

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

**до лабораторних робіт з дисципліни
«Кіберфізичні системи»**

для студентів спеціальності 123 «Комп'ютерна інженерія»
за освітньою програмою «Комп'ютерні системи та мережі»
денної форми навчання

І частина

2021

Методичні вказівки до лабораторних робіт з дисципліни «Кіберфізичні системи» для студентів спеціальності 123 «Комп'ютерна інженерія» за освітньою програмою «Комп'ютерні системи та мережі» денної форми навчання (І частина) / Укл.: М.Ю. Тягунова – Запоріжжя: НУ «Запорізька політехніка», 2021. – 54 с.

Укладач: М.Ю. Тягунова, к.т.н., доцент

Рецензент: Р.К. Кудерметов, к.т.н., доцент

Відповідальний
за випуск: М.Ю. Тягунова, к.т.н., доцент

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 10
від 18.05.2021 р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 10
від 28.05.2021 р.

ЗМІСТ

1 ЛАБОРАТОРНА РОБОТА №1. НАЛАШТУВАННЯ СЕРЕДОВИЩА ARDUINO IDE	4
1.1 Загальні теоретичні відомості	4
1.1.1 Загальні відомості про кіберфізичні системи	4
1.1.2 Загальні відомості про плати Arduino.....	5
1.1.3 Плати-аналоги Arduino.....	6
1.1.4 Детальний опис плати Arduino Uno	11
1.2 Установлення та налаштування Arduino IDE	15
1.3 Деталі роботи у програмному середовищі	22
1.4 Хід роботи	25
1.5 Зміст звіту.....	27
1.6 Контрольні питання.....	27
2 ЛАБОРАТОРНА РОБОТА №2. РЕАЛІЗАЦІЯ РОБОТИ МОДЕЛІ СВІТЛОФОРУ ЗА КНОПКОЮ НА БАЗІ ARDUINO UNO	28
2.1 Загальні теоретичні відомості	28
2.2 Хід роботи.	28
2.2.1 Робота у Fritzing.....	29
2.2.2 Проектування світлофору із таймером.....	29
2.3 Зміст звіту.....	53
2.4 Контрольні питання.....	53
ЛІТЕРАТУРА.....	54

1 ЛАБОРАТОРНА РОБОТА №1. НАЛАШТУВАННЯ СЕРЕДОВИЩА ARDUINO IDE

Мета роботи: отримати елементарні навички роботи з платою Arduino UNO.

1.1 Загальні теоретичні відомості

1.1.1 Загальні відомості про кіберфізичні системи

Кіберфізична система (CPS) – це механізм, що контролюється або відстежується комп'ютерними алгоритмами і тісно пов'язаний з Інтернетом та його користувачами. В кіберфізичних системах програмне забезпечення тісно пов'язано з фізичними об'єктами. Компоненти CPS взаємодіють на різних часових та просторових рівнях та можуть мати різні, відмінні одна від одної моделі поведінки та взаємодіяти одна з одною різними шляхами, які можуть змінюватися в залежності від контексту. Прикладами кіберфізичних систем можна вважати розумні енергосистеми, безпілотні автомобільні системи, автоматизовані системи керування, робототехнічні системи, самокеровані літальні апарати.

Джерелами розвитку кіберфізичних систем стали різні за своєю специфікою сфери людської діяльності: аерокосмічна галузь, автомобільна промисловість, громадська інфраструктура, енергетика, охорона здоров'я, транспорт, розваги, побутова техніка та ін.

На відміну від вже існуючих вбудованих систем, повноцінні CPS розробляються у вигляді мережі з декількох складових частин, які займаються переробкою сировини. Головна ідея кіберфізичних систем тісно пов'язана з роботами та сенсорними системами, які обладнані «розумними» механізмами з достатніми обчислювальними можливостями для достатнього керування. Завдяки постійному науковому прогресу, CPS зазнають періодичних змін, які покращують зв'язок між фізичними та обчислювальними компонентами за допомогою «розумних» механізмів, покращуючи таким чином здатність до адаптації, підвищуючи автономність, ефективність, надійність та безпечність та розширюючи функціональність кіберфізичних систем. Завдяки цьому потенціал їх використання розширюється у наступні виміри людської діяльності: втручання у процес водіння (напр., уникнення зіткнень), точність при проведенні

операцій (напр., роботохірургія), рятувальні або дослідницькі операції (за участь робототехніки), координація перевезень (напр., повітряний трафік).

CPS застосовують в якості обладнаних сенсорами автономних систем, які мають можливість зв'язуватися з оператором. Наприклад, велика кількість мереж з бездротовими сенсорами відслідковують певний параметр, а оброблену інформацію передають до центрального вузла. Інші типи кіберфізичних систем можуть включати розумні енергосистеми, автономне виробництво, моніторинг стану здоров'я, системи контролю процесів, робототехніку та самокерований авіатранспорт.

Одним із варіантів реалізації ядра кіберфізичної системи є використання плати Arduino. Розглянемо її більш детально у даній лабораторній роботі.

1.1.2 Загальні відомості про плати Arduino

Arduino представляє собою невелику плату з власним процесором і пам'яттю. На платі також є пара десятків контактів, до яких можна підключати всілякі компоненти: лампочки, датчики, мотори, чайники, роутери, магнітні дверні замки і взагалі все, що працює від електрики.

За допомогою Arduino можна завантажити програму, яка буде керувати всіма цими пристроями за заданим алгоритмом. Це апаратна обчислювальна платформа, основними компонентами якої є плата введення-виведення (рис.1.1) і середовище розробки (IDE).

Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, що виконується на комп'ютері, і бути частиною величезної кіберфізичної системи.

Завдяки відкритості Arduino, її пристрій відомо і допускає вільне модифікацію. Тому будь-який виробник плат може випускати аналог плати Arduino, вносити зміни в саму плату.

Програмне управління реалізується на C ++ спрощеної версії, часто званої також Wiring. Розробка керуючої програми може вестися і через будь-який зручний C / C ++ інструментарій. Платформа працює з операційними системами Linux, Windows і MacOS X.

Таким чином за допомогою плати Arduino можна створити нескінчену кількість унікальних класних гаджетів, зроблених своїми руками і за власною задумом.



Рисунок 1.1 - Arduino і сірникова коробка

Завдяки відкритості Arduino, її пристрій відомо і допускає вільне модифікацію. Тому будь-який виробник плат може випускати аналог плати Arduino, вносити зміни в саму плату, не кажучи вже про вільну комплектацію наборів (kits).

1.1.3 Плати-аналоги Arduino

Оскільки Arduino - це бренд виробника, аналоги мають інші, але зазвичай співзвучні з Arduino назви - Freduino, Freeduino, DCcduino (рис.1.2), Xdrduino, Funduino, Robotale і безліч інших. Більш того, бренд взагалі може бути не вказано, а буде присутній напис типу for Arduino.



Рисунок 1.2 – Плата DCcduino UNO

Оригінальні Arduino виробляються в Італії, більшість аналогів - в Китаї. Візуально більшість цих плат виглядає однаково (на Arduino присутній фірмовий логотип) і навіть мають однаковий синій колір (виключення є, але небагато, Funduino, наприклад, червоного

кольору). Також плати-аналоги Arduino зазвичай в своїй назві мають такі ж приставки, як і сама Arduino, наприклад, DCcduino UNO відповідає Arduino UNO.

Існує декілька версій платформ Arduino. Остання версія Leonardo базується на мікроконтролері ATmega32u4. Uno, як і попередня версія Duemilanove створені на мікроконтролері Atmel ATmega328. Старі версії платформи Diecimila та перша робоча Duemilanoves були створені на основі Atmel ATmega168, більш ранні версії використали ATmega8. Arduino Mega2560, у свою чергу, побудована на мікроконтролері ATmega2560. Далі представлені основні версії плат Arduino.

Arduino Nano - це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328 (Arduino Nano 3.0) або ATmega168 (Arduino Nano 2.x), адаптоване для використання з макетними платами (рис.1.3). За функціональністю пристрій схожий на Arduino Duemilanove, і відрізняється від нього розмірами, відсутністю роз'єму живлення, а також іншим типом (Mini-B) USB-кабелю. Arduino Nano розроблено і випускається фірмою Gravitech.



Рисунок 1.3 – Плата Arduino Nano

Arduino Pro Mini - це пристрій на базі мікроконтролера ATmega328. У його склад входить: 14 цифрових входів / виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 8 аналогових входів, кварцовий резонатор, кнопка скидання і контактні площадки для впаювання роз'ємів. Шестиконтактний роз'єм може служити для живлення та взаємодії з платою через USB за допомогою FTDI-перехідника або макетної плати Sparkfun.

Arduino Pro Mini призначений для напівстаціонарного монтажу в різні обладнання або установки (рис.1.4). Плата спеціально поставляється без впаяних роз'ємів, що дозволяє користувачеві паяти

дроти або використовувати необхідні типи роз'ємів на свій розсуд. По розташуванню висновків Arduino Mini Pro сумісний Arduino Mini.

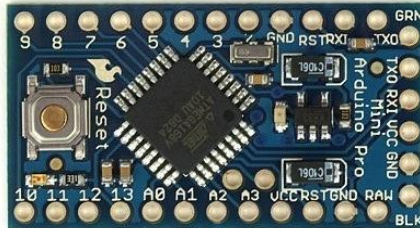


Рисунок 1.4 – Плата Arduino Pro Mini

Arduino Leonardo - до його складу входить все необхідне для роботи з мікроконтролером: 20 цифрових входів/виходів (7 з яких можуть працювати ШИМ-виходами, 12 - в якості аналогових входів), кварцевий резонатор на 16 МГц, роз'єм мікро- USB, роз'єм живлення, роз'єм для внутрішньо-схемного програмування ICSP (In-Circuit Serial Programming) і кнопка скидання (рис.1.5). Для роботи з Leonardo досить просто подати живлення від AC/DC- адаптера або батареї, або підключити його до комп'ютера за допомогою USB- кабелю.

Відмінність Leonardo від усіх попередніх плат полягає в тому, що його USB-контролер вбудований безпосередньо в мікроконтролер ATmega32U4, що виключає необхідність в додатковому процесорі. Завдяки цьому при під'єднанні до комп'ютера Leonardo може визначатися не лише як віртуальний (CDC) COM- порт, але і як звичайна миша або клавіатура. Крім того, така архітектура робить вплив і на поведінку плати.



Рисунок 1.5 – Плата Arduino Leonardo

Arduino Mega 2560 - до його складу входить усе необхідне для зручної роботи з мікроконтролером: 54 цифрові входи/виходи (з яких 15 можуть використовуватися як ШИМ-виходи), 16 аналогових входів, 4 UART (апаратних приймача для реалізації послідовних інтерфейсів), кварцевий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм ICSP для внутрішньо-схемного програмування і кнопка скидання (рис.1.6). Для початку роботи з пристроєм досить просто подати живлення від AC/DC- адаптера або батарейки, або підключити його до комп'ютера за допомогою USB- кабелю. Arduino Mega сумісний з більшістю плат розширення, розроблених для Arduino Duemilanove і Diecimila.

