Утверждена распоряжением декана ЭТФ № 1 от 21.01.2020 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ «Промышленная робототехника»

СОГЛАСОВАНО

Декан ФЭУ

_____ А.С. Гудим
(подпись)

«04 » июня 2021 г.

Заведующий кафедрой _____

_____ М.А. Горькавый
(подпись)

«04 » июня 2021 г.

УТВЕРЖДАЮ

Начальник отдела ОПРО

_____ В.В. Солецкий
(подпись)

«04 » июня 2021 г.

Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера

Комплект документации на управляющую программу для автоматизированной/роботизированной системы

Руководитель проекта _____ 04.06.2021 *В.С. Соколова*
(подпись, дата)

Ответственный исполнитель _____ 04.06.2021 *Д.Ю. Облогин*
(подпись, дата)

Комсомольск-на-Амуре 2021

Карточка проекта

Название	*Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера*
Тип проекта	*Учебная работа*
Исполнители	*Д.Ю. Облогин – 8МРб-1*
Срок реализации	*январь, 2021 - июнь, 2021*

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ

на разработку

Выдано студентам: *Д.Ю. Облогин – 8МРб-1* _____

_____

Название проекта: *Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера*.

Назначение: *Программа предназначена для автоматизированного контроля и измерения температурно-влажностного режима работы абсорбционного кондиционера.*

Область использования: *проведение контроля и измерения параметров микроклимата помещения при работе абсорбционного кондиционера, в том числе, проведение оперативного контроля процесса охлаждения воздушного потока, контроля стабильности показателей температуры и влажности, а также соответствия текущих значений заданным значениям.*

Функциональное описание управляющей программы:

Программа должна обеспечить управление основными режимными параметрами абсорбционного кондиционера - температурой и влажностью Программа позволит определить выходные значения переменных состояния объекта (абсорбционного кондиционера) путем управления скоростью циркуляционного вентилятора, который перемещает охлаждаемый воздушный поток, а также управления заслонкой, обеспечивающей контроль объема охлаждаемого воздуха. Дополнительно программа позволит

осуществлять мониторинг работы кондиционера с помощью систематизации данных; предоставления средних значений температуры и относительной влажности воздуха отклоняющихся от заданного диапазона с последующим выводом информации на дисплей.

Оборудование, для которого разрабатывается программа:

1. Датчик температуры DS18B20 - цифровой температурный датчик

2. Датчик скорости воздуха

3. Одноплатный компьютер типа Arduino

Требования:

Тип ЭВМ: IBM PC Pentium III, и выше

ОС: Microsoft Windows 9x/NT/2000/XP/Vista/7/8/8.1/10

План работ:

Наименование работ	Срок
Анализ требований к конечному продукту, постановка цели и задач	*январь.2021*
Разработка технического задания и проектирование	*март.2021*
Кодирование	*май.2021*
Тестирование и отладка	*июнь.2021*

Комментарии:

_____

_____

_____

_____

_____

_____

Перечень графического материала:

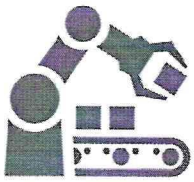
1. Структурная схема программного комплекса для контроля, управления и анализа работы абсорбционного кондиционера.

Руководитель проекта 29.01.2021 *В.С. Соколова*

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»



ПАСПОРТ

Управляющей программы для автоматизированной/роботизированной системы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера»

Руководитель проекта _____ 04.05.2021 *В.С. Соколова*

(подпись, дата)

Ответственный исполнитель _____ 04.05.2021. *Д.Ю. Облогин*

(подпись, дата)

Комсомольск-на-Амуре 2021

Содержание

1 Общие положения 8
1.1 Наименование программы 8
1.2 Наименования документов, на основании которых ведется проектирование системы 8
1.3 Перечень организаций, участвующих в разработке системы 8
1.4 Сведения об использованных при проектировании 9
нормативно-технических документах 9
2 Описание программы 10
2.1 Общие сведения 10
2.2 Функциональное назначение программы 11
2.3 Описание логической структуры 11
2.4 Используемые технические средства 11
2.5 Вызов и загрузка 11
2.6 Входные данные (при наличии) 12
2.7 Выходные данные (при наличии) 13
3 Руководство оператора 14
3.1 Назначение программы; 14
3.2 Условия выполнения программы; 14
3.3 Выполнение программы; 14
ПРИЛОЖЕНИЕ А 16
ПРИЛОЖЕНИЕ Б 23

1 Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основной структурой, особенностями и правилами эксплуатации управляющей программы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера» (далее «программа»).

Паспорт входит в комплект поставки программы. Перед запуском программы внимательно изучите правила ее эксплуатации.

