Drohobych Ivan Franko State Pedagogical University
Department of Physics and Information Systems

Roman LESHKO, Olha LESHKO

STEM-TECHNOLOGIES BASED ON THE ARDUINO BOARD

Guidelines for Performing Laboratory Works

Drohobych
2024

Дрогобицький державний педагогічний університет імені Івана Франка
Кафедра фізики та інформаційних систем

Роман ЛЕШКО, Ольга ЛЕШКО

STEM-ТЕХНОЛОГІЇ НА ОСНОВІ ПЛАТФОРМИ ARDUINO

Рекомендації для лабораторних робіт

Дрогобич
2024

UDC 004.9(072)
Л53

Recommended for publication by the Academic Council of Drohobych Ivan Franko State Pedagogical University as a study guide (protocol No. 11 dated 31.10.2024).

Reviewers:

- Associate Professor of the Department of Physics and Information Systems at Drohobych Ivan Franko State Pedagogical University, PhD in Physics and Mathematics, **Oleh KUZUK**.
- Associate Professor of the Department of Technological and Professional Education at Drohobych Ivan Franko State Pedagogical University, PhD in Physics and Mathematics, **Yuriy PAVLOVSKYI**.

Responsible for the publication – **Viktor BRYTAN**, PhD in Physics and Mathematics, Associate Professor of the Department of Physics and Information Systems at Drohobych Ivan Franko State Pedagogical University.

Roman Leshko, Olha Leshko.
STEM-technologies based on the Arduino board [Guidelines for Performing Laboratory Works]. Drohobych **:** Drohobych Ivan Franko State Pedagogical University, 2024. 58 p.

The study guide "**STEM-technologies based on Arduino board**" is written in accordance with the curricula of the academic discipline "**Information Control Systems and STEM-technologies**" for the teaching of master's level specialists in the fields of 014 Secondary Education (Physics), 104 Physics and Astronomy, and 122 Computer Science, approved by the Academic Council of Drohobych Ivan Franko State Pedagogical University.

The guide presents laboratory work that serves as a foundation for mastering information control systems and STEM-technologies, gathering the main methods and principles of working with Arduino boards. The guide is characterized by a combination of theoretical information with practical instructions and assignments for laboratory work.

Bibliography: 10 titles

УДК 004.9(072)
Л53

Рекомендовано вченою радою Дрогобицького державного педагогічного університету імені Івана Франка
(протокол № 11 від 31.102024 р.).

Рецензенти:

- **Олег КУЗИК,** доцент кафедри фізики та інформаційних систем Дрогобицького державного педагогічного університету імені Івана Франка, кандидат фізико-математичних наук, доцент;
- **Юрій ПАВЛОВСЬКИЙ**, доцент кафедри технологічної та професійної освіти Дрогобицького державного педагогічного університету імені Івана Франка, кандидат фізико-математичних наук, доцент.

Відповідальний за випуск **– Віктор БРИТАН**, доцент кафедри фізики та інформаційних систем Дрогобицького Державного педагогічного університету імені Івана Франка, кандидат фізико-математичних наук, доцент.

Роман Лешко, Ольга Лешко.
STEM-технології на основі платформи Arduino : рекомендації для виконання лабораторних робіт. Дрогобич : Дрогобицький державний педагогічний університет імені Івана Франка, 2024. 58 с.

Навчальний посібник «STEM-технології на основі платформи Arduino» складено відповідно до навчальних програм навчальної дисципліни «Інформаційні керуючі системи та STEM-технології» для підготовки фахівців магістратури за напрямами 014 Середня освіта (Фізика), 104 Фізика та астрономія та 122 Комп'ютерні науки, затверджених вченою радою Дрогобицького державного педагогічного університету імені Івана Франка.

У посібнику подані лабораторні роботи, які є основою для оволодіння інформаційно-керуючими системами та STEM-технологіями, зібрані основні методи та принципи роботи з платами Arduino. Посібник характеризується поєднанням теоретичних відомостей з практичними вказівками та завданнями до лабораторних робіт.
Бібліографія: 10 назв

CONTENTS

Зміст

PREFACE

Laboratory work in the discipline of "Information Control Systems and STEM Technologies" is a crucial part of the educational process, enabling students to gain practical skills in developing information systems and to become acquainted with modern technologies in the STEM field. One of the key technologies studied during these laboratory sessions is the Arduino platform.

Arduino is a globally recognized open-source platform for creating embedded systems that use microcontrollers. It provides the ability to create various interactive devices and systems that can be used across different fields of science and technology. Arduino offers students limitless opportunities for implementing innovative ideas and conducting research in the realm of information technology and automation.

Arduino is a compact microcontroller platform that has become very popular in the world of education, especially in STEM education (Science, Technology, Engineering, and Mathematics). It has several key advantages for studying information control systems and STEM technologies:

1. Ease of Use and Programming: Arduino features a simple and intuitive interface that allows students to learn the basics of programming quickly and effectively. Using Arduino enables students to grasp fundamental programming concepts, algorithms, and logic without delving into complex low-level aspects.

2. Low Cost and Accessibility: Arduino is known for its relatively low cost and accessibility, making it an ideal tool for teaching and use in educational institutions with limited budgets.

3. Large Community and Resources: Arduino boasts a large and active community that supports learning and knowledge sharing. There are numerous online resources, from tutorials to forums, that help solve problems and expand skills.

4. Wide Range of Additional Equipment: Arduino is compatible with a wide array of sensors, modules, and other hardware. This allows students to explore various aspects of STEM technologies and information control systems, including sensors, robotics, and automation.

5. Practical Application: Arduino offers the opportunity to create functional devices and projects. This facilitates a better understanding of theory and motivates further exploration.

6. Modularity and Flexibility: Arduino's modular design allows students to experiment with different components and configurations, providing flexibility in project design and implementation. This encourages creativity and experimentation in developing custom solutions.

7. Educational Resources and Support: Arduino is supported by a wealth of educational materials, including textbooks, online courses, and instructional videos. These resources are tailored to various learning levels and can be used to complement hands-on laboratory work.

8. Cross-disciplinary Learning: Arduino facilitates cross-disciplinary learning by integrating concepts from electronics, programming, mathematics, and engineering. This holistic approach helps students understand how different fields intersect and apply their knowledge in practical, real-world scenarios.

9. Scalability: Arduino projects can be scaled from simple introductory experiments to more complex and advanced applications. This scalability allows students to progressively challenge themselves and build upon their initial knowledge as they advance in their studies.

10. Encouragement of Collaborative Learning: Arduino projects often require teamwork and collaboration, which fosters the development of communication and problem-solving skills. Working on group projects helps students learn to share ideas, divide tasks, and work together effectively.

In this methodological guide, we will provide instructions and recommendations for performing laboratory work using the Arduino platform. You will learn how to connect components, program microcontrollers, create programs for reading data from sensors, manage data, and develop interactive projects. We hope that these laboratory exercises will help you acquire practical skills that will be valuable in your future career and research.

May this guide serve as your gateway to the world of Arduino, information control systems, and STEM technologies, where theoretical knowledge is combined with practical skills, and help you realize your ideas and projects in electronics and programming.

ПЕРЕДМОВА

Лабораторні роботи з дисципліни «Інформаційно-керуючі системи та STEM-технології» є важливою частиною навчального процесу, що дає змогу студентам здобувати практичні навички в розробці інформаційних систем та знайомитися з сучасними технологіями у сфері STEM. Однією з ключових технологій, які вивчаються під час лабораторних занять, є платформа Arduino.

Arduino – це всесвітньо визнана відкрита платформа для створення вбудованих систем з використанням мікроконтролерів. Вона надає можливість створювати різноманітні інтерактивні пристрої та системи, які можна використовувати в різних галузях науки і техніки. Arduino відкриває студентам безмежні можливості для реалізації інноваційних ідей і проведення досліджень у галузі інформаційних технологій та автоматизації.

Arduino є компактною платформою на основі мікроконтролера, яка здобула велику популярність у сфері освіти, особливо в STEM-освіті (наука, технології, інженерія та математика). Вона має кілька основних переваг для вивчення інформаційно-керуючих систем та STEM-технологій:

1. Простота використання та програмування: Arduino має простий та інтуїтивно зрозумілий інтерфейс, що дає змогу студентам швидко й ефективно освоювати основи програмування. Завдяки Arduino вони можуть зрозуміти основні концепції програмування, алгоритми та логіку без заглиблення у складні низькорівневі аспекти.

2. Низька вартість та доступність: Arduino відома відносно низькою вартістю та доступністю, що робить її ідеальним інструментом для навчання та використання в освітніх закладах з обмеженим бюджетом.

3. Велика спільнота та ресурси: Arduino має велику й активну спільноту, яка підтримує навчання та обмін знаннями. Існує багато онлайн-

ресурсів, від навчальних матеріалів до форумів, що допомагають вирішувати проблеми та розширювати навички.

4. Широкий спектр додаткового обладнання: Arduino сумісна з великою кількістю сенсорів, модулів та іншого обладнання. Це дає студентам можливість досліджувати різні аспекти STEM-технологій та інформаційно-керуючих систем, зокрема сенсори, робототехніку, автоматизацію.

5. Практичне застосування: Arduino надає можливість створювати функціональні пристрої та проєкти. Це сприяє кращому розумінню теорії та мотивує до подальшого вивчення.

6. Модульність і гнучкість: Модульний дизайн Arduino створює студентам можливість експериментувати з різними компонентами та конфігураціями, забезпечуючи гнучкість у розробці та реалізації проєктів. Це заохочує до творчості та експериментів у розробці унікальних рішень.

7. Освітні ресурси та підтримка: Arduino підтримується великою кількістю навчальних матеріалів, включаючи підручники, онлайн-курси та навчальні відео. Ці ресурси адаптовані до різних рівнів навчання і можуть використовуватися для доповнення лабораторної роботи.

8. Міждисциплінарне навчання: Arduino сприяє міждисциплінарному навчанню, інтегруючи поняття з електроніки, програмування, математики та інженерії. Такий комплексний підхід допомагає студентам зрозуміти, як перетинаються різні галузі, та застосовувати знання у практичних реальних ситуаціях.

9. Масштабованість: Проєкти на Arduino можна масштабувати від простих початкових експериментів до більш складних і передових застосувань. Така масштабованість дозволяє студентам поступово розширювати знання та будувати їх на основі попереднього досвіду.

10. Заохочення до спільного навчання: Проєкти на Arduino часто вимагають командної роботи та співпраці, що сприяє розвитку навичок

спілкування та усунення проблем. Робота над груповими проєктами допомагає студентам навчитися обмінюватися ідеями, розподіляти завдання та працювати разом ефективно.

У цьому методичному посібнику ми надамо інструкції та рекомендації щодо виконання лабораторних робіт з використанням платформи Arduino. Ви дізнаєтеся, як підключати компоненти, програмувати мікроконтролери, створювати програми для зчитування даних з сенсорів, управляти даними та розробляти інтерактивні проєкти. Сподіваємося, що ці лабораторні вправи допоможуть вам здобути практичні навички, які будуть цінними у вашій майбутній кар’єрі та наукових дослідженнях.

Нехай цей посібник стане для вас дверима у світ Arduino, інформаційно-керуючих систем та STEM-технологій, де теоретичні знання поєднуються з практичними навичками, а також допоможе реалізувати ваші ідеї та проєкти в електроніці й програмуванні.

Laboratory Work No. 1.
Arduino Board and Its Components

Objective

Study the components of the Arduino board and gain skills in connecting it to a personal computer. Learn to use the Arduino IDE environment.

