

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
National University "Zaporizhzhia Polytechnic"

LABORATORY GUIDELINES

in the discipline

"Integrated Computer Systems"

for students of specialty 123 "Computer Engineering"
according to the educational program "Specialized Computer Systems"

for all forms of education

2023

Laboratory guidelines in the discipline "Integrated Computer Systems" for students of specialty 123 "Computer Engineering" according to the educational program "Specialized Computer Systems" for all forms of education/ Compl.: M. Tiahunova – Zaporizhzhia: NU "Zaporizhzhia Polytechnic", 2023. – 31 p.

Compiler: M. Tiahunova, Ph.D., Associate Professor

Reviewer: H. Kyrychek, Ph.D., Associate Professor

Responsible
for issue: M. Tiahunova, Ph.D., Associate Professor

Approved at the meeting of the
Computer Systems and Networks
Department
(protocol № 5 of January 30, 2023)

Recommended for publication by the
Educational and Methodological
Commission of Computer Sciences and
Technologies Faculty
(protocol № 6 of January 31, 2023)

CONTENT

1 LABORATORY WORK №1. Working with the Arduino Uno on the TinkerCAD platform.....	4
1.1 General information.....	4
1.1.1 General information about cyber-physical systems	4
1.1.2 General information about Arduino boards.....	5
1.1.3 Arduino analogue boards.....	6
1.1.4 Features of the Arduino Uno board	13
1.1.5 Web-platform TinkerCAD	16
1.2 Progress	16
1.3 Tasks to complete.....	20
1.4 Contents.....	20
1.5 Review questions.....	20
2 LABORATORY WORK №2. Modeling of traffic light on the TinkerCAD platform	21
2.1 Progress	21
2.2 Tasks to complete.....	30
2.3 Contents.....	30
LITERATURE.....	31

1 LABORATORY WORK №1

Working with the Arduino Uno on the TinkerCAD platform

The purpose of the work is to get acquainted with the Arduino Uno board and work in web environments and TinkerCAD.

1.1 General information

1.1.1 General information about cyber-physical systems

A cyber-physical system (CPS) is a mechanism controlled or monitored by computer algorithms and closely linked to the Internet and its users. In cyber-physical systems, software is closely related to physical objects. CPS components interact at different temporal and spatial levels and may have different, different behaviors and interact with each other in different ways, which may vary depending on Context. Examples of cyber-physical systems can be considered smart power systems, unmanned car systems, automated control systems, robotic systems, self-driving aircraft.

The sources of development of cyber-physical systems have become different in their specificity spheres of human activity: aerospace, automotive, public infrastructure, energy, healthcare, transport, entertainment, household appliances, etc.

Unlike existing embedded systems, complete CPS systems are developed as a network of several components that process raw materials. The main idea of cyber-physical systems is closely related to robots and sensor systems, which are equipped with "smart" mechanisms with sufficient computing power for sufficient control. Thanks to constant scientific progress, CPS undergoes periodic changes that improve the connection between physical and computational components through "smart" mechanisms, thus improving adaptability, increasing autonomy, efficiency, reliability and security, and expanding the functionality of cyber-physical systems. This expands their potential into the following dimensions of human activity: driving intervention (e.g. collision avoidance), accuracy in operations (e.g. robotic surgery), rescue or research operations (e.g. robotics), transport coordination (e.g. air traffic).

CPS is used as sensor-equipped autonomous systems that have the ability to communicate with the operator. For example, a large number of networks with wireless sensors monitor a certain parameter, and the processed information is transmitted to a central node. Other types of cyber-physical systems may include smart grids, autonomous manufacturing, health monitoring, process control systems, robotics, and self-driving air transport.

One of the options for implementing the core of the cyber-physical system is to use the Arduino board. Consider it in more detail in this laboratory work.

1.1.2 General information about Arduino boards

Arduino is a small board with its own processor and memory. The board also has a couple of dozen contacts to which you can connect all sorts of components: light bulbs, sensors, motors, kettles, routers, magnetic door locks and in general everything that runs on electricity.

With Arduino, you can download a program that will control all these devices according to a given algorithm.

Arduino is a hardware computing platform, the main components of which are an I / O board and a development environment.

Arduino can be used both to create standalone interactive objects and connect to software running on a computer.

Thanks to the openness of the Arduino, its device is known and allows for free modification. Therefore, any manufacturer of boards can produce an analogue of the Arduino board, make changes to the board itself.

Program control is implemented in C ++ simplified version, often also called Wiring. The development of a control program can be carried out through any convenient C/ C ++ toolkit. The platform works with Linux, Windows and MacOS X operating systems.

Thus, using the Arduino board, you can create an infinite number of unique cool gadgets made with your own hands and according to your own idea.

Arduino is a hardware computing platform, the main components of which are a simple I / O board and software (a special shell for programming the board) or IDE (Integrated Development Environment). The board also has a couple of dozen contacts to which you can connect all sorts of

components: light bulbs, sensors, motors, kettles, routers, magnetic door locks and in general everything that is powered by electricity.

In the Arduino processor, you can download a program that will control all these devices according to a given algorithm. Thus, you can create an infinite number of unique cool gadgets made by hand and according to your own plan.

