КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСАШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

РОЗРОБКА ЛАБОРАТОРНОЇ РОБОТИ НА МІКРОКОНТРОЛЕРІ ARDUINO "ВИГОТОВЛЕННЯ ЕЛЕКТРОННОГО ЗАМКА"

Дипломна робота бакалавра
студента 4 курсу
Спеціальність: 123 «Комп'ютерна інженерія»
УСІКА Андрія

Науковий керівник:
кандидат фіз.-мат.наук БАУЖА Олександр,
доцент кафедри комп'ютерної інженерії

Рецензент
кандидат фіз.-мат. наук ГАВРИЛЬЧЕНКО Ірина
доцент кафедри нанофізики конденсованих середовищ Навчально-наукового інституту високих технологій

До захисту допускаю: Зав. кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри «__»_____2022р.,
протокол No___

КИЇВ 2022

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 41 с., 7 рис., 5 джерел, 2 додатки

В даній дипломній роботі мною було розроблено лабораторну роботу з виготовлення електронного замка на платформі Arduino.

Даний пристрій керує низкою датчиків та периферійних пристроїв в залежності від варіанту (RFID-датчик, сервопривод, кнопки, LCD-дисплей, п’єза, мембрана клавіатура).

Виконуючи дану лабораторну роботу, студенти навчаться працювати з платформою Arduino та керувати різними периферійними пристроями

Зміст

Вступ

Людство спостерігає стрімкий розвиток комп'ютерної електроніки з моменту її появи. Ще кілька десятиліть тому комп’ютери займали великі кімнати, а зараз майже у кожного вдома є свій персональний комп’ютер. Подальший розвиток дозволив ще більше зменшити розмір комп’ютера. Пристрої на основі мікроконтролерів (по суті мінікомпютери) зустрічаються все частіше та допомагають вирішити велику кількість завдань.

Мікроконтролер - це пристрій мікроелектроніки, який здатний виконувати основні функції комп’ютера (обробку та збереження інформації) в рамках однієї мікросхеми. Використання таких пристроїв значно зменшує споживання енергії та вартість побудованих на його базі пристроїв.

Виконання робіт на мікроконтролерах - це перший крок до розробки власних проектів, важливий досвід в роботі як з апаратним комп’ютерним забезпеченням, так і з програмним. Це дозволяє зрозуміти основи роботи різних пристроїв, та навчитися виконувати на них різні задачі.

1. Мікроконтролер

1.1. Поняття про мікроконтролер

Мікроконтролер - це виконаний у вигляді мікросхеми спеціалізований комп'ютер, що складається з таких частин:

- мікропроцесор
- оперативна та постійна пам'ять для збереження виконуваного коду програм і даних
- порти вводу-виводу
- блоки зі спеціальними функціями (лічильники, компаратори, АЦП та інші) .

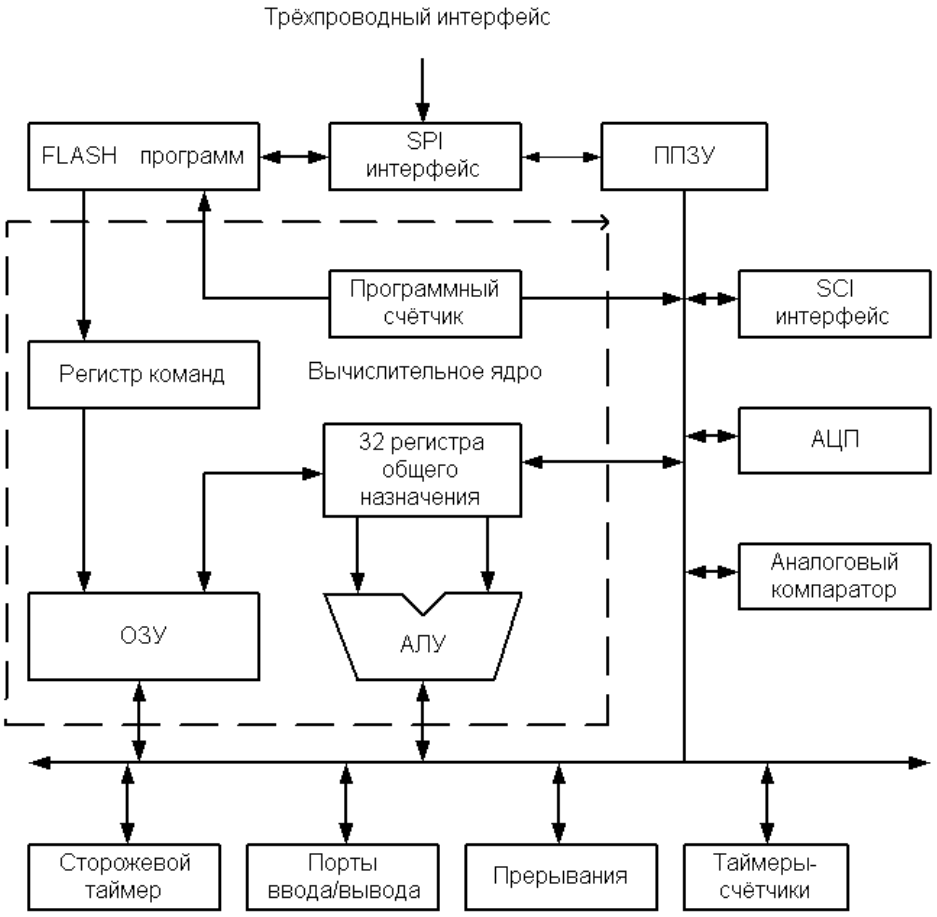


Рис.1. Спрощена схема мікроконтролера

Мікроконтролер використовується для керування електронними пристроями. По суті, це — однокристальний комп'ютер, здатний виконувати прості завдання. Використання однієї мікросхеми значно знижує розміри приладів, енергоспоживання і вартість пристроїв, що побудовані на базі мікроконтролерів.

Мікроконтролери зустрічаються в багатьох сучасних приладах. Це можуть бути телефони, пральні машини та інші. Вони відповідають за роботу двигунів і систем гальмування сучасних автомобілів, з допомогою мікроконтролерів створюються системи контролю і системи збору інформації. На основі цих пристроїв проектують та створюють вимірювальні прилади, системи керування об'єктами та процесами, вони є основою охоронних, протипожежних систем, домофонів, сигналізацій тощо. Більшість процесорів, що випускаються у світі — мікроконтролери.

Коли проектують мікроконтролери, то доводиться дотримуватись балансу між розмірами і вартістю з одного боку і гнучкістю та продуктивністю з іншого. Для мікроконтролерів різного призначення оптимальне співвідношення цих параметрів може розрізнятися в дуже великому діапазоні. Тому існує величезна кількість різних типів мікроконтролерів, які можуть відрізнятися архітектурою процесорного модуля або розміром і типом вбудованої пам'яті, набором периферійних пристроїв або типом корпусу.

В той час, як 8-розрядні процесори загального призначення вже повністю витіснені більш продуктивними моделями, 8-розрядні мікроконтролери до цього часу продовжують широко використовуватися в різних галузях. Це можна пояснити тим, багато таких застосувань не потребують високої продуктивності, але для них важлива низька вартість. В той же час, зараз ми маємо мікроконтролери, що характеризуються більшими обчислювальними можливостями, такі як, наприклад, цифрові сигнальні процесори.

