

**Министерство науки и высшего образования**

**Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Институт цифровых интеллектуальных систем

Дисциплина: «Программирование встроенных систем управления»

Лабораторная работа № 6

Радио интерфейс, передача данных в диапазоне 433Мгц Arduino.

Выполнил:

студент группы АДМ-21-05 \_\_\_\_\_\_\_\_\_\_ Абдулзагиров М.М.

(подпись) (ФИО)

Принял

преподаватель: \_\_\_\_\_\_\_\_\_\_ \_Панфилов П. В. \_\_

(подпись) (ФИО)

Дата:\_\_\_\_\_\_\_\_\_

Москва 2022

# Дисплеи, подключение дисплеев к Arduino.

# Задание (Для продвинутых пользователей)

Разработать программу удаленного пульта (интеллектуального датчика) простейшего терморегулятора, со следующими характеристиками:

1. В качестве датчика температуры использовать датчик LM35(A2). Текущая температура Тт. Точность расчетов и хранения температуры не ниже 0.1 градуса.

2. Устройство должно передавать данные по беспроводной сети используя передатчик на основе SYN113/SYN115, подключенного к выходу D7 макета. Устройство только передает данные и не принимает решения о включении нагревательного элемента. Должны передаваться: id — устройства для его идентификации Тт — текущая температура Тц — целевая температура N - номер пакета F — период передачи

3. Установку целевой температуры (Тц) проводить с помощью переменного резистора присоединенного в выходу A0. Точность установки до 0.1 градуса. Диапазон регулирования +5 … +30 градусов.

4. Частота передачи данных задается кнопками D2 (увеличение 1 сек.) и D3 (уменьшение 1 сек.). Диапазон регулирования от 1 до 12 секунд.

5. Для показа состояния регулятора использовать LCD дисплей подключенный к шине i2c тип дисплея 1602. Должны отображаться Тт, Тц, период посылки данных и число посланных пакетов (5 знаков).

6. В момент передачи для индикации загорается красный светодиод D12. Для продвинутых пользователей.

7. Добавить выполнение через консоль следующих команд:

state — вывод состояния устройства. Выводится:

• текущая температура

• целевая температурам

• id устройства

• число посланных пакетов

• период передачи

setTime — ввод значения периода передачи данных. Диапазон регулирования от 1 до 12 секунд. Проверка на валидность введенных данных.

setID — ввод идентификатора устройства. Диапазон 1-100. Проверка на валидность введенных данных.

ON — включить передачу данных

OFF — выключить передачу данных, при этом зажигаем светодиод D13.

help — вывести список команд.

info — информация о разработчике и версии программы, времени и даты сборки прошивки (макросы \_\_TIME\_\_ \_\_DATE\_\_). При старте программы в консоли выводится приглашение и подсказка об использовании команды help.

# Описание программы

В качестве IDE использовалась VS Code с расширением PlatfofmIO.

В основном цикле идет обработка вывода показаний на дисплей SSD1306 (в наличии только он), обработка нажатий кнопки D2 и D3 для увеличения и уменьшения значения периода отправки данных по радиомодулю (проверка программы производилась без радиомодуля из-за неработоспособности модуля 433 МГц), считывание показаний датчика температуры DH11 (LM35 не было в наличии)(D4), обработка принимаемых из сериал порта команд. Дребезг обрабатывается программно с помощью задержки.