The software runs on a personal computer and allows you to write the code that you have developed to the board. Arduino



Figure 1.1 - Arduino and matchbox

Thanks to the openness of the Arduino, its device is known and allows for free modification. Therefore, any manufacturer of boards can produce an analogue of the Arduino board, make changes to the board itself, not to mention the free configuration of sets (kits).

1.1.3 Arduino analogue boards

Since Arduino is a brand of the manufacturer, analogues have other, but usually consonant with Arduino names - Freduino, Freeduino, DCcduino, Xduino, Funduino, Robotale and many others.

Moreover, the brand may not be indicated at all, but there will be an inscription like for Arduino.



Figure 1.2 – Board DCcduino UNO

The original Arduino is made in Italy, most of the analogues - in China. Visually, most of these boards look the same (there is a corporate logo on the Arduino) and even have the same blue color (there are exceptions, but not much, Funduino, for example, red). Also, Arduino analogue boards usually have the same prefixes in their name as the Arduino itself, for example, DCcduino UNO corresponds to Arduino UNO.

There are several versions of Arduino platforms. The latest version of Leonardo is based on the ATmega32u4 microcontroller. Uno, like the previous version of Duemilanove created on the microcontroller Atmel ATmega328. Older versions of the Diecimila platform and the first working Duemilanoves were based on the Atmel ATmega168, earlier versions used ATmega8. The Arduino Mega2560, in turn, is built on the ATmega2560 microcontroller.

The following are the main versions of Arduino boards.

The Arduino Nano is a full-featured miniature device based on the ATmega328 (Arduino Nano 3.0) or ATmega168 (Arduino Nano 2.x) microcontroller, adapted for use with layout boards. In terms of functionality, the device is similar to the Arduino Duemilanove, and differs from it in size, the absence of a power connector, as well as another type of (Mini-B) USB cable. Arduino Nano is designed and manufactured by Gravitech.



Figure 1.3 – Arduino Nano board

The Arduino Pro Mini is a device based on the ATmega328 microcontroller. It includes: 14 digital inputs / outputs (of which 6 can be used as PWM outputs), 8 analog inputs, a quartz resonator, a reset button and contact pads for soldering connectors.

The six-pin connector can be used to power and interact with the board via USB using an FTDI adapter or a Sparkfun layout board.

Arduino Pro Mini is designed for semi-stationary installation in various equipment or installations. The board is specially supplied without soldered connectors, which allows the user to solder the wires or use the

required types of connectors at their discretion. By location you water the Arduino Mini Pro compatible Arduino Mini.

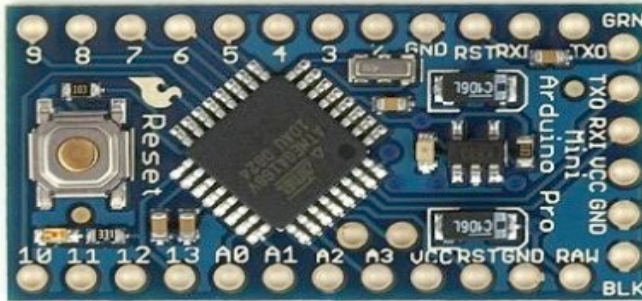


Figure 1.4 – Board Arduino Pro Mini

Arduino Leonardo - it includes everything you need to work with a microcontroller: 20 digital inputs / outputs (7 of which can work as PWM outputs, 12 - as analog inputs), quartz resonator at 16 MHz, micro-USB connector, power connector, connector for in-circuit programming ICSP (In-Circuit Serial Programming) and a reset button. To start working with Leonardo, simply supply power from an AC / DC adapter or battery, or connect it to a computer using a USB cable.

The difference between Leonardo and all previous boards is that its USB controller is built directly into the ATmega32U4 microcontroller, which eliminates the need for an additional processor. Thanks to this, when connected to a computer, Leonardo can be defined not only as a virtual (CDC) COM port, but also as a regular mouse or keyboard. In addition, this architecture has an impact on the behavior of the board.

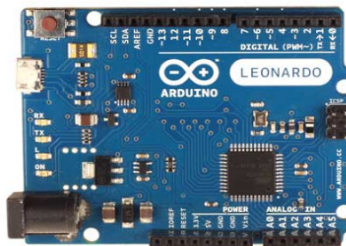


Figure 1.5 – Board Arduino Leonardo

Arduino Mega 2560 - it includes everything you need for convenient work with a microcontroller: 54 digital inputs / outputs (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UART (hardware receivers for the implementation of serial interfaces), a 16 MHz quartz resonator, a USB connector, a power connector, an ICSP connector for in-circuit programming and a reset button. To start working with the device, simply supply power from an AC / DC adapter or battery, or connect it to a computer using a USB cable. The Arduino Mega is compatible with most expansion boards designed for the Arduino Duemilanove and Diecimila.

The Arduino Mega 2560 differs from all previous boards in that it uses an ATmega16U2 microcontroller (ATmega8U2 in R1 and R2 versions of the R1 and R2) instead of an FTDI chip to convert USB-UART interfaces.

On the board Mega 2560 version R2 added resistor, pulling to the ground line HWB microcontroller 8U2. This measure simplifies the process of updating the firmware and switching the device to DFU mode.

