MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
“IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE”
DEPARTMENT OF BIOMEDICAL ENGINEERING

Shlykov V. V., Stasyuk Y. P.

MICROPROCESSOR TECHNICS

Workshop on discipline for students of specialties 163 "Biomedical Engineering" and 152 "Metrology and information-measuring devices"

Kyiv
Igor Sikorsky Kyiv Polytechnic Institute
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

Шликов В.В., Стасюк Ю.П.

МІКРОПРОЦЕСОРНА ТЕХНІКА

Практикум з дисципліни для студентів спеціальностей 163 «Біомедична інженерія» та 152 «Метрологія та інформаційно-вимірювальна техніка»

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Медична інженерія» спеціальності 163 «Біомедична інженерія»

Ухвалено на засіданні кафедри біомедичної інженерії ФБМІ (протокол № 4 від «18» листопада 2020 р.)

Київ
«КПІ ім. Ігоря Сікорського»
2020

Рецензент *Худецький І.Ю.,* д.м.н., проф., завідувач кафедри біобезпеки і здоров'я людини КПІ ім. Ігоря Сікорського,

Маринський Г.С., д.т.н., с.н.с., завідувач відділу зварювання та споріднених технологій в медицині та екології №017 Інституту електрозварювання ім.Є.О.Патона

Відповідальний редактор *Зубчук В.І.*, к.т.н., доц., доцент кафедри біомедичної інженерії КПІ ім. Ігоря Сікорського

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 14 від 14.12.2020 р.) за поданням Вченої ради факультету біомедичної інженерії (протокол № 7 від 30.11.2020 р.)

Електронне мережне навчальне видання

Шликов Владислав Валентинович, д-р техн. наук, доц.
Стасюк Юрій Петрович, асистент

МІКРОПРОЦЕСОРНА ТЕХНІКА

Практикум з дисципліни для студентів спеціальностей 163 «Біомедична інженерія» та 152 «Метрологія та інформаційно-вимірювальна техніка»

Мікропроцесорна техніка: Практикум з дисципліни для студентів спеціальностей 163 «Біомедична інженерія» та 152 «Метрологія та інформаційно-вимірювальна техніка» [Електронний ресурс]: навч. посіб. для студ. спеціальності 163 «Біомедична інженерія» та 152 «Метрологія та інформаційно-вимірювальна техніка» / В.В. Шликов, Ю.П. Стасюк; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,4 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 147 с.

Анотація: Лабораторний практикум з мікропроцесорної техніки містить практичні завдання з програмування на мові С++ мікроконтролера Arduino UNO і відображення показань давачів у системі LabView. Теоретичний та практичний матеріал, який викладено у навчальному посібнику допомагає накопичувати й ефективно використовувати отриману інформацію з дисципліни на всіх стадіях навчального процесу, що є важливим для підготовки бакалаврів та необхідною ланкою у науковому пізнанні практичних основ біомедичної електроніки.

Workshop on discipline for students of specialties 163 "Biomedical Engineering" and 152 "Metrology and information-measuring technique" / Shlykov V. V., Stasyuk Y.P. – Kyiv, Igor Sikorsky Kyiv Polytechnic Institute, 2020. – 147 p.

Educational and methodical edition

MICROPROCESSOR TECHNICS

Workshop on discipline for students of specialties 163 "Biomedical Engineering" and 152 "Metrology and information-measuring devices"

Authors: ***Vladyslav Shlykov***, Doctor of Engineering, Associate Professor,
***Yuri Stasyuk*,** Senior Lecturer, Department of Biomedical Engineering

Editor-in-Chief: *Viktor Zubchuk*, Ph.D., Associate Professor

Reviewers: *Igor Khudetskyy*, Doctor of Medicine, Professor,
Mariinsky George, Doctor of Engineering, Senior Research Associate

Edited by the authors

Підп. до друку Формат Папір друк. №3. Друк офс.
Ум. друк. арк. 0,93. Обл.-вид. арк. 1,0. Зам. № 000. Наклад 100 пр.

CONTENT

Introduction

The purpose of practical classes and laboratory work in the discipline "Microprocessor Engineering", performed using software in the programming environment of microprocessors Arduino UNO (Arduino Genuino), laboratory models of measuring equipment using software and information tools National Instruments LabVIEW 2010, are:

- studying the principles of construction of electronic components of microprocessor systems based on Arduino UNO processors;

- management of microprocessor systems using I2C interface and NI VISA visual components;

- acquisition of skills for working with sensors and sensors with microprocessor control;

- acquisition of skills of preparation, carrying out and documentation of results of measurement and functioning of microprocessor systems.

As a result of laboratory work, the student should be able to formulate requirements for microprocessor system (MPS) parameters. For designing a motor vehicle it is necessary to learn to consider the following basic characteristics:

- microprocessor system capabilities;

- the amount of ROM programs and RAM data of vehicles;

- set of commands and methods of addressing;

- digit and speed;

- requirements for the power source and power consumption;

- cost of vehicles in different variants of execution;

- Efficiency of programming and debugging tools using National Instruments LabVIEW 2010 software and information.

Laboratory research assignments are designed for 2 academic hours. The results of the research and measurements should be documented and presented to the teacher at the end of the class. In the course of laboratory work, a report shall be drawn up, which shall include:

1. Title page of the report.

2. Theoretical information on the topic of the study.

3. ICS schemes for which research and measurements were carried out.

4. LabVIEW schemes that have been used to debug a vehicle.

5. Program code that illustrates the processes studied.

6. Conclusions on the results of research and measurements.

7. Answers to control questions.

1. General information about the Arduino platform

An important feature of systems on the Arduino platform that sets them apart from regular computers is that performance is a key factor in their work. This means that it is not always easy enough to perform the desired function to solve the problem. It can be said that the solution should be non-standard. For example, it should run fast, or it should run at low power, or it should be cheap. And this is really the big difference between developing microprocessor systems and, for example, traditional computer software design.

In Fig. 1.1 is a schematic view of the general scheme. The microprocessor system (MPS) must receive data from the outside world, process it and then output the data to the outside world. So, first of all, it has a set of sensors for receiving data. These sensors can receive information about the outside world in different ways, so there are many different types of similar devices.

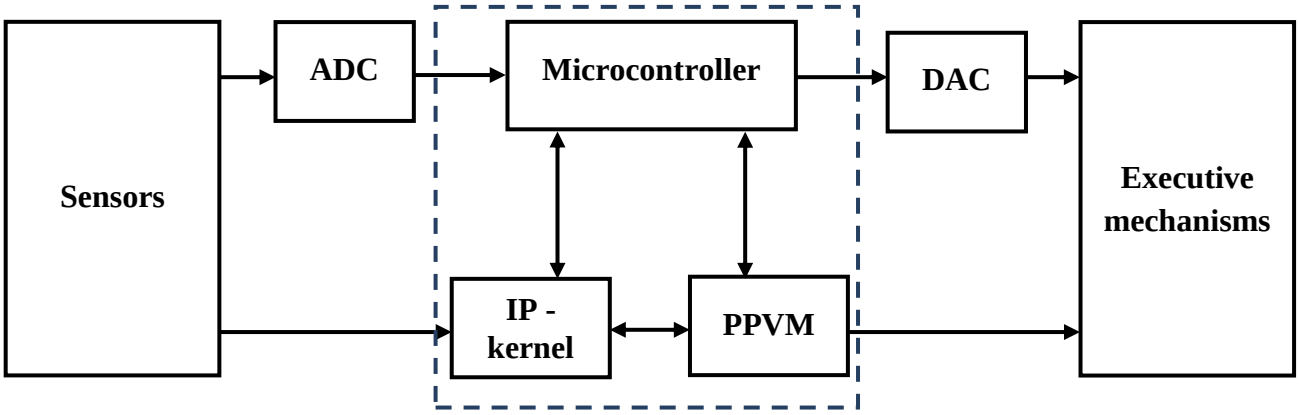

Fig. 1.1. Device of embedded systems

The simplest type of sensor is a button or something that receives data in a very simple way: pressed or not pressed. The system can also receive audio information using a microphone. Camcorders are also sensors that receive information as an image. The touch screen not only displays but also receives information. And so on - there are many different types of sensors through which the vehicle can receive information.

So, first the information enters the system and then it enters its core, which deals with its processing. At the end of this process, when the system has decided what to do with the information or what decision to make based on it, the ICU should produce some results. This is done by the actuators or actuators shown on the right side of the diagram in fig. 1.1. The actuator may be an LED that can burn or flash. For example, it may signal that your camcorder is recording or running out of power. But there are other enforcement mechanisms. For example, there are electric motors inside the camera that control the lens and sharpen it. These engines are actuators, and their movement is the output of the system.

There are many different types of actuators: the speakers reproduce the sound, the lamps allow the system to output light, TFT screens. Thus, the MPS receives data from the sensors, processes the information received, and then sends the appropriate signal to the actuators to force them to do something in real time in response to the data it received. In this sense, the MPS provides a link between the sensors and actuators.

In the center of the vehicle is a microcontroller. In addition to the microcontroller may be other components. Two of those that are quite common are shown in Fig. 1.1 is an IP kernel and a PCOS. IP is the English abbreviation for Intellectual Property, that is, an intellectual product. An IP kernel is a ready-made unit for designing devices, for example, it can be a ready-made chip that performs one function or several closely related functions focused on a specific task. That is, this unit is not a universal programmable general purpose chip. Most often, IP kernels are used for common tasks - for those that are running again and again and are in high demand. If a task occurs in only one specific system, it does not make sense for it to make a dedicated IP kernel.

Consider, for example, Texas Instruments controllers. Such firms produce many specialized circuits. For example, if you want to implement MPEG compression, the manufacturer Texas Instruments will find several dozen chips that make it, and choose the appropriate.

IP cores must interact with the microcontroller, which is the center of the entire system. It manages IP kernels and commands when they need to get started, passes information to them, and gets the result. For example, if the IP core compresses the video, then the microcontroller tells it when to start compressing the data and what data to compress. To do this, the microcontroller sends him a certain sequence of signals, and when the chip finishes its work, it sends certain signals to the microcontroller, informing that the result is ready.

Another type of component that is used in vehicles is programmable valve arrays, or PPVMs. These are quite sophisticated hardware devices, or, to be more precise, hardware-programmable circuits. A conventional programmable chip consists of a large number of transistor-based logic gates. The valves, in turn, are interconnected by a very complex number of connections, which, in fact, receive control and information signals. The scheme combines these logical devices and determines the logic of the chip operations.

In turn, PPVM can be figuratively compared to a set of tens of transistors, resistors, light bulbs, buttons, wires, terminals, actuators and other elements, which must perform some special task. Thus, PPVM is a complex chip that allows you to change the picture of the interconnections of their logical components, that is, essentially the hardware configuration of the device. Here it is important to emphasize once again that this is not about flashing the software stuff of the device, but about changing its hardware structure.

Thus, the configuration of each vehicle depends on the solution to which it is designed and the most efficient hardware configuration of the device is selected.

1.1.Arduino UNO microcontrollers

Microcontroller is the center of the vehicle. In Fig. 1.1.1 one of the Arduino boards is presented. It is not a microcontroller, it is a PCB with many elements, among which you can see a large black rectangular chip. This chip is a microcontroller that executes programs that control the device.

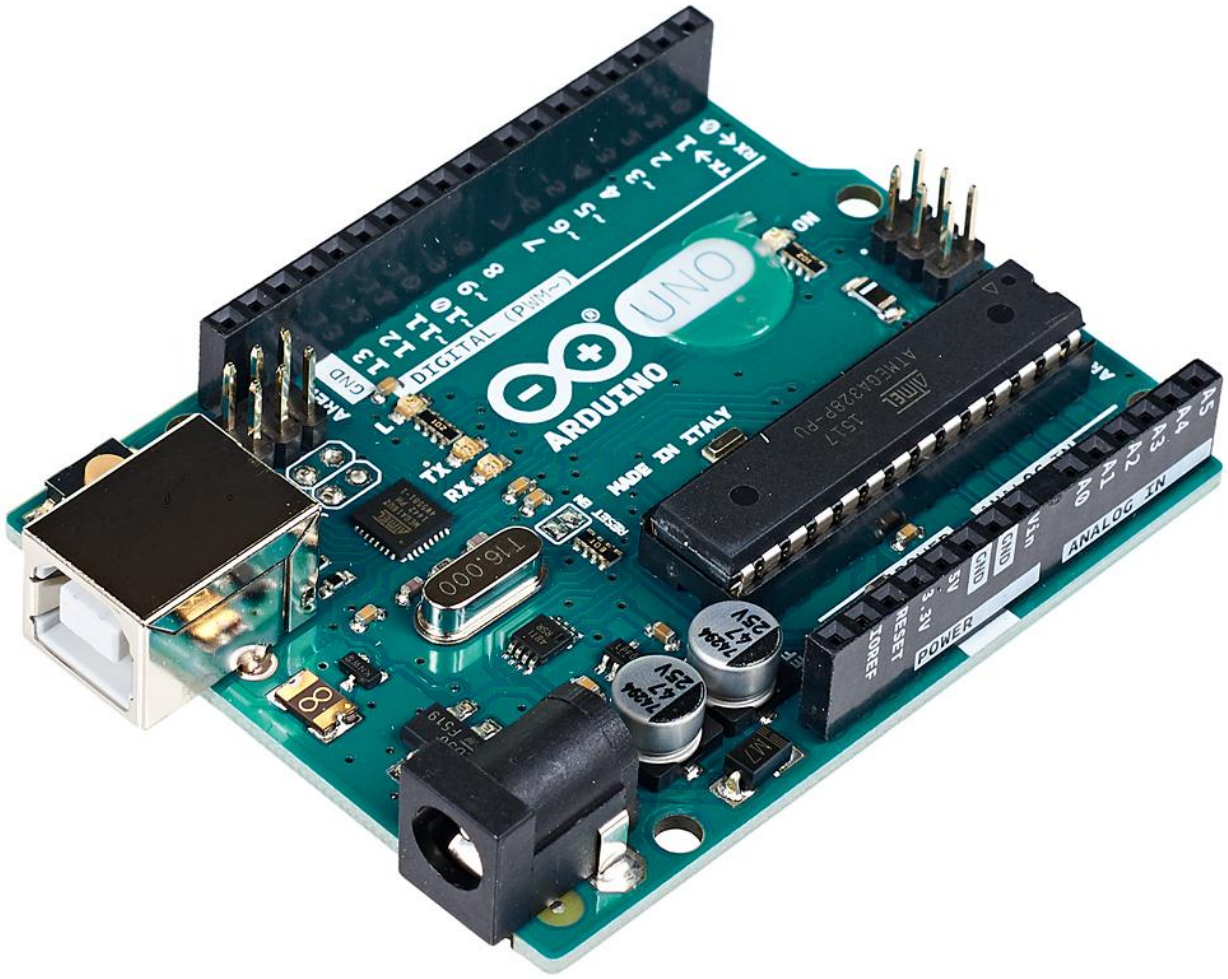

Fig. 1.1.1. Arduino UNO electronic circuit board

So, the job of this chip is to execute the code, which is the center of the whole system. The microcontroller reads data from other components, and it also controls other components. All ICU components should be energy efficient and cheap, and often do not require much performance like 6 kHz processors and 1 GHz. If you need to engage with a specific user, then performance is probably not needed, so a low-energy, low-energy processor is usually needed for the vehicle..

It is now quite common to use a 16 MHz band, although it may be even less, such as 8 or even 4 MHz. It is almost several hundred times slower than a regular computer processor. Of course, faster ICs can use up to 500 MHz or even up to 1 GHz. This is required, for example, to process audio and video. Another difference of MPS from ordinary computers is that in ordinary computers the processor and memory are separate circuits, and usually the memory is a few chips. We can modify and add memory regardless of the processor. In entry-level microcontrollers

everything is combined into one chip, that is, on one chip is a microprocessor, and memory, and other necessary components.

More productive microcontrollers use an external memory that is installed separately. This is less convenient, but allows you to choose the amount of memory required for a particular vehicle. This is usually the case when the microcontroller requires a large amount of memory to operate.

The hardware architecture of the microprocessor can not be changed, but in its memory you can download the program in the form of a set of instructions written in a language he understands. There are many programming languages in which you can write similar programs. The C language is used to program the microcontroller installed in the vehicle on the Arduino UNO board (Fig. 1.2).

Before executing, the program obviously needs to save the code. Therefore, the memory of the program must be inside the microcontroller. Now, this is usually a flash drive of the same type as in USB drives. Flash memory is an independent memory, that is, data that is not erased after the power is turned off. When power is supplied to the vehicle, the microcontroller starts the program from the beginning.

Applications for similar devices are usually written on a regular computer. This is due to the fact that the microcontrollers are relatively slow and weak. Therefore, the program is written and compiled on a powerful computer, and then the final version of the compiled program is written to the memory of the microcontroller. This is done with the help of a special device - the programmer (Fig. 1.1.2).

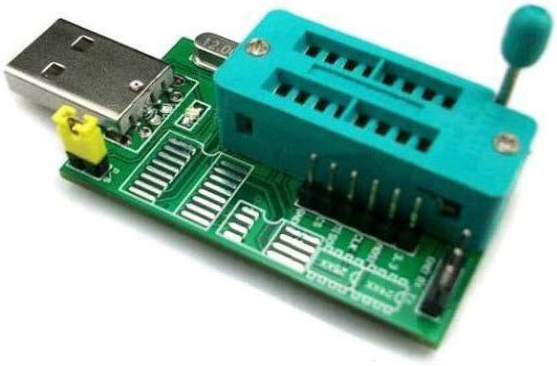

Fig. 1.1.2. Programmer

The programmer is a small board that connects to a computer via a USB port and plugs a microcontroller into the slot. There are special programs on your computer that communicate with this device and write the application through it to the memory of the microcontroller. This is one way to program a microcontroller, after which the microcontroller is programmed separately using a programmer, and then it is already installed in the MPS where it will work. Sometimes it happens that the MPS already has a special connector for programming, and then the microcontroller is programmed directly into the system. This special programming connector is used in the Arduino UNO board. It does not require a separate programmer, since it is already built into the board. You can simply grab an Arduino-based vehicle and plug it into your computer's (PC) USB port to program it directly in the Arduino UNO (Arduino Genuino) programming environment.

Arduino UNO board background. The Arduino Uno board is a device based on the ATmega328 microcontroller (Fig. 1.1.3 and Fig. 1.1.4). It includes everything you need to work comfortably with the microcontroller: 14 digital inputs / outputs (6 of which can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz resonator, a USB connector, a power connector , ICSP connector, and reset button. To get started, you must power the AC / DC adapter or connect it to your computer using a USB cable. To work with the Arduino Uno board in Windows, you need to install an Arduino IDE (Integrated Development Environment) on your computer.

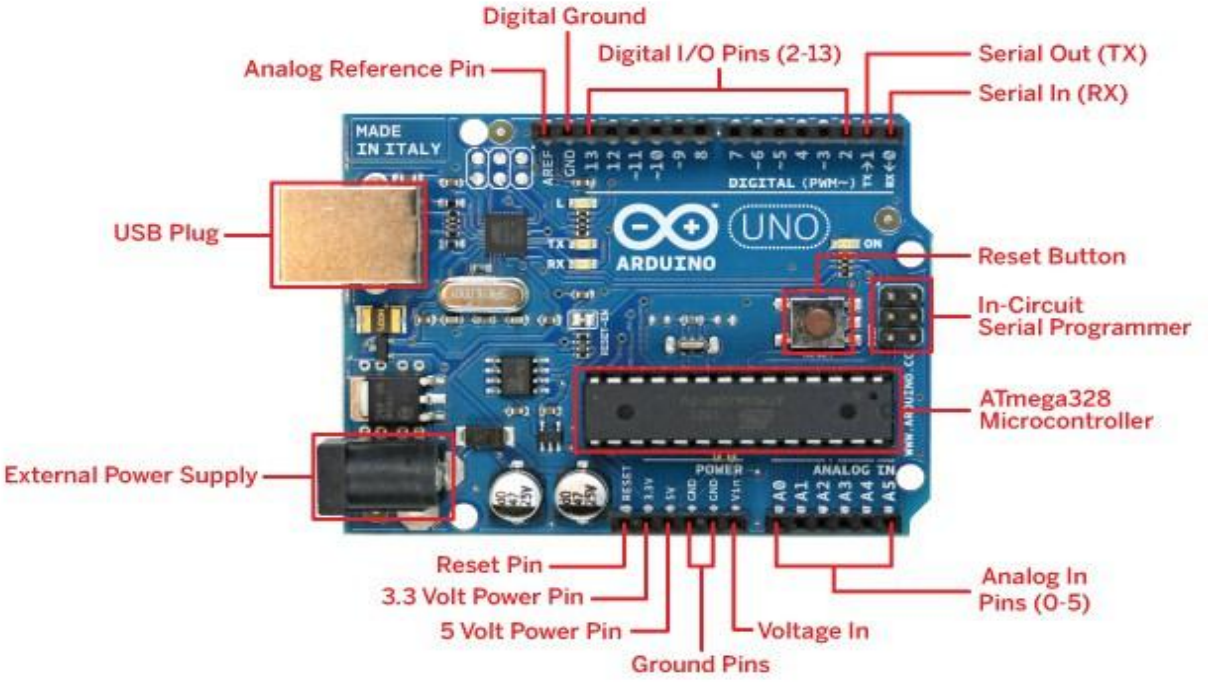

Fig. 1.1.3. Description of the elements of the Arduino UNO board

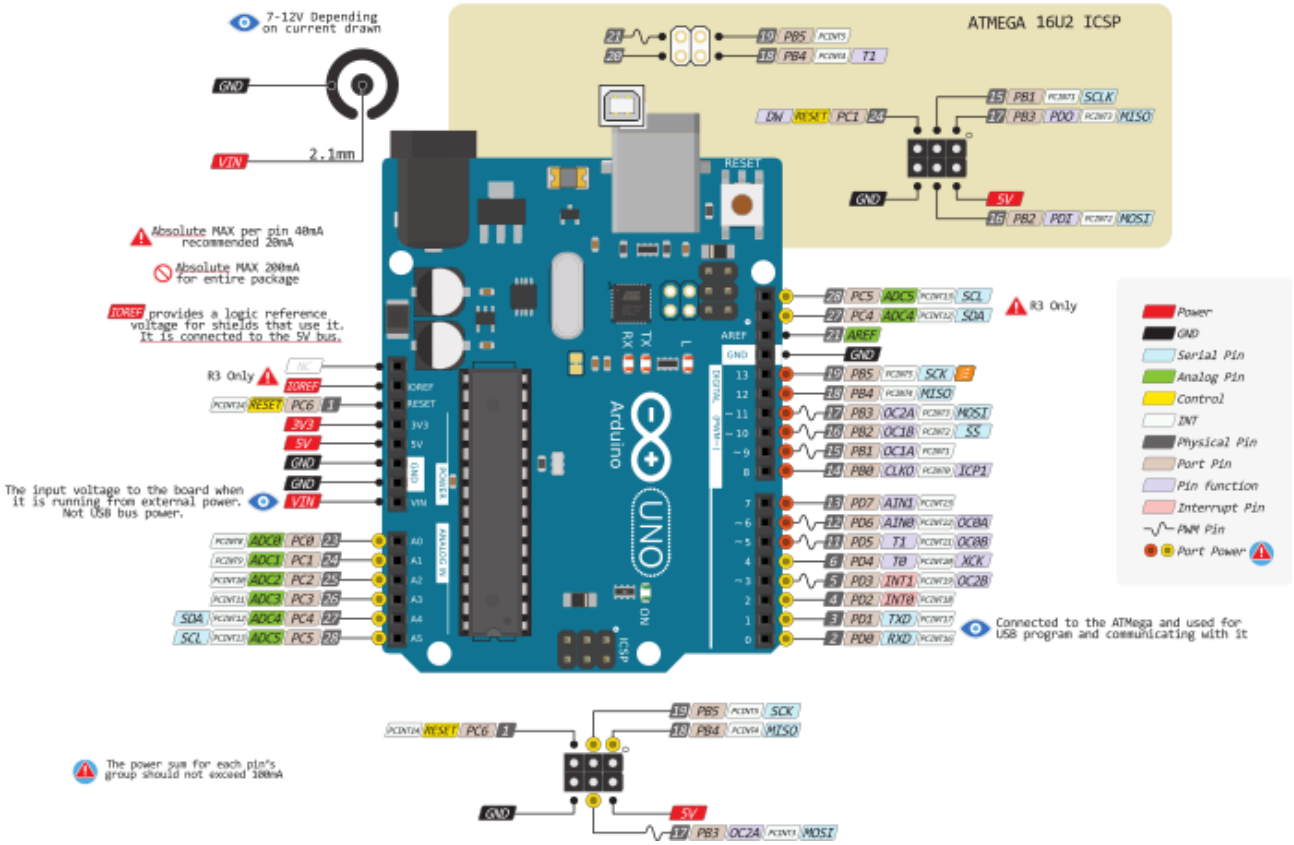

Fig. 1.1.4. Purpose of Arduino UNO board pins

Parametrs of the Arduino UNO board:

Microcontroller	ATmega328
Working voltage	5 V
Supply voltage (recommended)	7-12 V
Supply voltage (limit)	6-20 V
Digital inputs / outputs	14 (6 as PWM outputs)
Analog Inputs	6
Maximum current of one output	40 mA
Maximum output current of the 3.3 V output	50 mA
Flash-memory	32KB of which 0.5KB is a downloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock frequency	16 MHz

Входи і виходи плати Arduino UNO:

– Послідовний інтерфейс: виходи 0 (RX) і 1 (TX), що використовуються для отримання (RX) і передачі (TX) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними виводами мікросхеми ATmega8U2 на платі Arduino UNO, яка виконує роль перетворювача USB-UART.

– Зовнішні переривання: виводи 2 і 3, які можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих виводах.

– ШІМ-виводи 3, 5, 6, 9, 10 і 11 – інтерфейс SPI: висновки 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).

– Світлодіод: 13 представляє вбудований світлодіод, приєднаний до висновку 13. При відправці значення HIGH світлодіод включається, при відправці LOW - вимикається.

– 6 аналогових входів (A0 - A5), кожен з яких може уявити аналогову напругу у вигляді 10-бітного числа (1024 різних значення). За замовчуванням, вимір напруги здійснюється щодо діапазону від 0 до 5 В.

На платі Arduino UNO розташовані ще кілька додаткових виводів:

– Reset: формування низького рівня (LOW) на цьому виводу призведе до перезавантаження мікроконтролера. Зазвичай цей висновок служить для функціонування кнопки скидання на платах розширення.

– 3V3: напруга на даному виводі +3.3 В, що генерується вбудованим регулятором на платі. Максимальне споживання струму 50 мА. Від цього виводу можуть живитися деякі апаратні модулі.

– 5V: напруга на даному +5 V output generated by the integrated controller on the Arduino UNO board.

- GND: land or minus total. This output is the second mandatory output for any other device.

- Vin: Used to supply external power in the absence of 5V power from the USB connector, for example, when you need to run an Arduino card separately from your computer.

In addition, some of the analog inputs have additional features:

- TWI / I2C: A4 or SDA pin and A5 or SCL pin.

- AREF: The reference voltage at which one of the nodes of the microcontroller operates is an analog-to-digital converter that converts the voltage in volts to numbers.

The voltage level at the terminals is limited to 5V, and the maximum current that can output or consume one output is 40 mA. All terminals are connected to the internal tunable resistors of 20-50 кміна.

External power (not USB) can be supplied via a Vin connector or an XP14 expansion card connector. The platform can operate at an external voltage of 6 V to 20 V, but it is recommended to use a voltage in the range of 7 - 12 V to prevent overheating or unstable operation.

Some Arduino UNO pins may have additional functions:

- Using the pinMode (), digitalWrite () and digitalRead () functions, each of the 14 digital outputs can work as input or output.

- Using the analogWrite () function, 8-bit analog PWM values can be output.

- A 5V reference voltage for analog AREF inputs can be activated by the function analogReference ().

- External interrupts on pins 2 and 3 can be enabled by the attachInterrupt () function.

- With the use of the SPI library PWM outputs can communicate with the SPI interface.

- Using the Wire library, these outputs can communicate with the TWI interface.

Schematic diagram of the Arduino UNO card is presented in Fig. 1.1.5.

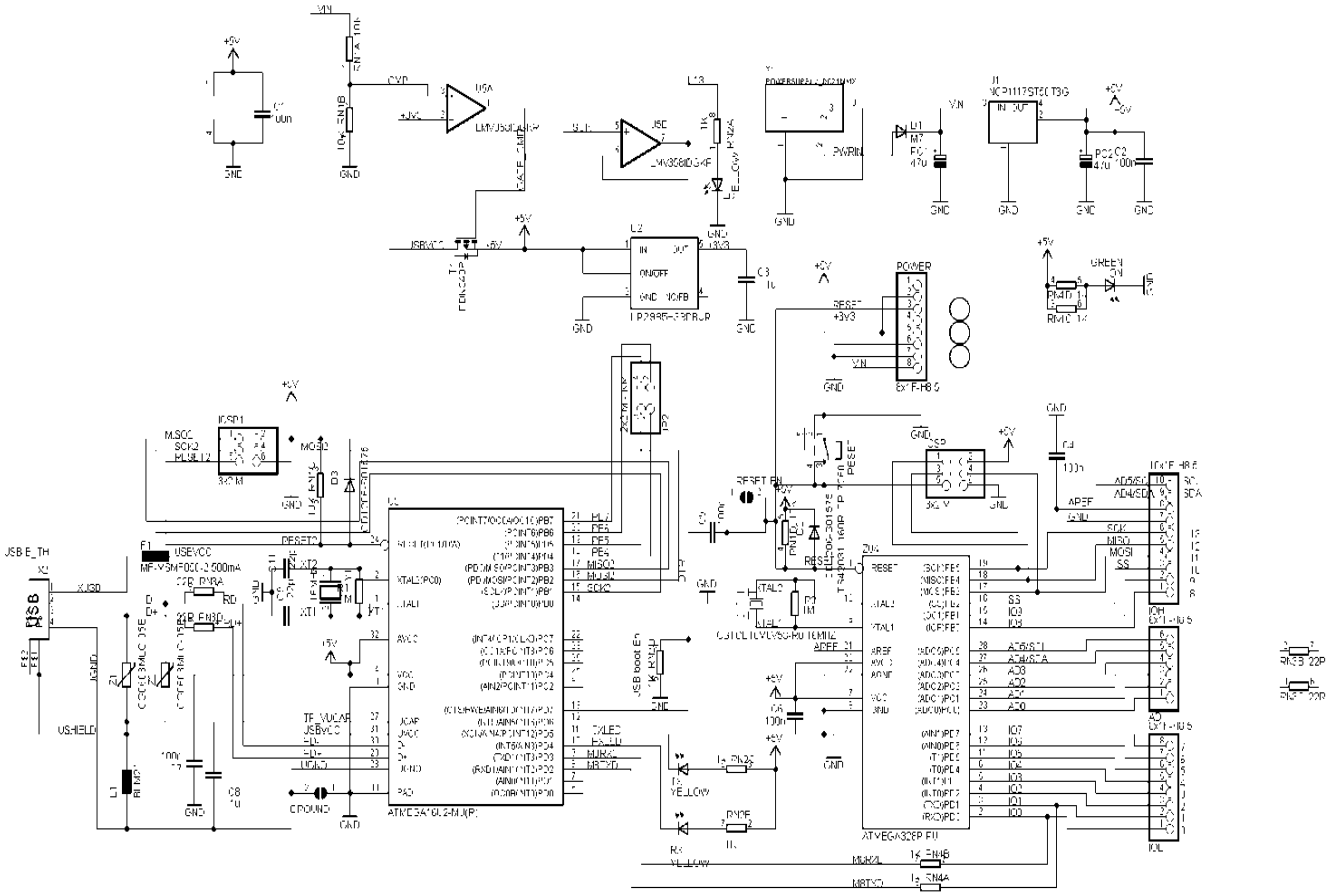

Fig. 1.1.5. Schematic diagram of the Arduino UNO

The ATmega16U2 microcontroller provides connection of the ATmega328P microcontroller to the USB port of the computer. When connected to a computer, the Arduino Uno is defined as a virtual COM port. The 16U2 chip firmware uses standard USB-COM drivers, so you do not need to install external drivers.

1.2. The Arduino development environment

All the features of the Arduino IDE (Integrated Development Environment) are available for use if an actual Arduino board is attached (Fig. 1.2.1).

The menu items of the Arduino IDE Editor (Fig. 1.2.1) include the following basic elements: file, edit, sketch, service, and help.

In detail, each menu consists of the following items:

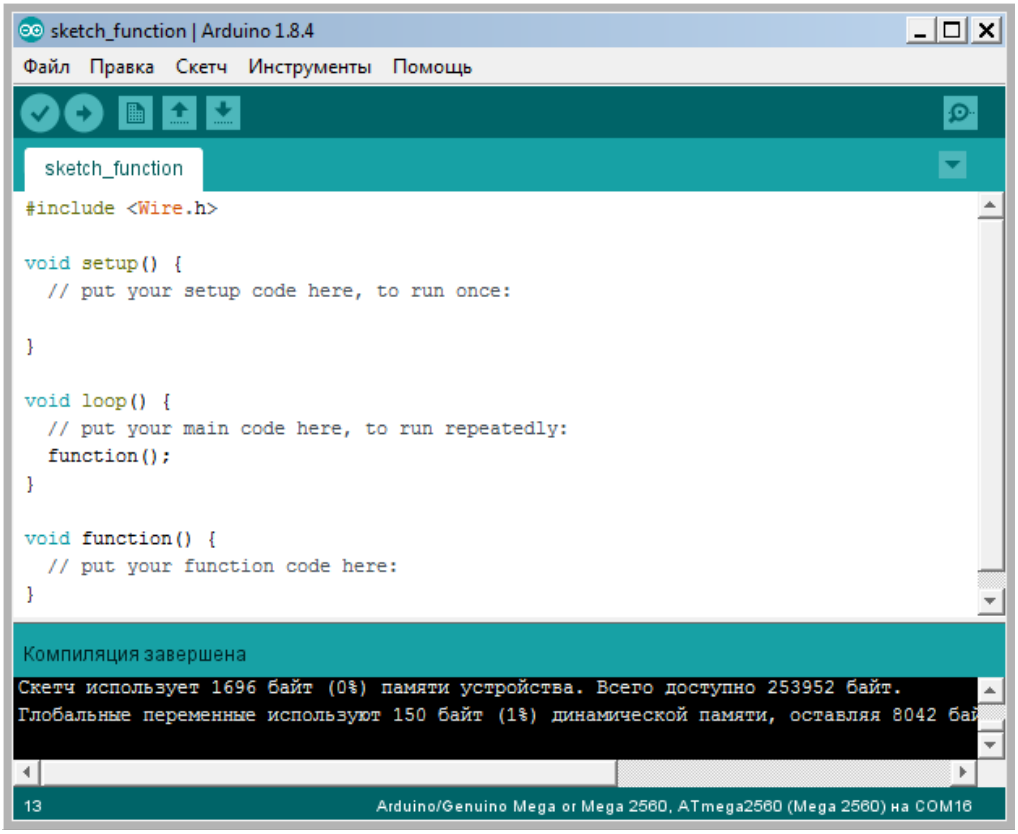

Fig. 1.2.1. Arduino Development Window Areas

1. The File menu contains commands responsible for creating a new program, downloading an existing program from the disk, saving changes, and commands for downloading the program to the microcontroller:

- "Create" - create a new program (in this environment programs are called sketches);
- "Open" - open an existing program;
- "Folder with sketches" - open the program from the specified folder;
- "Examples" - open an example program;
- "Close" - close the current window;
- "Save" - save changes;
- "Save As" - save the program in a new file;
- "Download" - download the program in Arduino;
- "Download using the programmer" - download the program using the programmer;
- "Print settings" - setting the printer;
- "Print" - output of the program code;
- "Settings" - settings of the editor;

- Exit - Exit Arduino IDE.

2. The Edit menu contains commands related to editing program text, which include: copying, pasting, indenting, and searching.

3. The Sketch menu contains commands to control the program's compilation process:

- "Check / Compile" - compile the program;
- "Show sketch folder" - opens the system folder with programs;
- "Add file" - add a data file or program to the project;
- "Import Library" - to connect the library to the application from the list of installed ones.

4. The Tools menu includes auxiliary functions for working with the microcontroller itself:

- "Auto-formatting" - automatic placement of indents, line transfers, etc .;
- "Archive sketch" - archive the program folder and save the archive to the specified location;
- "Port Monitor" - open a window for communication with the microcontroller;
- "Fee" - the choice of current payment (in our case Arduino Uno);
- Serial Port - select the port to which it is connected;
- "Programmer" - choice of programs (not used);
- "Burn bootloader" - recording bootloader program in the microcontroller (we also do not use it).

5. Help menu provides a detailed description of all the features of the Arduino IDE Editor itself, as well as various commands and techniques for working with the platform.

The most commonly used functions of the Arduino IDE Editor are the buttons on the toolbar. The buttons in the toolbar are described in Table 1.2.1.

Table 1.2.1. Toolbar buttons

Buttons	Purpose
	Check / Compile the program
	Download the code in Arduino
	Create a new application
	Open an existing application
	Save the application
	Serial Port Monitor

The program text is created and edited directly in the code editor window. This window is a typical text editor with syntax highlighting. At the bottom of the Arduino IDE window is an area for displaying error messages that occur during the compilation of the application or when the application is loaded into the microcontroller.

1.3. Pairing an Arduino with LabVIEW

The Arduino UNO-based MPS supports ATmega328P microcontroller communication via a USB port of a computer with an interface in LabVIEW 2010 (Fig. 1.3.1).

National Instruments Developer has official libraries for connecting, configuring, and running Arduino UNO with LabVIEW 2010. To support, you must install the VI Package Manager 2010 library, which can be downloaded via the FTP File Transfer Protocol: NI LabVIEW Interface for Arduino Toolkit.

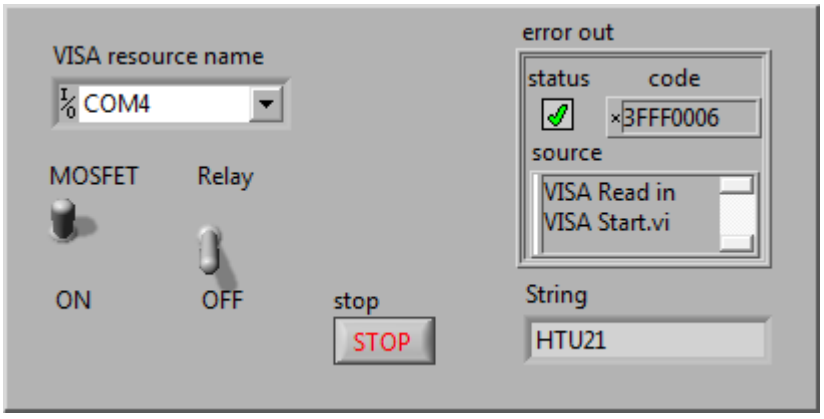

Fig. 1.3.1. LabVIEW 2010 interface to support Arduino UNO board

The purpose of this interface in LabVIEW is to support a serial communication terminal that allows the user to send a read / write command to an Arduino controller for communication via a USB port. This situation occurs when the client must be able to operate the device without resorting to Arduino reprogramming. In addition, using the interface will prevent the user from entering incorrect data, which can lead to dangerous situations in the actual design.

Creating a communication channel in LabVIEW for the Arduino UNO controller begins with the use of the VI component "VISA Configure Serial Port", which is intended to configure the serial port (Fig. 1.3.2).