Plan

1. Study of the Arduino board and its types.
2. Installation of Arduino IDE and port configuration.
3. Connecting the Arduino board to a personal computer.

Theoretical Information

Arduino is an open-source platform designed for the rapid and easy development of various electronic devices. Arduino can gather data about the surrounding environment through sensors and respond by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language and the Arduino development environment. Programming does not require a separate programmer, as the code is uploaded via a USB port. To get started, you only need the Arduino board and a computer with the Arduino IDE installed. The project, which originated in 2003, was named after a tavern near the IVREA Institute in Italy (Bar di Re Arduino or Bar del Rey Arduino). The project aimed to solve the problem of teaching students to quickly program electronic devices with microcontrollers without the need for soldering and assembling complex circuits.

Arduino was conceived by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis as part of a design course at the Interaction Design Institute Ivrea. The goal was to create a simple and affordable microcontroller platform that would make it easier for students and designers to

build interactive projects. The initial version of Arduino was based on the ATmega8 microcontroller and featured a minimalistic design to keep costs low.

The project's open-source nature was a key factor in its success, as it allowed a wide range of users to contribute to the development of both hardware and software. The Arduino team released the hardware designs and software under open-source licenses, enabling hobbyists and educators worldwide to use and modify them.

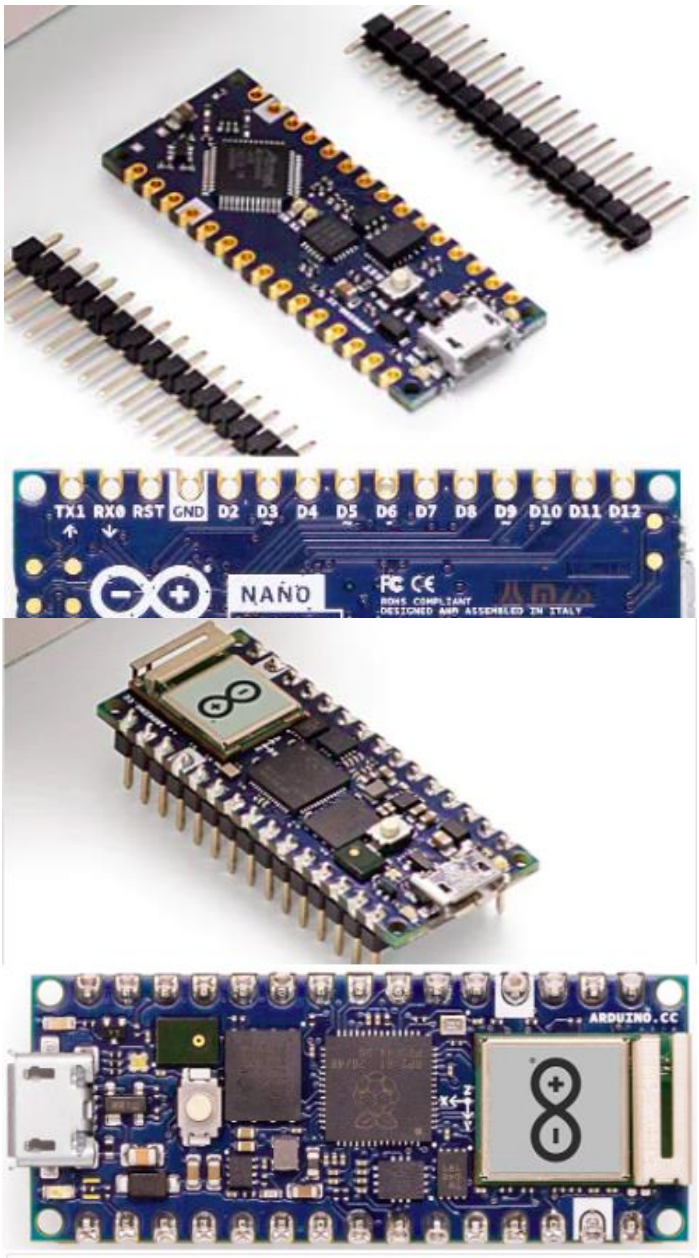
In 2005, the first official Arduino board, the Arduino Diecimila, was introduced. It featured improvements over the original prototype, including better USB connectivity and more memory. The board's name, "Arduino," was inspired by the local Italian tavern where the founders would often meet to discuss the project.

Over the years, Arduino has evolved significantly, with numerous versions and variants of the board being developed to cater to different needs and applications. The platform's success has led to a vibrant community of users and developers who continue to expand its capabilities and applications.

Today, Arduino is widely used in education, prototyping, and DIY electronics projects. It has played a crucial role in popularizing the maker movement and democratizing access to electronics and programming. The Arduino Foundation, founded in 2009, continues to support and develop the platform, ensuring its ongoing relevance and growth.

Today, various "families" of Arduino boards are known. Here are a few of them.

1. The **Nano** family consists of a range of compact boards with numerous features. It includes the affordable basic Nano Every and the multifunctional



Nano 33 BLE Sense / Nano RP2040 Connect, which come with Bluetooth® / Wi-Fi radio modules. These boards also feature a set of built-in sensors, such as temperature/humidity, pressure, gestures, microphone, and more. They can also be programmed using MicroPython and support machine learning.

2. The classic family includes boards such as the legendary Arduino UNO and other classic boards like the Leonardo and Micro. These boards are considered the foundation of the Arduino project and have been popular for many years (and continue to be in the future). The following discussion wil focus on these boards.

The Arduino microcontroller on the board is programmed using the Arduino programming language, which is based on C and C++, and its own development environment, which is available for free download. Arduino-based device projects can operate independently and interact with a computer.

To interact with Arduino, a USB cable and the Arduino IDE (Integrated Development Environment) are required. The appropriate software can be downloaded from https://www.arduino.cc/en/software for various platforms (Windows, Linux, Mac OS). After installation and the first launch, the program window appears as follows:

The menu, control buttons, program editor, and console, where some information is displayed. The console displays various types of information, including:

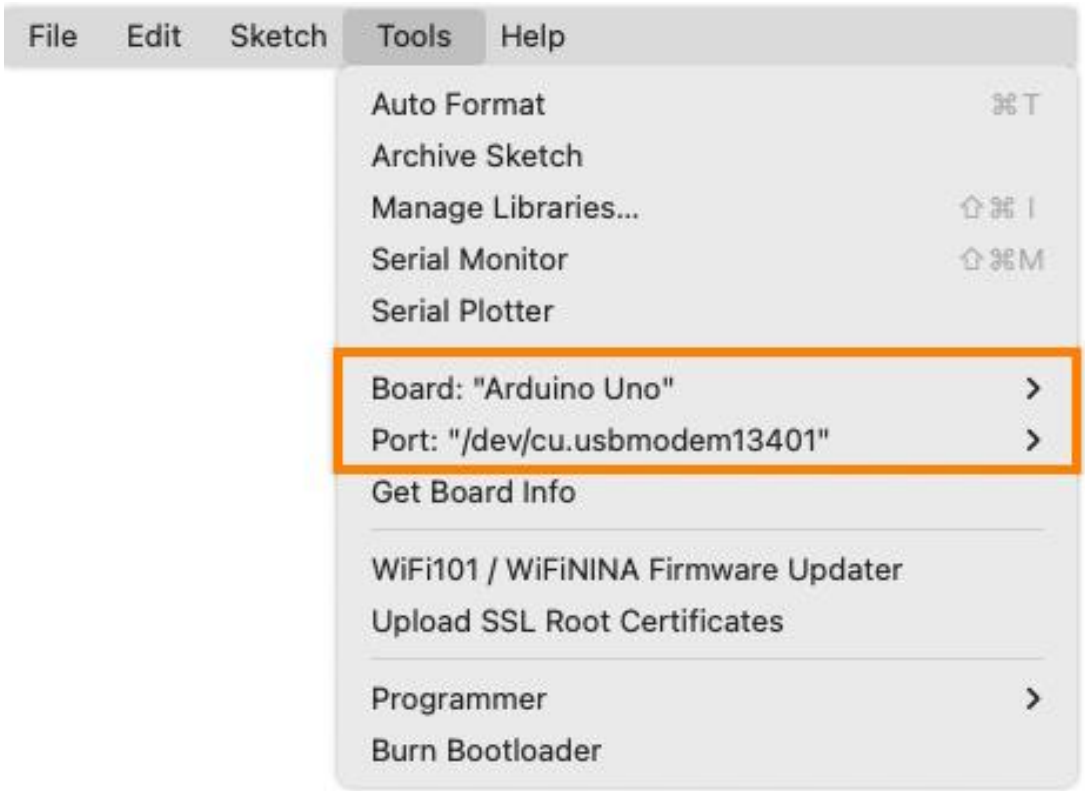
1. Compilation Messages: Information about the status of code compilation, including errors and warnings.

2. Upload Status: Messages indicating the progress of uploading the code to the Arduino board, including success or failure notifications.

3. Serial Monitor Output: Data sent from the Arduino board to the computer via the serial connection, useful for debugging and monitoring real-time data.

4. Program Execution Messages: Any print statements or debug messages included in the code that are output to the console for tracking program behavior.

After connecting the Arduino to the USB port, you need to specify the required port and select the board type in the software. For example, as shown in the image.

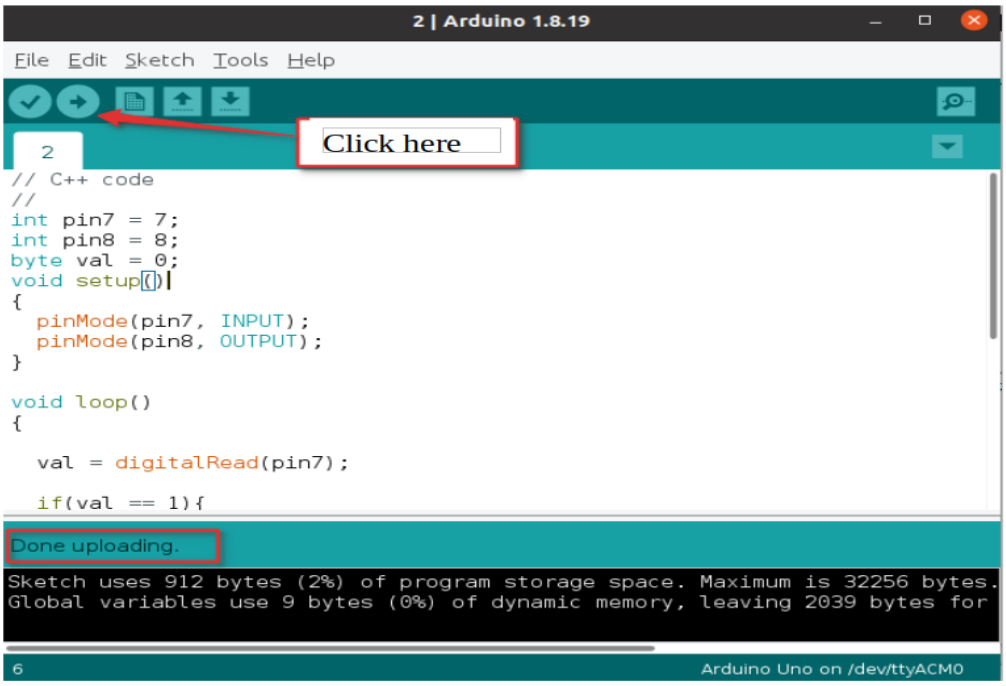


It is important to note that the port name may vary for each individual computer (it does not necessarily match the one shown in the image above).

