IRSTI 50.05.09, 50.05.13 DOI: https://doi.org/10.26577/JMMCS.2022.v113.i1.07

D.A. Dogalakov*, **Zh.Zh. Baigunchekov**, **Zh.T. Zhumasheva**
Al-Farabi Kazakh National University, Kazakhstan, Almaty
*e-mail: d.dogalakov@gmail.com

INTEGRATION OF THE LABVIEW 2019 DEVELOPMENT TOOL IN WORKING WITH MICROCONTROLLERS

When connecting various sensors to microcontrollers, there are problems of the data received from them arise: analysis, data visualization, transmission and storage on remote storage media, etc This usually requires additional connection to microcontrollers of modules with the necessary functions. The way to use applications developed using the LabVIEW tool in the microcontroller operation scheme is one of the simple and reliable solutions in such cases. The study proposes to analyze the developed connection diagram, the wireless data transmission algorithm and their processing, using the example of connecting the GY21 sensor module, the ESP32 NodeMCU hardware platform and the application developed in the LabVIEW 2019 environment. The main technical characteristics of these modules are also given. Software products created using this LabVIEW software package can be supplemented with code fragments developed in other traditional programming languages, such as C / C ++, Pascal, Basic, FORTRAN. Conversely, you can use modules developed in LabVIEW in projects created in other programming systems. Thus, LabVIEW allows you to develop almost any application that interacts with any kind of hardware supported by the PC operating system. Using the technology of virtual instruments, a developer can turn a standard personal computer and a set of arbitrary control and measuring equipment into a multifunctional measuring and computing complex that allows remote control and monitoring via the Internet.

Key words: LabVIEW, ESP32 NodeMCU, sensor module GY21, ARDUINO IDE, Wireless data transmission.

Д.А. Догалаков*, Ж.Ж. Байгунчеков, Ж.Т. Жумашева
Әл-Фараби атындағы Қазақ ұлттық университеті, Қазақстан, Алматы қ.
*e-mail: d.dogalakov@gmail.com

Микроконтроллерлермен жұмыс жасау кезінде LabVIEW 2019 әзірлеу құралын біріктіру

Датчиктердің барлық түрлерін микроконтроллерлерге қосқан кезде, олардан алынған деректерді одан әрі өңдеу міндеттері туындайды: талдау, деректерді визуализациялау, қашықтағы ақпарат тасығыштарда беру және сақтау және т.б. Бұл әдетте микроконтроллерлерге қажетті функциялары бар модульдерге қосымша қосылуды қажет етеді. Микроконтроллерлердің жұмыс сұлбасында LabVIEW құралын қолдана отырып жасалған қосымшаларды пайдалану әдісі мұндай жағдайларда қарапайым және сенімді шешімдердің бірі болып табылады. Зерттеуде GY21 датчик модулін, ESP32 NodeMCU аппараттық платформасын және LabVIEW 2019 ортасында жасалған қосымшаны қосу мысалында жасалынған байланыс сұлбасын, деректерді сымсыз беру және өңдеу алгоритмін талдауды ұсынылады. Сондай-ақ, осы модульдердің негізгі техникалық сипаттамалары келтірілген. Осы LabVIEW бағдарламалық кешенін қолдана отырып жасалған бағдарламалық өнімдер C/C++, Pascal, Basic, FORTRAN сияқты басқа дәстүрлі бағдарламалау тілдерінде жасалған код үзінділерімен толықтырылуы мүмкін. Керісінше, LabVIEW-де жасалған модульдерді басқа бағдарламалау жүйелерінде жасалған жобаларда қолдануға болады.
Осылайша, LabVIEW дербес компьютердің операциялық жүйесі қолдайтын кез-келген аппараттық құралдармен өзара әрекеттесетін кез-келген қосымшаны жасауға мүмкіндік береді. Виртуалды аспаптар технологиясын қолдана отырып, әзірлеуші стандартты дербес компьютерді және ерікті бақылау-өлшеу жабдықтарының жиынтығын Internet арқылы қашықтан басқаруға және бақылауға мүмкіндік беретін көп функциялы өлшеу-есептеу кешеніне айналдыра алады.