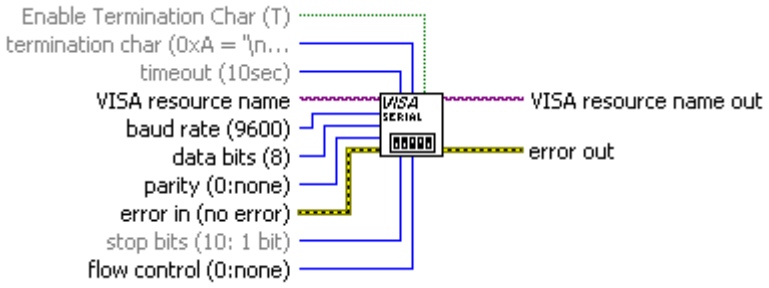

Fig. 1.3.2. Visual component VI "VISA Configure Serial Port"

This visual component (VC) is named VISA resource, which COM port will be occupied by an Arduino UNO board (such as COM6), which is configured in the Data Communication> Protocols> Serial Palette menu.

To create an LCD, right-click on the name of the resource node and go to the menu Create> Control. The control unit should look like this (Fig. 1.3.3):

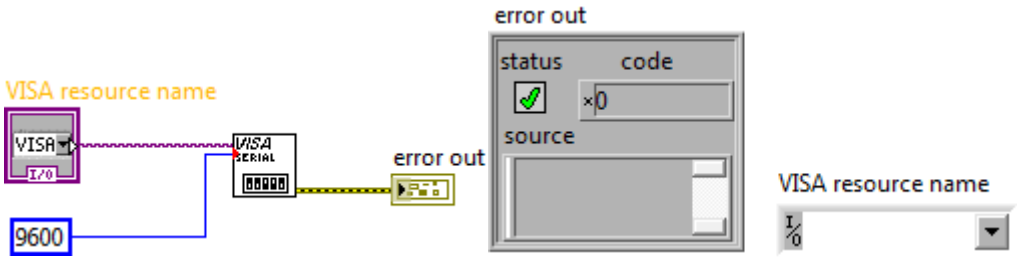

Fig. 1.3.3. Setting up VISA Configure Serial Port LCD

By renaming this block to Serial Communication Port, you can set the baud rate of 9600 by right-clicking on the baud rate and selecting Create> Constant to set it to 9600.

You can now open a message coming through a serial communication port by selecting the visual component VI "VISA Open" (Fig. 1.3.4):

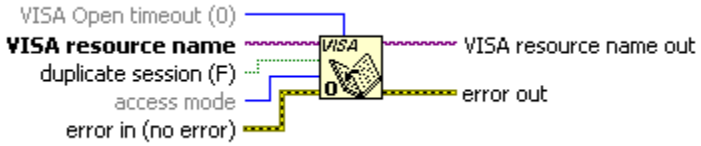

Fig. 1.3.4. Visual component VI "VISA Open"

You can find this LCD by pressing Ctrl + Space and typing "VISA Open", or via the Instrument I / O> VISA> VISA Advanced menu, as shown below (Fig. 1.3.5):

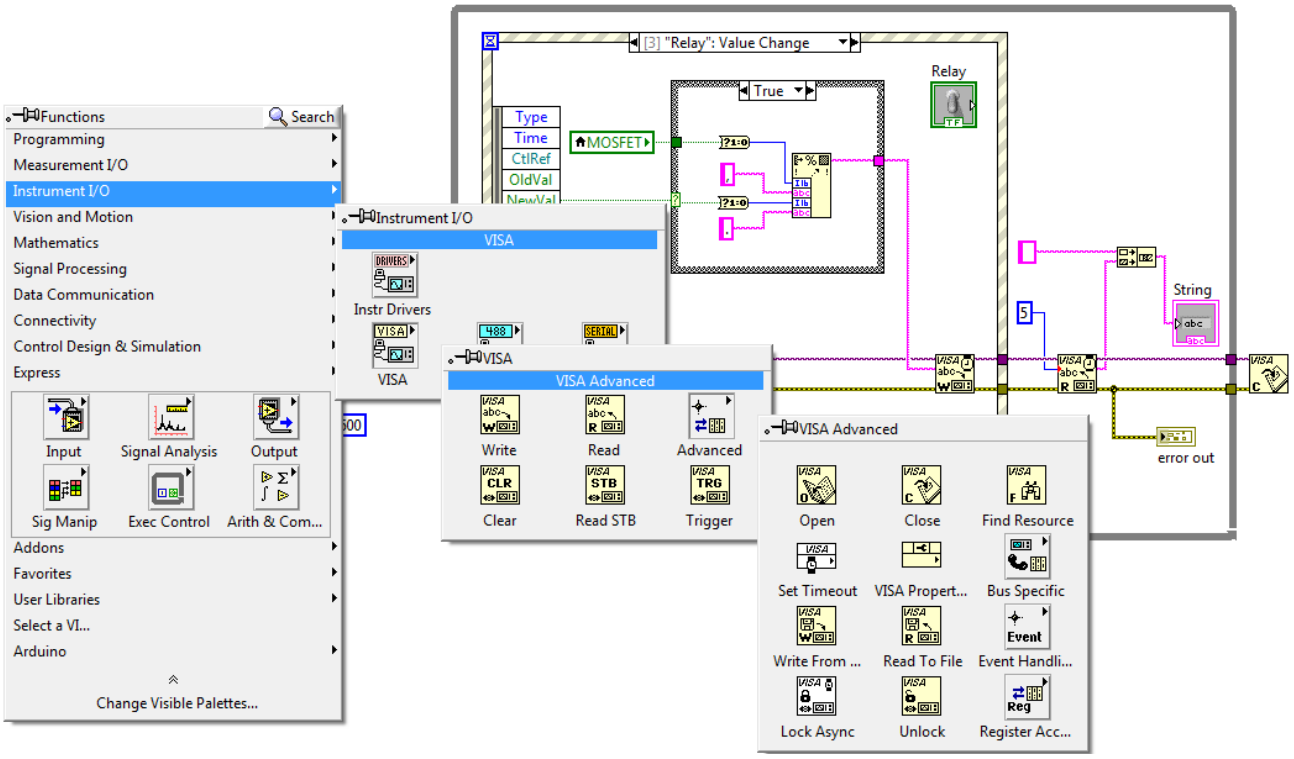

Fig. 1.3.5. Setting up VISA Open LCD

You can then create an Event Structure within the new While Loop loop and add a visual component, VISA Close VI (Fig. 1.3.6), which will close the serial communication port when the Stop button is pressed.

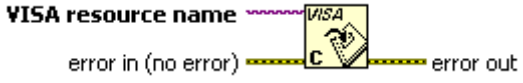

Fig. 1.3.6. Visual component VI "VISA Close"

On the front panel, you can add two toggle switches, one called "MOSFET" and the other "Relay", the levers can be found in the Buttons button palette.

Then you can add 3 events to the Event Structure:

1. Add one MOSFET toggle switch event: Value change;

2. Add one event to toggle the Relay toggle: Value change;

3. Add one event to toggle the STOP: Value change switch.

The code inside the Event Structure event blocks will be executed whenever its corresponding toggle switch or button is actuated:

1. The case of the event "MOSFET: Value change" (Fig. 1.3.7)

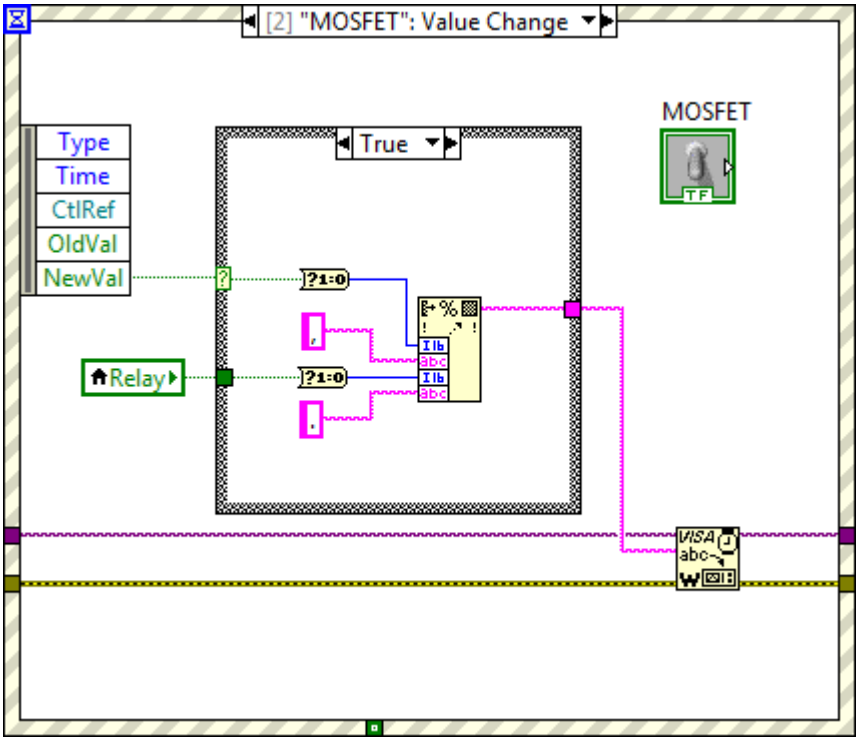

Fig. 1.3.7. MOSFET: Value change event

In the event of a MOSFET: Value change event, it is assumed that a message will be sent to the Arduino to enable or disable the MOSFET module.

The value of the MOSFET switch when switching the toggle switch gives access to the NewVal terminal in the Event Structure event structure. On the other hand, the value of the "Relay" switch gives access to the local variable "Local Variable" (Fig. 1.3.8).

Fig. 1.3.8. Local Variable Local Variable

To create a local variable "Local Variable", you must press Ctrl + Space and search for a local variable (Fig. 1.3.9).

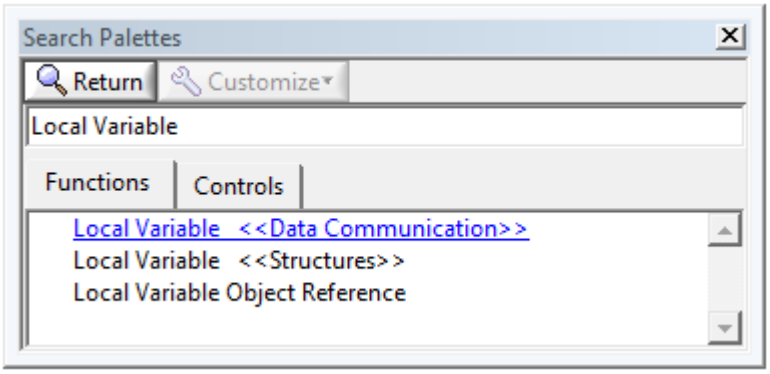

Fig. 1.3.9. Finding a Local Variable

A block should appear. Right-click on this block and select - change as follows "Change to Read" and then see that the arrow moves to the right in the Local Variable clock bar. Then left-click on the modified block and select "MOSFET" or "Relay" and the local variable will be set to the desired tumbler value (Fig. 1.3.10).

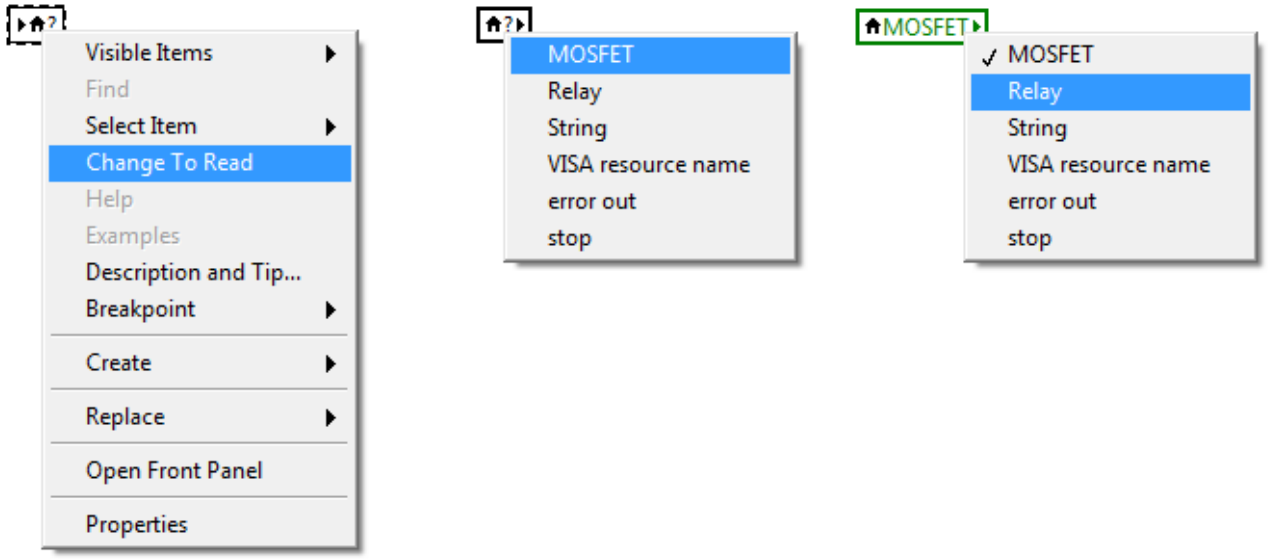

Fig. 1.3.10. Change to Read menu

Since buttons and switches in LabVIEW have a logical data type (for example, the value TRUE / FALSE), you can use the "Boolean To (0,1)" VI function to convert logical values into integers.

Moreover, the color of the wires in the connection of the components varies depending on the type of data they contain, the wires connected to the boolean I / O "Boolean input / output" will always be green. These values should then be formatted into the format format Into String (Fig. 1.3.11).

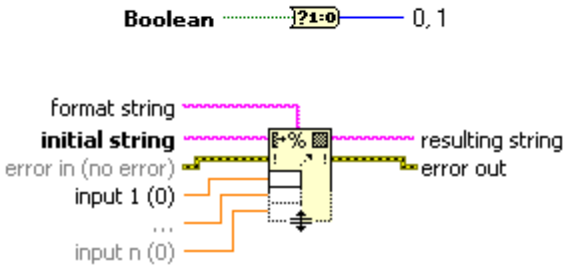

Fig. 1.3.11. Boolean Functions and Format Into String Components

The received period should be sent to the microcontroller through the component «VISA Write» VIs (Fig. 1.3.12).

Fig. 1.3.12. Visual component VI "VISA Write"

The code inside the Event Structure event blocks for the event "MOSFET: Value change" should look like this (Fig. 1.3.13):

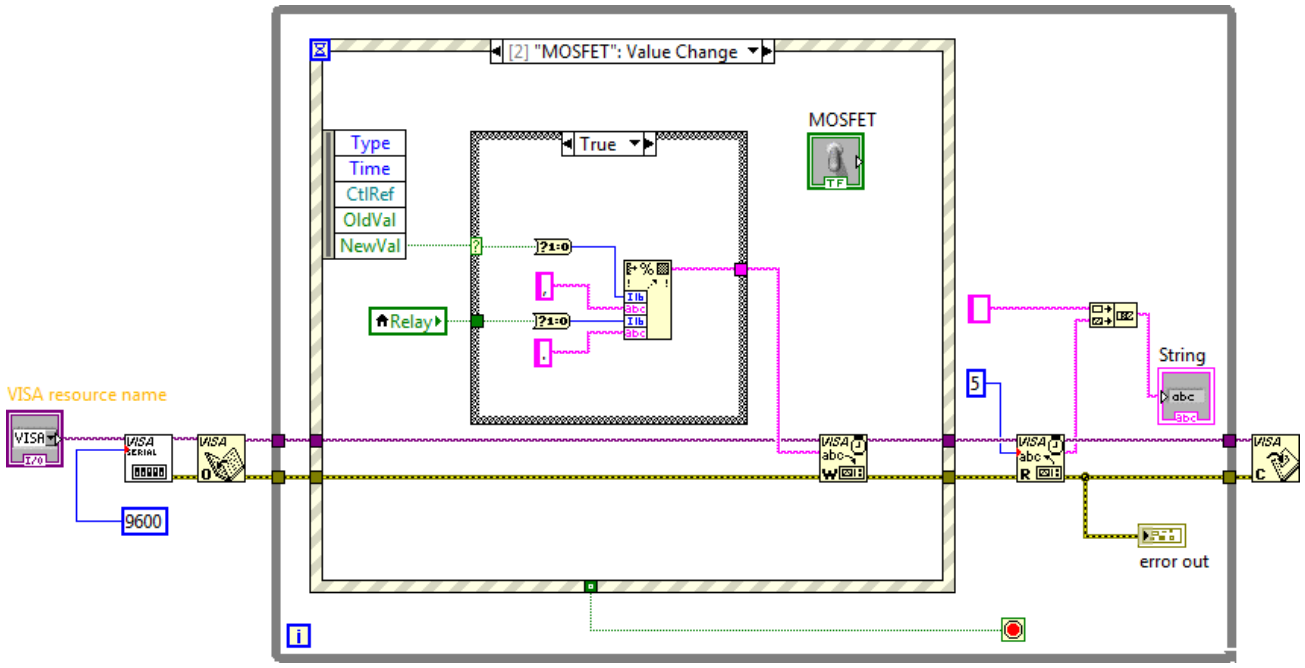

Fig. 1.3.13. MOSFET: Value change event flowchart

For the event "Relay: Value change" the code has the following form (Fig. 1.3.14):

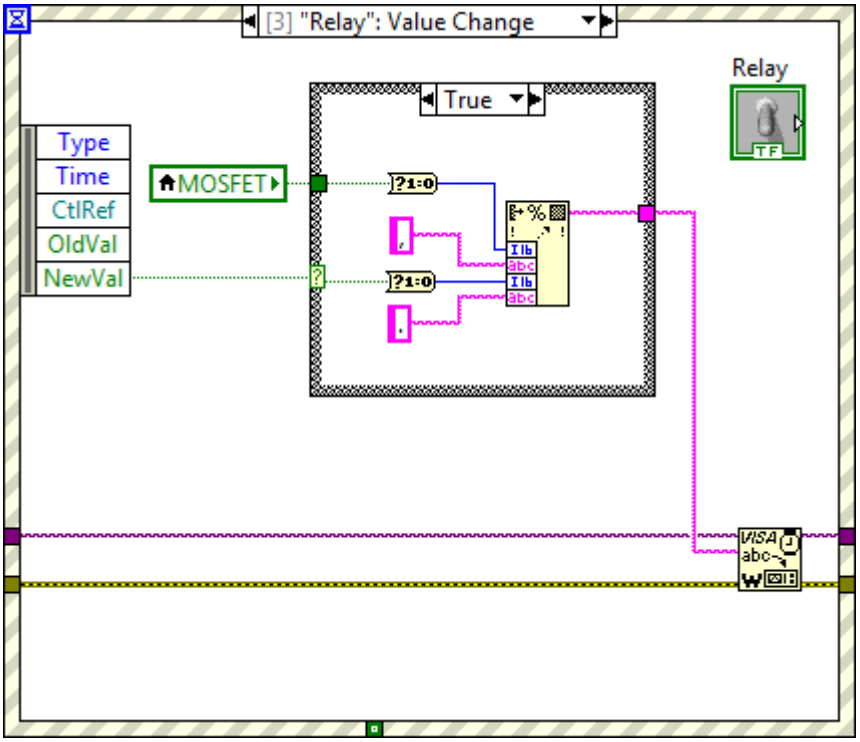

Fig. 1.3.14. Relay: Value change event

In this case, events to send messages from LabVIEW to Arduino must be enabled or disabled in the Relay: Value change module. Since buttons and switches are of Boolean data type (for example, TRUE / FALSE), you can use the Boolean To (0,1) VI function to convert logical values to integers.

Thus, the structure of the code is very similar to what we have seen in the case of the event "MOSFET: Value change", and the event "Relay: Value change" has the following form (Fig. 1.3.15):

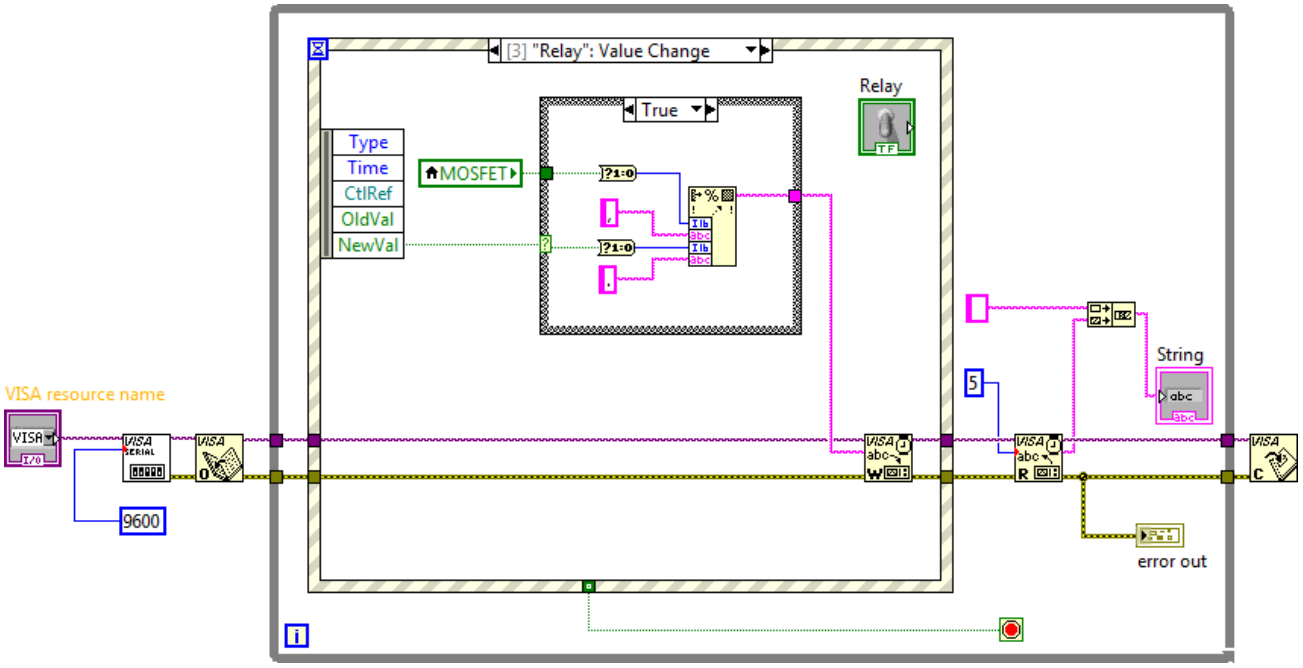

Fig. 1.3.15. Flowchart for Relay: Value change event

The STOP: Value change event should look like this (Fig. 1.3.16). This case is very simple: you need to move the stop button "STOP" inside the block of event structure "Event Structure" and move the button output to the terminal of the condition of the loop "While Loop". Pressing the STOP button interrupts the While Loop cycle and closes the program.

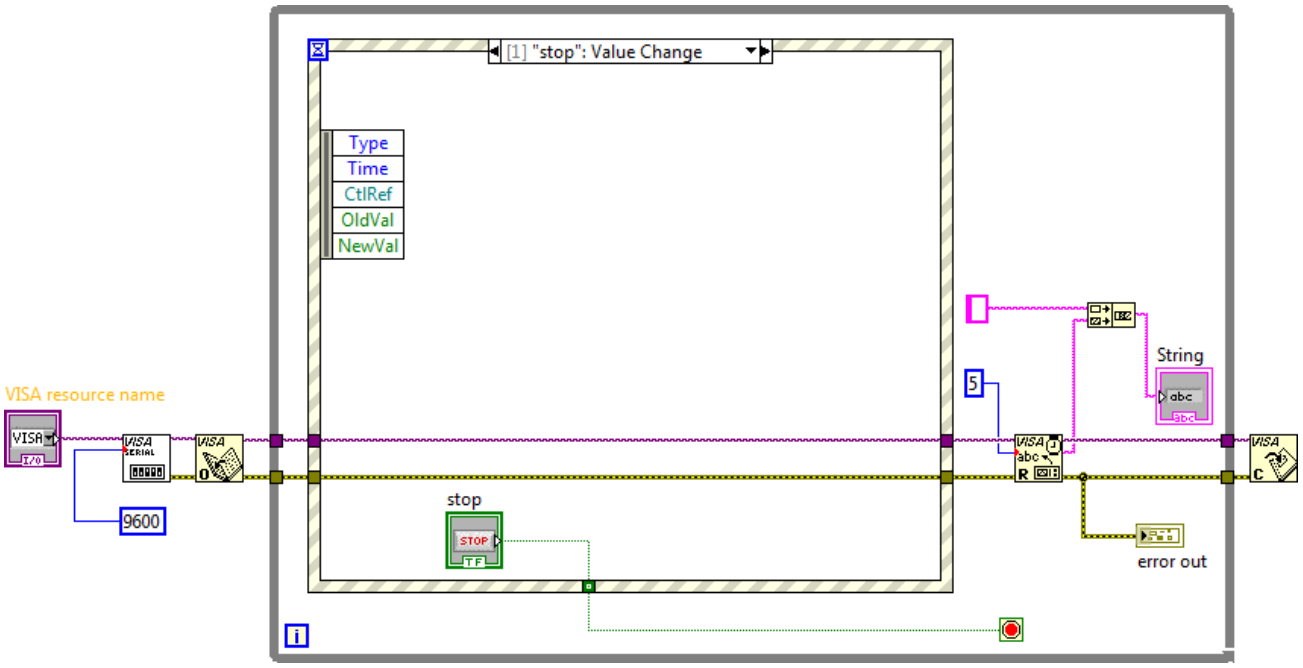

Fig. 1.3.16. Flowchart for the STOP: Value change event

The case of Timeout Event can be left with the code not written (Fig. 1.3.17). This structure can realize a time delay while waiting for an event message. The value of the Timeout terminal in the upper field of the Timeout Event structure indicates the number of milliseconds that is expected when waiting for an event.

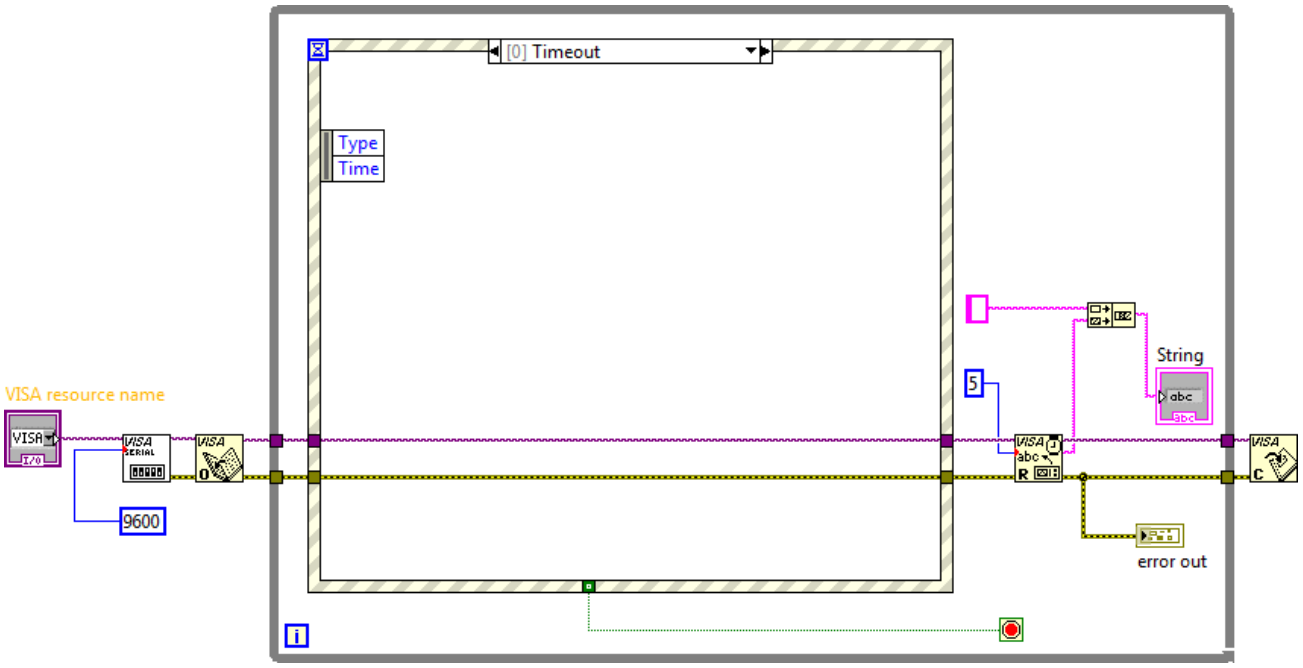

Fig. 1.3.17 Timeout Event Flowchart

The default timeout value is -1, indicating that the timeout is not set.

When testing a program in LabVIEW, if all component outputs are connected correctly, the program must start. You must select the COM port of the Arduino connection and enable / disable the modules by clicking on the "MOSFET" and "Relay" toggle switches. You can stop the program by pressing the STOP button.

1.4. Program structure and features

Programming in the Arduino environment is done in C language with a minimal set of C ++ elements. In the following, for the sake of brevity, we will call programming language in the Arduino environment C language. This language is very similar to C language, but has some differences. The program for Arduino must be written in one file, called the sketch or program "sketch for Arduino". The structure of the sketch has differences from the structure of a normal C program, which is related to the difference between the program running on a regular computer and in microprocessor systems.

When running a program on an ordinary C-computer, main () is responsible for executing the program. When you start the program, this function is started, from which other functions of the program can be called. When main () returns control with return, the program shuts down.

But for a program that manages a microcontroller, this behavior is pointless. While the vehicle is powered, the microcontroller must control the device, and therefore must constantly execute its program. This program never ends on its own, it only stops working if you turn off the microcontroller. Therefore, the structure of the program for Arduino is different from the structure of the normal program in C language. There is no main () function in the program for Arduino, instead there are two functions in the microcontroller: setup () and loop ().

The setup () function is called once at the very beginning of the program execution immediately after the Arduino microcontroller is turned on. This function configures the microcontroller, in particular, which outputs will be used as inputs and which outputs will be used as outputs and so on. This feature can otherwise be used to prepare the device for work, for example, to include additional features, or vice versa, to make sure that nothing else is turned on.

The loop() function runs all the time while the Arduino microcontroller is running. When completed, the Arduino environment restarts it. It is in this function that you need to implement the algorithm of the device. Usually, the loop () function interrogates the sensors connected to the microcontroller or the microcontroller inputs themselves, analyzes the information received and issues commands to the actuators.

We can assume that the Arduino library already has a main () function, which can be arranged as follows:

```
void main () {
        setup ();
        while (true)
        loop ();
}
```

This main() function does indeed exist in the Arduino library, but it is more complicated. In the Arduino environment, just like normal programs, you can create any other user functions and call them from the setup () and loop () functions. If you develop a simple program, you can write all the code just inside the setup () and loop () functions. However, if you do something difficult, then you should break the program code into separate functions.

In Arduino you can call the standard functions that are in the library. You do not need to add #include directives. In standard C language, this is required even for standard functions. One of the features that does not require the addition of #include directives is the delay (time) function, which performs a delay for the specified time.

An example of a delay (time) function that performs a delay for a specified time, set in milliseconds, is:

```
delay (1000);
```

There are several standard features in the Arduino program that you can write to the program immediately. These functions control the basic capabilities of the microcontroller.

The great advantage of the Arduino environment is the large number of expansion boards from various manufacturers that can be connected to the Arduino and used with the vehicle. To make these cards convenient to use, libraries are written for them, which contain convenient features for working with these devices. Such libraries are separate components that can be connected to an Arduino board.

To use the library, you must put the #include directive at the beginning of the program:

#include <library_name.h>

For example, a display library is called LiquidCrystall, and you must first write to use it:

#include <LiquidCrystal.h>

In C, in Arduino, C ++ elements are used to work with libraries. In fact, the libraries themselves write in C ++, but in sketch you can use only very simple features of this language. C ++ allows you to create classes. The classes contain user-defined data types that describe the set of fields and operations that can be done with them. Using this mechanism in C ++, for example, you can make a NewLiquidCrystall class that inherits the properties of the LiquidCrystall class and an object that belongs to that class. To make such an object, you must declare it as normal variables declare. For example, to use a display, you need to make a variable of the type LiquidCrystall:

LiquidCrystal lcd (4, 5, 6, 7, 8, 9);

In this example, LiquidCrystall is the type name that is described in the documentation in the LiquidCrystall library. The lcd variable is the name of an object that can be used as any variable. Numbers are parameters that are passed to an object when created. The parameters to be specified are also described in the documentation. For the display this is the pin numbers of the microcontroller to which it is connected.

To handle objects, they use special functions that are only available for this variable and represent class methods. Invoke class methods in the following way:

variable_name. function_name (options);

For example, text displays a print () function that can be called as follows:

lcd.print ("Hello!");

The example first specifies the name of the variable, so the print () function is called just for that variable. This way, you can connect multiple displays and make a variable for each one.

Consider other features of the C syntax for Arduino UNO, the knowledge of which is necessary for writing programs.

Each C language expression ends with a semicolon. Example expression:

a = b + c;

The body of functions and constituent operators (if, else, for, while) are separated by curly brackets, for example:

```
if (a> 0) {
b = a - c;
}
```

String variables are separated by double quotes:

```
lcd.print ("some text");
```

The characters are in single quotes:

```
symbol = 'a';
```

Connection of libraries is carried out by means of design:

```
#include <math.h>
```

Comments on the program begin with two straight slash types:

```
// this is a comment
```

The declaration of a variable in the C language is done using the construct of the form:

```
variable_name;
```

For example, a variable declaration looks like:

```
int x, y; // declared variables x and y are of the integer type
```

Arduino applications can use the following types for numeric data:

byte - an integer from 0 to 255;

int is an integer from -32768 to +32767;

word - an integer from 0 to 65535;

long is an integer from -2147483648 to 2147483647;

float and double are floating point numbers that correspond to the float type for regular computers, the type double is equivalent to float.

Arrays in C language are declared constructions of the form:

```
element type array_name [size];
```

For example, an array ad looks like:

```
int values [10]; // an array of ten integers
```

The C language provides a type for storing characters, this type is char. In a vector and an array of type char, the last character has a code 0 indicating the end of the array.

For example, a variable-type ad like:

```
char data_str [10]; // vector with a maximum length of nine characters
```

The boolean type is used to represent the true () and false () values. In addition, any integer value can be used as a logical value, with 0 being false () and any non-zero value being true ().

The const type before declaring a variable indicates that its value cannot be changed during program execution.

The last type of data is the type of void used when declaring functions that have no arguments or that do not return a result.

In C, a standard set of mathematical operators is used, including +, -, *, and /,% is used to calculate the remainder of the division. In addition, there are special combinations with the assignment operator to reduce the record:

x + = 4; // means x = x + 4

Such forms are for all operators. To increase or decrease a given number by one is increment and decrement:

x ++; // means x = x + 1

y--; // means y = y - 1

The set of comparison operators is standard and includes operators <,>, <=,> =. Comparison is written with two equal signs (==) and inequality is recorded! =. Logical negation is written with an exclamation mark!. Logical expressions can be written using the operators presented in Table 1.4.1

Table 1.4.1. C ++ logical operations

Operation	Definition	Example	Description of a complex condition
and	&&	a == 3 && b > 4	true if both are simple conditions
or	||	a == 3 || b > 4	true, if true, at least one of the simple conditions
no	!	!(a == 3)	true if a is not 3

The conditional statement has the following syntax:

if (condition)

the code that is executed if the condition is true;

If, if the condition is true, multiple operators need to be executed, then they are combined into one block using braces:

if (condition) {

the code that is executed if the condition is true;

}

The full form of the conditional statement includes the else block, the Code in which another condition is satisfied, if the true condition is false:

```
if (condition) {
the code that is executed if the condition is true;
}
else {
code that is executed if the condition is incorrect;
}
```

The switch statement allows you to select one of the code branches depending on the value of a numeric variable, for example:

```
switch (mode) {
default:
case 0:
code for mode 0;
break;
case 1:
code for mode 1;
break;
case 2:
code for mode 2;
break;
}
```

The value of the mode variable determines at which point the code will execute the program after executing the switch statement. Next, all actions will be taken until a break occurs, which interrupts further code execution inside the switch. If the expression value does not match any of the labels, then the code is executed starting with the default label. In this example, in the absence of a break after default control will fall into "code for mode 0".

The while loop allows you to execute the program code as long as the condition is true. It has the syntax:

```
while (condition)
code that runs in a loop;
```

More complicated is the for loop:

for (initialization; condition; step)

code that runs in a loop;

In this loop, the expression specified in the initialization block is executed once before the cycle begins. Typically, it sets initial values for cycle variables. The condition is checked every time at the beginning of the next iteration of the loop and if it becomes erroneous, then the loop stops.

The "step" expression is executed after the execution of the loop body and usually increases (or decreases) the cycle variables. There are several variables in the initialization and step blocks if you write comma-separated expressions.

An example of a cycle that calculates the sum of an arithmetic progression:

int i, sum;

for (i = 0, sum = 0; i <= 10; i ++)

sum + = i;

The C program code must necessarily be broken down into functions. The following syntax is used to declare a function:

function_name function_name (options)

{

code executed within the function;

return result_function;

}

The function type describes the type of value that the function will return. For example, the standard cos function that calculates a cosine returns a float value.

A function name can take any form that starts with a letter, containing only letters, numbers, and underscores. Parameters are a list of variables that accept values passed to a function. Variables are specified by comma, first write the type of parameter, and then - its name.

The return statement is used to return a value from the function. It immediately interrupts the execution of the function, and the result of the function becomes the value specified after the return expression.

If the function does not return anything, type_function is specified as void. If the function does not pass any arguments, then you can write nothing inside parentheses or you can write the word void there.

For example, a function that sums two numbers of type int has the form:

```
int sum (int a, int b) {
int result;
result = a + b;
return result;
}
```

To call a function, it is necessary to write its name and in parentheses specify the values of the arguments, for example:

```
r = sum (3, 10);
```

C programs can declare variables outside of any function. Such variables are global and can be used in any functions that are located after the variable is declared. For example, if the led variable is used in setup () and loop () functions, it is described as global:

```
const int led = 2;
void setup () {
pinMode (ice, OUTPUT);
}
void loop () {
digitalWrite (ice, HIGH);
delay (1000);
digitalWrite (ice, LOW);
delay (1000);
}
```

One of the easiest steps a microcontroller can do is switch the status of the digital input-output contacts on an Arduino UNO board. Any of the digital outputs of the microcontroller can be programmatically switched between high (+ 5V) and low (0V).

This switch controls simple devices:

1. Switching the status of the digital input-output contacts on the Arduino UNO board is carried out by the function digitalWrite (), the call of which looks like:

```
digitalWrite (contact_number, signal_ level);
```

2. Switching the status of the analog input-output contacts on the Arduino UNO board is carried out by analogWrite (), the call of which looks like:

analogWrite (contact_number, signal_ level);

3. Reading the status of the digital input-output contacts on the Arduino UNO board is carried out by the function digitalRead (), the call of which looks like:

variable_name = digitalRead (contact_number);

4. Reading the status of the analog input-output contacts on the Arduino UNO board is performed by the function analogRead (), the call of which looks like:

variable_name = analogRead (contact_number);

The signal level parameter can take two values: HIGH (high, + 5V) or LOW (low, 0V). The contact_number parameter is the number corresponding to the contact number on the Arduino board. The variable_name parameter is a variable that records the numeric equivalent of a signal set for a given contact number on an Arduino UNO board.