Arduino Mega 2560 відрізняється від усіх попередніх плат тим, що в ній для перетворення інтерфейсів USB - UART замість мікросхеми FTDI використовується мікроконтролер ATmega16U2 (ATmega8U2 у версіях плат R1 та R2).

На платі Mega 2560 версії R2 доданий резистор, що підтягує до землі лінію HWB мікроконтролера 8U2. Подібна міра дозволяє спростити процес оновлення прошивки і перехід пристрою в режим DFU.



Рисунок 1.6 – Плата Arduino Mega 2560

Плата Arduino Uno складається з мікроконтролера Atmel AVR (ATmega328) і елементного обв'язування для програмування і інтеграції з іншими схемами (рис.1.7). На кожній платі обов'язково є присутніми лінійний стабілізатор напруги 5 В і 16 МГц кварцевий генератор. В мікроконтролер попередньо прошитий завантажувач тому зовнішній програматор не потрібен. Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера AC/DC або батареї.

Контролер програмується з інтегрованого середовища програмного забезпечення Arduino (IDE). Програмування відбувається

під управлінням резидентного завантажувача по протоколу STK500. Апаратний програматор при цьому не потрібен. Мікроконтролер можна запрограмувати через роз'єм для внутрисхемного програматора ICSP, не використовуючи завантажувач. Вихідний код програми-завантажувача знаходиться у вільному доступі. До самої плати можна підключати різну периферію.

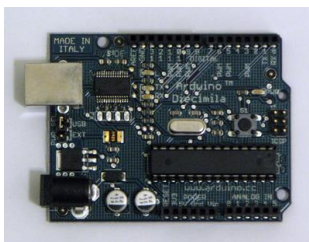


Рисунок 1.7 – Плата Arduino Uno

Порівняльні характеристики перелічених плат надані в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристики плат Arduino

Arduino:	Leonardo	Mega 2560	Pro Mini	Nano	Uno
Мікроконтролер	ATmega32u4	ATmega2560	ATmega168 или ATmega328	ATmega2560	ATmega328
Робоча напруга	5В	5В	3.3В	5В	5В
Напруга живлення (рекомендована)	7-12В	7-12В	3.35-12В	7-12В	7-12В
Напруга живлення (граничне)	6-20В	6-20В		6-20В	6-20В
Цифрові входи/виходи	20	54	14	54	14
Аналогові входи	12	16	8	16	6
Максимальний струм одного виведення	40 мА	40 мА	40 мА	40 мА	40 мА
Максимальний вихідний струм виведення 3.3V	50 мА	50 мА		50 мА	50 мА
Flash- пам'ять	32 КБ	256 КБ	16 КБ	256	32 КБ
SRAM	2.5 КБ	8 КБ	1 КБ	8 КБ	2 КБ
EEPROM	1 КБ	4 КБ	512 байт	4 КБ	1 КБ
Тактова частота	16 МГц	16 МГц	8 МГц	16 МГц	16 МГц

1.1.4 Детальний опис плати Arduino Uno

У даній лабораторній роботі будемо використовувати Arduino UNO, детальну інформацію про яку надано на рис. 1.8.

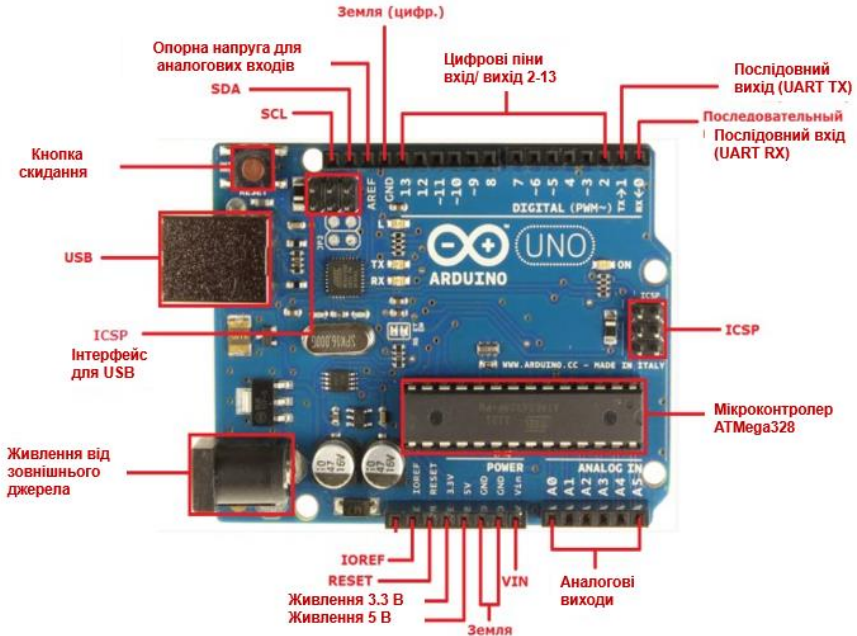


Рисунок 1.8 – Деталізація плати Arduino Uno

Живлення плати.

Arduino Uno може отримувати живлення через підключення USB або від зовнішнього джерела живлення. Джерело живлення вибирається автоматично.

Зовнішнє живлення (НЕ USB) може подаватися через перетворювач напруги AC / DC (блок живлення) або акумуляторною батареєю. Перетворювач напруги підключається за допомогою роз'єму 2.1 мм з центральним позитивним полюсом. Провід від батареї підключаються до висновків Gnd і Vin роз'єму живлення.

Платформа може працювати при зовнішньому живленні від 6 В до 20 В. При напрузі живлення нижче 7 В, висновок 5V може

видавати менше 5 В, при цьому платформа може працювати нестабільно. При використанні напруги вище 12 В регулятор напруги може перегрітися і пошкодити плату. Рекомендований діапазон від 7 В до 12 В.

Виводи живлення:

- **VIN.** Вхід використовується для подачі живлення від зовнішнього джерела (за відсутності 5 В від роз'єму USB або іншого регульованого джерела живлення). Подача напруги живлення відбувається через даний висновок;

- **5V.** Регульований джерело напруги, що використовується для живлення мікроконтролера і компонентів на платі. Харчування може подаватися від виведення VIN через регулятор напруги, або від роз'єму USB, або іншого регульованого джерела напруги 5 В;

- **3V3.** Напруга на виводі 3.3 В генерується вбудованим регулятором на платі. Максимальне споживання струму 50 мА;

- **GND.** Виводи заземлення.

Пам'ять.

Мікроконтролер ATmega328 має 32 кБ флеш пам'яті, з яких 0.5 кБ використовується для зберігання завантажувача, а також 2 кБ ОЗУ (SRAM) і 1 кБ EEPROM. (Яка читається і записується за допомогою бібліотеки EEPROM).

Входи і Виходи.

Кожен з 14 цифрових виходів Uno може бути налаштований як вхід або вихід, використовуючи функції `pinMode ()`, `digitalWrite ()`, і `digitalRead ()`. Виходи працюють при напрузі 5 В. Кожен вивід має навантажувальний резистор (за замовчуванням відключений) 20-50 кОм і може пропускати до 40 мА. Деякі виводи мають особливі функції:

- **Послідовна шина: 0 (RX) і 1 (TX).** Виводи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виводи підключені до відповідних виводів мікросхеми послідовної шини ATmega8U2 USB-to-TTL;

- **Зовнішнє переривання: 2 і 3.** Дані виводи можуть бути налаштовані на виклик переривання або на молодшому значенні, або на передньому чи задньому фронті, або при зміні значення. Детальна інформація знаходиться в описі функції `attachInterrupt ()`;

- **ШІМ: 3, 5, 6, 9, 10, і 11.** Будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції `analogWrite ()`;
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** За допомогою даних висновків здійснюється зв'язок SPI, для чого використовується бібліотека SPI;
- **LED: 13.** Вбудований світлодіод, підключений до цифрового висновку 13. Якщо значення на виведення має високий потенціал, то світлодіод горить.

На платформі Uno встановлені 6 аналогових входів (позначених як A0 .. A5), кожен дозволом 10 біт (тобто може приймати 1024 різних значення). Стандартно виводи мають діапазон вимірювання до 5 В щодо землі, проте є можливість змінити верхню межу за допомогою виведення AREF і функції `analogReference ()`. Деякі виводи мають додаткові функції, наприклад, **I2C**.

I2C: 4 (SDA) і 5 (SCL). За допомогою висновків здійснюється зв'язок I2C (TWI), для створення якої використовується бібліотека Wire.

Додаткова пара виводів платформи:

- **AREF.** Опорна напруга для аналогових входів. Використовується з функцією `analogReference ()`;
- **Reset.** Низький рівень сигналу на виводі перезавантажує мікроконтролер. Зазвичай застосовується для підключення кнопки перезавантаження на платі розширення, що закриває доступ до кнопки на самій платі Arduino.

Зв'язок

На платформі Arduino Uno встановлено кілька пристроїв для здійснення зв'язку з комп'ютером, іншими пристроями Arduino або мікроконтроллерами. ATmega328 підтримують послідовний інтерфейс UART TTL (5 В), здійснюваний висновками 0 (RX) і 1 (TX). Встановлена на платі мікросхема ATmega8U2 направляє даний інтерфейс через USB, програми на стороні комп'ютера "спілкуються" з платою через віртуальний COM порт. Прошивка ATmega8U2 використовує стандартні драйвера USB COM, ніяких сторінок драйверів не потрібно, але на Windows для підключення потрібно файл ArduinoUNO.inf. Моніторинг послідовної шини (Serial Monitor) програми Arduino дозволяє посилати і отримувати текстові дані при підключенні до платформи. Світлодіоди RX і TX на платформі будуть мигати при передачі даних через мікросхему FTDI або USB

підключення (але не при використанні послідовної передачі через висновки 0 і 1).

Бібліотекою SoftwareSerial можливо створити послідовну передачу даних через будь-який з цифрових виводів Uno.

ATmega328 підтримує інтерфейси I2C (TWI) і SPI. В Arduino включена бібліотека Wire для зручності використання шини I2C.

Програмування

Платформа програмується за допомогою ПЗ Arduino IDE. З меню Tools> Board вибирається «Arduino Uno» (згідно зі встановленим мікроконтролеру). Детальна інформація знаходиться в довіднику і інструкціях.

Мікроконтролер ATmega328 поставляється з записаним завантажувачем, що полегшує запис нових програм без використання зовнішніх програматоров. Зв'язок здійснюється оригінальним протоколом STK500.

Є можливість не використовувати завантажувач і запрограмувати мікроконтролер через висновки ICSP (внутрішньосхемне програмування).