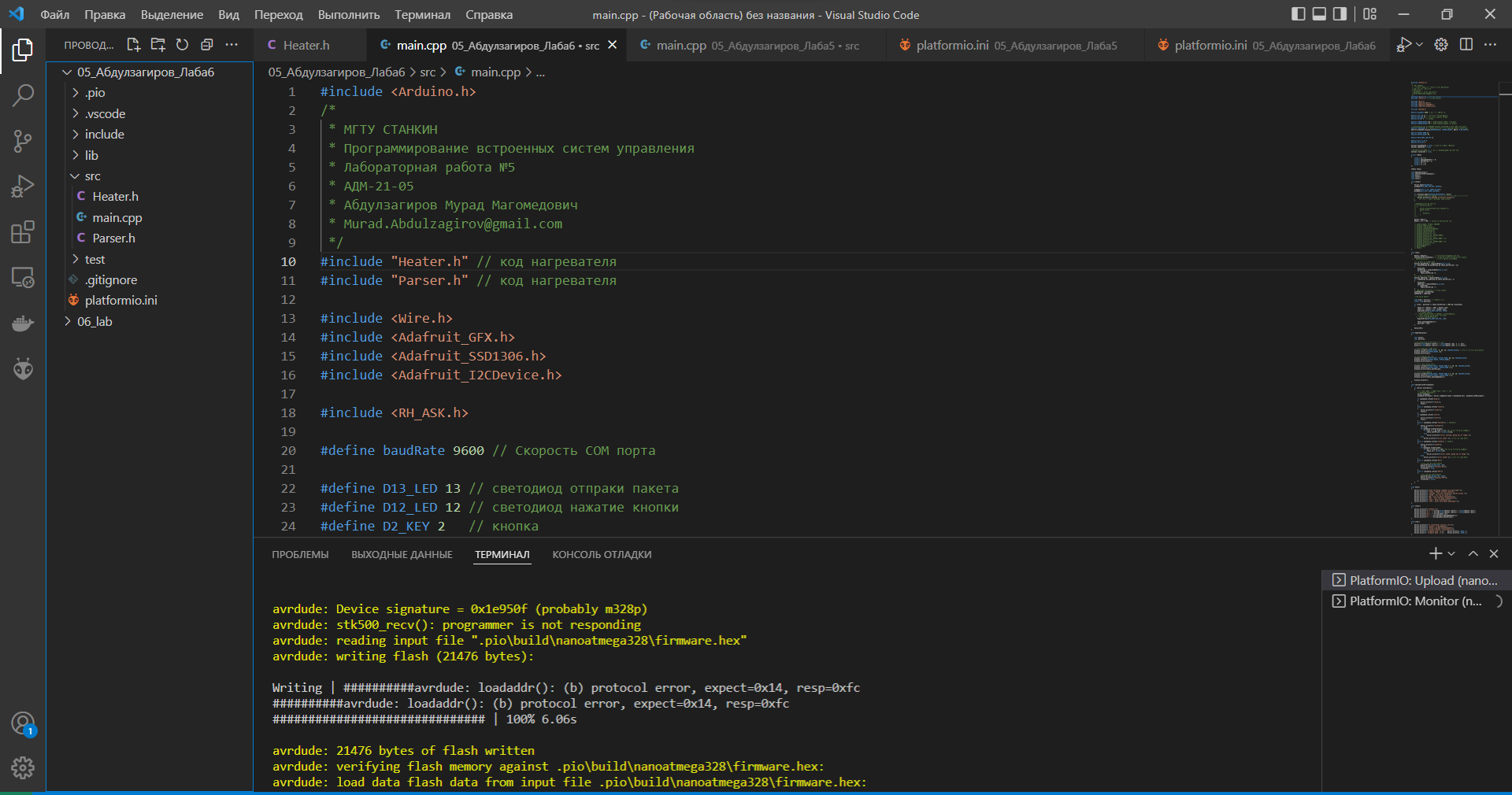


Рис 1. Окно VS Code.

Для добавления библиотеки датчика DHT11, дисплея и библиотеки RH\_ASK воспользуемся интерфейсом PlatformIO.

