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 № 1

Введение в программирование в среде Arduino IDE

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

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

Дата:_____

Москва 2022

Задание для лабораторной работы №1

Создание простейшего командного интерпретатора. Должны поддерживаться команды:

- help — экран помощи command1 — отображение в консоли введенной команды command1
- time - отображение в консоли результата вызова функции millis()
- setInt — ввод целого числа от -100 до +200 с выводом ошибки или введенного числа
- setStr — ввод строки до 8 символов с отображение введенной строки (только латинские буквы и цифры), вывод ошибки.
- info — версии интерпретатора, автора разработки (задаются константами в коде)

Реализация продвинутого варианта:

- реализовать интерпретатор без использования класса String
- реализация дополнительной команды setFloat — ввод дробного числа от -30 до +50 с точностью 2 десятичных разряда и выводом ошибки или введенного числа.

Описание программы

Программа состоит из 2 файлов (основной и подключаемый). В качестве IDE использовалась VS Code с расширением PlatfofmIO

Программа представляет собой интерпретатор команд. Ввод команд осуществляется посредством ввода одной строкой текстовой команды и её атрибута (при наличии). Была также добавлена команда setLed - управляет ШИМ сигналом 10-го пина (от 0 до 255), что является проверкой конвертера строки в число int.

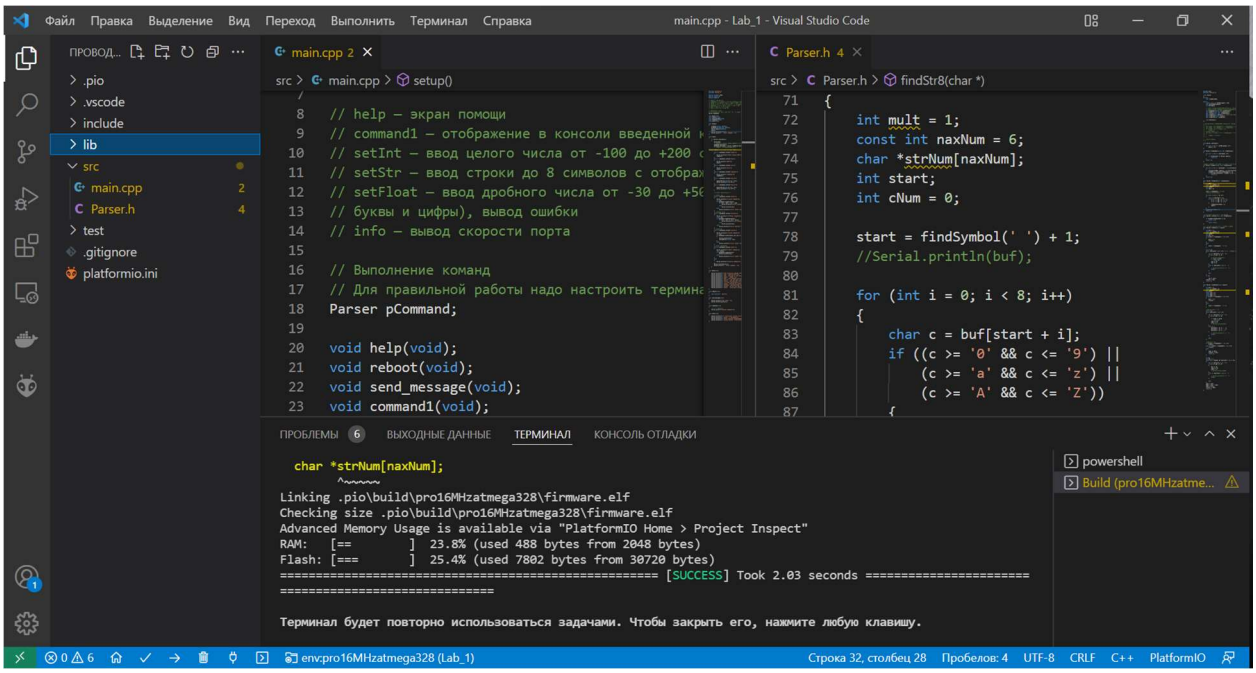

Рис 1. Окно VS Code.