Түйін сөздер: LabVIEW, ESP32 NodeMCU, GY21 датчик модулі, ARDUINO IDE, деректерді сымсыз жіберу.

Д.А. Догалаков*, Ж.Ж. Байгунчеков, Ж.Т. Жумашева
Казахский национальный университет имени аль-Фараби, Казахстан, г. Алматы
*e-mail: d.dogalakov@gmail.com

Интеграция инструмента разработки LabVIEW 2019 в работе с микроконтроллерами

При подключении всевозможных датчиков к микроконтроллерам возникают задачи дальнейшей их обработки получаемых от них данных: анализа, визуализации данных, передачи и хранения на удаленных носителях информации и т.д. Для этого как правило требуется дополнительное подключение к микроконтроллерам модулей с необходимыми функциями. Способ использования приложений, разработанных с помощью инструмента LabVIEW в схеме работы микроконтроллеров, является в таких случаях одним из простых и надежных решений. В исследовании разбирается разработанная схема подключения, алгоритм беспроводной передачи данных и их обработки, на примере подключения модуля датчика GY21, аппаратной платформы ESP32 NodeMCU и приложения разработанного в среде LabVIEW 2019. Также приводятся основные технические характеристики указанных модулей. Программные продукты, созданные с использованием данного программного комплекса LabVIEW, могут быть дополнены фрагментами кода, разработанными на других традиционных языках программирования, например C/C++, Pascal, Basic, FORTRAN. И наоборот можно использовать модули, разработанные в LabVIEW в проектах, создаваемых в других системах программирования. Таким образом, LabVIEW позволяет разрабатывать практически любые приложения, взаимодействующие с любыми видами аппаратных средств, поддерживаемых операционной системой ПК. Используя технологию виртуальных приборов, разработчик может превратить стандартный персональный компьютер и набор произвольного контрольно-измерительного оборудования в многофункциональный измерительно-вычислительный комплекс, допускающий удаленное управление и наблюдение через Internet.

Ключевые слова: LabVIEW, ESP32 NodeMCU, модуль датчика GY21, ARDUINO IDE, беспроводная передача данных.

1 Introduction

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) is a specialized, flexible programming environment used to create unique utilities and applications for measuring instruments. LabVIEW is a data collection, analysis and processing system that includes powerful tools for visualizing results. It is also used to control technical objects and technological processes [1].

Currently, LabVIEW is widely used in the following areas [2]:

– Automotive industry;
– Telecommunications;
– Aerospace industry;
– Semiconductor industry;
– Development and production of electronics;
– Extractive industry;
– Shipbuilding industry;
– Biomedicine;

ESP32 NodeMCU is a highly integrated, combined (Wi-Fi + Bluetooth) chip designed for solutions that require the lowest power consumption figures, shown in (Fig. 1).

ESP32 NodeMCU supports the entire stack of Wi-Fi 802.11n and BT4.2 protocols, providing this functionality via SPI / SDIO or I^2C / UART [3].

ESP32 NodeMCU Module Specifications:

CPU:	Xtensa Dual-Core 32-bit LX6, 160 MHz или 240 MHz (до 600 DMIPS);
Memory:	520 KByte SRAM, 448 KByte ROM;
Flash:	1, 2, 4 . . . 64 Mb;
Wireless:	Wi-Fi: 802.11b/g/n/e/i, до 150 Mbps с HT40;
Bluetooth:	v4.2 BR/EDR и BLE;
Manufacturer:	Espressif Systems;

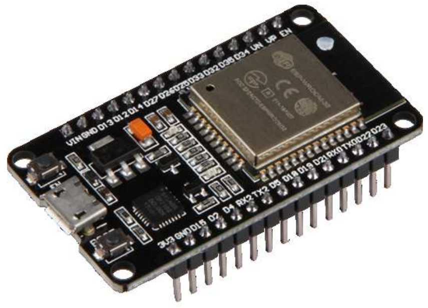


Figure 1: Module ESP32 NodeMCU

The GY21 humidity and temperature sensor module based on the SHT21 sensor is a high-precision module for measuring temperature and humidity, shown in (Fig.2). It has a very low error in its class: when measuring temperature, it is 0.4%, and humidity is 2%.