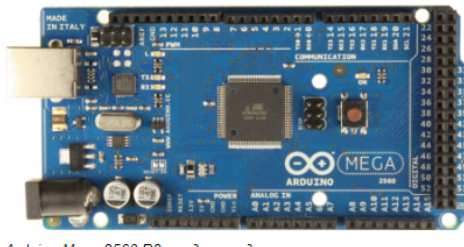


Figure 1.6 – Board Arduino Mega 2560

The Arduino Uno board consists of an Atmel AVR microcontroller (ATmega328) and element strapping for programming and integration with other circuits. Each board must have a linear voltage regulator 5 V and a 16 MHz quartz generator. The bootloader is pre-flashed into the microcontroller, so an external programmer is not needed. To work, you need to connect the platform to a computer using a USB cable, or supply power using an AC/DC adapter or battery.

The controller is programmed from the Arduino Software Integrated Environment (IDE). Programming takes place under the control of a resident uploader according to the STK500 protocol. A hardware programmer is not

needed. The microcontroller can be programmed through the ICSP in-circuit programmer connector without using a bootloader. The source code of the downloader program is freely available

You can connect various peripherals to the board itself - buttons, some types of sensors (temperature, pressure, light, acceleration, etc.), LEDs, LCD indicators to write a program and make you interact all this as you like.

Also connecting a variety of devices, for example, shields (shields), we add various functions - this way you can control all sorts of engines, servo machines, network load (light, heater, etc.).

You can also connect a GPS or GSM module and receive satellite coordinates or send data to your phone, for example, the coordinates of your car. You can connect an Ethernet module and send the necessary data to your site or write everything to an SD card, add communication, radio or Bluetooth channels.

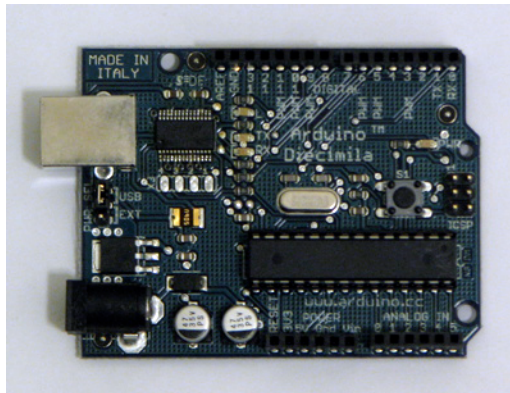
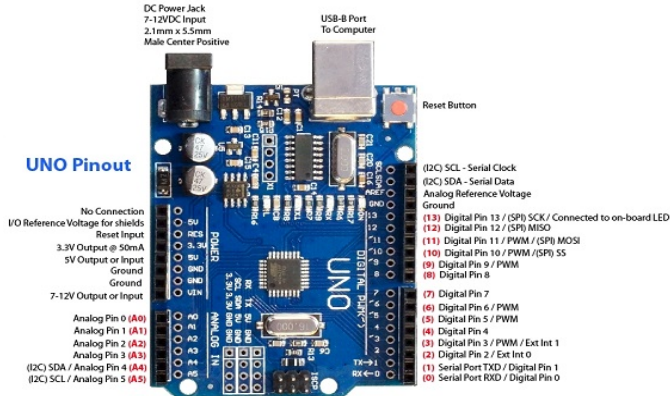


Figure 1.7 – Board Arduino Uno

Comparative characteristics of the above boards are given in Table 1.1.

In this laboratory work we will use Arduino UNO, detailed information about which is provided in Fig. 1.8.



Red numbers in paranthesis are the name to use when referencing that pin.
Analog pins are references as A0 thru A5 even when using as digital I/O

Figure 1.8 – Detailing the board Arduino Uno

Table 1.1 – Comparative characteristics of Arduino boards

Arduino:	Leonardo	Mega 2560	Pro Mini	Nano	Uno
Microcontroller	ATmega32u4	ATmega2560	ATmega168 or ATmega328	ATmega2560	ATmega328
Operating voltage	5B	5B	3.3B	5B	5B
Supply voltage (recommended)	7-12V	7-12V	3.35-12V	7-12V	7-12V
Supply voltage (limiting)	6-20V	6-20V		6-20V	6-20V
Digital I/O	20	54	14	54	14
Analog inputs	12	16	8	16	6
Maximum current of one Output	40 mA	40 mA	40 mA	40 mA	40 mA
Maximum output current	50 mA	50 mA		50 mA	50 mA
3.3V output					
Flash-memory	32 KB	256 KB	16 KB	256	32 KB
SRAM	2.5 KB	8 KB	1 KB	8 KB	2 KB
EEPROM	1 KB	4 KB	512 bytes	4 KB	1 KB
Clock speed	16 MHz	16 MHz	8 MHz	16 MHz	16 MHz

1.1.4 Features of the Arduino Uno

Power board. *The* Arduino Uno can receive power via a USB connection or from an external power source. The power source is selected automatically.

External power supply (not USB) can be supplied via AC / DC voltage converter (power supply) or rechargeable battery. The voltage converter is connected using a 2.1 mm connector with a central positive pole. The battery wires are connected to youwaterthe Gnd and Vin power connectors.

The platform can operate with external power supply from 6 V to 20 V. When the supply voltage is below 7 V, you can produce less than 5 V from K 5V, while the platform may be unstable. When using voltages above 12 V, the voltage regulator may overheat and damage the board. The recommended range is from 7V to 12V.