Исходный код программы (был соединён в один файл с расширением ino)

Листинг 1.

```
/*
 * МГТУ СТАНКИН
 * Программирование встроенных систем управления
 * Лабораторная работа №1
 * АДМ-21-05
 * Абдулзагиров Мурад Магомедович
 * Murad.Abdulzagirov@gmail.com
 */
//#include "Parser.h" //созданный для парсинга класс

#define baudRate 9600 // Скорость COM порта
#define redLed 10     // светодиод, управляемый через setLed
//#define DEBUG

// help – экран помощи
// command1 – отображение в консоли введенной команды command1
// time - отображение в консоли результата вызова функции millis()
// setInt – ввод целого числа от -100 до +200 с выводом ошибки или введенного числа
// setStr – ввод строки до 8 символов с отображение введенной строки (только
латинские
// setFloat – ввод дробного числа от -30 до +50 с точностью 2 десятичных разряда
// буквы и цифры), вывод ошибки
// info – вывод скорости порта

/*
    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 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
}
```

```
bool Parser::findStr8(char *findContext)
{
    // int mult = 1;
    // const int naxNum = 6;
    // char *strNum[naxNum];
    int start;    // начальный индекс
    int cNum = 0; // индекс записи
    // начинаем поиск с пробела
    start = findSymbol(' ') + 1;

    for (int i = 0; i < 8; i++) // cNum <= 8; i++)
    {
        char c = buf[start + i];
        // проверка, это латинский символ или число
        if ((c >= '0' && c <= '9') ||
            (c >= 'a' && c <= 'z') ||
            (c >= 'A' && c <= 'Z'))
        {
            // записываем во временный буфер и сдвигаем индекс
            findContext[cNum] = c;
            cNum++;
        }
    }
    findContext[cNum] = 0; //обозначаем конец
    return 1;
}

///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 2х знаков после запятой
 * если найдено значение, то возвращает true
 * результат возвращает через ссылку result */
bool Parser::findFloat(float *result)
{
    int mult = 1;
    const int naxNum = 6;
    char *strNum[naxNum];
    int start;
    float num1 = 0; // число до запятой
    float num2 = 0; // число после запятой
    float 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
    while ((i < naxNum) && (buf[start + i] != 0))
    {
        int n = buf[start + i] - '0';
        if (n >= 0 && n <= 9)
        {
            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 + num2; // складываем целую и дробную части
    fNum *= mult;
    *result = fNum;
    return 1;
}

//////Parser//////

Parser pCommand; // объект класса для хранения и распознавания строки

//прототипы функций
void help(void);
void reboot(void);
void send_time(void);
void command1(void);
void info(void);

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

```
    help();                 // вывод помощи
    Serial.print(millis()); //вывод текущего времени
    Serial.print(F(" : Enter command > "));
}

void loop()
{
    if (Serial.available())
    {
        delay(400); //чтобы все данные успели передаться
        //отчищаем буфер и записываем в него строку
        pCommand.bufClean();
        pCommand.bufLength = Serial.readBytes((byte *)(pCommand.buf),
pCommand.bufMaxLength);

#pragma region вывод отладочной информации
#ifdef DEBUG
        Serial.println("bufLength " + String(pCommand.bufLength));
        Serial.println("findStr " + String(pCommand.findStr("setInt")));
        Serial.println("strcmp " + String(strcmp("setInt", pCommand.buf)));
        Serial.println(pCommand.findSymbol(' '));
#endif
#pragma endregion

        //проверяем, какая команда введена
        if (pCommand.isFind("reboot"))
        {
            Serial.println(F("Reset arduino UNO"));
            reboot();
        }
        else if (pCommand.isFind("help"))
        {
            Serial.print(F("help"));
            help();
        }
        else if (pCommand.isFind("info"))
        {
            Serial.print(F("info"));
            info();
        }
        else if (pCommand.isFind("command1"))
        {
            Serial.print(F("command1"));
            command1();
        }
        else if (pCommand.isFind("time"))
        {
            Serial.println(F("time"));
            send_time();
        }
        else if (pCommand.isFind("led_on"))
        {
            //включаем встроенный всетодиод
            digitalWrite(LED_BUILTIN, HIGH);
            Serial.println(F("Led ON"));
        }
        else if (pCommand.isFind("led_off"))
        {
            //выключаем встроенный всетодиод
            digitalWrite(LED_BUILTIN, LOW);
            Serial.println(F("Led OFF"));
        }
```