1.1 Наименование программы

Полное наименование программы – «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера».

1.2 Наименования документов, на основании которых ведется проектирование системы

Создание программы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера» осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке системы

Заказчиком создания программы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера» является Федеральное государственное бюджетное образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 27.

Исполнителями работ по созданию программы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера» являются Конструкторы студенческого конструкторского бюро (далее СКБ), студенты группы 8МРб-1, Облогин Данил Юрьевич.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 19.001-77. Единая система программной документации (ЕСПД). Общие положения.

ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения

ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.

ГОСТ 19.102-77. ЕСПД. Стадии разработки.

ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.402-78. ЕСПД. Описание программы.

ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.

ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению.

2 Описание программы

2.1 Общие сведения

Для исследования возможности реализации исследовательской работы, программная часть контроллера должна обеспечивать не только опрос датчиков, но и сохранение полученных значений для последующего анализа.

Задача обработки числовых данных и их графическое представление несет важное практическое значение в области обработки экспериментальных данных и научных исследований. Часто на практике при выполнении расчетов без применения средств автоматизации выполняют вычисления значений функциональных зависимостей вручную, что приводит к большой погрешности вычислительных данных при больших затратах труда и времени.

Программный комплекс позволяет решить ряд проблем: возможность сформировать графическое представление полученных в ходе эксперимента данных, хранить результаты экспериментов в базе данных для последующего изменения или отображения, а также автоматизировать процесс расчета значений функциональных зависимостей.

Кроме представления отдельных результатов эксперимента, так же интерес представляет расчет промежуточных значений с применением различных методов: аппроксимации, интерполяции и экстраполяции. В результате применения этих методов можно получить не только дискретные значения каки-либо параметров, но и заменить дискретные значения непрерывными (кусочно-непрерывными) функциями с целью применения к результатам эксперимента различных методов анализа данных.

2.2 Функциональное назначение программы

Программа применяется для автоматизированного контроля и измерения температурно-влажностного режима работы абсорбционного кондиционера.

2.3 Описание логической структуры

Программный комплекс для контроля, управления и анализа работы абсорбционного кондиционера является сложным программно-аппаратным комплексом, состоящим из независимых подсистем, осуществляющих обмен данными с применением различных протоколов. Общая структурная схема программного комплекса приведена на рисунке 1.

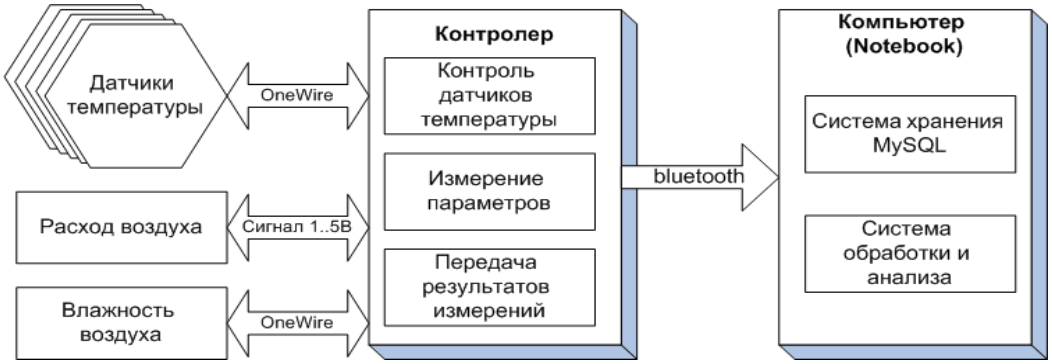


Рисунок 1 – Структурная схема программного комплекса для контроля, управления и анализа работы абсорбционного кондиционера

Текст программы приведен в Приложении А.

2.4 Используемые технические средства

Для функционирования программы необходимо следующее техническое обеспечение:

- процессор Intel Core 2 Duo 2 ГГц;
- ОС Windows 7 или выше;

- оперативная память 1024 Мб для нормального функционирования операционной системы Windows;

- свободное место на диске размером не менее 100 Мб;

- видеоадаптер SVGA и выше;

- дисплей XGA (1024*768);

- клавиатура;

- манипулятор «мышь».

2.5 Вызов и загрузка

Основная задача управляющего модуля заключается в опросе датчиков, сохранение результатов опроса в оперативной памяти и передачи данных в режиме реального времени на ноутбук.

Результаты измерений в цифровом виде передаются с контроллера на компьютер (ноутбук) посредством модуля Bluetooth по plain/text протоколу; модуль сохранения данных, работающий на ноутбуке, сохраняет полученные данные в базе данных MySQL , работающей по типу простого OPC-сервера. Сервер базы данных MySQL может располагаться как непосредственно на том же компьютере, так и на сервере в сети. Если ноутбук работает в автономных условиях из-за недоступности сервера (например, отсутствует возможность подключиться к локальной сети во время проведения лабораторных экспериментов), то данные лабораторной установки сохраняются локально и выгружаются на сервер после подключения ноутбука к сети.

