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).

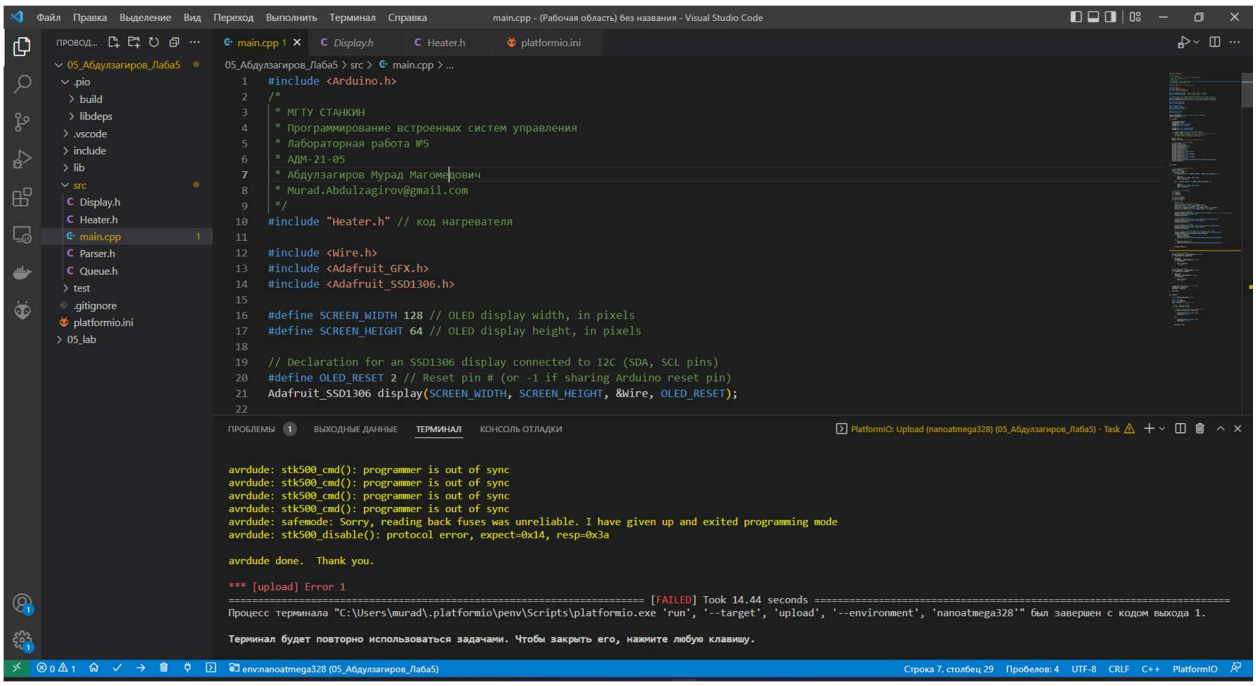

Рис 1. Окно VS Code.

Для добавления библиотеки датчика DHT11 и дисплея воспользуемся интерфейсом PlatformIO.