After configuring the settings, you can check if the interaction is set up correctly. To do this, write a simple empty program (which does nothing) and upload it to the board. Programs for Arduino are called sketches. Once the sketch is uploaded, you can verify the connection by observing whether the board responds as expected, such as by displaying an indicator LED or similar signal. This process ensures that the Arduino board is correctly recognized by the computer and is ready for further programming and development.

```
// C++ code
void setup()
{
}
void loop()
{
}
```

The structure of the program is as follows: code placed in the `**setup**()` block is executed only once when the board is powered on; code added to the `**loop**()` block runs continuously while the board is powered. If everything is done correctly, after uploading the program, a message will be displayed.



If something is configured incorrectly, the console will display an error message along with a link to an online resource describing the issue and possible solutions.

Tasks

1. Download and install the Arduino IDE software.
2. Configure the board type and port.
3. Explore the menu and upload an empty sketch to the board.

Control Questions

1. What is Arduino?
2. What information can be obtained about Arduino?
3. What types and families of Arduino boards exist?
4. What are the typical boards belonging to the Arduino Nano family?
5. What are the typical boards belonging to the classic Arduino family?
6. What is needed to connect Arduino to a computer?
7. What is Arduino IDE and where can it be downloaded?
8. What is required to configure Arduino IDE?
9. For which platforms can Arduino IDE be installed?
10. What programming languages are used to write programs for Arduino?
11. What is a sketch?
12. What are the features of the `setup` and `loop` blocks in the program?
13. Can Arduino operate independently without a computer?
14. How can you verify that Arduino is correctly connected and recognized by the computer?
15. What types of messages are displayed in the Arduino IDE console?
16. What steps should be taken if an error occurs during the upload process to the Arduino board?

Laboratory Work No. 2.

Online platform Tinkercad

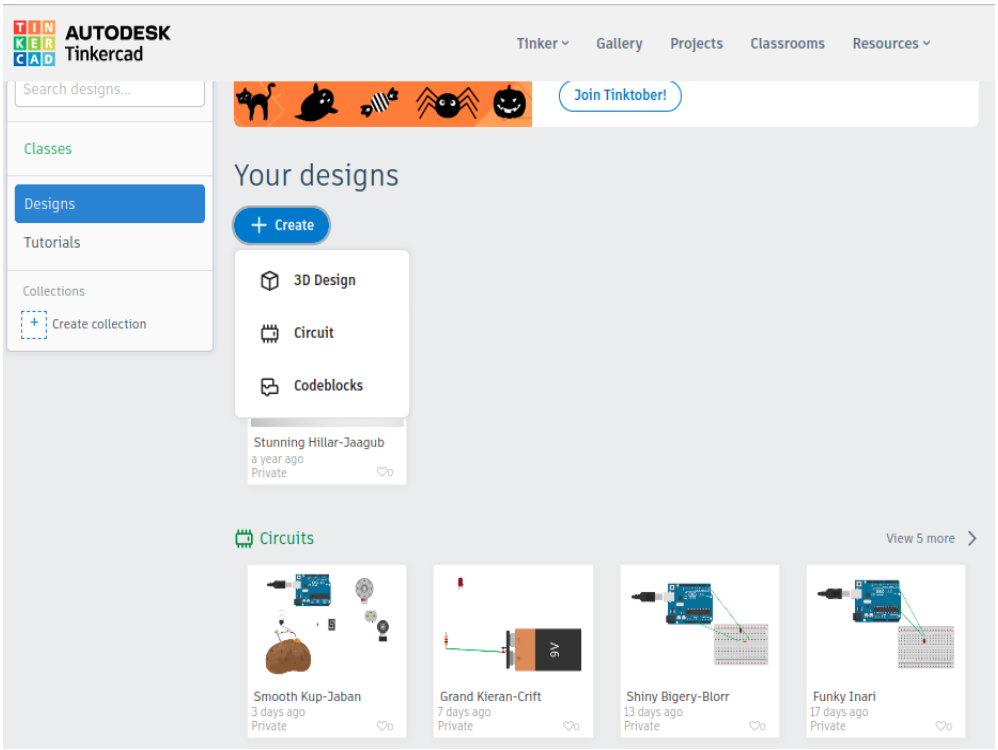
Objective

Learn the basics of working on the Tinkercad platform and simulate the operation of an Arduino controller and other electronic components.

Plan

1. Exploring the Tinkercad platform.
2. Creating accounts and projects.
3. Learning about the simulation features for Arduino on the Tinkercad platform.

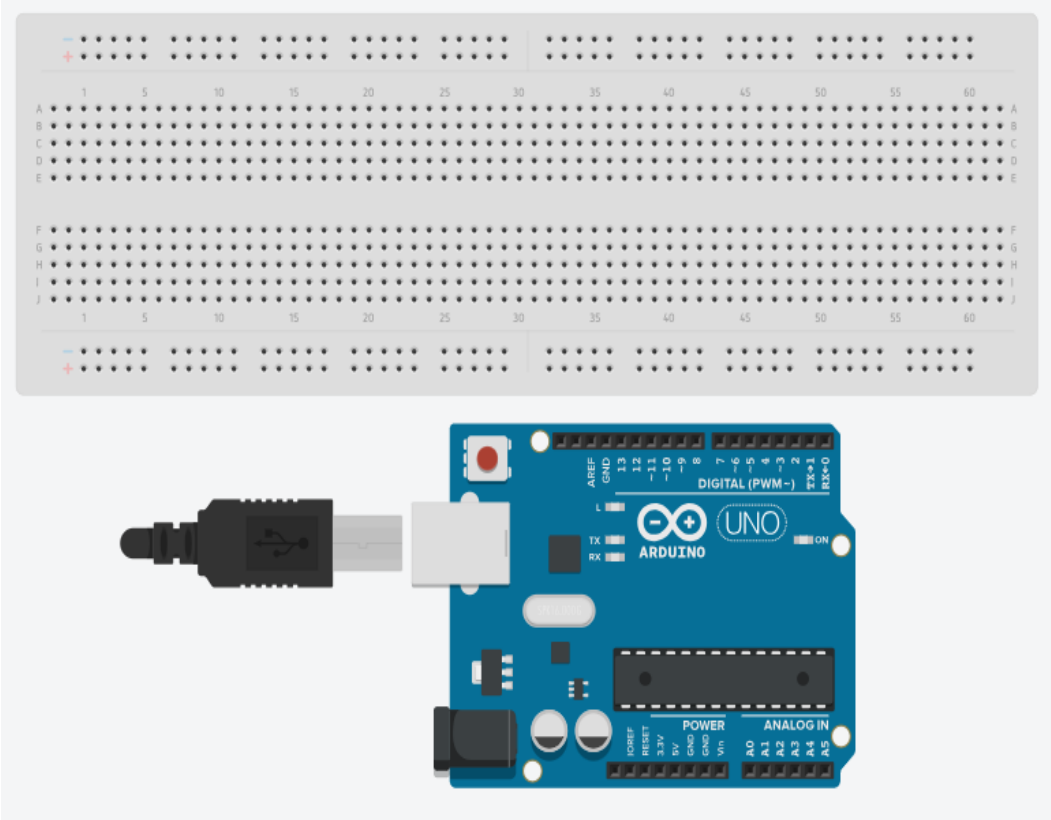
Theoretical Information



To simulate the operation of an Arduino controller, it is not necessary to have the actual Arduino board. It is sufficient to register on the Autodesk Tinkercad

web platform (https://www.tinkercad.com/). There are various registration options and methods, but each allows access to the online simulator.

After registering, you should select "**Designs**." In this section, you may find user-created circuits and also have the option to create a new circuit: **Create → Circuits**. After creating a new circuit, a window will open as shown in the image below.

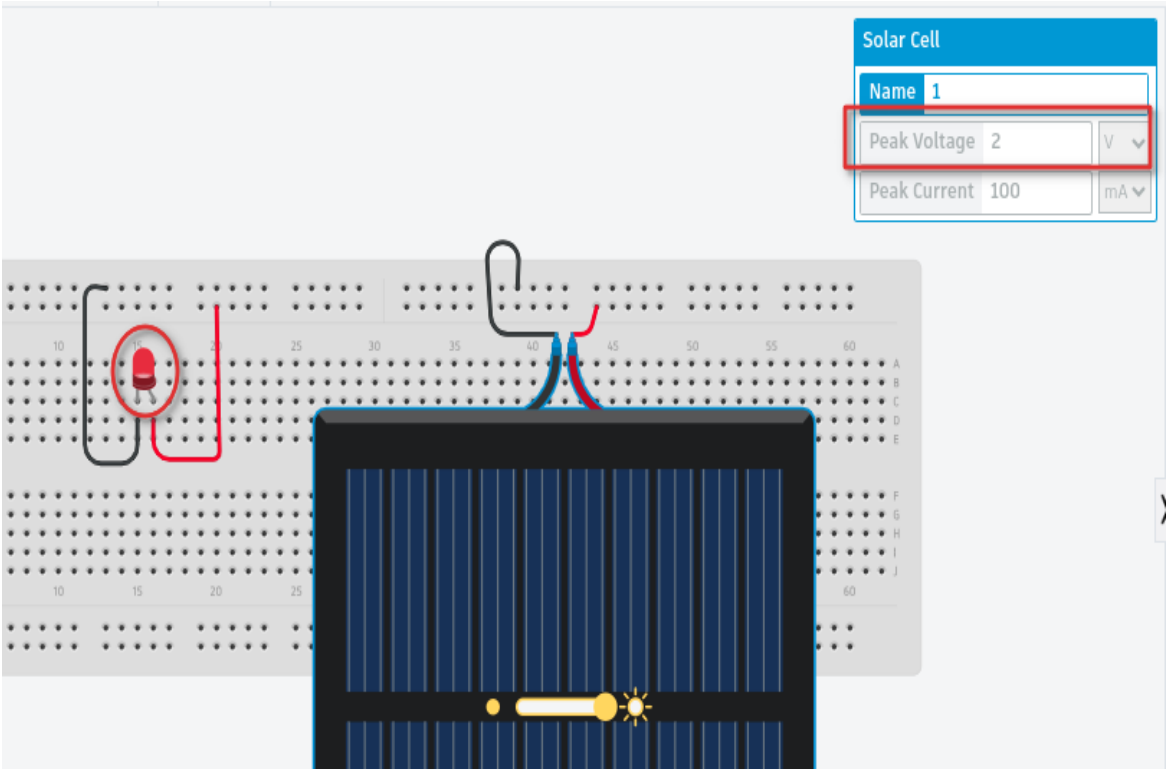


In the component list, you will find both the Arduino board and a breadboard, which we will use for assembling educational circuits and models. The breadboard consists of four areas. At the top and bottom, there are two rows of holes designed for creating power rails. All the holes in a row are connected to each other. We assume that a power supply voltage (positive and negative terminals) is connected to the holes in one row of these rails, and then components of our circuit can be connected to the power supply through the other holes.

The holes in each of the two vertical areas are vertically connected, organized in rows A-B-C-D-E and F-G-H-I-J. These areas are used for placing and connecting electronic components.

Let's add an LED and a 9-volt power source. Connect them as shown in the image and start the simulation.

The simulation indicated that the LED burned out because the voltage of 9V was too high for the LED. To troubleshoot the circuit, stop the simulation and either add a 450-ohm resistor or change the power source to a lower voltage of 2V. Each component has its own properties where parameters can be set. For example, the "solar battery" power source allows you to specify the maximum voltage (which we set to 2V). As seen, when the simulation is restarted, the LED operates and lights up.



Thus, the Autodesk Tinkercad platform allows you to test any circuits and their connections (including Arduino boards), simulate their

operation,experiment, and not worry about damaging real electronic components.

