

**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 № 5

Дисплеи, подключение дисплеев к 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 не продвинутых пользователей)

# Разработать программу простейшего терморегулятора, со следующими

# характеристиками:

# 1. В качестве датчика температуры использовать датчик DHT11(D4). Текущая температура Тт. Точность расчетов и хранения температуры не ниже 0.1 градуса.

# 2. Установку целевой температуры (Тц) проводить с помощью двух кнопок D2-увеличение D3-уменьшение температуры. Шаг 1 градус. Диапазон регулирования +10 … +30

# градусов.

# 3. Считать нагревательным элементом светодиод красного цвета (D12).

# 4. Терморегулятор должен иметь гистерезис (Гс), при этом температура включения нагревательного элемента равна целевая температура минус гистерезис (вкл=Тц-Гс), а выключения — целевая температура (выкл=Тц). Гистерезис (Гс) по умолчанию 2 градуса

# 5. Для показа состояния регулятора использовать LCD дисплей подключенный к шине i2c тип дисплея 1602. Должны отображаться Тт, Тц, Гс и включение нагрева (D12).

# 6. При переходе из одного состояния в другое (вкл-вкл и выкл-вкл D12) подавать короткий (50 мсек.) звуковой сигнал биппером D5

# 7. При Тт<Тц-Гс должен мигать светодиод D13 с частотой 2 гц

# Описание программы

В качестве IDE использовалась VS Code с расширением PlatfofmIO.

Устройство включает нагреватель (красный светодиод) при значении температуры, меньше регулируемой, и выключает его при достижении нужной температуры (с учётом гистерезиса). При ошибке датчика устройство выключается, пока ошибка не исчезнет.

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 пороговой температуры с шагом в 1 градус, считывание показаний датчика температуры и влажности DH11 (D4) Дребезг обрабатывается программно с помощью задержки. При включенном нагревателе (D12) также включается мигание светодиода (D13) с частотой 2 Герц (в цикле yield). При переключении состояния (с ВКЛ на ВЫКЛ и обратно) подаётся звуковой сигнал биппером (D5 методом tone).

