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 № 3

Аналоговый ввод в среде Arduino, аналоговые датчики

Выполнил:
студент группы АДМ-21-05 _____ Абдулзагиров М.М.
(подпись) (ФИО)

Принял
преподаватель: _____ _Панфилов П. В. __
(подпись) (ФИО)

Дата:_____

Москва 2022

Работа с кнопками D2 D3.

Задание (Для продвинутых пользователей)

Написать программу которая выполет следующие действия:

Разработать программу 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. Установку целевой температуры (Тц) проводить с помощью переменного резистора присоединенного в выходу A0. Точность установки до 0.1 градуса. Диапазон регулирования +2 … +35 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. Кнопка D2 включает и выключает работу (регулирование) терморегулятора. При включении работы загорается светодиод синего цвета (D13).
6. Цикл управления 3 сек. В каждом цикле управления выводить состояние устройства (Тт, Тц, Гс, D12, D2) в консоль.
7. Добавить отображение текущего состояния регулятора, использую трех цветный светодиод (D9-D11). Отображать следующие состояния:

- Регулирование выключено — все светодиоды выключены.
- (Тт=Тц-Гс)&&(Тт<=Тц) — горит зеленый светодиод (D10)
- (Тт>Тц) — горит синий светодиод (D11)

Описание программы

В качестве IDE использовалась VS Code с расширением PlatfofmIO. В основном цикле идет обработка ввода консольных команд, обработка нажатий кнопки D2 для включения и выключения устройства, считывание паказаний датчика температуры (из-за отсутствия датчика он был представлен как потенциометр с диапазоном температуры от 2 до 35 градусов) и потенциометра регулирования требуемой температуры. Дребезг обрабатывается программно с помощью задержки. Вывод статуса каждые 3 секунды находится в цикле yield. Через ввод консольных команд можно выводить помощь, статус, включать и выключать устройство, вводить необходимый гистерезис в виде числа int и вводить величину ошибки в виде числа float с точностью 1 знак после запятой (по идее изменив 2 числа, можно сделать и 2 числа после запятой), при этом используется метод созданный метод findFloatHowInt, который не использует float переменные для своих преобразований и как результат возвращает число int со сдвигом на 1 десятичный разряд для хранения одного числа после запятой (умноженное на 10) .

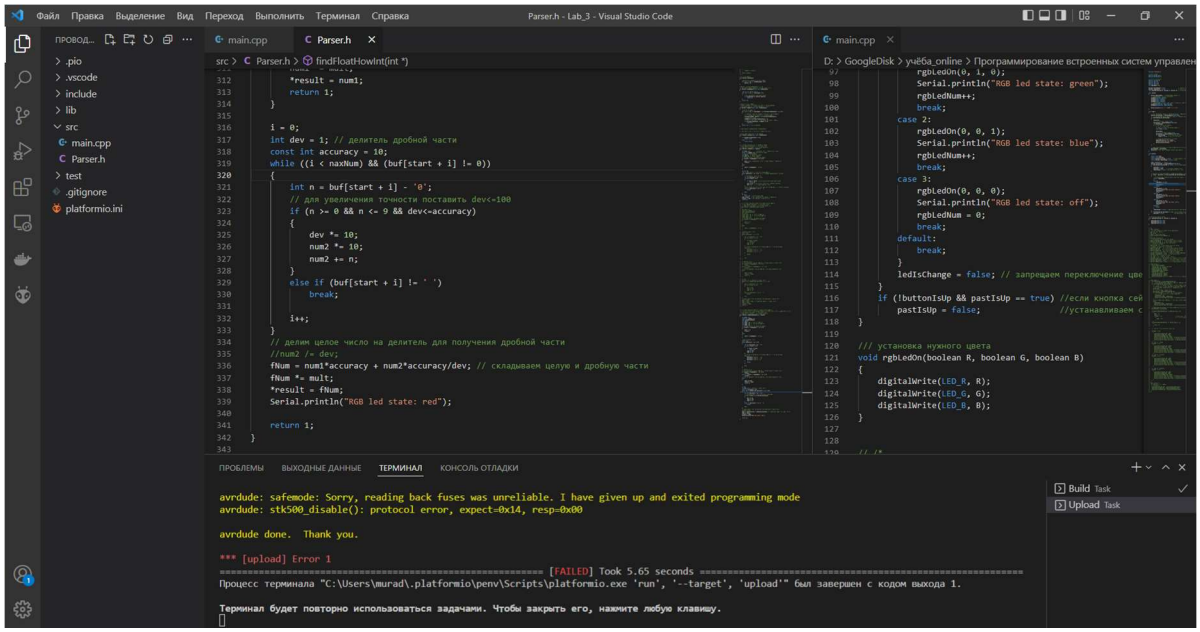

Рис 1. Окно VS Code.

