КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«РОЗРОБКА МАКЕТУ СИСТЕМИ СИГНАЛІЗАЦІЇ ДЛЯ ЗАХИСТУ БУДИНКУ»

Виконав:
студент 4-го курсу бакалаврату
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Дударенко Святослав Сергійович _____

Науковий керівник:
кандидат фізико-математичних наук, доцент
Баужа Олександр Стасісович _____

Рецензент:
_____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

Р Е Ф Е Р А Т

Випускна кваліфікаційна робота бакалавра : 46 с., 26 рис., 2 додатка, 18 джерел.

У дипломній роботі описана розробка макету системи сигналізації на базі *Arduino* з керуванням через *GSM/GPRS* модуль.

Метою роботи є розробка стаціонарної охоронної системи, розробка програмного коду для подальшого налаштування системи кінцевим користувачем

В процесі роботи були обрані електронні компоненти, розроблена загальна схема пристрою та був проведений аналіз ефективності та доцільності рішень з можливими подальшими вдосконаленнями, ризиками в сфері безпечності користування та розробки пристрою.

ЗМІСТ

ВСТУП

Електронні охоронні системи стали невід'ємною частиною повсякденного життя і є дуже важливими елементами забезпечення безпеки. В сучасному світі електронні охоронні системи використвуються приктично в будь-якому місці, де є необхідність забезпечення безпеки. Електронні охоронні системи для будинків та приміщень почали розвиватися ще в 1970-х роках. У той час вони були простішими і мали обмежені функції. Однак з плином часу, електронні охоронні системи стали все більш складними та розвинутими.

Однією з основних переваг електронних охоронних систем для будинків та приміщень є їх широкий функціонал. Сучасні системи можуть включати в себе не тільки сигналізацію, але й системи відеоспостереження, контроль доступу, датчики пожежі та інші важливі функції. Багато систем також можуть бути віддалено керованими, в тому числі через мобільні пристрої.

До недавнього часу одним з основних недоліків електронних охоронних систем були дроти, складність в протягуванні їх по периметру приміщення, вразливість до неумисного та умисного механічного пошкодження, додаткові витрати та громіздкість, все це робило електронні охоронні системи не зручними та вразливими.

Рішенням проблем з дротами стали *GSM* сигналізації. Основна перевага полягає в тому, що GSM сигналізації не потребують проводів для передачі інформації. Це дозволяє встановлювати їх без будівельних робіт та витрат на проводку. Крім того, *GSM* сигналізації можуть бути підключені до мобільної мережі і відправляти повідомлення про спрацювання сигналізації на вибраний номер телефону. Це робить їх більш зручними у використанні та забезпечує швидку реакцію на події, які спричиняють спрацювання сигналізації. Крім того, *GSM* сигналізації зазвичай мають більш високу надійність, оскільки вони не залежать від провідної мережі, яка може бути підірвана чи пошкоджена. Вони також можуть бути програмовані для автоматичного оновлення програмного забезпечення.

Зараз на ринку розміщується достатньо компаній які продають якісні та надійнійні сигналізації, проте вони досить дорогі.

В цій дипломній роботі буде розроблено бюджетний варіант *GSM* сигналізації на базі *Arduino Nano* та розроблене програмне забезпечення для неї, аби в результаті кінцевий користувач отримав досить не дорогу електронну охоронну систему і зміг самотужки її налаштувати.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд існуючих аналогів на ринку

Сигналізація *“Ajax StarterKit 2 Black”,* зображено на рисунку 1.1

Рисунок 1.1 - *“Ajax StarterKit 2 Black”*

Характеристики *“Ajax StarterKit 2 Black” :*

- *Канали зв'язку Ethernet, 2G (GSM900/DCS1800 (B3/B8)), 3G (WCDMA 850/900/2100 (B1/B5/B8)), LTE (FDD B1/B3/B5/B7/B8/B20);*

- *Максимальна кількість користувачів 199;*

- *До 15 години автономної роботи;*

- *Підтримка SIM-карток: 2 SIM-картки;*

- *Час доставки сигналу тривоги 0.15с;*

- *Мінімальна робоча температура -10°C;*

- *Максимальна робоча температура 50°C;*

- *Вартість 10300 грн.*

Отже, *Ajax StarterKit 2* - це електронна сигналізація для дому або офісу, яка має високу надійність і безпеку, а також проста у використанні. Вона складається з центральної блокувальної одиниці, датчиків руху та дверей, дистанційного пульта управління і мобільного додатка.