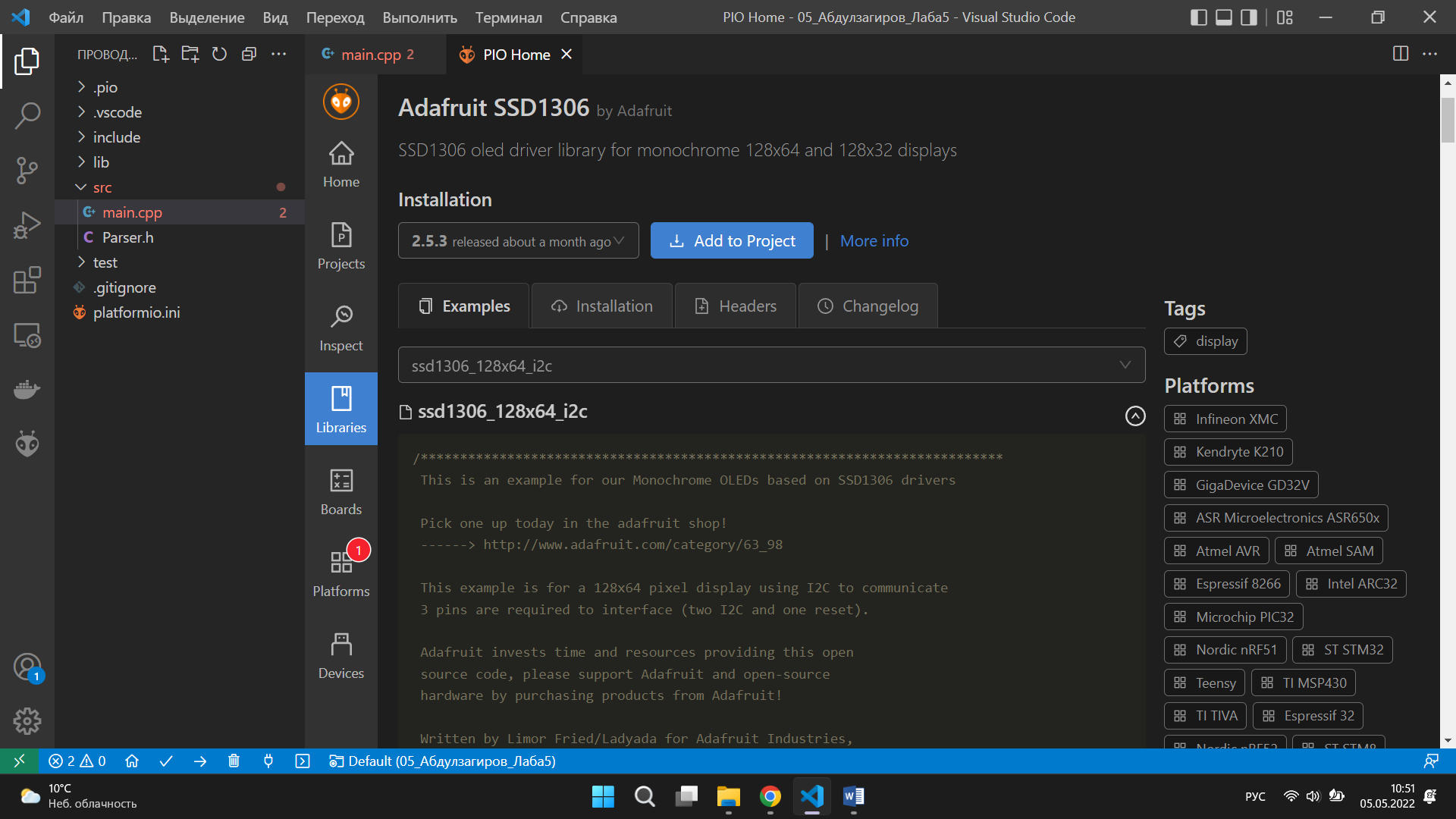


Рис 2. Окно VS Code.

Здесь также можно найти пример программного кода.