Tasks

1. Register on Autodesk Tinkercad.
2. Assemble the circuit shown above and check its operation.
3. Experiment with components and circuits on the simulator.

Control Questions

1. What is needed to simulate the operation of Arduino?
2. What is Autodesk Tinkercad?
3. How to register on Autodesk Tinkercad?
4. What areas does the breadboard have?
5. How are the holes on the breadboard connected?
6. How to start a simulation in Autodesk Tinkercad?
7. Why might an LED burn out if 9V is applied to it?
8. Is it possible to adjust component parameters in Autodesk Tinkercad?
9. Name the components available in Autodesk Tinkercad.
10. Where can you write sketches in Autodesk Tinkercad?
11. How to set the color of wires and connection type?
12. Where can you switch from real component images to schematic views in Autodesk Tinkercad?
13. How can you simulate different power sources in Autodesk Tinkercad?
14. What are the benefits of using Tinkercad for electronic circuit design and simulation?
15. How do you add and configure a resistor in Autodesk Tinkercad?
16. Can you export your Tinkercad designs for physical prototyping?

Laboratory Work No. 3.

Digital input-outputs of Arduino. Working with an LED

Objective

Study the possibilities of connecting LEDs to the Arduino controller. Setting up a program to control the LED.

Plan

1. Study of digital outputs of the Arduino board.

2. Investigation of LED characteristics and the electrical circuits for their activation.

2. Connecting an LED to Arduino and configuring its operation via digital ports.

3. Performing virtual simulations and real-world operation of the LED with the Arduino controller.

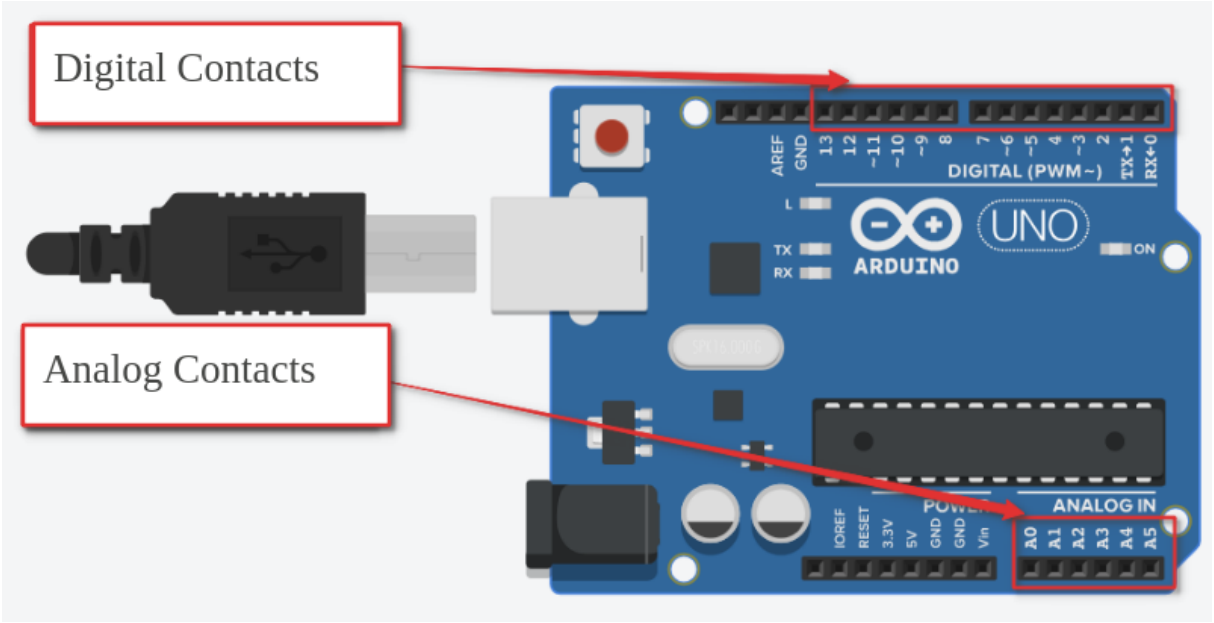
4. Connecting multiple LEDs.

Theoretical Information

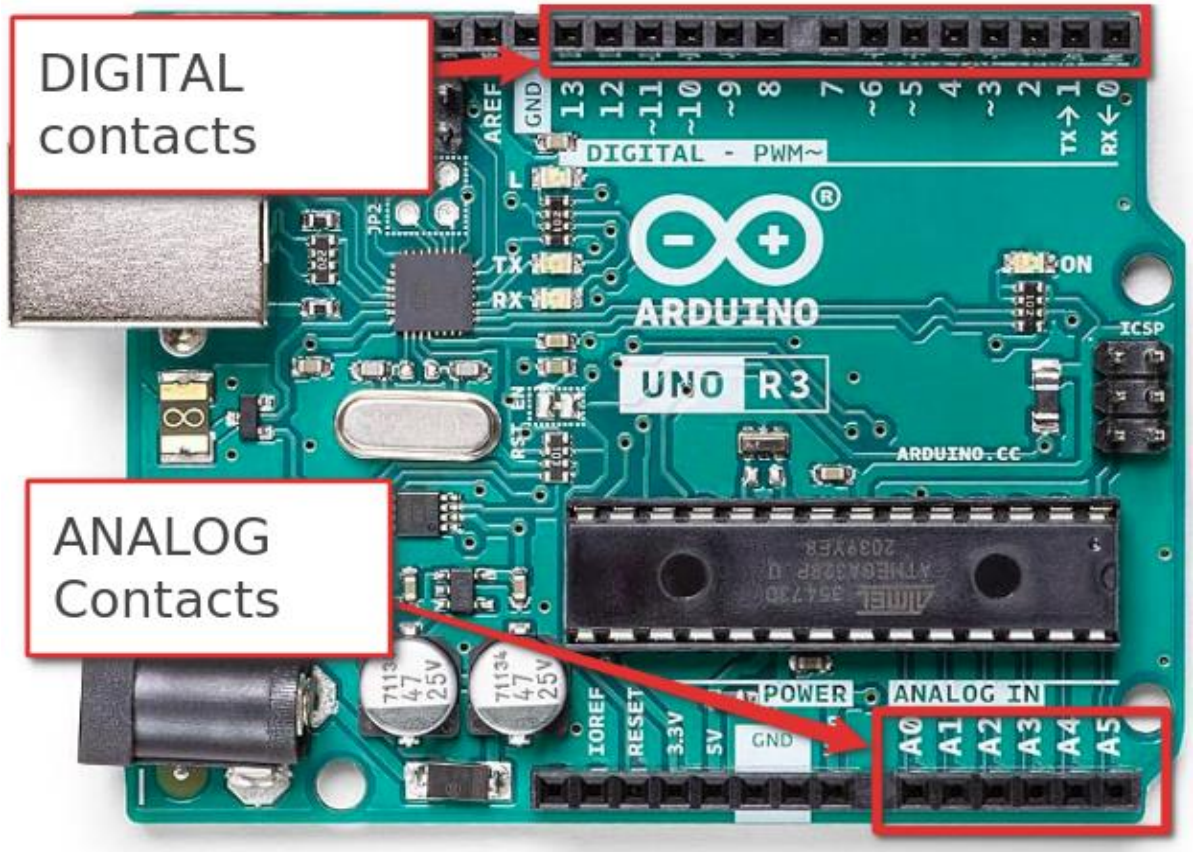
One of the simplest actions that the Arduino controller can perform is switching the states of digital input-output pins. Any of the digital input-output pins of the Arduino microcontroller can be programmatically switched between high level (+5V) and low level (0V). Such switching is used to control simple devices. The switching between levels is done using the command:

digitalWrite(_pin_number_, signal_level);

Remember that commands in C++ end with the symbol ";" (semicolon).



The signal parameter (**_signal_level_**) can take two values: **HIGH** and **LOW** (high and low). The **_pin_number_** is a number corresponding to the pin of the Arduino board.



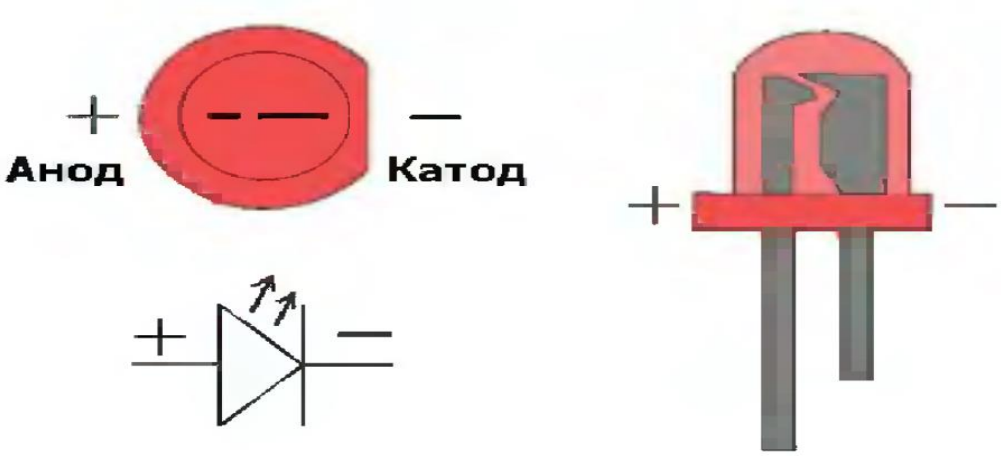
Note that the same digital pins can function as both input and output. Therefore, before using them, it is necessary to specify the role in which the corresponding pins will be used (input or output). This is done using the function:

pinMode(_pin_number_, _mode_);

where **_mode_** – has two possible value: **OUTPUT** or **INPUT**. **OUTPUT** sets the pin as an output for sending signals, while **INPUT** configures the pin to receive signals. Before using a pin as an input or output, it is crucial to initialize it correctly to ensure proper functionality. Incorrect pin configuration may lead to unexpected behavior or failure to control connected devices. Make sure to use the **pinMode**() function to set the pin mode before calling functions to read or write to the pin. So, to set a high signal level on pin 2, the following program can be executed:

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

Let's consider one of the simplest connections for actuators that can be controlled using a digital pin: connecting an LED. This is a semiconductor device that can emit light when an electric current flows through it in the forward direction (from the cathode to the anode). To correctly connect an LED in an electrical circuit, it is necessary to distinguish between the cathode and the anode. This can be done by three characteristics:

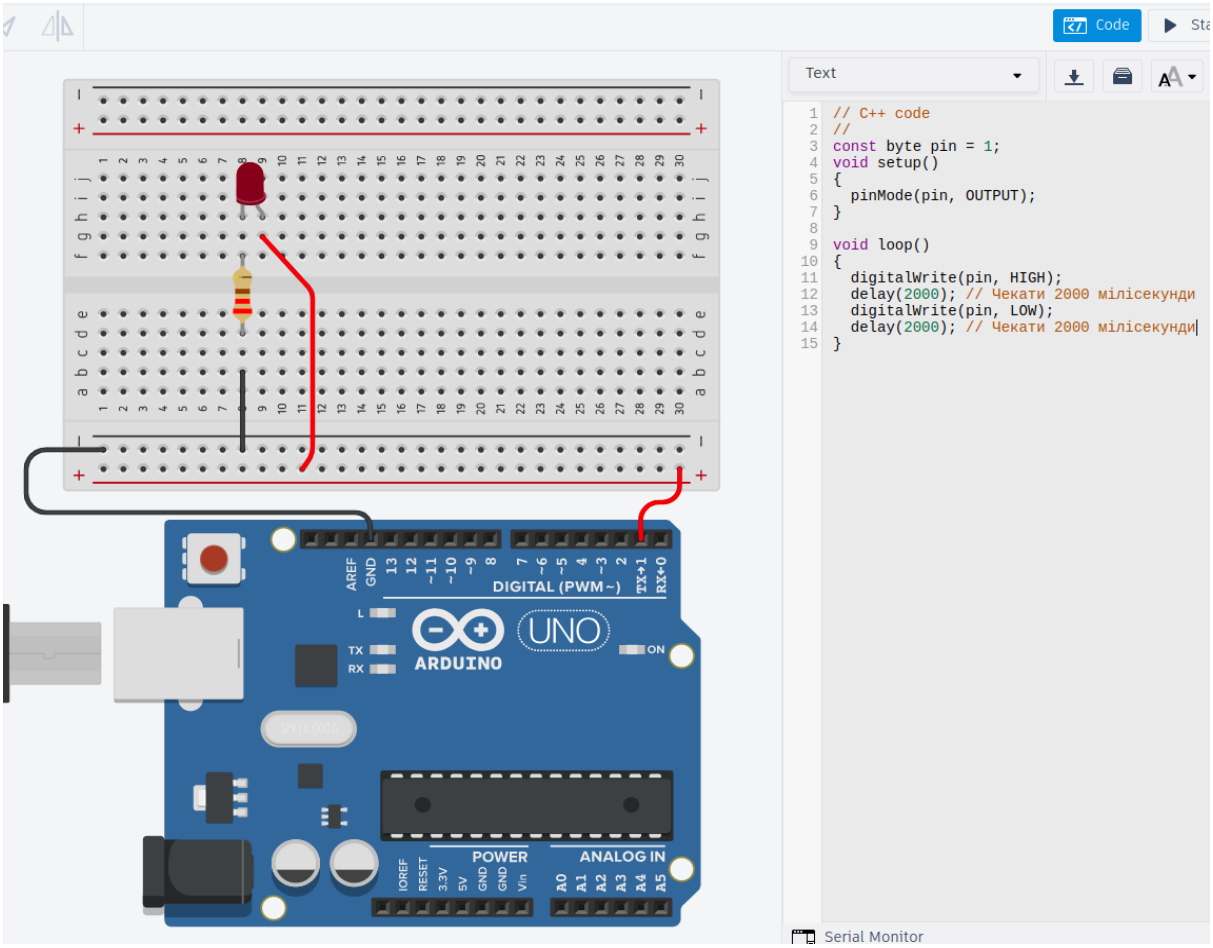


1) the anode has a longer lead.

2) on the side of the cathode is the LED's body.

3) the cathode inside the LED is wider.

Let's connect the LED to the Arduino board and "make" it blink with a 2-second interval. To do this, we will build the circuit shown in the diagram. The resistor value will be set to 220 Ω.



We will also enter the program into the appropriate window. After starting the simulation, the LED will blink. The same circuit can be built in reality, connect the Arduino board to the computer, and upload the program through the Arduino IDE environment.

Tasks

1. Implement the circuit described above.
2. Connect another blue LED and arrange for alternating blinking (blue-red), as if they were flashing warning lights.
3. Create a circuit where an LED fades in and out smoothly. Use PWM (Pulse Width Modulation) to gradually increase and decrease the brightness of the LED. Implement a program that controls this fading effect.
4. Design a circuit with three different colored LEDs (red, green, and blue) connected to the Arduino. Write a program to make the LEDs cycle through all possible colors by adjusting the brightness of each LED. For example, create a rainbow effect by varying the intensity of the red, green, and blue LEDs in sequence.

Control Questions

1. What are the inputs and outputs on the Arduino board?
2. What voltage can be applied to a digital pin?
3. How many digital pins are there on the Arduino UNO board?
4. In what modes can the pins on the Arduino board operate?
5. How do you set the operating mode of a pin on the Arduino board?
6. What is a Ground pin?
7. How many pins does an LED have?
8. How do you distinguish between the cathode and the anode?
9. What voltage sign should be applied to the cathode?
10. What voltage sign should be applied to the anode?

11. Which command tells the Arduino board to pause?
12. What is the function of a resistor in an LED circuit?
13. How do you use a `**digitalWrite()**` function in an Arduino program?
14. What is the purpose of the `**pinMode()**` function in Arduino programming?
15. How can you test if an LED is connected correctly in a circuit?
16. What are some common problems that can occur when working with LEDs and how can they be resolved?
17. How does PWM (Pulse Width Modulation) control the brightness of an LED?
18. What is the difference between `**HIGH**` and `**LOW**` in the context of Arduino digital pins?

Laboratory Work No. 4.
Connecting Buttons and Switches

Objective

Explore the possibilities of connecting buttons, switches, and buzzers to the Arduino controller. Configure a program to control an LED through switches.

Plan

1. Study the characteristics of switches and buttons and their activation.
2. Connect an LED, buttons, switches, and a buzzer to Arduino, and configure their operation.
3. Perform virtual simulations and real-world testing of the LED and buttons with the Arduino controller.

Theoretical Information

A switch is a device that allows you to close or open an electrical circuit. The state of the switch can change in various ways depending on the type of device. In this lab exercise, we will use a simple mechanical pushbutton switch, which closes its two contacts when pressed. For ease of mounting, these are often made with 4 contacts, where the outputs are paired together.

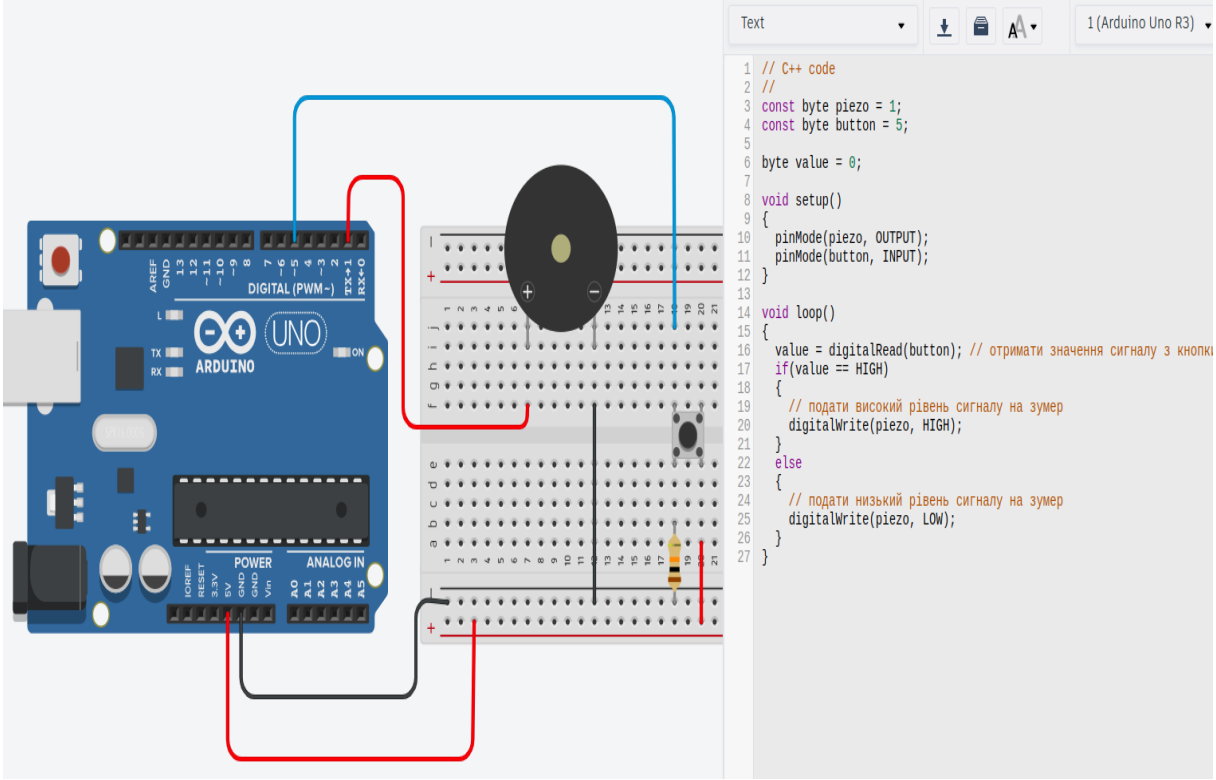
To read a signal from one of the Arduino pins, you need to use the function

res = digitalRead(_pin_number_);

After calling this function, the variable **res** will store the level of the digital signal detected on the corresponding pin.

To read the state of a button, it should be connected in such a way that pressing the button results in a high signal (+5V) at the input, and releasing it results in a low signal (0V). To achieve this, connect a button and a resistor in series between the power supply and ground (**GROUND**), and connect the microcontroller input between them. When the button is NOT pressed, the

microcontroller output will be connected through the resistor to ground and will have a low signal level. When the button is pressed, the input will be connected to the power supply and will have a high signal level. With this connection scheme, a current will flow through the resistor, depending on its resistance. To avoid high power consumption, use a resistor with a high resistance, such as 10 kΩ.



We will also connect a Piezo buzzer to the board (a device that produces a sound when a high voltage level is applied to it). We will configure its operation as follows: when the button is pressed, the buzzer will produce a sound, and when the button is released, there will be no sound. This way, you can generate an audible signal.

Tasks

1. Reproduce the circuit shown in the instructions on both the simulator and the real device.
2. Add an LED to the previous circuit. Configure it so that when the button is NOT pressed, the LED is lit and the buzzer is off, and when the button is pressed, the LED turns off and the buzzer produces a sound.
3. Connect a second button to the circuit and program the Arduino to toggle the LED on and off with each button press, while the buzzer remains off.
4. Add a resistor to limit the current through the LED, and create a program that makes the LED blink at a 1-second interval while the button is pressed, with the buzzer turned off.

Control Questions

1. How do you read the signal level from a digital pin on Arduino?
2. What is a pushbutton?
3. How are the contacts in a 4-pin button connected?
4. How do you connect a buzzer to the Arduino board?
5. What is the purpose of connecting a resistor in the circuit described in the instructions?
6. What do the constants HIGH and LOW represent?
7. How can you make the buzzer sound continuously for 10 seconds?
8. How can you "remember" in the program that the button has been pressed?
9. Can you adjust the voltage on the pins marked 5V and 3.3V?
10. How can you debounce a button in your Arduino program to avoid false triggering?
11. What is the purpose of using `digitalRead()` and `digitalWrite()` functions in Arduino programming?

12. How can you modify the LED blink rate in the Arduino program based on button presses?
13. What are the common troubleshooting steps if the buzzer does not sound as expected?
14. How does the internal pull-up resistor feature work in Arduino, and how can it be enabled?
15. What are the differences between analog and digital pins on the Arduino board?

Laboratory Work No. 5.
Seven-segment LED display

Objective

Study the seven-segment display. Learn how to configure the display and organize the output of information through it.