Система *Ajax StarterKit 2* працює в бездротовому режимі, що дозволяє легко і швидко встановити її в будь-якому приміщенні. Центральний блок має вбудований GSM-модуль і підтримує роботу з *SIM*-картами, що дозволяє отримувати повідомлення на мобільний телефон в разі спрацювання сигналізації.

Загалом, *Ajax StarterKit 2* є досить надійною і зручною системою електронної сигналізації, яка допоможе забезпечити безпеку будинку або офісу.

Сигналізація *"PiPo WL-99CGST "*

Сигналізація *"PiPo WL-99CGST",* зображено на рисунку 1.2

Рисунок 1.2 - *"PiPo WL-99CGST"*

Характеристики *“PiPo WL-99CGST ” :*

- *Підтримувана частота GSM: 900/1800/850/1900 МГц*
- *Частота бездротового модуля 433 МГц 2262/1.5-4.7M EV1527/300K*
- *7 годин автономної роботи*
- *6 номерів для голосових повідомлень та 6 номерів для СМС*
- *Вбудована сирена 110 дБ*
- *Робоча температура: -10℃ + 50℃*
- *Вартість 3200 грн.*

Загалом Комплект бездротової *GSM+ Wi-Fi* сигналізації *PiPo WL-99CGST* є досить простим і зручним рішенням для захисту дому або офісу. Він має багато корисних функцій, таких як бездротовість, вбудований *GSM*-модуль та *Wi-Fi*-модуль, датчики руху, дверей та вікон, дистанційне управління, вбудовані датчики температури та вологості, інтеграцію з іншими системами через *RS485* та *CAN* протоколи.

Особлива увага приділена зручності використання - з комплектом поставки поставляється дистанційний пульт управління, а також можливість керування через мобільний додаток або веб-інтерфейс. Крім того, наявність вбудованого *GSM*-модуля дозволяє отримувати повідомлення про спрацювання сигналізації на мобільний телефон, що забезпечує більшу надійність та безпеку.

Загалом, *PiPo WL-99CGST* є надійним і простим у використанні комплектом бездротової *GSM+ Wi-Fi* сигналізації, який може бути використаний для захисту будь-якого приміщення від небажаних вторгнень.

Сигналізація *“CoVi Security GSM-200 Kit Wi-Fi Tuya Smart ”*

Сигналізація *“CoVi Security GSM-200 Kit Wi-Fi Tuya Smart”,* зображено на рисунку 1.3

Рисунок 1.3 - *“CoVi Security GSM-200 Kit Wi-Fi Tuya Smart”*

Характеристики *“CoVi Security GSM-200 Kit Wi-Fi Tuya Smart” :*

- *Тип з'єднання: Wi-Fi 2.4 ГГц 802.11b/g/n/GSM 850/900/1800/1900 МГц (слот під SIM карту)*

- *Вбудована сирена: 85 дБ*

- *Вбудований акумулятор на 4 години автономної роботи*

- *Кнопка тривоги SOS: на корпусі пристрою*

- *Кількість номерів sms-оповіщення: 10*

-*Кількість номерів додзвону: 6*

- *Вартість 2500 грн.*

*CoVi Security GSM-200 Kit Wi-Fi Tuya Smar*t - це комплект бездротової *GSM+ Wi-Fi* сигналізації, яка підтримує протокол *Tuya Smart*. *Tuya Smart* - це платформа для розумного дому, яка дозволяє керувати різними пристроями за допомогою мобільного додатку або голосових повідомлень.

Комплект містить вбудований *GSM*-модуль та *Wi-F*i-модуль, датчики руху, дверей та вікон, пульт дистанційного управління, інтегровану клавіатуру для введення коду доступу, а також можливість керування через мобільний додаток та голосові повідомлення з підтримкою *Amazon Alexa* та *Google Assistant.*

Окрім того, сигналізація підтримує інші протоколи підключення, такі як *RS485*, для інтеграції з іншими системами, наприклад, відеоспостереженням або домофоном.