Due to such a low error and universal I2C interface, this module is suitable for temperature and humidity measurement in industrial areas [4].

GY-21 Specifications:

– Sensor:	SHT21
– Interface:	I2C
– Sensor operating voltage range:	$1.9 - 3.6V$
– Module supply voltage:	$5V$
– Current consumption in measuring mode:	$300\mu A$
– Standby current consumption:	$0.15\mu A$
– Humidity: operating range:	0 to 100%
– Measurement accuracy:	$\pm 3\%(max), 0 - 80\%$
– Temperature: operating range:	$-40 to + 125°C$
– Measurement accuracy:	$\pm 0.4°C(max), -10 to 85°C$

– Factory calibration
– Built-in battery discharge detector (sets the flag if the supply voltage drops below 2.25 V)
– Built-in heater for self-diagnosis of sensors

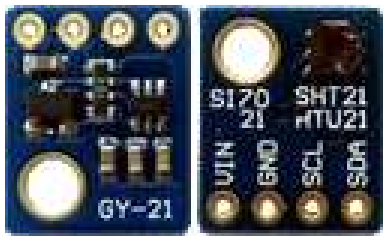


Figure 2: Sensor module GY21 (SHT21)

2 Material and methods

The algorithm of actions of our circuit is as follows: Sensor module GY21 (SHT21) connected to the ESP32 NodeMCU Module will transmit temperature and humidity parameters to it according to the program executed on the ESP32 NodeMCU Module. To do this, we will develop the necessary program (sketch for uploading) in the **Arduino IDE** [5]. The received data, already available in the RAM of the Module ESP32 NodeMCU, upon request from the application developed in LabVIEW, will be read and processed on the computer via the TCP IP protocol wirelessly.

Our experiment starts with assembling a circuit diagram, which were designed in the program Fritzing [6]. The connection diagram of the GY21 humidity and temperature sensor module to the ESP32 NODMCU v.3 hardware platform is shown in (Fig. 3).

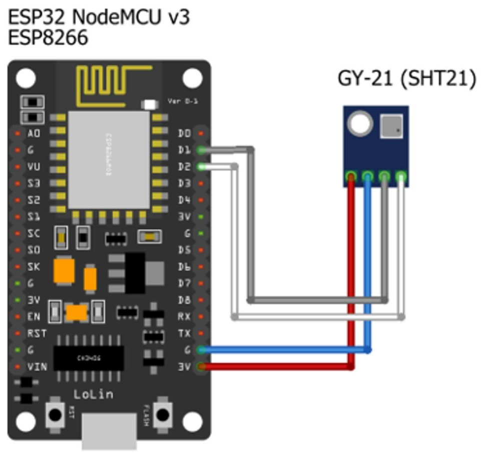


Figure 3: Connection diagram

The next step is to write a sketch in the Arduino IDE program.

Arduino IDE is an integrated development environment for Windows, MacOS and Linux, developed in C and C ++, designed to create and download programs on Arduino-compatible boards, as well as on boards from other manufacturers.

The ESP8266 WiFi.h library was used to work with the ESP32 hardware platform interface:

– The library "Sodaq_SHT2x.h" was also connected to work with the selected module of the humidity and temperature sensor GY21;