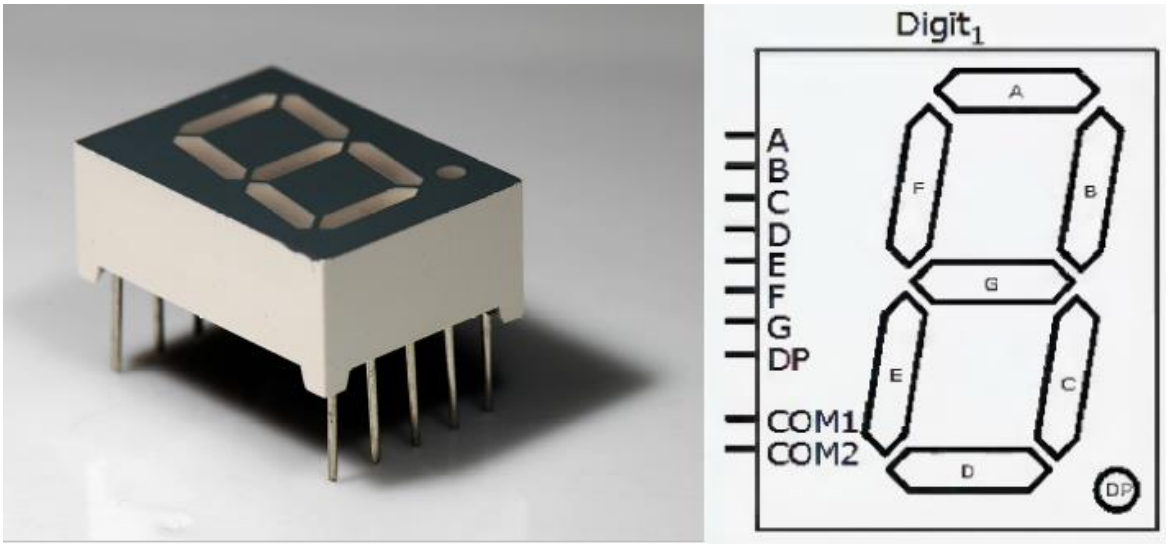
Plan

1. Study the characteristics of the seven-segment display.
2. Connect the seven-segment display to the Arduino and configure its operation.
3. Perform virtual simulations and real-world testing of the seven-segment display with the Arduino controller.
4. Set up a timer.

Theoretical Information

A regular LED can easily inform the user about binary events, that is, events where it is important to know whether they have occurred or not. For example, turning a device on or off, exceeding a threshold value, etc.

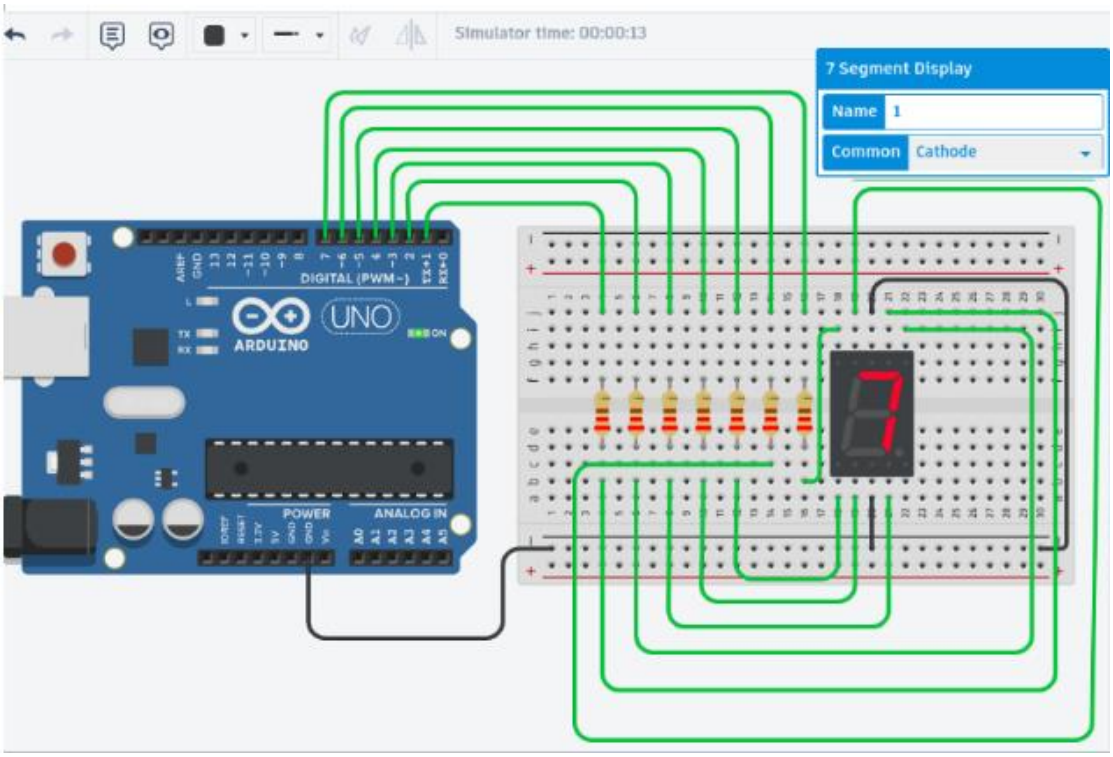
But what if you need to provide the user with more complex information, such as numerical data? For these purposes, electronics often use a seven-segment LED display. This device consists of a set of LEDs arranged in the shape of the number "8" so that by turning some of them on, you can create different figures.



According to their construction, seven-segment LEDs are divided into groups with a common anode and a common cathode. All segments are marked with Latin letters from "**A**" to "**G**." Usually, the numbering starts from the top segment and proceeds clockwise. The dot is marked separately with the word "dot." In addition to the contacts connected to the segments, the schematic also shows two contacts **COM** (for the common cathode or anode) and one **DP** (for the dot).

Connecting a seven-segment display is similar to connecting an LED. Since the display has 7 contacts (in addition to the dot and the common cathode or anode), you will need 7 digital outputs from the Arduino and 7 resistors of 220 ohms each.

We will provide a connection schematic and a program that will display the number "7" on the display. We will use a common cathode display. Below is the connection diagram and the program code.




```
// C++ code
//
const byte s_a = 1;
const byte s_b = 2;
const byte s_c = 3;
const byte s_d = 4;
const byte s_e = 5;
const byte s_f = 6;
const byte s_g = 7;

void setup()
{
  pinMode(s_a, OUTPUT);
  pinMode(s_b, OUTPUT);
  pinMode(s_c, OUTPUT);
  pinMode(s_d, OUTPUT);
  pinMode(s_e, OUTPUT);
  pinMode(s_f, OUTPUT);
  pinMode(s_g, OUTPUT);

  digitalWrite(s_a, HIGH);
  digitalWrite(s_b, HIGH);
  digitalWrite(s_c, HIGH);
}

void loop()
{

}
```

As seen from the figure, the number "7" is displayed on the indicator.

Tasks

1. Reproduce the schematic shown above.

2. Implement a counter that displays numbers from 0 to 9 with a 1-second interval.

3. Modify the counter to display numbers from 0 to 9 in ascending order every 0.5 seconds.

4. Create a program that makes the seven-segment display show a countdown from 9 to 0, with each number displayed for 2 seconds.

Control Questions

1. What is a seven-segment LED display composed of?
2. How are the contacts on a seven-segment display numbered?
3. How is the dot marked on the display?
4. How many contacts does a single-digit seven-segment LED display have?
5. What types of seven-segment LED displays are there?
6. How should a seven-segment LED display be connected to Arduino?
7. What is needed to display a digit on the indicator?
8. What type of single-digit seven-segment LED display is used in the lab work?
9. How will the connection schematic change if the type of single-digit seven-segment LED display is changed?
10. What is needed to implement a counter?
11. How do you write a program to display different digits on a seven-segment display using Arduino?
12. What is the purpose of the resistors connected to the segments of the seven-segment display?
13. How can you troubleshoot if the seven-segment display is not showing the correct digits?

14. What are the differences in wiring between a common cathode and a common anode seven-segment display?
15. How can you use the Arduino delay() function to control the timing of digit changes on the seven-segment display?

Laboratory Work No. 6.
Liquid Crystal Display (LCD)

Objective

Study the characteristics of the liquid crystal display (LCD) and its usage.

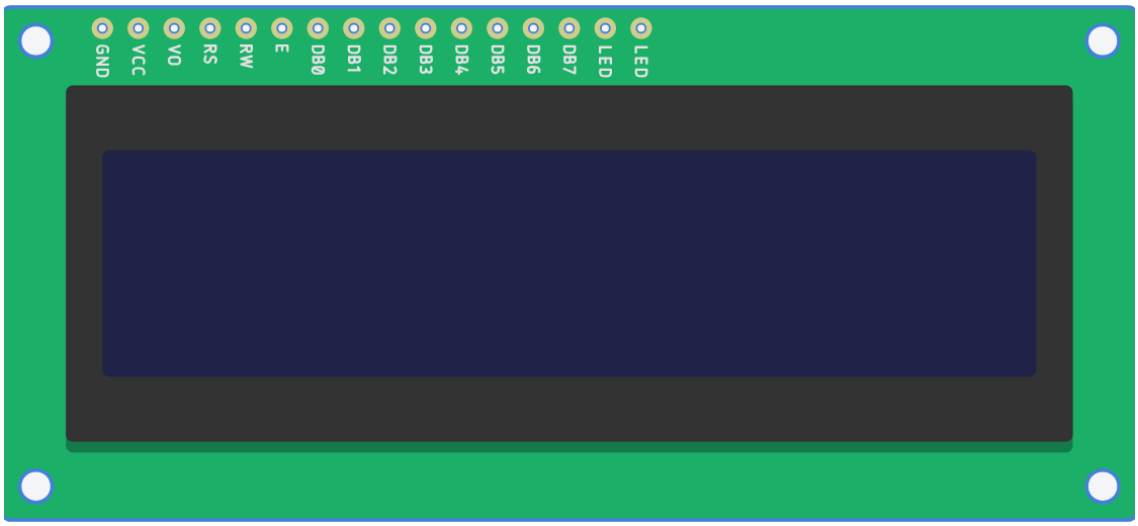
Plan

1. Study the characteristics of the liquid crystal display (LCD).
2. Connect the liquid crystal display to Arduino, configure its operation, and display data on it.
3. Perform virtual simulations and real-world testing of text output on the screen.

Theoretical Information

In the previous work, you were introduced to the seven-segment LED display. However, a more informative and simultaneously more complex type of indicator is the display. In this lab work, we will work with a liquid crystal display (LCD), which is found in many devices: electronic clocks, calculators, and screens of old (often button-based) phones. All of these devices use liquid crystal displays to some extent. LCDs are widely used due to their ability to display a wide range of characters and graphics. They offer a clear and versatile way to present information in various electronic devices.

We will be using a 16x2 display. This means it can display 2 lines of 16 characters each. The appearance and designation of the display are shown in the diagram below.