Power outputs:

VIN. *The* input is used to supply power from an external source (in the absence of 5 V from a USB connector or other adjustable power supply). The supply voltage is supplied through the data you are from.

5V. Adjustable voltage source used to power the microcontroller and components on the board. Power can be supplied from the VIN output through a voltage regulator, or from a USB connector, or another adjustable 5 V voltage source.

3V3. The output voltage of 3.3 V is generated by the built-in regulator on the board. Maximum current consumption 50 mA.

GND. Grounding terminals.

Memory. The ATmega328 microcontroller has 32 kB of flash memory, of which 0.5 KB is used to store the bootloader, as well as 2 kB of RAM and 1 KB of EEPROM.

Entrances and exits. *Each* of the 14 Uno digital outputs can be configured as input or output using the pinMode(), digitalWrite(), and digitalRead(),. The outputs operate at a voltage of 5 V. Each pin has a load resistor (disabled by default) of 20-50 kΩ and can pass up to 40 mA. Some pins have special functions:

Serial bus: 0 (RX) and 1 (TX). The pins are used to receive (RX) and transmit (TX) TTL data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial bus chip.

External interrupts: 2 and 3. These outputs can be configured to invoke an interrupt either at a lower value, or on the front or back front, or

when the value changes. For more information, see the `attachInterrupt()` function.

PWM: 3, 5, 6, 9, 10, and 11. Either pin provides an 8-bit PWM using `analogWrite()`.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). With the data youwaters, an SPI relationship is carried out, for which the SPI library is used.

LED: 13. Built-in LED connected to digital youwaterin 13. If the output value has a high potential, then the LED is on.

The Uno platform has 6 analog inputs (designated as A0 .. A5), each resolution is 10 bits (i.e. can take 1024 different values). By default, the pins have a measurement range of up to 5 V relative to the ground, but it is possible to change the upper bound using the AREF output and the `analogReference()` function. Some pins have additional functions:

I2C: 4 (SDA) and 5 (SCL). With the help of waters, I2C communication (TWI) is carried out, for the creation of which the Wire library is used.

Additional pair of platform pins:

AREF. Reference voltage for analog inputs. Used with `analogReference()` function.

Reset. Low signal strength on the output reboots the microcontroller. It is usually used to connect a reboot button on the expansion board, which closes access to the button on the Arduino board itself.

Communication. On the Arduino Uno platform, several devices are installed to communicate with a computer, other Arduino devices or microcontrollers. ATmega328 support a serial interface UART TTL (5 V) carried out by youwatersAMI 0 (RX) and 1 (TX). The ATmega8U2 chip installed on the board sends this interface via USB, programs on the computer side "communicate" with the board via a virtual COM port. ATmega8U2 firmware uses standard USB COM drivers, no third-party drivers are required, but on Windows, the ArduinoUNO.inf file is required to connect. Serial bus monitoring (Monitor) program Arduino allows you to send and receive text data when connected to the platform. The RX and TX LEDs on the platform will flash when transmitting data via an FTDI chip or USB connection (but not when using serial transmission through youwaters 0 and 1).

The SoftwareSerial library is possible to create serial data transmissions through any of the digital pins of Uno.

ATmega328 supports I2C interfaces (TWI) and SPI. The Arduino includes the Wire library for ease of use of the I2C bus.

Programming. The platform is programmed using Arduino software. From the Tools > Board menu, select "Arduino Uno" (according to the installed microcontroller). Detailed information is in the directory and instructions.

The ATmega328 microcontroller comes with a recorded bootloader that facilitates recording new programs without using external programmers. Communication is carried out by the original protocol STK500.

It is possible not to use the bootloader and program the microcontroller through the Arduino IDE and ICSP (in-circuit programming). Detailed information is in this manual.

Automatic (software) reboot. Uno is designed in such a way that before writing a new code, the reboot is carried out by the Arduino program itself on the computer, and not by pressing a button on the platform. One of the DTR lines of the ATmega8U2 chip, which controls the flow of data (DTR), is connected to the DTR pin in the ATmega328 microcontroller through a 100 nF capacitor. Activation of this line, that is, the supply of a low level signal, reboots the microcontroller. The Arduino program, using this feature, loads the code at the click of the Upload button in the programming environment itself. Low-level signaling over the DTR line is coordinated with the start of code writing, which shortens the loader timeout.

The function has another application. Uno reboots every time you connect to the Arduino app on a Mac X or Linux computer (via USB). The next half a second after the reboot, the bootloader works. During programming, the first few bytes of code are delayed to prevent the platform from receiving incorrect data (all but the code of the new program). If you are performing a one-time debugging of a sketch recorded in the platform, or entering any other data at the first start, you need to make sure that the program on the computer waits for a second before transferring data.

On Uno, it is possible to disable the automatic reboot line by breaking the corresponding line. Chip contacts at both ends of the line can be connected for recovery purposes. The line is marked "RESET-EN". Disabling automatic reboot is also possible by connecting a 110 ohm resistor between the 5V source and this line.

Current protection of the USB connector. The Arduino Uno has a built-in fuse (automatic) that can be independently restored, which protects the computer's USB port from short-circuit currents and overcurrents. Although almost all computers have similar protection, nevertheless, this fuse provides an additional barrier. The fuse is triggered when a current of more than 500 mA passes through the USB port and opens the circuit until normal current values are restored.