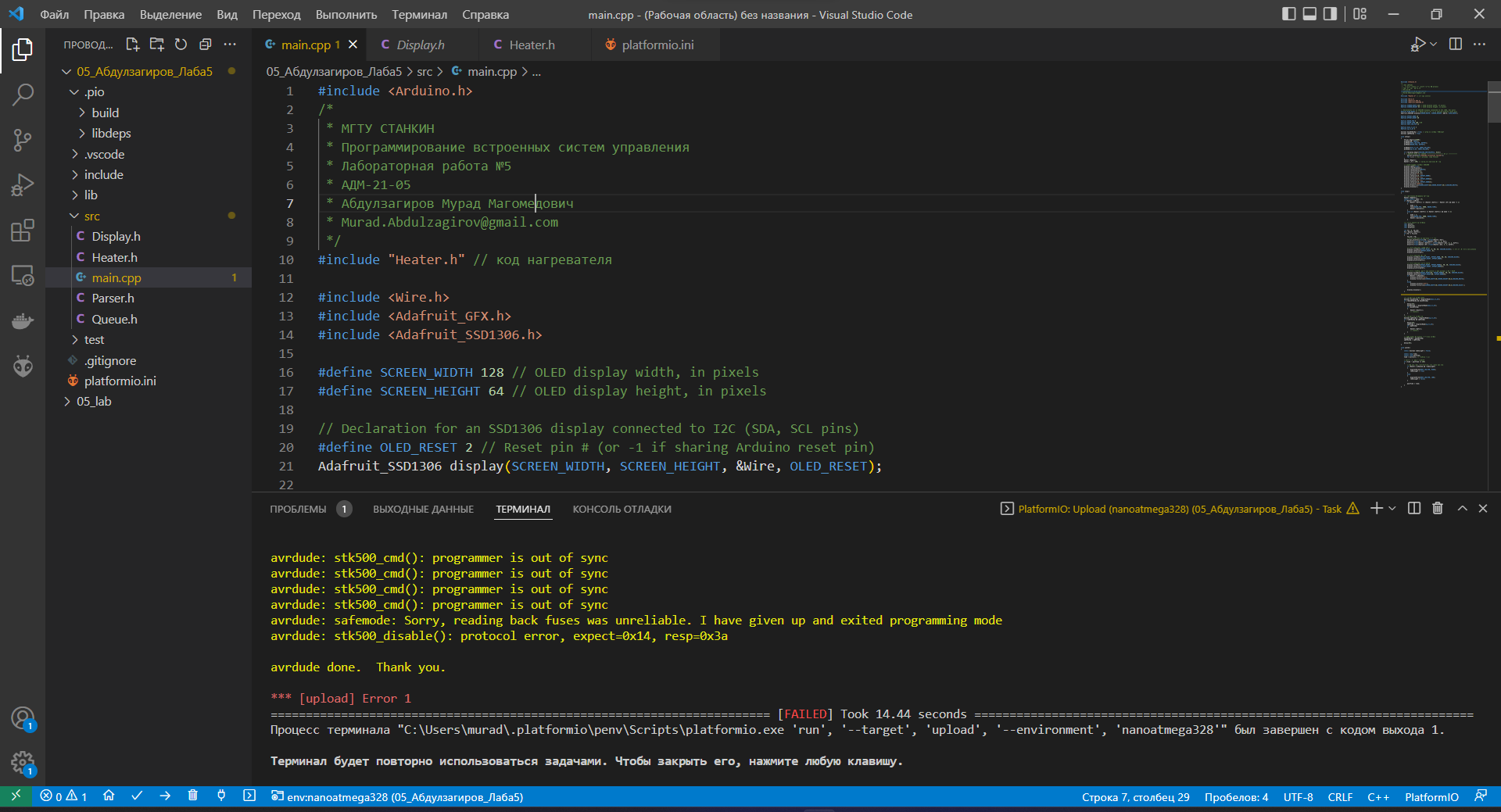


Рис 1. Окно VS Code.

Для добавления библиотеки датчика DHT11 и дисплея воспользуемся интерфейсом PlatformIO.