Обмеження за ціною і енергоспоживанням стримують також зростання тактової частоти контролерів. Більшість виробників зараз прагнуть забезпечити роботу своїх виробів на високих частотах. Але вони, також,

надають замовникам вибір, коли випускають модифікації, що розраховані на різні частоти і напругу живлення. Багато моделей мікроконтролерів використовують статичну пам'ять для ОЗП і внутрішніх регістрів. Це допомагає контролеру працювати на менших частотах і при цьому не втрачати дані навіть при повній зупинці тактового генератора. Часто можуть бути передбачені різні режими енергозбереження, при яких можливе відключення частини периферійних пристроїв і обчислювальний модуль.

Окрім ОЗП, мікроконтролер часто має вбудовану незалежну пам'ять для того, щоб зберігати програми і дані. У багатьох контролерах взагалі відсутні шини для підключення зовнішньої пам'яті. Найдешевші типи пам'яті дозволяють лише одноразовий запис. Такі пристрої найбільше підходять для масового виробництва тоді, коли програма контролера не буде оновлюватися. Більш досконалі модифікації контролерів дозволяють багато разів перезаписувати незалежну пам'ять. Часто мікроконтролери відрізняються від процесорів загального призначення використанням гарвардської архітектури.[1]

1.2. Історія мікроконтролерів

З процесом появи і розвитку однокристальних мікроЕОМ пов'язаний початок масового застосування комп'ютерної автоматизації в управлінні. Мабуть, це і визначило термін "контролер" (англ. controller - регулятор, керуючий пристрій).

У 1971 інженерам М. Кочрену і Г. Буну, співробітникам американської Texas Instruments був виданий, перший патент на однокристальну мікроЕОМ. Саме вони запропонували одному кристалі розмістити не тільки процесор, а ще і пам'ять із пристроями вводу-виводу.

У 1976 американською фірмою Intel був випущений мікроконтролер i8048. У 1978 році фірма Motorola випустила свій перший мікроконтролер

MC6801, який був сумісний за системою команд з мікропроцесором MC6800 що був випущений раніше. Наступний мікроконтролер i8051 Intel випускає в 1980 році . Успіх на ринку цьому мікроконтролеру забезпечили не тільки вдалий набір периферійних пристроїв, а ще і можливість гнучкого вибору зовнішньої або внутрішньої програмної пам'яті та невисока вартість. Мікроконтролер i8051 технологічно став для того часу досить складним виробом тому, що у його кристалі використали 128 тис. транзисторів, а це аж в чотири рази перевищує кількість транзисторів у 16-розрядному мікропроцесорі i8086.

У СРСР розробляли оригінальні мікроконтролери, також випускали клони найбільш вдалих зарубіжних зразків . У 1979 році в НДІ ТТ розробили однокристальну 16-розрядну ЕОМ К1801ВЕ1, мікроархітектура якої отримала назву "Електроніка НЦ".

У 2013-ому році вже існувало більше двохсот різних модифікацій мікроконтролерів, що були сумісними з i8051. Вони випускалися двома десятками компаній. Також була відома велика кількість мікроконтролерів інших типів. Найбільш популярні для виробників та користувачів на той час вважаються 8-бітні, 16-бітні та 32-бітові мікроконтролери PIC фірми Microchip Technology, мікроконтролери AVR фірми Atmel (з 2016 року виробляються фірмою Microchip), 16-бітні MSP430 фірми TI. Користувалися попитом також 32- архітектури ARM, які розробляла фірма ARM Limited що продавала ліцензії іншим фірмам для їхнього виробництва. Всі ці модифікації мікроконтролерів були популярні і на території України. Але, незважаючи на це, у 2009-му році світовий рейтинг обсягу продаж, за даними Gartner Group, виглядає по іншому: на першому місці беззаперечно і з великим відривом від інших була Renesas Electronics, на другому - Freescale, на третьому - Samsung, потім Microchip і TI і всі інші виробники [2]

1.3. AVR Мікроконтролер

Мікроконтролер AVR містить: швидкий RISC-процесор, два типи енергонезалежної пам'яті (Flash-пам'ять програм та пам'ять даних EEPROM), оперативну пам'ять RAM, порти вводу/виводу та різні периферійні інтерфейсні схеми.

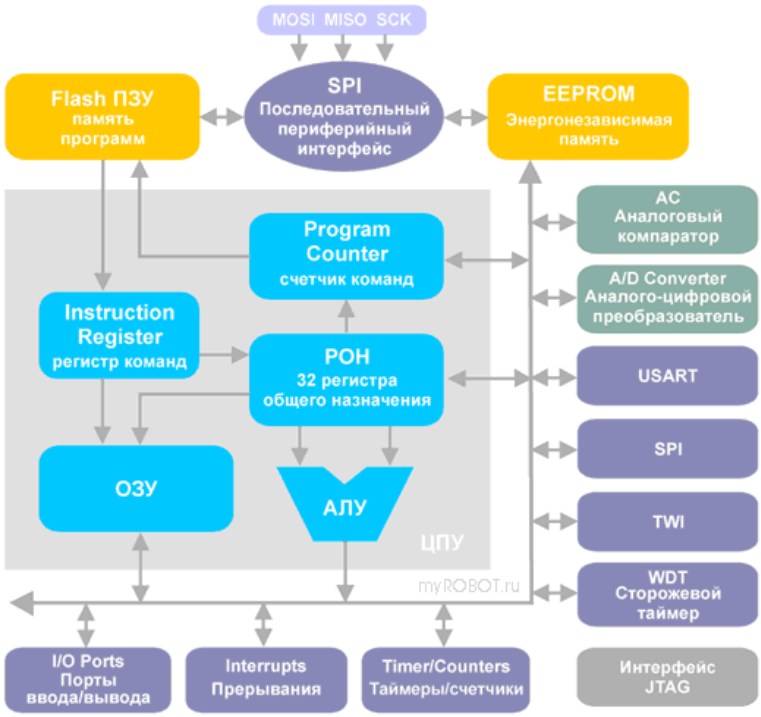


Рис.2 узагальнена схема avr мікроконтролера

Версії контролерів:

- AT (mega/tiny)xxx – базова версія.
- ATxxx**L** — версії контролерів, які працюють при зниженій (Low) напрузі живлення (2,7 В).
- ATxxx**V** — версії контролерів, які працюють при низькій напрузі живлення (1,8 В).
- ATxxx**P** — версії що мають мале енергоспоживання (до 100 нА в режимі Power-down), застосована технологія *picoPower* (анонсовані у липні 2007), по-вивідно а також є функціонально сумісними з попередніми версіями.
- ATxxx**A** — зменшений струм споживання, перекривається увесь діапазон тактових частот і напруг живлення двох попередніх версій (також, у деяких моделях, додані нові можливості та нові регістри, але збережена повна

сумісність із попередніми версіями). Мікроконтролери «А» і «не-А» зазвичай мають однакову сигнатуру і це викликає труднощі, оскільки Fuse біти відрізняються.[4,5]

1.3.1. Процесор

Серце мікроконтролерів AVR - це 8-бітове мікропроцесорне ядро або центральний процесорний пристрій (ЦПП), який побудували на принципах RISC-архітектури. Основа цього блоку - арифметико-логічний пристрій (АЛП). За системним тактовим сигналом із пам'яті програм відповідно до вмісту лічильника команд (Program Counter - PC) вибирається чергова команда і виконується АЛП. Коли вибирається команда з пам'яті програм, то відбувається виконання попередньої вибраної команди, що дозволяє досягти швидкодії 1 MIPS на 1 МГц.