Physical characteristics. The length and width of the Uno PCB are 6.9 and 5.3 cm respectively. The USB connector and power connector are beyond these dimensions. Four holes in the board allow you to fix it on the surface. The distance between the digital pins of 7 and 8 is 0.4 cm, although between the other pins it is 0.25 cm.

1.1.5 Web-platform TinkerCAD

TinkerCAD is an online service and modeling environment for working with 3D objects and electronic circuits, now owned by Autodesk. Distinctive features are openness, free access, rich functionality of the editor. Group work, exchange of ready-made results, integration with popular catalogs of 3D models and remote 3D printing systems are supported.

The TinkerCAD electronic circuit simulation environment consists of the following components:

- editor of electronic circuits;
- emulator of work on basic electronic components;
- Arduino controller emulator;
- editing of sketches, in particular visual;
- system of debugging and simulation of projects using Arduino.

The presence of such components in the virtual space and an intuitive interface makes it possible to work with the Arduino board without losing functions that can be implemented in real space.

1.2 Progress

Register on the website <https://www.tinkercad.com/> and go to your profile. Join to the class created by the teacher at the link <https://www.tinkercad.com/joinclass/4XBSZNJ7A> or by the code 4XBSZNJ7A.

To find ourselves in the dashboard (Fig.1.9).

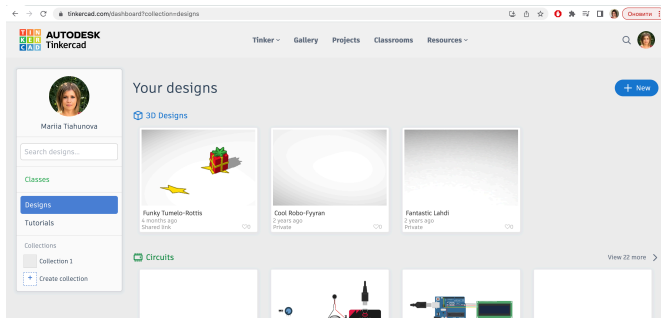


Figure 1.9 - Dashboard

Next, click "Design" - "New". And choose "Circuit". The sequence is clearly displayed in Fig. 1.10.

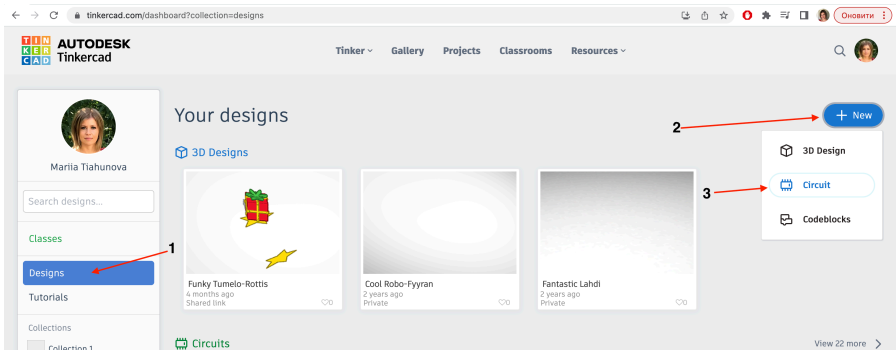


Figure 1.10 – Creating a project

We find ourselves in the project creation panel (fig. 1.11).

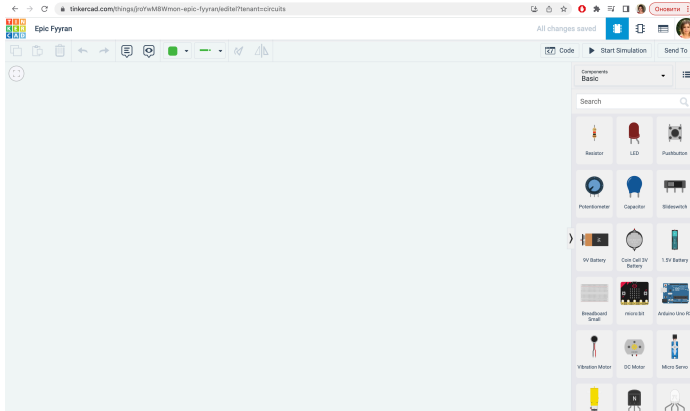


Figure 1.11 – Project creation panel

Next, when creating a project, in the search field by name in English, we find the devices corresponding to the task.

First, find the Arduino board. To do this, change "Basic" to "All" and enter "arduino" in the search field (fig. 1.12).