Автоматичне (програмне) перезавантаження

Uno розроблена таким чином, щоб перед записом нового коду перезавантаження здійснювалася самою програмою Arduino на комп'ютері, а не натисканням кнопки на платформі. Одна з ліній DTR мікросхеми ATmega8U2, керуючих потоком даних (DTR), підключена до виходу перезавантаження мікроконтролеру ATmega328 через 100 нФ конденсатор. Активація даної лінії, тобто подача сигналу низького рівня, перезавантажує мікроконтролер. Програма Arduino, використовуючи цю функцію, завантажує код одним натисканням кнопки Upload в самому середовищі програмування. Подача сигналу низького рівня по лінії DTR скоординована з початком запису коду, що скорочує таймаут завантажувача.

Функція має ще одне застосування. Перезавантаження Uno відбувається кожного разу при підключенні до програми Arduino на комп'ютері з ОС Mac X або Linux (через USB). Наступні півсекунди після перезавантаження працює завантажувач. Під час програмування відбувається затримка декількох перших байтів коду, щоб уникнути отримання платформою некоректних даних (всіх, окрім коду нової програми). Якщо проводиться разова налагодження скетчу, записаного в платформу, або введення будь-яких інших даних при

першому запуску, необхідно переконатися, що програма на комп'ютері очікує протягом секунди перед передачею даних.

На Uno є можливість відключити лінію автоматичної перезавантаження розривом відповідної лінії. Контакти мікросхем з обох кінців лінії можуть бути з'єднані з метою відновлення. Лінія маркована «RESET-EN». Вимкнення автоматичного перезавантаження також можливо підключивши резистор 110 Ом між джерелом 5 В і даною лінією.

Струмівий захист роз'єму USB. В Arduino Uno вбудований запобіжник (автомат), який може самостійно відновлюватися, що захищає порт USB комп'ютера від струмів короткого замикання і надструмів. Хоча практично всі комп'ютери мають подібний захист, тим не менш, цей запобіжник забезпечує додатковий бар'єр. Запобіжник зпрацьовує при проходженні струму більше 500 мА через USB порт і розмикає ланцюг до тих поки нормальні значення струмів НЕ будуть відновлено.

Фізичні характеристики. Довжина і ширина друкованої плати Uno складають 6.9 і 5.3 см відповідно. Роз'єм USB і силовий роз'єм виходять за межі даних розмірів. Чотири отвори в платі дозволяють закріпити її на поверхні. Відстань між цифровими висновками 7 і 8 дорівнює 0,4 см, хоча між іншими висновками воно становить 0,25 см.

1.2 Установлення та налаштування Arduino IDE

Для роботи з платою, необхідно підключити її до комп'ютера кабелем USB (з рознімами типу USB-A та USB-B).

Далі необхідно завантажити середовище розробки за посиланням <https://www.arduino.cc/en/Main/Donate> або з робочої студентської папки.

Далі необхідно установити програмне середовище згідно наступних рисунків 1.9 – 1.16.