АЛУ підключено до регістрів загального призначення РЗП (General Purpose Registers – GPR). Всього є 32 регістри загального призначення вони мають байтовий формат, тобто кожен з них складається з восьми біт. Регістри знаходяться на початку адресного простору оперативної пам'яті, але фізично не є її частиною. Це дозволяє звертатися до них двома способами (як до регістрів і як до пам'яті). Таке рішення часто підвищує ефективність роботи та продуктивність мікроконтролера і є важливою особливістю AVR

Відмінність між регістрами і оперативною пам'яттю у тому, що з регістрів можна робити будь-які операції (арифметичні, логічні, бітові), а за допомогою оперативної пам'яті можна лише записувати дані з регістрів.[5]

1.3.2. Пам’ять

Як писалось раніше, особливістю мікроконтролерів AVR є те, що у них реалізована Гарвардська архітектура, у якій розділені, як адресні простори пам'яті програм і пам'яті даних, так і шини доступу до них. Також розташована у своєму адресному просторі кожна з областей пам'яті даних (оперативна пам'ять та EEPROM). Пам'ять програм забезпечує зберігання послідовності команд, управляючих функціонуванням мікроконтролера, і має 16-ти бітну організацію. Всі AVR мають Flash-пам'ять програм, яка може бути

різного розміру – від 1 до 256 КБайт. Головною перевагою цієї програми є те, що вона побудована на принципі електричної перепрограмованості, а отже дозволяє багато разів стирати та записувати інформацію. Програма може бути занесена до Flash-пам'яті AVR як за допомогою звичайного програматора, так і SPI-інтерфейсом, у тому числі і на зібраній платі. Такою можливістю, що забезпечує внутрішньосхемне програмування (функція ISP) через комунікаційний інтерфейс SPI обладнані всі мікроконтролери AVR, крім Tiny11 і Tiny28.

Також всі мікроконтролери сімейства Mega мають можливість самостійної зміни вмісту пам'яті програм, тобто самопрограмування. Це дає можливість створювати на їх основі гнучкі системи, алгоритм роботи яких може бути зміненим мікроконтролером самостійно, в залежності від внутрішніх умов чи зовнішніх подій.

Гарантована кількість циклів перезапису Flash-пам'яті у мікроконтролерів AVR другого покоління становить щонайменше 10 тис. циклів при типовому значенні 100 тис. циклів. (В офіційній технічній документації Atmel Corp. вказується значення 10 тис. циклів.)

Пам'ять складається з трьох частин, а саме це регістрова пам'ять, оперативна пам'ять (ОЗП - оперативний запам'ятовуючий пристрій або RAM) та енергонезалежна пам'ять (ЕСППЗУ або EEPROM).

Регістрова пам'ять складається з 32-х регістрів загального призначення (РЗП або GPR), що об'єднані у файл, та службових регістрів введення/виводу (РВВ). Вони розташовуються в адресному просторі ОЗП, але не відносяться до його частин.

В області регістрів введення/виводу також розташовуються різні службові регістри, наприклад, регістри управління мікроконтролером, регістри стану тощо, а також регістри управління периферійними пристроями, які є складовою частиною мікроконтролера. Отже процес управління мікроконтролером забезпечується управлінням цими регістрами.

Щоб інформація, яка може змінюватися в процесі роботи мікроконтролерної системи, зберігалася довго, використовується пам'ять EEPROM. У всіх AVR є блок енергонезалежної пам'яті даних EEPROM, який електрично перезаписується, від 64 Байт до 4 КБайт. Цей тип пам'яті, доступний програмі мікроконтролера безпосередньо під час виконання. Він зручний для зберігання проміжної інформації, різних констант, коефіцієнтів, серійних номерів, ключів та інших даних. EEPROM може бути завантажена за допомогою звичайного програматора або з зовні через інтерфейс SPI. Число циклів стирання/запис може бути не менше 100 тис.

Внутрішня статична пам'ять Static RAM (SRAM) використовується для оперативного зберігання даних і має байтовий формат.

У різних чіпів розмір оперативної пам'яті може змінюватись від 64 Байт до 4 КБайт. Хоч число циклів читання і запису в RAM необмежене, але при відключенні напруги живлення вся інформація втрачається.

Можлива організація підключення зовнішнього статичного ОЗП для деяких мікроконтролерів.

1.3.3 Периферія

Периферія мікроконтролерів AVR складається з портів (від 3 до 48 ліній введення та виведення) також з підтримки зовнішніх переривань, таймерів-лічильників, сторожових таймерів, аналогових компараторів, 10-розрядних 8-канальних АЦП, інтерфейсів UART, JTAG і SPI, пристроїв живлення, широтно-імпульсних модуляторів.

Порти введення/виводу (I/O)

Порти вводу/виводу AVR мають від 3 до 53 незалежних ліній "вхід/вихід". Кожна лінія порту може бути запрограмована на вхід або вихід. Струмову здатність навантаження 20 мА на лінію порту при максимальному значенні 40 мА забезпечують потужні вихідні драйвери, що дає можливість користувачам підключати до мікроконтролера світлодіоди і біполярні

транзистори. Загальне струмове навантаження на всі лінії одного порту не повинно бути більшим за 80 мА (всі значення для напруги живлення 5 В).

Архітектурною особливістю побудови портів введення/виводу у AVR є те, що для кожного фізичного виведення (піна) існує 3 біти контролю/управління, а не 2, як у найбільш поширених 8-розрядних мікроконтролерів (Intel, Microchip, Motorola і т.д.). Це підвищує швидкість роботи мікроконтролера при роботі із зовнішніми пристроями, особливо в умовах зовнішніх електричних перешкод та відміняє потребу в копії вмісту порту в пам'яті для безпеки зберігання.

Переривання (INTERRUPTS)

Система переривань є однією з найважливіших систем мікроконтролера. Усі мікроконтролери AVR мають багаторівневу систему переривань. Переривання припиняє нормальний хід програми для виконання пріоритетного завдання. Це завдання визначається внутрішньою або зовнішньою подією.

Для кожної такої події розробляється підпрограма обробки запиту на переривання (підпрограма переривання), яка розміщується у пам'яті програм.

Якщо подія викликає переривання, то мікроконтролер зберігає вміст лічильника команд, також перериває виконання центральним процесором поточної програми і потім переходить до виконання підпрограми обробки переривання.

Відновлення попередньо збереженого лічильника команд здійснюється після виконання підпрограми переривання, потім процесор повертається до виконання перерваної програми.

Можна встановити пріоритет для кожної події. Пріоритет означає, що підпрограма переривання може бути перервана іншою подією тільки за умови, якщо вона має більш високий пріоритет, ніж поточна. Якщо пріоритет нижчий,

то центральний процесор перейде до обробки нової події лише після закінчення попередньої обробки.[5]

Таймери/лічильники (TIMER/COUNTERS)

Мікроконтролери AVR мають у своєму складі від 1 до 4 таймерів/лічильників з розрядністю 8 або 16 біт, які можуть працювати як лічильники зовнішніх подій і як таймери від внутрішнього джерела тактової частоти.

Ці прилади можна використовувати як для точного формування часових інтервалів так і для підрахунку імпульсів на виходах мікроконтролера, а також для формування послідовності імпульсів, тактування приймача послідовного каналу зв'язку. Таймер/лічильник може бути широтно-імпульсним модулятором у режимі ШІМ (PWM) і використовуватись для генерування сигналу з програмованими шпаруватістю і частотою. Таймери/лічильники можуть виробляти запити переривань, якщо перемикають процесор, що їх обслуговує по подіям і звільняють його від необхідності періодично опитувати стан таймерів. Таймери/лічильники є одним із найважливіших елементів мікроконтролерів, оскільки в основному мікроконтролери застосовуються у системах реального часу.[5]

Сторожовий таймер (WDT)

Сторожовий таймер (WatchDog Timer) використовується для того, щоб запобігти катастрофічним наслідкам від випадкових збоїв програми. У нього є власний RC-генератор, який використовує для роботи частоту 1 МГц. Значення 1 МГц є наближеним так само, як і для основного внутрішнього RC-генератора, і залежить від величини напруги живлення мікроконтролера і від температури.