Note that the same pins of the microcontroller can act as both digital outputs and digital inputs. Therefore, before using them, you need to specify in what role the contact will be used: input or output. This is done using the pinMode () function:

pinMode (contact_number, contact_node);

The argument of this function to contact_contact can be: OUTPUT and INPUT. Thus, to set the digital output (2) first to a high signal level (HIGH) and then to a delay (Delay) 1000 ms to set a low signal level (LOW), it is sufficient to execute the following program:

```
const int pin = 2;
void setup () {
pinMode (pin, OUTPUT);
digitalWrite (pin, HIGH);
}
void loop () {
digitalWrite (ice, HIGH);
delay (1000);
digitalWrite (ice, LOW);
delay (1000);
}
```

Recall that the Arduino controller is clocked at 16MHz, so it can turn on and off an LED connected to the output (2) thousands of times per second. In modern microelectronics, miniature LEDs are used for surface mounting. Such indicators, for example, are on the Arduino Uno to inform the user about the status of the system.

It is important to note that the operating voltage of the LED varies from 1.85 V to 2.5 V at the recommended current of not more than 20 mA. To prevent the current from exceeding this value, a series limiting resistor with a resistance of 200 to 500 Ohms must be added to the circuit when the LED is connected to the output of a 5 V voltage-controlled microcontroller.

2. Measuring instruments with microprocessor control

The Arduino UNO-based microprocessor system enables the implementation of microprocessor-controlled measuring devices, which may include electronic motors, sensors and sensors:

- Digital touch sensor;
- Universal sound sensor;
- Stepper motor control 28BYJ-48;
- Speed sensor with digital and analog outputs;
- HTU21 humidity and temperature sensor;
- Digital module with thermistor;
- SW-420 vibration sensor;
- Digital impact sensor;
- Hall KY-03 digital sensor;
- Tilt sensor on LM393;
- MQ-3 alcohol sensor;
- Digital fluid level sensor;
- Digital fire sensor;
- Analog-digital light sensor;
- Ultrasonic distance sensor HC-SR04;
- Digital interference sensor;
- Digital module with thermistor.

The aforementioned sensors and sensors have been used in laboratory layouts of microprocessor-based Arduino UNO cards for educational purposes in the development of measuring equipment using the Arduino software and the National Instruments LabVIEW information tool.

2.1.Digital touch sensor

The purpose of the work is to design a digital device in a Microprocessor-controlled LabVIEW to control the TTP223B touch sensor in the Arduino microprocessor programming environment.

Theoretical information

The sensor module is built on the chip of the capacitive touch sensor TTP223B (Fig. 2.1.1). Normally, the logic output of the SIG module is low. Touching the touch pad at the output sets a high logical level.

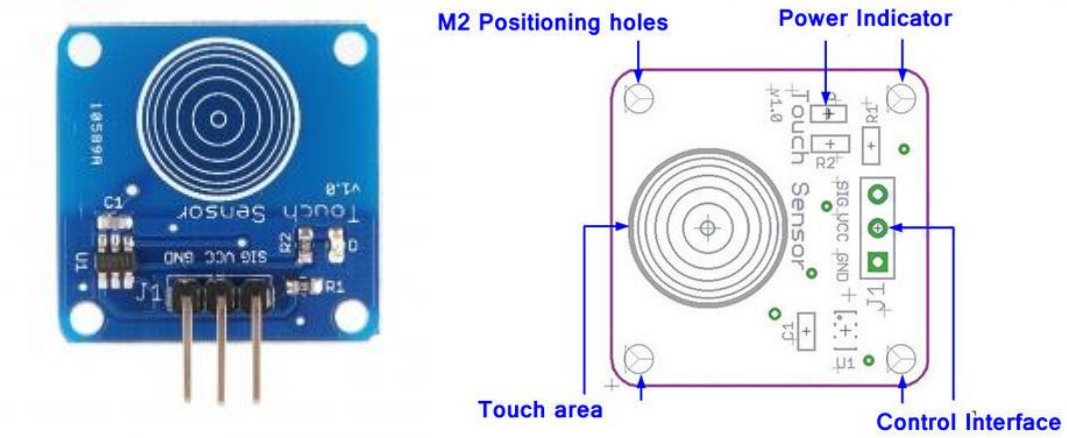

Fig. 2.1.1. TTP223B capacitive touch sensor chip

The very low power consumption combined with the low power consumption mode allows the module to be used on standalone systems. The sensitivity of the sensor allows it to be mounted on surfaces of small thickness, which is very convenient for using it as hidden buttons. The sensor operates at distances up to 4 mm.

Features of the TTP223B sensor:

Parameter	Min.	Average	Max.	Unit
Operating voltage VCC	2.0	3	5.5	V
High VOH Output Level	-	0.8VCC	-	V
Low output signal VOL	-	-	0.3VCC	V
Output current (VCC = 3V, VOL = 0.6V)	-	8	-	mA
Output current (VCC = 3V, VOH = 2.4V)	-	4	-	mA
Response Time (Low power mode)	-	-	220	ms
Response Time (Touch mode)	-	-	60	ms
Size	24x24x7.2			mm

Purpose of module outputs:

- power supply range: from 2 to 5.5V;
- interface: GND - general, VCC - power, SIG (DI) - output;
- power indicator: green LED illuminates when supply voltage is applied;
- sensor area: marked area on the module board;
- mounting holes: 4 M2 holes with a diameter of 2.2mm at the corners of the module.

The connection of the capacitive touch sensor chip TTP223B to the Arduino UNO vehicle is shown in fig. 2.1.2. If the green LED on the touch button module lights up, then the power supply is correct.

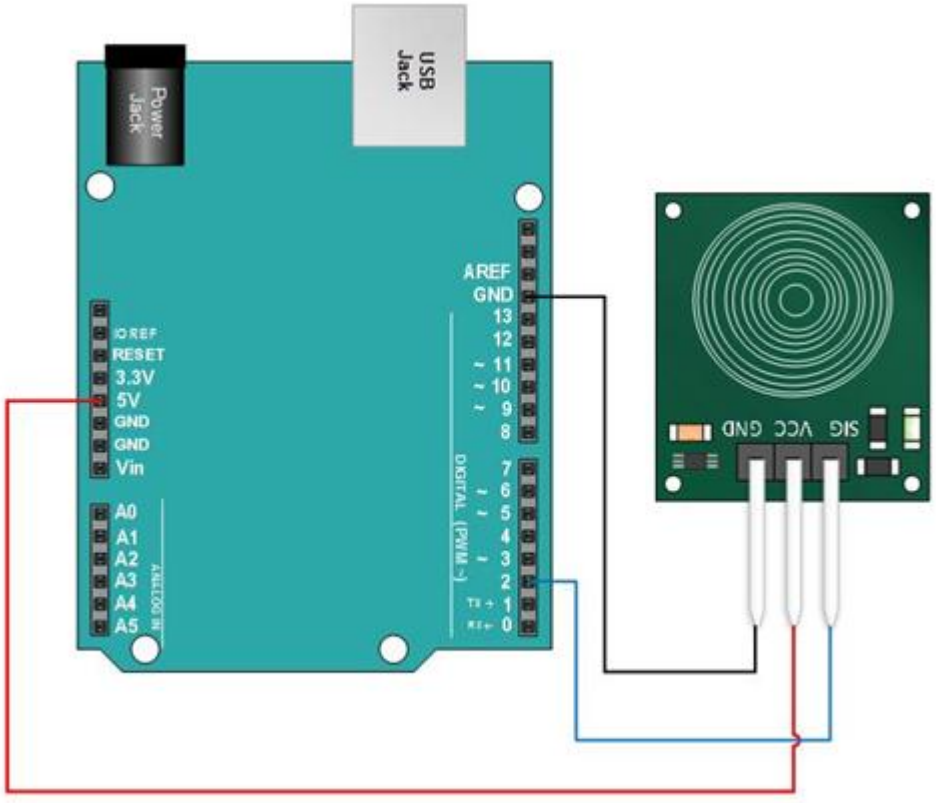

Fig. 2.1.2. Connect the capacitive touch sensor chip TTP223B to the vehicle

To connect to the Arduino UNO microcontroller, the sensor module has three contacts: for power connection (VCC), ground (GND) and output signal (SIG).

Work task

1. To compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.1.3, corresponding to the launch of the vehicle. The block diagram of the virtual device for the case of False in the structure "Case Structure" is presented in Fig. 2.1.4, which corresponds to the stop of the vehicle when pressing the button "Stop"

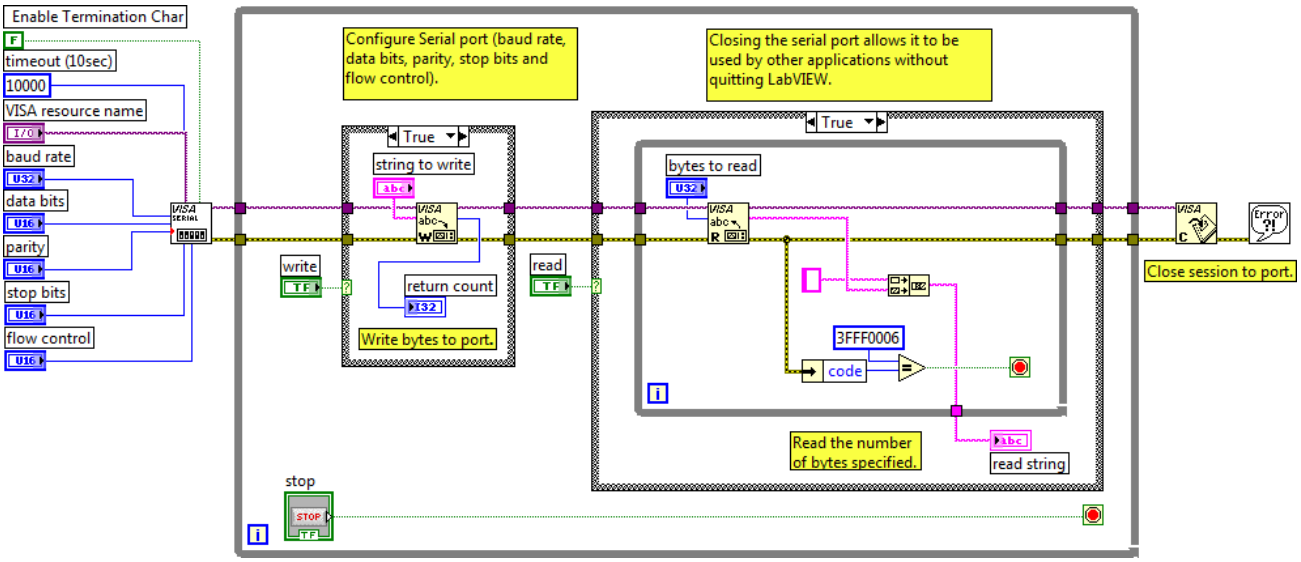

Fig. 2.1.3. Device diagram in NI LabVIEW 2010 for True case

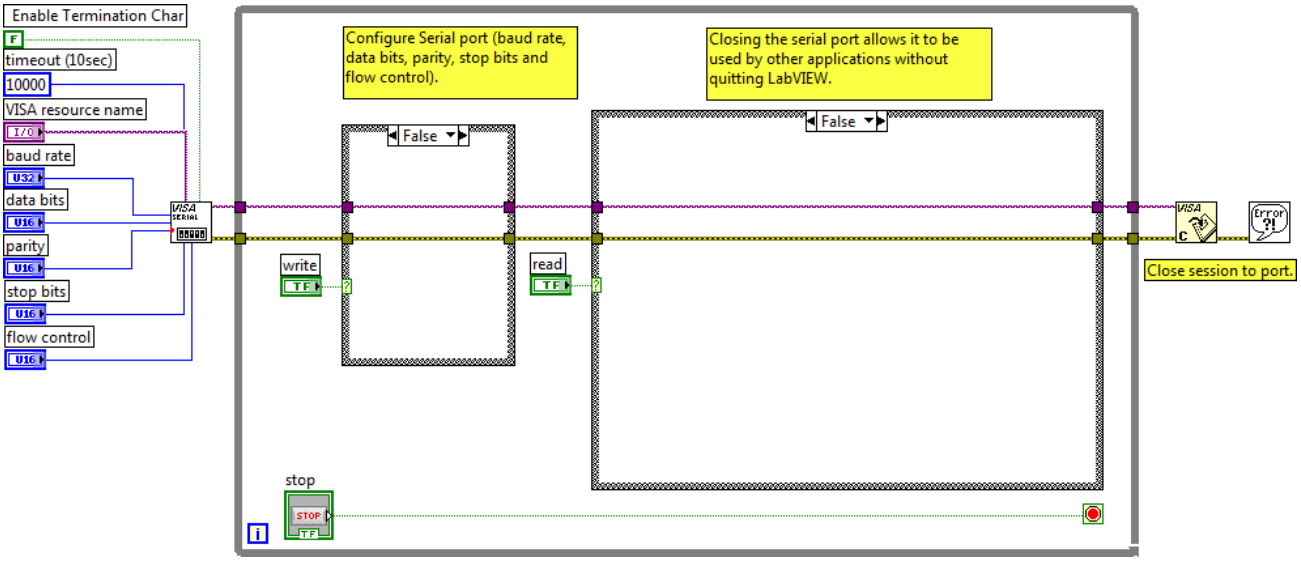

Fig. 2.1.4. Device diagram in NI LabVIEW 2010 for False case

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.1.5.

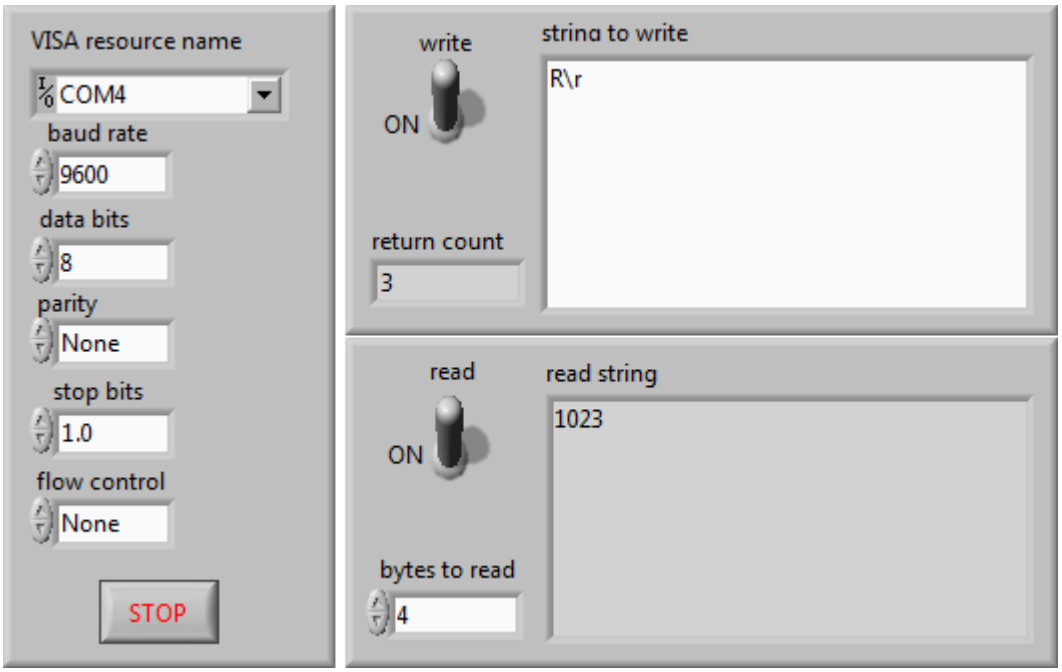

Fig. 2.1.5. The appearance of the vehicle control system

1. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

2. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

To formulate in the report on the completed work the conclusions of the research results and to prepare the answers to the control questions.

Methodical instructions

1. The National Instruments LabVIEW 2010 controls shown in fig. 2.1.3 and Fig. 2.1.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define ctsPin 2 // pin for capacitive touch sensor
#define keyPin A0
#define ledPin 5 // pin for LED
```

```
void discount (int count, int maxcount); // user function
const int maxcount = 1024;
int keyState = 0;
int count = 0;
int oldstat = 0;
int ctsValue;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (ledPin, OUTPUT);
 pinMode (ctsPin, INPUT);
pinMode (keyPin, INPUT);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
 ctsValue = digitalRead (ctsPin);
keyState = analogRead (keyPin);
 if (ctsValue == HIGH) {
 // Sig Output in high, sensor pressed
 digitalWrite (ledPin, HIGH);
 Serial.println ("TOUCHED");
 } else {
digitalWrite (ledPin, LOW);
 Serial.println ("not touched");
}
if (keyState! = oldstat) {
    count = 0;
    oldstat = keyState;
```

```
}
discount (count, maxcount);
delay (500);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
void discount (int count, int maxcount) {
if (count <maxcount) {
   digitalWrite (ledPin, HIGH);
   Serial.println ("level Count:");
   Serial.println (keyState); // display
}
else {
   Serial.println ("max Count:");
   Serial.println (keyState); // display
   digitalWrite (ledPin, LOW);
 }
 if (count <= maxcount + 1) {
   ++ count;
 }
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment into the lab layout of the MPS. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain how Serial Ports work.

2.2. Universal sound sensor

The purpose of the work is to design a digital device in a LabVIEW with microprocessor control to control the universal sound sensor in the Arduino microprocessor programming environment.

Theoretical information

The universal sound sensor is designed to detect the sound and determine the sound threshold (Fig. 2.2.1). The audio sensor has the following modules: sensitive microphone, integrated voltage comparator, analog and digital output.

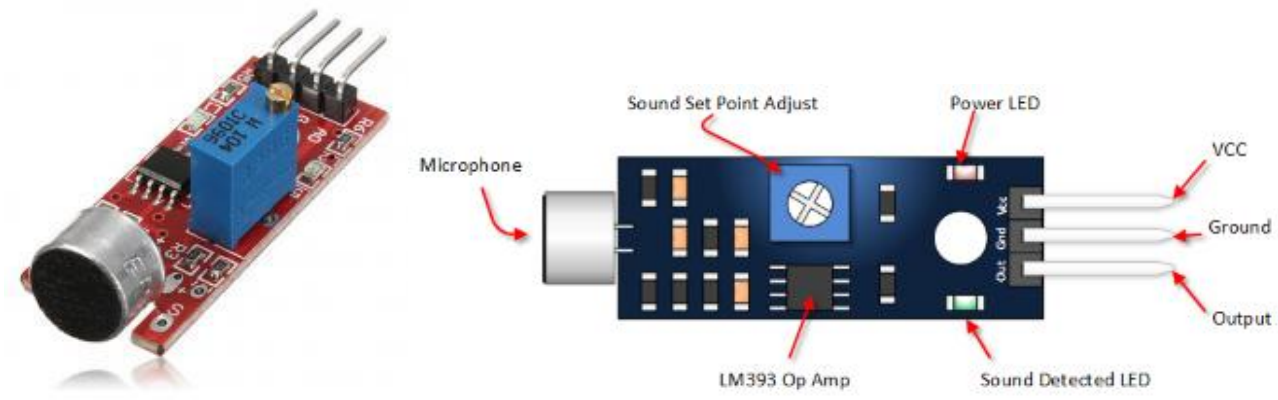

Fig. 2.2.1. Universal sound sensor

The built-in modules make this sensor convenient for use in Arduino UNO-based vehicles and robotics. The threshold of the comparator LM393 is regulated by a potentiometer.

Sensor characteristics:

- analog voltage output from the microphone;
- digital output of the threshold comparator;
- voltage comparator: LM393;
- power indicator;
- digital output status indicator;
- operating voltage: 4-6V;
- mounting hole 3 mm;
- dimensions: 32x17x8mm;

Purpose of module outputs:

- VCC: 5V DC power supply;

- Ground: land from the Arduino UNO board;

- Out: data transfer, connect to Arduino digital input;

- Power LED: power indicator that lights up when the power is turned on;

- Sound Detection LED: LED that lights up when sound is detected;

- Sound Set Point Adjust: CW - increase sensitivity; CCW - sensitivity reduction.

The connection of the chip of the universal sound sensor to the vehicle based on Arduino UNO is shown in Fig. 2.2.2.

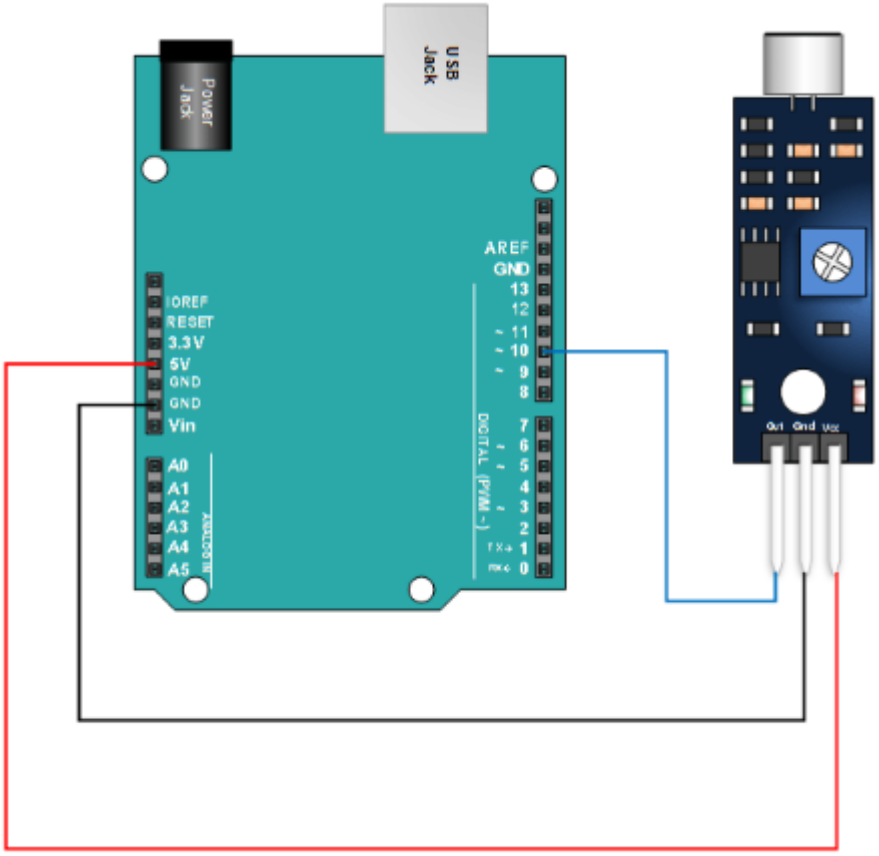

Fig. 2.2.2. Connecting the universal sound sensor chip to the vehicle

When the sound level exceeds the setpoint, the LED lights up on the sensor and the output goes low (LOW).

Work task

1. To collect the tested scheme of the virtual instrument to control the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.2.3, which corresponds to the

launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Fig. 2.2.4, which corresponds to stopping the vehicle when pressing the button "Stop".

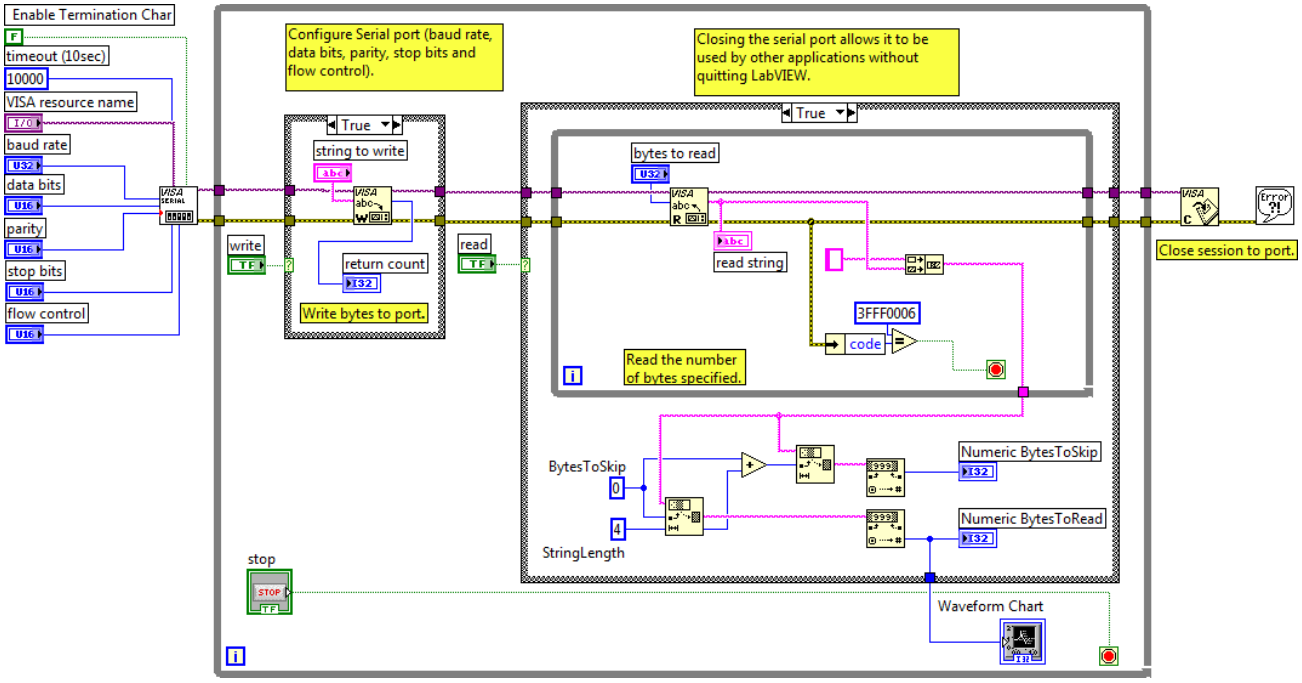

Fig. 2.2.3. Device diagram in NI LabVIEW 2010 for True case

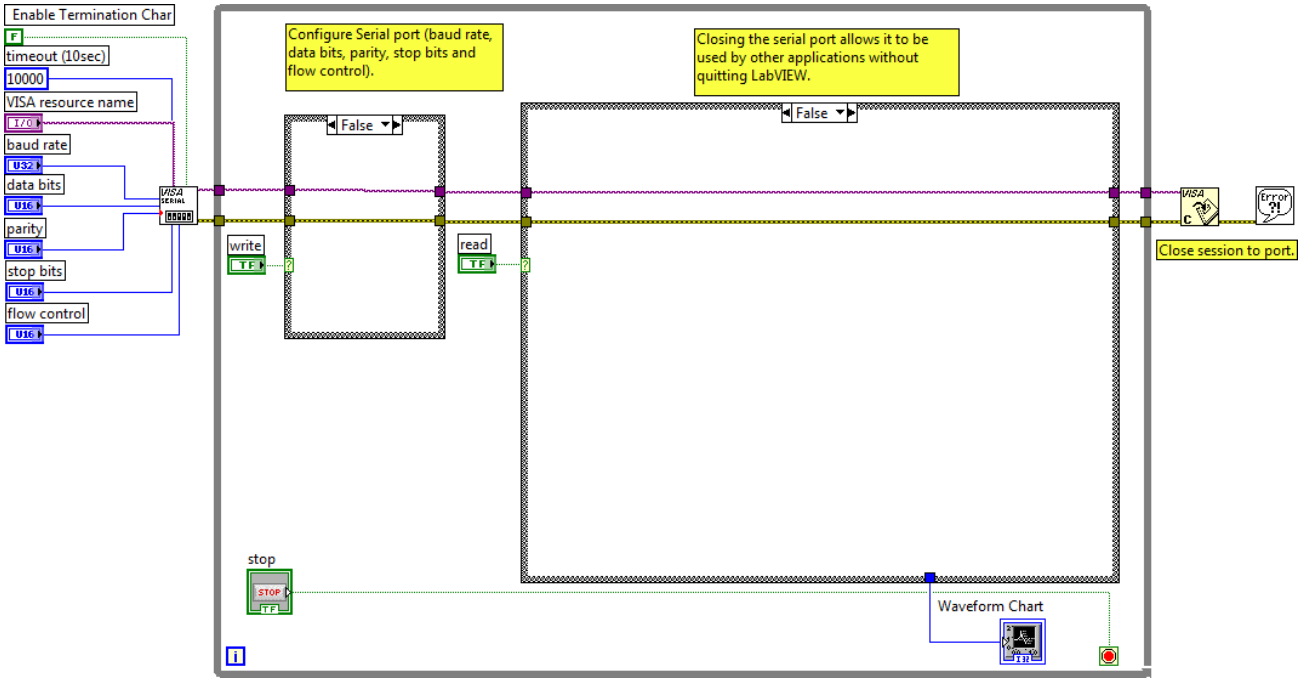

Fig. 2.2.4. Device diagram in NI LabVIEW 2010 for False case

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.2.5.

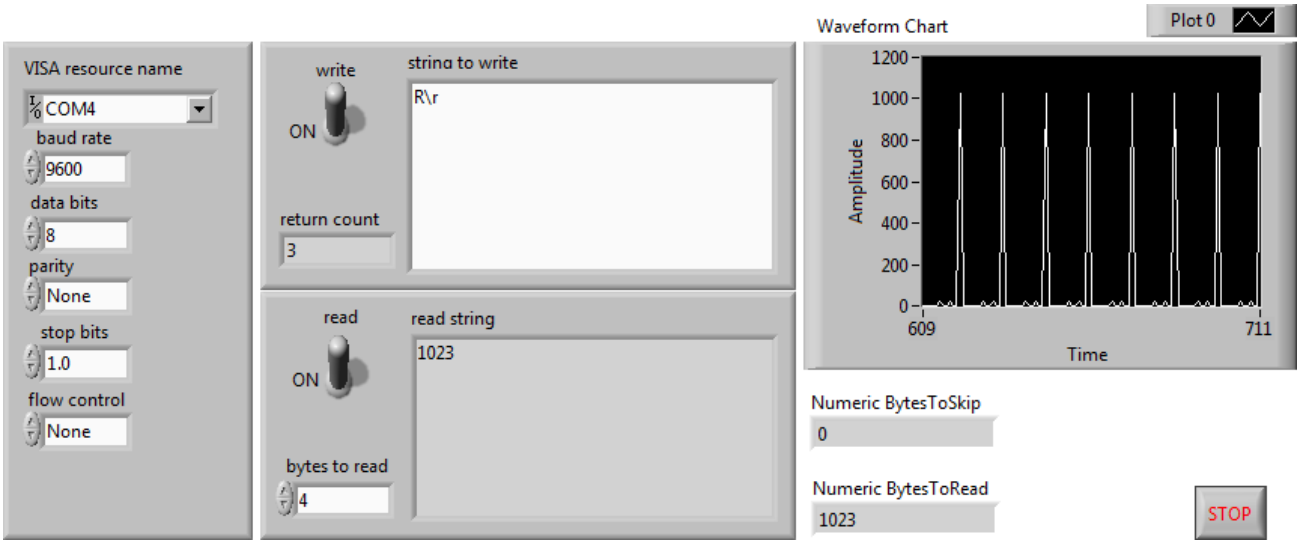

Fig. 2.2.5. The appearance of the vehicle control system

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

To formulate in the report on the completed work the conclusions of the research results and to prepare the answers to the control questions.

Methodical instructions

1. The National Instruments LabVIEW 2010 controls shown in fig. 2.2.3 and Fig. 2.2.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define ctsPin 10 // audio sensor pin
int pinRead (int levPin); // user function
```

```
int levValue; // max Value = 1024;
boolean bAlarm = false;
unsigned long lastSoundDetectTime; // Record the time that we measured a sound
int soundAlarmTime = 500; // Number of seconds to keep the sound alarm high
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (ctsPin, INPUT);
 digitalWrite (ctsPin, HIGH);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
 levValue = pinRead (ctsPin);
if (levValue == LOW) {
lastSoundDetectTime = millis (); // record the time of the sound alarm
// The following is so you don't scroll on the output screen
if (! bAlarm) {
// Sig Output Level
Serial.println ("Level =");
 Serial.println (levValue);
bAlarm = true;
}
} else {
if ((millis () - lastSoundDetectTime)> soundAlarmTime && bAlarm) {
Serial.println ("not sound");
bAlarm = false;
}
}
```

```
 delay (500);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int pinRead (int levPin) {
 return digitalRead (levPin);
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment into the lab layout of the MPS. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain how the sound sensor works with the LM393 comparator.

2.3. Stepper motor control 28BYJ-48

The purpose of the work is to design a digital device in a LabVIEW with microprocessor control for controlling the 28BYJ-48 5V stepper motor in the Arduino microprocessor programming environment.

Theoretical information

The five-volt 28BYJ-48 5V stepper motor (Fig. 2.3.1) is used in robotics, DIY devices, rotary blinds for air conditioners, small fans, etc. All engine specifications comply with National Electronic Standard (USA) SJ / T10689-95.

Fig. 2.3.1. Five-volt 28BYJ-48 5V stepper motor

Features of stepper motor:

- Rated supply voltage: 5 V (DC)
- number of phases: 4
- number of steps: 64
- number of microsteps: 4096
- Step: 5.625 Degrees
- Rated frequency: 100 Hz
- rated winding resistance (at 25□C): 50 Ohms
- idle frequency (clockwise): 600 Hz
- idle frequency (anti-clockwise): 1000 Hz
- Torque (clockwise at 120 Hz): 34.3 N / m
- Torque: 34.3 N / m
- friction torque (resistance to rotation): 600-1200 g / cm
- rated draft: 3500 g / cm
- Electrical safety class: A
- noise level <40dB

For the 28BYJ-48 stepper motor hardware control, the ULN2003 stepper motor driver is used, which allows the external power supply of 5 - 12 V to be connected to the stepper motor (Fig. 2.3.2).

Fig. 2.3.2. ULN2003 Stepper Motor Driver

Stepper motor driver (suitable for connector to 28BYJ-48 5V stepper motor), allows to work with both 5V and 12V motors. Compatible with the standard Arduino Stepper library, it has 4 LEDs to indicate operation (LED per channel).

An example of connecting a five-volt 28BYJ-48 5V stepper motor and a ULN2003 stepper motor driver to an Arduino UNO board is shown in fig. 2.3.3.

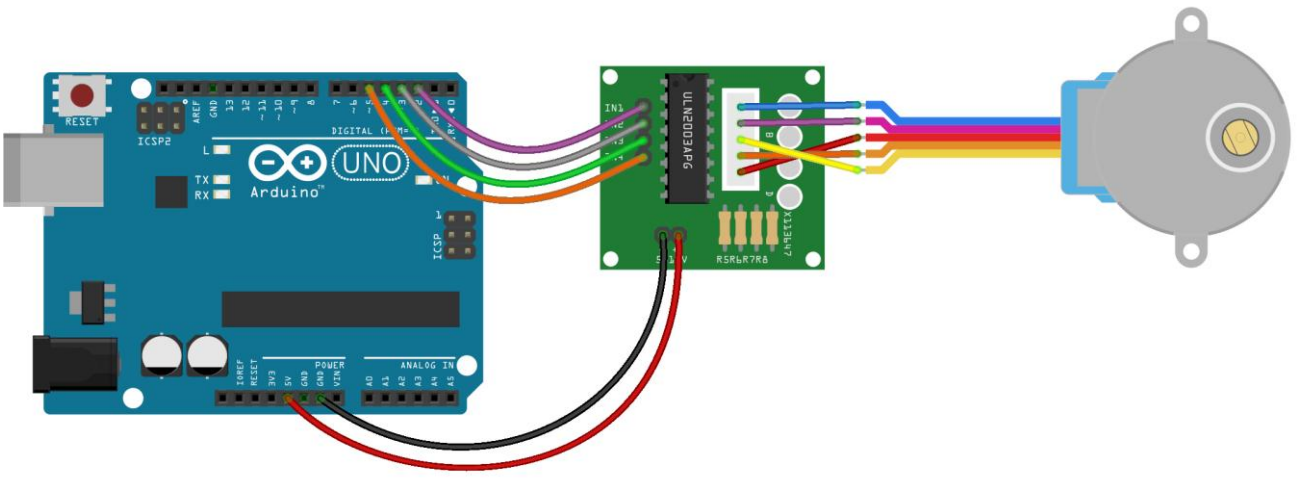

Fig. 2.3.3. Connecting devices to the Arduino UNO board

To make the engine move faster or slower, it is necessary to change the duration of the pause between the switches of the motor terminals. If you pause - the engine will spin slower, decrease the pause - the engine will spin faster.

Work task

1. To compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.3.4, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Figure 2.3.5, which corresponds to the stop of the vehicle when pressing the button "Stop".