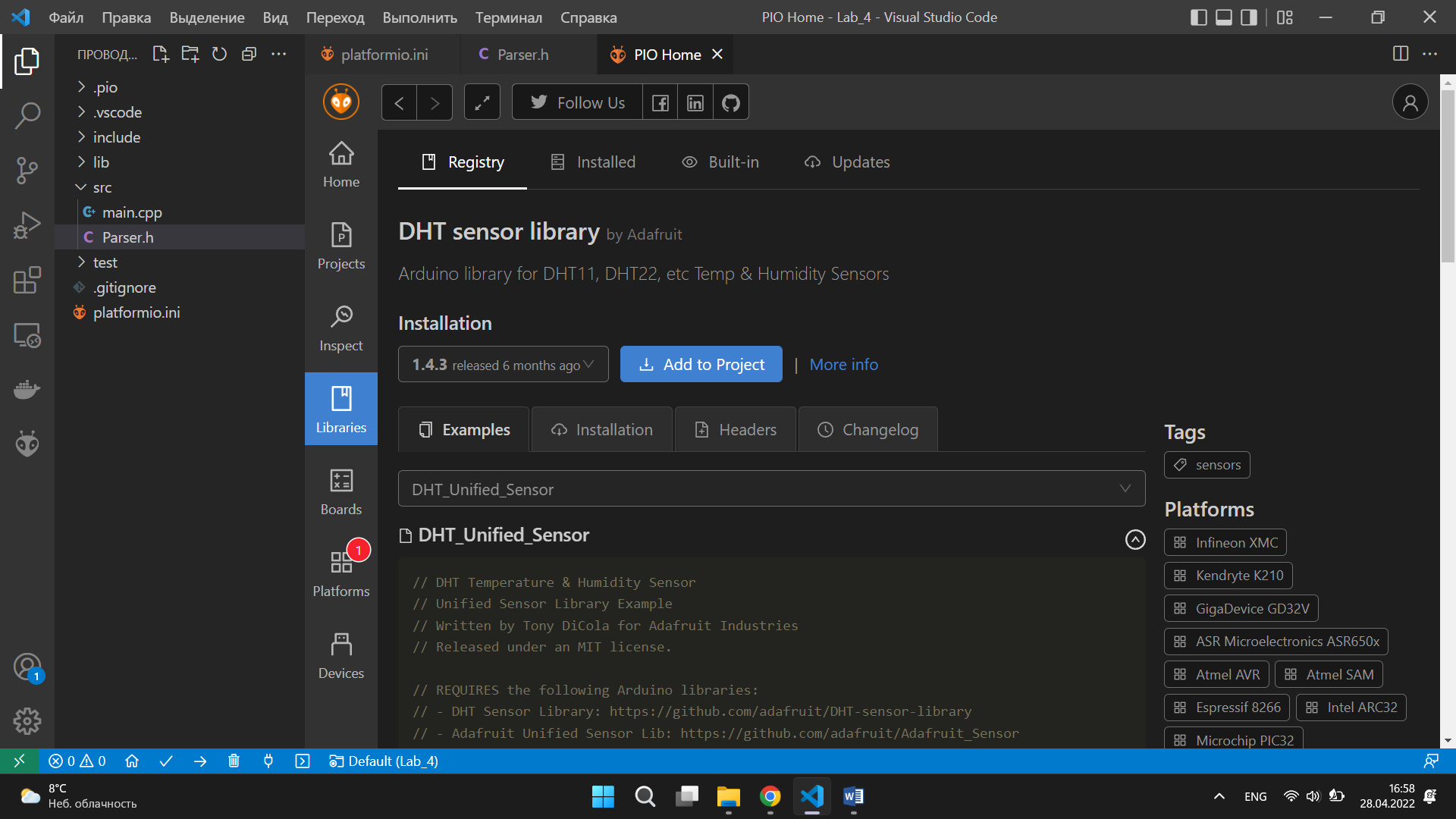


Рис 3. Окно VS Code.

Исходный код программы:

Листинг 1. Файл main.cpp

#include <Arduino.h>

/\*

 \* МГТУ СТАНКИН

 \* Программирование встроенных систем управления

 \* Лабораторная работа №5

 \* АДМ-21-05

 \* Абдулзагиров Мурад Магомедович

 \* Murad.Abdulzagirov@gmail.com

 \*/

#include "Heater.h" // код нагревателя

#include "Parser.h" // код нагревателя

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include <Adafruit\_I2CDevice.h>

#include <RH\_ASK.h>

#define baudRate 9600 // Скорость COM порта

#define D13\_LED 13 // светодиод отпраки пакета

#define D12\_LED 12 // светодиод нажатие кнопки

#define D2\_KEY 2   // кнопка

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

#define OLED\_RESET 2 // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

#define OFFSET\_DOWN 16

#define OFFSET\_RIGHT 50

#define DATA\_SENT\_LED\_PIN 12

#define down\_tc\_bt 2

#define up\_tc\_bt 3

boolean downBtWasUp = true; // была ли кнопка отпущена?

boolean upBtWasUp = true;

RH\_ASK driver(2000, 8, 7, 0); // Инициализация передатчика

boolean transminOn = true;

struct TxData

{

    uint8\_t id = 5;

    uint8\_t pakskageNumbers = 0;

    uint8\_t SentPeriod = 4;

    uint8\_t Tt = 0;

    uint8\_t Tc = 0;

};

TxData tData;

void UpdateDisplay();

void CheckSerialPortCommand();

void help();

void state();

void info();

void setup()

{

    Serial.begin(baudRate);

    pinMode(DATA\_SENT\_LED\_PIN, OUTPUT);

    pinMode(down\_tc\_bt, INPUT\_PULLUP);

    pinMode(up\_tc\_bt, INPUT\_PULLUP);

    if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C))

    { // Address 0x3C for 128x64 !!!!!!!!!!!!!!!!!!!!! Aдрес !!!!!!!!!!!

        Serial.println(F("SSD1306 allocation failed"));

        for (;;)

            ; // Don't proceed, loop forever

    }

    //Инициализируем передатчик

    if (!driver.init())

    {

        Serial.println(F("RF init failed!"));

        while (true)

        {

            delay(1);

        }

    }

    Heater::Begin();

    Heater::\_Tc = 290; // начальное значение порога

    // инициялизация дисплея SSD1306

    display.cp437(true);

    display.clearDisplay();

    display.setTextColor(WHITE);

    display.setTextSize(2);

    display.setCursor(0, 0);

    display.println("Tt =");

    display.setCursor(0, OFFSET\_DOWN);

    display.println("Tc =");

    display.setCursor(0, OFFSET\_DOWN \* 2);

    display.println("F =");

    display.setCursor(0, OFFSET\_DOWN \* 3);

    display.println("N =");

    display.display();

    delay(2000);

    help();

}

void loop()

{

    Heater::Update();         // обновление показаний датчика

    CheckSerialPortCommand(); // обработка команд из Serial интерфейса

    UpdateDisplay();          // вывод данных на дисплей

    /// обработчик кнопки down

    boolean downBtIsUp = !digitalRead(down\_tc\_bt);

    if (!downBtWasUp && downBtIsUp && tData.SentPeriod < 12)

    {

        delay(10);

        downBtIsUp = !digitalRead(down\_tc\_bt);

        if (downBtIsUp)

            tData.SentPeriod++;

    }

    /// обработчик кнопки up

    boolean upBtIsUp = !digitalRead(up\_tc\_bt);

    if (!upBtWasUp && upBtIsUp && tData.SentPeriod > 1)

    {

        delay(10);

        upBtIsUp = !digitalRead(up\_tc\_bt);

        if (upBtIsUp)

            tData.SentPeriod--;

    }

    // запоминаем последнее состояние кнопки

    downBtWasUp = downBtIsUp;

    upBtWasUp = upBtIsUp;

    // передача данных

    long time = millis(); // текущее время

    static long pastTime;

    if (time - pastTime >= tData.SentPeriod \* 1000 && transminOn)