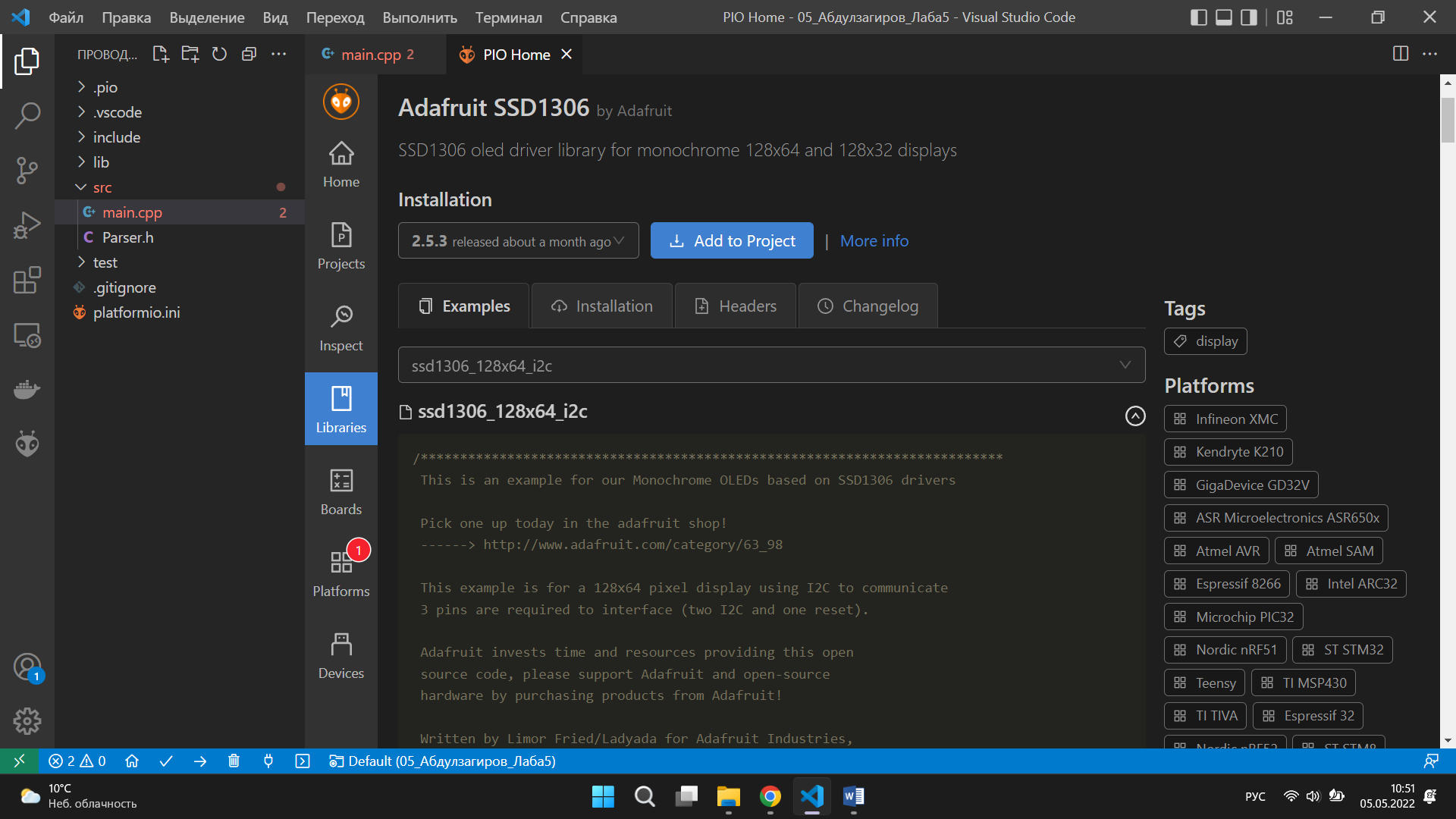


Рис 2. Окно VS Code.

Здесь также можно найти пример программного кода.