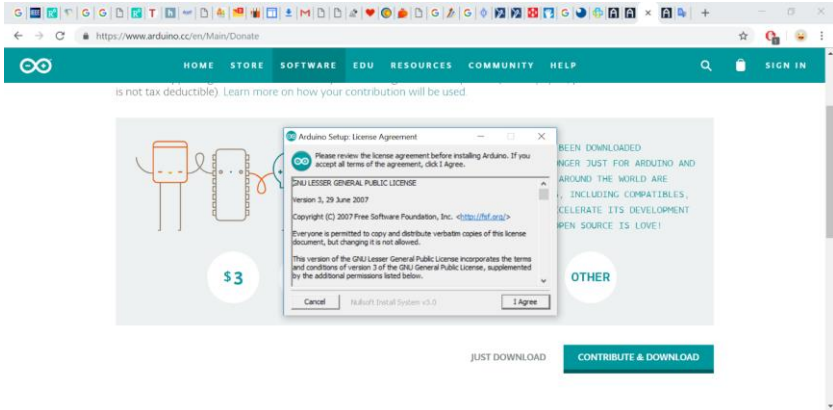


Рисунок 1.9 – Установлення Arduino етап1

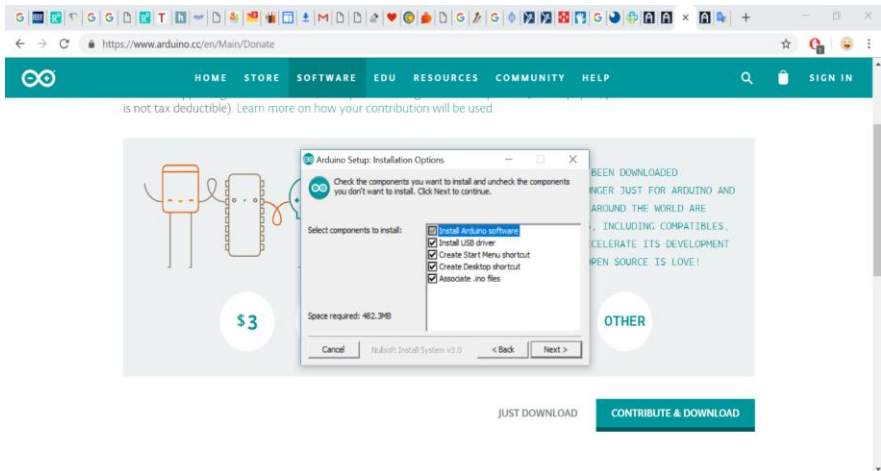


Рисунок 1.10 – Установлення Arduino етап2

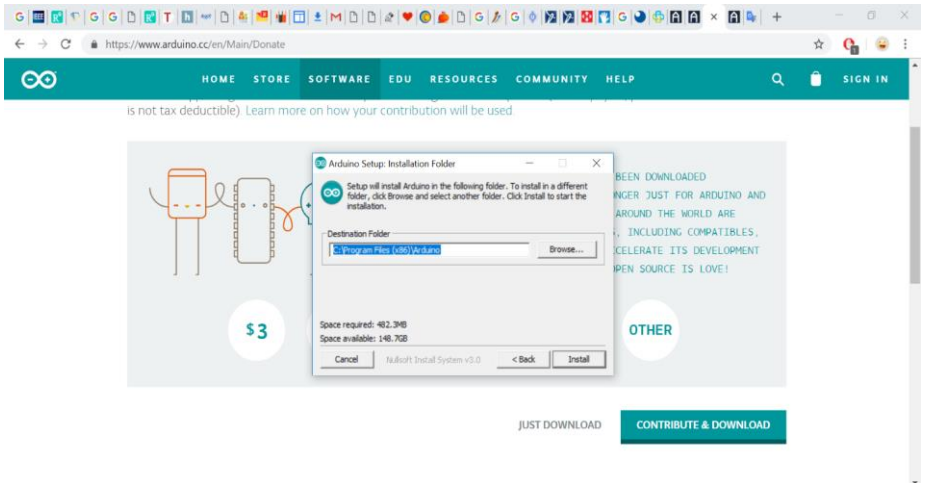


Рисунок 1.11 – Установлення Arduino етап3

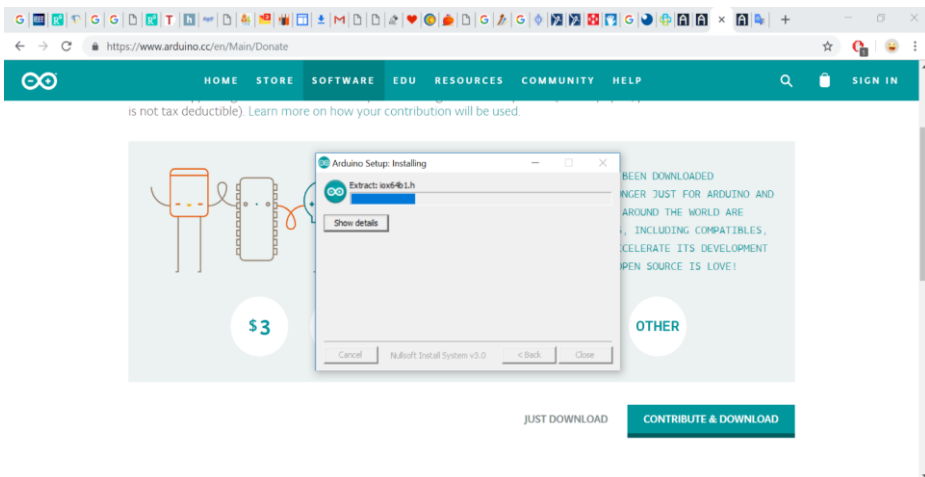


Рисунок 1.12 – Установлення Arduino етап4

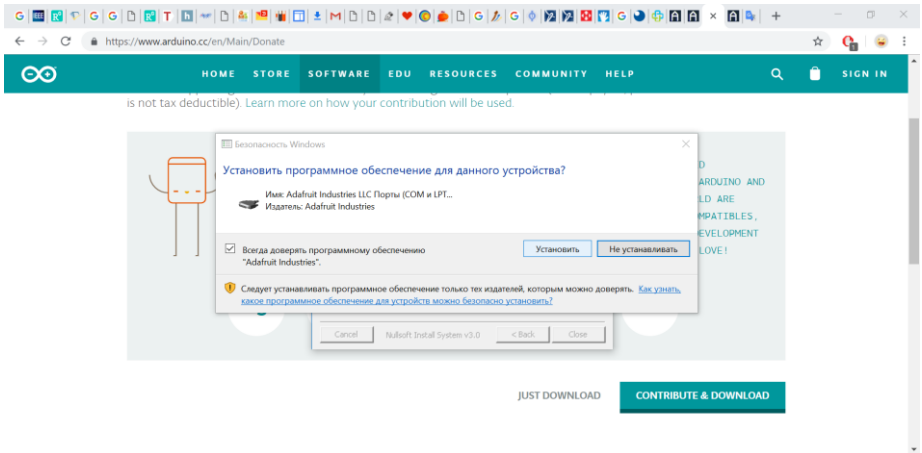


Рисунок 1.13 – Установления Arduino этап5

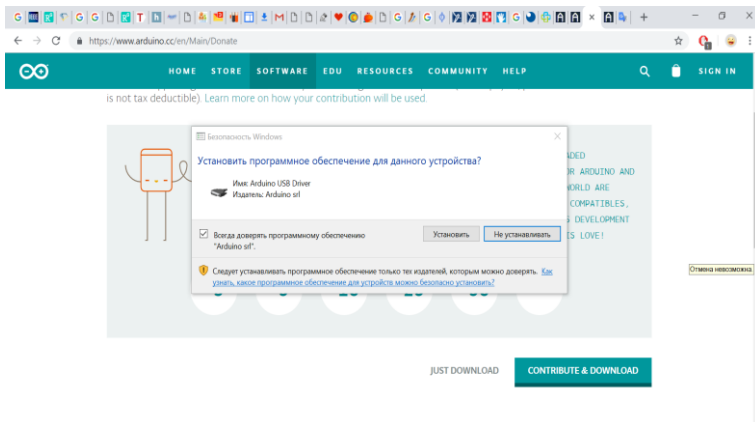


Рисунок 1.14 – Установления Arduino этап6

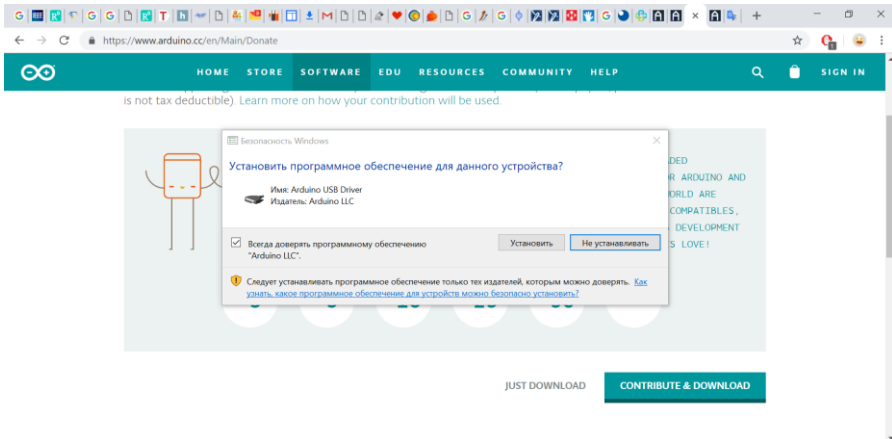


Рисунок 1.15 – Установлення Arduino етап7

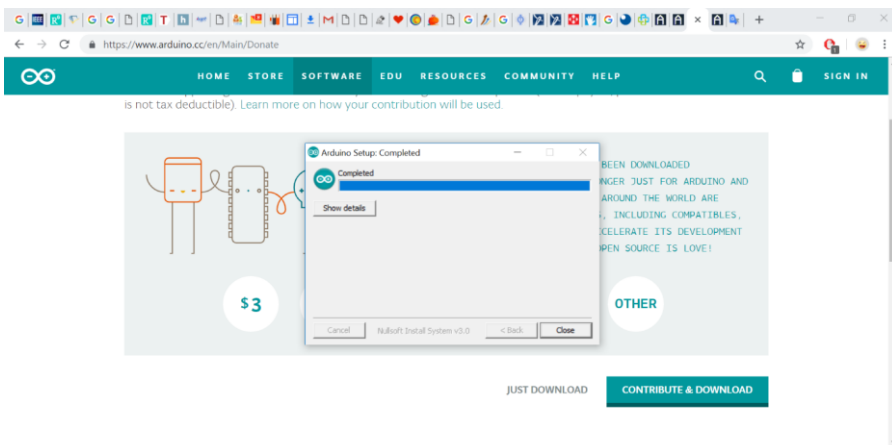


Рисунок 1.16 – Установлення Arduino етап8

Після установки ПЗ, необхідно підключити плату до комп'ютера та налаштувати драйвер (рис.1.17).

Підключіть вашу плату і почекайте, поки Windows почне процес установки драйвера. Через деякий час, не дивлячись на всі її спроби, процес закінчиться безрезультатно.

Натисніть на кнопку ПУСК і відкрийте Панель управління.

В панелі керування перейдіть на вкладку Система і безпека (System and Security). Потім виберіть Система. Коли відкриється вікно Система, виберіть Диспетчер пристроїв (Device Manager).

Зверніть увагу на порти (COM і LPT). Ви побачите відкритий порт під назвою «Arduino UNO (COMxx)».

Клацніть на назві «Arduino UNO (COMxx)» правою кнопкою мишки і виберіть опцію «Оновити драйвер» (Update Driver Software).

Натисніть "Browse my computer for Driver software".

Для завершення виберіть завантажений драйвера для Uno - «ArduinoUNO.inf», розташований в папці Drivers програмного забезпечення для Arduino (не в підкаталозі «FTDI USB Drivers»).

На цьому Windows закінчить установку драйвера.

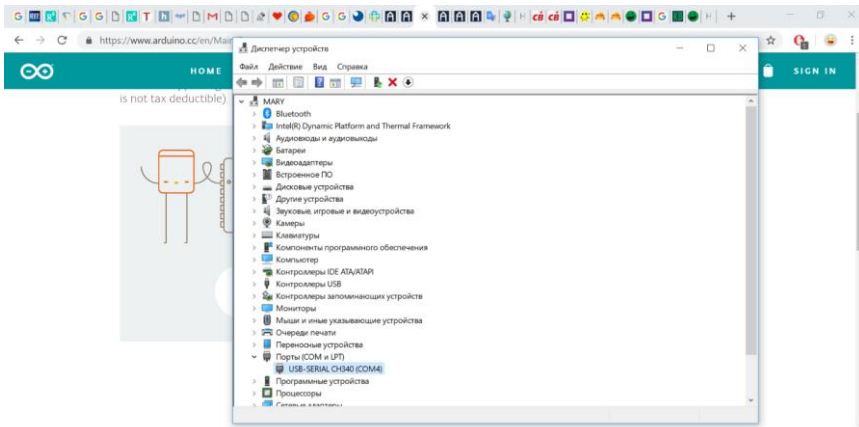


Рисунок 1.17 – Установка драйверу

Далі завантажуюмо установлений додаток (рис.1.18).

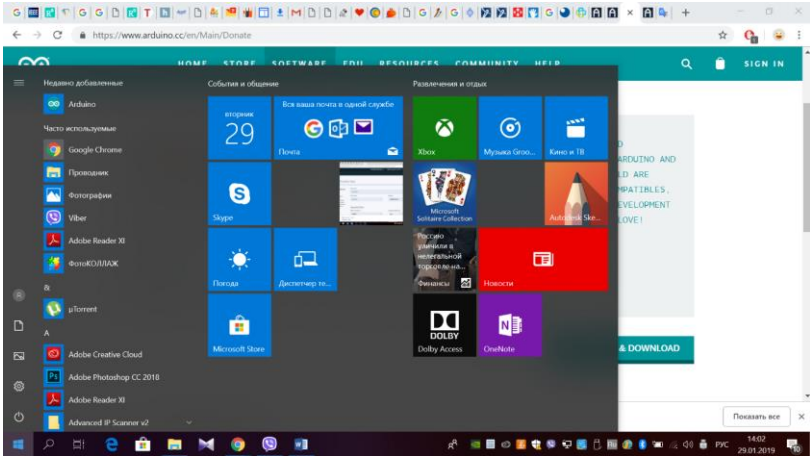


Рисунок 1.18 – Завантаження додатка Arduino

При цьому відкривається вікно, у якому можна побачити скетч та дописати його у відповідності до обраного завдання.

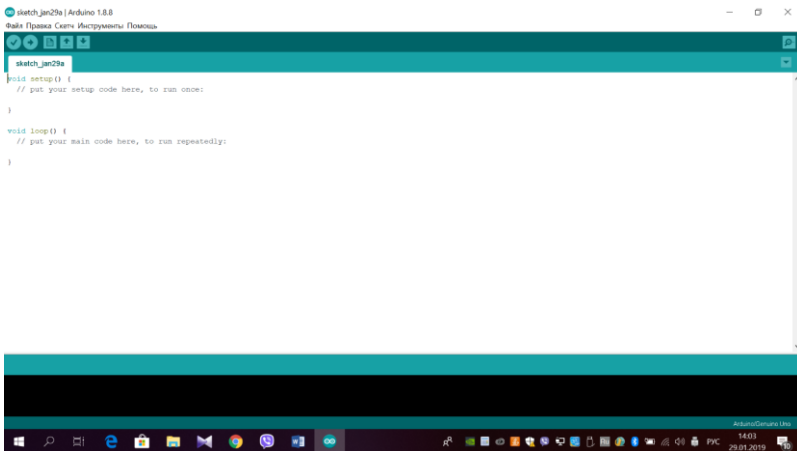


Рисунок 1.19 – Вікно завантаженого додатка Arduino

Середовище розробки Arduino складається з вбудованого текстового редактора програмного коду, області повідомлень, вікна виведення тексту (консолі), панелі інструментів з кнопками часто використовуваних команд і декількох меню. Для завантаження

програм і зв'язку середовище розробки підключається до апаратної частини Arduino (рис.1.19).

Далі перевіримо (а у разі необхідності, налаштуємо) усі необхідні налаштування згідно рис.1.20, 1.21.

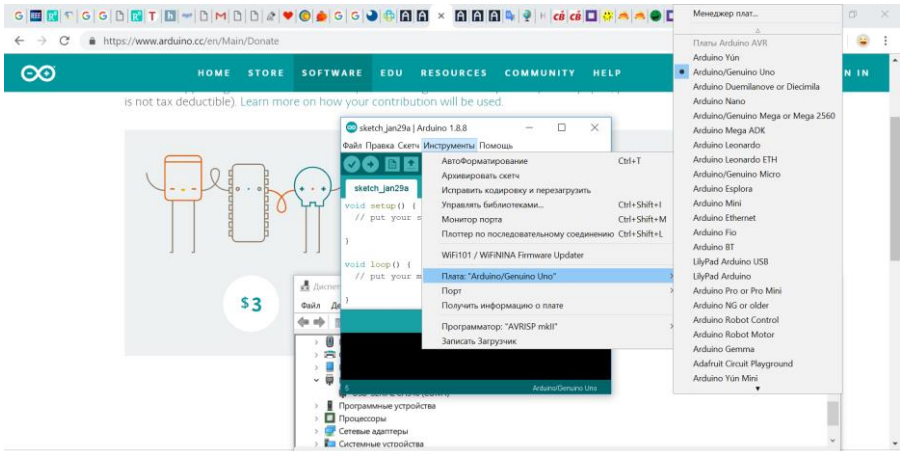


Рисунок 1.20 – Вікно вибору плати Arduino UNO у програмному середовищі

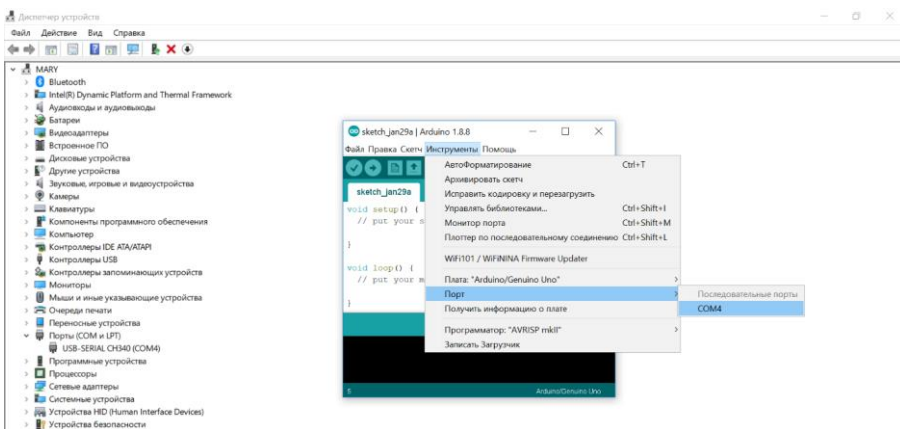


Рисунок 1.21 – Вікно вибору порту Arduino UNO у програмному середовищі

1.3 Деталі роботи у програмному середовищі

Програма, написана в середовищі Arduino, називається **скетч**. Скетч пишеться в текстовому редакторі, що має інструменти вирізки / вставки, пошуку / заміни тексту. Під час збереження і експорту проекту в області повідомлень з'являються пояснення, також можуть відображатися виникли помилки. Вікно виведення тексту (консоль) показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дозволяють перевірити і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини:

Ⓢ компіляція кода Verify / Compile – перевірка програмного коду на помилки, компіляція.