Ідея використання сторожового таймера дуже проста, вона полягає в регулярному його скиданні. Цей процес відбувається під управлінням програми або зовнішнього впливу. Він починається до того, як закінчиться період часу і в тому випадку, якщо не відбудеться скидання процесу. Команда скидання сторожового таймера повинна регулярно виконуватися, якщо програма працює правильно. Це захищає процесор від скидання. Якщо ж мікропроцесор випадково вийшов за межі програми (наприклад, від сильної перешкоди по ланцюгу живлення) або зациклився на будь-якій ділянці програми, команда скидання сторожового таймера швидше за все не буде виконана протягом достатнього часу. що приводить систему в робочий стан.[5]

Тактовий генератор

Для синхронізації всіх вузлів мікроконтролера тактовий генератор виробляє імпульси . Внутрішній тактовий генератор AVR запускається від декількох різних джерел опорної частоти а саме від зовнішнього генератора, зовнішнього кварцового резонатора, внутрішнього або зовнішнього RC-ланцюжка. Мінімальна допустима частота нічим не обмежена аж до покрокового режиму. Максимальна робоча частота буде визначатися конкретним типом мікроконтролера і вказана Atmel у характеристиках мікроконтролера, хоча будь-який AVR-мікроконтролер із робочою частотою, наприклад, 10 МГц при кімнатній температурі легко може бути “розігнаний” до 12 МГц і вище.[5]

1.2.4 Живлення

AVR працюють при напругах живлення від 1,8 до 6,0 Вольт. Струм споживання в активному режимі залежить від величини напруги живлення та частоти, на якій працює мікроконтролер і становить менше 1 мА для 500 кГц, 5...6 мА для 5 МГц та 8...9 мА для частоти 12 МГц.

AVR можуть бути переведені програмним шляхом в один із трьох режимів зниженого енергоспоживання.

Режим холостого ходу (IDLE) це режим,що припиняє роботу лише процесора і фіксує вміст пам'яті даних, а внутрішній генератор синхросигналів, таймери, система переривань та сторожовий таймер продовжують функціонувати. Струм споживання не перевищує 2,5 мА на частоті 12 МГц.

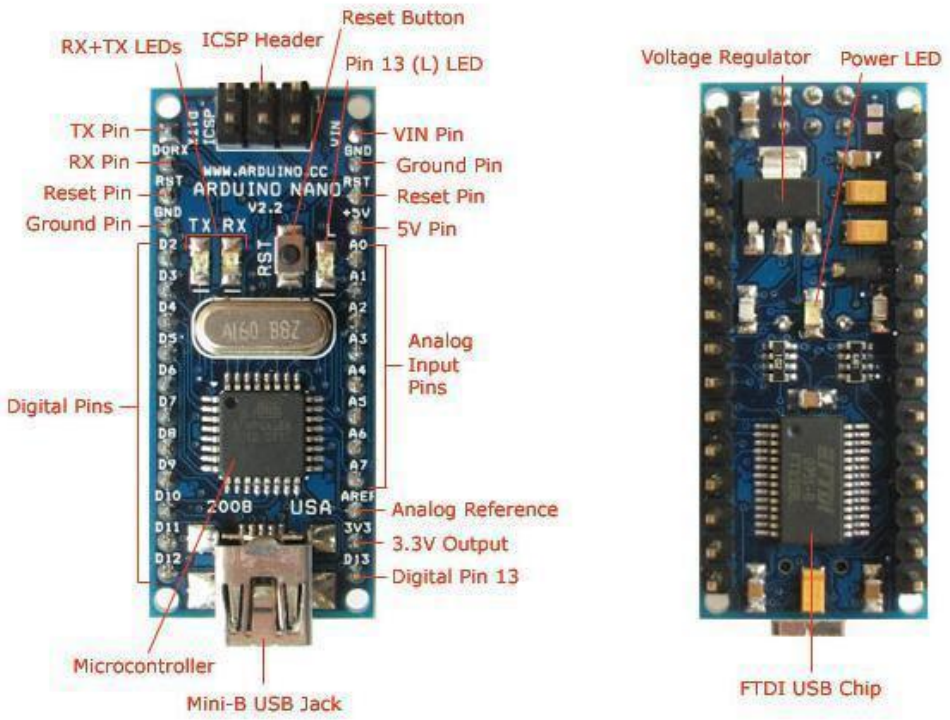
Стоповий режим (POWER DOWN) зберігає вміст реєстрового файлу, але зупиняє внутрішній генератор синхросигналів, і, отже, зупиняються всі функції, доки надійде сигнал зовнішнього переривання чи апаратного скидання. При включеному сторожовому таймері струм споживання цьому режимі становить близько 80 мкА, а при вимкненому - менше 1 мкА. (Всі наведені значення справедливі для напруги живлення 5В).

Економний режим (POWER SAVE) це режим при якому продовжує працювати тільки генератор таймера, що забезпечує безпеку тимчасової бази. Всі інші функції відключені. [5]

2. Arduino Nano та периферійні пристрої

Arduino Nano – це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328 (Arduino Nano 3.0) або ATmega 168 (Arduino Nano 2.x), який представляє собою макетну плату.

Короткі характеристики: робоча напруга 5 В, 14 цифрових входи-виходи (з яких 6 можуть використовуватися як ШИМ-виходи), 8 аналогових входів-виходів, максимальний струм одного виводу 40 мА, тактова частота 16 МГц. [3]



2.1. Доцільність проекта на Arduino

Оскільки мікросхеми Arduino досить легкі в освоєнні, та написанні скриптів, то їх доцільно використовувати для вивчення програмування на мікроконтролерах. Готовність до використання (Ready to use) є, безперечно, головним плюсом, тому,що на мікросхемах вже є і мікроконтролер, і

програматор, також є інтерфейси для підключення пристроїв та програмні бібліотеки.

Безліч периферії, яка здатна працювати з даною мікросхемою, дає можливість створювати велику кількість пристроїв різного призначення методом комбінації периферійних пристроїв та написання коду для роботи системи з цих пристроїв та мікросхеми.

2.2. RFID модуль RC522

Радіочастотна ідентифікація (RFID) – це технологія безконтактної ідентифікації об'єктів за допомогою радіочастотного каналу зв'язку. Ідентифікація об'єктів проводиться за унікальним ідентифікатором, який має кожна електронна мітка. Зчитувач випромінює електромагнітні хвилі певної частоти. Мітки надсилають у відповідь інформацію – ідентифікаційний номер, дані пам'яті та ін.

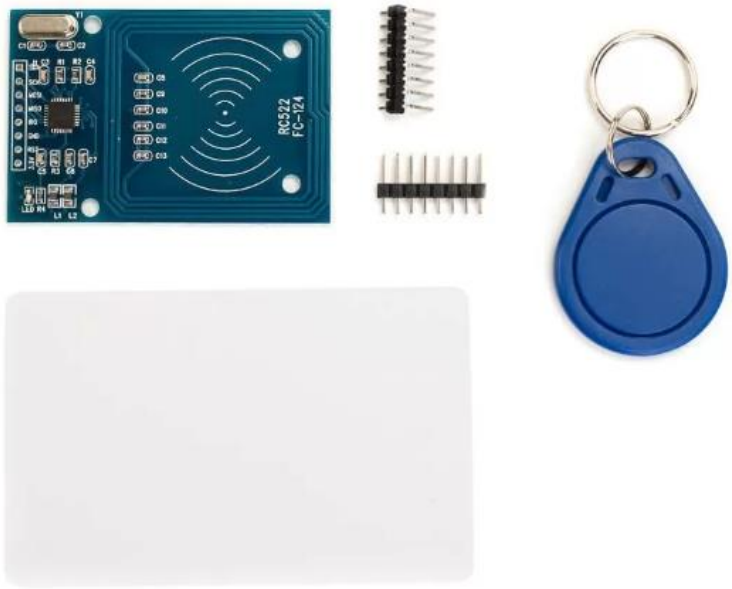


Рис. 3. RC522

В розробці лабораторної роботи буде використано RFID модуль RC522 зображений на рисунку 3.