    {

        tData.Tt = Heater::\_Tdht / Heater::dev;

        tData.Tc = Heater::\_Tc / Heater::dev;

        digitalWrite(DATA\_SENT\_LED\_PIN, HIGH);

        // Передаём данные

        driver.send((uint8\_t \*)&tData, sizeof(tData));

        // Ждем пока передача будет окончена

        driver.waitPacketSent();

        digitalWrite(DATA\_SENT\_LED\_PIN, LOW);

        tData.pakskageNumbers++;

        pastTime = time;

    }

    delay(100);

}

void UpdateDisplay()

{

    char buf[6];

    char bufTc[6];

    // преобразовние из значений в строки

    dtostrf((float)Heater::GetTt() / (float)Heater::dev, 4, 2, buf);

    dtostrf((float)Heater::GetTc() / (float)Heater::dev, 4, 2, bufTc);

    // вывод значения температуры

    display.fillRect(OFFSET\_RIGHT, 0, 80, 20, SSD1306\_BLACK); // Стереть прошлое изображение

    display.setCursor(OFFSET\_RIGHT, 0);

    display.println(buf);

    // вывод значения порога

    display.fillRect(OFFSET\_RIGHT, OFFSET\_DOWN, 80, 20, SSD1306\_BLACK);

    display.setCursor(OFFSET\_RIGHT, OFFSET\_DOWN);

    display.println(bufTc);

    // вывод  периода передачи

    display.fillRect(OFFSET\_RIGHT, OFFSET\_DOWN \* 2, 40, 20, SSD1306\_BLACK);

    display.setCursor(OFFSET\_RIGHT, OFFSET\_DOWN \* 2);

    display.println(tData.SentPeriod);

    // вывод номера пакета

    display.fillRect(OFFSET\_RIGHT, OFFSET\_DOWN \* 3, 40, 20, SSD1306\_BLACK);

    display.setCursor(OFFSET\_RIGHT, OFFSET\_DOWN \* 3);

    display.println(tData.pakskageNumbers);

    display.display();

}

void CheckSerialPortCommand()

{

    if (Serial.available())

    {

        //отчищаем буфер и записываем в него строку

        // pCommand.bufClean();

        Parser pCommand;

        pCommand.bufLength = Serial.readBytes((byte \*)(pCommand.buf), pCommand.bufMaxLength);

        if (pCommand.isFind("help"))

        {

            Serial.println(F("help"));

            help();

        }

        else if (pCommand.isFind("state"))

        {

            Serial.println(F("state"));

            state();

        }

        if (pCommand.isFind("info"))

        {

            Serial.println(F("info"));

            info();

        }

        else if (pCommand.isFind("setTime")) // setTime 4

        {

            Serial.println(F("setTime"));

            int Time;

            if (pCommand.findInt(&Time))

                if (Time >= 1 && Time <= 12) //проверка выхода за диапазон

                    tData.SentPeriod = (uint8\_t)Time;

                else

                    Serial.println(F("error setTime: going out of range "));

            else

                Serial.println(F("error setGT ")); //число не обнаружено

        }

        else if (pCommand.isFind("setID")) // setID 2

        {

            Serial.println(F("setID"));

            int ID;

            if (pCommand.findInt(&ID))

                if (ID >= 0 && ID <= 100) //проверка выхода за диапазон

                    tData.id = (uint8\_t)ID;

                else

                    Serial.println(F("error setID: going out of range "));

            else

                Serial.println(F("error setID ")); //число не обнаружено

        }

        else if (pCommand.isFind("ON"))

        {

            //выключаем передачу данных

            digitalWrite(LED\_BUILTIN, LOW);

            Serial.println(F("Transmit ON"));

            transminOn = true;

        }

        else if (pCommand.isFind("OFF"))

        {

            //включаем передачу данных

            digitalWrite(LED\_BUILTIN, HIGH);

            Serial.println(F("Transmit OFF"));

            transminOn = false;

        }

    }

}

void help()

{

    Serial.println(F("\nThe following commands are described:"));

    Serial.println(F(" state - Heater current state"));

    Serial.println(F(" setTime - set the transmition period values "));

    Serial.println(F(" setID - set the id value"));

    Serial.println(F(" ON - turn on data transmetion"));

    Serial.println(F(" OFF - turn off data transmetion"));

    Serial.println(F(" help - help information"));

    Serial.println(F(" info - print info about developers"));

}

void state()

{

    Serial.println(F("\nState:"));

    Serial.println("Tt = " + String((float)Heater::GetTt() / (float)Heater::dev));

    Serial.println("Tc = " + String((float)Heater::GetTc() / (float)Heater::dev));

    Serial.println("id = " + String(tData.id));

    Serial.println("N = " + String(tData.pakskageNumbers));

    Serial.println("F = " + String(tData.SentPeriod));