```
        /////task realization/////

        else if (pCommand.isFind("setInt"))
        {
            Serial.println(F("setInt command"));
            int res;                          //буфер для хранения результата
            if (pCommand.findInt(&res))       //если было найдено какое либо число int
                if (res > 200 || res < -100) //проверка выхода за диапазон
                    Serial.println(F("error setInt: going out of range "));
                else
                    Serial.println(res); //если проверка пройдена, выводим результат
            else
                Serial.println(F("error setInt ")); //число не обнаружено
        }

        else if (pCommand.isFind("setFloat"))
        {
            Serial.println(F("setFloat command"));
            float res;                          //буфер для хранения результата
            if (pCommand.findFloat(&res))       //если было найдено какое либо число
float
                if (res > 50.0 || res < -30.0) //проверка выхода за диапазон
                    Serial.println(F("error setFloat: going out of range "));
                else
                    Serial.println(res); //если проверка пройдена, выводим результат
            else
                Serial.println(F("error setFloat ")); //число не обнаружено
        }
        else if (pCommand.isFind("setLed"))
        {
            Serial.println(F("setLed command"));
            int res; //буфер для хранения результата
            // было ли найдено какое либо число int в пределах от 0 до 255
            if (pCommand.findInt(&res) && (res >= 0 && res <= 255))
            {
                Serial.println(res);      //вывод результата
                analogWrite(redLed, res); //установка ШИМ сигнала
            }
            else
                Serial.println(F("error setInt "));
        }
        // setStr abc53tf
        else if (pCommand.isFind("setStr"))
        {
            Serial.println(F("setStr command"));
            char res[8];                //буфер для хранения результата
            if (pCommand.findStr8(res)) //если строка была найдена
                Serial.println(res);    //вывод результата
            else
                Serial.println(F("error setInt "));
        }
        else
        {
            Serial.println(F("Invalid command"));
        }
        Serial.print(millis());
        Serial.print(F(" : Enter command > "));
    }
}

void help(void)
```

```
{
    Serial.println(F("\nThe following commands are described:"));
    Serial.println(F(" time - send millis()"));
    Serial.println(F(" info - versions of the interpreter, the author of the
development"));
    Serial.println(F(" reboot - reboot "));
    Serial.println(F(" help - show command list"));
    Serial.println(F(" led_off - led (pin d13) off"));
    Serial.println(F(" led_on - led (pin d13) on"));
    Serial.println(F(" /////task realization/////"));
    Serial.println(F(" setInt - parsing Int value"));
    Serial.println(F(" setFloat - parsing Float value"));
    Serial.println(F(" setLed - set 10 (red led) PWM output light value"));
    Serial.println(F(" setStr  - entering a string of up to 8 characters"));
}

void reboot(void)
{
    asm volatile("  jmp 0");
}

void send_time(void)
{
    Serial.print(F("Time (msec) "));
    Serial.println(millis());
}

void command1(void)
{
    //что-то выполняем
    Serial.print(F("execute command1"));
}