Технічні характеристики RFID-модуля RC522:

Напруга живлення: 3.3V;

Споживаний струм: 13-26mA;

Робоча частота: 13.56 МГц;

Діапазон зчитування: 0 – 60 мм;

Інтерфейс: SPI;

Швидкість передачі: максимальна 10 Мбіт/с;

Розмір: 40мм х 60мм;

Мікросхема MFRC522 підтримує інтерфейси SPI, UART та I2C (див. рис.4.). Вибір інтерфейсу здійснюється установкою логічних рівнів певних виходах мікросхеми. На цьому модулі вибрано інтерфейс SPI.

Рис. 4. Призначення виводів.

Призначення виводів інтерфейсу SPI:

SDA – вибір вводу;

SCK-сигнал синхронізації;

MOSI – передача від master до slave;

MISO – передача від slave до master;

RST - вивід для скидання;

IRQ - вивід переривання;

GND – земля;

Vcc-живлення 3.3 В.

Сигнал скидання RST – це сигнал, що надходить з цифрового виходу

контролера. При надходженні сигналу LOW відбувається перезавантаження зчитувача. Також контролер установкою на RST низького рівня повідомляє, що знаходиться в режимі сну, для виведення модуля з сну необхідно подати на даний вивід сигнал HIGH.

2.3. I2C LCD дисплей

Класичний LCD дисплей з зеленим фоном та білими буквами. Розмір 1602 (2 рядка, 16 стовпчиків).

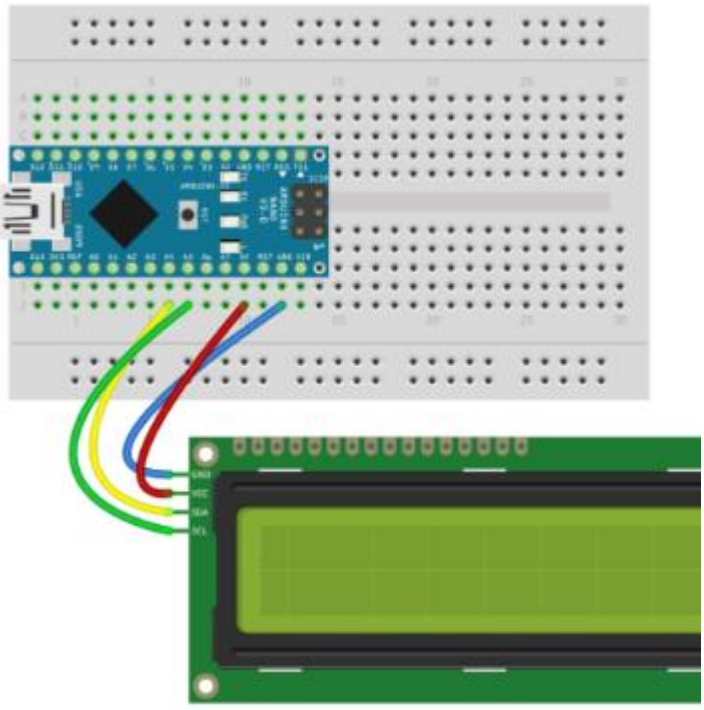


Рис. 5. Схема підключення до arduino.

Дисплей вимагає для підключення шести цифрових пінів, але в даному випадку використовується з перехідником на шину I2C на базі PCF8574, що значно спрощує підключення і економить піни.

На перехіднику також розпаяний потенціометр налаштування контрастності (синій паралелепіпед з крутилкою під хрестову викрутку). Залежно від напруги живлення потрібно вручну налаштувати контрастність.

2.4. Сервопривід SG90

SG90 - це аналоговий серводвигун з обертальним моментом 1.8 кг.см при напрузі 4.8 В і діапазоном обертання 180°, ідеально підходить для робототехніки.

Особливості:

- Діапазон обертання: 180 °;
- Напруга живлення: 4.8 ... 6 В;
- Обертальний момент: 1.8 кг.см при 4.8 В;
- Швидкість обертання: 60 ° за 0.12 с при 4.8 В;
- внутрішній інтерфейс: аналоговий;
- Напрямок: CCW (проти годинникової стрілки);
- матеріал шестерень: нейлон;
- матеріал корпусу: пластик;
- довжина проводів: 25 см;
- Розміри: 23×12.2×29 мм;
- Вага: 9 г.

2.5. Мембрана клавіатура

Клавіатура виконана у вигляді матриці 4x4, кожна кнопка є областю повітряного проміжку між двома діелектричними шарами з нанесеним на них струмопровідним покриттям. Доріжки струмопровідного покриття одного шару нанесені горизонтально (виводи 1-4), а іншого вертикально (виводи 5-8). Натискання на кнопку призводить до з'єднання доріжки одного шару з доріжкою іншого, і як наслідок, замикання одного з виводів 1-4 з одним з виводів 5-8.

Рис. 6. Клавіатура

2.6. П’єзокристальний випромінювач

Являє собою найпростіший генератор звуку (пищалку). Має свої полярності. використовується як індикатор у багатьох схемах на Arduino. За допомогою функцій можна регулювати звук який вона видає.

3. Створення електронного замка

Завданням для студентів буде створення електронного замка з підключенням різних периферійних пристроїв. Замок повинен мати можливість виводу на дисплей інформації про статус замка після спроби авторизації (відкритий чи закритий) та подачу відповідного звукового сигналу через випромінювач звуку, індикатор закриття дверей, можливість відкривати двері зсередини, механізм відкриття дверей на серводвигуні. Програмування здійснюється в середовищі розробки Arduino IDE на мові C++.

3.1. Варіант 1.

Пристрій вводу - мембранна клавіатура.

Індикатор закриття дверей - геркон.

Пристрої виведення - LCD 1602 та п'єзокристальний випромінювач.

Механізм відкриття замка - серводвигун SG90.

Кнопка для відкриття дверей зсередини.

На рисунку 6 зображений приклад створеної мною схеми для реалізації цього завдання.