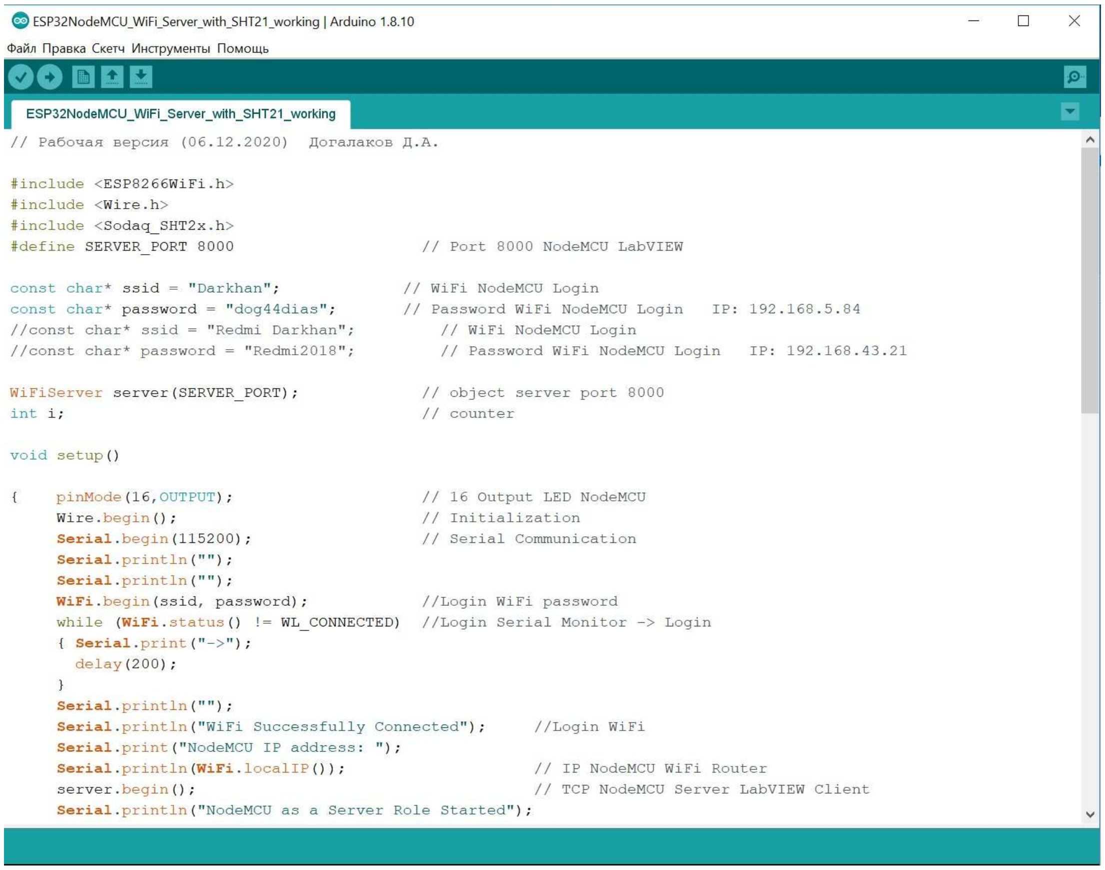
– Protocol for data transfer between ESP32 NodeMCU and the main control program: TCP IP;

– Number of data transmission port: 8000;

– Size of data transmission packet: 12 bytes;