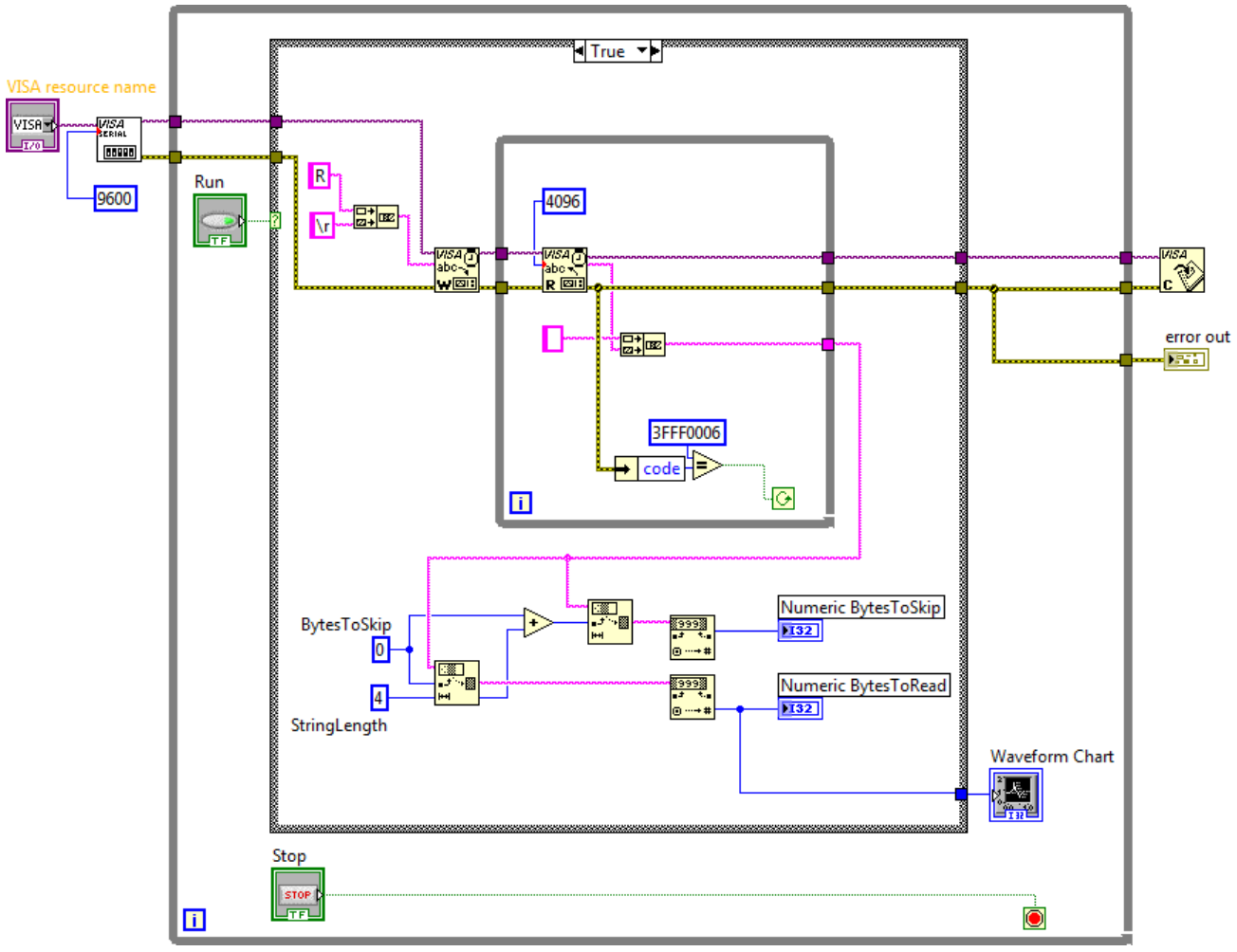

Fig. 2.3.4 Device diagram in NI LabVIEW 2010 for True

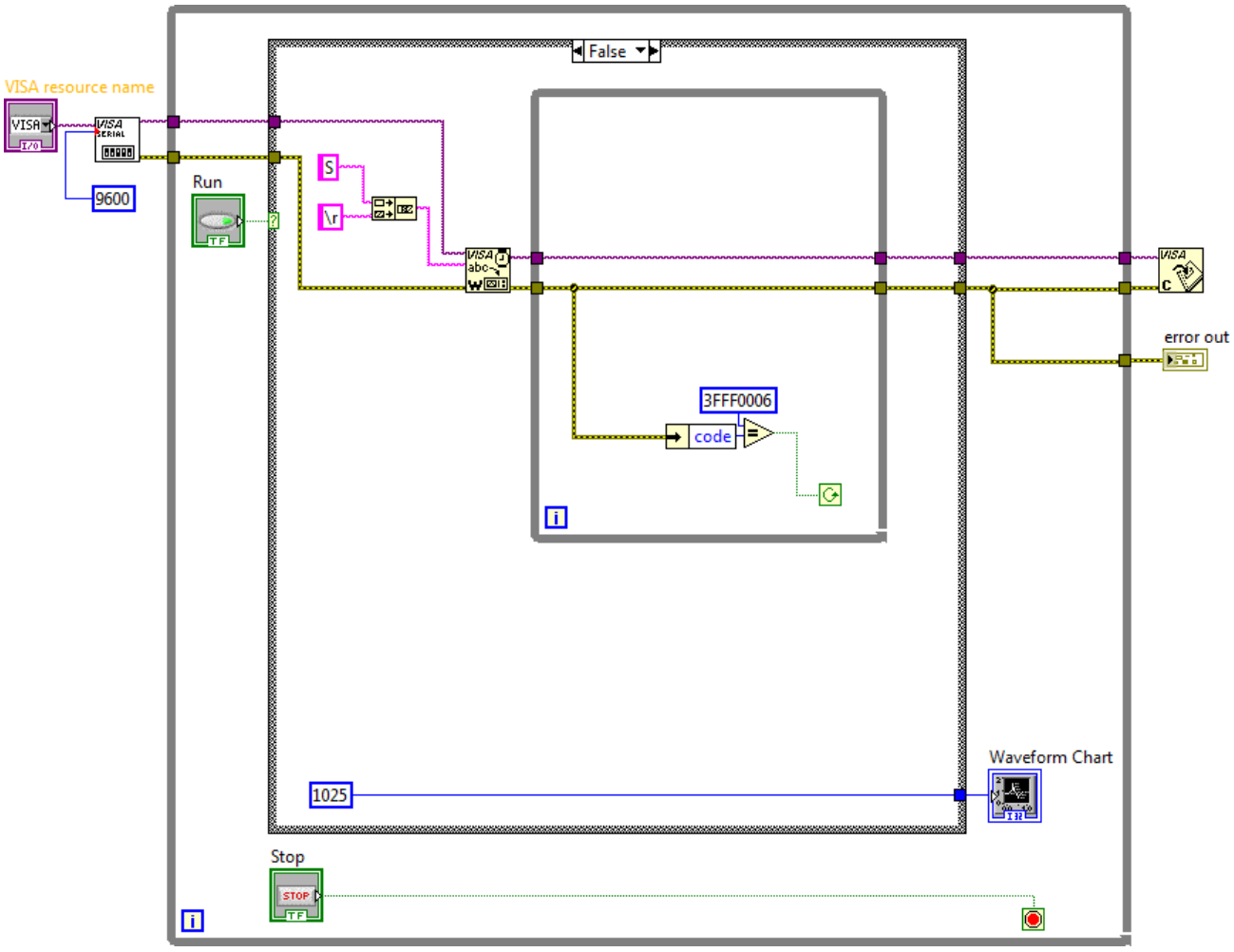

Fig. 2.3.5 Device diagram in NI LabVIEW 2010 for False case

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.3.6.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board.

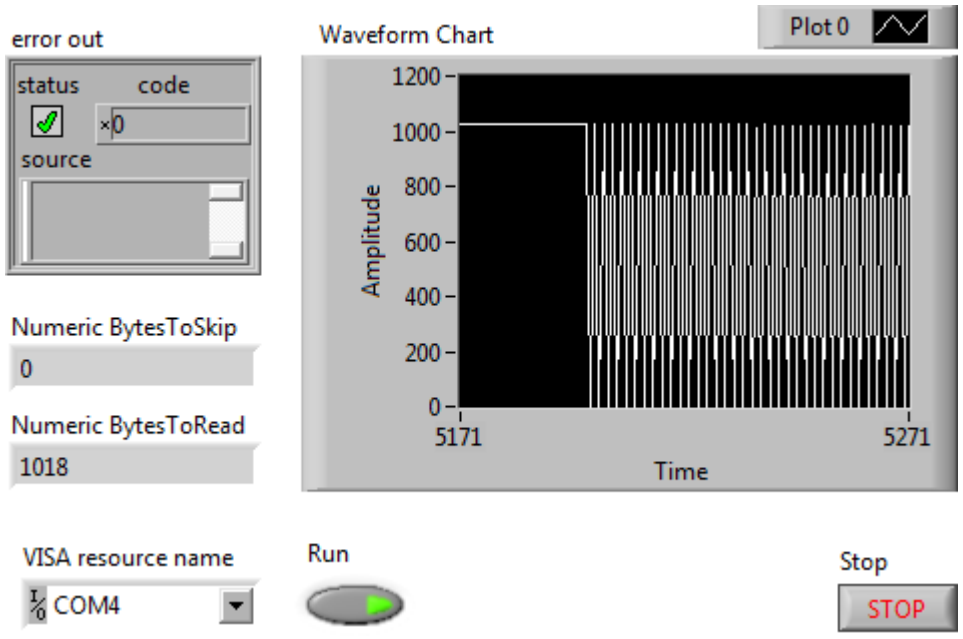

Fig. 2.3.6 Appearance of the vehicle control system

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. The National Instruments LabVIEW 2010 controls shown in fig. 2.3.4 and Fig. 2.3.5. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11
#define HOLES_DISC 2
unsigned long timeOld;
int Steps = 0;
boolean Direction = true;
unsigned long last_time;
unsigned long currentMillis;
int steps_left = 24095;
long time;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
Serial.begin (9600);
pinMode (IN1, OUTPUT);
pinMode (IN2, OUTPUT);
pinMode (IN3, OUTPUT);
pinMode (IN4, OUTPUT);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
  while (steps_left> 0) {
  currentMillis = micros ();
  if (currentMillis-last_time> = 1000) {
   stepper (1);
   time = time + micros () - last_time;
   last_time = micros ();
  steps_left--;
  }
  if (millis () - timeOld> = 1000) {
   timeOld = millis ();
  }
 }
 Serial.println (time);
 Serial.println ("Wait ...!");
 delay (2000);
  Direction = Direction;
          steps_left = 24095;
}
```

1. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
// User functions
void stepper (int xw) {
 for (int x = 0; x <xw; x ++) {
```

```
 switch (Steps) {
 case 0:
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, LOW);
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 break;
case 1:
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, LOW);
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, HIGH);
 break;
 case 2:
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, LOW);
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 break;
 case 3:
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 digitalWrite (IN3, HIGH);
digitalWrite (IN4, LOW);
 break;
 case 4:
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, LOW);
break;
 case 5:
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, HIGH);
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, LOW);
```

```
 break;
 case 6:
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, LOW);
 break;
 case 7:
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
digitalWrite (IN3, LOW);
digitalWrite (IN4, HIGH);
break;
 default:
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, LOW);
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, LOW);
 break;
 }
 SetDirection ();
}
}
void SetDirection () {
if (Direction == 1) {Steps ++;}
if (Direction == 0) {Steps--; }
if (Steps> 7) {Steps = 0;}
if (Steps <0) {Steps = 7; }
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment into the lab layout of the MPS. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications**.**

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. 2. What is the function of NI VISA serial I / O port?

3. Explain the methods of working with Serial Ports.
4. Explain the principle of switching the leads of the 28BYJ-48 stepper motor.

2.4. Speed sensor with digital and analog outputs

The purpose of the work is to design a digital device in a LabVIEW with microprocessor control to control the speed sensor on the LM393 chip in the Arduino microprocessor programming environment.

Theoretical information

A speed sensor is designed to measure rotation speed with a disc having slots. The principle of action is to record interruptions of light radiation as it passes through the slots (shutters) in the object (Fig. 2.4.1).

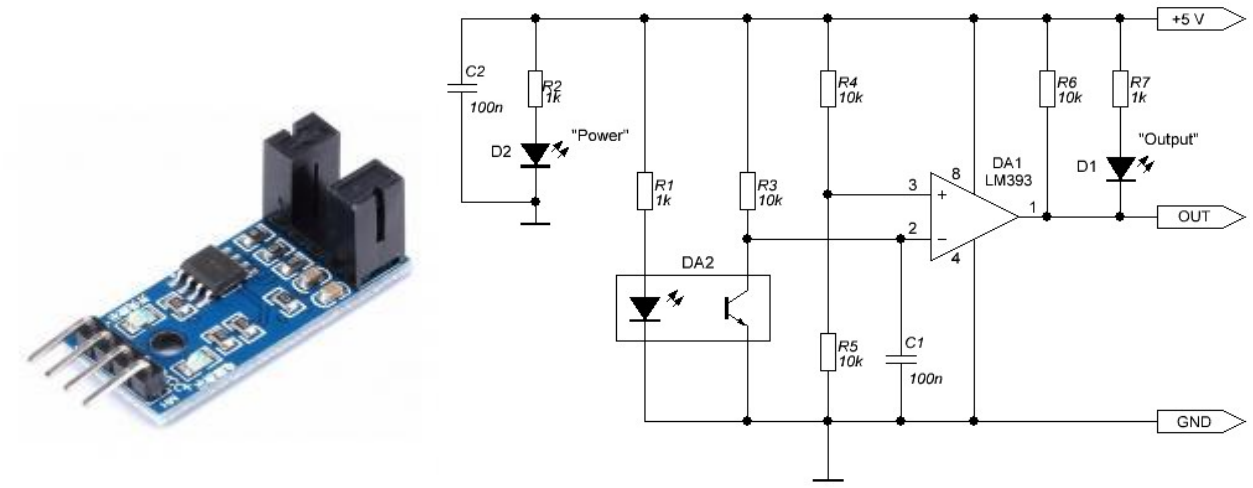

Fig. 2.4.1. Speed sensor with digital and analog outputs

This module can be used as an endpoint sensor to determine the position of an object (ENDSTOP). Due to the design of the sensor, the movement of the object, which is an obstacle to the passage of light, overlaps very precisely.

Characteristics of the speed sensor:

- supply voltage: 3.6 to 5V;
- output type: analog and digital;
- current comparator: LM393;
- indicators: power, output status;

Purpose of module outputs:

- VCC: power supply + 5V;
- GND: total land;
- DO: digital output;
- AO: analog output.

The connection of the speed sensor on the LM393 chip to the Arduino UNO-based vehicle is shown in fig. 2.4.2.

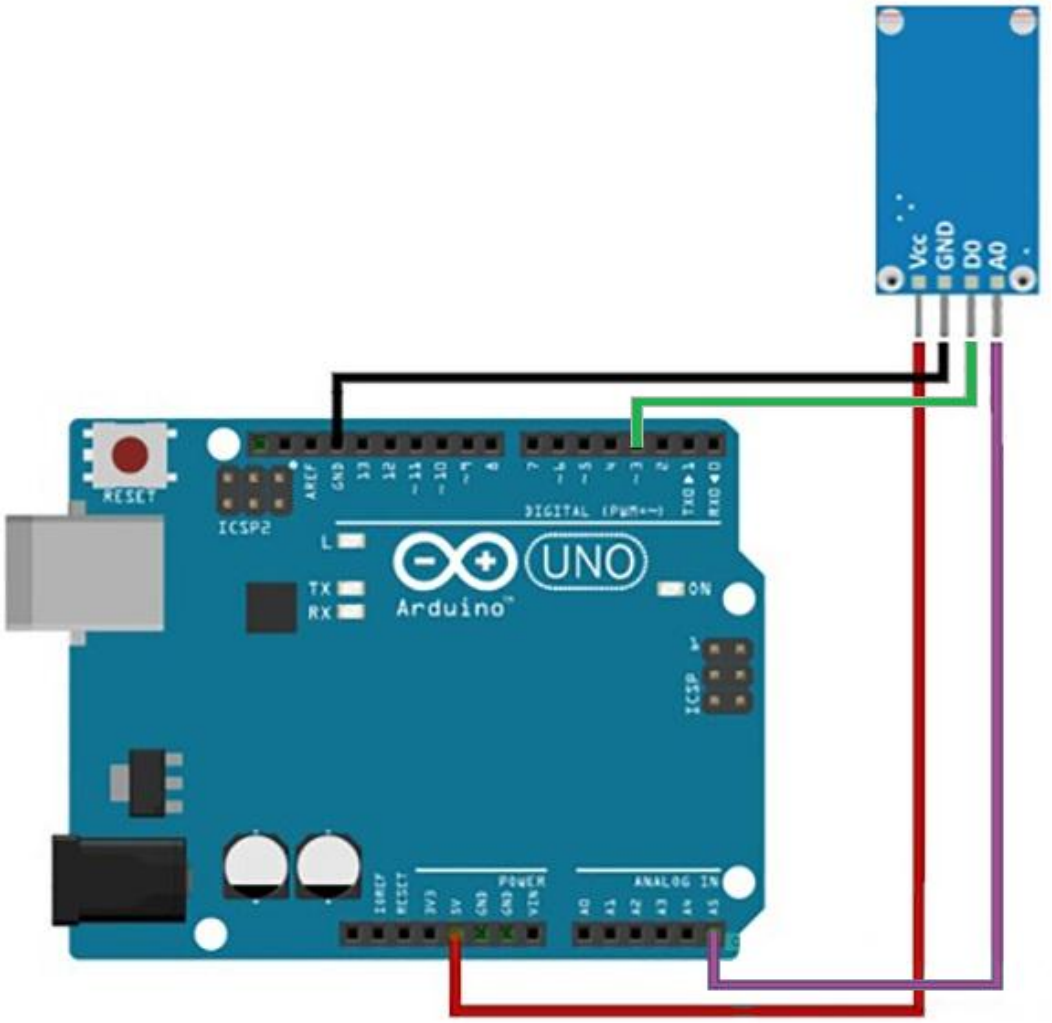

Fig. 2.4.2. Connecting the speed sensor chip to the vehicle

To connect the board uses a 4-pin connector - supply voltage + 5V, ground GND, digital output DO, and analog output. Speed sensor with special holes with holes called slot holes works.

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.3.4, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Figure 2.3.5, which corresponds to the stop of the vehicle when pressing the button "Stop".

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.3.6.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

To formulate in the report on the completed work the conclusions of the research results and to prepare the answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.3.4 and Fig. 2.3.5. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define IN1 8 // Setting up the motor contacts
#define IN2 9
#define IN3 10
#define IN4 11
#define HOLES_DISC 2
#define IN_DO 3 // Setting the speed sensor contacts
float rpm; // Variable speed
unsigned long timeOld;
volatile unsigned int pulses; // Variable number of revolutions
int Steps = 0; // Variable number of steps
```

```
boolean Direction = true;
unsigned long last_time;
unsigned long currentMillis;
int steps_left = 24095;
long time;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
Serial.begin (9600);
pinMode (IN1, OUTPUT); // Setting up the motor contacts
pinMode (IN2, OUTPUT);
pinMode (IN3, OUTPUT);
pinMode (IN4, OUTPUT);
pinMode (IN_DO, INPUT); // Setting the speed sensor contacts
pulses = 0;
timeOld = 0;
attachInterrupt (digitalPinToInterrupt (IN_DO), counter, FALLING);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
 while (steps_left> 0) {
  currentMillis = micros ();
// Timer programming
  if (currentMillis-last_time> = 1000) {
  stepper (1);
   time = time + micros () - last_time;
  last_time = micros ();
   steps_left--;
  }
```

```
// Determination of rotation speed
if (millis () - timeOld> = 1000) {
  detachInterrupt (digitalPinToInterrupt (IN_DO));
  rpm = (pulses * 60) / (HOLES_DISC);
  Serial.println (rpm); // Output rotation speed
 pulses = 0; // The number of revolutions
  timeOld = millis ();
  attachInterrupt (digitalPinToInterrupt (IN_DO), counter, FALLING);
 }
}
Serial.println (time);
Serial.println ("Wait ...!");
delay (2000);
// Change the direction of rotation
Direction = Direction;
steps_left = 24095;
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
// Engine control functions
   void stepper (int xw) {
for (int x = 0; x <xw; x ++) {
switch (Steps) {
case 0:
digitalWrite (IN1, LOW);
digitalWrite (IN2, LOW);
digitalWrite (IN3, LOW);
digitalWrite (IN4, HIGH);
break;
case 1:
digitalWrite (IN1, LOW);
```

```
digitalWrite (IN2, LOW);
digitalWrite (IN3, HIGH);
digitalWrite (IN4, HIGH);
break;
case 2:
digitalWrite (IN1, LOW);
digitalWrite (IN2, LOW);
digitalWrite (IN3, HIGH);
digitalWrite (IN4, LOW);
break;
case 3:
digitalWrite (IN1, LOW);
digitalWrite (IN2, HIGH);
digitalWrite (IN3, HIGH);
digitalWrite (IN4, LOW);
break;
case 4:
digitalWrite (IN1, LOW);
digitalWrite (IN2, HIGH);
digitalWrite (IN3, LOW);
digitalWrite (IN4, LOW);
break;
case 5:
digitalWrite (IN1, HIGH);
digitalWrite (IN2, HIGH);
digitalWrite (IN3, LOW);
digitalWrite (IN4, LOW);
break;
case 6:
digitalWrite (IN1, HIGH);
digitalWrite (IN2, LOW);
```

```
digitalWrite (IN3, LOW);
digitalWrite (IN4, LOW);
break;
case 7:
digitalWrite (IN1, HIGH);
digitalWrite (IN2, LOW);
digitalWrite (IN3, LOW);
digitalWrite (IN4, HIGH);
break;
default:
digitalWrite (IN1, LOW);
digitalWrite (IN2, LOW);
digitalWrite (IN3, LOW);
digitalWrite (IN4, LOW);
break;
}
SetDirection (); // Change the direction of rotation
}
   }
void SetDirection () {
// Count the number of steps
if (Direction == 1) {Steps ++;}
if (Direction == 0) {Steps--; }
if (Steps> 7) {Steps = 0;}
if (Steps <0) {Steps = 7; }
}
void counter () {
pulses ++; // The number of revolutions
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010.

Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. The technical characteristics of the MPS based on the Arduino microcontroller?
2. What is the operation of functions with a serial I / O port NI VISA?
3. Explain how to work with serial I / O ports (Serial Port).

2.5. HTU21 humidity and temperature sensor

The purpose of the work is to design a digital instrument in a microprocessor-controlled LabVIEW to control the humidity and temperature sensor HTU21 in the Arduino microprocessor programming environment.

Theoretical information

The HTU21 humidity and temperature sensor is designed to accurately measure humidity and temperature. The sensor uses an HTU21 chip with an I2C interface, which provides high accuracy of temperature measurements of 0.05 C (Fig. 2.5.1).

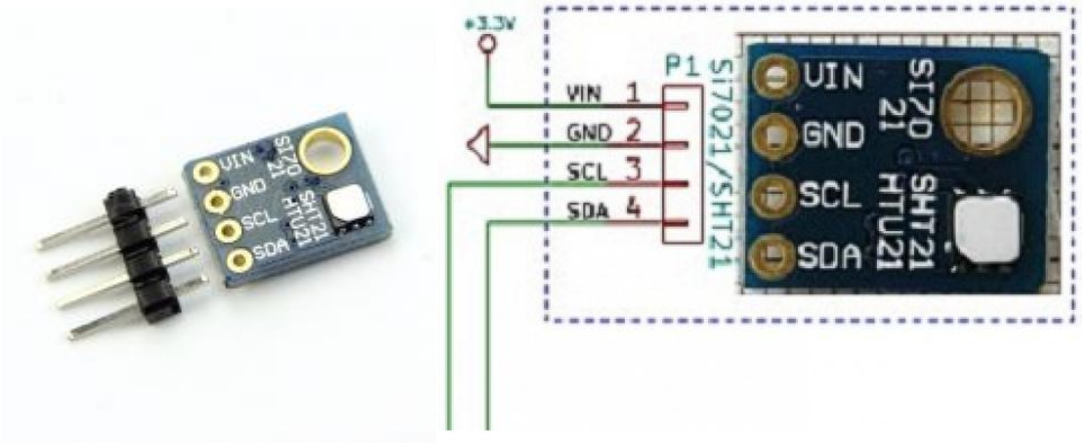

Fig. 2.5.1. HTU21 humidity and temperature sensor

The patented industry standard technology, using polymeric dielectrics for humidity sensing, allows CMOS sensors to be created with low drift, hysteresis and long-term stability of

readings. The crystal contains: analog-to-digital signal processing module, calibration data and I2C interface, a sensor element in the form of a monolithic CMOS sensor. The module has a low drift control system and low power consumption.

HTU21 Humidity and Temperature Sensor Characteristics:

- relative accuracy of humidity sensor: ± 3% RH (max.) In the range 0 - 80% RH;
- accuracy of the temperature sensor: ± 0. 4 ° C (max) in the range from -10 to 85 ° C;
- operating range: 0 to 100% RH;
- operating temperature range: -40 to + 125 ° C;
- sensor operating voltage range: 1.9 - 3.6V;
- module supply voltage: 5 - 6V;
- current in active mode: 150 mA;
- standby current: 60 pA;
- interface: I2C;
- I2C device address: 0x40;
- heater: integrated on the module board.
- dimensions: 3x3 mm DFN housing.

Connecting the HTU21 humidity and temperature sensor to the Arduino UNO-based vehicle is shown in Fig. 2.5.2.

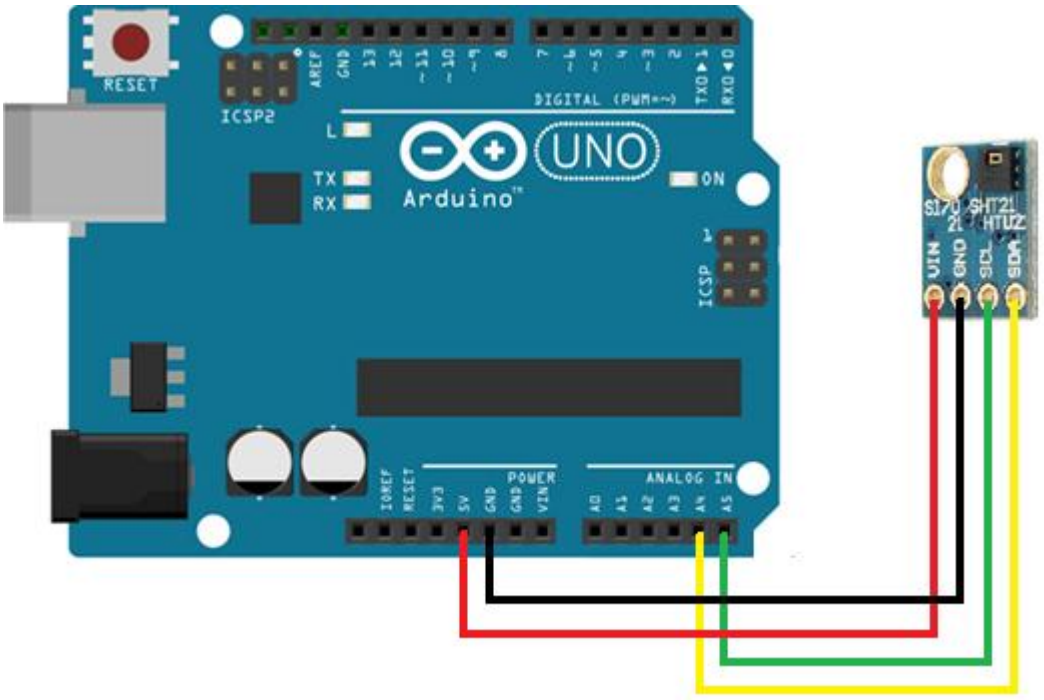

Fig. 2.5.2 Connecting the HTU21 humidity and temperature sensor to the Arduino

The calibration data is recorded at the factory and stored in independent memory. This ensures that the sensors are completely interchangeable without calibration or software change.

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case Numeric = 1 in the structure "Case Structure" is presented in Fig. 2.5.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case Numeric = 0 in the structure "Case Structure" is presented in Figure 2.5.4, which corresponds to the stop of the vehicle when pressing the button "Stop".

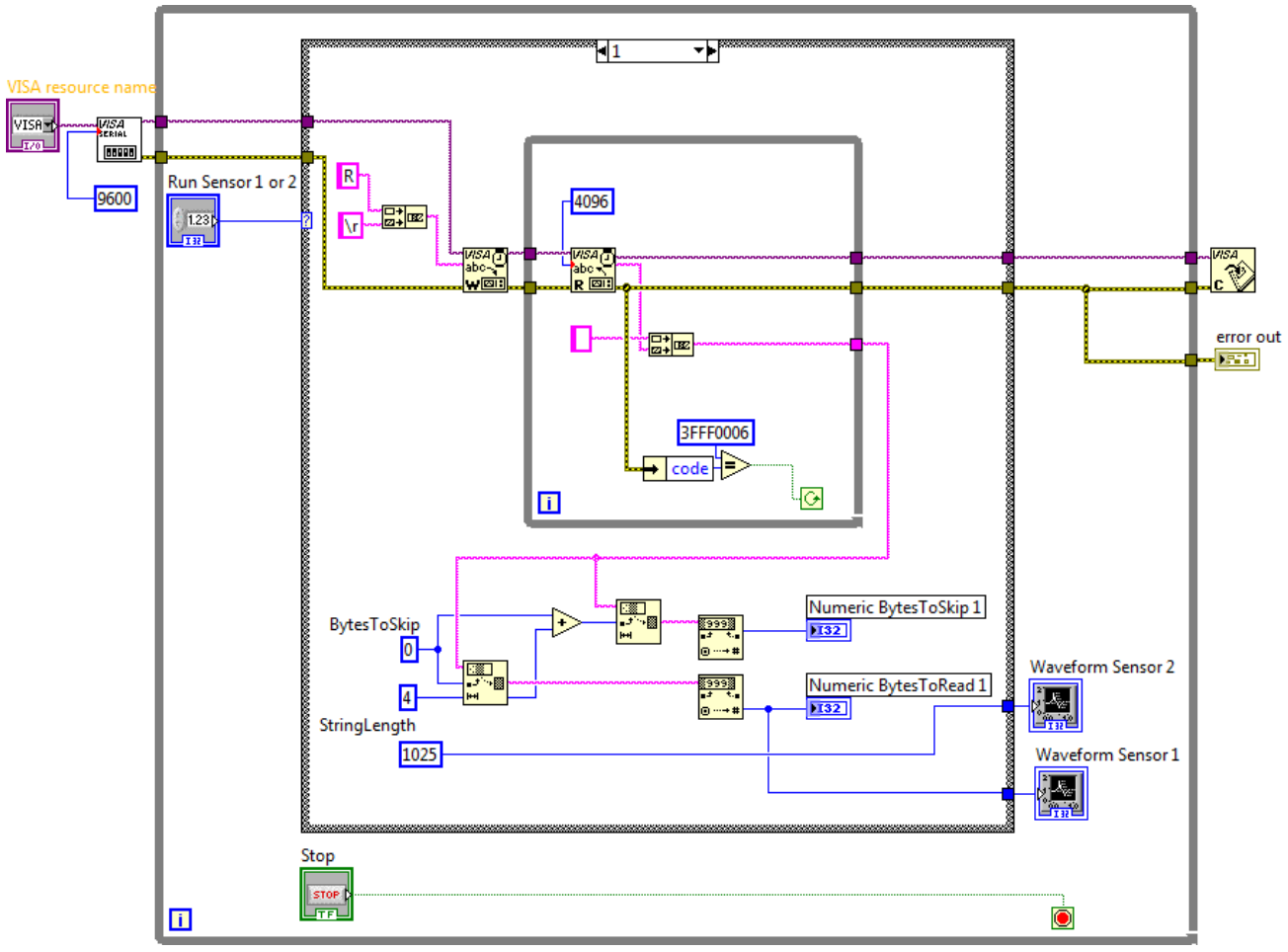

Fig. 2.5.3. Schematic of the instrument in NI LabVIEW 2010 for the case Numeric = 1

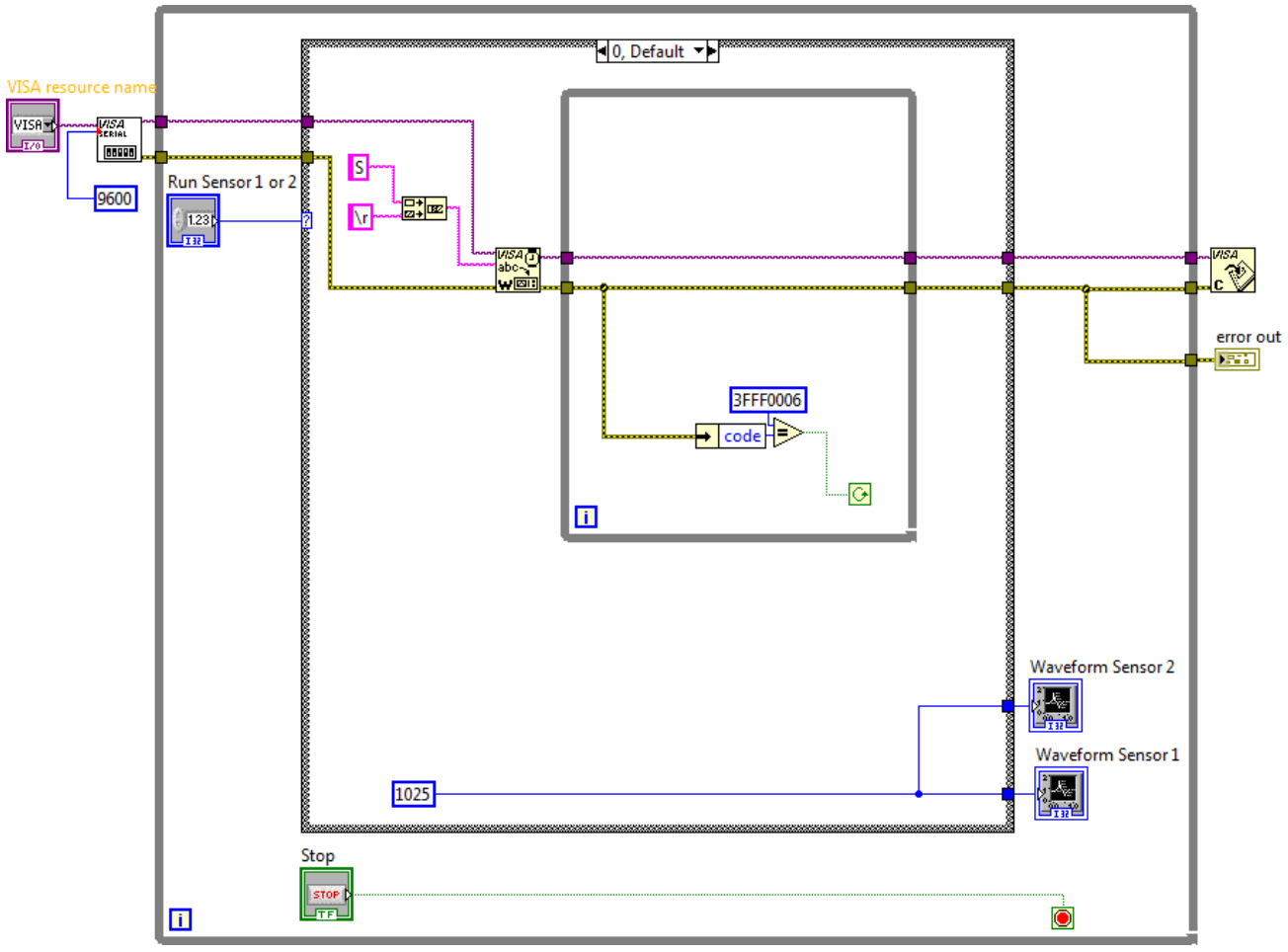

Fig. 2.5.4. Schematic of the instrument in NI LabVIEW 2010 for the case Numeric = 0

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.5.5.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.
3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board.

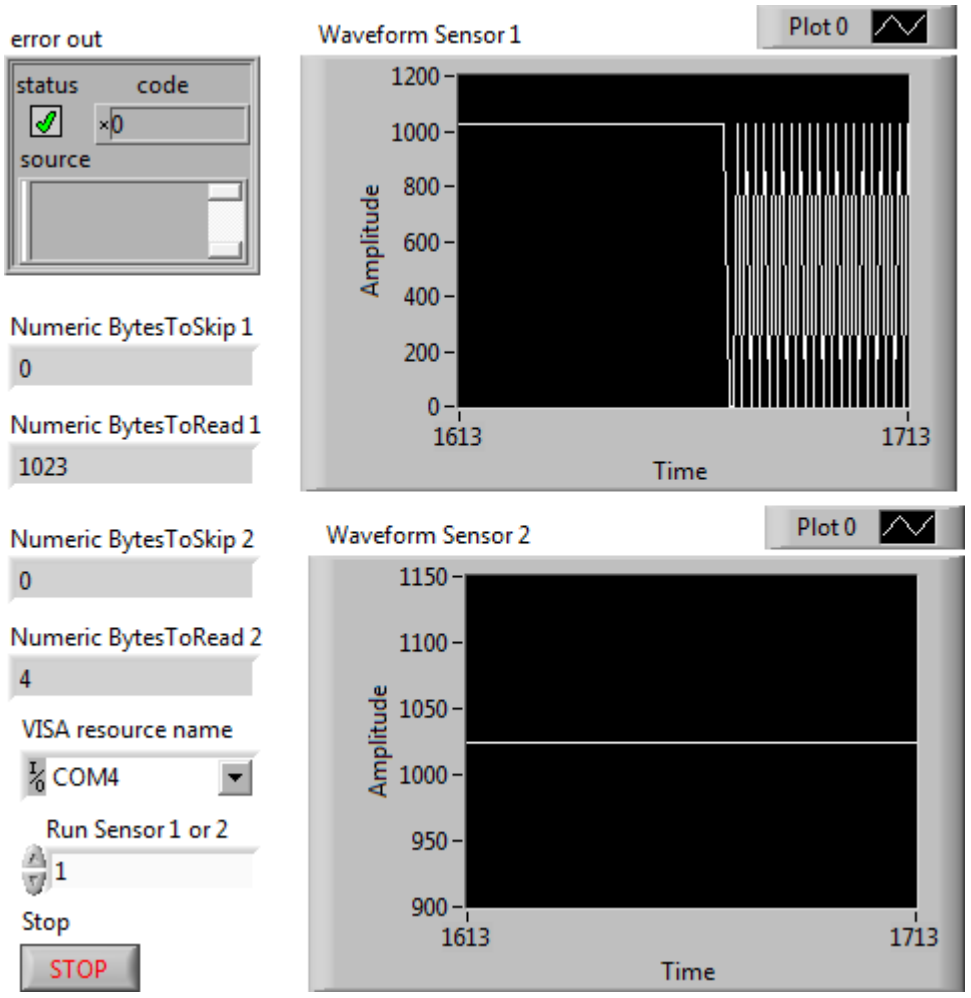

Fig. 2.5.5 Appearance of control interface

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.5.3 and Fig. 2.5.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

// Initialize data and variables

/ * Hardware Connections (Break outboard to Arduino):

VCC = 3.3V

GND = GND

```
 SDA = A4 (use inline 10k resistor if your board is 5V)
 SCL = A5 (use inline 10k resistor if your board is 5V) * /
 #include <Wire.h>
#include "HTU21D.h" // HTU21 Temperature Sensor Library
int count = 0; // Control of sensor activation
int oldStat = 0;
HTU21D myHumidity; // Create an instance of the object
float ReadTemp (); // User function for temperature
float ReadHumd (); // User function for humidity
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
  Serial.begin (9600);
  Serial.println ("HTU21D Sensor!");
  myHumidity.begin ();
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
  count = Serial.read ();
  if (count <0) {
  delay (100);
  count = oldStat;
  }
  if (count == 'R') {
  float humd, temp;
  humd = ReadHumd ();
  temp = ReadTemp ();
  Serial.print ("Time:"); // Time output
  Serial.print (millis ());
```

```
  Serial.print ("Temperature:"); // Output temperature
  Serial.print (temp, 1);
  Serial.print ("C");
  Serial.print ("Humidity:"); // Output humidity
  Serial.print (humd, 1);
  Serial.print ("%");
  Serial.println ();
  delay (1000);
  count = 'R';
  }
  if (count == 'S') {
  delay (100);
  count = 'S';
  }
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
// Temperature measurement
float ReadTemp () {
  return myHumidity.readTemperature ();
}
// Measurement of humidity
float ReadHumd () {
  return myHumidity.readHumidity ();
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?

2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the use of CMOS sensors for temperature measurement.

2.6. Digital module with thermistor

The purpose of the project is to design a digital device in a microprocessor-controlled LabVIEW to control a digital module with a thermistor in the Arduino microprocessor programming environment.

Theoretical information

The digital module with thermistor is intended for measurement of ambient temperature or closed space (Fig. 2.6.1).

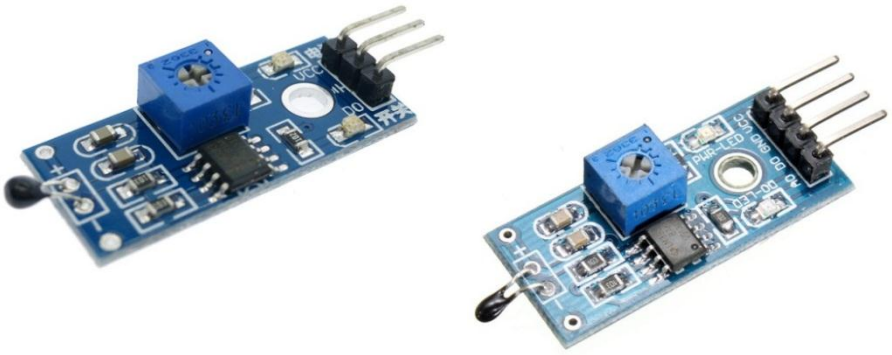

Fig. 2.6.1. Digital module with thermistor

The high precision and the ability to directly connect it to an Arduino controller or relay module allows it to be widely used in structures where you need to accurately and quickly measure temperature: thermometers, thermostats, ventilation and air conditioning systems.