void info(void)
{
    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**********"));
}
```

Результаты выполнения программы

Протестируем программу на отладочной плате arduino pro mini с микроконтроллером atMega 328P

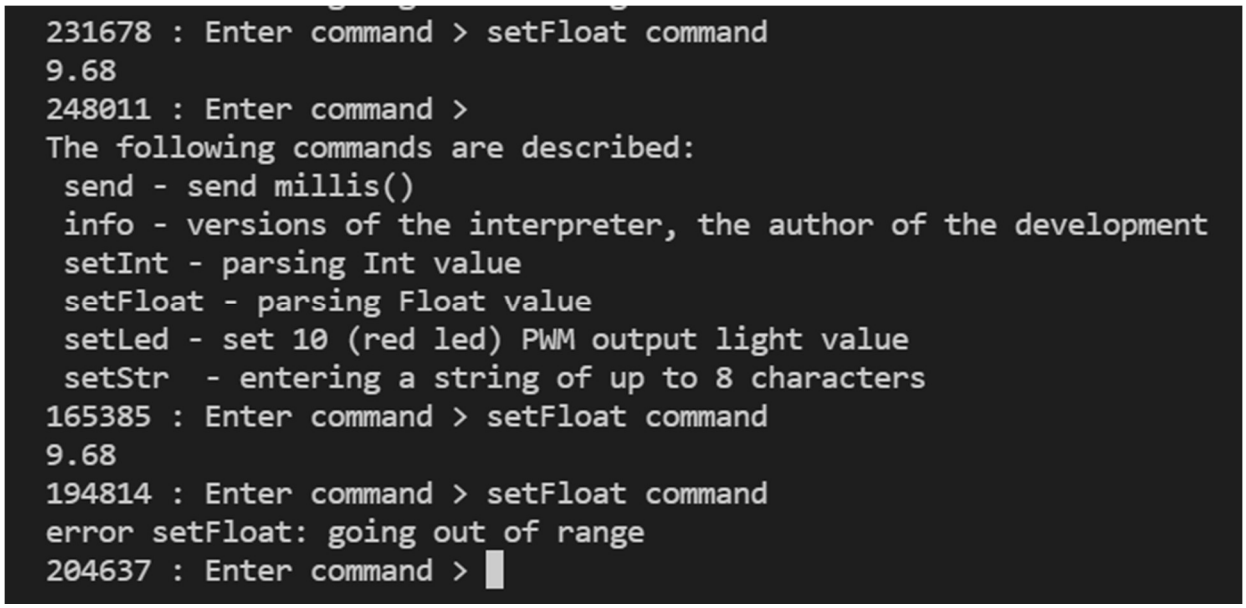

Рис.2. Консоль com порта.

Во второй команде было введено значение setFloat -90.679