2.6 Входные данные (при наличии)

В качестве входных данных при каждом новом запуске программы задаются значения максимальной и минимальной температуры и влажности воздуха.

2.7 Выходные данные (при наличии)

При выполнении программы генерируются в качестве выходных при выполнении программы показания датчиков и состояние оборудования.

3 Руководство оператора

3.1 Назначение программы;

Результат выполнения программы приведен в Приложении Б.

3.2 Условия выполнения программы;

Условия эксплуатации отдельных компонент (датчиков) могут отличаться от условий эксплуатации контрольно-измерительной системы и определяются условиями и параметрами измеряемой среды. Требования к надёжности приведены в таблице 1.

Таблица 1 – Требования к надёжности

Параметр	Единица измерения	Значение
Режим работы оборудования	Непрерывный с остановками на НИР	
Срок службы оборудования, не менее	лет	5

По эксплуатационным характеристикам комплекс оборудования, входящего в измерительную систему, должен удовлетворять требованиям ГОСТ Р 55064-2012, ГОСТ ISO 12100-2013, а также другим нормам и правилам, действующим на территории Российской Федерации.

Режим эксплуатации круглогодичный, прерывистый. Данные требования распространяются на контрольно - измерительную систему экспериментальной установки лабораторного образца абсорбционного кондиционера.

В качестве датчика температуры используется датчик DS18B20 - цифровой температурный датчик, который имеет множество функций. По внутренней структуре датчик температуры DS18B20 - это сложный микроконтроллер, который хранит значения измерений, сигнализирует о выходе температуры за установленные пределы, установленные пользователем. Точность представления результата измерения температуры может быть про-

граммно изменена от 9 до 12 бит при диапазоне измеряемой температуры -55..+125^{0}C.

3.3 Выполнение программы;

Для считывания пакета данных на устройство выдается команда на выполнение внутренней операции преобразования. Для DS18B20 это команда Convert T (0x44) byte. В библиотеке OneWire он выдается как ds.write (0x44), где ds - это экземпляр класса OneWire. После подачи этой команды устройство считывает внутренний АЦП, и по завершении этого процесса копирует данные в регистры блокнота. Продолжительность этого процесса преобразования зависит от разрешения и указана в техническом описании устройства. DS18B20 требует от 94 (9-битное разрешение) до 750 мс (12-битное разрешение) для преобразования температуры (см. техническое описание DS18B20). Во время преобразования устройство может быть опрошено, например используя команду ds.read () в OneWire, чтобы узнать, успешно ли выполнено преобразование.

После преобразования данные копируются в память электронного блокнота, откуда их можно прочитать. Обратите внимание, что электронный блокнот может быть прочитан в любое время без команды преобразования, чтобы вызвать самое предыдущее чтение, а также разрешение устройства и другие параметры конфигурации, зависящие от устройства.

ПРИЛОЖЕНИЕ А

(обязательное)