Characteristics of the digital module:

- sensor type: NTC thermistor;
- measured temperature range: 20 - 80 ° C;
- max. output current: 15 mA;
- digital comparator: LM393;
- output threshold: adjustable;

- Built-in indicators: supply voltage and output status;

- operating voltage: 3 - 5V;

- size: 3.2 x 1.4 cm

The connection of the digital thermistor module to the Arduino UNO-based MFP is shown in Fig. 2.6.2.

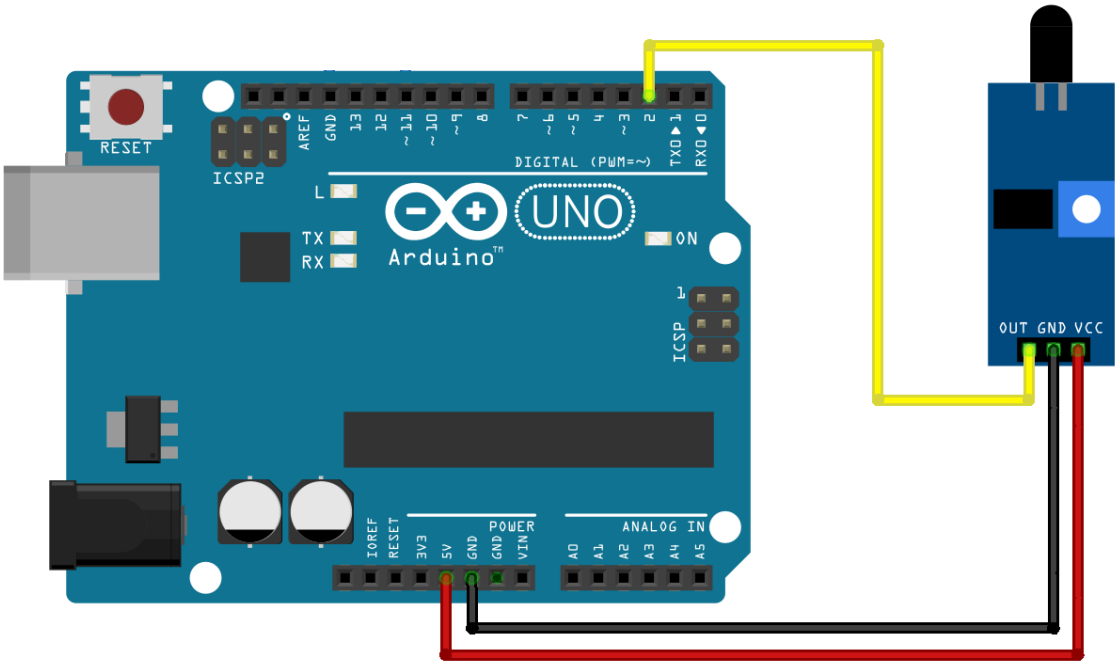

Fig. 2.6.2. Connection of digital module with thermistor to MPS

Used digital thermistor module measures ambient temperature to an accuracy of 0.5 C.

Work task

1. Compile tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case Numeric = 2 in the Case Structure structure is presented in Fig. 2.6.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case Numeric = 0 in the structure "Case Structure" is presented in Fig. 2.6.4, which corresponds to the stop of the vehicle when pressing the button "Stop".

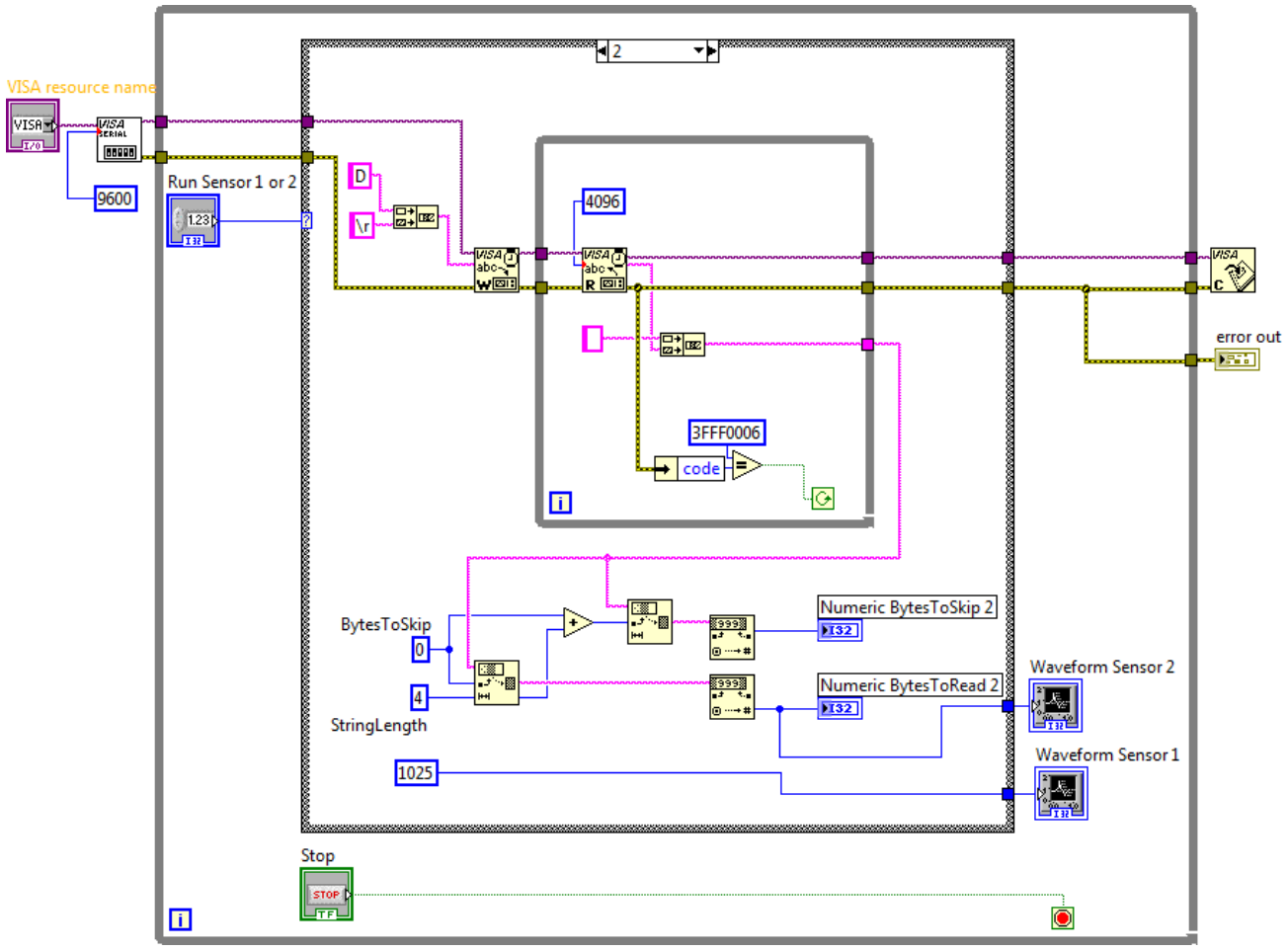

Fig. 2.6.3. Schematic of the instrument in NI LabVIEW 2010 for the case Numeric = 2

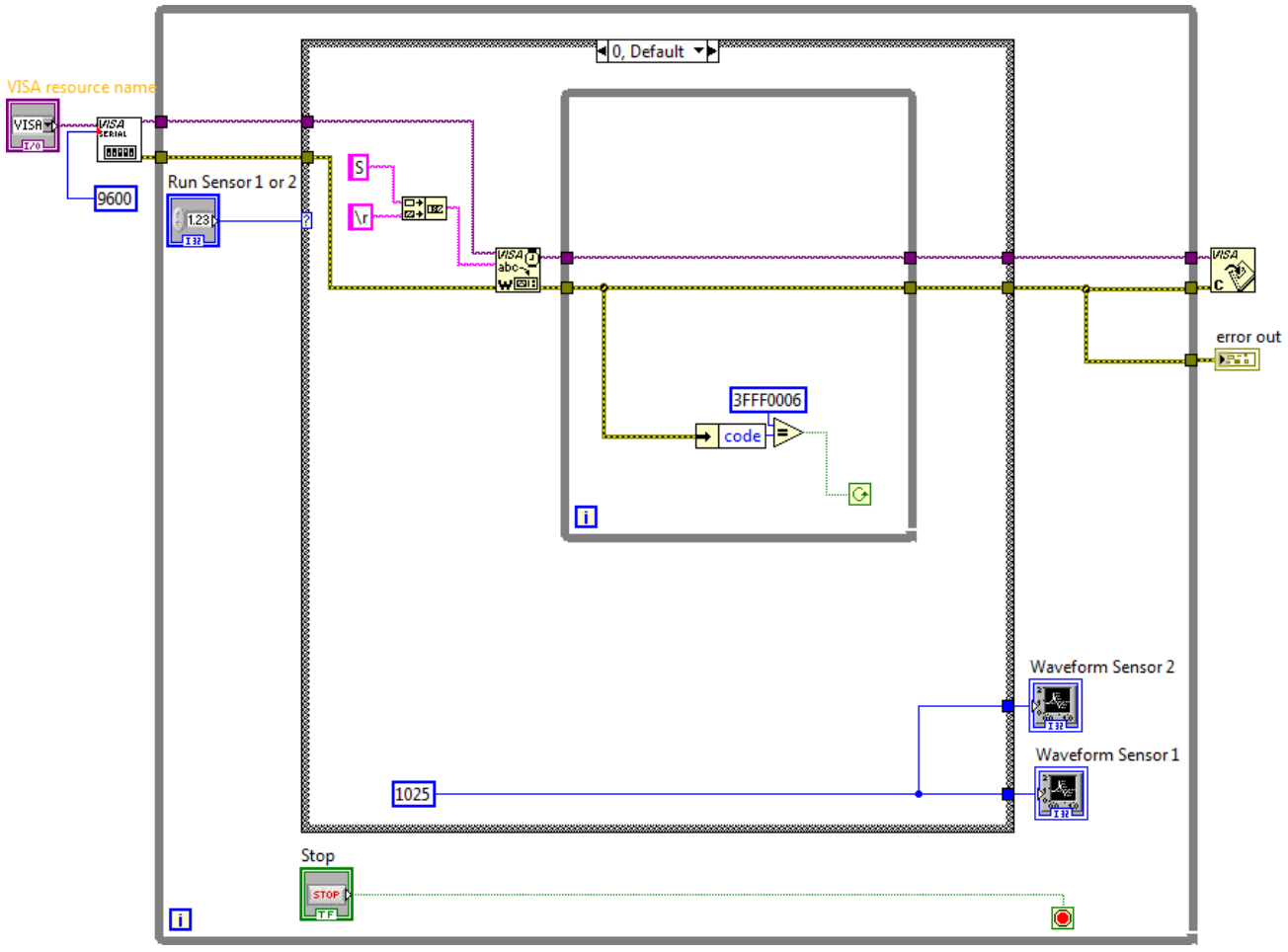

Fig. 2.6.4. Schematic of the instrument in NI LabVIEW 2010 for the case Numeric = 0

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.6.5.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.
3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board.

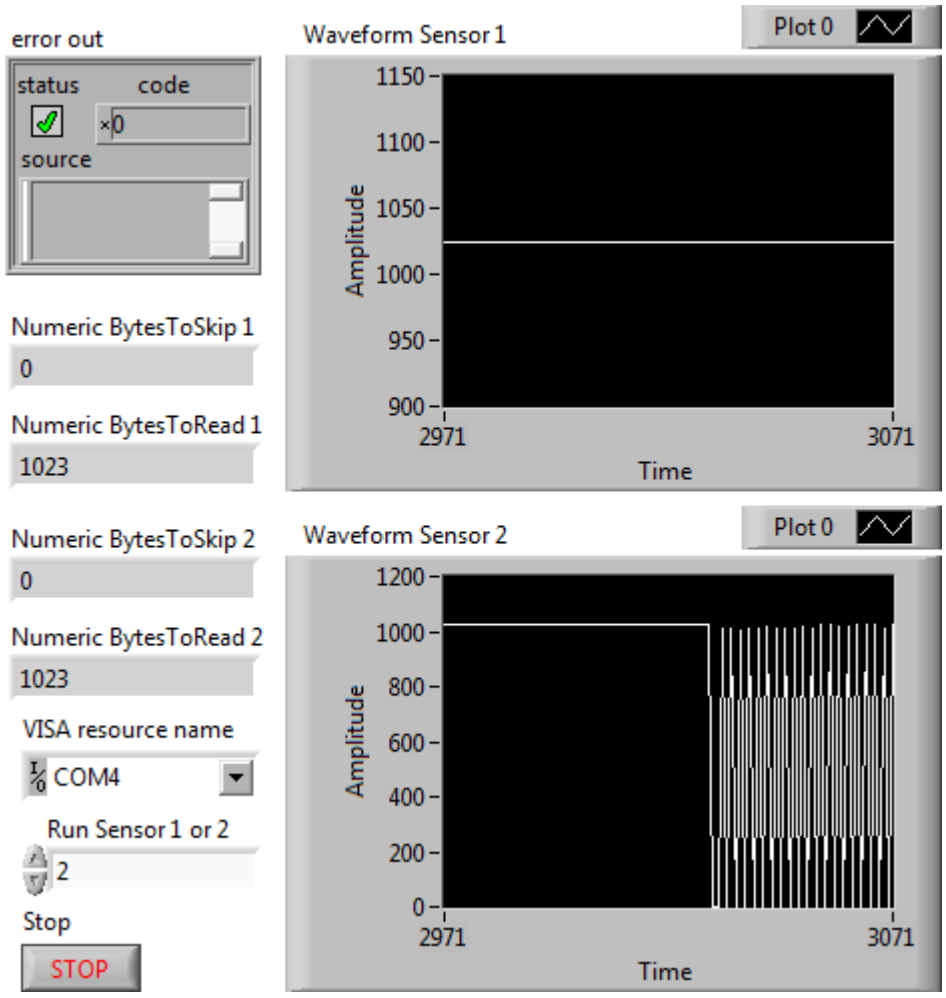

Fig. 2.6.5. The appearance of system interface

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.6.3 and Fig. 2.6.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

// Initialize data and variables

#define sensorPin A2 // Select the input pin for the potentiometer;