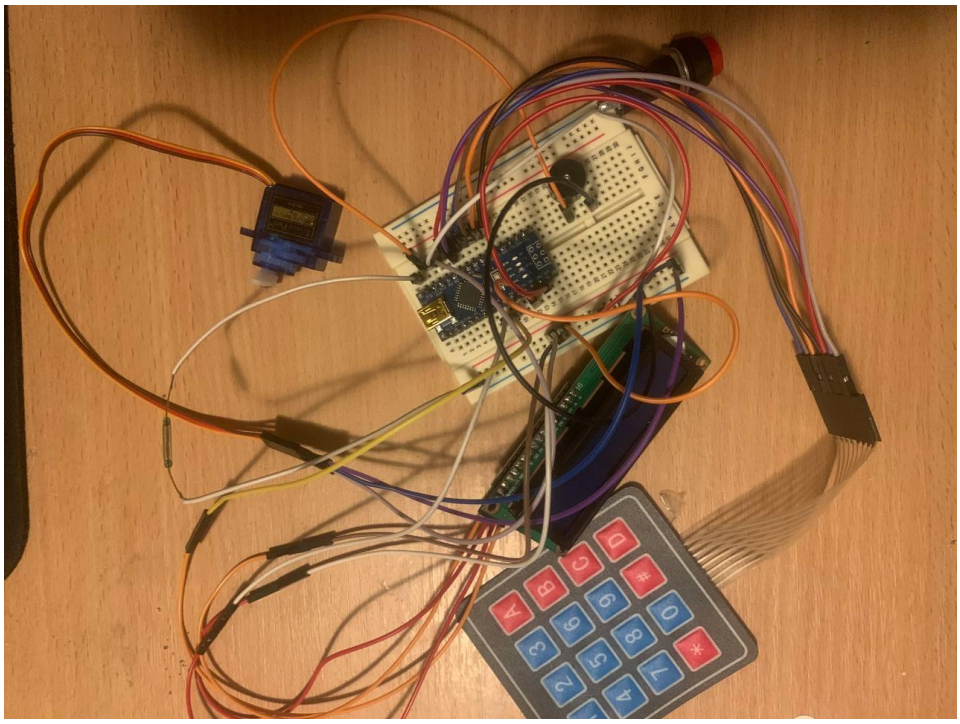


Рис. 7. Приклад створеної схеми для варіанту 1.

Розпіновка схеми.

- Дисплей SDA- A4, SCL - A5.
- Клавіатура - D2-D8.
- Сервопривод - D9.
- Кнопка - D10.
- П’єза - D11.
- Геркон - D12.

Функціонал схеми.

Спочатку ми записуємо у пам’ять пароль. Це реалізовано шляхом вводу 4 цифр і натисканням після цього символу ‘#’. Для відкриття замка потрібно ввести пароль, та натиснути ‘*’. Якщо пароль вірний, то отримаємо звуковий сигнал і вивід на дисплей “Door is open”. Якщо пароль невірний, то отримаємо інший звуковий сигнал і вивід “Door is close”. Для автоматичного закриття дверей використано геркон. Для відкриття дверей зсередини на схему додана кнопка, при натисканні якої, відбувається відкриття дверей і відповідний вивід та звуковий сигнал. Механізм відкривання реалізований сервоприводом. Розроблений код прикладу прошивки знаходиться в Додатку А.

Демонстрація роботи.

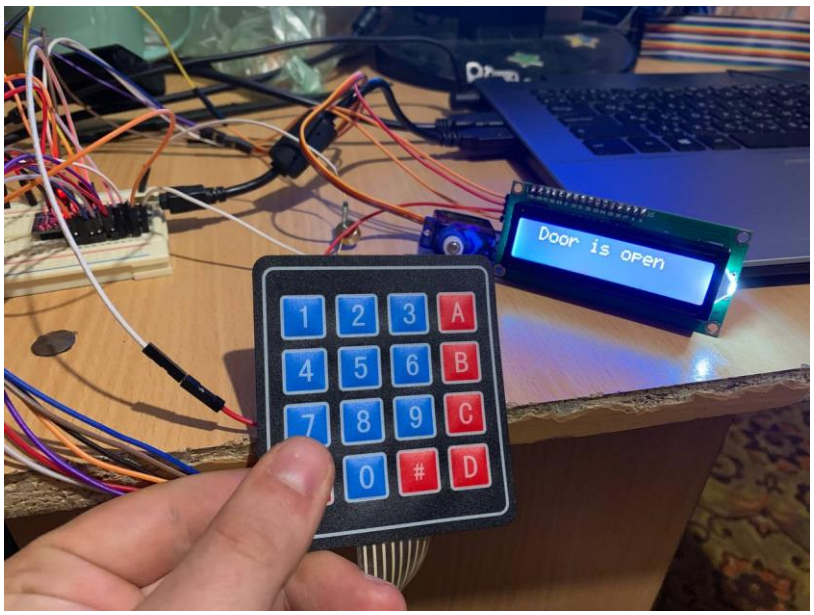


Рис. 8. Приклад роботи створеної схеми для варіанту 1.

3.2. Варіант 2

Пристрій вводу - RFID модуль.

Індикатор закриття дверей - реалізація через кнопку 1.

Пристрої виведення - LCD 1602 та п'єзокристальний випромінювач.

Механізм відкриття замка - серводвигун SG90.

Кнопка 2 для відкриття дверей зсередини та вшивання в пам’ять нових карт.

На рисунку 7 зображений приклад створеної мною схеми для реалізації цього завдання.

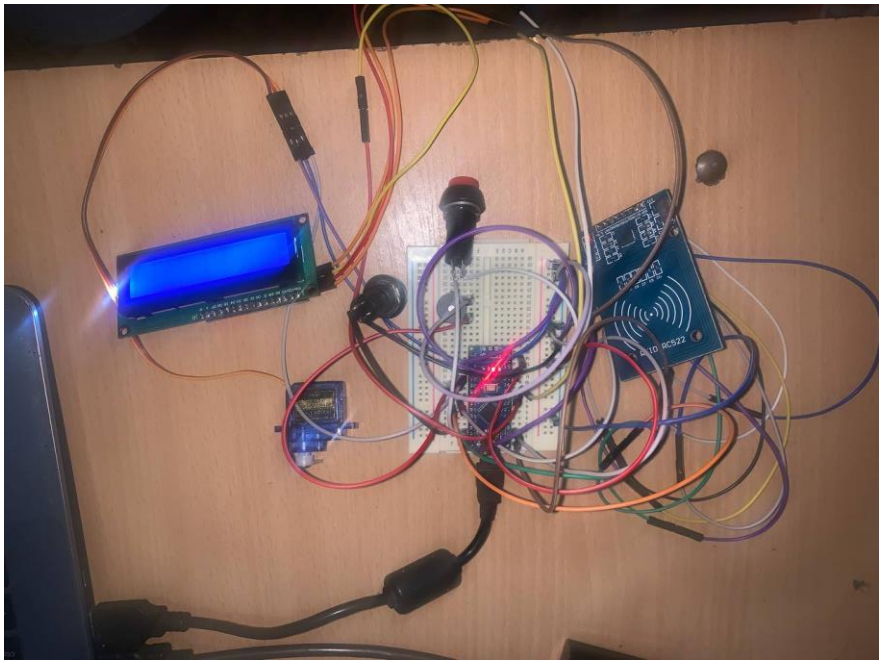


Рис. 9. Приклад створеної схеми для варіанта 2.

Розпіновка схеми.

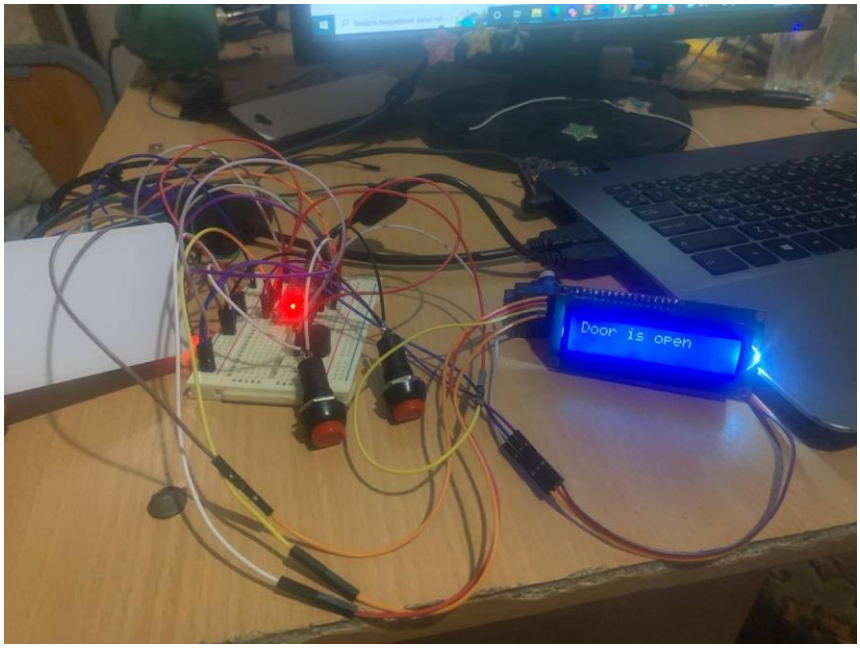
- Сервопривод: D2.
- П’єза: D3.
- Кнопка відкриття/запису: D8.
- Індикатор закриття дверей: D9.
- RFID RC522.
  - 3.3V: 3V3.
  - RST: D6.
  - GND: GND.
  - MISO: D12.