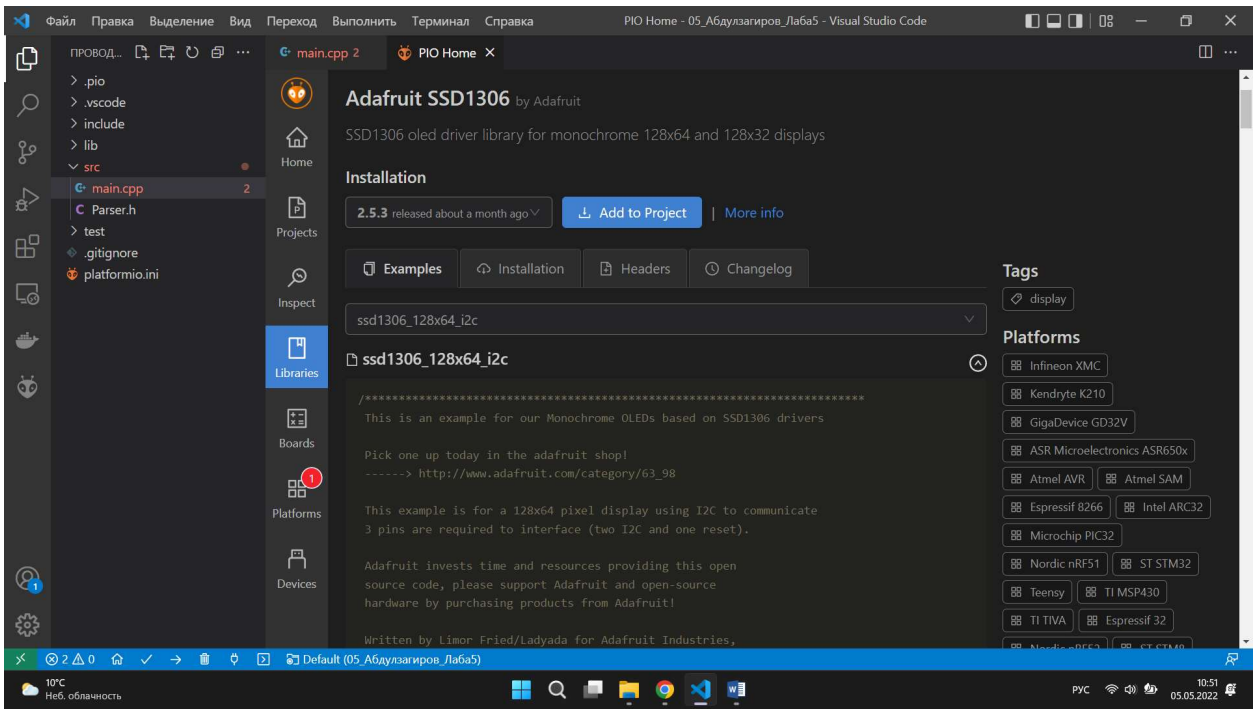

Рис 2. Окно VS Code.

Здесь также можно найти пример программного кода.

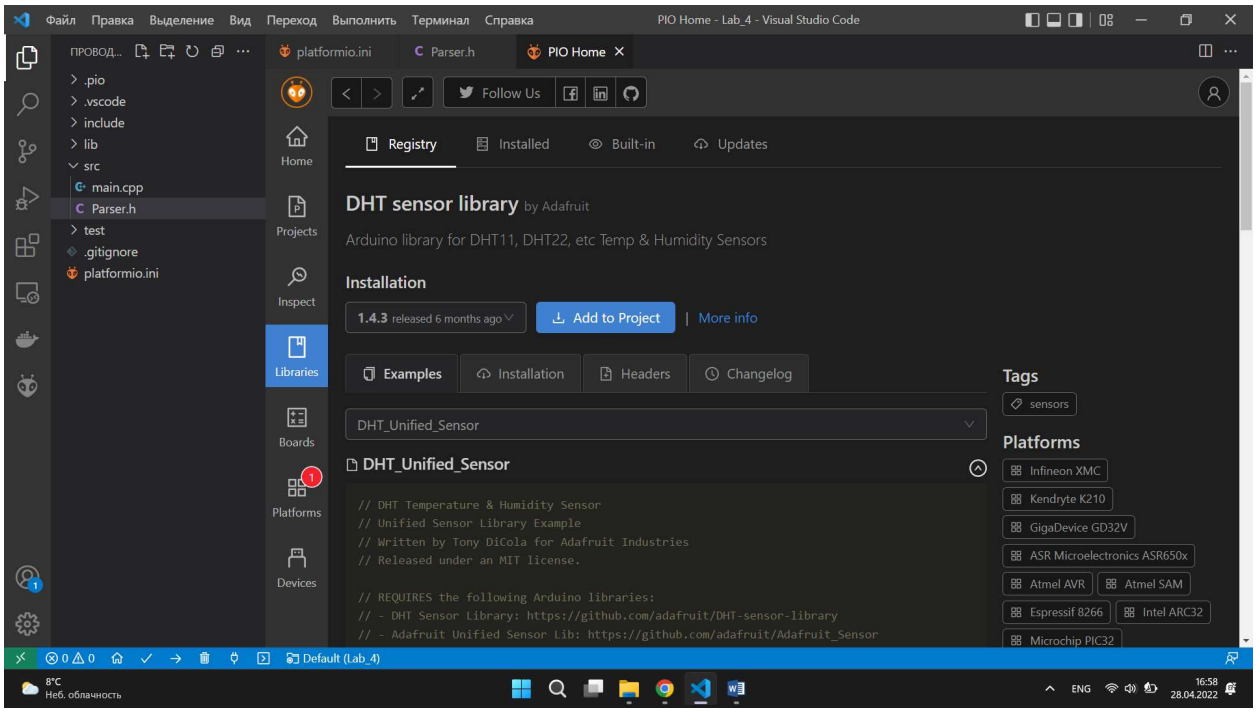

Рис 3. Окно VS Code.

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 !!!!!!!!!!!!!!!!!!!! Адрес !!!!!!!!!!
        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.