int count = 0; // Sensor activation control;

```
int oldStat = 0;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (sensorPin, INPUT);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
  count = Serial.read ();
  if (count <0) {
  delay (100);
  count = oldStat;
  }
  if (count == 'D') {
 int readVal = analogRead (sensorPin); // Read digital data
 double temp = Thermistor (readVal); // User function
 if (readVal! = 0) {
 Serial.println ("Convert Kelvin to Celsius:");
 Serial.println (temp); // display temperature
 Serial.println ("Temperature of Thermistor:");
 Serial.println (readVal); // display temperature
}
else {
 pinMode (sensorPin, LOW);
 Serial.println ("not measurement");
}
 delay (500);
  count = 'D';
```

```
 }
 if (count == 'S') {
 delay (100);
 count = 'S';
 }
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
double Thermistor (int RawADC) {
 double Temp;
 Temp = log (10000.0 * ((512.0 / RawADC-1)));
 Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp)) * Temp);
 Temp = 273.15 - Temp; // Convert Kelvin to Celsius
 // Temp = (Temp * 9.0) / 5.0 + 32.0; // Convert Celsius to Fahrenheit
  return Temp;
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the principle of using thermistors for temperature measurement.

2.7. HC-SR04 ultrasonic distance sensor

The purpose of the project is to design a digital device in a Microprocessor-controlled LabVIEW to control the HC-SR04 ultrasonic distance sensor in the Arduino microprocessor programming environment.

Theoretical information

The HC-SR04 ultrasonic sensor is a stable and accurate sensor (Ultrasonic Sonar) for measuring distances that do not have "blind spots". It can measure the distance to the object from 0 mm to 1500 mm with an accuracy of up to 3 mm (Fig. 2.7.1).

Fig. 2.7.1. HC-SR04 ultrasonic distance sensor

The principle of operation of the ultrasonic sensor HC-SR04 is as follows:

1. A high level is sent to the Trig output for at least 10mx;

2. The module starts to send ultrasonic pulses with a frequency of 40 kHz and receives them back, if there are any obstacles in the area of view;

3. If the signal returns, the module lowers the Echo output by 150ms. After the elapsed time of setting the high signal level Trig output to the low signal level at the Echo output, the distance to the obstacle can be calculated by the formula:

Distance = (time * sound velocity) / 2,

where time is the measured pulse time,
sound velocity - speed of sound (340 m / s).

Features of the HC-SR04 ultrasonic sensor:

- operating voltage: 3.8 - 5.5V;
- sensor type: HC-SR04;
- current: 8 mA;
- frequency: 40 kHz;
- maximum distance: 1500 mm;
- minimum distance: 0 cm;
- resolution: 3 mm;
- pulse width: 10 μs;
- angle: 15 degrees;
- external dimensions: 37x20x15 mm.

The connection of the HC-SR04 ultrasonic sensor to the Arduino UNO-based vehicle is shown in fig. 2.7.2.

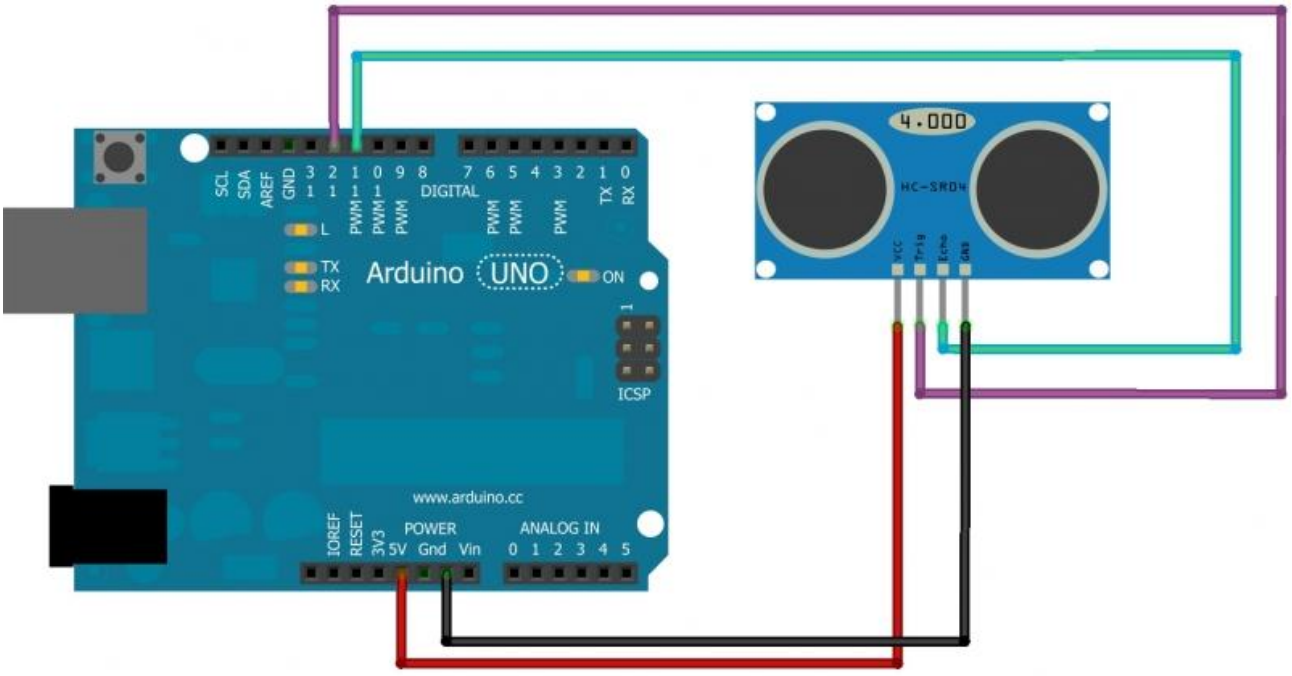

Fig. 2.7.2. Connection of the HC-SR04 ultrasonic sensor to Arduino

Most distance sensors connected to the Arduino UNO have 4 to 5 pins, which are sufficient for normal operation of the vehicle: power, ground, trigger, output.

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Figure 2.7.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Figure 2.7.4, which corresponds to the stop of the vehicle when pressing the button "Stop".

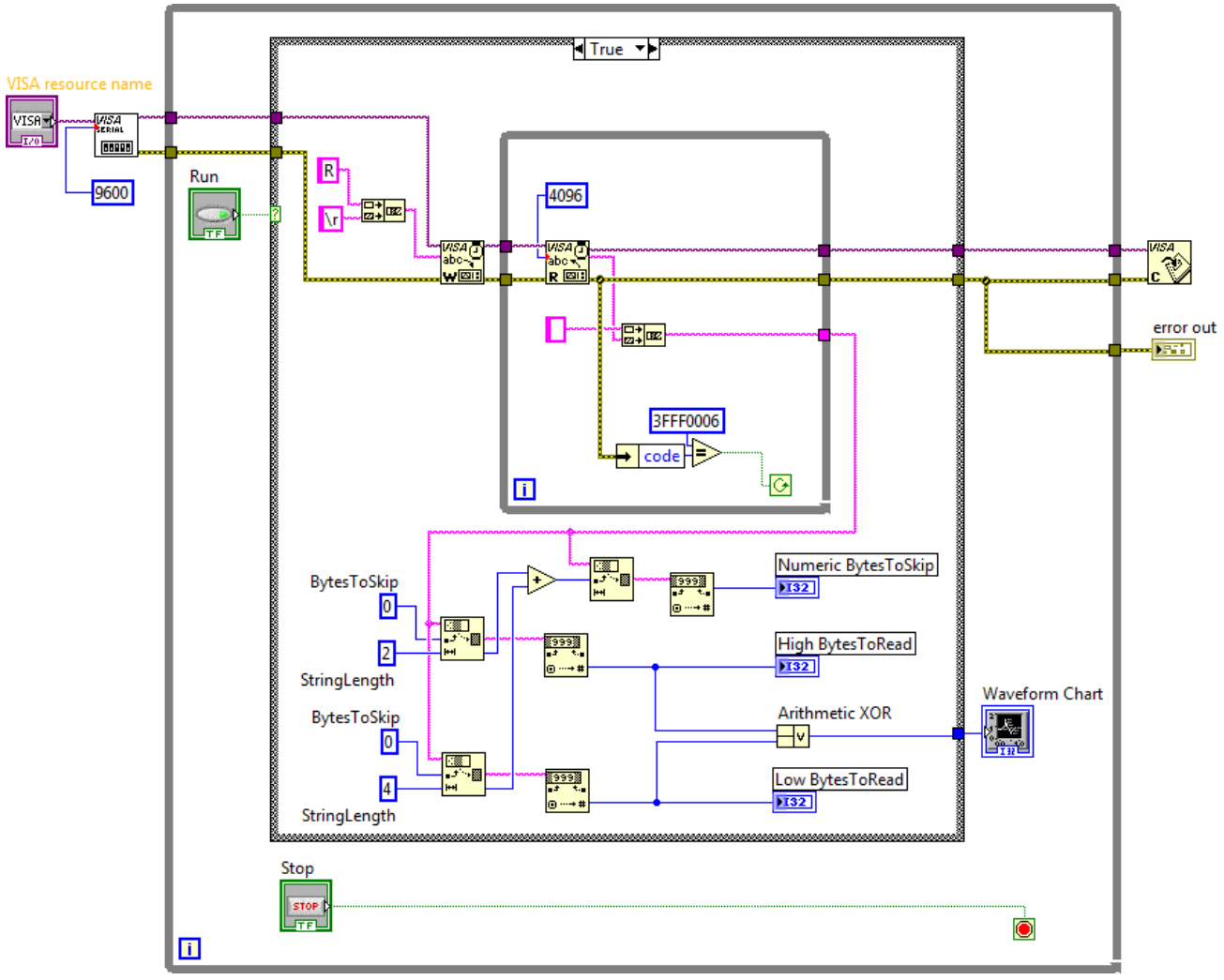

Fig. 2.7.3. Device diagram in NI LabVIEW 2010 for True case

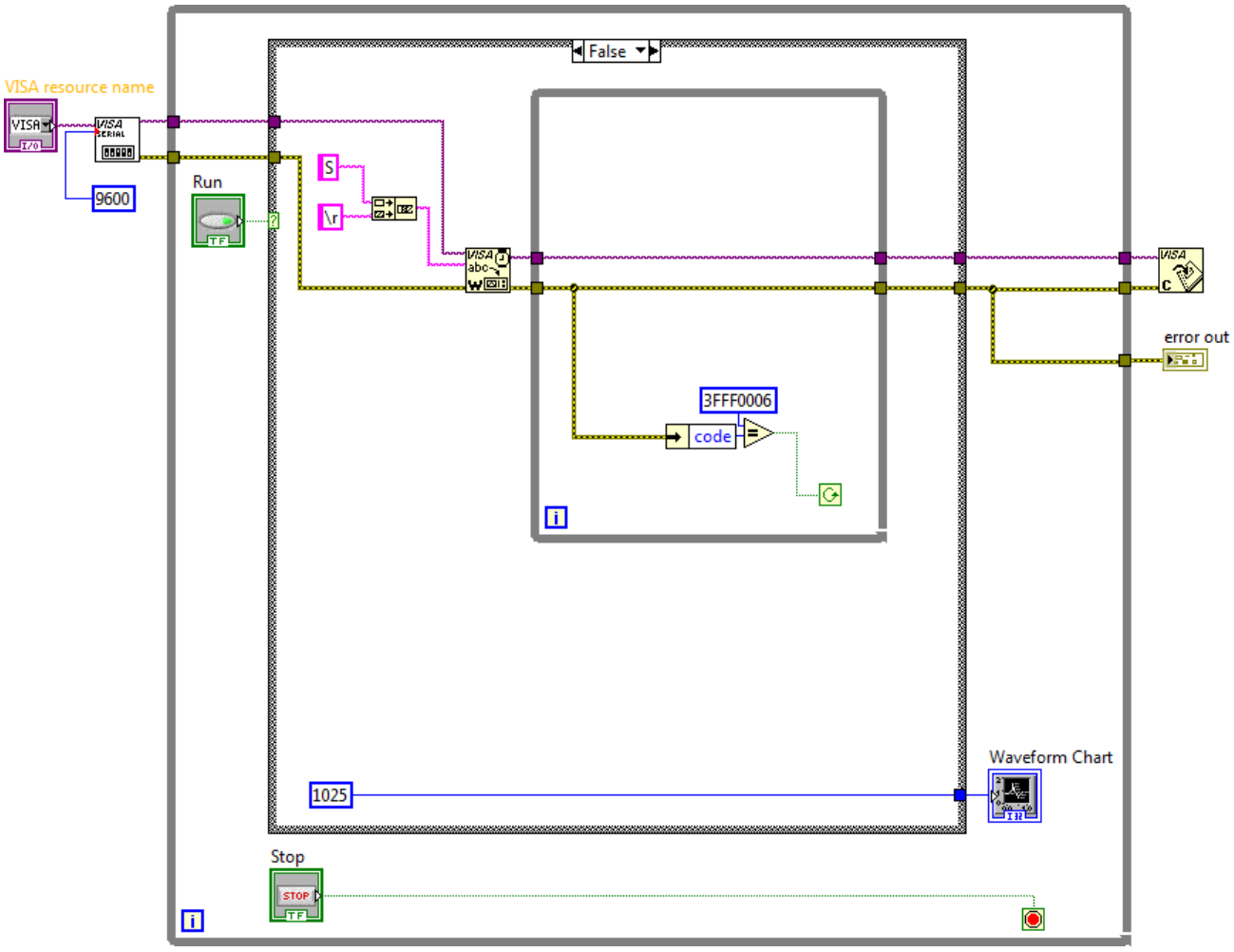

Fig. 2.7.4 Schematic of the instrument in NI LabVIEW 2010 for the False case

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.7.5.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.
3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control an Arduino UNO board.

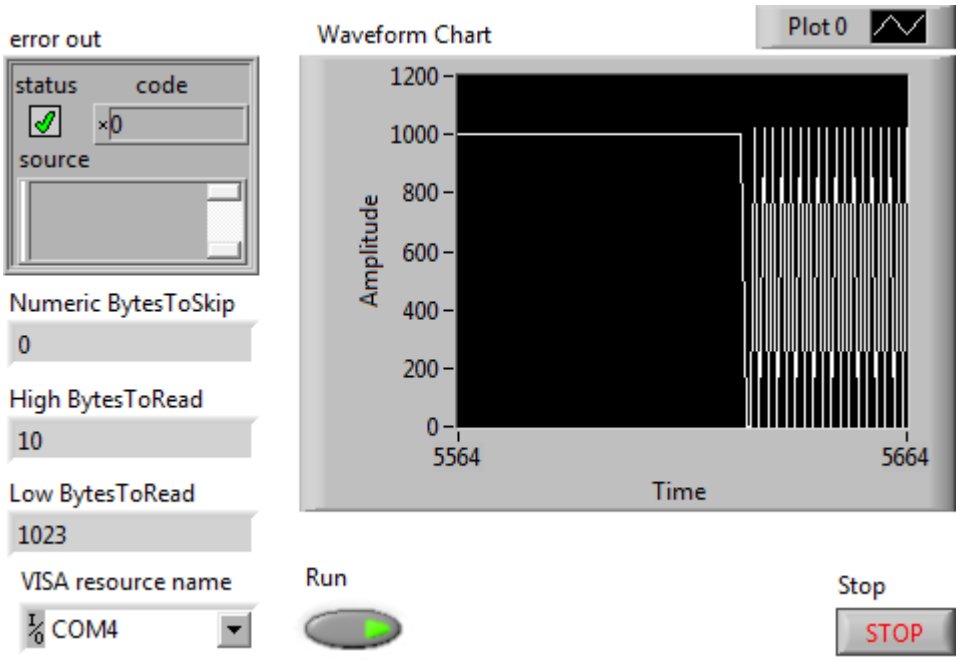

Fig. 2.7.5. The appearance of the interface control system

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.7.3 and Fig. 2.7.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

// Initialize data and variables

#define echoPin 4 // Echo Pin

#define trigPin 5 // Trigger Pin

int maximumRange = 200; // Maximum range needed

int minimumRange = 0; // Minimum range needed

long duration, distance; // Duration used to calculate distance;

long ReadSensor (int setPin); // Measurement of distance

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming

environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (trigPin, OUTPUT);
 pinMode (echoPin, INPUT);
};
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
/ * The following trigPin / echoPin cycle is used to determine the
 distance of the nearest object by bouncing sound waves off of it. * /
 // Turn Ultrasonic Sensor:
 digitalWrite (trigPin, LOW);
 delayMicroseconds (2);
 // Calculate the distance (in cm) based on the speed of sound:
 digitalWrite (trigPin, HIGH);
 delayMicroseconds (10);
 digitalWrite (trigPin, LOW);
 duration = ReadSensor (echoPin); // Measurement of distance
 distance = duration / 58.2;
 if (distance> = maximumRange || distance <= minimumRange) {
 / * Send negative number to computer and Turn LED ON
 to indicate "out of range" * /
```

```
Serial.println ("- 1");
digitalWrite (trigPin, HIGH);
}
else {
/ * Send the distance to the computer using Serial protocol, and
turn LED OFF to indicate successful reading. * /
Serial.println ("Distance =");
Serial.println (distance);
digitalWrite (trigPin, LOW);
}
// Delay 50ms before next reading.
delay (500);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
// Distance reading function
long ReadSensor (int setPin) {
  return pulseIn (setPin, HIGH);
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?

2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. What variables are included in the formula for calculating the distance to the obstacle?
5. Explain the operation of the HC-SR04 ultrasonic distance sensor.

2.8.Digital interference sensor

The purpose of the work is to design a digital device in a LabVIEW with microprocessor control to control the interference sensor in the Arduino microprocessor programming environment.

Theoretical information

The interference sensor is designed to detect the presence of surrounding objects. The digital output of the sensor can be directly connected to the Arduino or other microcontrollers (Fig. 2.8.1).

Fig. 2.8.1. Digital interference sensor

Characteristics of digital interference sensor:

- obstacle detection distance: from 2 to 30 cm;

- operating voltage from 3.3 to 5 V;

- voltage comparator: LM393;

- mounting holes with a diameter of 3 mm;

- module dimensions: 3.2 x 1.4 cm;

Description of sensor outputs:

- VCC: 3.3 - 5V connection + power supply

- GND: total land;

- OUT: digital output;

The connection of the digital interference sensor to the Arduino UNO-based vehicles is shown in fig. 2.8.2.

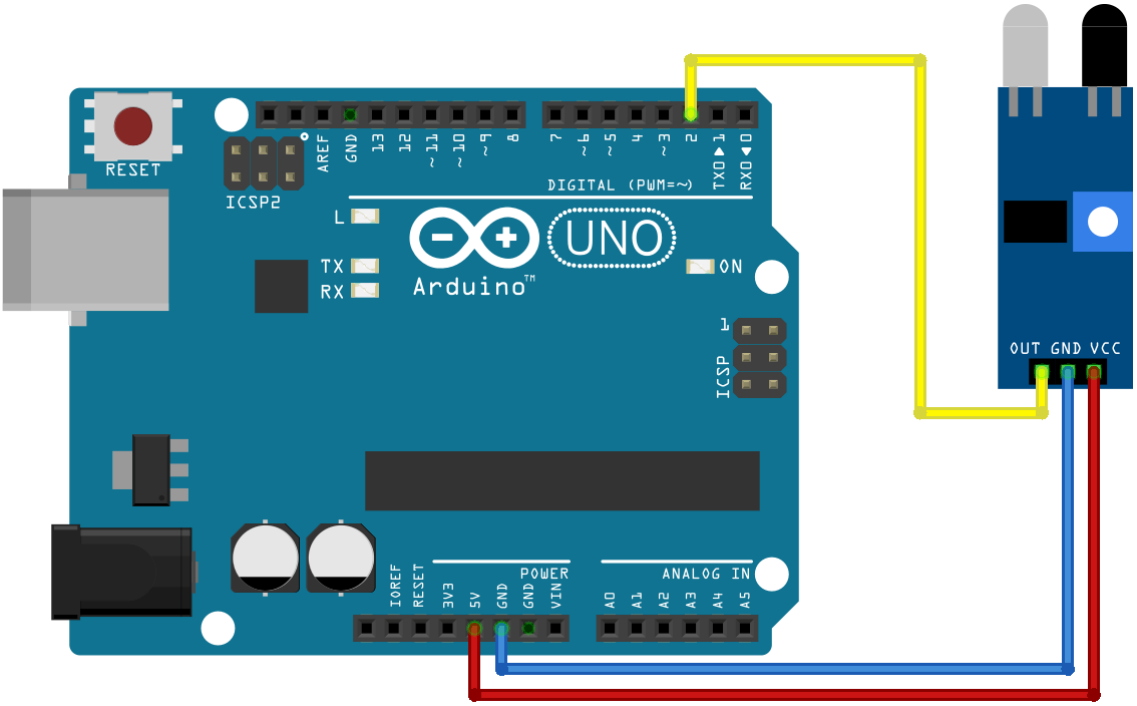

Fig. 2.8.2. Connecting a digital interference sensor to the vehicle

A digital sensor for interference detection is used in robotics due to the small size of the sensor.

Work task

1. Compile tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Figure 2.7.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure

"Case Structure" is presented in Fig. 2.7.4, which corresponds to stopping the vehicle when pressing the "Stop" button.

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.7.5.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.7.3 and Fig. 2.7.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define echoPin 4 // Echo Pin
#define trigPin 5 // Trigger Pin
#define ledPin 2 // Hindrance Pin
int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance
long duration_duo, distance_duo; // Duration used to calculate Hindrance
long ReadSensor (int setPin); // User function
long ReadHindrance (int setPin); // User function
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
```

```
 pinMode (trigPin, OUTPUT);
 pinMode (echoPin, INPUT); // Port for distance
 pinMode (ledPin, INPUT); / / Port for hindrance
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
/ * The following trigPin / echoPin cycle is used to determine the
 distance of the nearest object by bouncing sound waves off of it. * /
 // Calculate the Hindrance based on the speed of sound:
 duration_duo = ReadHindrance (ledPin); // User function for distance
 distance_duo = duration_duo;
 // Turn Ultrasonic Sensor:
 digitalWrite (trigPin, LOW);
 delayMicroseconds (2);
// Calculate the distance (in cm) based on the speed of sound:
 digitalWrite (trigPin, HIGH);
 delayMicroseconds (10);
 digitalWrite (trigPin, LOW);
 duration = ReadSensor (echoPin); // User function for hindrance
 distance = duration / 58.2;
  if (distance> = maximumRange || distance <= minimumRange) {
 / * Send negative number to computer and Turn LED ON
 to indicate "out of range" * /
 Serial.println ("- 1");
 digitalWrite (trigPin, HIGH);
 }
 else {
 / * Send the distance to the computer using Serial protocol, and
 turn LED OFF to indicate successful reading. * /
 Serial.println ("Distance =");
```

```
Serial.println (distance);
digitalWrite (trigPin, LOW);
// The Hindrance Based on the Speed of Sound:
Serial.println ("Hindrance =");
Serial.println (distance_duo);
}
 // Delay 50ms before next reading.
delay (500);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
long ReadSensor (int setPin) {
  return pulseIn (setPin, HIGH); // User function for distance
}
long ReadHindrance (int setPin) {
  return digitalRead (setPin); // User function for hindrance
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. What variables are included in the formula for calculating the distance to the obstacle?

2.9.The Hall KY-024 digital sensor

The purpose of the work is to design a digital device in a LabVIEW with microprocessor control to control the Hall KY-024 sensor in the Arduino microprocessor programming environment.

Theoretical information

The Hall KY-024 magnetic sensor is a linear sensor with a digital interface. The sensor has both digital and analog outputs. It is used to determine the magnetic field near the sensor (Fig. 2.9.1).

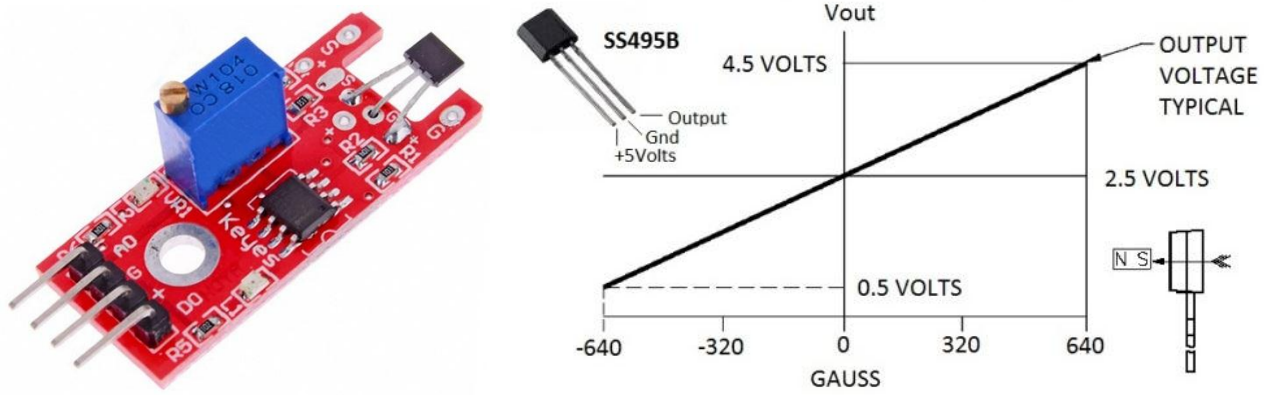

Fig. 2.9.1. Hall magnetic sensor KY-024

The Hall sensor KY-024 is used to detect the magnetic field, it can act as a limit switch, speed sensor, encoder, etc. Unlike reed switch sensors it has no moving parts, so it is much more durable.

Features of the magnetic sensor:

- type: Hall sensor (SS49E);
- double comparator: LM393;
- operating voltage: DC 5V;
- sensitivity: adjustable by a potentiometer;
- size: 32x11x20 mm.

Hall magnetic linear sensor outputs:

- A0: real-time output voltage (analog output);
- D0: digital output, threshold voltage is regulated by potentiometer;

- G: GND (general);

- power supply: "+" 3.3 - 5 V.

The connection of the Hall sensor KY-024 to the motor vehicle is carried out similarly to the connection of the speed sensor on the LM393 chip to the motor vehicle (Fig. 2.4.2).

The KY-024 board of the magnetic sensor has an LED that glows when magnetic field is detected.

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.9.2, which corresponds to the launch of the vehicle. The block diagram of the virtual device for the case of False in the structure "Case Structure" is presented in Fig. 2.9.3, which corresponds to the stop of the vehicle when pressing the button "Stop".

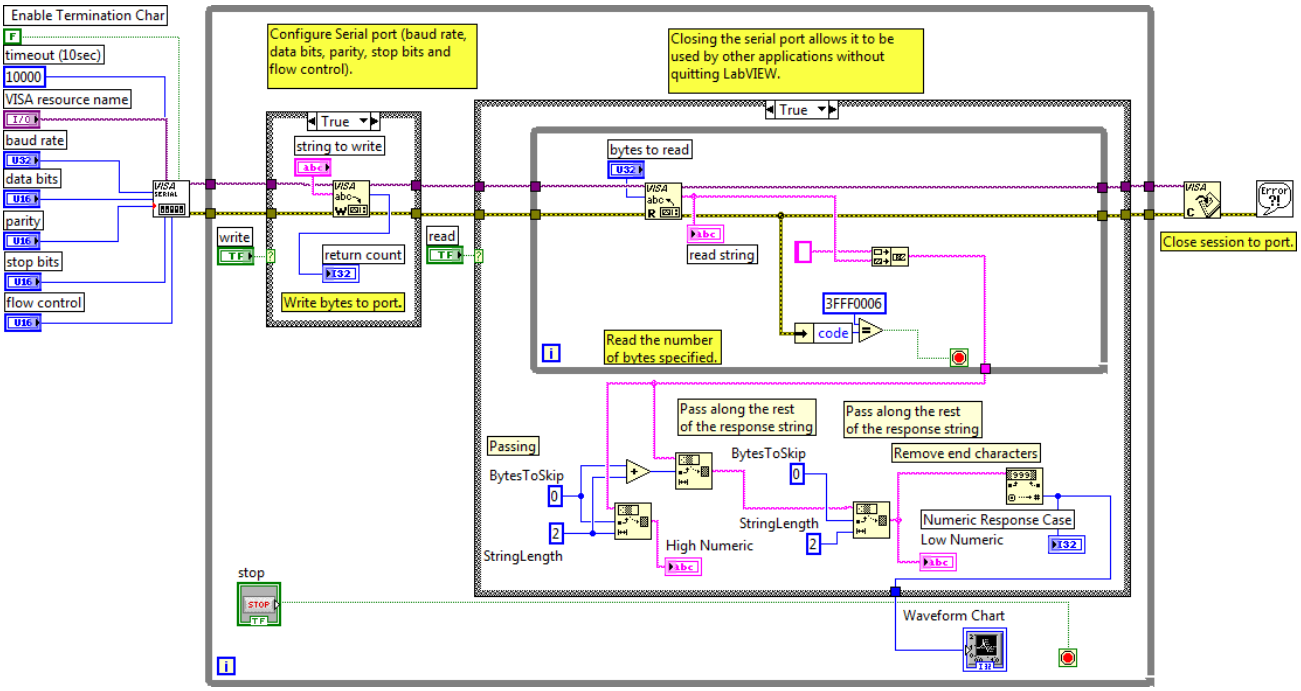

Fig. 2.9.2. Device diagram in NI LabVIEW 2010 for True case

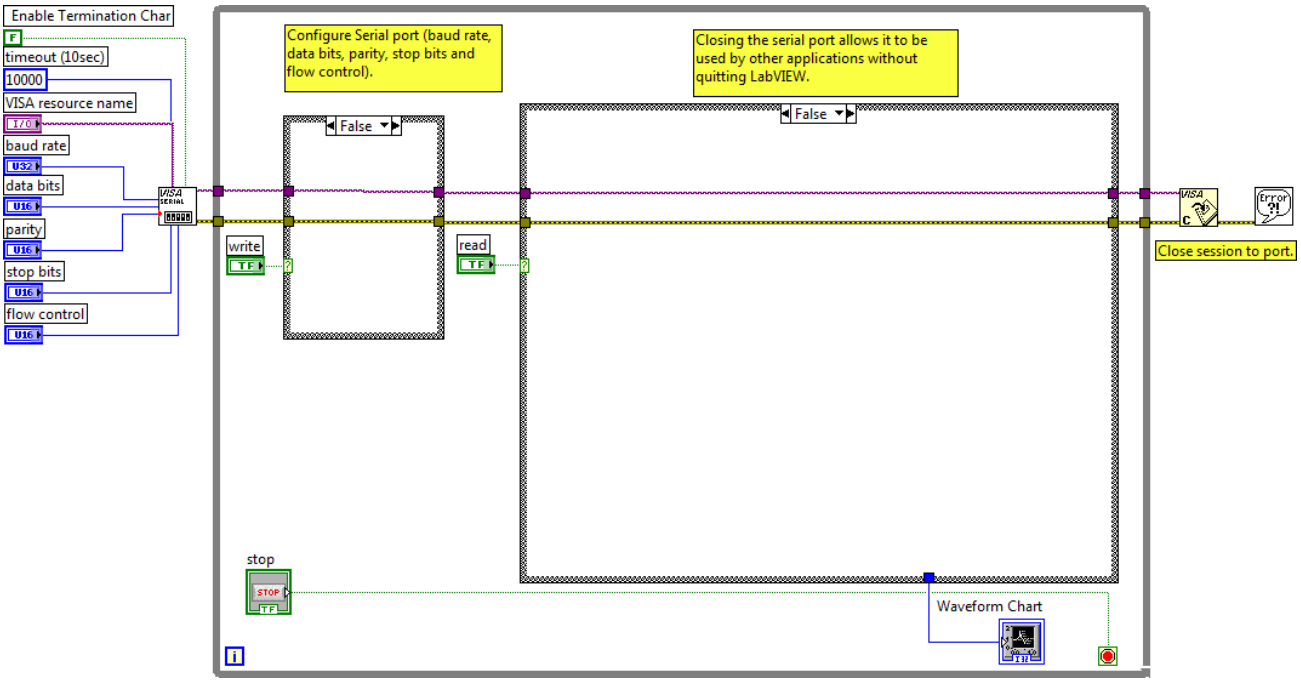

Fig. 2.9.3. Device diagram in NI LabVIEW 2010 for False case

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.9.4.

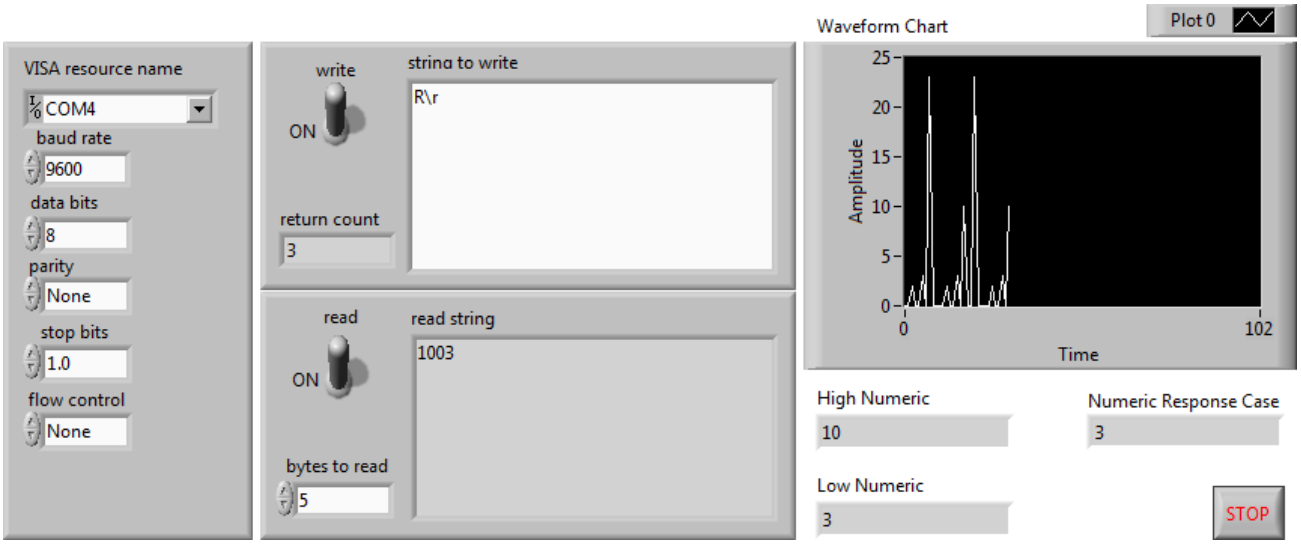

Fig. 2.9.4. The appearance of interface control system

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.9.2 and Fig. 2.9.3. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define Led 5 // define LED Interface
#define SENSOR A5 // define the Hall magnetic sensor interface;
int analog_val; // define numeric variables;
int digital_val; // define numeric variables;
int readSensor (int valPin); // user function read;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
  Serial.begin (9600);
  pinMode (Ice, INPUT); // define LED as output interface;
  pinMode (SENSOR, INPUT); // define the Hall magnetic sensor line as input;
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
analog_val = readSensor (SENSOR); // user function
digital_val = digitalRead (Led); // read sensor line
```

```
if (digital_val == HIGH) // when the Hall sensor detects a magnetic field
{
Serial.println ("Hall sensor detects:");
Serial.println (analog_val);
digitalWrite (Ice, HIGH);
}
if (analog_val <526) // level of detection of a magnetic field
{
Serial.println ("Not sensor detects");
digitalWrite (Led, LOW);
}
delay (1000);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int readSensor (int valPin) {
  return analogRead (valPin); // read sensor line
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain how the KY-024 Hall sensor works.

2.10. LM393 Tilt Sensor

The purpose of the work is to design a digital device in a LabVIEW with microprocessor control to control the tilt sensor on the LM393 chip in the Arduino microprocessor programming environment.

Theoretical information

The digital tilt sensor with the comparator LM393 is intended for use in car alarm systems and smart home systems (Fig. 2.10.1). The threshold of the sensor is regulated by the setting resistor on the board.

Features of Tilt Sensor:

- comparator: LM393;
- supply voltage: 3.3 - 5 V;
- output indicator: LED;
- output signal: digital with sensitivity threshold;
- sensitivity: adjustable (adjustable by resistor);
- board sizes: 30 x15 mm.

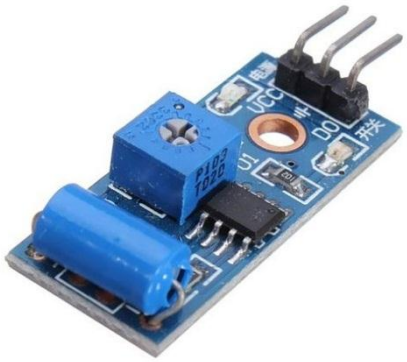

Fig. 2.10.1. Tilt sensor on the LM393 chip

The connection of the digital tilt sensor on the LM393 chip to the MPS based on the Arduino UNO controller is shown in fig. 2.10.2.

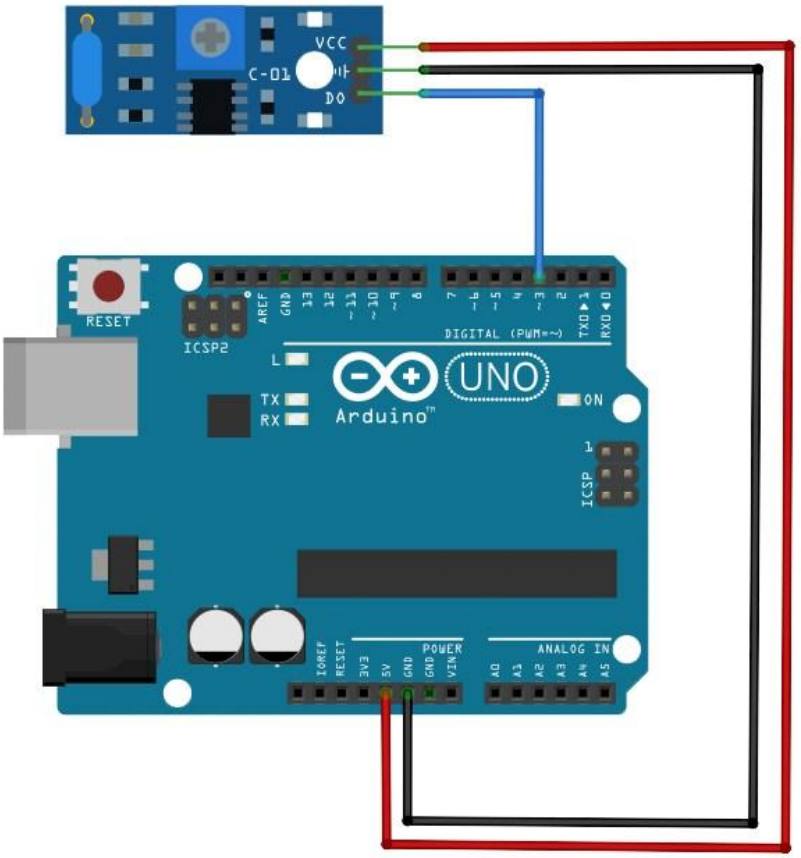

Fig. 2.10.2. Connect the digital tilt sensor on the LM393 chip to Arduino

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.9.2, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Fig. 2.9.3, which corresponds to stopping the vehicle when pressing the button "Stop".

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.9.4.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.9.1 and Fig. 2.9.2. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define Led 5 // define LED Interface
#define SENSOR 13 // define the Slope sensor interface;
int sensor_val; // define numeric variables;
int signal_val; // define numeric variables;
int readSensor (int valPin); // user function read;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (Ice, INPUT); // define LED as output interface;
 pinMode (SENSOR, INPUT); // define the Slope sensor line as input;
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
sensor_val = readSensor (SENSOR); // user function
signal_val = digitalRead (Led); // read sensor line
```

```
if (signal_val == HIGH) // when the Slope sensor detects an angle
{
Serial.println ("Slope sensor detects:");
Serial.println (sensor_val);
digitalWrite (Ice, HIGH);
}
if (signal_val <0) // level of detection an angle
{
Serial.println ("Not sensor detects");
digitalWrite (Led, LOW);
}
delay (1000);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int readSensor (int valPin) {
int data;
data = digitalRead (valPin); // read sensor line
if (data> 0) // level of detection a slope
{
digitalWrite (Ice, HIGH);
} else {
digitalWrite (Led, LOW);
}
return data;
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the MPS based on the microcontroller Arduino?
2. What is the operation of functions with a serial I / O port NI VISA?
3. Explain how to work with serial I / O ports (Serial Port).
4. Explain the principle of operation of the tilt sensor with a comparator LM393.

2.11. Alcohol sensor MQ-3

The purpose of the project is to design a digital device in a Microprocessor-controlled LabVIEW to control an MQ-3 alcohol sensor in an Arduino microprocessor programming environment.

Theoretical information

The MQ-3 alcohol sensor is designed to record and measure the concentration of alcohol vapors (2.11.1). The module is built on the basis of the gas analyzer MQ-3, which allows to detect the presence of alcohol vapors: from perfumes or spirits, in the air or breathing.

Characteristics of alcohol MQ-3:

- supply voltage: 5 V;
- current: 150 mA;
- measurement range: 0.05 mg / l - 10 mg / l;
- sensitivity: adjustable by a resistor (trimmer).

Fig. 2.11.1. Digital Alcohol Sensor MQ-3

The gas analyzer has a built-in heating element that is required for a chemical reaction. Therefore, the sensor will be hot during operation, which is normal. To get stable readings, the new sensor must be warmed up (left on) once for 24 hours. After that, the stabilization of the characteristics after switching on the MQ-3 gas analyzer will take about a minute.

The connection of an MQ-3 digital alcohol sensor to an Arduino UNO-based vehicle is shown in fig. 2.11.2

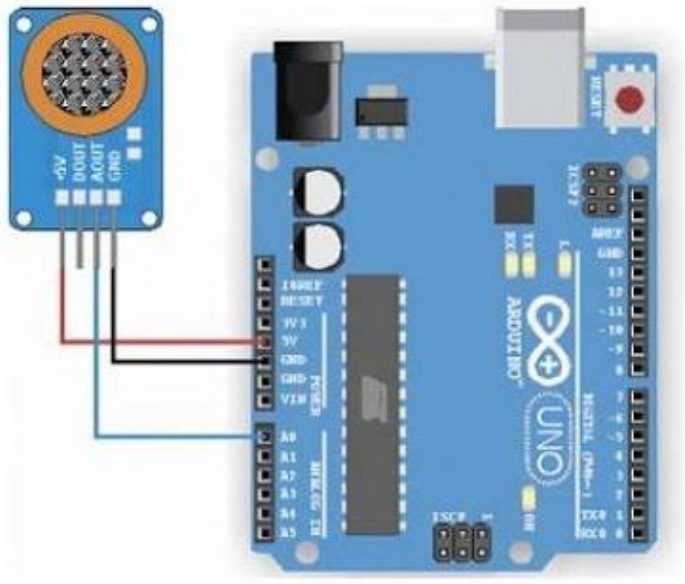

Fig. 2.11.2. Connecting the MQ-3 digital alcohol sensor to Arduino

The output is an analog signal proportional to the concentration of alcohol vapor around the gas analyzer. The sensitivity of the MQ-3 alcohol sensor can be adjusted using the trimmer on the sensor board.

Sensor readings depend on the effect of temperature and humidity of the surrounding air. Therefore, when using a gas sensor in a changing environment, to obtain accurate readings, it is necessary to compensate for these parameters.

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case

Numeric = 1 in the structure "Case Structure" is presented in Fig. 2.11.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case Numeric = 0 in the structure "Case Structure" is presented in Figure 2.11.4, which corresponds to the stop of the vehicle when pressing the button "Stop".

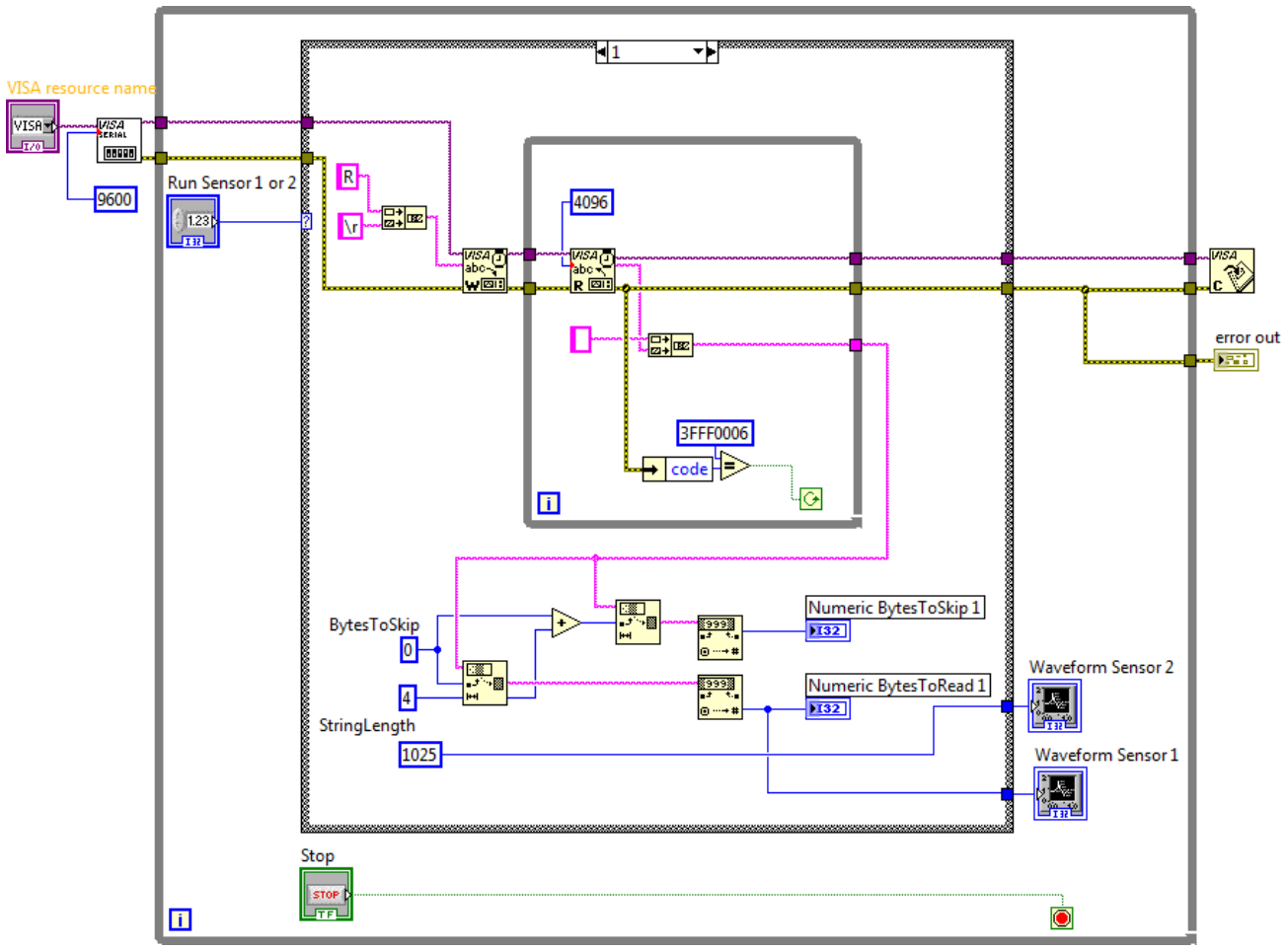

Fig. 2.11.3. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 1

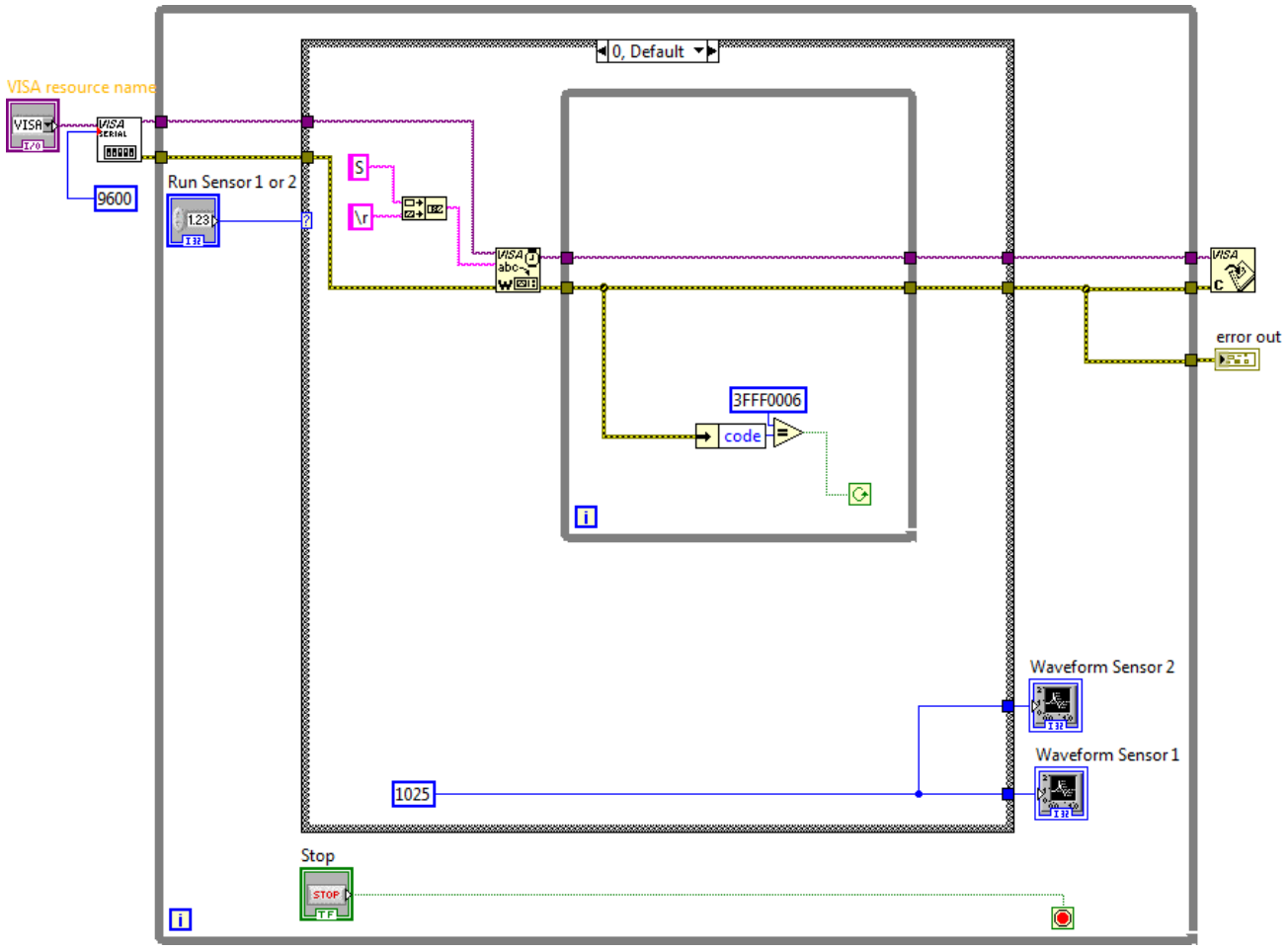

Fig. 2.11.4. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 0

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.11.5.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board.

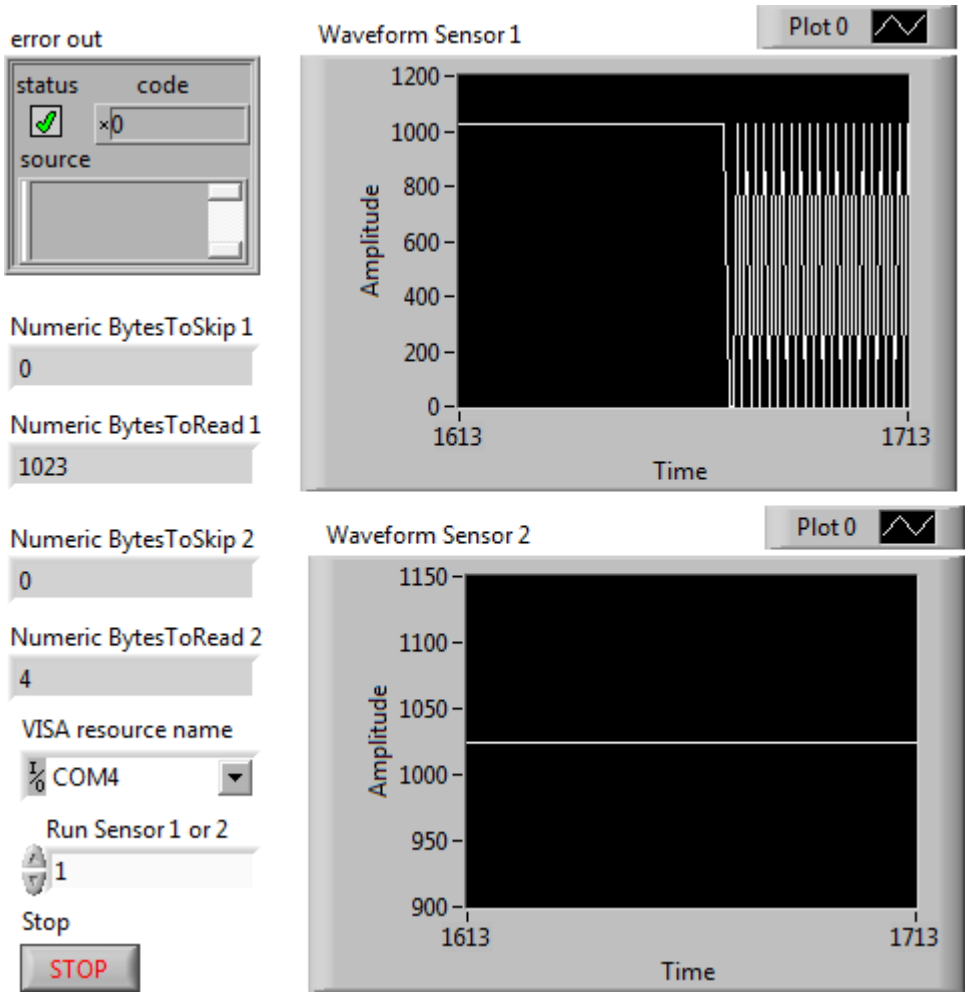

Fig. 2.11.5 Appearance of the inferface control system

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.11.1 and Fig. 2.11.2. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define ctsPin A0 // Alcohol sensor connection
#define levPin 13 // LED Connection
#define Nulled 0 // Default Value
```

```
int count = 0;
int oldStat = 0;
int keyValue;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (levPin, OUTPUT);
 pinMode (ctsPin, INPUT);
 digitalWrite (levPin, HIGH);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
count = Serial.read ();
if (count <0) {
delay (100);
count = oldStat;
}
if (count == 'R') {
keyValue = SetDetectionA0 ();
if (keyValue! = 0) {
Serial.println (keyValue);
digitalWrite (levPin, HIGH);
}
delay (100);
count = 'R';
}
if (count == 'S') {
digitalWrite (levPin, LOW);
```

```
delay (100);
count = 'S';
}
delay (500);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int SetDetectionA0 ()
{
  int data;
  data = analogRead (ctsPin); // Reading the alcohol sensor
  return data;
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the principle of operation of the MQ-3 alcohol vapor sensor.

2.12. Digital fluid level sensor

The purpose of the work is to design a digital device in a Microprocessor-controlled LabVIEW to control a digital liquid level sensor in an Arduino microprocessor programming environment.

Theoretical information

The fluid sensor (Water Sensor) is designed to measure the level of surfactants (Fig. 2.12.1). This easy-to-use and inexpensive liquid level sensor is widely used in automation systems and smart home projects.

Fig. 2.12.1. Digital fluid level sensor

Technical characteristics of the liquid level:

- operating voltage: DC3-5V;

- operating current: less than 20 mA;

- sensor type: analog;

- detection area: 40 x 16 mm;

- production process: FR4 bilateral HASL;

- operating temperature: 10 – 30 C;

- humidity: 10% - 90% without condensation;

- sensor weight: 3.5 g;

- size: 62 x 20 x 8 mm.

The connection of the digital liquid level sensor to the vehicle based on the Arduino UNO controller is shown in fig. 2.12.2

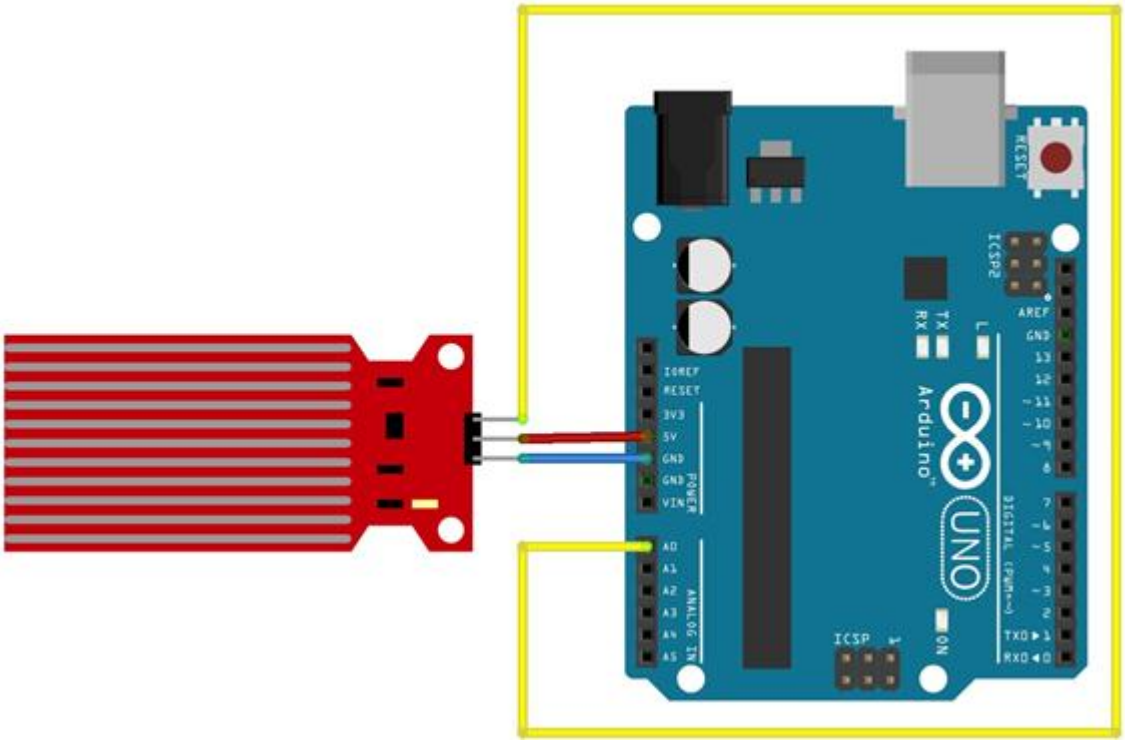

Fig. 2.12.2 Connecting the digital fluid level sensor to Arduino

The conductive method is used to measure the fluid level. The sensor assembly consists of aluminum conductors applied to the textolite. The output signal corresponds to quantization levels calibrated according to the levels of filling the tank.

Work task

1. Compile tested scheme of the virtual instrument for control of the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case Numeric = 2 in the structure "Case Structure" is presented in Fig. 2.12.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case Numeric = 0 in the structure "Case Structure" is presented in Fig. 2.12.4, which corresponds to the stop of the vehicle when pressing the button "Stop".

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.12.5.

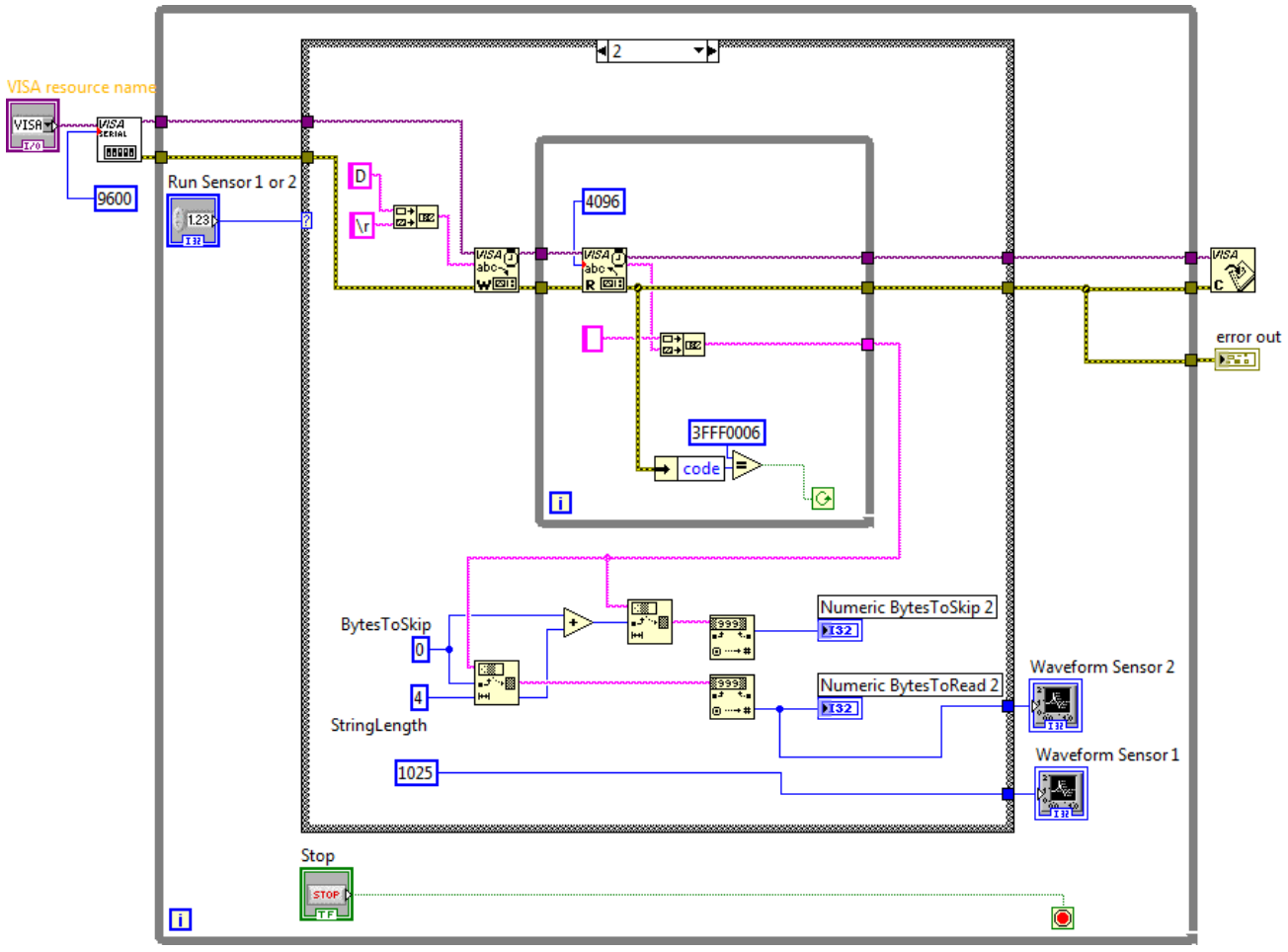

Fig. 2.12.3. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 2

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

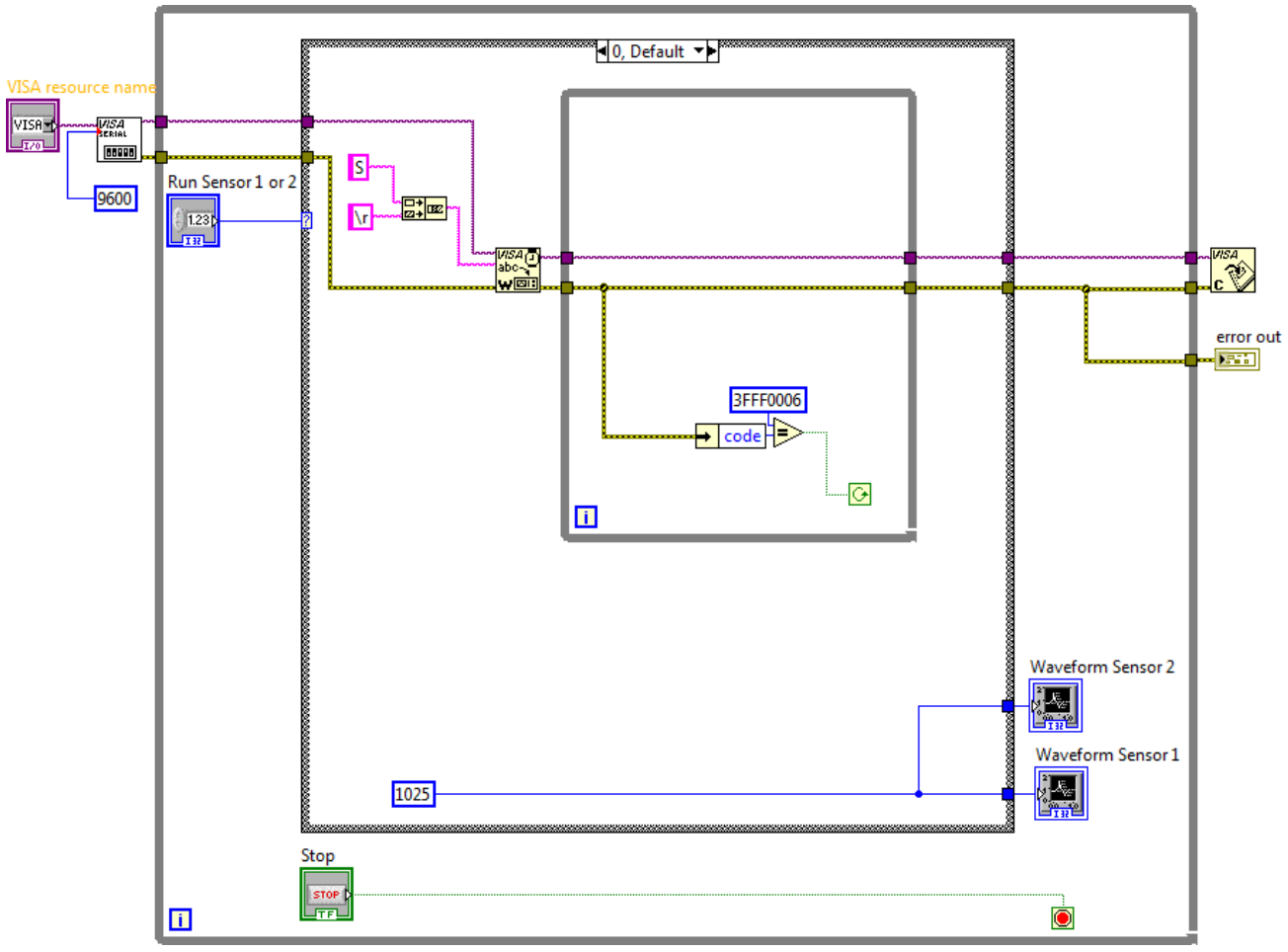

Fig. 2.12.4. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 0

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board

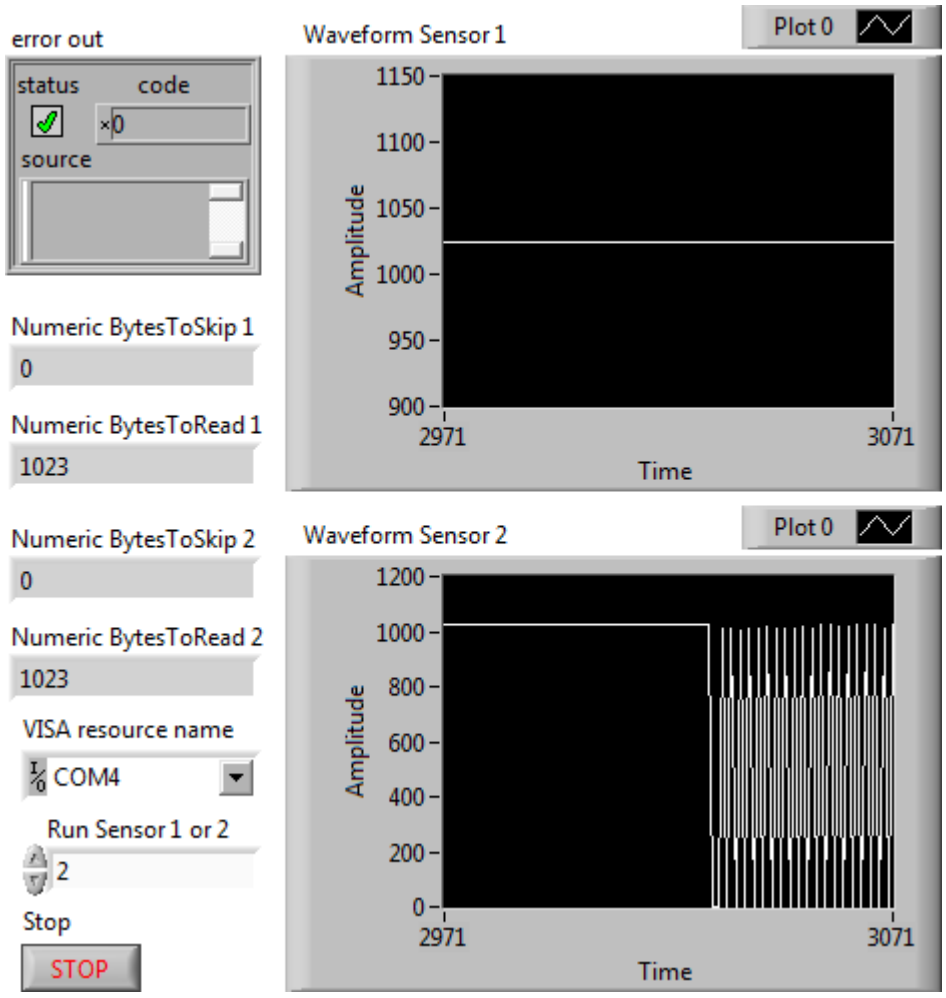

Fig. 2.12.5. The appearance of the inferface control system

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.12.3 and Fig. 2.12.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define ctsPin A0 // Alcohol sensor connection
#define keyPin A5 // Connect the fluid level sensor
#define levPin 13 // LED Connection
```

```
#define Nulled 0 // Default Value
int count = 0;
int oldStat = 0;
int keyValue;
int ctsValue;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
Serial.begin (9600);
pinMode (levPin, OUTPUT);
pinMode (ctsPin, INPUT);
pinMode (keyPin, INPUT);
 digitalWrite (levPin, HIGH);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
count = Serial.read ();
if (count <0) {
delay (100);
count = oldStat;
}
if (count == 'R') {
keyValue = SetDetectionA0 ();
if (keyValue! = 0) {
Serial.println (keyValue);
digitalWrite (levPin, HIGH);
}
delay (100);
count = 'R';
```

```
}
if (count == 'D') {
ctsValue = SetDetectionA5 ();
if (ctsValue! = 0) {
Serial.println (ctsValue);
digitalWrite (levPin, HIGH);
}
delay (100);
count = 'D';
}
if (count == 'S') {
digitalWrite (levPin, LOW);
delay (100);
count = 'S';
}
delay (500);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int SetDetectionA0 ()
{
int data;
data = analogRead (ctsPin); // Reading the alcohol sensor

return data;
}
int SetDetectionA5 ()
{
int data;
data = analogRead (keyPin); // Read fluid level sensor
return data;
```