Текст программы

```
// Tested with OneWire Version 2.3
// https://github.com/PaulStoffregen/OneWire
#include "OneWire.h"
#define pin_onewire 7
OneWire *ds;

volatile uint64_t timer_us;
volatile uint64_t timer_interval_us;

void print_hex(uint8_t value) {
  if (value < 16) Serial.write('0');
  Serial.print(value, HEX);
}

void print_array(uint8_t *data, int n, char sep = ',') {
  int idx;
  for (idx=0; idx<n; idx++) {
    print_hex(data[idx]);
    if (idx != n-1)
      Serial.write(sep);
  }
}

bool read_scratchpad(uint8_t *addr, uint8_t *buff9) {
  ds->reset();
  ds->select(addr);
  ds->write(0xBE); // read scratchpad
  int idx;
  for (idx=0; idx<9; idx++)
    buff9[idx] = ds->read();
  return 0 == OneWire::crc8(buff9, 9);
}

int lengthFF(uint8_t *buff, int len) {
 (NB: may be 0)
  while (len>0) {
    if (buff[len-1] != 0xff) break;
    len--;
  }
  return len;
}

void send_reset(uint8_t *addr) {
  ds->reset();
  ds->select(addr);
  ds->write(0x64);
  delay(10);
}

uint8_t one_byte_return(uint8_t *addr, uint8_t cmd) {
  ds->reset();
  ds->select(addr);
  ds->write(cmd);
  return ds->read();
}
```

```
uint8_t bit_invert(uint8_t a) {
  uint8_t b = 0;
  int i;
  for (i=0; i<8; i++) {
    b *= 2;
    b += (a & 0x01);
    a /= 2;
  }
  return b;
}

void param2trim(uint16_t offset_param_11bit, uint8_t curve_param_5bit, uint8_t *param1, uint8_t *param2) {
  *param1 = bit_invert(offset_param_11bit & 0x0ff);
  *param2 = curve_param_5bit * 8 + offset_param_11bit / 256;
}

void trim2param(uint8_t param1, uint8_t param2, uint16_t *offset_param_11bit, uint8_t *curve_param_5bit) {
  *offset_param_11bit = bit_invert(param1) + ((uint16_t)(param2 & 0x07)) * 256;
  *curve_param_5bit = param2 / 8;
}

void get_trim_A(uint8_t *addr, uint8_t *trim1, uint8_t *trim2) {
  ds->reset();
  if (addr) ds->select(addr);
  else ds->write(0xCC);
  ds->write(0x93);
  *trim1 = ds->read();
  ds->reset();
  if (addr) ds->select(addr);
  else ds->write(0xCC);
  ds->write(0x68);
  *trim2 = ds->read();
}

void set_trim_A(uint8_t *addr, uint8_t trim1, uint8_t trim2) {
  ds->reset();
  ds->select(addr);
  ds->write(0x95);
  ds->write(trim1);

  ds->reset();
  ds->select(addr);
  ds->write(0x63);
  ds->write(trim2);
  // don't store permanently, don't call reset
}

void get_trim_params_A(uint8_t *addr, uint16_t *offset_param_11bit, uint8_t *curve_param_5bit) {
  uint8_t trim1, trim2;
  get_trim_A(addr, &trim1, &trim2);
  trim2param(trim1, trim2, offset_param_11bit, curve_param_5bit);
}

void set_trim_params_A(uint8_t *addr, uint16_t offset_param_11bit, uint8_t curve_param_5bit) {
  uint8_t trim1, trim2;
  param2trim(offset_param_11bit, curve_param_5bit, &trim1, &trim2);
  set_trim_A(addr, trim1, trim2);
}

bool is_valid_A_scratchpad(uint8_t *buff) {
  if ((buff[4] != 0x7f) && (buff[4] != 0x5f) && (buff[4] != 0x3f) && (buff[4] != 0x1f)) return false;
```

```
  if ((buff[0] == 0x50) && (buff[1] == 0x05) && (buff[6] == 0x0C)) return true; // power-up
  if ((buff[0] == 0xff) && (buff[1] == 0x07) && (buff[6] == 0x0C)) return true; // unsuccessful conversion
  return buff[6] == (0x10 - (buff[0] & 0x0f));
}

bool is_all_00(uint8_t *buff, int N) {
  int i;
  for (i=0; i<N; i++)
    if (buff[i] != 0x00) return false;
  return true;
}

void trigger_convert(uint8_t *addr, uint8_t conf, uint16_t wait) {
  ds->reset();
  ds->select(addr);
  ds->write(0x4E);
  ds->write(0xaf);
  ds->write(0xfe);
  ds->write(conf);

  ds->reset();
  ds->select(addr);
  ds->write(0x44, 1);
  delay(wait);
}

int curve_param_prop(uint8_t *addr) {
  uint8_t buff[9];
  uint16_t off = 0x3ff; // half way

  set_trim_params_A(addr, off, 0x0f);
  trigger_convert(addr, 0x7f, 800);
  if (!read_scratchpad(addr, buff)) read_scratchpad(addr, buff);
  if (0 != OneWire::crc8(buff,9)) return -3;
  if (!is_valid_A_scratchpad(buff)) return -2;
  int16_t r0f = buff[0] + 256*buff[1];
  {
    uint16_t o;
    uint8_t c;
    get_trim_params_A(addr, &o, &c);
    if ((o != off) || (c != 0x0f)) return 0;
  }

  set_trim_params_A(addr, off, 0x00);
  trigger_convert(addr, 0x7f, 800);
  if (!read_scratchpad(addr, buff)) read_scratchpad(addr, buff);
  if (0 != OneWire::crc8(buff,9)) return -3;
  if (!is_valid_A_scratchpad(buff)) return -2;