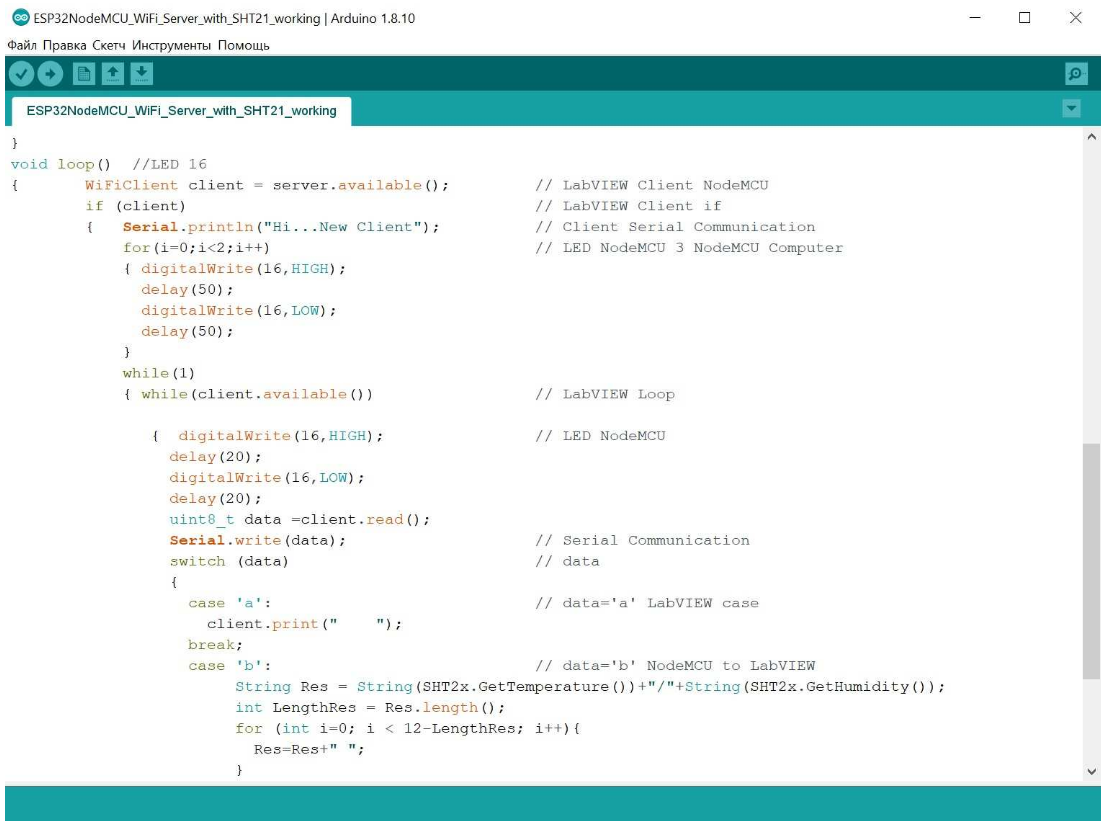
When forming a data packet of temperature and humidity readings before sending to the control program, its size (in bytes) may not be constant due to the dimension of the readings themselves, for example, the temperature may be $T = -12°C$ or $0°C$ or $25°C$, respectively for temperature readings we must to allocate 3 or 1 or 2 bytes. To do this, we set the maximum packet size for send 12 bytes, taking into account the humidity readings and other possible parameters in the future, and in the case of missing bytes from the data, we fill our packet with characters ("non-breaking space"code Alt + 255) up to 12 bytes.

Below in (Fig. 4) and (Fig. 5) there is a listing (code) of the program loaded into the memory of the ESP32 NODMCU v.3 module:


ESP32NodeMCU_WiFi_Server_with_SHT21_working | Arduino 1.8.10
Файл Правка Скетч Инструменты Помощь
ESP32NodeMCU_WiFi_Server_with_SHT21_working

```
// Рабочая версия (06.12.2020)  Догалаков Д.А.

#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Sodaq_SHT2x.h>
#define SERVER_PORT 8000                   // Port 8000 NodeMCU LabVIEW

const char* ssid = "Darkhan";            // WiFi NodeMCU Login
const char* password = "dog44dias";      // Password WiFi NodeMCU Login   IP: 192.168.5.84
//const char* ssid = "Redmi Darkhan";        // WiFi NodeMCU Login
//const char* password = "Redmi2018";        // Password WiFi NodeMCU Login   IP: 192.168.43.21

WiFiServer server(SERVER_PORT);            // object server port 8000
int i;                                     // counter

void setup()

{    pinMode(16,OUTPUT);                   // 16 Output LED NodeMCU
     Wire.begin();                         // Initialization
     Serial.begin(115200);                 // Serial Communication
     Serial.println("");
     Serial.println("");
     WiFi.begin(ssid, password);           //Login WiFi password
     while (WiFi.status() != WL_CONNECTED)  //Login Serial Monitor -> Login
     { Serial.print("->");
       delay(200);
     }
     Serial.println("");
     Serial.println("WiFi Successfully Connected");      //Login WiFi
     Serial.print("NodeMCU IP address: ");
     Serial.println(WiFi.localIP());                     // IP NodeMCU WiFi Router
     server.begin();                                     // TCP NodeMCU Server LabVIEW Client
     Serial.println("NodeMCU as a Server Role Started");
```

Figure 4: Listing of the program for loading into the ESP32 NODMCU v.3 module (beginning)