Исходный код программы:

Листинг 1. Файл main.cpp

```
#include <Arduino.h>
/*
 * МГТУ СТАНКИН
 * Программирование встроенных систем управления
 * Лабораторная работа №3
 * АДМ-21-05
 * Абдулзагиров Мурад Магомедович
 * Murad.Abdulzagirov@gmail.com
 */

#include "Parser.h" //созданный для парсинга класс

#define baudRate 9600 // Скорость COM порта
//#define DEBUG

#define heater_out 12
#define work_status_led 13

#define LED_R 9
#define LED_G 10
#define LED_B 11

#define temp_LM35pin A2
#define setings_pot A0
#define job_status_bt 2

Parser pCommand;            // объект класса для хранения и распознавания строки
boolean buttonWasUp = true; // была ли кнопка отпущена?

// пространство имён нагревателя (вместо синглетон класса)
namespace Heater
{
    const int dev = 10;  // точность значений - 1 числа после запятой
    int _Tt = 0;            // показания датчика
    int _Tc = 0;
    const int Tmin = 2*dev;
    const int Tmax = 35*dev;

    int GT = 2*dev;
    int ERR = 0;

    boolean isWork = 1;
    boolean isHeated = 0;

    void Begin()
    {
        pinMode(heater_out, OUTPUT);
        pinMode(work_status_led, OUTPUT);
        pinMode(temp_LM35pin, INPUT);
        pinMode(setings_pot, INPUT);

        pinMode(job_status_bt, INPUT_PULLUP);
        digitalWrite(work_status_led, 1);
    }
```

```
    void SetTt(int temp)
    {
        //  if (temp>=Tmin && temp<= Tmax)
        _Tt = map(temp, 0, 1023, Tmin, Tmax) + ERR; // для потенциометра!
    }
    int GetTt() { return _Tt; }

    void SetTc(int temp)
    {
        // if (temp>=Tmin && temp<= Tmax)
        _Tc = map(temp, 0, 1023, Tmin, Tmax);
    }
    int GetTc() { return _Tc; }

    // установить статус работы нагревателя
    void SetWorkStatus(boolean);

    // включить или выключить нагрев
    void Switch(boolean);
}

//прототипы функций
void help();
void state();
void rgbLedOn(boolean R, boolean G, boolean B); // установка нужного цвета

void setup()
{
    //Устанавливаем режимы работы пинов как выходы
    pinMode(LED_BUILTIN, OUTPUT);
    Serial.begin(baudRate); //инициализируем последовательный порт и устанавливаем скорость
9600
    help();                 // вывод помощи

    pinMode(LED_R, OUTPUT);
    pinMode(LED_G, OUTPUT);
    pinMode(LED_B, OUTPUT);

    Heater::Begin();
    Serial.print(millis()); //вывод текущего времени
    Serial.print(F(" : Enter command > "));
}

void loop()
{
    if (Serial.available())
    {
        //отчищаем буфер и записываем в него строку
        pCommand.bufClean();
        pCommand.bufLength = Serial.readBytes((byte *)(pCommand.buf), pCommand.bufMaxLength);

        if (pCommand.isFind("help"))
        {
            Serial.print(F("help"));
            help();
        }
        else if (pCommand.isFind("state"))
        {
            Serial.print(F("state"));
```

```
            state();
        }
        else if (pCommand.isFind("setGT"))
        {
            Serial.print(F("setGT"));
            int GT;
            if (pCommand.findInt(&GT))
                if (GT > 5 || GT < -5) //проверка выхода за диапазон
                    Serial.println(F("error setGT: going out of range "));
                else
                    Heater::GT = GT * Heater::dev; //если проверка пройдена, выводим результат
            else
                Serial.println(F("error setGT ")); //число не обнаружено
        }
        else if (pCommand.isFind("setERR"))  // setERR 2.6
        {
            Serial.println(F("setERR"));
            int ERR;
            if (pCommand.findFloatHowInt(&ERR))
                if (ERR > 4*Heater::dev || ERR < -4*Heater::dev) //проверка выхода за диапазон
                    Serial.println(F("error setERR: going out of range "));
                else
                    Heater::ERR = ERR; //если проверка пройдена, выводим результат
            else
                Serial.println(F("error setERR ")); //число не обнаружено
        }
        else if (pCommand.isFind("ON"))
        {
            //включаем встроенный всетодиод