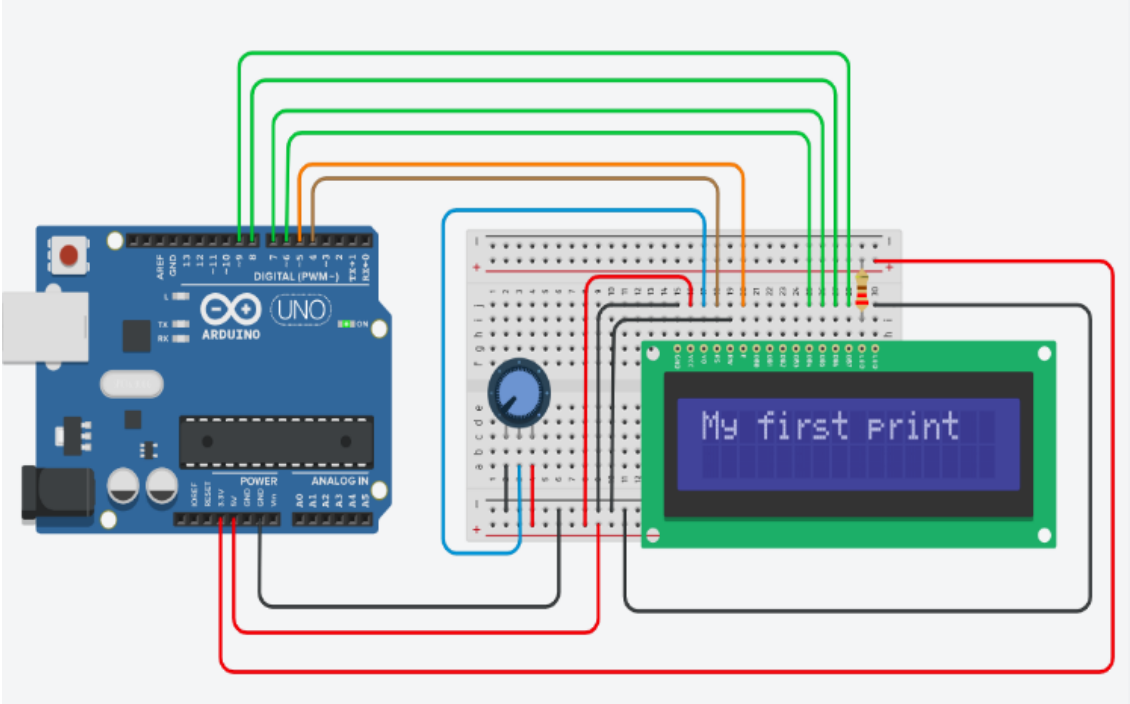
The indicators are labeled as follows:

- GND – ground (negative power supply)
- VCC – power supply (+5V)
- V0 – contrast (display contrast control)
- RS – register select
- RW – data direction (read or write)
- E – (enabled) synchronization
- DB0-DB7 – data bus
- Led- – backlight cathode
- Led+ – backlight anode

The connection diagram for the display is shown in the diagram below. In addition to connecting the information signals and power supply, it is necessary to adjust the display contrast. To do this, connect a 10 kΩ potentiometer to the circuit. If backlighting is also needed, it should be connected using the LED contacts through a resistor of at least 100 Ω.

To work with a liquid crystal display (LCD) in Arduino, there is a special library called LiquidCrystal. It is included with the directive **#include <LiquidCrystal.h>**. After that, you need to initialize the display by specifying which contacts are used for the Arduino.

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




```
#include <LiquidCrystal.h>

// C++ code
//

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

void setup()
{
  lcd.begin(16,2);
  lcd.print("My first print");
}

void loop()
{


}
```

Here, the first contact is **RS**, the second is Enabled, and the third through sixth are **DB4** to **DB7**. Other display contacts do not need to be connected to the Arduino. The library also includes a function to set the cursor position: **lcd.setCursor(0,1)** (column with index 0 and row with index 1). To clear the screen, use **lcd.clear()**.

The diagram above shows the schematic and program that displays the message "My first print" on the screen. It also includes backlighting connected through a 220 Ω resistor to a +3V power supply.

Tasks

1. Reproduce the schematic shown above.
2. Implement a scrolling text effect with any desired message.
3. Display the current time on the LCD using a real-time clock module (RTC).
4. Create a program that shows a static message on the LCD and then scrolls through a different message when a button is pressed.

Control Questions

1. What is the purpose of using a liquid crystal display (LCD)?
2. What examples of liquid crystal displays are you familiar with?
3. What type of display is used in the lab work?
4. How are the contacts labeled on a liquid crystal display?
5. Which contact regulates the display contrast?
6. Is it mandatory to connect the LED contacts? If so, what are they used for?
7. What library is used for working with a liquid crystal display?
8. How is the library for working with the liquid crystal display initialized?
9. Which directive is used to include the library for working with the liquid crystal display?
10. Which contacts are sufficient to connect to be able to display something on the screen?
11. How do you set the cursor position?
12. How do you clear the screen?
13. What command displays a message on the screen?
14. What concept should be applied when creating a scrolling text effect?
15. How can you adjust the backlight brightness of the LCD?
16. What are the differences between a common anode and a common cathode LCD?
17. How do you control multiple LCDs with one Arduino?

18. What function is used to print text on the LCD?
19. How do you display numerical values on the LCD?
20. How can you update the displayed text dynamically without clearing the screen?
21. How can you use the LCD to display characters other than standard alphanumeric characters, such as symbols or custom characters?
22. What is the purpose of the **lcd.begin**() function in the **LiquidCrystal** library?
23. How do you use custom characters on the LCD?
24. What is the purpose of the **lcd.print()** function, and how does it differ from **lcd.write()**?

Laboratory Work No. 7.

Sensors. Resistive Sensors

Objective

Study the features of sensor operation. Learn to process analog signals.

Plan

1. Study the characteristics of analog signals.
2. Connect a rheostat or potentiometer to Arduino, configure its operation, and read data from it.
3. Perform virtual simulation and real operation to display the voltage value from the rheostat.

Theoretical Information

In previous lab exercises, resistors were used to limit the current through an LED. In addition to fixed resistors, there are similar devices with variable resistance, known as potentiometers or rheostats. The image above shows pictures of rheostats and their symbols in circuit diagrams.

What is a potentiometer used for? Firstly, as described, it can be used to set a specific resistance value when a suitable fixed resistor is not available or when precise adjustment of an existing device is needed. Secondly, a potentiometer

can create a voltage divider – a device that can supply a load with only a portion of the voltage from the source. In this lab exercise, we will use a rheostat as a voltage divider, changing the value on one of the analog contacts of Arduino.

During work with buttons, we have already encountered the **digitalRead** function, which can read a digital signal from a given contact. There is also an analog version of this function, **analogRead**, which allows you to do the same for an analog signal:

res = analogRead(_pin_number_);

After calling this function, the variable `res` will store the value of the analog signal read from the corresponding contact. The `analogRead` function returns a number from 0 to 1023, where 0 represents 0V, and 1023 represents +5V. Intermediate voltage values correspond to intermediate numbers in the range of 1-1022. You should connect the potentiometer to contacts A0-A5.

Tasks

1. Connect the potentiometer to the Arduino (similarly to the previous work).
2. Write a program that will measure the voltage value from the potentiometer every 2 seconds and display it on the liquid crystal display.
3. Create a program to display a static message on the LCD indicating the current position of the potentiometer (e.g., "Potentiometer Value: X").
4. Modify the program to show the potentiometer value on the LCD in a bar graph format, where the length of the bar corresponds to the value read from the potentiometer.

Control Questions

1. What is a potentiometer?
2. What is the purpose of a potentiometer?
3. What is a voltage divider?
4. How does the analogRead function work?

5. What values can be read from analog contacts?
6. How are analog contacts labeled?
7. How many analog contacts are there on an Arduino UNO board?
8. What is the concept behind displaying voltage values on a display?
9. How can you connect a potentiometer to an Arduino?
10. What is the range of values that analogRead can return?
11. How can you calibrate a potentiometer in your Arduino code?
12. What is the purpose of using a resistor with a potentiometer in a circuit?
13. How do you calculate the voltage from the analogRead value?
14. What are the limitations of using analogRead for precise measurements?
15. How can you display the potentiometer's value in different units on an LCD?
16. How does changing the resistance of a potentiometer affect the voltage read by the Arduino?
17. What is the purpose of using a resistor in series with an LED?
18. How can you display the analog value from a potentiometer on an LCD?
19. What changes would you need to make to the program if you use a different analog pin?
20. How can you convert the raw analog value to a corresponding voltage?

Laboratory Work No. 8.

Analog Temperature Sensor

Objective

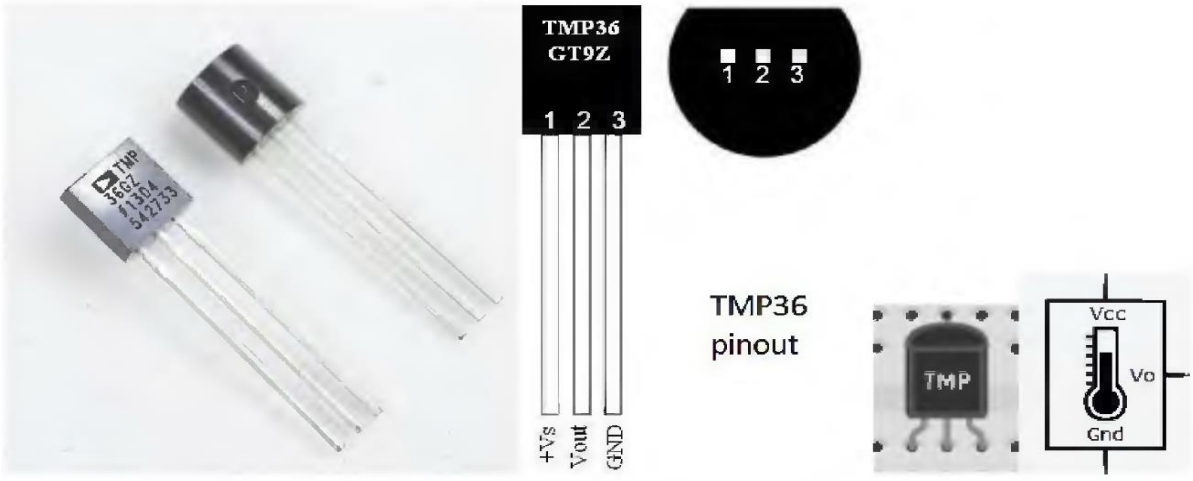
Study the features of working with a temperature sensor. Learn to process temperature data.

Plan

1. Study the temperature sensor and its parameters.
2. Calibrate the temperature sensor.
3. Conduct virtual simulation and real-world testing to display the temperature reading.

Theoretical Information

A common type of sensor with an analog output is the temperature sensor. The contact labels for such sensors are as follows:



Vs – power supply voltage from 2.7 to 5.5 V (for Arduino 5V);

Vout – output voltage that depends on the temperature;

Gnd – ground.

The output voltage of the sensor depends only on the temperature and is not affected by the supply voltage provided to the sensor.

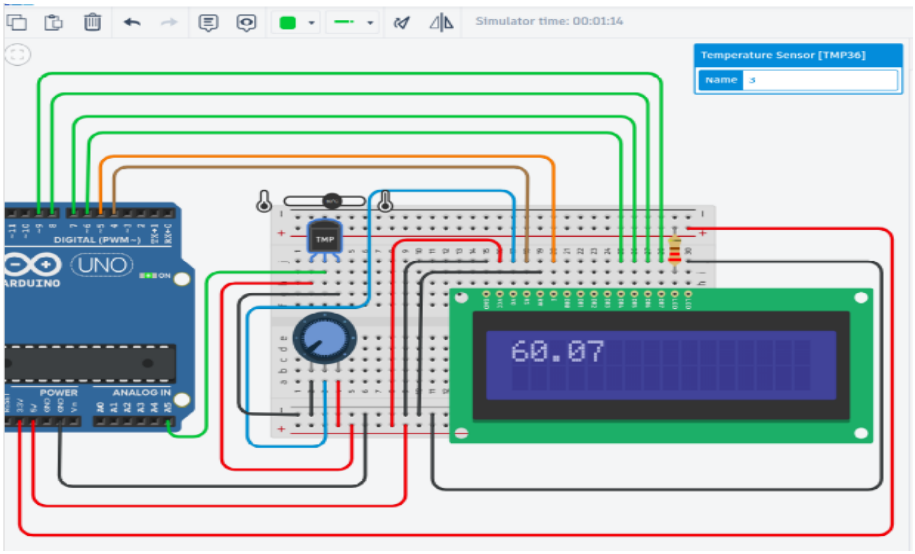
To determine the temperature, you need to find the maximum temperature value ($\mathbf{T_{max}}$) and the minimum temperature value ($\mathbf{T_{min}}$) that the sensor can measure. In other words, you need to calibrate the scale. For the minimum and maximum temperature values, there are minimum ($\mathbf{V_{min}}$) and maximum ($\mathbf{V_{max}}$) voltage values that are read by the Arduino contact. Then, considering the linear scale, you can establish the relationship between the voltage read by the Arduino contact and the temperature:

$$T = (V - V_{min})fracT_{max} - T_{min}V_{max} - V_{min} + T_{min}.$$

Here, **V** is the output voltage in volts, and **T** is the temperature in degrees Celsius. **V** is also the value obtained from calling the function **analogRead** on the Arduino.