ESP32NodeMCU_WiFi_Server_with_SHT21_working | Arduino 1.8.10
Файл Правка Скетч Инструменты Помощь
ESP32NodeMCU_WiFi_Server_with_SHT21_working

```
}
void loop()  //LED 16
{       WiFiClient client = server.available();          // LabVIEW Client NodeMCU
        if (client)                                      // LabVIEW Client if
        {   Serial.println("Hi...New Client");           // Client Serial Communication
            for(i=0;i<2;i++)                             // LED NodeMCU 3 NodeMCU Computer
            { digitalWrite(16,HIGH);
              delay(50);
              digitalWrite(16,LOW);
              delay(50);
            }
            while(1)
            { while(client.available())                  // LabVIEW Loop

               {  digitalWrite(16,HIGH);                 // LED NodeMCU
                 delay(20);
                 digitalWrite(16,LOW);
                 delay(20);
                 uint8_t data =client.read();
                 Serial.write(data);                     // Serial Communication
                 switch (data)                           // data
                 {
                   case 'a':                             // data='a' LabVIEW case
                     client.print("    ");
                   break;
                   case 'b':                             // data='b' NodeMCU to LabVIEW
                        String Res = String(SHT2x.GetTemperature())+"/"+String(SHT2x.GetHumidity());
                        int LengthRes = Res.length();
                        for (int i=0; i < 12-LengthRes; i++){
                          Res=Res+" ";
                        }
```

Figure 5: Listing of the program for loading into the ESP32 NODMCU v.3 module (continuation)

Further, in the development environment LABVIEW 2019 in the "Block diagram" we create a general block diagram of the operation of our control program, (Fig. 6).

Part 1 describes the instructions and settings required to connect to the ESP32 NODMCU using the TCP IP protocol. To scan the network and determine the IP address assigned by the WI-FI network to the ESP32 NODMCU, the free Hercules SETUP utility was used [7].

Part 2 executes command "b" – sending a request for data transfer to the ESP32 NODMCU module.

Part 3 and part of **Part 7** output the temperature and humidity readings to special separate areas of the program interface in the form of graphs.

Part 4 reads the system time and generates a new line according to the specified format.

Part 5 and **Part 6** describes an algorithm for dividing the received data packet of 12 bytes into two parts, to separate the temperature and humidity readings from it separately.

Part 7 describes the procedure for writing the finally formed data line to a text file at the specified path.

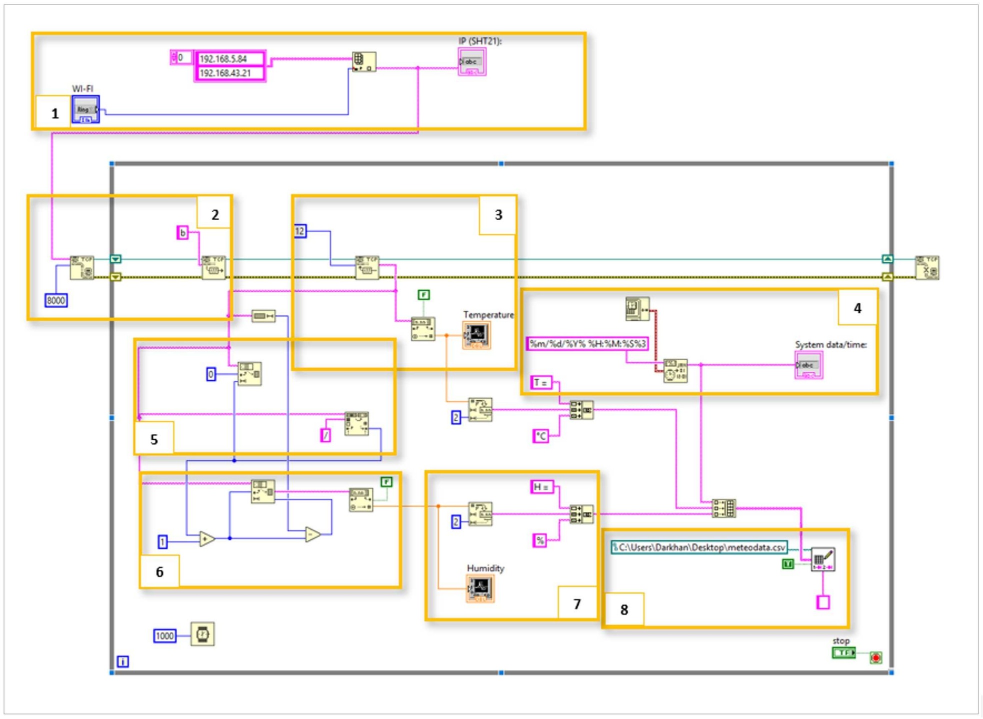


Figure 6: Algorithm of the control program in the LABVIEW 2019 environment

The control program interface developed in the LABVIEW environment is shown in (Fig. 7).