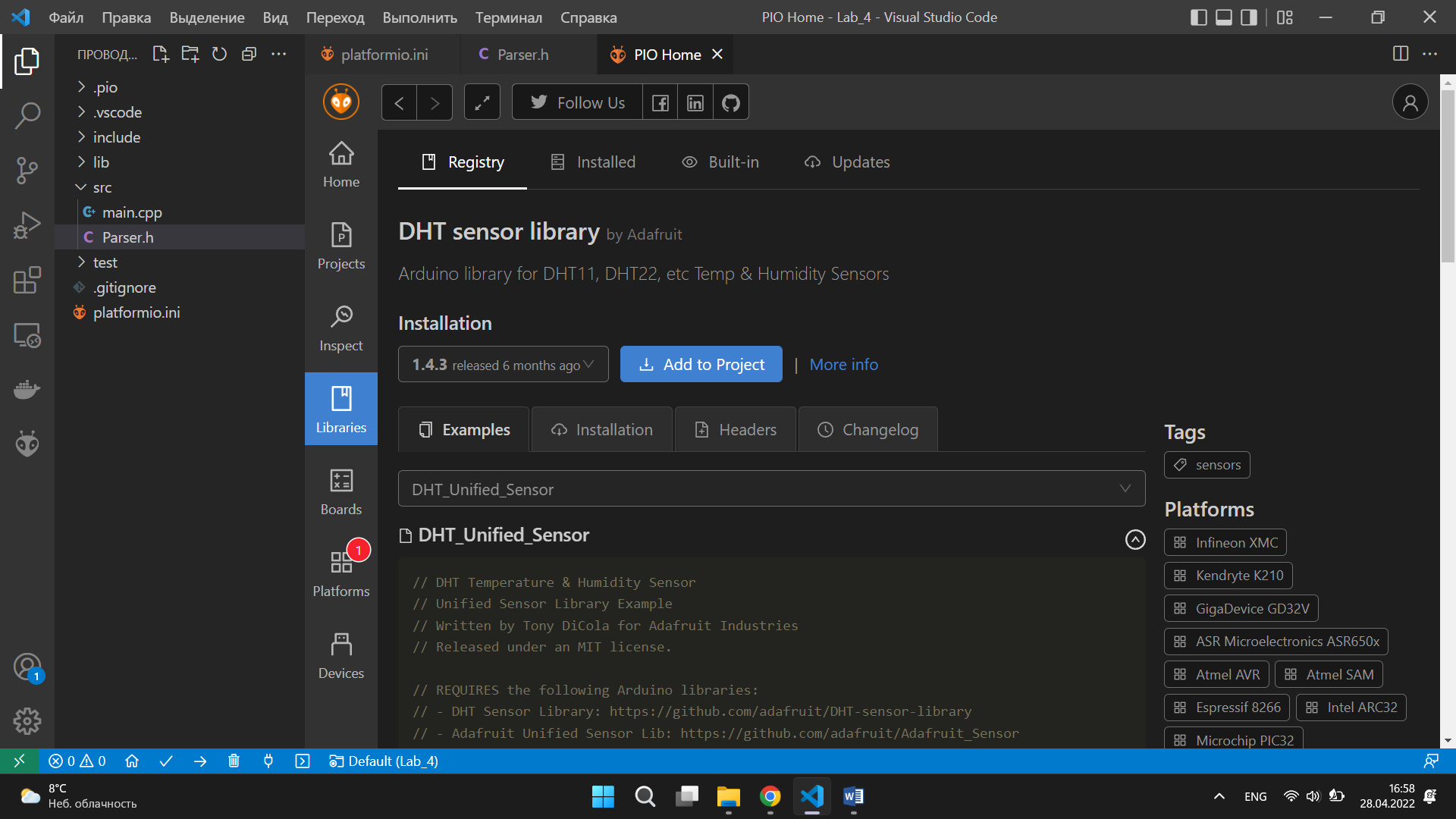


Рис 3. Окно VS Code.

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 <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#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 BUZER\_PIN 5

#define BUZER\_TIME 100 //50

#define HEAT\_LED\_PIN 13

#define down\_tc\_bt 2

#define up\_tc\_bt 3

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

boolean upBtWasUp = true;

void setup()

{

    Serial.begin(115200);

    pinMode(A0, INPUT);

    pinMode(HEAT\_LED\_PIN, OUTPUT);

    pinMode(BUZER\_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

    }

    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("GT =");

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

    display.println("Heat");

    display.drawCircle(SCREEN\_WIDTH-10,SCREEN\_HEIGHT-10,7,SSD1306\_WHITE);

    display.display();

}

void loop()

{

    /// обновление показаний датчика

    Heater::Update();

    static uint8\_t mode = 0;

    if (Heater::isWork)

        if (Heater::GetTt() <= (Heater::GetTc() - Heater::GT) && mode != 1)

        {

            mode = 1;

            tone(BUZER\_PIN, 5000, BUZER\_TIME);

            Heater::Switch(1);

        }

        else if (Heater::GetTt() >= Heater::GetTc() && mode != 2)

        {

            mode = 2;

            tone(BUZER\_PIN, 5000, BUZER\_TIME);

            Heater::Switch(0);

        }

    /// вывод данных на дисплей

    char buf[6];

    char bufTc[6];

    char bufGT[6];

    int Td = 0, Td\_old;

    Td = Heater::GetTt();

    if (Td != Td\_old)

    {

        Td\_old = Td;

        // преобразовние из значений в строки

        Serial.println((float)Td / (float)Heater::dev);

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

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

        dtostrf((float)Heater::GT/ (float)Heater::dev, 4, 2, bufGT);

        // 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 значения гистерезиса

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

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

        display.println(bufGT);

        // вывод статуса работы нагревателя и закрашиваем круг в углу

        display.fillRect(OFFSET\_RIGHT+10, OFFSET\_DOWN\*3, 40, 20, SSD1306\_BLACK);

        display.setCursor(OFFSET\_RIGHT+10, OFFSET\_DOWN\*3);

        if (Heater::isHeated){

            display.println("on");

            display.fillCircle(SCREEN\_WIDTH-10,SCREEN\_HEIGHT-10,6,SSD1306\_WHITE);

        }

        else{

            display.println("off");

            display.fillCircle(SCREEN\_WIDTH-10,SCREEN\_HEIGHT-10,6,SSD1306\_BLACK );