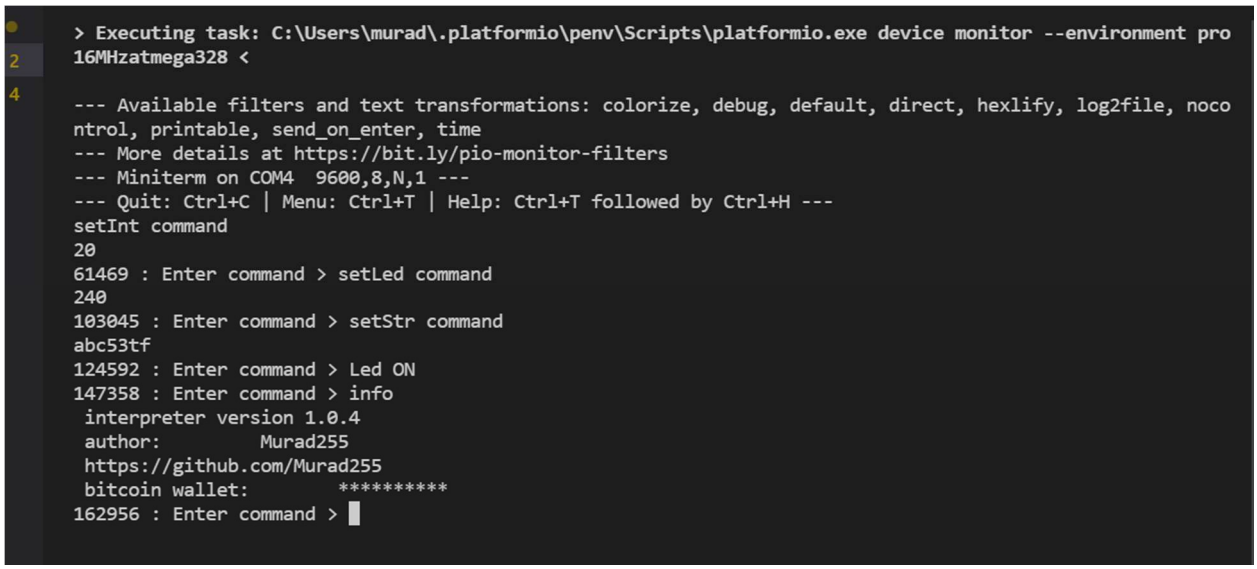

Рис.3. Консоль VS com порта.

Info выводит информацию о версии прошивки и авторе. Была также добавлена команда setLed - управляет ШИМ сигналом 10-го пина (от 0 до 255), что является проверкой конвертера строки и число int. При выполнении команды setLed 20 было установлено слабое свечение светодиода, подключённого к пину 10.