⏏ зупинка моніторинга Stop – зупинка моніторингу послідовної шини (Serial monitor) або затемнення інших кнопок.

📄 New - створення нового скетчу.

📂 відкрити файл скетча Open – відкриття меню доступу до всіх скетчам в блокноті. Відкривається натисканням в поточному вікні.

Примітка: через наявність помилки в Java дане меню не може прокручуватися; при необхідності відкрити скетч з цього списку зверніться в меню File | Sketchbook.

💾 Save – збереження скетчу.

📤 завантаження скетчу в контроллер Upload to I / O Board – компілює програмний код і завантажує його в пристрій Arduino. Опис завантаження наведено нижче.

📡 Serial Monitor – відкриття моніторингу послідовної шини (Serial monitor).

Додаткові команди згруповані в п'ять меню: File, Edit, Sketch, Tools, Help. Доступність меню визначається роботою, виконуваної в даний момент.

Edit:

Copy for Discourse – копіює в буфер обміну відповідний для розміщення на форумі код скетчу з виділенням синтаксису.

Copy as HTML – копіює код скетчу в буфер обміну як HTML код, для розміщення на веб-сторінках.

Sketch

Verify / Compile – перевірка скетчу на помилки.

Import Library – додає бібліотеку в поточний скетч, вставляючи директиву #include в код скетчу. Детальна інформація в описі бібліотек нижче (Libraries).

Show Sketch Folder – відкриває папку, яка містить файл скетчу, на робочому столі.

Add File ... – додає файл в скетч (файл буде скопійований з поточного місця розташування). Новий файл з'являється в новій закладці в вікні скетчу. Файл може бути видалений з скетчу за допомогою меню закладок.

Tools

Auto Format – дана опція оптимізує код, наприклад, вибудовує в одну лінію по вертикалі відкриває і закриває дужки і поміщає між ними твердження.

Board – вибір використовуваної платформи. Список з описом платформ наводиться нижче.

Serial Port – також можуть бути доступними список послідовних пристроїв передачі даних (реальних і віртуальних) на комп'ютері. Список оновлюється автоматично кожного разу при відкритті меню Tools.

Burn Bootloader

Пункти даного меню дозволяють записати Завантажувач (Bootloader) в мікроконтролер на платформі Arduino. Дана дія не потрібно в поточній роботі з Arduino, але стане в нагоді, якщо є новий ATmega (без завантажувача). Перед записом рекомендується перевірити правильність вибору платформи з меню. При використанні AVR ISP необхідно вибрати відповідний програматору порт з меню Serial Port.

Блокнот (Sketchbook)

Середовищем Arduino використовується принцип блокнота: стандартне місце для зберігання програм (скетчів). Скетчі з блокнота відкриваються через меню File> Sketchbook або кнопкою Open на панелі інструментів. При першому запуску програми Arduino автоматично створюється директорія для блокнота. Розташування блокнота змінюється через діалогове вікно Preferences.

Закладки, Файли і Компіляція

Дозволяють працювати з декількома файлами скетчів (кожен відкривається в окремій закладці). Файли коду можуть бути стандартними Arduino (без розширення), файлами C (розширення *.c), файлами C++ (*.cpp) або головними файлами (.h).

Завантаження скетчу в Arduino

Перед завантаженням скетчу потрібно задати необхідні параметри в меню Tools> Board і Tools> Serial Port. Платформи описуються далі по тексту. В ОС Mac послідовний порт може позначатися як dev / tty.usbserial-1B1 (для плати USB) або /dev/tty.USA19QW1b1P1.1 (для плати послідовної шини, підключеної через адаптер Keyspan USB-to-Serial). В ОС Windows порти можуть позначатися як COM1 або COM2 (для плати послідовної шини) або COM4, COM5, COM7 і вище (для плати USB). Визначення порту USB проводиться в поле Послідовною шини USB Диспетчера пристроїв Windows. В ОС Linux порти можуть позначатися як / dev / ttyUSB0, / dev / ttyUSB1.

Після вибору порту і платформи необхідно натиснути кнопку завантаження на панелі інструментів або вибрати пункт меню File> Upload to I / O Board. Сучасні платформи Arduino перезавантажуються автоматично перед завантаженням. На старих платформах необхідно натиснути кнопку перезавантаження. На більшості плат під час процесу будуть мигати світлодіоди RX і TX. Середовище розробки Arduino виведе повідомлення про закінчення завантаження або про помилки.

При завантаженні скетчу використовується Завантажувач (Bootloader) Arduino, невелика програма, що завантажується в мікроконтролер на платі. Вона дозволяє завантажувати програмний код без використання додаткових апаратних засобів.

1.4 Хід роботи

1. Установіть програмне середовище Arduino IDE, згідно описаних вище кроків;

2. Підключіть плату Arduino та зробіть відповідні необхідні налаштування (описані вище);

3. Запустіть приклад скетчу Blink (рис.1.22 – 1.23), та поясніть отримані результати;

4. Змініть час горіння світлодіоду на 5 с, а потім на 10 с, та при цьому час паузи необхідно задати рівним 3 с та 5 с відповідно.

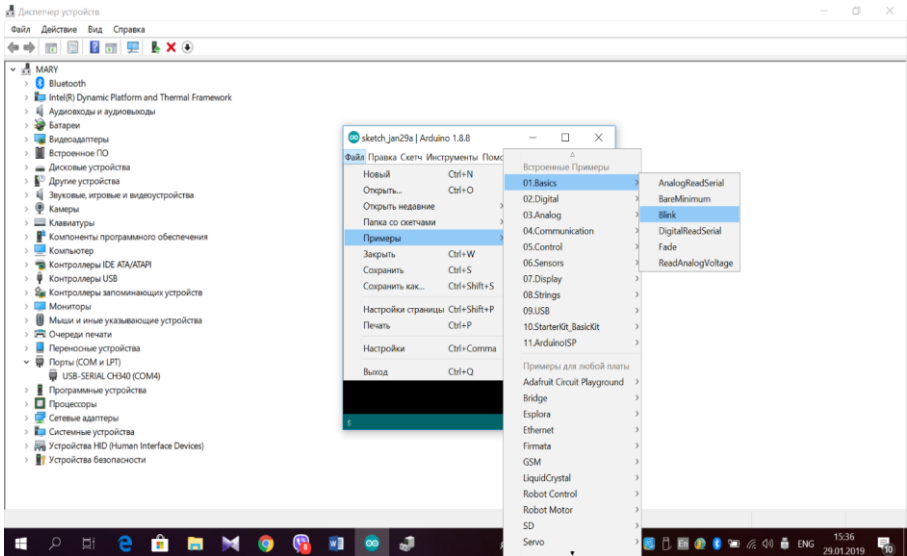


Рисунок 1.22 – Відкриття прикладу Blink

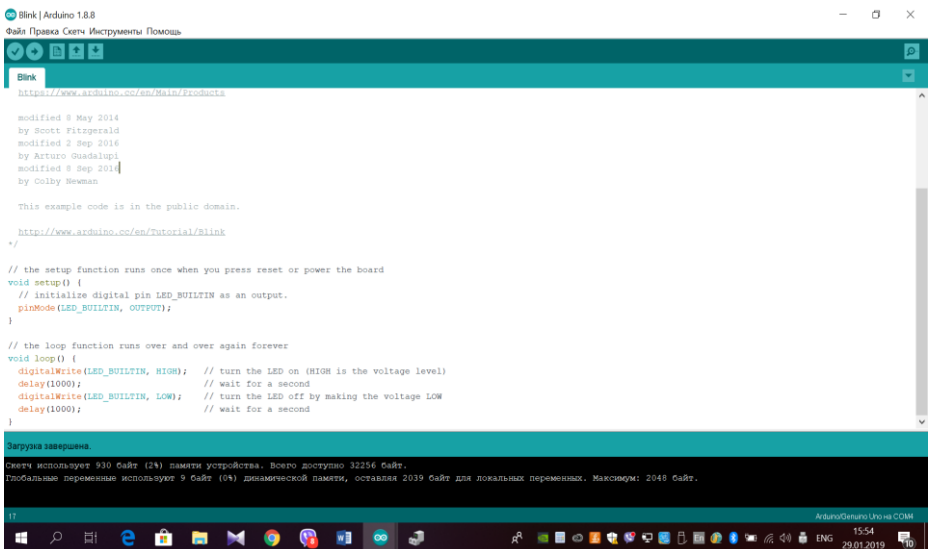


Рисунок 1.23 – Результати запуску прикладу Blink

1.5 Зміст звіту

Після виконання усіх етапів лабораторної роботи необхідно оформити звіт з наведеними власними результатами роботи.

Вміст звіту має таку структуру:

- титульний лист;
- мета роботи;
- скріншоти власних результатів самостійного виконання лабораторної роботи;
- відповіді на контрольні питання.

1.6 Контрольні питання

2. Поясніть поняття кіберфізичної системи;
3. Опишіть призначення та технічні особливості плати Arduino UNO;
4. Наведіть аналоги плати Arduino UNO;
5. Наведіть аналоги Arduino;
6. Що є скетч?
7. Опишіть перелік дій, які необхідно виконати, щоб працювати, при першому використанні плати Arduino UNO.

2 ЛАБОРАТОРНА РОБОТА №2. РЕАЛІЗАЦІЯ РОБОТИ МОДЕЛІ СВІТЛОФОРУ ЗА КНОПКОЮ НА БАЗІ ARDUINO UNO

Мета роботи: навчитись проектувати схеми систем та пристроїв у програмному середовищі Fritzing.

2.1 Загальні теоретичні відомості

Fritzing – це безкоштовна програма проектування і макетування схем.

Програма Fritzing призначена для розробки електронних пристроїв від прототипу у вигляді макетної плати до кінцевого продукту у вигляді друкованої плати.

Прилади створюються з готових елементів, список яких можна подивитися в правому верхньому кутку програми, які включають монтажну плату Arduino, різні аналогові і цифрові мікросхеми, транзистори, світлодіоди, резистори, конденсатори, кнопки, макетні плати, батарейки та навіть моторчики.

