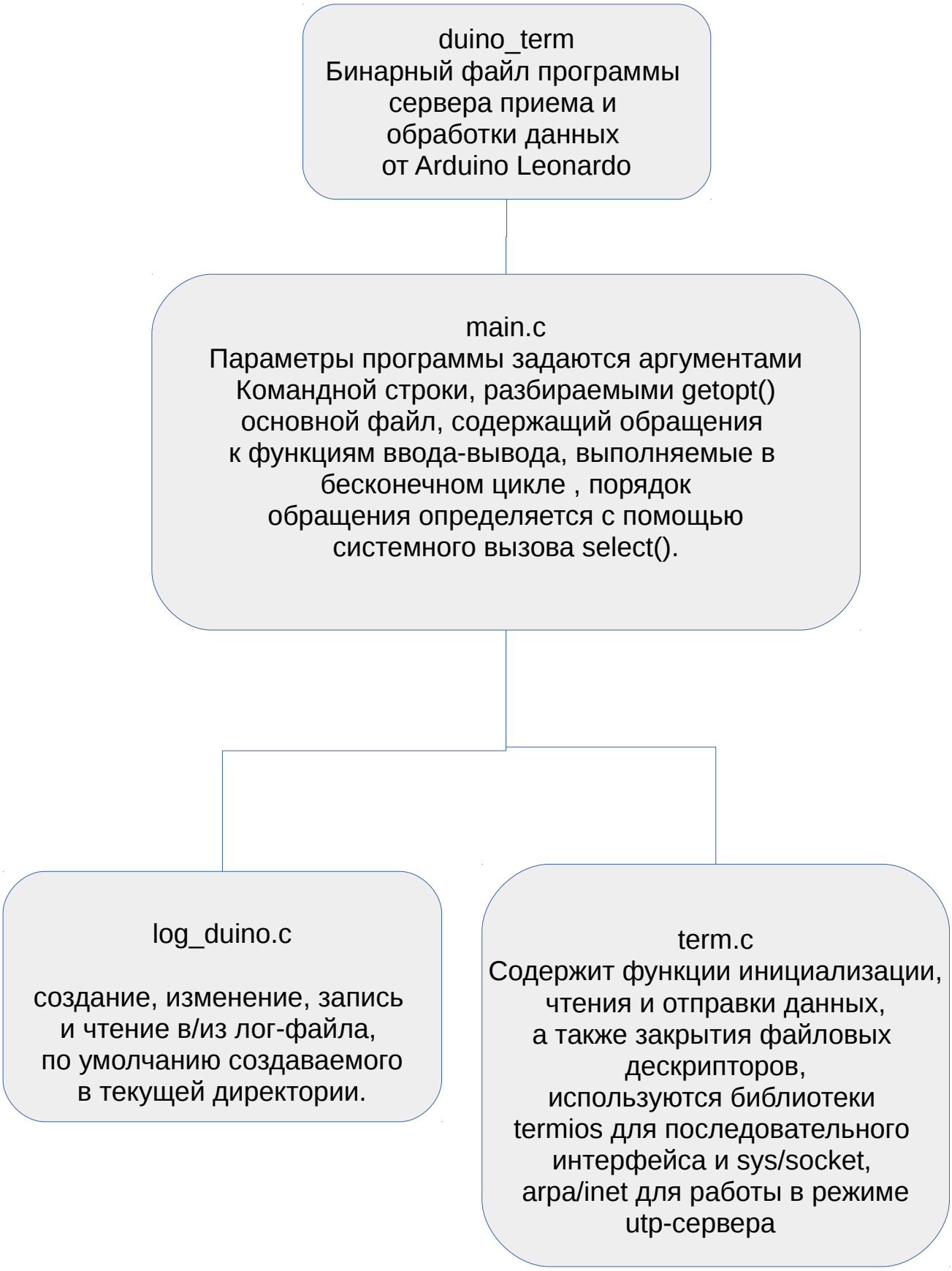
```
    Serial.print("single err\r\n");
  }

  Serial.print("setup end\r\n");
}

void loop(void)
{
  char pong_buf[64];
  unsigned int i = 0;
  temp_sensor();
  while (Serial.available()) {
    pong_buf[i] = Serial.read();
    if ( i > 0 &&
    pong_buf[i] == '!' &&
    pong_buf[i - 1] == 'P' ) // Определение, получен ли PONG! в ответ
      answer_ping++;
    i++;
  }
  if (!answer_ping && millis() < 30000) { //или ждём PONGa или 30 сек до
//успешной установки связи с ПК
     serial_ping(); // Отслылка PING? пока не будет ответа от сервера
     return;
  }
  else {
     led_stable(); // Соединение установлено или мы не дождались ответа
по COM-порту
  }
  delay(10000);
}
```

Реализация программы удаленного сервера на языке С.

В учебных целях для большего охвата, используемых технологий и уяснения теоретических положений дисциплины ИТ было принято решение о реализации программы сервера на языке `C`, достоинствами которого являются гибкость, скорость работы решений, проработанность, широта применения и универсальность. Исходные тексты программы, находятся под управление системы контроля версий `git` на ресурсе github.com, репозиторий создавался, специально для практики в работе с `git`. Используемый в среде ОС Linux инструментарий для раработки: редактор `Vim`, компилятор `gcc-4.8`, утилита сборки `make`, система контроля версий `git`.

Все исходные файлы собираются и линкуются посредством стандартной Unix-утилиты `make`, директивы которой записаны в файле Makefile:

```
CC = gcc
CLEAN = rm -f
TARGET = duino_term
SOURCE = main.c term.c log_duino.c

DEFINES = -DDEBUG_DUINO

all : ${TARGET}

${TARGET} : ${SOURCE:.c=.o}
	${CC} ${SOURCE:.c=.o} -o $@

${SOURCE:.c=.o} : ${SOURCE}
	${CC} ${DEFINES} -c ${SOURCE}

clean :
	${CLEAN} *.o ${TARGET}
```


duino_term
Бинарный файл программы
сервера приема и
обработки данных
от Arduino Leonardo
main.c
Параметры программы задаются аргументами
Командной строки, разбираемыми getopt()
основной файл, содержащий обращения
к функциям ввода-вывода, выполняемые в
бесконечном цикле , порядок
обращения определяется с помощью
системного вызова select().
log_duino.c
создание, изменение, запись
и чтение в/из лог-файла,
по умолчанию создаваемого
в текущей директории.
term.c
Содержит функции инициализации,
чтения и отправки данных,
а также закрытия файловых
дескрипторов,
используются библиотеки
termios для последовательного
интерфейса и sys/socket,
arpa/inet для работы в режиме
utp-сервера

Заключение

В ходе выполнения работы были освоены основные понятия и положения теоретических материалов по дисциплине «Информационные Технологии». Изучена работа с инструментарием Arduino, позволяющим работать с электронными устройствами, на базе микроконтроллеров и ИС, используя элементы языков программирования высокого уровня и упрощенный синтаксис скетчей, компилируемых и загружаемых с мощью загрузчика для микроконтроллеров серии Atmega. Собран действующий макет устройства с применением датчиков, исполнительных и сетевых элементов, обеспечивающих работу системы. Намечены основные пути развития и исправления допущенных ошибок этого проекта. Применены знания по концепциям разработки ПО, а также имеющиеся навыки программирования.