V = analogRead(_pin_number_).

An example of connecting such a sensor is shown in the figure below.




```
#include <LiquidCrystal.h>

// C++ code
//
const byte pin_A = 5;
float res;
LiquidCrystal lcd(4,5,6,7,8,9);

float const V_min = 20;
float const V_max = 358;
float const T_min = -40;
float const T_max = 125;

void setup()
{
  lcd.begin(16,2);
  pinMode(pin_A, INPUT);
}

void loop()
{

  float V = analogRead(pin_A);
  float T = (V - V_min) * ((T_max - T_min) / (V_max - V_min)) + T_min;

  lcd.clear();

  lcd.print(T);
  delay(100);


}
```

In the example in the simulator, the maximum and minimum voltages are 20V and 385V, and the temperature range is [-40...125] °C.

Tasks

1. Recreate the schematic shown above.
2. Connect an LCD display to the circuit and output the temperature reading. Calibrate the scale.
3. Modify the program to include a temperature alert feature that triggers an LED if the temperature exceeds a certain threshold (e.g., 30°C).
4. Implement a program that logs temperature readings every minute and displays the average temperature on the LCD display.
5. Design a system where an LED or buzzer activates when the temperature exceeds a predefined threshold. Display the current temperature and the warning status on the LCD.

Control Questions

1. How does the temperature sensor work?
2. What type of sensor is the temperature sensor?
3. How many contacts does the temperature sensor have, and what are they?
4. What does the output voltage of the temperature sensor depend on?
5. What is the scale of the temperature sensor?
6. In what units is the signal from the temperature sensor measured?
7. How are voltage and temperature related in the temperature sensor?
8. How can the formula for the relationship between voltage and temperature in the temperature sensor be derived?
9. What supply voltage can be applied to the temperature sensor?
10. How can the temperature be adjusted in the simulator?
11. To which Arduino contact should the temperature sensor's Vout be connected?

12. What is the typical range of temperatures that a temperature sensor can measure?
13. How does the output voltage change with temperature in a typical temperature sensor?
14. How can you calibrate the temperature sensor to ensure accurate temperature readings?
15. What kind of analog signal does the temperature sensor output?
16. Why is it important to use a specific calibration formula for the temperature sensor?
17. How does the analogRead function help in measuring the temperature from the sensor?
18. What is the purpose of connecting a temperature sensor to the Arduino?
19. How does the sensor's output voltage correspond to the temperature range specified in its datasheet?
20. What should you do if the temperature readings from the sensor seem inaccurate or inconsistent?

Laboratory Work No. 9.

Light sensor

Objective

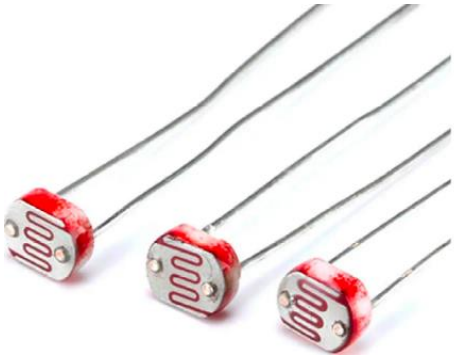
Learn the features of working with a light sensor. Learn how to process lighting data.

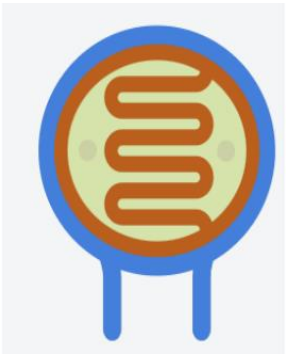
Plan

1. Study the light sensor (photoresistor) and its parameters.
2. Calibrate the light sensor.
3. Conduct virtual simulation and real-world operation of a device that automatically turns on/off the light based on external lighting conditions.

Theoretical Information

In this lab, we will examine another type of analog sensor – the light sensor, also known as a photoresistor.



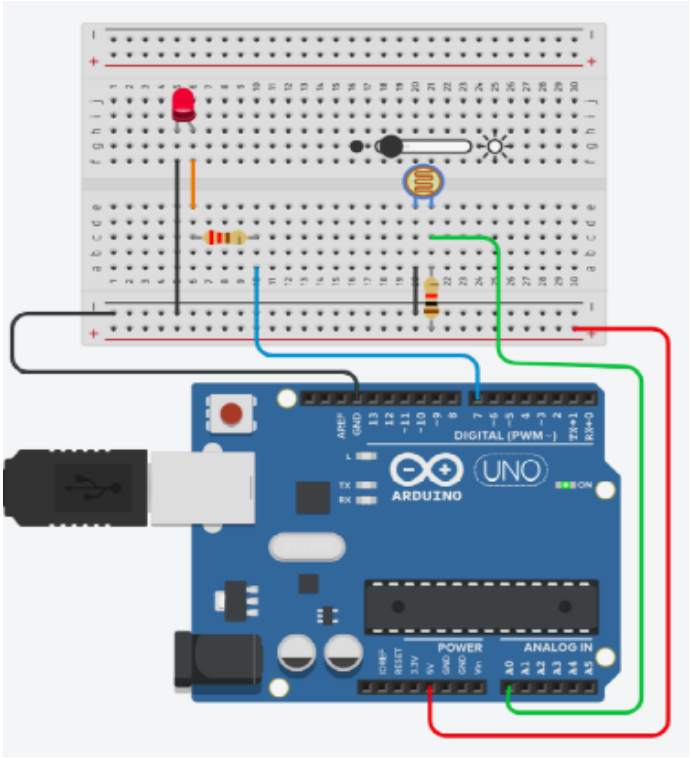


A photoresistor is a simple component with two contacts that changes its resistance depending on the level of light. Unlike a temperature sensor, a photoresistor does not provide a voltage but rather changes its resistance in response to varying light levels.

To read the values from a photoresistor, you first need to convert the change in resistance into a change in voltage. This is done using a "voltage divider" circuit. Connect one contact of the photoresistor to the supply voltage, and connect the other contact to a standard resistor, which is then connected to ground. The voltage across the resistors will be proportional to their resistances. For example, if the resistance of the photoresistor is 20 kΩ and the standard resistor is 10 kΩ, the voltage across the photoresistor will be twice that across the standard resistor. With a supply voltage of 5V, the resulting voltages would be 3.333V and 1.666V, respectively. As the resistance of the photoresistor changes with light levels, the voltages will also change. By connecting the midpoint between the resistors to an analog input on Arduino, you can measure the voltage across the standard resistor and infer the light intensity. The output voltage from the voltage divider circuit will vary with the amount of light falling on the photoresistor, allowing you to gauge the light intensity. By reading the analog voltage with Arduino, you can then display or use this data to control other devices, such as turning on lights when it gets dark. Calibration may be required to map the raw voltage readings to meaningful light intensity values.

Photodiodes included with Arduino boards typically have a resistance that varies from 200 kΩ (in complete darkness) to 1 kΩ (at a brightness of 10 lux). When experimenting with the sensor, you'll notice that the analog readings don't change linearly. This is a key difference from temperature sensors, which have a linear relationship.

We'll create a circuit with a resistor, a photoresistor, and an LED. We'll set it up so that the LED turns on at minimal lighting conditions.




```
// C++ code
//
const int digit_i = 7;
const int analog_r = A0;

const float rate = 500;
void setup()
{
  Serial.begin(9600);
  pinMode(digit_i, OUTPUT);
  pinMode(analog_r, INPUT);

}

void loop()
{


  float light = analogRead(analog_r);
  delay(10);

  if(light > rate){
    // вимк світло
    digitalWrite(digit_i, HIGH);
  }else{
    digitalWrite(digit_i, LOW);
  }

}
```

rate level

Tasks

1. Reproduce the circuit shown above.

2. Connect an LED to the analog pin and adjust its brightness based on the voltage from the photoresistor.

3. Test the light sensor by varying the light intensity and record the changes in the analog reading.

4. Create a program that turns on the LED when the light level falls below a certain threshold and turns it off when the light level exceeds that threshold.

Control Questions

1. What is a photoresistor?
2. To which type of signals (analog or digital) does a photoresistor belong?
3. How many contacts does a photoresistor have?
4. How should the contacts of a photoresistor be connected? Does it matter which is “+” and which is “-”?
5. What does a photoresistor change in response to changes in lighting?
6. How is a "voltage divider" connection implemented?
7. When is the resistance of a photoresistor higher: in complete darkness or in bright light?
8. Is the resistance of a photoresistor linear or nonlinear?
9. How can you adjust the lighting falling on the photoresistor in a simulator?
10. Is an LED different from a photoresistor?
11. To which Arduino pin should the photoresistor be connected?
12. What is the “rate” constant mentioned in the program?
13. How does changing the value of the fixed resistor in the voltage divider affect the readings from the photoresistor?
14. What is the typical range of resistance values for a photoresistor in low and high light conditions?
15. How does the light intensity affect the voltage output in the voltage divider circuit?
16. Can a photoresistor be used for detecting very small changes in light levels? Why or why not?
17. How can you calibrate a photoresistor for specific lighting conditions?

18. What are some practical applications of photoresistors in electronic projects?
19. How can you implement a threshold value in a program to trigger an action based on light levels detected by a photoresistor?
20. What are the limitations of using a photoresistor in your projects, and how can they be mitigated?

Referenses

1. Arduino.ua [Internet resource]. Access: https://arduino.ua/ .
2. Arduino blog [Internet resource]. Access: https://blog.arduino.cc/.
3. Arduino-DIY [Internet resource]. Access: http://arduinodiy.com/.
4. Margolis M. Arduino Cookbook. O'Reilly Media, 2011. 662 p.
5. Perea F. Arduino Essentials, 2015. 206 p.
6. Tinkercad [Internet resource]. Access: https://www.tinkercad.com/.
7. Microsoft C++. [Internet resource]. Access: https://learn.microsoft.com/uk-ua/cpp/?view=msvc-170.
8. Arduino Documentation. [Internet resource]. Access: https://www.arduino.cc/en/Tutorial/HomePage.
9. Web-site habr.com [Internet resource]. Access: https://habr.com/search/?q=Arduino#h.
10. John Nussey. Arduino for Dummies : John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774

Електронне навчально-методичне видання

Роман ЛЕШКО, Ольга ЛЕШКО
Roman LESHKO, Olha LESHKO

STEM-ТЕХНОЛОГІЇ НА ОСНОВІ ПЛАТФОРМИ ARDUINO

STEM-TECHNOLOGIES BASED ON THE ARDUINO BOARD

Дрогобицький державний педагогічний університет
імені Івана Франка

Редактор
Ірина Невмержицька
Технічний редактор
ІринаАртимко

Здано до набору 07.11.2024 р. Формат 60х90/16. Гарнітура Times. Ум. друк. арк. 3.37. Зам. 108.

Дрогобицький державний педагогічний університет імені Івана Франка. (Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників та розповсюджувачів видавничої продукції ДК № 5140 від 01.07.2016 р.). 82100, Дрогобич, вул. Івана Франка, 24, к. 203