- MOSI: D11.
- SCK: D13.
- SDA: D7

Функціонал схеми.

На початку записано 0 ключів, через це двері не можуть закритися, щоб ми не залишилися ззовні. За допомогою кнопки відкриття дверей, яка виконує також функцію запам’ятовування RFID міток, Arduino запам’ятовує одну або кілька міток. Після цього можна закрити двері. Тепер при піднесенні мітки при закритих дверях, якщо вона буде наявна в пам’яті, двері будуть відкриватись з звуковим сигналом та текстом на дисплей “Door is open”. Якщо мітки немає в базі то буде виводитись текст “Door is close” і інший звуковий сигнал. За допомогою кнопки відкриття дверей також реалізоване видалення міток з пам’яті.

Демонстрація роботи.



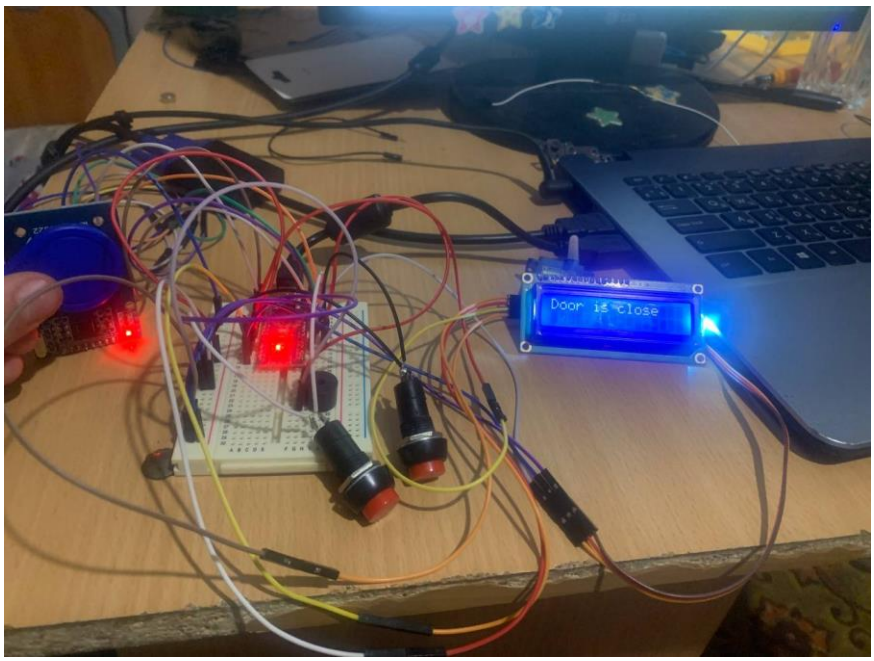


Рис. 10. Приклад роботи створеної схеми для варіанта 2.

Можна побачити, що для обох варіантів схеми реалізовані вірно, і працюють так як і планувалось в завданні.

Висновки

Використовуючи за основу мікроконтролер Arduino Nano, було розроблено два варіанти лабораторної роботи, та створено два макета, де були виконані завдання з цих варіантів:

- Макет з мембранною клавіатурою;
- Макет з RFID модулем RC522;

Виконуючи дану лабораторну роботу студенти зможуть ознайомитись з різними підходами до програмування мікроконтролерів, навчитися працювати з різними периферійними пристроями, дізнатися можливості мікроконтролера Arduino.

Дані варіанти та макети можна використовувати у якості лабораторної роботи для курсу “Периферійні пристрої ”.

Джерела

1. Лисенков М. О. Мікроконтролери в приладах і пристроях: підруч. для студ. техн. спец. вищ. навч. закл. / М. О. Лисенков, І. І. Ключник ; МОН України, Харк. нац. ун-т радіоелектроніки. — Харків: ХНУРЕ, 2014

2. Ю.В. Новиков и П.К. Скоробогатов. Основи микропроцессорной техники. / Иуит; Бинок, 2009

3. Справочник языка Ардуино - Контроллеры Arduino [Електронний ресурс]. - Режим доступу: URL: http://arduino.ru/

4. Atmel - Atmel AVR 8-bit and 32-bit Microcontrollers [Електронний ресурс]. - Режим доступу: URL: http://atmel.com/products/microcontrollers/avr/

5. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL / А.В. Евстифеев. - М.: Издательский дом «Додэка-XXI», 2004.

Додатки

Додаток А.

Програмний код написаний для замка на Arduino Nano з підключеною клавіатурою.

```
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Servo.h>
#define SERVO_PIN       9
#define LOCK_TIMEOUT  1000
#define BTN_PIN         10
#define DOOR_LOCK       12
int buzzer = 11;
int val=1;
int door = 1;
Servo doorServo;
LiquidCrystal_I2C lcd(0x27, 16, 2);
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {5, 4, 3, 2};
byte colPins[COLS] = {8, 7, 6};
char last_four[4] = {0, 0, 0, 0};
char password[4] = {0, 0, 0, 0};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS
);

void setup(){
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(BTN_PIN, INPUT_PULLUP);
  pinMode(DOOR_LOCK, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
}
void loop(){
 val = digitalRead(BTN_PIN);
 door = digitalRead(DOOR_LOCK);
```

```
char key = keypad.getKey();
if(val==0 && door==0){
      unlock();
      delay(10000);
      lock();
      }

if (key){
  if (key != '*' & key != '#') {
    last_four[0]= last_four[1];
    last_four[1]= last_four[2];
    last_four[2]= last_four[3];
    last_four[3]= key;
  }
  else {
    if (key == '#'){
      password[0]= last_four[0];
      password[1]= last_four[1];
      password[2]= last_four[2];
      password[3]= last_four[3];
    }
    else{
      Serial.print(last_four[0]);
      Serial.print(last_four[1]);
      Serial.print(last_four[2]);
      Serial.print(last_four[3]);
      bool pass = true;
      for (int i =0; i <4;i++){
        if (password[i] != last_four[i]) { pass =false; }
      }
      if (pass){
        unlock();
        delay(10000);
        lock();
      }
      else{
          lock();
          delay(1500);
            digitalWrite(buzzer, HIGH);
            delay(1000);
            digitalWrite(buzzer, LOW);
             delay(500);
```

```
        }

      }
    }
  }
}

void lock(void) {
  lcd.setCursor(1, 0);
  lcd.print("Door is close");
  doorServo.attach(SERVO_PIN);
  doorServo.write(170);
  delay(1000);
  doorServo.detach();
  lcd.clear();
}
void unlock(void) {
             digitalWrite(buzzer, HIGH);
            delay(70);
            digitalWrite(buzzer, LOW);
            delay(70);
            digitalWrite(buzzer, HIGH);
            delay(70);
            digitalWrite(buzzer, LOW);
            delay(70);
            digitalWrite(buzzer, HIGH);
            delay(70);
            digitalWrite(buzzer, LOW);
            delay(70);
            digitalWrite(buzzer, HIGH);
            delay(70);
            digitalWrite(buzzer, LOW);
            delay(70);
            digitalWrite(buzzer, HIGH);
            delay(70);
            digitalWrite(buzzer, LOW);
            delay(70);
            digitalWrite(buzzer, HIGH);
            delay(70);
            digitalWrite(buzzer, LOW);
            delay(70);
            lcd.setCursor(1, 0);
```

```
        lcd.print("Door is open");
        doorServo.attach(SERVO_PIN);
        doorServo.write(10);
        delay(1000);
        doorServo.detach();
        delay(6000);
        lcd.clear();
}
```