The panel displays: selectable shared wireless network (WI-FI), the IP address received by the ESP32 NODMCU from the network and the graphics of the read data;

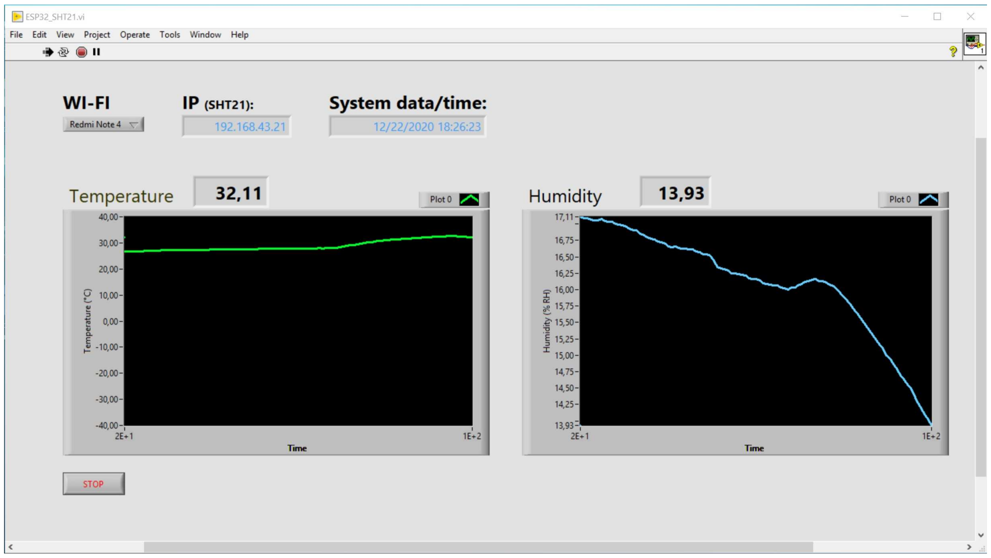
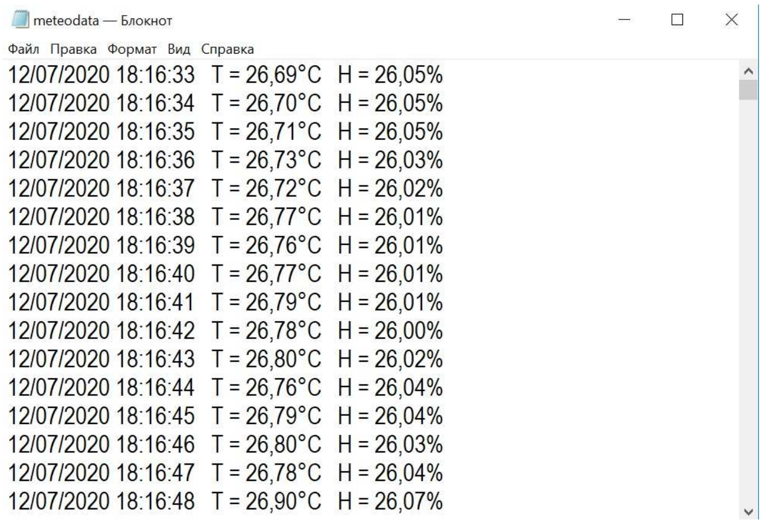