Щоб помістити їх на схему, їх досить обрати зі списку і перетягнути на робочий простір лівою кнопкою мишки.

Малювати схему можна як в режимі макетної плати, так і в режимі принципової схеми. При цьому, друга схема буде будуватися автоматично. Тобто, накидавши схему на макетній платі, вона автоматично побудується і у вигляді принципової схеми.

Завантажити програмне середовище можна за посиланнями <http://softrare.ru/windows/fritzing/download> або з офіційного сайту <http://fritzing.org/home/>.

2.2 Хід роботи

1. Завантажити та запустити програмне середовище Fritzing. Якщо є у цьому потреба, завантажити запропоновані оновлення пакету.

2. Спроектувати роботу світлофору із таймером у даному середовищі.

3. Зробити звіт до лабораторної роботи.

2.2.1 Робота у Fritzing

На рис.2.1 показано відкрите вікно нового створеного файлу у режимі макетної плати. Інтерфейс програми є інтуїтивно зрозумілим, тому не будемо детально його описувати. Зазначимо лише, що для з'єднання елементів необхідно лише провести мишею із затисненою лівою клавішею від одного елементу до іншого. Таким чином буде створено лінію зв'язку елементів.

Усі основні можливості позначено на рис. 2.1.

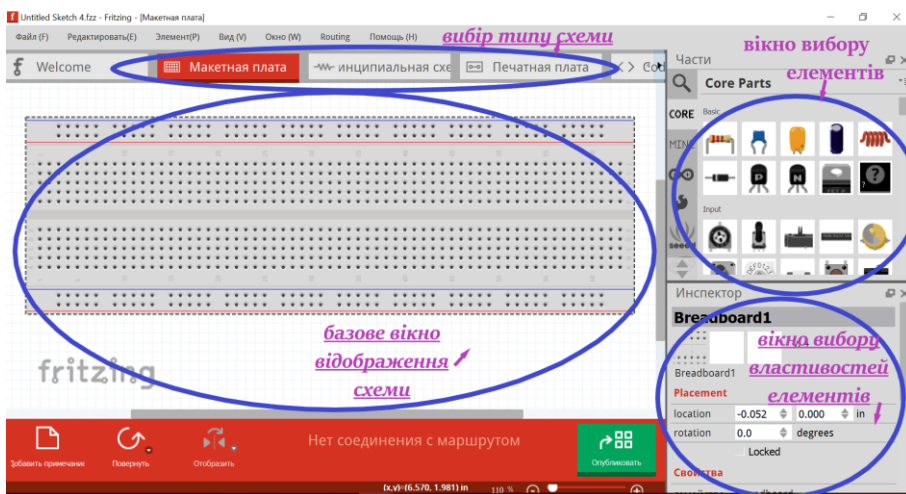


Рисунок 2.1 – Вікно нового створеного файлу у режимі макетної плати

2.2.2 Проектування світлофора із таймером

Модель світлофора із таймером складається з наступних елементів:

- плата Arduino UNO;
- макетна плата;
- резистори;
- семисегментний індикатор;
- з'єднувальні дроти input-input;
- світлодіоди.

Зупинимося на деяких з них більш детально.

Світлодіоди мають деяку особливість підключення. Вони мають дві ніжки: анод (+) і катод (-). До джерела живлення їх потрібно

обов'язково підключати через резистор. У нашому прикладі досить резистора на 220 Ом.

З'єднання елементів між собою

З огляду на описані технічні характеристики використовуваних для прикладу компонентів, опишемо, як саме вони будуть з'єднані між собою.

Світлодіоди червоного, жовтого і зеленого кольору під'єднаємо до плати Arduino Uno з використання макетної плати. Як вже говорилося вище, їх можна приєднувати тільки через резистори. У нашому випадку, достатньо резисторів 220 Ом.

Для реалізації у лабораторній роботі моделі світлофора необхідно з'єднати 4-й контакт плати Arduino Uno (GND) до лінії мінуса макетної плати. Далі з'єднаємо будь-який контакт з негативної лінії зв'язку з будь-яким контактом з контактів A1-J64, наприклад, з контактом A14. Резистор 220 Ом ставимо однією ніжкою на лінію 14, наприклад, B14, інший - на B18. Далі підключаємо зелений світлодіод в контакти макетної плати C18 і C19. Лінію 19 через контакт A19 з'єднуємо з будь-яким з цифрових пінів 2-13, наприклад, з 12. Далі, аналогічним чином з'єднуємо жовтий світлодіод з платою Arduino UNO через резистор і з піном 10, а червоний - з платою Arduino UNO через резистор і з піном 8.

Для реалізації описаної моделі у пакеті Fritzing, створюємо новий файл: Файл – New.

При цьому відкривається вікно, представлене на рис.2.2. В даному випадку у вікні присутній тільки макетна плата.

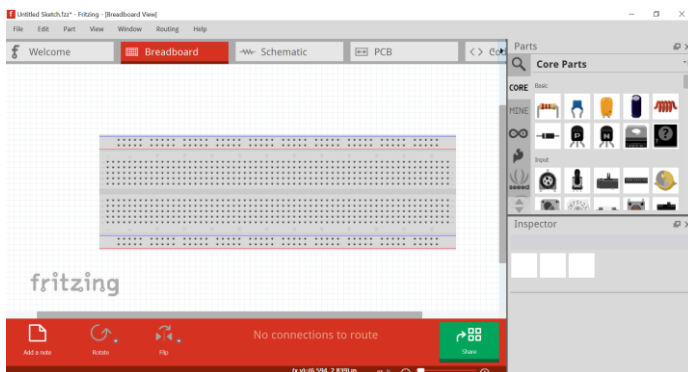


Рисунок 2.2 – Вигляд нового файла

Оскільки нам необхідно створити світлофор на базі Arduino Uno, то клацаємо лівою кавішею миші на значку Arduino в лівому ряду вікна вибору елементів і обираємо відповідну версію Arduino - Arduino Uno. Переносимо її в базове вікно монтажної схеми Макетна плата (рис. 2.3).

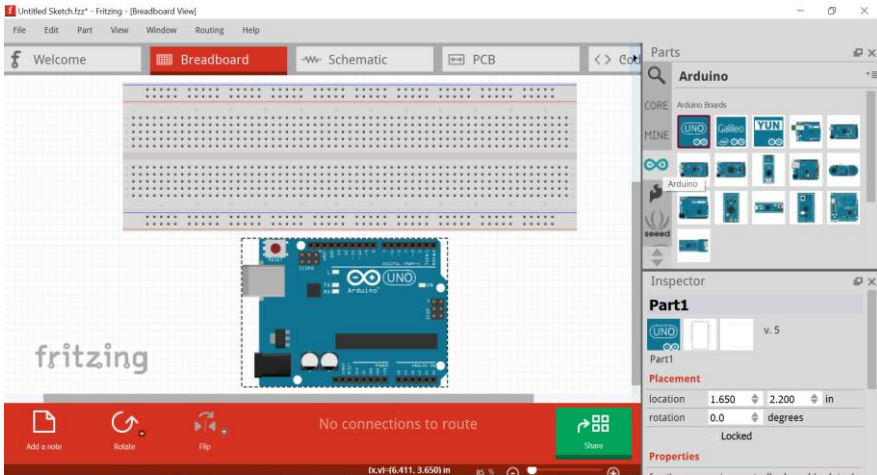


Рисунок 2.3 - Додавання плати Arduino Uno

Далі пошуком знаходимо світлодіоди (рис.2.4) і резистори опором 220 Ом і додаємо їх на монтажні схему, згідно з рисунком 2.5 і описаною вище моделі світлофора.

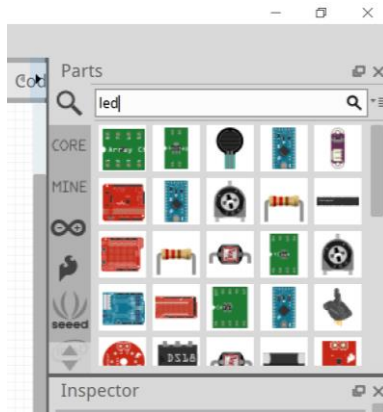


Рисунок 2.4 - Пошук світлодіодів

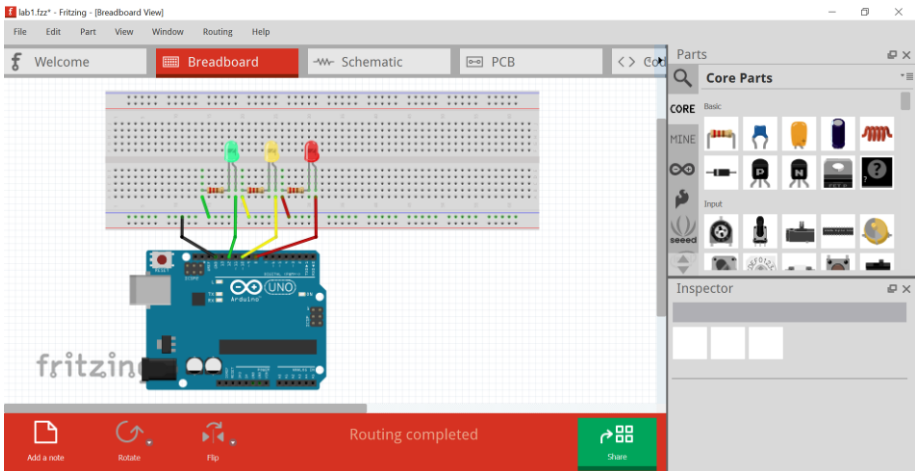


Рисунок 2.5 - Монтажна схема світлофора

Напис «Трасування завершено» сигналізує про правильне підключенні елементів. У разі помилки напис буде типу, представлений на рисунку 2.6.

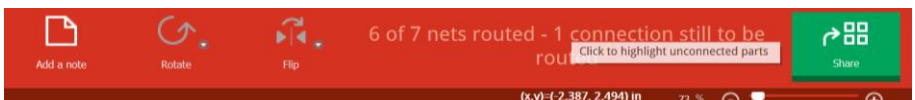


Рисунок 2.6 – Помилка трасування

Поміняти колір світлодіодів на відповідний можна в інспектор властивостей (рис.2.7).

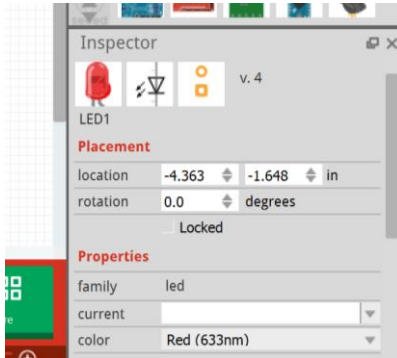


Рисунок 2.7– Inspector of LED properties

Для наочності відображення, можна поміняти колір проводів як показано на рис.2.8.