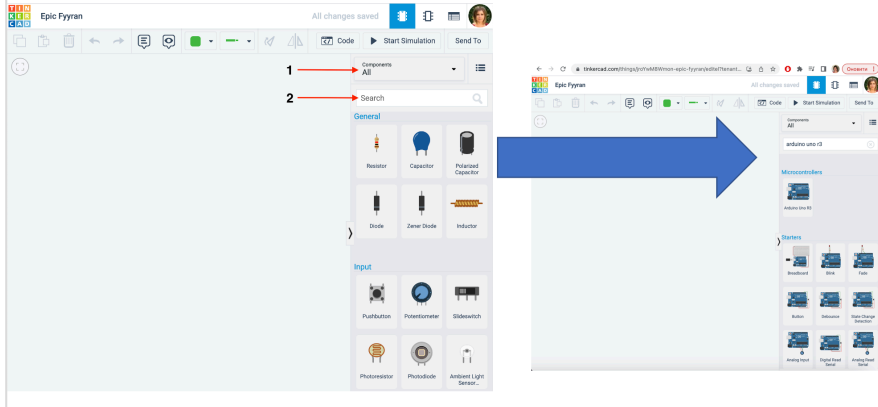


Figure 1.12 – Search for components

Left-click on a small image of the board, and it will appear in the development window. After that, click "Start Simulation", as shown in Fig. 1.13.

The simulation process begins.

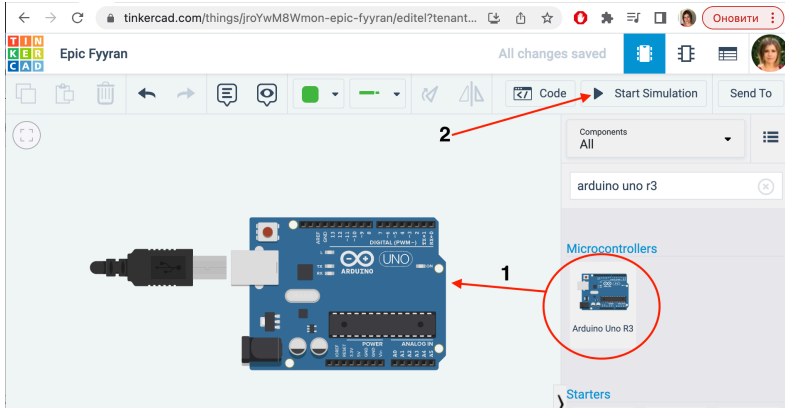


Figure 1.13 – Search for components

Please note that when starting the simulation of the board, the built-in LED will light up, when working with a real board, corresponds to the fact that the board is correctly connected to the computer and the corresponding drivers are installed (fig. 1.14 with pointer 1).

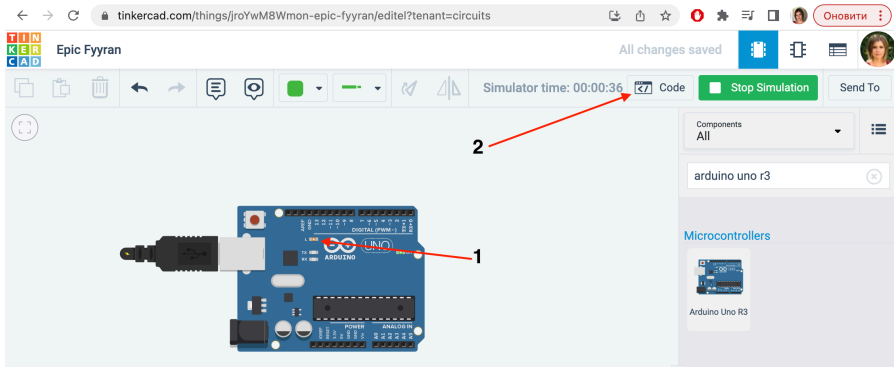


Figure 1.14 – Simulation of the board

Next, click on the "Code" (fig. 1.14 with a pointer 2) and select "Text". As a result, you get the program code responsible for the flickering of the embedded led (Fig. 1.15).

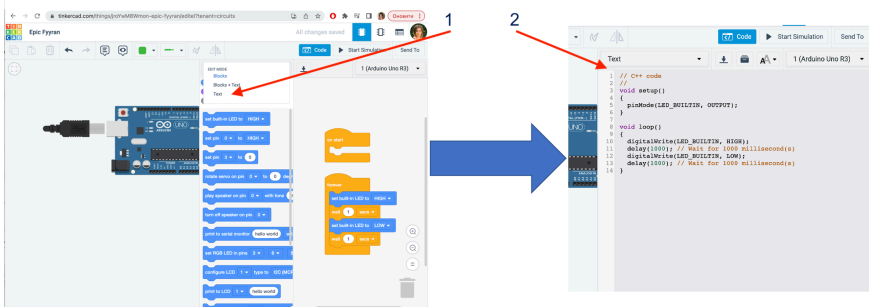


Figure 1.15 – Changing the program code of the board

1.3 Tasks to complete

Task 1. Change the burn time of the led and look at the results.

Task 2. Select one of the Starters Arduino Uno database kits presented in the Tincercad and analyze its work. Present the results in Report on Laboratory Work.

1.4 Contents

The report should have the following components:

- title page;
- the purpose of the work;
- the result of the tasks.

1.5 Review questions

1. Explain the concept of a cyber-physical system;
2. Describe technical features of the Arduino UNO board;
3. Describe analogues of Arduino;
4. Describe analogues of TincerCAD.

2 LABORATORY WORK №2

Modeling of traffic light on the TinkerCAD platform

The purpose of the work is to learn how to simulate the operation of traffic light on the TinkerCAD platform.

2.1 Progress

It is necessary in this laboratory work to collect according to the scheme in Fig. 2.1 Model of traffic light with a countdown indicator in TinkerCAD environment. Creating a system and checking its operation must be done in stages.