  int16_t r00 = buff[0] + 256*buff[1];

  set_trim_params_A(addr, off, 0x1f);
  trigger_convert(addr, 0x7f, 800);
  if (!read_scratchpad(addr, buff)) read_scratchpad(addr, buff);
  if (0 != OneWire::crc8(buff,9)) return -3;
  if (!is_valid_A_scratchpad(buff)) return -2;
  int16_t r1f = buff[0] + 256*buff[1];

  set_trim_params_A(addr, off, 0x10);
  trigger_convert(addr, 0x7f, 800);
  if (!read_scratchpad(addr, buff)) read_scratchpad(addr, buff);
  if (0 != OneWire::crc8(buff,9)) return -3;
  if (!is_valid_A_scratchpad(buff)) return -2;
```

```
  int16_t r10 = buff[0] + 256*buff[1];

  int16_t mini = min(r00, min(r0f, min(r10, r1f)));
  int16_t maxi = max(r00, max(r0f, max(r10, r1f)));
  bool is_signed = (r0f-r10 > r1f-r00);
  bool is_unsigned = (r0f-r10 < r1f-r00);

  if (is_signed && (maxi-mini > 20*16)) return 2; // A2
  if (is_unsigned && (maxi-mini > 1*16) && (maxi-mini < 6*16)) return 1; // A1

  return -1;
}

void setup() {
  Serial.begin(115200);
  Serial.println();
    ds = new OneWire(pin_onewire);

  Serial.print(F("  1-wire датчик подключен на пин "));
  Serial.println(pin_onewire, DEC);
}

void loop() {
  // ROM address of current sensor
  uint8_t addr[8];

  while (Serial.available())
    Serial.read();
  while (!Serial.available());
  while (Serial.available())
    Serial.read();

  int sensor_count = 0;
  ds->reset_search();
  while (ds->search(addr)) {
    sensor_count ++;

    // dump ROM
    print_array(addr, 8, '-');
    if (0 != OneWire::crc8(addr, 8)) {
      Serial.print(F(" (CRC Error)"));
    }

    Serial.print(F(":"));

    int identified = 0;

    { // test for family A
      uint8_t r68 = one_byte_return(addr, 0x68);
      uint8_t r93 = one_byte_return(addr, 0x93);
      if (r68 != 0xff) {
        int cpp = curve_param_prop(addr);
        if (cpp == 1) Serial.print(F(" Family A1 (Genuie Maxim).")); // unsigned and 3.8 oC range
        else if (cpp == 2) Serial.print(F(" Family A2 (Clone).")); // signed and 32 oC range
        else if (cpp == -3) Serial.print(F(" (Error reading scratchpad register [A].)"));
        else if (cpp == -1) {
          Serial.print(F(" Family A, unknown subtype (0x93="));
          print_hex(one_byte_return(addr, 0x93));
          Serial.print(F(", 0x68="));
          print_hex(r68);
          Serial.print(F(")."));
        }
```

```
    // cpp==0 or cpp==-2: these aren't Family A as we know them.
    if ((cpp==1) || (cpp==2) || (cpp==-1)) identified++;
   }
   send_reset(addr);
  }

  { // test for family B
   uint8_t r97 = one_byte_return(addr, 0x97);
   if (r97 == 0x22) Serial.print(F(" Family B1 (Clone)."));
   else if (r97 == 0x31) Serial.print(F(" Family B2 (Clone)."));
   else if (r97 != 0xFF) {
    Serial.print(F(" Family B (Clone), unknown subtype (0x97="));
    print_hex(r97);
    Serial.print(F(")."));
   }
   if (r97 != 0xff) identified++;
  }

  { // test for family D
   uint8_t r8B = one_byte_return(addr, 0x8B);
   // mention if parasitic power is known not to work.
   if (r8B == 0x06) Serial.print(F(" Family D1 (Clone w/o parasitic power)."));
   else if (r8B == 0x02) Serial.print(F(" Family D1 (Clone)."));
   else if (r8B == 0x00) Serial.print(F(" Family D2 (Clone w/o parasitic power)."));
   else if (r8B != 0xFF) {
    Serial.print(F(" Family D (Clone), unknown subtype (0x8B="));
    print_hex(r8B);
    Serial.print(F(")."));
   }
   if (r8B != 0xff) identified++;
  }

  {
   uint8_t buff[9];
   uint8_t cfg1, cfg2;
   ds->reset();
   ds->select(addr);
   ds->write(0x4E);
   ds->write(0xaa);
   ds->write(0x55);
   ds->write(0x00);
   if (!read_scratchpad(addr, buff)) read_scratchpad(addr, buff);
   if (is_all_00(buff, 9) || (0 != OneWire::crc8(buff,9))) goto err_C;
   cfg1 = buff[4];
   ds->reset();
   ds->select(addr);
   ds->write(0x4E);
   ds->write(0xaa);
   ds->write(0x55);
   ds->write(0xff);
   if (!read_scratchpad(addr, buff)) read_scratchpad(addr, buff);
   if (is_all_00(buff, 9) || (0 != OneWire::crc8(buff,9))) goto err_C;
   cfg2 = buff[4];
   ds->reset();
   ds->select(addr);
   ds->write(0x64);
   delay(10);
   if (cfg1 == cfg2) {
    Serial.print(F(" Family C (Clone)."));
    identified++;
   }
   if (0) {
```