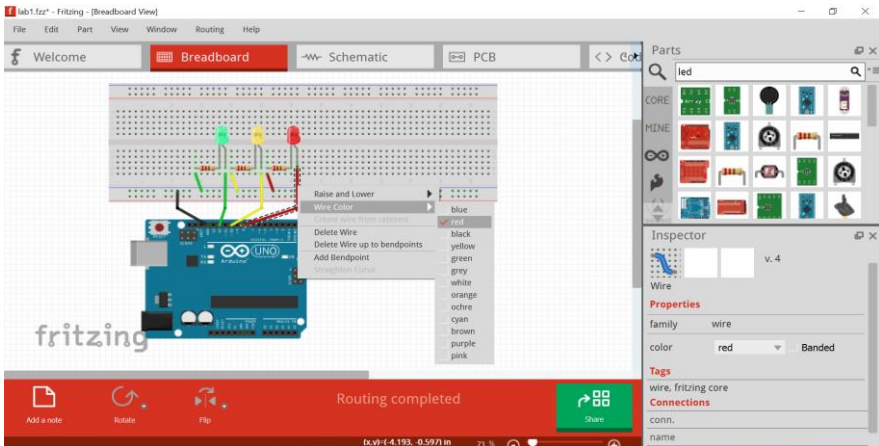


Рисунок 2.8 – Зміна кольорів дротів

Перейшовши на вкладку Принципова схема, побачимо відображення спроектованого світлофора у вигляді автоматично побудованої принципової схеми (рис.2.9).

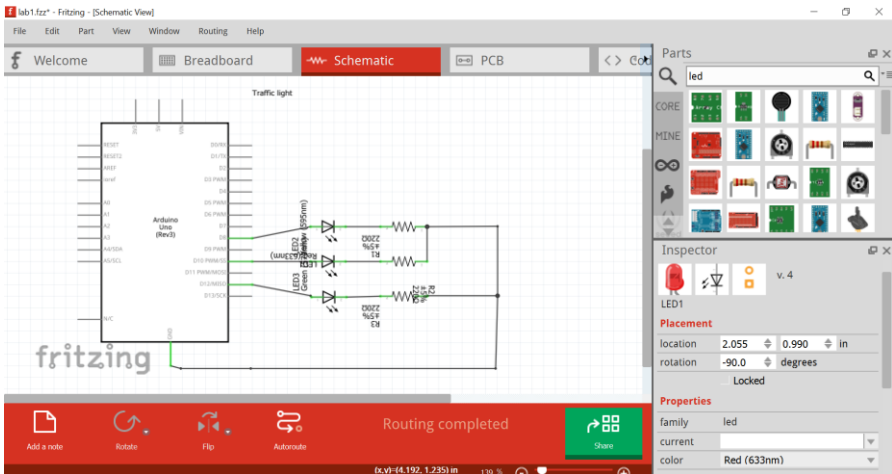


Рисунок 2.9 - Принципова схема світлофора

Наступним етапом зберімо реальну схему світлофора із зазначених на схемі елементів, згідно спроектованої під Fritzing схемою.

Зібране пристрій буде мати приблизно такий вигляд (рис.2.10).

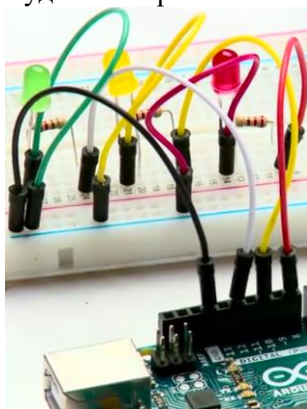


Рисунок 2.10 – Підключення з використанням реальних елементів

Напишімо програмний код, який реалізує алгоритм роботи світлофора. Скетч роботи світлофора описаний нижче в лістингу 2.1.

Лістинг 2.1 - Скетч роботи світлофора

```
int led_red = 8; // пін підключення 8
int led_yellow = 10; // пін підключення 10
int led_green = 12; // пін підключення 12
void setup() { pinMode(led_red, OUTPUT);
pinMode(led_yellow, OUTPUT);
pinMode(led_green, OUTPUT); }
void loop() { digitalWrite(led_red, HIGH);
delay(10000); // Чекаємо
digitalWrite(led_yellow, HIGH);
delay(2000); // Чекаємо
digitalWrite(led_green, HIGH);
digitalWrite(led_red, LOW);
digitalWrite(led_yellow, LOW);
delay(10000); // Чекаємо
digitalWrite(led_yellow, HIGH);
digitalWrite(led_green, LOW);
delay(2000); // Чекаємо
digitalWrite(led_yellow, LOW); }
```

Додаємо даний програмний код в вікно Code ПО Fritzing (рис. 2.11).

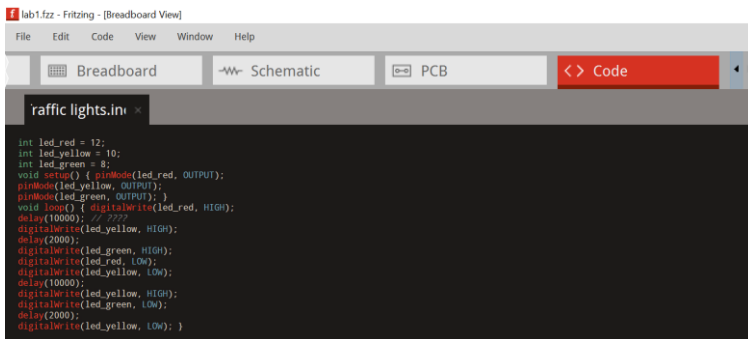


Рисунок 2.11 – Вікно Code ПО Fritzing

Далі необхідно прописати шлях до встановленого ПО Arduino IDE: Edit - Preferences - Code View (рис. 2.12).

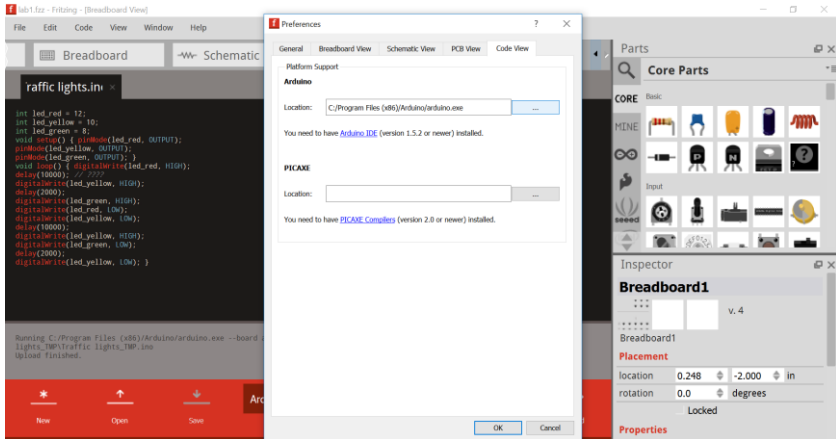


Рисунок 2.12 – Вказуємо шлях до встановленого ПО Arduino IDE

Далі в нижній частині вікна вибираємо порт, до якого підключена плата, і натискаємо Upload (рис.2.13).

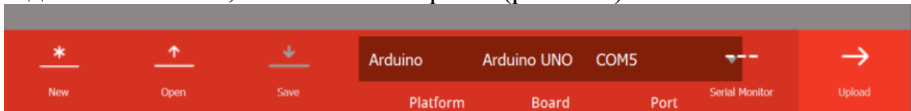


Рисунок 2.13 – Підключення плати

Далі робимо складнішим завдання і додаємо до моделі нашого світлофора індикацію «таймера зворотного відліку». Для цього додаємо на схему два елементи семисегментного дисплея індикації, як показано на рис 2.14.

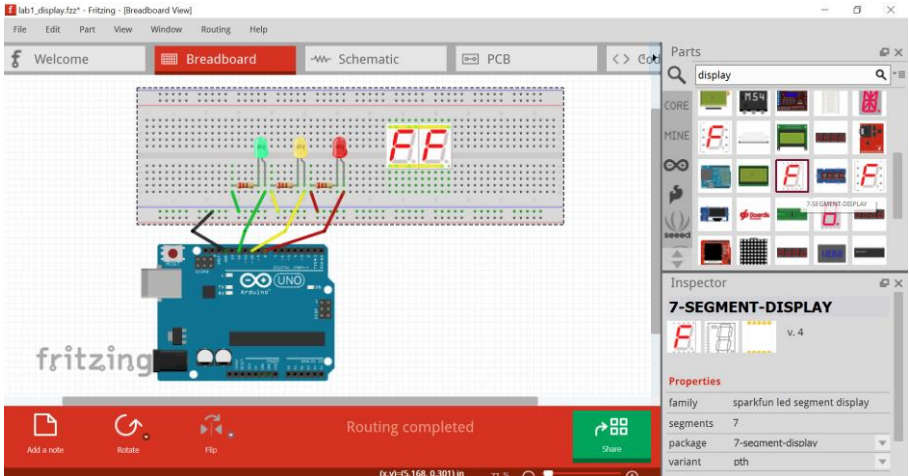


Рисунок 2.14 – Два елементи семисегментного дисплея індикації

Схематично семисегментний дисплей виглядає як на рисунку 2.15.

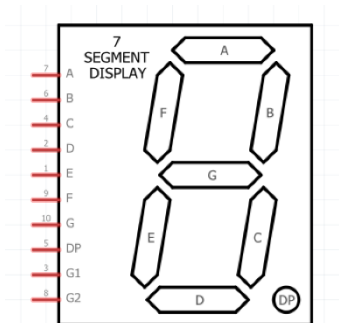


Рисунок 2.15 - Схема семисегментного дисплея

Виберемо семисегментний дисплей із загальним катодом D5611. Терморегулятори дисплея наочно представлені на рис.2.16.

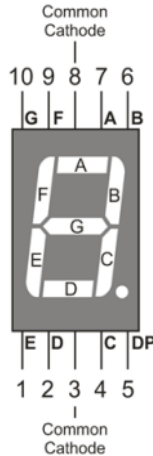


Рисунок 2.16 - Семисегментний дисплей із загальним катодом D5611

Для того, щоб зменшити кількість підключаються до Arduino Uno контактів (з 14 до 9) з'єднаємо два семисегментних дисплея між собою (контакт А з контактом А, контакт В з контактом В, контакт С з контактом С і т.д.). Перемикання управління кожним з дисплеїв окремо будемо регулювати подачею напруги з плати Arduino Uno на певний дисплей. Для цього підключимо контакт G1 першого семисегментного дисплея до 9 цифровому піну плати Arduino Uno, контакт G2 другого семисегментного дисплея до 11 цифровому піну плати Arduino Uno. При цьому підключати будемо через резистор, що відповідає обраному дисплею.

Далі підключаємо контакти А ... G семисегментного дисплея до цифрових пінам 2 - 7 та 13 плати Arduino Uno відповідно.