Design system in TinkerCAD such as shown in Fig. 2.1. The program code for the operation of this system is presented in the Listing 2.1.

Please note that if pin connections with other numbers were used in the system you assembled, then such changes must be made to the program code for the correct operation of your system.

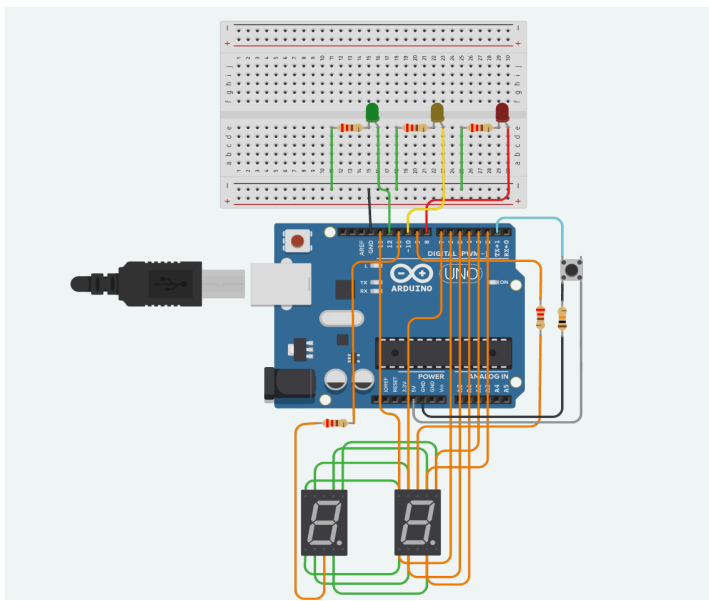


Figure 2.1 - Traffic light scheme with countdown

In this scheme, a new element has been added – a seven-segment indicator for the implementation of the countdown. Schematically, the 7 segment display looks like in Figure 2.2.

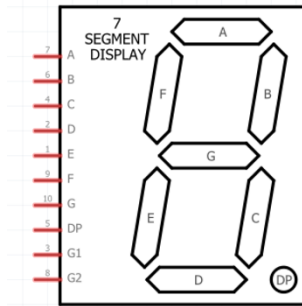


Figure 2.2 – 7 segment display scheme

Choose a seven-segment display with a common cathode D5611. The thermostats of the display are clearly presented in Fig. 2.3.

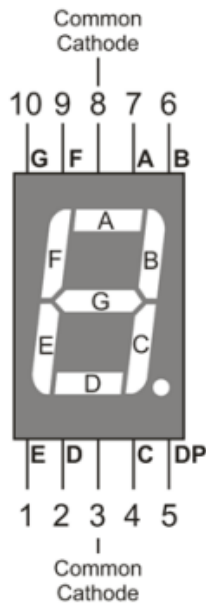


Figure 2.3 – 7 segment display with common cathode D5611

In order to reduce the number of pins connected (from 14 to 9) to the Arduino Uno, we will connect two seven-segment displays to each other (contact A with pin A, pin B with pin B, contact C with contact C, etc.). Switching the control of each of the displays separately will be regulated by the supply of voltage from the Arduino Uno board to a specific display. To do this, connect the G1 pin of the first seven-segment display to 9 digital th foam board Arduino Uno, pin G2 of the second seven-segment display to the 11th digitally foam board Arduino Uno. In this case, we will connect through a resistor corresponding to the selected display.

Next, connect the contacts A ... G seven-segment display to digital pins 2 - 7 and 13 Arduino Uno boards respectively.

Listing 2.1 – Sketch of traffic light with a timer

```
int buttonState = 0;
int t = 10;
int value;
int timer = 100;
int pin1 = 6;
int pin2 = 5;  //
int pin3 = 4;  //
int pin4 = 13; //
int pin5 = 7;  //
int pin6 = 2;  //
int pin7 = 3;
int gnd1 = 9;
int gnd2 = 11;
int led_red = 8;
int led_yellow = 10;
int led_green = 12;
void setup(){
  pinMode(1, INPUT);
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT); //The following sets up
  pinMode(pin5, OUTPUT); // all of the pins for use.
  pinMode(pin6, OUTPUT);
  pinMode(pin7, OUTPUT);
  pinMode(gnd1, OUTPUT);
```

```
--6--
5 |   | 7
  |--4--|
1 |   | 3
  --2--
```

```

    pinMode (gnd2, OUTPUT);
    pinMode(led_red, OUTPUT);
    pinMode(led_yellow, OUTPUT);
    pinMode(led_green, OUTPUT);
}
void loop() {
    digitalWrite(led_red, HIGH);
    for (int i = 0; i < timer; i++) {
// The for loop, for running the program 500 times.
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B0);
digitalWrite (pin4, B1);
digitalWrite (pin5, B0);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);
delay (t); // Writes 20 to the display.
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i++) {
digitalWrite (pin1, B0);
digitalWrite (pin2, B0);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B0);
digitalWrite (pin6, B0);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);
delay (t); // Writes 19 to the display.
digitalWrite (pin1, B0);

```

```

digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i++) {
digitalWrite (pin1, B0);
digitalWrite (pin2, B0);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B0);
digitalWrite (pin6, B0);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);
delay (t); // Writes 18 to the display.
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i++) {
digitalWrite (pin1, B0);
digitalWrite (pin2, B0);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B0);
digitalWrite (pin6, B0);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);