}

void info()

{

    Serial.println(F("\n interpreter version 1.0.4"));

    Serial.println(F(" author: \t Murad255"));

    Serial.println(F(" https://github.com/Murad255"));

    Serial.println(F(" bitcoin wallet: \t\*\*\*\*\*\*\*\*\*\*"));

    Serial.print(F(" firmware time: \t"));

    Serial.println(\_\_DATE\_\_);

    Serial.print(F(" firmware data: \t"));

    Serial.println(\_\_TIME\_\_);

}

Листинг 2. Файл Heater.h

#pragma once

#include <Arduino.h>

#include <DHT.h>

#include <Adafruit\_Sensor.h> // требуется для бибилиотеки DHT

#define DHTPIN 4

//#define heater\_out 12

#define work\_status\_led 8

#define setings\_pot A0

// пространство имён нагревателя (вместо синглетон класса)

namespace Heater

{

    DHT dht(DHTPIN, DHT11);

    const int dev = 10; // точность значений - 1 числа после запятой

    int \_Tdht = 0;      // показания датчика температуры

    int \_Hdht = 0;      // показания датчика влажности

    int \_Tc = 0;

    const int Tmin = 10 \* dev;

    const int Tmax = 30 \* dev;

    int GT = 2 \* dev;

    int ERR = 0;

    boolean isWork = 1;   // статус работы устройства

    boolean isHeated = 0; // статус работы нагревателя

    boolean isError = 0;  // true, если есть ошибки в работе

    void Begin()

    {

        //pinMode(heater\_out, OUTPUT);

        pinMode(work\_status\_led, OUTPUT);

        digitalWrite(work\_status\_led, 1);

        pinMode(setings\_pot, INPUT);

        dht.begin();

    }

    // увеличить Tc на 1 градус

    void UpTc()

    {

        if ((\_Tc+dev)<=Tmax)

        \_Tc =\_Tc+dev;

    }

    // уменьшить Tc на 1 градус

    void DownTc()

    {

        if ((\_Tc-dev)>=Tmin)

        \_Tc =\_Tc-dev;

    }

    int GetTt() { return \_Tdht; }

    int GetTc() { return \_Tc; }

    // установить статус работы нагревателя

    void SetWorkStatus(boolean);

    // установить статус ошибки

    void SetErrorStatus(boolean status);

    // включить или выключить нагрев

    void Switch(boolean);

    void Update()

    {

        \_Tc = map(analogRead(setings\_pot), 0, 1023, Tmin, Tmax);

        float h = dht.readHumidity();    //Измеряем влажность

        float t = dht.readTemperature(); //Измеряем температуру

        if (isnan(h) || isnan(t))

        { //Проверка. Если не удается считать показания,выводится «Ошибка считывания», и программа завершает работу

            Serial.println("Error read DHT11");

            SetErrorStatus(true);

        }

        else

        {

            SetErrorStatus(false);

            \_Tdht = t \* dev;

            \_Hdht = h \* dev;

        }

    }

}

// установить статус работы нагревателя (если setStatus == false, но не менять значение статуса)

void Heater::SetWorkStatus(boolean status)

{

    Heater::isWork = status;

    digitalWrite(work\_status\_led, status);

    Heater::Switch(Heater::isHeated);

}

// включить или выключить нагрев

void Heater::Switch(boolean heat)

{

    //нагрев регулируется только при включённом устройстве

    //digitalWrite(heater\_out, heat && Heater::isWork);

    Heater::isHeated = heat && Heater::isWork;

}

// выключаем устройство, если есть ошибка

void Heater::SetErrorStatus(boolean status)

{

    static boolean pastIsWork;

    if (status != Heater::isError)

    {

        if (status)

        {

            // запоминаем предыдущее состояние

            pastIsWork = Heater::isWork;

            Heater::SetWorkStatus(false);

        }

        else

        {

            Heater::SetWorkStatus(pastIsWork);

        }

    }

    Heater::isError = status;

}

Листинг 2. Parser.h

#pragma once

#include <Arduino.h>

/\*

    Класс для поиска и  возврата значения после ключевого слова.

    Для поиска требуется в буфер buf записать строку и

    в bufLength указать её длину, и затем вызвать нужный метод.