Будемо проектувати модель світлофора з таймером зворотного відліку часу в уже створеному файлі, що містить спроектовану модель світлофора без таймера зворотного відліку.

Для зручності підключення, переходимо на вкладку Schematic of Fritzing software і проектуємо описані вище з'єднання пристроїв, як показано на рис.2.17 нижче.

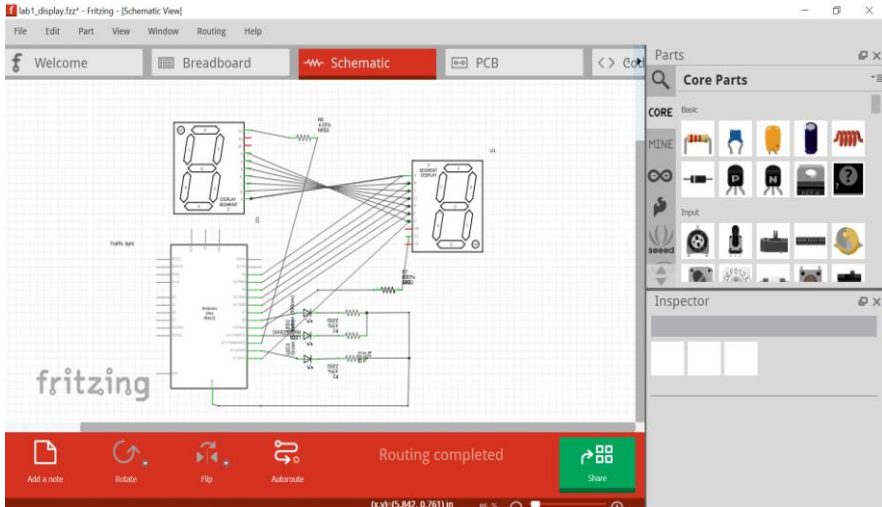


Рисунок 2.17 - Вкладка Schematic of Fritzing software

Далі переходимо на вкладку Breadboard. На ній автоматично з'явиться наступна схема (рис. 2.18).

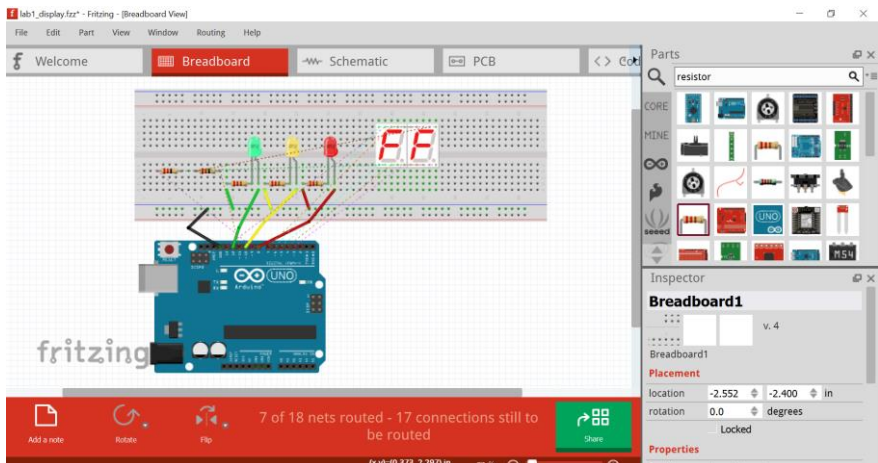


Рисунок 2.18 – Схема світлофора з таймером

Тонкими пунктирними лініями показані з'єднання, створені на вкладці Schematic. Щоб з'єднання відобразити правильно на даному

виді відображення (Breadboard) необхідно стати курсором миші і з затиснутою лівою клавішею провести уздовж лінії. Отримаємо наступне відображення (рис. 2.19).

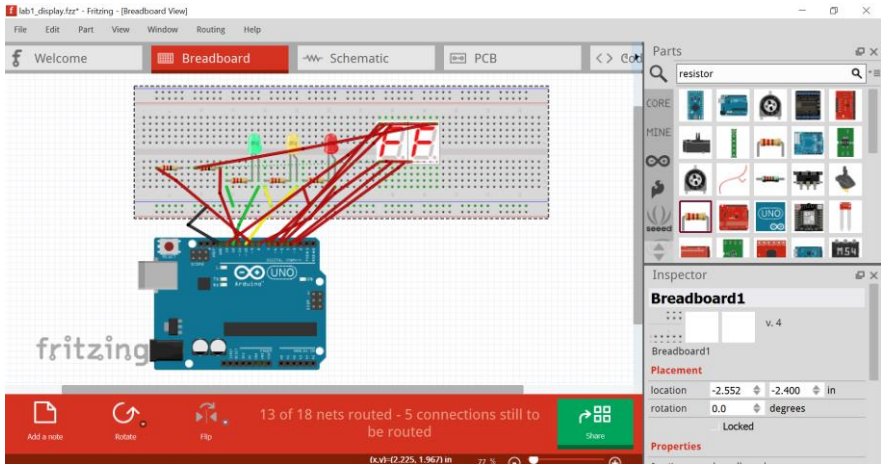


Рисунок 2.19 – Оновлена схема світлофора з таймером

Для наочності наведемо вище сконструйовану схему до виду представленому на наступному рисунку 2.20.

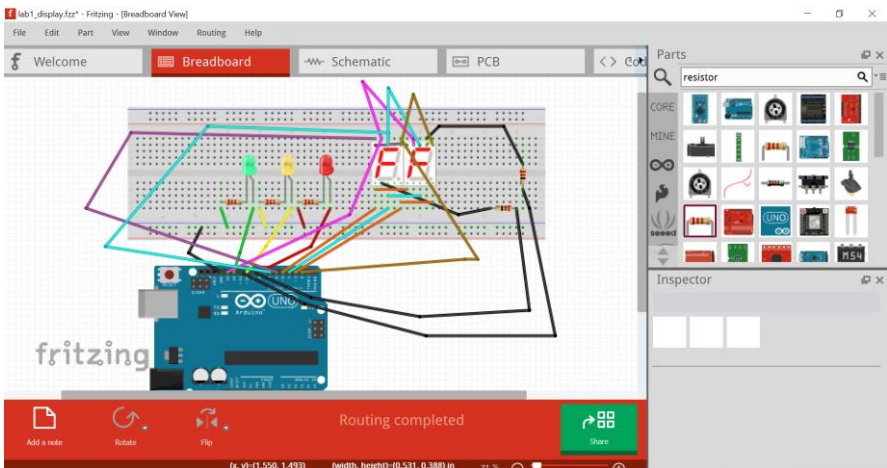


Рисунок 2.20 – Наочна схема світлофора з таймером

Далі перейдемо на вкладку Code і внесемо зміни в код, додавши опис роботи семисегментного дисплея.

Лістинг 2.2

```
int pin1 = 2;
int pin2 = 3;
int pin3 = 4;
int pin4 = 5;
int pin5 = 6;
int pin6 = 7;
int pin7 = 8;
int gnd1 = 9; // gnd1 is display 1's gnd
int gnd2 = 11; // gnd2 is display 2's gnd
int timer = 500; //A timer, to run the for loop 500
times, which turns out as 1 second.
int value; //The value, part of the FADING display

void setup () {
  pinMode (pin1, OUTPUT);
  pinMode (pin2, OUTPUT);
  pinMode (pin3, OUTPUT);
  pinMode (pin4, OUTPUT); //The following sets up all
of the pins for use.
  pinMode (pin5, OUTPUT);
  pinMode (pin6, OUTPUT);
  pinMode (pin7, OUTPUT);
  pinMode (gnd1, OUTPUT);
  pinMode (gnd2, OUTPUT);

  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 (0.5); // 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 (0.5);

}

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 (0.5); // Writes 19 to the display.
digitalWrite (pin1, B0);
digitalWrite (pin2, B0);
digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B1);
digitalWrite (pin6, B1);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (0.5);

}
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 (0.5); // 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 (0.5);

}
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 (0.5); // 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 (0.5);

}
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 (0.5); // Writes 16 to the display.
digitalWrite (pin1, B1);
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 (0.5);

}
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 (0.5); // Writes 15 to the display.
digitalWrite (pin1, B0);
digitalWrite (pin2, B1);
digitalWrite (pin3, B1);
digitalWrite (pin4, B1);
digitalWrite (pin5, B1);
digitalWrite (pin6, B0);
digitalWrite (pin7, B1);
digitalWrite (gnd1, B0);
digitalWrite (gnd2, B1);
delay (0.5);

```

```

}
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 (0.5); // Writes 14 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 (0.5);

}
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 (0.5); // Writes 13 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 (0.5);

}
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 (0.5); // Writes 12 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 (0.5);

}
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 (0.5); // Writes 11 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 (0.5);

}
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 (0.5); // Writes 10 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 (0.5);

}
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 (0.5); // Writes 09 to the display.
    digitalWrite (pin1, B0);
    digitalWrite (pin2, B0);
    digitalWrite (pin3, B1);
    digitalWrite (pin4, B1);
    digitalWrite (pin5, B1);
    digitalWrite (pin6, B1);
    digitalWrite (pin7, B1);
    digitalWrite (gnd1, B0);
    digitalWrite (gnd2, B1);
    delay (0.5);

}

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 (0.5); // Writes 08 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 (0.5);

}

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 (0.5); // Writes 07 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 (0.5);

}
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 (0.5); // Writes 06 to the display.
digitalWrite (pin1, B1);
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 (0.5);

}

```

```

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 (0.5); // 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 (0.5);

}
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 (0.5); // 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 (0.5);

}
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 (0.5); // 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 (0.5);

}
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 (0.5); // 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 (0.5);

}
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(0.5); // 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(0.5);

}
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(0.5); // 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(0.5);

}

}

void loop() {
    ...
    ...
}

```

2.3 Зміст звіту

Після виконання усіх етапів лабораторної роботи необхідно оформити звіт з наведеними власними результатами роботи.

Вміст звіту має таку структуру:

- титульний лист;
- мета роботи;
- скріншоти власних результатів самостійного виконання лабораторної роботи;
- відповіді на контрольні питання.

2.4 Контрольні питання

1. Що таке проектування системи?
2. Що таке макетування системи?
3. Опишіть коротко програми-аналоги Fritzing?
4. У чому їх переваги та недоліки?

ЛІТЕРАТУРА

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. Дистрибутив Arduino IDE з офіційного вебресурсу [Електронний ресурс] / Режим доступа: <http://www.arduino.cc/en/Main/Software>

3. Lee, E. A. and S. A. Seshia, 2015: Introduction to Embedded Systems. v2.0. URL: <http://LeeSeshia.org>.

4. 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>.

5. 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.

6. 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

7. Офіційного сайту з програмного забезпечення Fritzing [Електронний ресурс] / Режим доступа: <http://fritzing.org/home/>