        }

        display.display();

    }

    /// обработчик кнопки down

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

    if (!downBtWasUp && downBtIsUp)

    {

        delay(10);

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

        if (downBtIsUp)

        {

            Heater::DownTc();

            // state();

        }

    }

    /// обработчик кнопки up

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

    if (!upBtWasUp && upBtIsUp)

    {

        delay(10);

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

        if (upBtIsUp)

        {

            Heater::UpTc();

            // state();

        }

    }

    // запоминаем последнее состояние кнопки

    downBtWasUp = downBtIsUp;

    upBtWasUp = upBtIsUp;

    delay(50);

}

void yield()

{

    static boolean ledIsLight = false;

    static long time;

    static long pastTime;

    time = millis(); // текущее время

    // повтор 2 раза в секунду

    if (time - pastTime >= 250)

    {

        // переключаем светодиод при включении нагрева

        if (Heater::isHeated && !ledIsLight)

        {

            digitalWrite(HEAT\_LED\_PIN, HIGH);

            ledIsLight = true;

        }

        else

        {

            digitalWrite(HEAT\_LED\_PIN, LOW);

            ledIsLight = false;

        }

        pastTime = 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

// 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);

        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()

    {

        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;

}

# Результаты выполнения программы

Протестируем программу на отладочной плате arduino nano с микроконтроллером atMega 328P

При уменьшении температуры включается регулятор (красный светодиод) и далее при увеличении температуры регулятор выключается.

Тест работы устройства был записан на видео:

Яндекс диск

<https://disk.yandex.ru/i/pvangdTQRzR9lg>

Гугл диск

https://drive.google.com/file/d/17QRQLkY\_Q-LdZtRgxjkAysquQY-XvA0K/view?usp=sharing

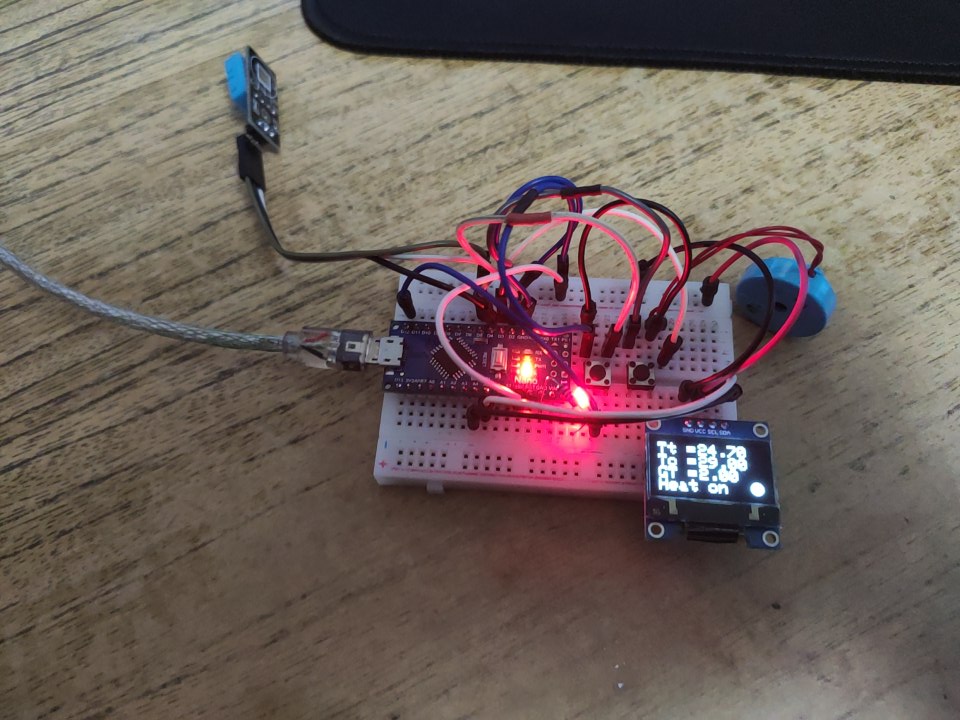


Рис 4. Отладочная плата.