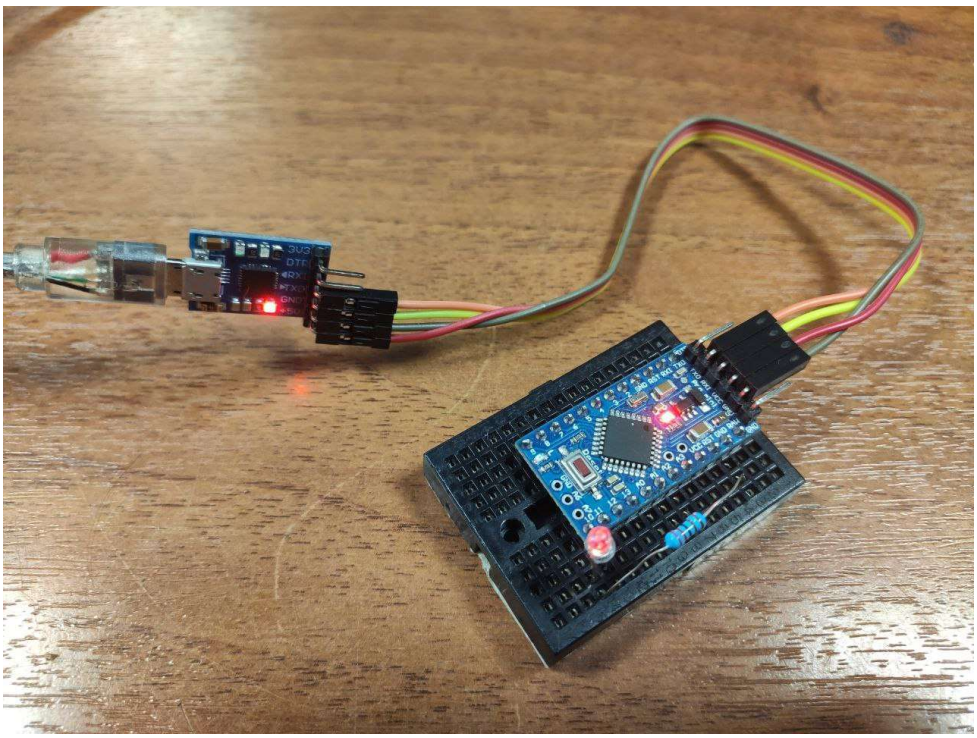

Рис.4. Отладочная плата.

При setLed 240 свечение увеличилось.

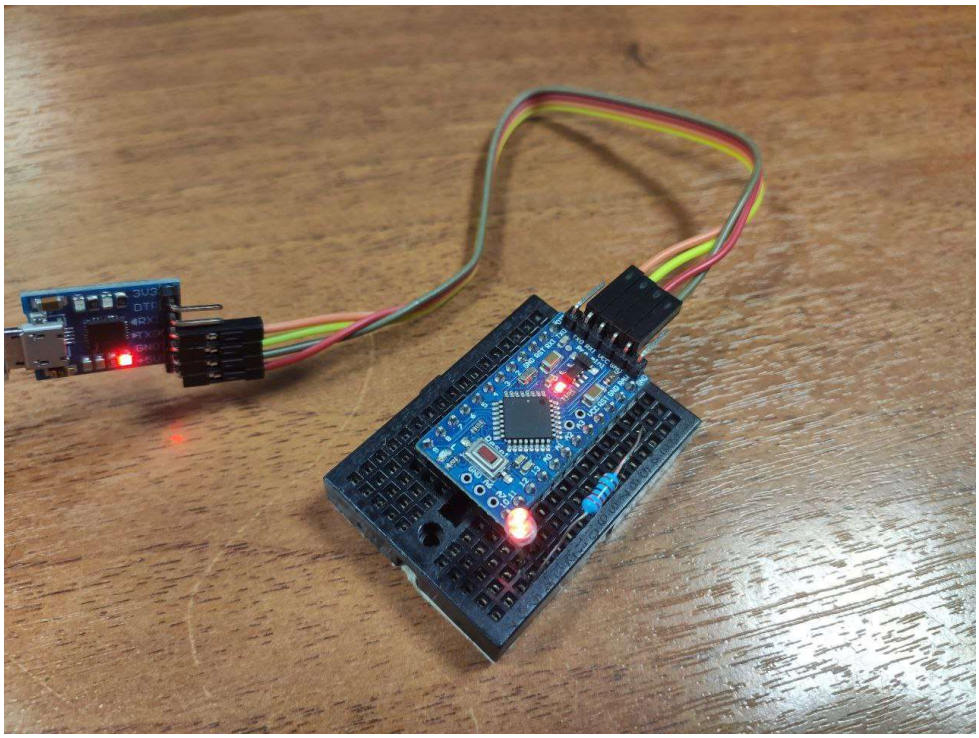

Рис.5. Отладочная плата.

Команда led_on включает всетодиод на плате (13 пин по умолчанию).

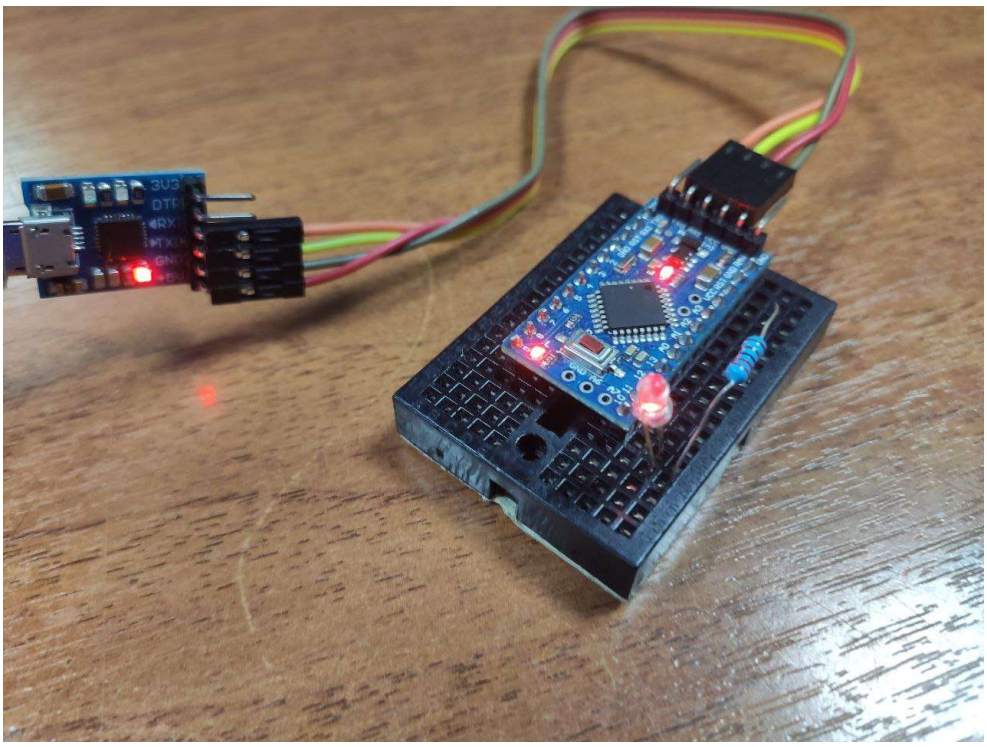

Рис.6. Отладочная плата.