Всі ці характеристики роблять *CoVi Security GSM-200 Kit Wi-Fi Tuya Smart* досить привабливим варіантом для тих, хто шукає зручну та надійну сигналізацію для свого будинку або офісу, яка підтримує різні форми зв'язку та може бути легко інтегрована в розумну систему дому з підтримкою *Tuya Smart.*

2 ВИБІР ТА ОБГРУНТУВАННЯ СХЕМОТЕХНІЧНИХ РІШЕНЬ

2.1 Розробка схеми

На рис 2.1 показана структурна схема сигналізації на базі *Arduino Nano*

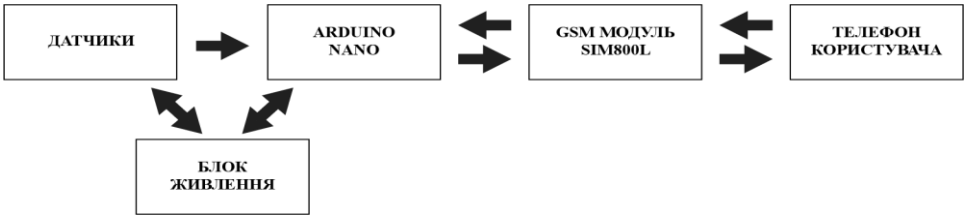

Рисунок 2.1 - Структурна схема сигналізації

Схемою показано порядок та структуру взаємодії між елементами

На датчики та на *Arduino Nano* з блоку живлення подається напруга 5V та 3V відповідно.

Датчики та *GSM* модуль під'єднані до *Arduino Nano* яка виступає основним модулем керування системою. *GSM* модуль в свою чергу виконує зв'язок з телефоном користувача шляхом надсилання *SMS* повідомлень

2.2 Вибір мікроконтролера з платою

В ході аналізу та порівняння характеристик різних мікроконтролерів було обрано мікроконтролер *Atmega328*. На рис. 2.2 зображено мікроконтролер *Atmega328.*

рис. 2.2 мікроконтролер Atmega328.

Мікроконтролер *ATmega328* гарний вибір для системи сигналізації за такими причинами:

- Надійність: *ATmega328* є дуже надійним мікроконтролером з багатою історією успіху в різних проектах. Він має низьку ймовірність відмов, яка є дуже важливою у системі сигналізації, де точність та надійність є ключовими вимогами.

- Легкість програмування: *ATmega328* є легким у програмуванні мікроконтролером з великою спільнотою розробників та доступними ресурсами для початківців. Це робить його дуже привабливим вибором оскільки однією з цілей роботи створити легкопрограмовану систему сигналізації

- Низька вартість: *ATmega328* є доступним мікроконтролером, що робить його привабливим оскільки одна з вимог зробити сигналізацію б'юджетною.

- Периферійні пристрої: *ATmega328* має широкий спектр периферійних пристроїв, таких як *АЦП, ШІМ*, таймери та інші, які можна використовувати для розробки різноманітних функцій у системі сигналізації.

Отже, *ATmega328* є хорошим вибором для системи сигналізації через його надійність, легкість програмування, доступну ціну та багатий функціонал периферійних пристроїв.

Як платформу яка містить в основі *ATmega328* було обрано пристрій *Arduino Nano* *Arduino Nano* - це мініатюрна версія платформи *Arduino*, яка базується мікроконтролері *Atmega328*. На рис. 2.3 зображено *Arduino Nano*

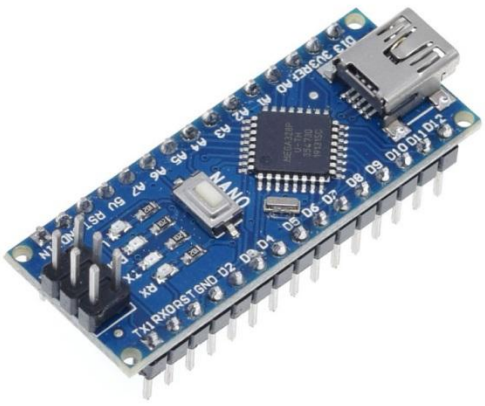

Рис. 2.3 Arduino Nano

Основні характеристики *Arduino Nano :*

- *Розміри: 18x45 мм;*
- *Вбудований USB-конвертер для програмування та зчитування даних;*

- *Напруга живлення: 5V або 3.3V;*
- *Швидкість процесора: 16 МГц;*
- *Кількість цифрових входів/виходів: 14;*
- *Кількість аналогових входів: 8.*

Arduino Nano - це компактна мікроконтролерна платформа, яка базується на мікроконтролері *ATmega328P* від *Microchip*. Це досить популярна модель в середовищі розробки електроніки, що забезпечує зручне та просте програмування мікроконтролерів.