```
err_C:
        Serial.print(F(" (Error reading scratchpad register [C].)"));
       }
     }
     if (identified == 0)
       Serial.print(F(" (Could not identify Family.)"));
     if (identified > 1)
       Serial.print(F(" (Part may belong to a Family not seen in 2019.)"));
     Serial.println();
   } // end iterate over all sensors

   if (sensor_count == 0) {
     Serial.println(F("No sensors detected."));
   }
   Serial.println(F("----------")); // indicate end
}

#include <OneWire.h> // библиотека для работы с термодатчиками DS18B20
OneWire ds(2);  // шина датчиков подключены к контакту D2
// НАСТРОЙКИ И ПРЕДВАРИТЕЛЬНЫЕ УСТАНОВКИ ПАРАМЕТРОВ
void setup(void) {
// Инициализируем COM-порт
Serial.begin(115200);
// ОСНОВНОЙ ЦИКЛ ПРОГРАММЫ
void loop(void) {
  byte ready_DS18xx = 0; // готовность датчика к обмену данными
  byte type_DS18xx; // типа термодатчика
  byte data_DS18xx[12]; // данные от датчика
  byte addr_DS18xx[8]; // 64-х битный адрес датчика
  float temper_DS18xx_celsius; // температура, град.цельсия
  // при опросе датчиков температуры, если все датчики опрошены,
  // либо не обнаружено ни одного датчика, выводим
  // соответствующее сообщение в COM-порт
  // повторный поиск выполняется через 250мс
  if (!ds.search(addr_DS18xx)) {
    Serial.println("No more addr_DS18xx."); Serial.println();
    ds.reset_search();
    delay(250); return;
  }
  // Определяем уникальный адрес датчика и выводим его
  // в COM-порт в HEX виде
  Serial.print("ROM =");
  for(int i = 0; i < 8; i++) { Serial.write(' '); Serial.print(addr_DS18xx[i], HEX); }
            // Проверка контрольной суммы адреса датчика
  if (OneWire::crc8(addr_DS18xx, 7) != addr_DS18xx[7]) {
      Serial.println("CRC for DS18xx is not valid!");
      return;
  }
  Serial.println();
  // Тип датчика (семейство датчиков) определяем по значению в
  // нулевом байте адреса этого датчика
  switch (addr_DS18xx[0]) {
    case 0x28: Serial.println("  Chip = DS18B20"); type_DS18xx = 0; break;
    case 0x22: Serial.println("  Chip = DS1822"); type_DS18xx = 0; break;
    case 0x10: Serial.println("  Chip = DS18S20"); type_DS18xx = 1; break;
    default: Serial.println("Device is not a DS18x20 family device.");
    return;
  }
  ds.reset(); // сброс шины передачи данных
  ds.select(addr_DS18xx); // работаем с текущим датчиков
  // для получения значения температуры в градусах цельсия,
  // подаем на дачик команду 0x44
```

```
  ds.write(0x44, 1);
  // оцифровка текущего значения температуры занимает 750мс
  // пауза не менее 750мс (с небольшим запасом делаем 1000мс)
  delay(1000);
  ready_DS18xx = ds.reset(); // сброс шины передачи данных
  ds.select(addr_DS18xx); // повтор выбора текущего датчика
  // для чтения значения текущей температуры
  // подаем на дачик команду 0xBE
  ds.write(0xBE);
  // значение температуры занимает 9 байт
  Serial.print("  data_DS18xx = "); Serial.print(ready_DS18xx, HEX); Serial.print(" ");
  for (int i = 0; i < 9; i++) {
    data_DS18xx[i] = ds.read();
    Serial.print(data_DS18xx[i], HEX); Serial.print(" ");
  }
// контрольная сумма
  Serial.print(" CRC=");
  Serial.println(OneWire::crc8(data_DS18xx, 8), HEX);
  // дешифровка пакета данных от датчика
  // объединяем байты 0 и 1 в одно 16-ти битное число
  int16_t raw = (data_DS18xx[1] << 8) | data_DS18xx[0];
  // Разрешение данных определяется типом датчика (type_DS18xx)
  // DS18B20 и DS1822 — 12бит, DS18S20 — 9бит
  if (type_DS18xx) { // Если датчик относится к семейству DS18S20
    raw = raw << 3; // разрешение по умолчанию равно 9 бит
    if(data_DS18xx[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data_DS18xx[6];
    }
  }
  else {
   // точность измерения для текущего датчика
    byte cfg = (data_DS18xx[4] & 0x60);
    // поскольку значения младших бит не определено, обнуляем
    if (cfg == 0x00) raw = raw & ~7;  // 9 бит (93.75 ms)
    else if (cfg == 0x20) raw = raw & ~3; // 10 бит (187.5 ms)
    else if (cfg == 0x40) raw = raw & ~1; // 11 бит (375 ms)
  }
  // расчет значения и вывод в COM-порт
  temper_DS18xx_celsius = (float)raw / 16.0;
  Serial.print("  Temperature = ");
  Serial.print(temper_DS18xx_celsius); Serial.print(" temper_DS18xx_celsius, ");
}
```