Додаток Б

Програмний код написаний для Arduino Nano з підключеним RFID модулем

```
#include <Servo.h>
#include <SPI.h>
#include <MFRC522.h>
#include <EEPROM.h>
#include <LiquidCrystal_I2C.h>
#define LOCK_TIMEOUT  1000
```

```
#define MAX_TAGS        3
#define SERVO_PIN       2
#define BUZZER_PIN      3
#define RST_PIN         6
#define CS_PIN          7
#define BTN_PIN         8
#define DOOR_PIN        9
#define EE_START_ADDR   5
#define EE_KEY          100
MFRC522 rfid(CS_PIN, RST_PIN);
Servo doorServo;
LiquidCrystal_I2C lcd(0x27, 16, 2);
#define DECLINE 0
#define SUCCESS 1
#define SAVED   2
#define DELITED 3
bool isOpen(void) {
  return digitalRead(DOOR_PIN);
}
void lock(void) {
  doorServo.attach(SERVO_PIN);
  doorServo.write(170);
  delay(1000);
  doorServo.detach();
  Serial.println("lock");
}
void unlock(void) {
  doorServo.attach(SERVO_PIN);
  doorServo.write(10);
  delay(1000);
  doorServo.detach();
  Serial.println("unlock");
}
bool locked = true;
bool needLock = false;
uint8_t savedTags = 0;
void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  pinMode(BTN_PIN, INPUT_PULLUP);
  pinMode(DOOR_PIN, INPUT_PULLUP);
```

```
  pinMode(BUZZER_PIN, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(1, 0);
  uint32_t start = millis();
  bool needClear = 0;
  while (!digitalRead(BTN_PIN)) {
    if (millis() - start >= 3000) {
      needClear = true;
      indicate(DELITED);
      break;
    }
  }
  if (needClear or EEPROM.read(EE_START_ADDR) != EE_KEY) {
    for (uint16_t i = 0; i < EEPROM.length(); i++) EEPROM.write(i, 0x00);
    EEPROM.write(EE_START_ADDR, EE_KEY);
  } else {
    savedTags = EEPROM.read(EE_START_ADDR + 1);
  }

  if (savedTags > 0) {
    if (isOpen()) {
      lcdSetup(SUCCESS);
      locked = false;
      unlock();
    } else {
      lcdSetup(DECLINE);
      locked = true;
      lock();
    }
  } else {
    lcdSetup(SUCCESS);
    locked = false;
    unlock();
  }
}
void loop() {
  static uint32_t lockTimeout;
  if (locked and !digitalRead(BTN_PIN)) {
    unlock();
    indicate(SUCCESS);
    lockTimeout = millis();
```

```
    locked = false;
  }
  if (isOpen()) {
    lockTimeout = millis();
  }

  if (savedTags > 0 and !locked and millis() - lockTimeout >= LOCK_TIMEOUT)
{

    lock();
    lcdSetup(DECLINE);
    locked = true;
  }
  static uint32_t rfidTimeout;
  if (rfid.PICC_IsNewCardPresent() and rfid.PICC_ReadCardSerial()) {
    if (isOpen() and !digitalRead(BTN_PIN) and millis() - rfidTimeout >= 500) {
      saveOrDeleteTag(rfid.uid.uidByte, rfid.uid.size);
    } else if (locked) {
      if (foundTag(rfid.uid.uidByte, rfid.uid.size) >= 0) {
        unlock();
        indicate(SUCCESS);
        lockTimeout = millis();
        locked = false;
      } else if (millis() - rfidTimeout >= 500) {
        indicate(DECLINE);
      }
    }
    rfidTimeout = millis();
  }
  static uint32_t rfidRebootTimer = millis();
  if (millis() - rfidRebootTimer > 500) {
    rfidRebootTimer = millis();
    digitalWrite(RST_PIN, HIGH);
    delay(1);
    digitalWrite(RST_PIN, LOW);
    rfid.PCD_Init();
  }
}
void lcdSetup(bool state) {
  if (state) {
    lcd.print("Door is open");
    delay(2000);
```

```
    lcd.clear();
  } else {
    lcd.print("Door is close");
    delay(2000);
    lcd.clear();
  }
}
void indicate(uint8_t signal) {
  lcdSetup(signal);
  switch (signal) {
    case DECLINE:
      Serial.println("DECLINE");
      for (uint8_t i = 0; i < 2; i++) {
        tone(BUZZER_PIN, 100);
        delay(300);
        noTone(BUZZER_PIN);
        delay(100);
      }
      return;
    case SUCCESS:
      Serial.println("SUCCESS");
      tone(BUZZER_PIN, 890);
      delay(330);
      noTone(BUZZER_PIN);
      return;
    case SAVED:
      Serial.println("SAVED");
      for (uint8_t i = 0; i < 2; i++) {
        tone(BUZZER_PIN, 890);
        delay(330);
        noTone(BUZZER_PIN);
        delay(100);
      }
      return;
    case DELITED:
      Serial.println("DELITED");
      for (uint8_t i = 0; i < 3; i++) {
        tone(BUZZER_PIN, 890);
        delay(330);
        noTone(BUZZER_PIN);
        delay(100);
      }
```

```
      return;
    }
  }
  bool compareUIDs(uint8_t *in1, uint8_t *in2, uint8_t size) {
    for (uint8_t i = 0; i < size; i++) {
      if (in1[i] != in2[i]) return false;
    }
    return true;
  }
  int16_t foundTag(uint8_t *tag, uint8_t size) {
    uint8_t buf[8];
    uint16_t address;
    for (uint8_t i = 0; i < savedTags; i++) {
      address = (i * 8) + EE_START_ADDR + 2;
      EEPROM.get(address, buf);
      if (compareUIDs(tag, buf, size)) return address;
    }
    return -1;
  }
  void saveOrDeleteTag(uint8_t *tag, uint8_t size) {
    int16_t tagAddr = foundTag(tag, size);
    uint16_t newTagAddr = (savedTags * 8) + EE_START_ADDR + 2;
    if (tagAddr >= 0) {
      for (uint8_t i = 0; i < 8; i++)  {
        EEPROM.write(tagAddr + i, 0x00);
        EEPROM.write(tagAddr + i, EEPROM.read((newTagAddr - 8) + i));
        EEPROM.write((newTagAddr - 8) + i, 0x00);
      }
      EEPROM.write(EE_START_ADDR + 1, --savedTags);
      indicate(DELITED);
    } else if (savedTags < MAX_TAGS) {
      for (uint16_t i = 0; i < size; i++) EEPROM.write(i + newTagAddr, tag[i]);
      EEPROM.write(EE_START_ADDR + 1, ++savedTags);
      indicate(SAVED);
    } else {
      indicate(DECLINE);
      lcdSetup(SUCCESS);
    }
  }
```