Arduino Nano має на борту мікроконтролер, кілька входів/виходів *(GPIO),* а також можливість розширення за допомогою додаткових модулів та сенсорів. Це дозволяє створювати різноманітні проекти, від простих до складних, що включають в себе різні функції, наприклад, звукові та світлові ефекти, моніторинг температури та вологості, контроль над механізмами тощо.

Arduino Nano має компактний розмір (18 х 45 мм) та низьку вартість, що робить його досить популярним в середовищі розробки електроніки, особливо в хобі-проектах та прототипуванні.

Arduino Nano можна програмувати за допомогою *Arduino IDE* - це безкоштовна інтегрована середовище розробки, що підтримує велику кількість мікроконтролерів та дозволяє швидко та просто розробляти програми для *Arduino*.

Узагалі, *Arduino Nano* є потужною, функціональною та досить доступною мікроконтролерною платформою, яка підходить для широкого кола проектів, включаючи системи сигналізації, *IoT*-проекти, робототехніку та багато інших.

2.3 Вибір *GSM* модуля

Як ми вже визначили в технічній частині, керування системою сигналізації буде виконуватись за допомогою *GSM* модуля.

В процесі огляду варіантів було обрано *GSM* модуль *SIM800L,* зображено на рис. 2.4

Рис. 2.4 GSM модуль SIM800L

SYM800L - це модуль *GSM/GPRS* для передачі даних, який дозволяє підключатися до мобільних мереж з метою передачі даних в режимі реального часу. Основні характеристики та особливості модуля *SYM800L* наступні:

- *Підтримка мереж GSM 850/900/1800/1900 МГц та GPRS Class 10.*
- *Має вбудований SIM-слот, що дозволяє безпосередньо встановлювати SIM-карту.*
- *Підтримка декількох різних комунікаційних протоколів, включаючи TCP/UDP, HTTP і FTP.*
- *Має різні інтерфейси для з'єднання з іншими пристроями, включаючи UART, USB і GPIO.*
- *Підтримка вбудованої GPS-функції для визначення географічного положення.*
- *Низька споживана потужність, що дозволяє використовувати модуль в батарейних пристроях.*
- *Вбудована антена для забезпечення якісного зв'язку.*
- *Для роботи модуля SYM800L необхідно забезпечити живлення в діапазоні від 3,4 В до 4,4 В.*
- Має компактні розміри (25,5 х 17,8 мм) та легка в монтажі.

Підключення *SIM800L до Arduino Nano* на базі мікроконтролера *ATmega328* включає декілька особливостей, які потрібно враховувати при розробці системи.

Особливості:

- *Живлення: SIM800L потребує стабільного живлення від 3.7 до 4.2 Вольт, тому перед підключенням SIM800L до Arduino Nano необхідно переконатися, що живлення мікроконтролера забезпечує потрібний рівень напруги.*
- *Рівні логіки: SIM800L працює на рівні 3.3 В, тоді як мікроконтролер Arduino Nano використовує логіку на рівні 5 В. Тому для підключення SIM800L до мікроконтролера необхідно використовувати логічний рівень збільшувача напруги, щоб забезпечити сумісність між SIM800L та Arduino Nano.*
- *Підключення до порту UART: SIM800L підключається до мікроконтролера через порт UART, тому необхідно з'єднати TX SIM800L з RX Arduino Nano та RX SIM800L з TX Arduino Nano. Також потрібно встановити швидкість передачі даних, зазвичай це 9600 бод.*
- *Керування живленням: SIM800L може вимикатися та включатися за допомогою підключення до піни PWRKEY. Для керування живленням SIM800L можна використовувати окремий пін або підключити PWRKEY до додаткового резистора та транзистора для керування через мікроконтролер.*
- *Керування станом SIM800L: SIM800L має кілька станів, таких як стан готовності до роботи, стан відправки повідомлень або даних, стан прийому даних та ін. Щоб забезпечити правильну роботу SIM800L та відслідковувати стан пристрою, необхідно правильно налаштувати взаємодію між SIM800L та мікроконтролером Arduino Nano.*
- *Використання додаткових модулів: Для розширення функціональності можна використовувати додаткові модулі, наприклад, для забезпечення зв'язку з мережею Wi-Fi, GPS-модуля для визначення координат, сенсорів для вимірювання температури, вологості, тиску тощо. При цьому необхідно правильно налаштувати взаємодію між модулями та мікроконтролером.*

Підключення покроково :

1. Припаяти або підключити виносну антену до *GSM*-модуля.