ПРИЛОЖЕНИЕ Б

(информационное)

Результаты выполнения программы

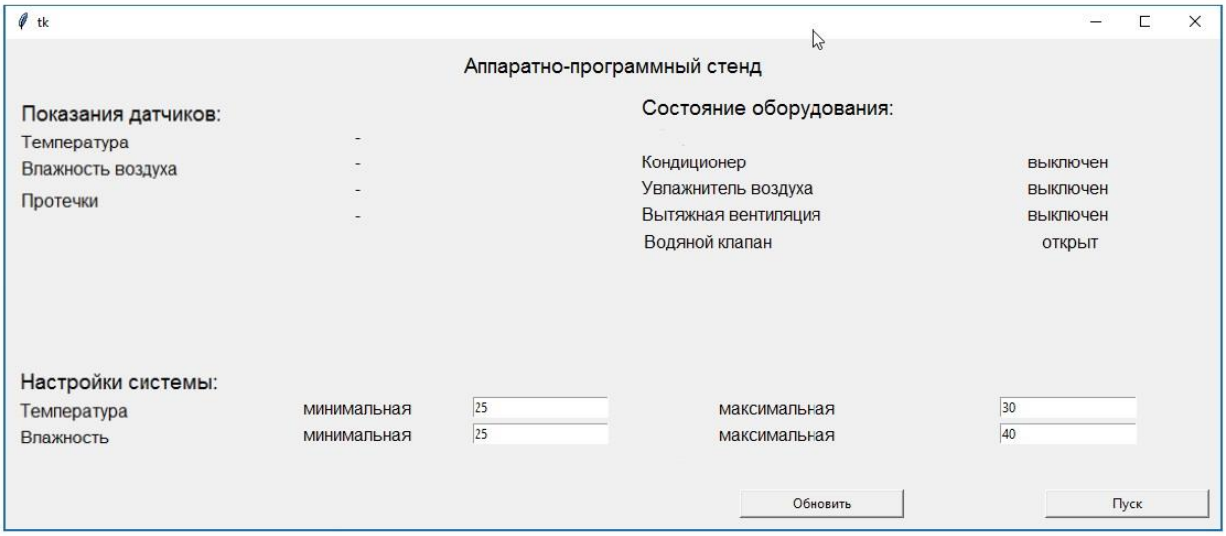


Рисунок Б1 – Главное окно программы перед запуском стенда

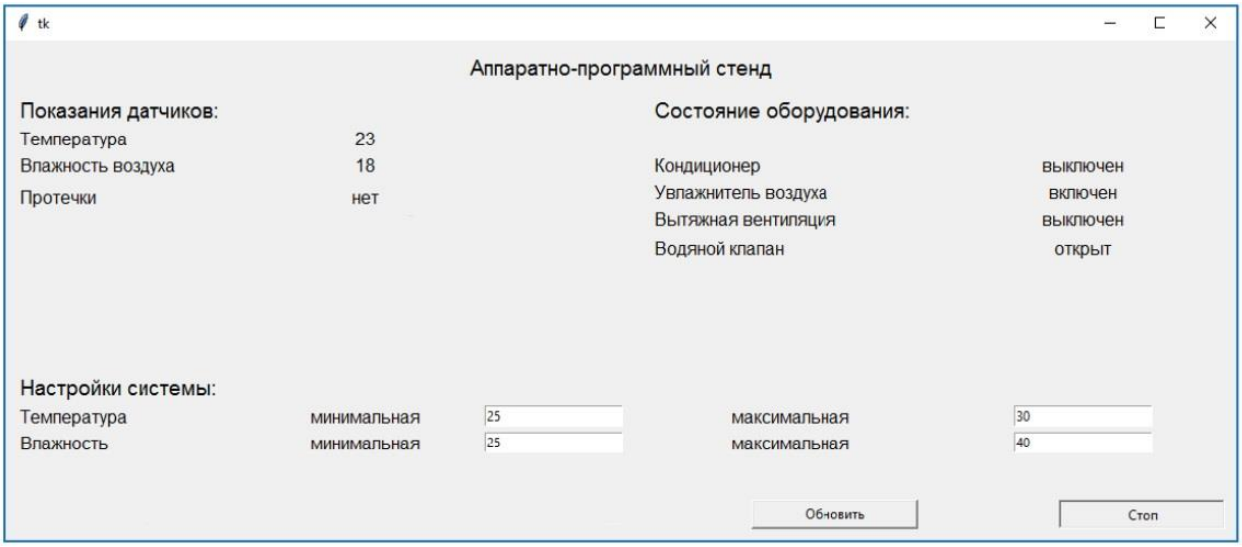


Рисунок Б2 – Главное окно программы после запуска стенда

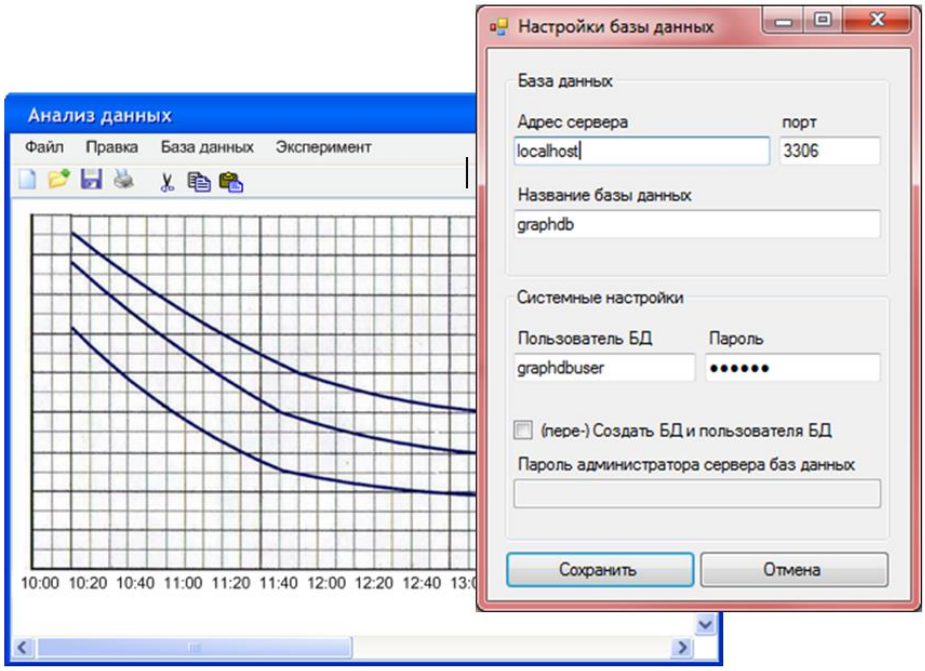


Рисунок Б3 – Результат выполнения программы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера»

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

УТВЕРЖДАЮ	СОГЛАСОВАНО
Декан ФЭУ	Заведующий кафедрой _____
_____ А.С. Гудим	_____М.А. Горькавый
(подпись)	(подпись)
«_04_» _____ 2021 г.	«_04_» _____ 2021 г.

АКТ

о приемке в эксплуатацию управляющей программы для автоматизированной/роботизированной системы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера»

г. Комсомольск-на-Амуре « 04 » июня 2021 г.

Комиссия в составе представителей:

заказчика

- *В.С. Соколова* – руководитель *проекта* ,
- *М.А. Горькавый* – Заведующий кафедрой *УИПП*,
- *А.С. Гудим* – декана ФЭУ

исполнителя

- *Д.Ю. Облогин – 8МРб-1,*

составила акт о нижеследующем:

«Исполнитель» передает управляющую программу для автоматизированной/роботизированной системы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера», в составе:

Программное обеспечение:

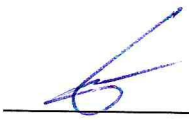
- Текст управляющей программы

Эксплуатационная документация:

- Паспорт управляющей программы для автоматизированной/роботизированной системы

Управляющая программа для автоматизированной/роботизированной системы «Контрольно-измерительный программный комплекс для поддержания температурно-влажностного режима работы абсорбционного кондиционера» прошла апробацию с «01» июня по «04» июня 2021 г. и признана годной к эксплуатации. Были протестированы все режимы функционирования, отказы системы, а также аварийные отключения по вине системы не наблюдались.

Руководитель проекта



_____/ С.И. Сухоруков /

Ответственный исполнитель

_____/ Д.Ю. Облогин /

Таблица учета проектной работы в учебных дисциплинах

Дисциплина	Форма учтенной работы (номер ЛР, КП, КР, РГЗ, зачет, итоговая оценка , экзамен)	Преподаватель (дата, ФИО, подпись)	Примечание (ЗУН полученные при выполнении проекта)