\*/

class Parser

{

private:

    char findBuf[100];

public:

    static const int bufMaxLength = 100; // максимальное число символов

    char buf[bufMaxLength];              // буфер строки поиска

    int bufLength;                       // длина заданной строки

    int findSymbol(const char findContext); // возвращает индекс искомого символа, если не найден, то -1

    int findStr(const char \*findContext);   // возвращает индекс искомой строки, если не найден, то -1

    bool isFind(const char \*findContext);   // возвращает true, если искомая строка найдена, иначе false

    bool findInt(int \*result);              // считывает значение int от -32760 до 32760

//  bool findFloat(float \*result);          // считывает значение float до 2х знаков после запятой

    bool findFloatHowInt(int \*result);      // считывание значения float с возвратом его в виде int переменной со сдвигом на 1 число

//  bool findStr8(char \*findStr8);          // считывает первые 8 символов после команды setStr

    void bufClean();                        //отчистка буфера

};

///отчистка буфера

void Parser::bufClean()

{

    //посимвольно зануляем символы в буфере

    for (int i = 0; i < bufMaxLength; i++)

        buf[i] = 0;

}

/\* поиск  символа findContext в строке

 \* возвращает индекс найденого символа, иначе -1 \*/

int Parser::findSymbol(const char findContext)

{

    //проверяем каждый символ

    for (int i = 0; i < bufLength; i++)

    {

        // при нахождении возвращаем индекс

        if (findContext == Parser::buf[i])

            return i;

    }

    return -1;

}

/\* поиск строки в буфере

 \* аналог indexOf, возвращает -1, если строка не найдена,

 \* или индекс первого символа найденной строки \*/

int Parser::findStr(const char \*findContext)

{

    // проверяем буфер до

    for (int i = 0; i < bufLength - (int)strlen(findContext) + 1; i++)

    {

        // копируем равную по длинне с искомой строку из буфера

        // со сдвигом на i символов во временный буфер

        strncpy(findBuf, &buf[i], (int)strlen(findContext));

        //обозначаем конец

        findBuf[(int)strlen(findContext)] = 0;

        // если строки равны, то возвращаем её индекс в буфере

        if (strcmp(findContext, findBuf) == 0)

            return i;

    }

    return -1; // если не найдена

}

///возвращает true если строка найдена

bool Parser::isFind(const char \*findContext)

{

    if (findStr(findContext) >= 0)

        return true;

    else

        return false;

}

/\* считывает значение int от -32760 до 32760

 \* если найдено значение, то возвращиет true

 \* результат возвращает через ссылку result \*/

bool Parser::findInt(int \*result)

{

    int mult = 1;         //множитель для отрицательного числа

    const int naxNum = 6; //предел для int16

    // char \*strNum[naxNum];

    int start;   // начальный индекс

    int num = 0; // индекс записи

    // начинаем поиск с пробела или знака -

    if ((start = findSymbol('-')) >= 0)

    {

        mult = -1; // полученное число в конце сделаем отрицательным

        ++start;

    }

    else

    {

        start = findSymbol(' ') + 1;

    }

    int i = 0;

    //продолжаем поиск со старта до конца буфера

    while (buf[start + i] != 0) //(( i < naxNum) &&(buf[start + i]!=0))

    {

        // определяем, число ли это

        int n = buf[start + i] - '0';

        if (n >= 0 && n <= 9)

        {

            if (num >= 3276) // проверка на выход за пределы int16

                return 0;

            num \*= 10; // десятичный сдвиг влево текущего значения

            num += n;  // в конец прибавляем число

        }

        //если следующий символ не пробел (его пропускаем), то выходим из цикла

        else if (buf[start + i] != ' ')

            break;

        i++;

    }

    if (!i)

        return 0;  // если не было цифр

    num \*= mult;   // при необходимости делаем отрицательным

    \*result = num; // присваиваем ссылку на число

    return 1;

}

/\* считывает значение float до 1 знаков после запятой

 \* с возвратом его в виде int переменной со сдвигом на 1 число (регулируется до 2)

 \* если найдено значение, то возвращиет true

 \* результат возвращает через ссылку result \*/

bool Parser::findFloatHowInt(int \*result)

{

    int mult = 1;

    const int naxNum = 6;

    char \*strNum[naxNum];

    int start;

    int num1 = 0; // число до запятой

    int num2 = 0; // число после запятой

    int fNum = 0;

    // начинаем поиск с пробела или знака -

    if ((start = findSymbol('-')) >= 0)

    {

        mult = -1;

        ++start;

    }

    else

    {

        start = findSymbol(' ') + 1;

    }

    //находим целое число

    int i = 0;

    while (buf[start + i] != 0)

    {

        int n = buf[start + i] - '0';

        if (n >= 0 && n <= 9)

        {

            if (num1 >= 3276)

                return 0;

            num1 \*= 10;

            num1 += n;

        }

        // выходим из цикла при достижении запятой, точки или пробела

        else if (

            buf[start + i] != ' ' ||

            buf[start + i] != '.' ||

            buf[start + i] != ',')

        {

            break;

        }

        i++;

    }

    // находим дробь

    // если найдена точка иля запятая, отмечаем начало дробной части

    if ((start = findSymbol('.')) >= 0)

        ++start;

    else if ((start = findSymbol(',')) >= 0)

        ++start;

    // если не найдена точка или запятая, возвращаем найденное число

    else

    {

        num1 \*= mult;

        \*result = num1;

        return 1;