            digitalWrite(LED_BUILTIN, HIGH);
            Serial.println(F("Heater ON"));
            Heater::SetWorkStatus(1);
        }
        else if (pCommand.isFind("OFF"))
        {
            //выключаем встроенный всетодиод
            digitalWrite(LED_BUILTIN, LOW);
            Serial.println(F("Heater OFF"));
            Heater::SetWorkStatus(0);
        }
    }

    // обработчик кнопки
    boolean buttonIsUp = !digitalRead(job_status_bt);
    if (!buttonWasUp && buttonIsUp)
    {
        delay(10);
        buttonIsUp = !digitalRead(job_status_bt);
        if (buttonIsUp)
        {
            Heater::SetWorkStatus(!Heater::isWork);
            state();
        }
    }
    // запоминаем последнее состояние кнопки
    buttonWasUp = buttonIsUp;

    // считывание показания датчика температуры и потенциометра
    Heater::SetTt(analogRead(temp_LM35pin));
    Heater::SetTc(analogRead(setings_pot));
```

```
    // установка состояния RGB светодиода
    if (Heater::isWork)
        if (Heater::GetTt() <= (Heater::GetTc() - Heater::GT))
        {
            Heater::Switch(1);
            rgbLedOn(1, 0, 0);
        }
        else if (Heater::GetTt() >= Heater::GetTc())
        {
            Heater::Switch(0);
            rgbLedOn(0, 0, 1);
        }
        else
            rgbLedOn(0, 1, 0);
    else
        rgbLedOn(0, 0, 0);

    delay(10);

}

void yield()
{
    static long time;
    static long pastTime;
        time = millis(); // текущее время

    // повтор каждые 3 секунды
    if (time - pastTime >= 3000){
        state(); // вывод статуса

        pastTime=time;
    }
}

void help()
{
    Serial.println(F("\nThe following commands are described:"));
    Serial.println(F(" state - Heater current state"));
    Serial.println(F(" setGT - set hysteresis parameters "));
    Serial.println(F(" setERR - set the value of the added error"));
    Serial.println(F(" ON - turn on Heater"));
    Serial.println(F(" OFF - turn off Heater"));
    Serial.println(F(" help - help information"));
}

void state()
{
    Serial.println(F("\nState:"));
    Serial.println("Tt = " + String((float)Heater::_Tt / (float)Heater::dev));
    Serial.println("Tc = " + String((float)Heater::_Tc / (float)Heater::dev));
    Serial.println("GT = " + String((float)Heater::GT / (float)Heater::dev));
    Serial.println("ERR = " + String((float)Heater::ERR / (float)Heater::dev));
    if (Heater::isHeated)
        Serial.println(F("Heated = on"));
    else
        Serial.println(F("Heated = off"));
    if (Heater::isWork)
        Serial.println(F("Heater is Work"));
```

```
    else
        Serial.println(F("Heater is not Work"));
}

/// установка нужного цвета
void rgbLedOn(boolean R, boolean G, boolean B)
{
    digitalWrite(LED_R, R);
    digitalWrite(LED_G, G);
    digitalWrite(LED_B, B);
}

// установить статус работы нагревателя
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;
}
```

Листинг 2. Файл 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
    void bufClean();                        //отчистка буфера
};

// String Parser::findTextXML( String str, String findContext)
// {
//  int find1 = str.indexOf("<" + findContext + ">");
//  int find2 = str.indexOf("</" + findContext + ">");
//  if ((find1 < 1) || (find2 < 1))
//      return "";
//  String findStr = "";
//  for (int i = find1 + ("<" + findContext + ">").length(); i < find2; i++)
//  {
//      findStr += str[i];
//  }
//  return findStr;
// }

///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 возвращает 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 возвращает 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

При уменьшении температуры (потенциометр вне схемы) включается регулятор (красный светодиод) и происходит переключение цветов RGB светодиода.

Т.К. в ЭОС ограничение по размеру файла в 2 мегабайта, полный отчёт оставлю по ссылке

https://disk.yandex.ru/i/5eNzM00Z3IvL_g

на всякий продублирую

https://drive.google.com/file/d/12rCeI4D2usN917Y9B4ZSgIrne34weDbH/view?usp=sharing