}

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the principle of operation of the water sensor (Water Sensor).

2.13. Digital fire sensor

The purpose of the work is to design a digital instrument in LabVIEW with microprocessor control to control a digital fire sensor in the Arduino microprocessor programming environment.

Theoretical information

A digital fire sensor lets you detect a flame or light source with a wavelength of 760 - 1100 nm. The sensor confidently detects flames at distances up to 80 cm (Fig. 2.13.1).

Sensor response angle up to 60 degrees. The output of the sensor is digital, 0 or 1. The module is built on the comparator LM393. There is a mounting hole on the board for reliable mounting of the module.

Fig. 2.13.1. Digital fire sensor

Characteristics of the fire sensor:

- operating voltage 3.3 - 5 V;

- wavelength: 760 - 1100 nm;

- distances up to 80 cm;

- operating angle: up to 60 degrees;

- Sensor sensitivity is controlled by setting the potentiometer;

- dimensions: 47.2 x 14 x 7.3 mm;

- weight: 2.1 gr.

The connection of the digital fire sensor to the vehicle based on the Arduino UNO controller is similar to the connection of the digital interference sensor to the vehicle (Fig. 2.8.2).

Used flame sensor with inverted output, which means that it will return False if within its visibility is a flame and true (True) in the absence of an open flame.

Work task

1. To compile the test scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.13.2, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Fig. 2.13.3, which corresponds to stopping the vehicle when pressing the button "Stop".

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.13.4.

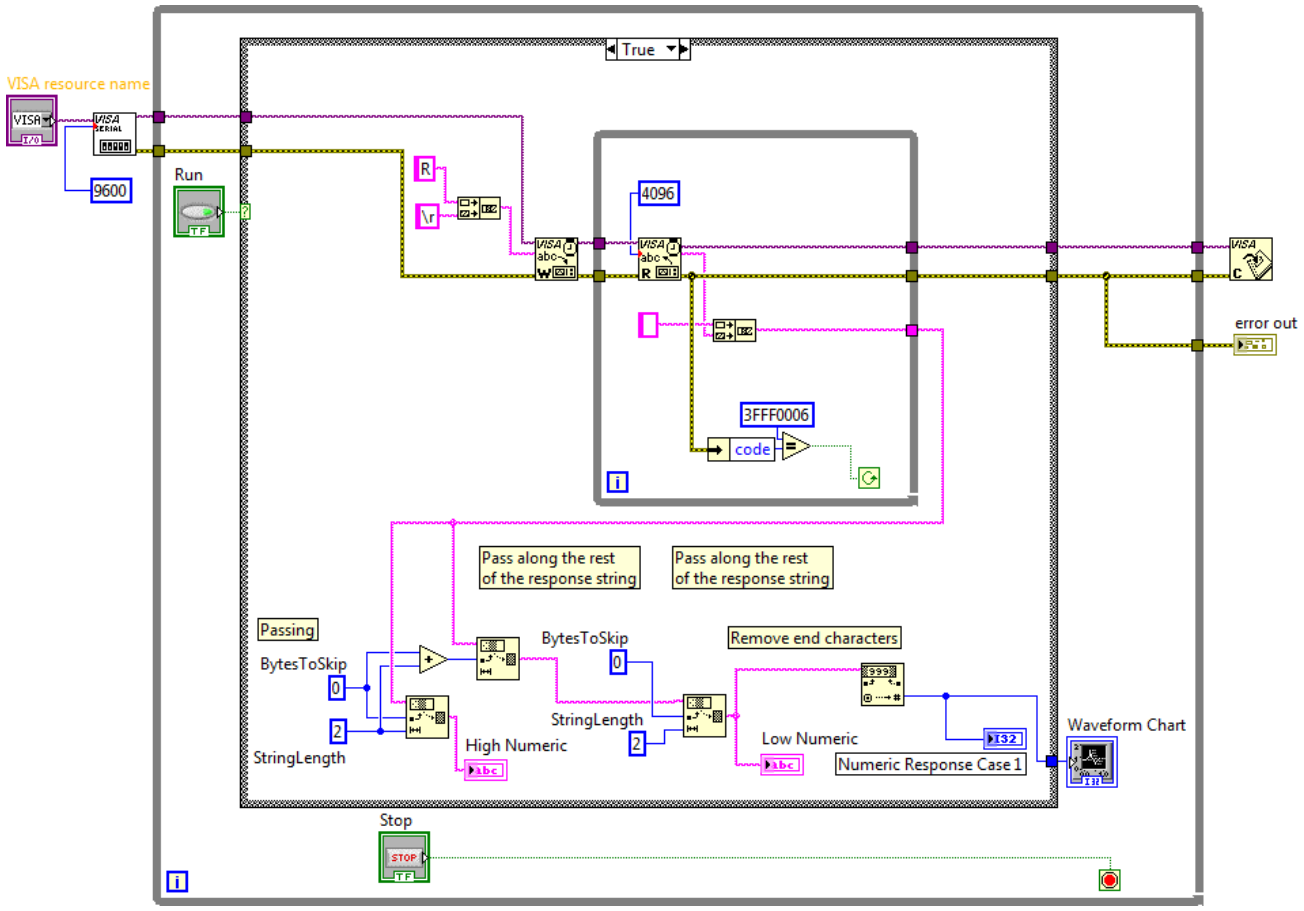

Fig. 2.13.2. Device diagram in NI LabVIEW 2010 for True case

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

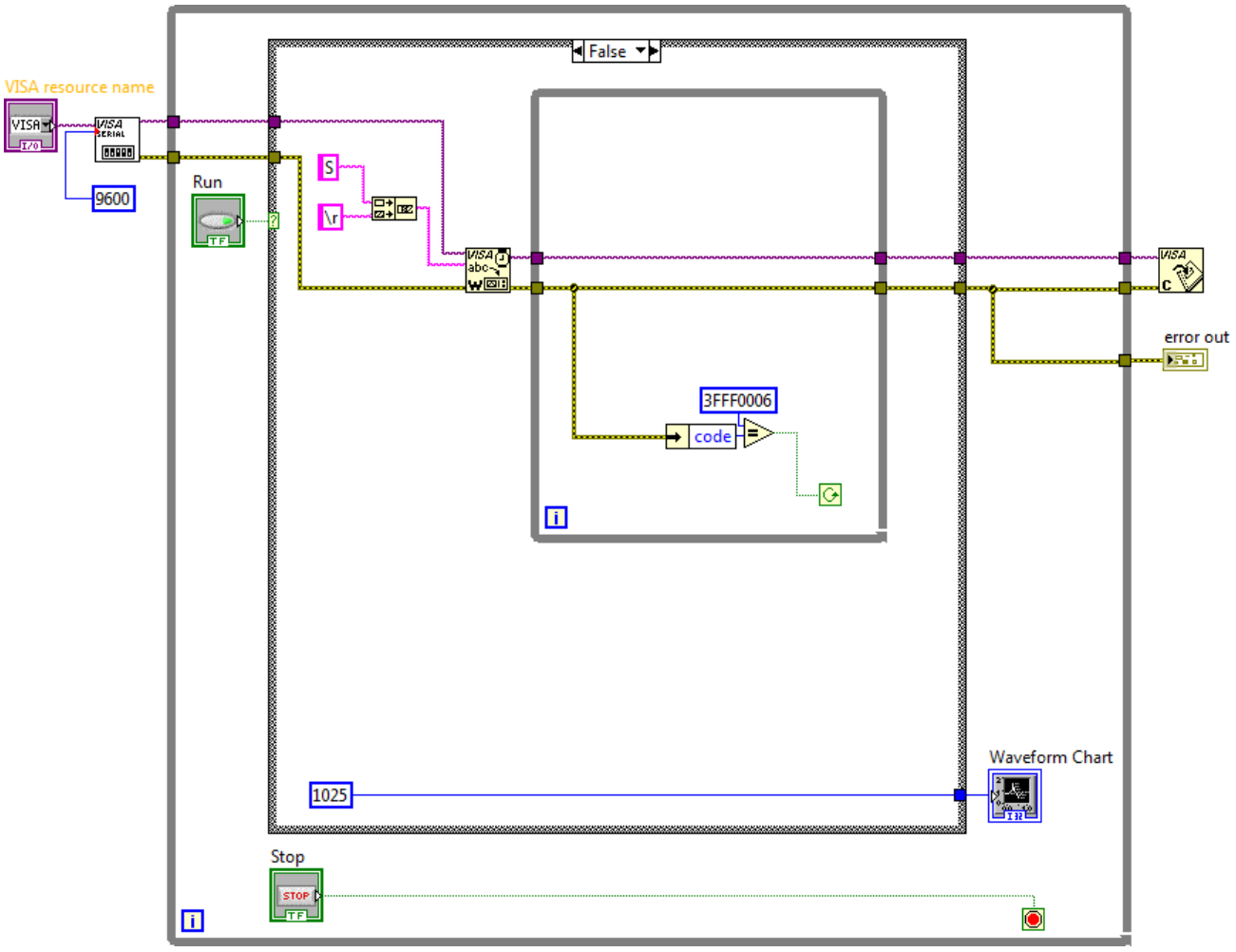

Fig. 2.13.3. Device diagram in NI LabVIEW 2010 for False case

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board.

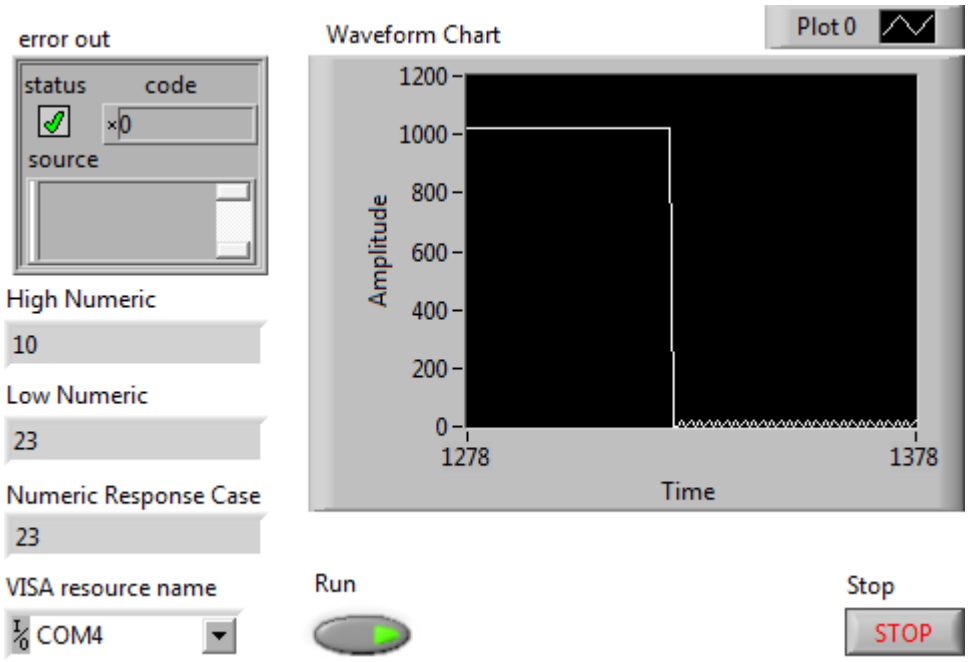

Fig. 2.13.4. The appearance of the interface control system

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.13.2 and Fig. 2.13.3. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define Led 5 // define LED Interface
#define SENSOR 2 // define the Fire sensor interface;
int sensor_val; // define numeric variables;
int signal_val; // define numeric variables;
int readSensor (int valPin); // user function read;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
```

```
 Serial.begin (9600);
 pinMode (Ice, INPUT); // define LED as output interface;
 pinMode (SENSOR, INPUT); // define the Fire sensor line as input;
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
sensor_val = readSensor (SENSOR); // user function
signal_val = digitalRead (Led); // read sensor line
if (signal_val == HIGH) // when the Fire sensor detects light
{
Serial.println ("Fire sensor detects:");
Serial.println (sensor_val);
digitalWrite (Ice, HIGH);
}
if (signal_val <0) // level of detection a light
{
Serial.println ("Not sensor detects");
digitalWrite (Led, LOW);
}
delay (1000);
};
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int readSensor (int valPin) {
int data;
data = digitalRead (valPin); // read sensor line
if (data> 0) // level of detection a light
{
digitalWrite (Ice, HIGH);
} else {
```

```
digitalWrite (Led, LOW);
}
return data;
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the principle of operation of the Fire sensor.

2.14. Analog-digital light sensor

The purpose of the work is to design a digital device in a Microprocessor-controlled LabVIEW to control an analog-to-digital light sensor in the Arduino microprocessor programming environment.

Theoretical information

Analog-digital light sensor is a compact module which includes: light sensor (photoresistor), threshold comparator LM393, analog-to-digital converter, trimmer of the threshold adjustment (Fig. 2.14.1).

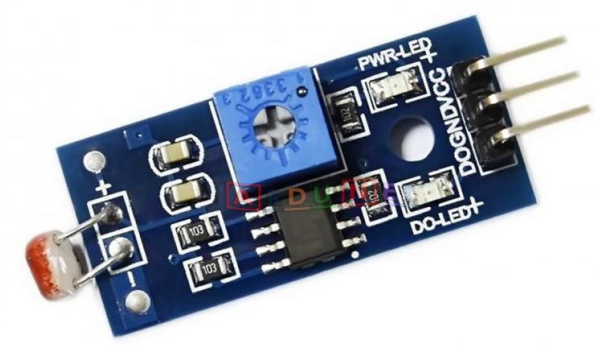

Fig. 2.14.1. Analog-digital light sensor

Characteristics of the light sensor:

- sensitive element: photoresistor;
- comparator output: more than 15 mA;
- operating voltage: from 3.3 V to 5 V;
- comparator LM393;
- adjustment of the operating threshold: variable resistor (trimmer);
- digital output of the comparator (0 and 1);
- analog output of the light sensor;
- mounting hole;
- dimensions: 3.2 cm x 1.4 cm

Purpose of sensor outputs:

- VCC: 3.3-5 V supply voltage input
- GND: total land
- DO: digital output of the comparator
- AO: Analog output of the photosensitive sensor

Sensitive element - the photoresistor changes its resistance depending on the intensity of the incident light. For example, in the dark, the resistance of the photoresistor VT83N1 is 12 kOhm, and under certain test illumination, 100 kOhm. In addition to the photoresistor, the light sensors use a photodiode and a phototransistor.

The connection of the analog-to-digital light sensor to the vehicle using the Arduino UNO controller is similar to the connection of the sensor to the vehicle (Fig. 2.8.2).

Light sensor used in automatic lighting systems, obstacle detection in robotic systems.

Work task

1. To compile the test scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 2.13.2, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Fig. 2.13.3, which corresponds to stopping the vehicle when pressing the button "Stop".

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.13.4.

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.13.2 and Fig. 2.13.3. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define Led 5 // define LED Interface
#define SENSOR A2 // define the Light sensor interface;
int analog_val; // define numeric variables;
int digital_val; // define numeric variables;
int readSensor (int valPin); // user function read;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
  Serial.begin (9600);
  pinMode (Ice, INPUT); // define LED as output interface;
  pinMode (SENSOR, INPUT); // define the Light sensor line as input;
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
```

```
void loop () {
analog_val = readSensor (SENSOR); // user function
digital_val = digitalRead (Led); // read sensor line
if (digital_val == HIGH) // when the Light sensor detects a color
{
Serial.println ("Light sensor detects:");
Serial.println (analog_val);
digitalWrite (Ice, HIGH);
}
if (analog_val <128) // level of detection a light
{
Serial.println ("Not sensor detects");
digitalWrite (Led, LOW);
}
delay (1000);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
int readSensor (int valPin) {
  return analogRead (valPin); // read sensor line
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain how an analog-to-digital light sensor works.

2.15. SW-420 vibration sensor

The purpose of the project is to design a digital instrument in a LabVIEW with microprocessor control to control the vibration sensor SW-420 in the programming environment of Arduino microprocessors.

Theoretical information

Vibration Sensor SW-420 responds to vibration. The module is used for vibration detection in anti-theft systems, burglar alarms and even for earthquakes (Fig. 2.15.1).

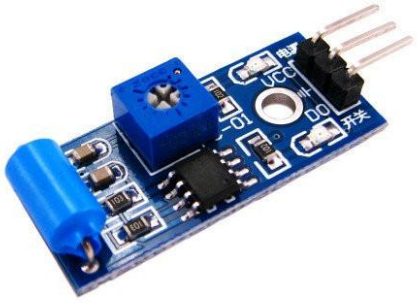

Fig. 2.15.1. SW-420 vibration sensor

There are two LEDs in the module body. The red LED is connected to the power outlet and emits when the power is supplied to the module. Green is connected to the digital output and lights up when the sensor is triggered. The sensitivity of the sensor is governed by the setting of the resistor located on the board.

Features of SW-420 Vibration Sensor:

- supply voltage: 3.3 to 5V;
- output: digital;
- comparator: LM393;
- mounting: mounting hole;
- board sizes: 3.2x1.4 cm.

The connection of the vibration sensor SW-420 to the vehicle based on Arduino UNO is similar to the connection of the tilt sensor on the chip LM393 to the vehicle (Fig. 2.10.2).

The digital output D0 has a logic 1 if no trip and a logic 0 if the sensor is vibrating.

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case Numeric = 1 in the structure "Case Structure" is presented in Fig. 2.15.2, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case Numeric = 0 in the structure "Case Structure" is presented in Fig. 2.15.3, which corresponds to the stop of the vehicle when pressing the button "Stop".

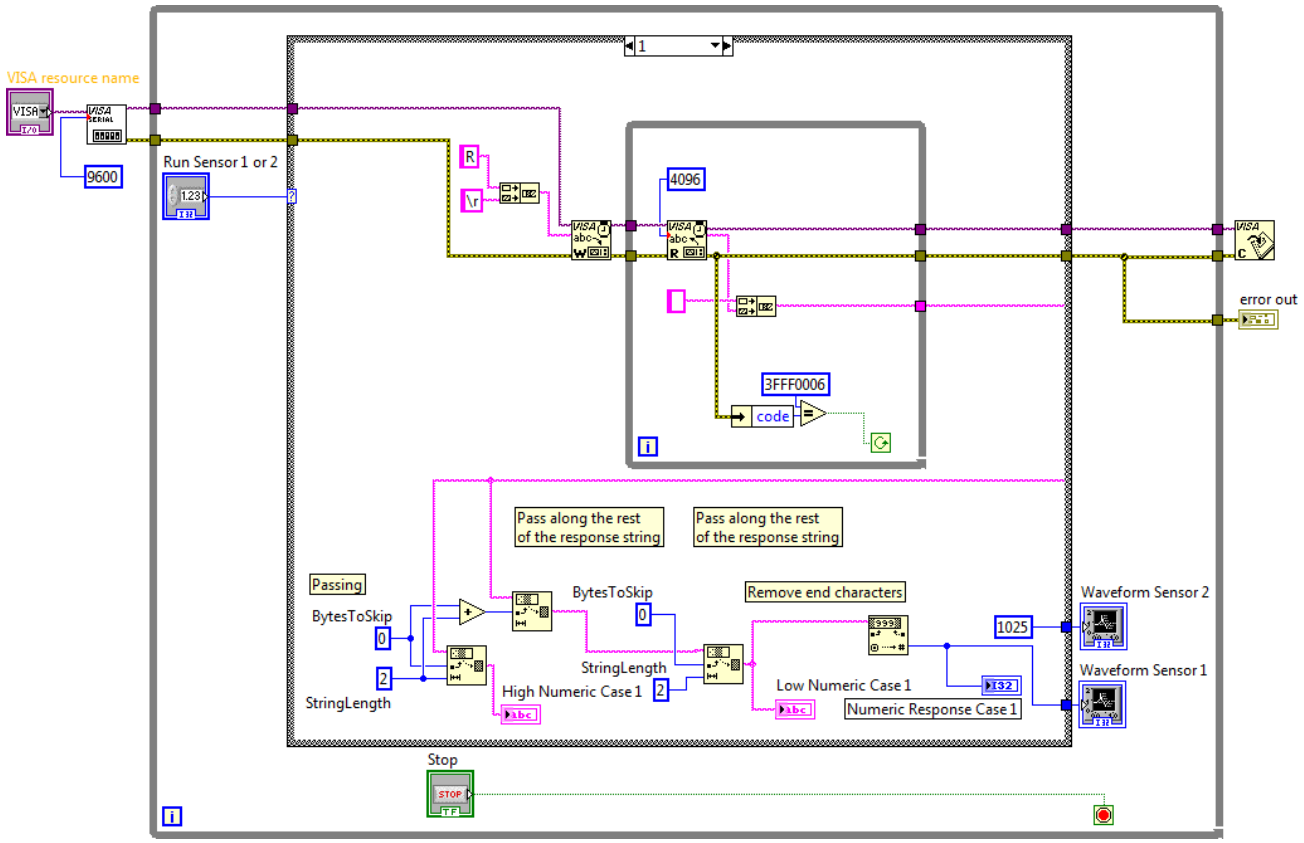

Fig. 2.15.2. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 1

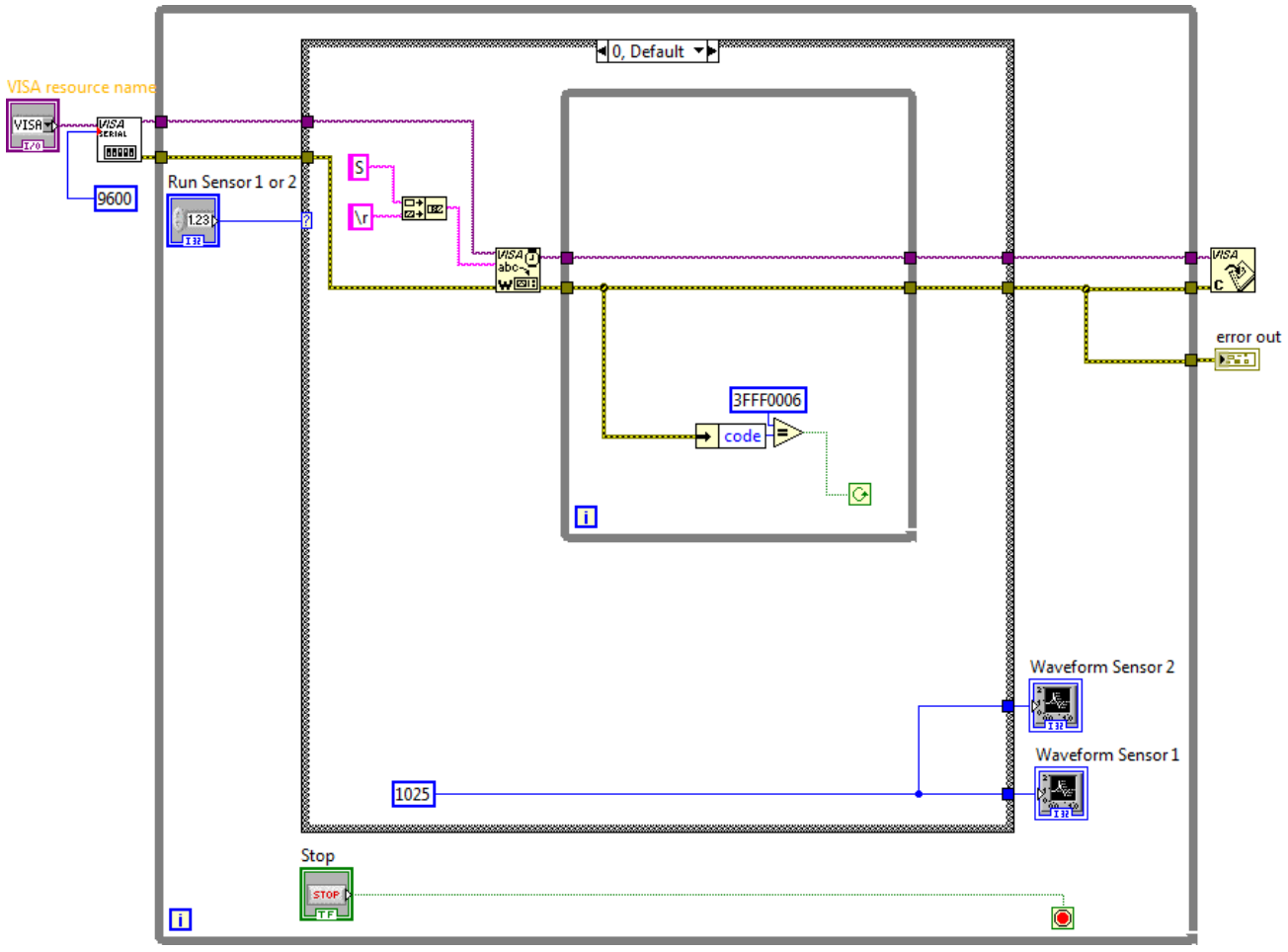

Fig. 2.15.3. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 0

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.15.4.

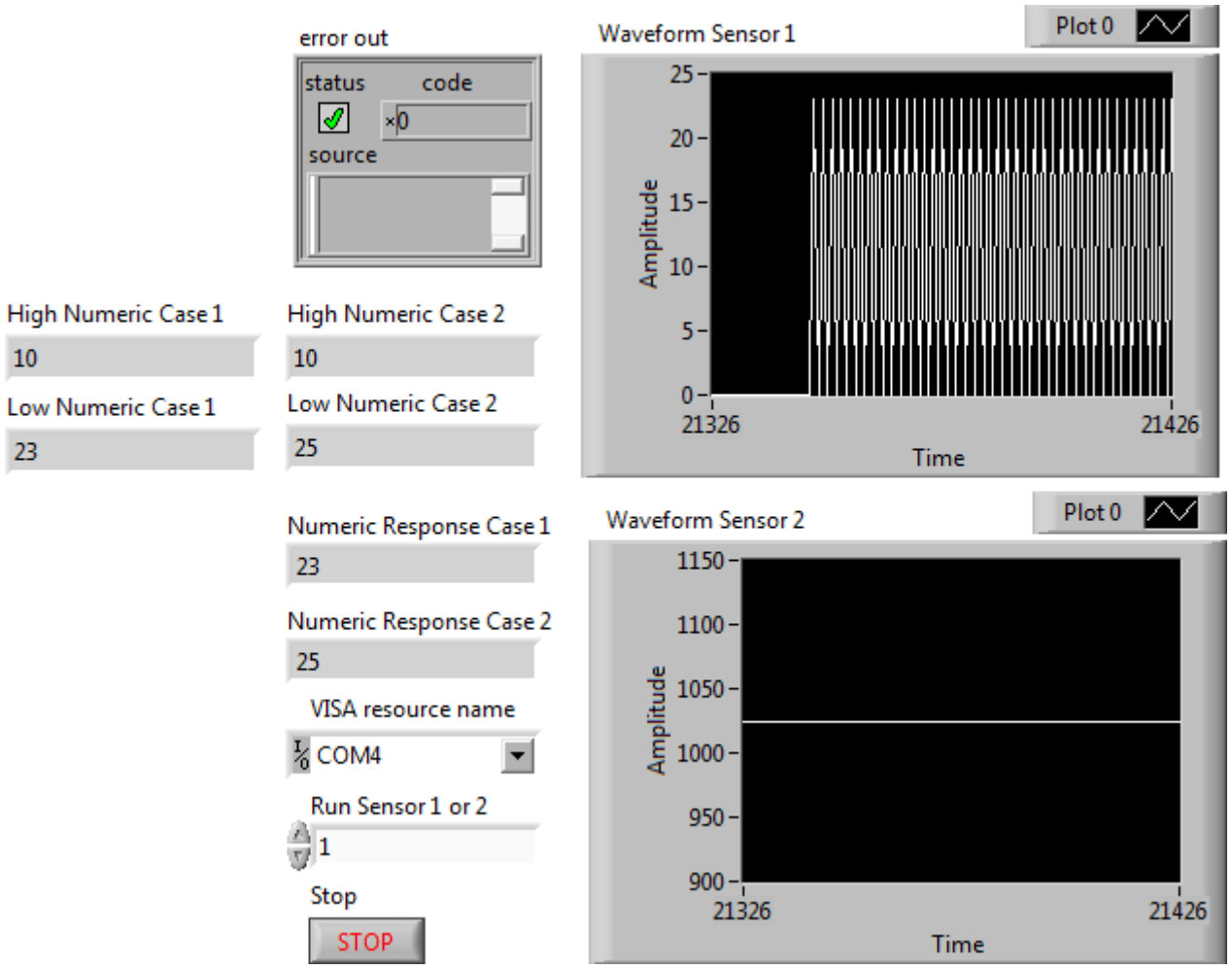

Fig. 2.15.4. The appearance of the interface control system

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.15.2 and Fig. 2.15.3. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define ctsPin 13 // Connect a vibration sensor
#define keyPin A0
#define ledPin 5 // Connect the LED
void vibcount (int count, int maxcount);
const int maxcount = 1024;
int keyState = 0;
int count = 0;
int oldStat = 0;
int ctsValue;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
 Serial.begin (9600);
 pinMode (ledPin, OUTPUT);
 pinMode (ctsPin, INPUT);
 pinMode (keyPin, INPUT);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
```

```
void loop () {
count = Serial.read ();
if (count <0) {
delay (100);
count = oldStat;
}
if (count == 'R') {
ctsValue = digitalRead (ctsPin);
keyState = analogRead (keyPin);
if (ctsValue == HIGH) {
// Sig Output in high, sensor switched on
digitalWrite (ledPin, HIGH);
Serial.println ("TOUCHED");
} else {
digitalWrite (ledPin, LOW);
Serial.println ("not touched");
}
if (keyState! = oldStat) {
count = 0;
oldStat = keyState;
}
discount (count, maxcount);
delay (500);
count = 'R';
```

```
}
if (count == 'S') {
delay (100);
count = 'S';
}
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
void vibcount (int count, int maxcount) {
 if (count <maxcount) {
    digitalWrite (ledPin, HIGH);
    Serial.println ("level Count:");
    Serial.println (keyState); // display
 }
 else {
    Serial.println ("max Count:");
    Serial.println (keyState); // display
    digitalWrite (ledPin, LOW);
 }
 if (count <= maxcount + 1) {
    ++ count;
 }
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010.

Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain how the SW-420 vibration sensor works.

2.16. Digital Impact Sensor

The purpose of the work is to design a digital device in a Microprocessor-controlled LabVIEW to control a digital impact sensor in the Arduino microprocessor programming environment.

Theoretical information

The KY-031 digital impact sensor detects impacts on the body of the object in which it is mounted (Fig. 2.16.1). It is installed in the security systems of cars, motorcycles, bicycles. The impact sensor module is used in smart home security systems.

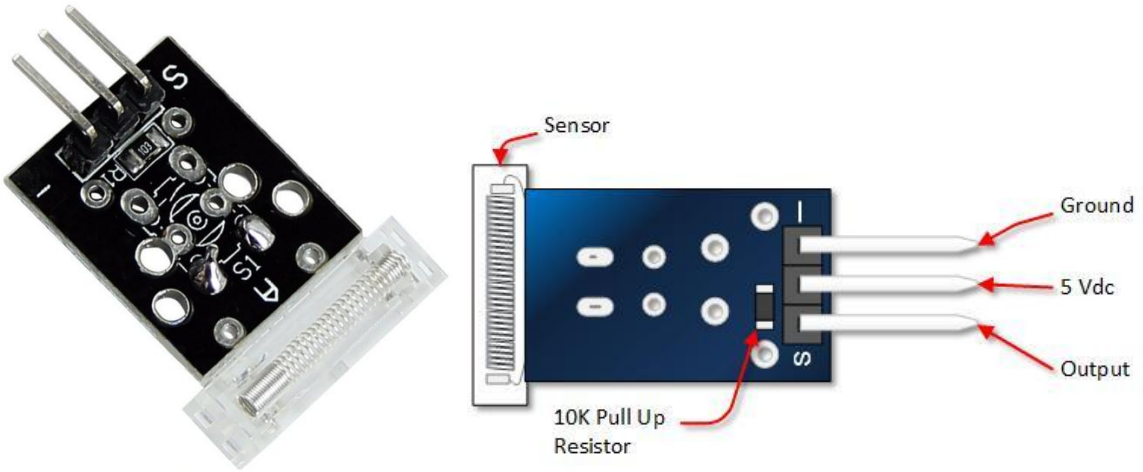

Fig. 2.16.1. Digital Impact Sensor

The sensor module is most sensitive to shocks directed across the plane of the board. Impact is perceived by a sensitive element, which is a spring, the end of which is surrounded by contacts. A 10 kO resistor is located between the power input and the impact sensor output. When struck, the spring bends, the end of the spring touches the contacts and the sensor circuit KY-031 closes. When the sensor is triggered, the contact closes, which can be connected to the input of various devices.

Connecting a digital impact sensor to an Arduino UNO-based vehicle is shown in Fig. 2.16.2.