Figure 7: The interface of the control program in the LABVIEW 2019 environment

Read and transmitted temperature and humidity data from the ESP32 NodeMCU to the main control program on the PC. Writing data to a text file with a time interval of 1 second is shown in (Fig. 8).



```
12/07/2020 18:16:33  T = 26,69°C  H = 26,05%
12/07/2020 18:16:34  T = 26,70°C  H = 26,05%
12/07/2020 18:16:35  T = 26,71°C  H = 26,05%
12/07/2020 18:16:36  T = 26,73°C  H = 26,03%
12/07/2020 18:16:37  T = 26,72°C  H = 26,02%
12/07/2020 18:16:38  T = 26,77°C  H = 26,01%
12/07/2020 18:16:39  T = 26,76°C  H = 26,01%
12/07/2020 18:16:40  T = 26,77°C  H = 26,01%
12/07/2020 18:16:41  T = 26,79°C  H = 26,01%
12/07/2020 18:16:42  T = 26,78°C  H = 26,00%
12/07/2020 18:16:43  T = 26,80°C  H = 26,02%
12/07/2020 18:16:44  T = 26,76°C  H = 26,04%
12/07/2020 18:16:45  T = 26,79°C  H = 26,04%
12/07/2020 18:16:46  T = 26,80°C  H = 26,03%
12/07/2020 18:16:47  T = 26,78°C  H = 26,04%
12/07/2020 18:16:48  T = 26,90°C  H = 26,07%
```

Figure 8: Received data written to text file

You can also use other GPIOs on the ESP32 NodeMCU board to turn on or off peripheral equipment such as relays [8].

3 Results and discussion

The proposed scheme of collaboration between the LABVIEW environment and microcontrollers for data transmission and processing allows you to exclude the purchase of additional output and recording devices. At the same time, the volume of data received can be significantly increased, and their consolidation and visualization in applications using other additional tools of the LABVIEW environment becomes convenient and practical. The selected mechanism for using the LABVIEW environment in integration with microcontrollers is also planned to be used in other tasks related to technical vision for controlling relays and electric drives on electric vehicles [9]. One of the examples of such solutions is the using NI LabVIEW software and the NI Vision Development Module to develop a system to monitor the Paris RER. The key element of the infrastructure of the transport system are rails. After the installation of the controls, their position may change depending on the conditions of the environment, for example, the temperature. This integrated system measures these rail position changes [10].

4 Conclusion

This article presented the joint operation of the LabVIEW 2019 software package and one of the ESP32 NodeMCU hardware platforms with the GY21 module connected to it. The main technical characteristics of these devices are described, as well as commands for connecting and transferring data between them. The great opportunities inherent in the graphical programming and execution environment of LabVIEW 2019 programs (various libraries, integration and data exchange components, visual components) generally show and provide unlimited opportunities for software developers, both for research tasks (data collection and analysis), and for automation of various technological processes. This practical example we have analyzed confirms this.

References

[1] LabVIEW, https://ru.wikipedia.org/wiki/LabVIEW. Last accessed 25 Oct. 2021

[2] Dostoinstva LabVIEW, https://studbooks.net/2138894/informatika/dostoinstva_labview. Last accessed 25 Oct. 2021

[3] ESP32 microcontroller description, http://micpic.ru/home/proekty-na-esp32/194-opisanie-mikrokontrollera-esp32.html. Last accessed 31 Dec. 2020

[4] Temperature and humidity sensor HTU-21 (GY-21), https://3d-diy.ru/wiki/arduino-datchiki/datchik-temperatury-i-vlazhnosti-gy-21/. Last accessed 31 Dec. 2020

[5] Software (Downloads), https://www.arduino.cc/en/software. Last accessed 25 Oct. 2021

[6] Installing Fritzing, https://fritzing.org/. Last accessed 28 Oct 2021

[7] Hercules SETUP utility, https://www.hw-group.com/software/hercules-setup-utility. Last accessed 03 Jan. 2021

[8] MicroPython: ESP32/ESP8266 Relay Module Web Server (AC Appliances), https://randomnerdtutorials.com/micropython-relay-web-server-esp32-esp8266/. Last accessed 28 Oct. 2021

[9] Automated Car Braking System using Labview, https://www.researchgate.net/figure/Labview-Simulation-GUI_fig2_261454197. Last accessed 03 Jan. 2021

[10] https://www.ni.com/ro-ro/innovations/case-studies/19/monitoring-rail-positions-with-labview-and-ni-vision-software.html. Last accessed 28 Oct. 2021