    }

    i = 0;

    int dev = 1; // делитель дробной части

    const int accuracy = 10;

    while ((i < naxNum) && (buf[start + i] != 0))

    {

        int n = buf[start + i] - '0';

        // для увеличения точности поставить dev<=100

        if (n >= 0 && n <= 9 && dev<=accuracy)

        {

            dev \*= 10;

            num2 \*= 10;

            num2 += n;

        }

        else if (buf[start + i] != ' ')

            break;

        i++;

    }

    // делим целое число на делитель для получения дробной части

    //num2 /= dev;

    fNum = num1\*accuracy + num2\*accuracy/dev; // складываем целую и дробную части

    fNum \*= mult;

    \*result = fNum;

    Serial.println("RGB led state: red");

    return 1;

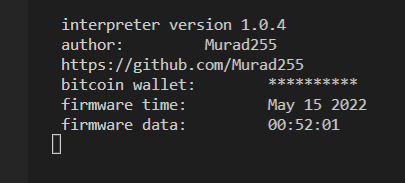
}

# Результаты выполнения программы

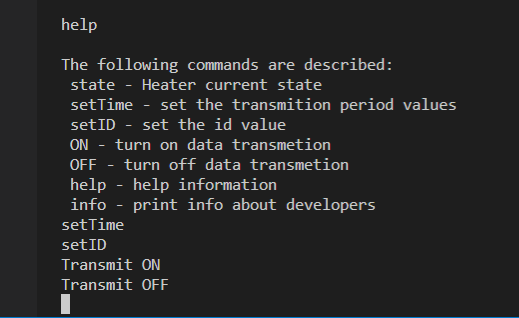
Протестируем программу на отладочной плате arduino nano с микроконтроллером atMega 328P (из-за сбоев при прошивке не удалось проверить некоторые модули).

При вращении потенциометра изменяется значение Tc. При нажатии на кнопки уменьшается и увеличивается знаяение периода F, это также отображается на дисплее.

При отправке команды info выводится информация о прошивке и разработчике.



При отправке команды help выводится информация о командах. При отправке команды setID 2 устанавливается ID номер устройства.



При отправке команды setTime 5 устанавливается значение периода, равное 5. При отправке команд ON и OFF соответственно включается и выключается отправка данных через радиомодуль, при этом при отключении отправки загарается светодиод D13.

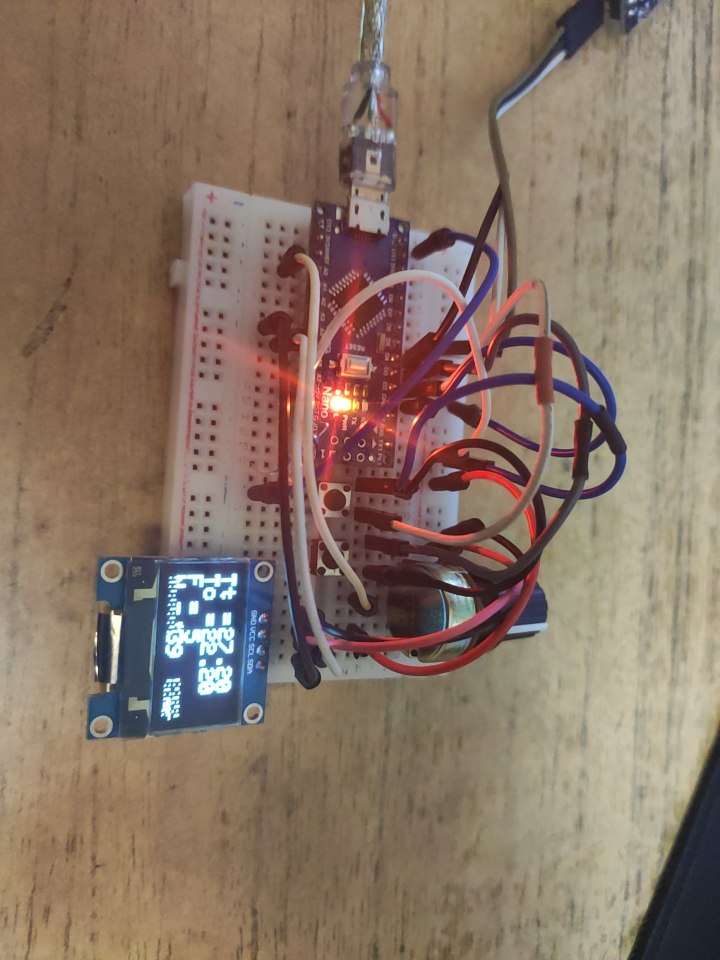


Рис 4. Отладочная плата.

Тест работы устройства был записан на видео:

Яндекс диск

https://disk.yandex.ru/i/vmcUdYqiB0OgTg

Гугл диск

https://drive.google.com/file/d/18ZWeYalAGM3ht-X1PBbzy5fWsc12A0Km/view?usp=sharing