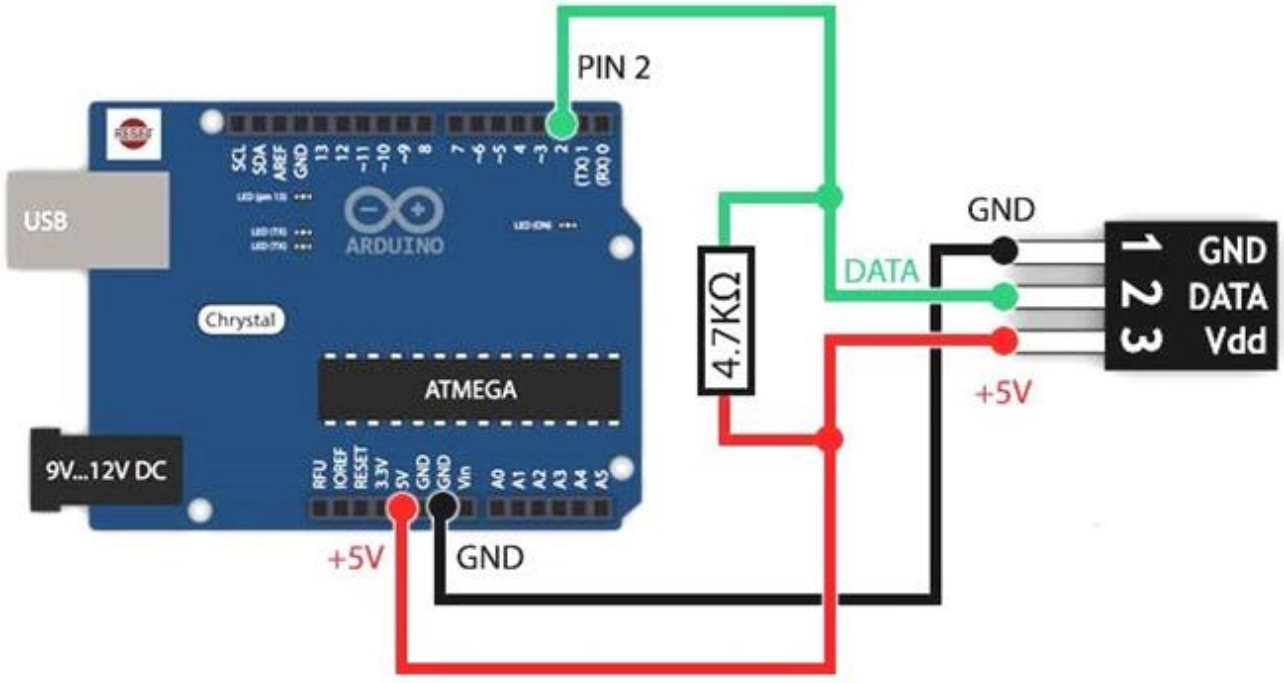

Fig. 2.16.2. Connecting the digital impact sensor to the vehicle

When using a power contact (VDC) at the output of the sensor (Output) in the absence of operation, the voltage is present, and when triggered short pulses are recorded in the range from 5 V to 0 V.

Work task

1. Compile the tested scheme of the virtual instrument for control of the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the

case Numeric = 2 in the structure "Case Structure" is presented in Fig. 2.16.3, which corresponds to the launch of the vehicle. The block diagram of the virtual instrument for the case Numeric = 0 in the structure "Case Structure" is presented in Fig. 2.16.4, which corresponds to the stop of the vehicle when pressing the button "Stop".

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 2.16.5.

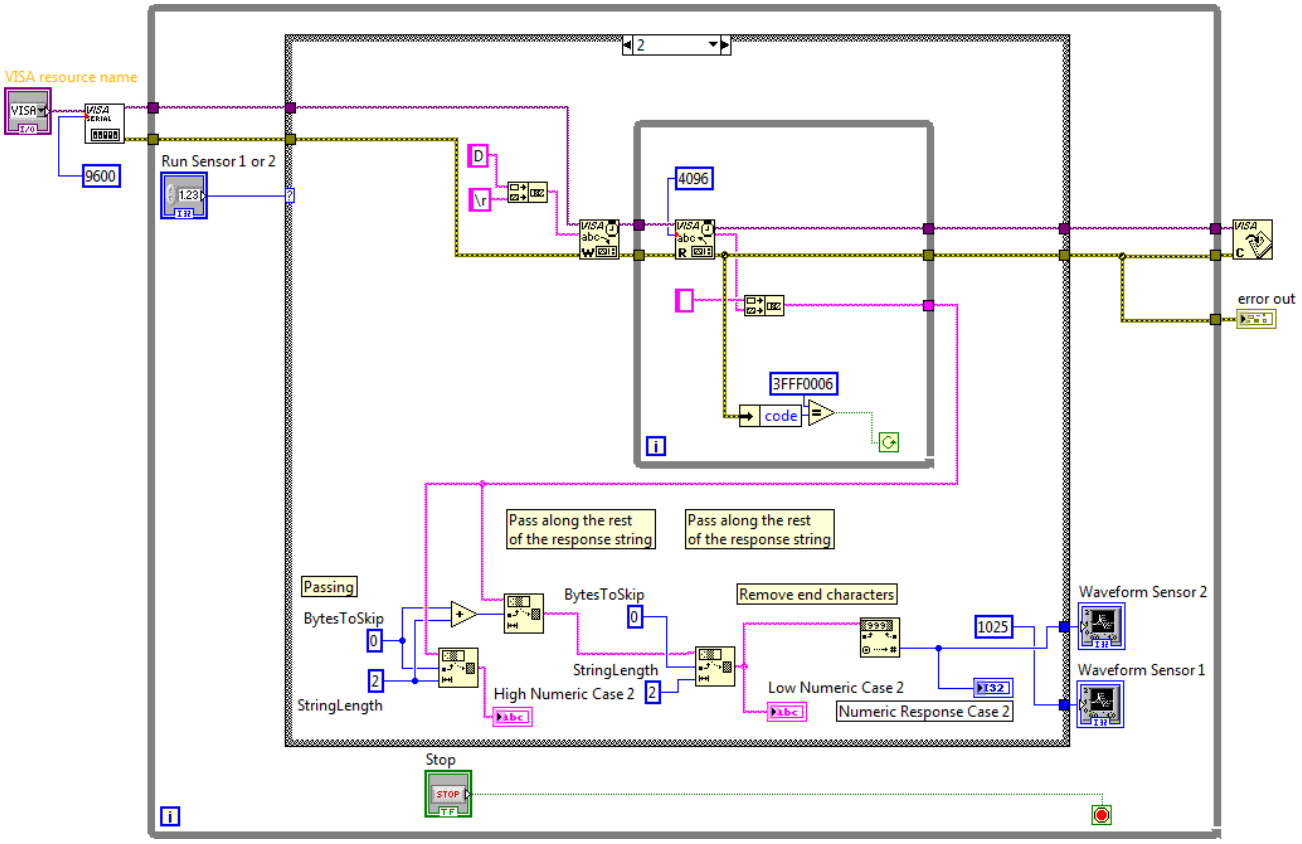

Fig. 2.16.3. Schematic of the instrument in NI LabVIEW 2010 for the case Num = 2

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

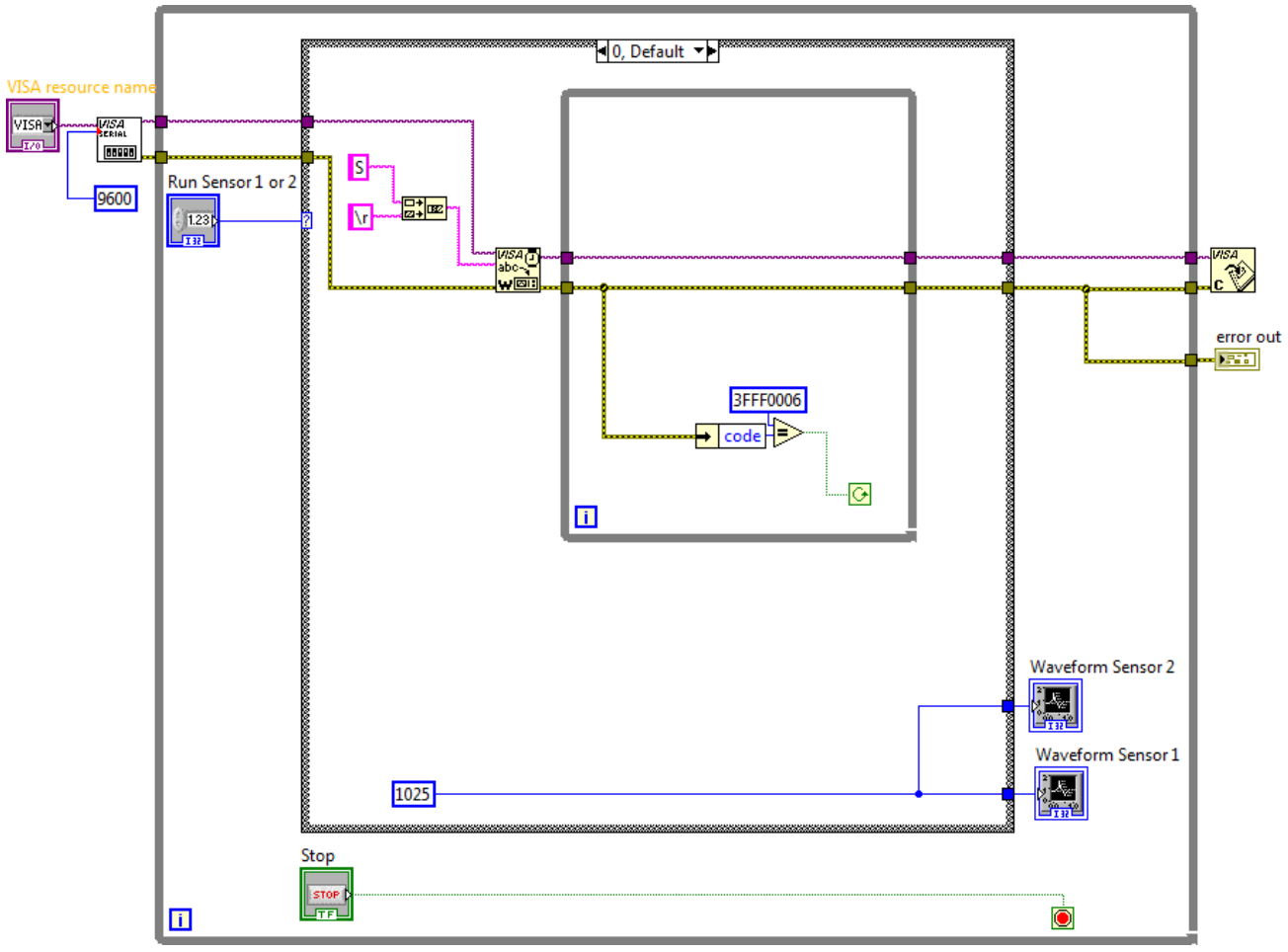

Рис. 2.16.4. Схема приладу в NI LabVIEW 2010 для випадку Numeric = 0

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on an Arduino UNO board.

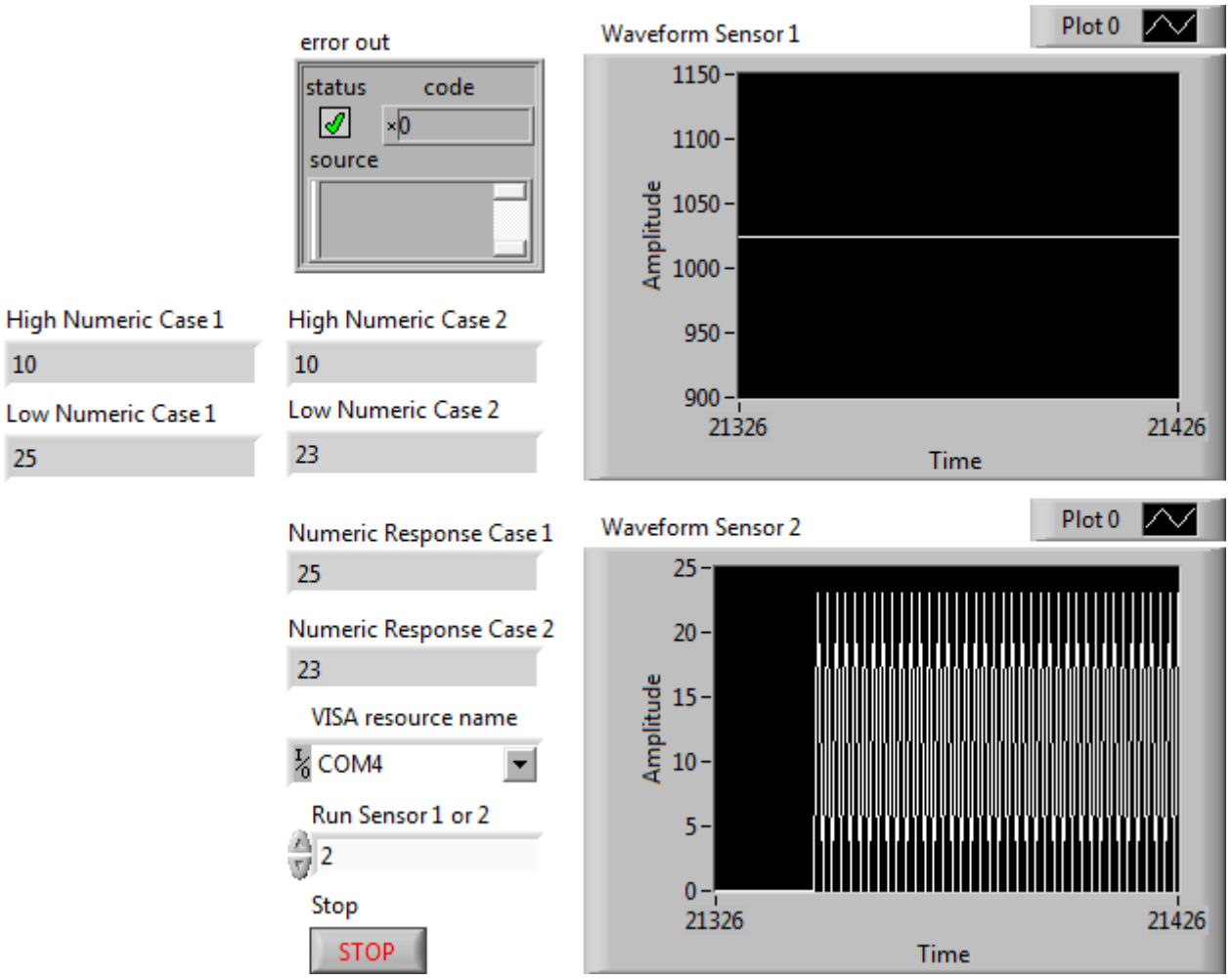

Fig. 2.16.5. The appearance of the interface control system

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 2.16.3 and Fig. 2.16.4. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables
#define ctsPin 13 // Connect a vibration sensor
#define keyPin A2 // Connect the shock sensor
```

```
int count = 0;
int oldStat = 0;
int keyState;
int ctsValue;
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
  Serial.begin (9600);
  pinMode (ctsPin, INPUT); // Adjust the vibration sensor
  pinMode (keyPin, INPUT); // Adjust the impact sensor
  Serial.println ("Start");
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
 count = Serial.read ();
 if (count <0) {
 delay (100);
 count = oldStat;
 }
 if (count == 'R') {
 keyState = SetDetectionD13 ();
 Serial.println (keyState);
 delay (100);
 count = 'R';
 }
 if (count == 'D') {
 ctsValue = SetDetectionA5 ();
 Serial.println (ctsValue);
 delay (100);
```

```
 count = 'D';
 }
 if (count == 'S') {
 delay (100);
 count = 'S';
 }
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
// Read the vibration sensor
int SetDetectionD13 () {
 int data;
 data = digitalRead (ctsPin);
 return data;
}
// Read the impact sensor
int SetDetectionA5 () {
 int data;
 data = analogRead (keyPin);
 return data;
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the operation of the KY-031 impact sensor.

3. Board for connection of sensors

Arduino Microcontroller Boards allow you to connect a wide variety of modules such as sensors, servo drives, relays, buttons, potentiometers, and more. All peripherals for Arduino boards offered by hardware manufacturers include sockets in a variety of form factors to further connect to end modules. The board or shield is worn directly on the Arduino controller and does not require additional commutation and soldering - which is very easy to design and operate.

3.1. Multifunctional educational signboard

The purpose of the work is to design a digital device in a Microprocessor-controlled LabVIEW to control multiple digital sensors using a multifunction shield in the Arduino microprocessor programming environment.

Theoretical information

The multifunctional training signage is designed to familiarize you with the Arduino platform, control the peripherals and work with the sensors. The board gives you hands-on experience in working with multiple digital sensors, shift registers, control of seven-segment indicators, analog and digital temperature sensors, input from buttons, Bluetooth modules, etc. Shield can be used both for training and in industrial designs. The presented implementation of multifunctional shielded MPS minimizes the risk of failure of the Arduino UNO controller elements if they are not properly connected.

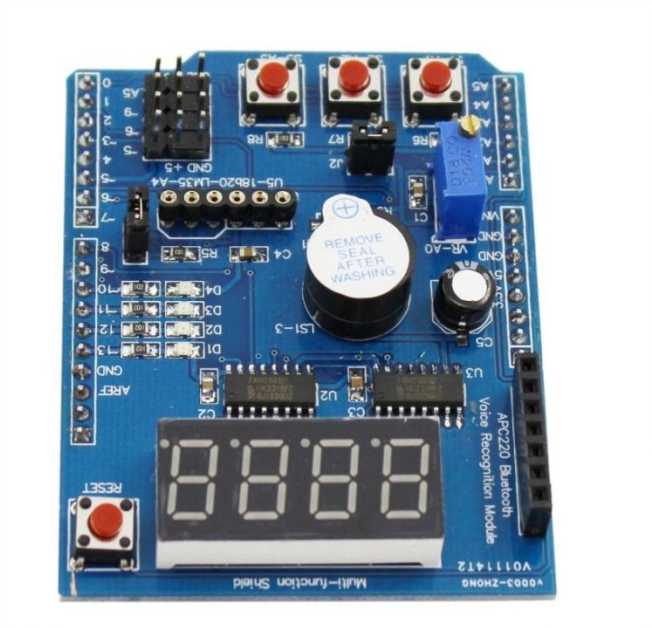

Fig. 3.1.1. Multifunction Shield for Arduino UNO

Features of multifunction shield:

- supply voltage: 5V;
- input devices: clock buttons (A1, A2, A3);
- dimensions: 69 x 53.5 mm.

Input interfaces:

- digital: DS18B20 (A4) (with tripping resistor);
- Analog: LM35 (A4), replaceable resistor (A0).

Output interfaces:

- sound emitter (D3) with additional transistor;
- LEDs (D10, D11, D12, D13);
- seven-segment LED matrix (SPI);
- Bluetooth connector APC220;
- APC220 voice module connector.

Additional interfaces:

- additional power connector (Vcc, Gnd);
- connectors for servomotors connection (D5, D6, D9);
- connector for connection of analog sensors (A5);
- infrared sensor connector (D2).

Work task

1. Compile the tested scheme of the virtual instrument for controlling the Arduino UNO board in National Instruments LabVIEW 2010. The block diagram of the virtual instrument for the case True in the structure "Case Structure" is presented in Fig. 3.1.2 and fig. 3.1.3, which corresponds to the launch of a vehicle for measurement with two sensors. The block diagram of the virtual instrument for the case of False in the structure "Case Structure" is presented in Fig. 3.1.4 and fig. 3.1.5, which corresponds to the waiting and stopping of the vehicle when the tumbler is switched off or the Stop button is pressed.

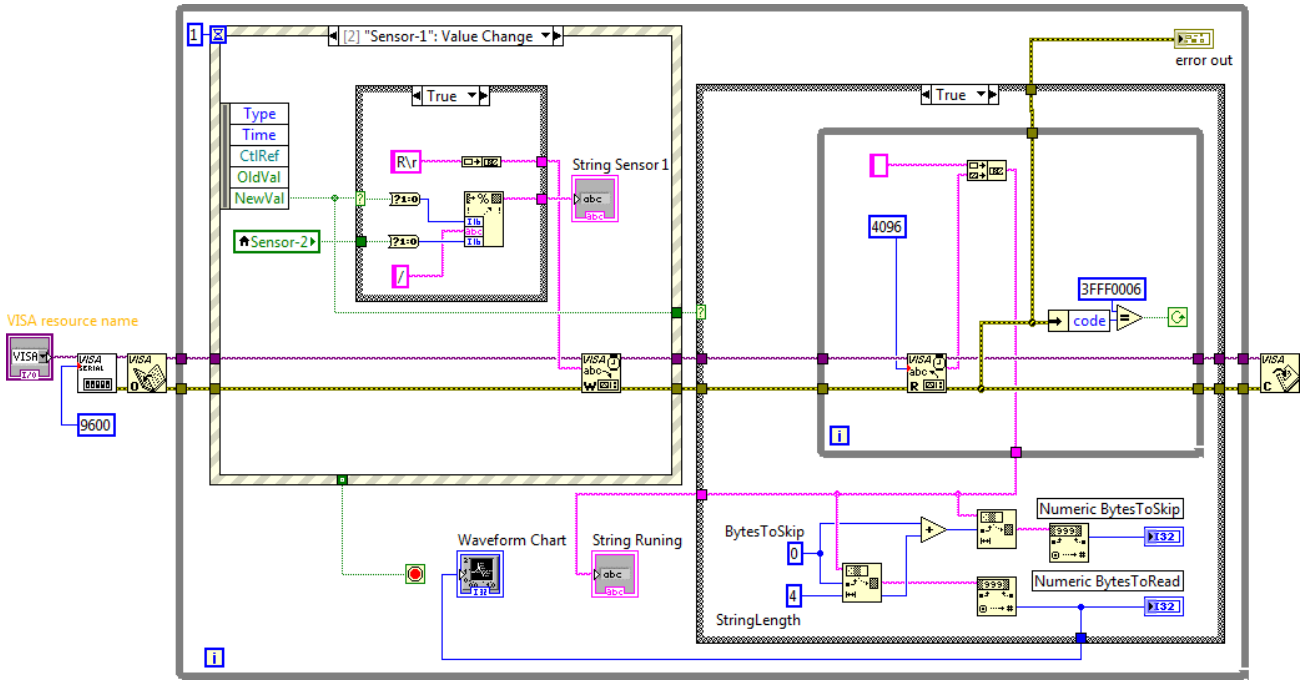

Fig. 3.1.2. Schematic of the instrument in NI LabVIEW 2010 for True Sensor - 1

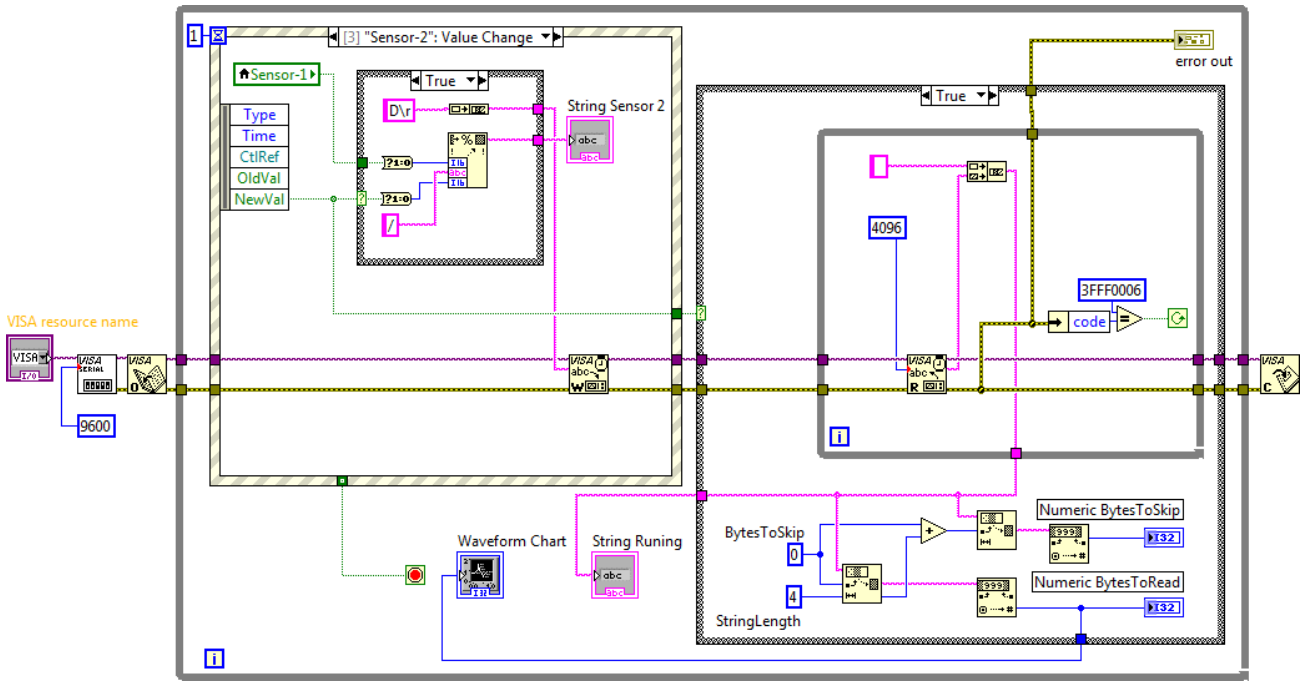

Fig. 3.1.3. Schematic of the instrument in NI LabVIEW 2010 for True Sensor - 2

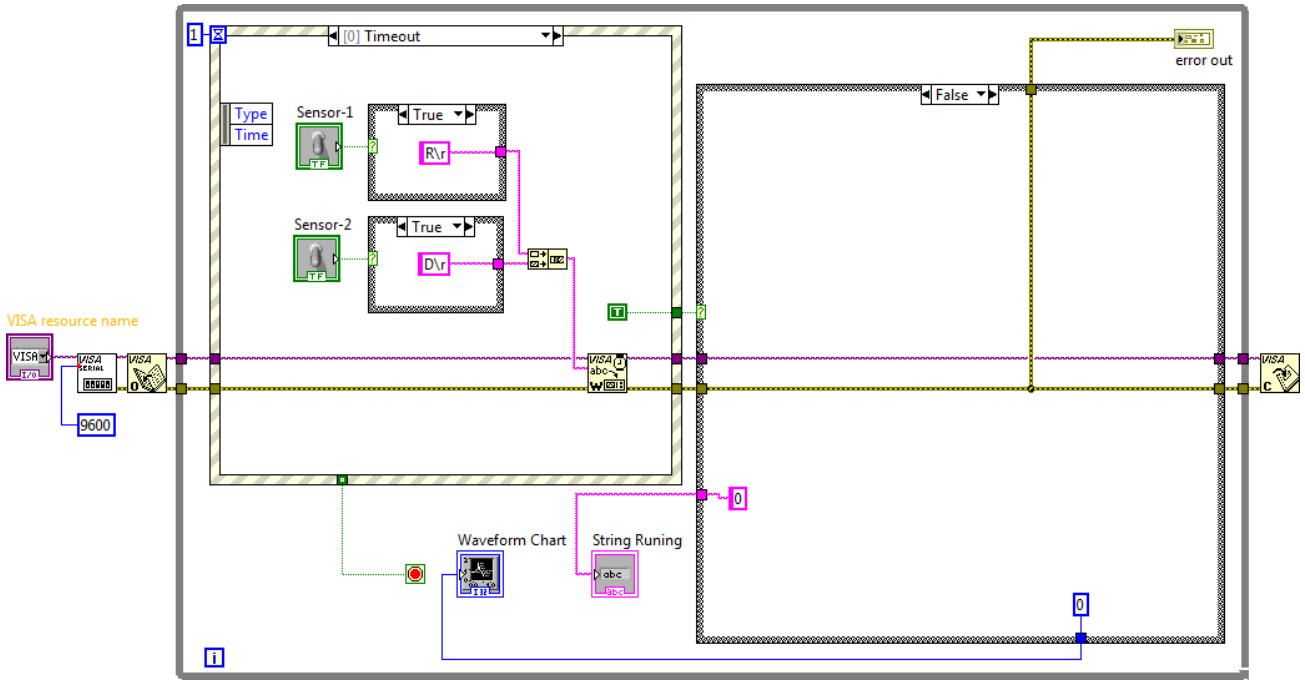

Fig. 3.1.4. Schematic of the instrument in NI LabVIEW 2010 for False - Timeout

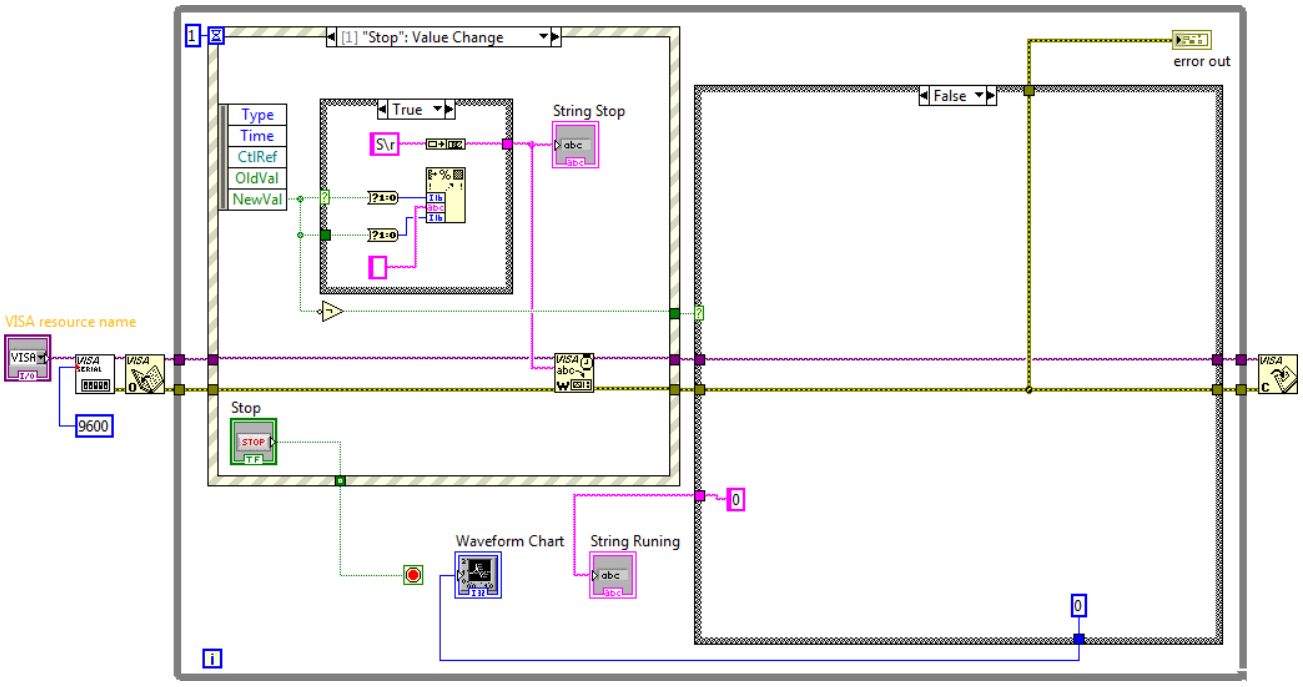

Fig. 3.1.5. Schematic of the instrument in NI LabVIEW 2010 for False - Stop

The appearance of the system in the National Instruments LabVIEW 2010 for control of vehicles based on the Arduino UNO microcontroller is shown in Fig. 3.1.6.

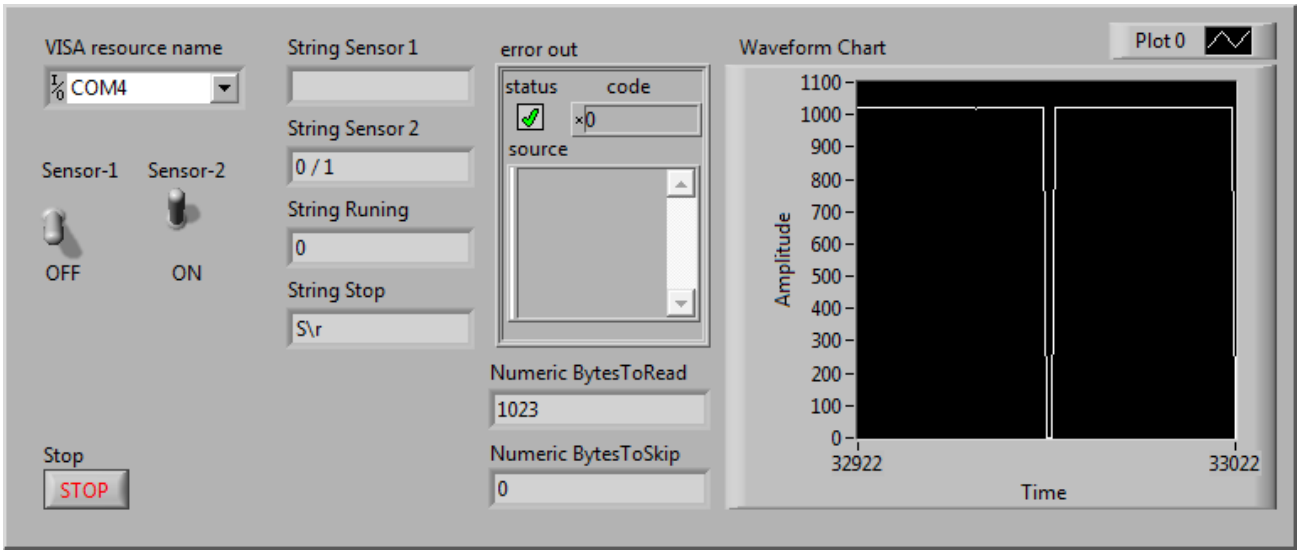

Fig. 3.1.6. The appearance of the interface control system

2. Explore the functionality of LabVIEW 2010 components for the NI Programming Function VI virtual interface. Describe the purpose and functions of the NI VISA serial I / O port in the Arduino microprocessor programming environment.

3. Document the data obtained in LabVIEW using the Waveform Graph tools. Explain the code execution algorithm in the Arduino programming environment and the Serial Port read / write functions to control the vehicle on the Arduino UNO board.

Formulate in the report on the completed work the findings of the research and prepare answers to the control questions.

Methodical instructions

1. To perform this laboratory work, you must use the controls National Instruments LabVIEW 2010, shown in Fig. 3.1.2 - Fig. 3.1.5. Initialize data and variables in a work project in the Arduino microprocessor programming environment:

```
// Initialize data and variables

#define levPin A5 // Sensor connection output

#define trigPin 5 // Trigger Pin

/ * Define shift register pins used for seven segment display * /

#define LATCH_DIO 4

#define CLK_DIO 7

#define DATA_DIO 8

#define Pot1 0;

/ * Segment byte maps for numbers 0 to 9 * /

const byte SEGMENT_MAP [] =
{0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0X80,0X90};

/ * Byte maps to select digit 1 to 4 * /

const byte SEGMENT_SELECT [] = {0xF1,0xF2,0xF4,0xF8};

int ledPin = 9; // LED connection output

int levValue; // max Value = 780
```

2. Pre-configure the Arduino UNO ICU in the Arduino microprocessor programming environment:

```
// Configure the microcontroller
void setup () {
Serial.begin (9600);
/ * Set DIO pins to outputs * /
pinMode (LATCH_DIO, OUTPUT);
pinMode (CLK_DIO, OUTPUT);
pinMode (DATA_DIO, OUTPUT);
pinMode (trigPin, OUTPUT);
pinMode (ledPin, OUTPUT);
pinMode (levPin, INPUT);
}
```

3. Use Serial Port methods in Arduino microprocessor programming environment:

```
/ / Implementation of the control of the vehicle
void loop () {
int PotValue;
levValue = analogRead (levPin);
PotValue = analogRead (Pot1);
Serial.print ("Sig Output Level:");
Serial.println (levValue);
/ * Update the display with the current counter value * /
WriteNumberToSegment (0, levValue / 1000);
WriteNumberToSegment (1, (levValue / 100)% 10);
WriteNumberToSegment (2, (levValue / 10)% 10);
```

```
WriteNumberToSegment (3, levValue% 10);
}
```

4. Implement procedures for processing user functions in the Arduino microprocessor programming environment:

```
/ * Write a decimal number between 0 and 9 to one of the 4 digits of the display * /
void WriteNumberToSegment (byte Segment, byte Value) {
digitalWrite (LATCH_DIO, LOW);
shiftOut (DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_MAP [Value]);
shiftOut (DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_SELECT [Segment]);
digitalWrite (LATCH_DIO, HIGH);
}
```

Download Arduino UNO microcontroller software developed in the Arduino programming environment to the MPS laboratory layout. Run a virtual instrument project in NI LabVIEW 2010. Use the Waveform Graph tools to get virtual instrument specifications.

Control questions

1. What are the technical characteristics of the vehicle based on the Arduino microcontroller?
2. What is the function of NI VISA serial I / O port?
3. Explain the methods of working with Serial Ports.
4. Explain the principle of data output to the LED matrix (SPI)

Literature

1. Gusev, V.G. Electronics and microprocessor technology: Textbook / V.G. Gusev, Yu.M. Gusev. - M .: KnoRus, 2013. - 800 p.

2. Bramm, P. Microprocessor 80386 and its programming / P. Bramm, B. Bramm. - M .: Mir, 1990.

3. Grebenko, Yu.A. Microprocessors / Yu.A. Grebenko, V.K. Rakov. - M .: Publishing House of MEI, 2000.

4. Gurevich, VI Microprocessor protection relays. Device, problems, prospects / VI Gurevich. - M .: Infra-Engineering, 2011. - 336 p.

5. Ivannikov, AD Modeling of microprocessor systems / AD. Ivannikov. - M .: Energoatomizdat, 1990.

6. Kalabegov, BA Digital devices and microprocessor systems. Kalabegov. - M .: Hotline-Telecom, 2007.-336c.

7. Kalabekov, BA Digital Devices and Microprocessors. Kalabekov. - M .: Radio and communication, 1997.

8. Kalashnikov, VI Electronics and microprocessor technology: A textbook for students. institutions of higher education. prof. arr. / VI. Kalashnikov, S.V. Nefedov. - M .: IC Academy, 2012. - 368 p.

9. Kuzin, A.V. Microprocessor technology: A textbook for students. average prof. education / A.V. Kuzin, MA Larks. - M .: IC Academy, 2013. - 304 p.

10. Novikov, Yu.V. Fundamentals of microprocessor technology: a lecture course / Yu.V. Novikov, PK Soon rich. - M .: INTUIT.RU, 2003. - 440 p.

11. Novikov, Yu.V. Fundamentals of microprocessor technology: Textbook / Yu.V. Novikov, PK Soon rich. - M .: BINOM. LZ, INTUIT.RU, 2012. - 357 p.

12. Novozhilov, OP Basics of microprocessor technology. In 2 vols T. 2. Fundamentals of microprocessor technology: Textbook / OP. Novozhilov. - M .: IP RadioSoft, 2011. - 336 p.

13. Shevkoplyas, B.V. Microprocessor structures. Engineering Solutions: A Handbook / B.V. Shevkoplyas. - M .: Radio and communication, 1993.

14. Schonfelder, G. Measuring devices based on the ATmega microprocessor: Trans. with English. / G. Schoenfelder, K. Schneider. - St. Petersburg: BHC-Petersburg, 2012. - 288 p.