```

```

delay (t); // Writes 17 to the display.
digitalWrite (pin1, B0);
digitalWrite (pin2, B0);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B0);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
pb:
    delay(100);
    digitalWrite(gnd1, B0);
    digitalWrite(gnd2, B0);
    buttonState = digitalRead(1);
    if (buttonState == HIGH) {
        goto cn;
    }
    else {
        goto pb;
    }
cn:
    digitalWrite(led_yellow, HIGH);
    delay(2000);
    digitalWrite(led_green, HIGH);
    digitalWrite(led_red, LOW);
    digitalWrite(led_yellow, LOW);
    for (int i = 0; i < timer; i ++) {
        digitalWrite (pin1, B1);
        digitalWrite (pin2, B1);
        digitalWrite (pin3, B1);
        digitalWrite (pin4, B0);
        digitalWrite (pin5, B1);
        digitalWrite (pin6, B1);
        digitalWrite (pin7, B1);
        digitalWrite (gnd1, B1);
        digitalWrite (gnd2, B0);
        delay (t); // Writes 05 to the display.
        digitalWrite (pin1, B0);
        digitalWrite (pin2, B1);

```

```

digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B0);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i++) {
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);
delay (t); // Writes 04 to the display.
digitalWrite (pin1, B0);
digitalWrite (pin2, B0);
digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B1);
digitalWrite (pin6, B0);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i++) {
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);
delay (t); // Writes 03 to the display.

```

```

digitalWrite (pin1, B0);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B0);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i ++ ) {
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);
digitalWrite (gnd2, B0);
delay (t); // Writes 02 to the display.
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B0);
digitalWrite (pin4, B1);
digitalWrite (pin5, B0);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (t);
}
for (int i = 0; i < timer; i ++ ) {
digitalWrite (pin1, B1);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B0);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B1);

```

```

digitalWrite (gnd2, B0);
delay(t); // Writes 01 to the display.
digitalWrite(pin1, B0);
digitalWrite(pin2, B0);
digitalWrite(pin3, B1);
digitalWrite(pin4, B0);
digitalWrite(pin5, B0);
digitalWrite(pin6, B0);
digitalWrite(pin7, B1);
digitalWrite(gnd1, B0);
digitalWrite(gnd2, B1);
delay(t);
}
for (int i=0; i<timer; i++){
digitalWrite(pin1, B1);
digitalWrite(pin2, B1);
digitalWrite(pin3, B1);
digitalWrite(pin4, B0);
digitalWrite(pin5, B1);
digitalWrite(pin6, B1);
digitalWrite(pin7, B1);
digitalWrite(gnd1, B1);
digitalWrite(gnd2, B0);
delay(t); // Writes 00 to the display.
digitalWrite(pin1, B1);
digitalWrite(pin2, B1);
digitalWrite(pin3, B1);
digitalWrite(pin4, B0);
digitalWrite(pin5, B1);
digitalWrite(pin6, B1);
digitalWrite(pin7, B1);
digitalWrite(gnd1, B0);
digitalWrite(gnd2, B1);
delay(t);
}
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B0);
digitalWrite(led_yellow, HIGH);
    delay(2000);
digitalWrite(led_green, LOW);
digitalWrite(led_yellow, LOW);
}

```

2.2 Tasks to complete

Task 1. Simulate the operation of the traffic light according to the above scheme.

Task 2. Refine the program code so that for red led the countdown on the indicator continues from 20 to 0. The results of task 2 should be provided in the Report on the Verification work.

2.3 Contents

The report should have the following components:

- title page;
- the purpose of the work;
- the result of the task 1.

LITERATURE

1. Kudermetov R. K. Model-based systems engineering for the cyberphysical systems [Text] / R. K. Kudermetov, M.Y. Tiahunova, O.V. Polska // Internet of Things Technologies for Cyber Physical Systems: Practicum / H. I. Vorobets, V. S. Kharchenko (Eds.) – Chernivtsi National University, National Aerospace University “KhAI”, Zaporizhzhia National Technical University, 2020. – Section 4. – P. 102–132.
2. Lee, E. A. and S. A. Seshia, 2015: Introduction to Embedded Systems. v2.0. URL: <http://LeeSeshia.org>.
3. Lee, E. A., S. A. Seshia, and J. C. Jensen, 2013: EECS 149 - Introduction to Embedded Systems. Course website. URL: <http://chess.eecs.berkeley.edu/eecs149>.
4. R. G. Sanfelice. Analysis and Design of Cyber-Physical Systems. A Hybrid Control Systems Approach // Cyber-Physical Systems: From Theory to Practice / D. Rawat, J. Rodrigues, I. Stojmenovic. — CRC Press, 2016. — ISBN 978-1-4822-6333-6.
5. Gong L., Yanga W., Zhoub Z., Mana D., Caic H., Zhoud X., Yange Z. An adaptive wireless passive human detection via fine-grained physical layer information // Ad Hoc Networks. 2016. Vol. 38. P. 38 - 50
6. Forms of NU «Zaporizhzhya Polytechnic». URL: <https://zp.edu.ua/novi-blanki-zntu>.