2. Вставити *SIM*-карту в роз'єм *GSM*-модуля.

3. Підключити вивід Tx модуля до виводу 3 на *Arduino Nano.*

4. Підключення виводу Rx модуля безпосередньо до виводу *Arduino Nano* не можливе через різну напругу (5В та 3,3 В відповідно). Тому необхідно скористатися дільником напруги на резисторах. Для цього потрібно підключити резистор на 10 кОм між виводами Rx (*SIM800*L) та виводом 2 (Arduino), а другий резистор на 10 кОм між виводами Rx (*SIM800L*) та GND. 5. Підключити живлення *GSM*-модуля. У прикладі використовується стабілізатор напруги на *LM2596*.

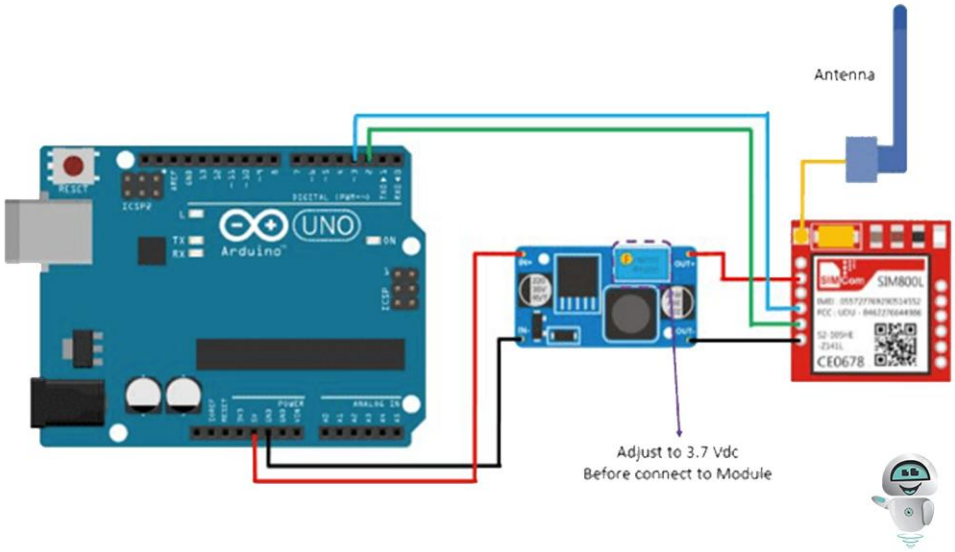

Рис. 2.5 - схема підключення SIM800L до Arduino

2.4 Вибір датчика відкриття дверей

Один з бюджетних та хороших варіантів датчика відкриття дверей для системи сигналізації на базі *Arduino Nano* - це датчик *MC-38*.

Рис. 2.6 - Магнітно-контактний герконовий датчик відкриття дверей МС-38

МС-38 - це нормально закритий *(NC)* магнітний датчик, що складається з магнітного контакту та магнітного елементу. При закритті контакту відбувається замикання контактів, що спричиняє зміну стану виводу датчика.

Даний датчик має наступні характеристики:

- *Напруга живлення: 100 В постійного струму (ВПС)*
- *Максимальний струм: 0.5 А*
- *Максимальна потужність: 10 Вт*
- *Максимальна напруга комутації: 200 ВПС*
- *Максимальний струм комутації: 0.5 А*
- *Максимальна частота комутації: 10 Гц*
- *Довжина кабелю: 30 см*
- *Розміри: 27 х 14 х 8 мм*

При підключенні до *Arduino Nano*, вивід контакту датчика можна підключити до цифрового входу (наприклад, D2), а вивід землі до землі *Arduino*. При відкритті дверей та розриві контакту на вході буде змінено логічний рівень, що можна використати для подальшої обробки в програмному коді.

МС-38 є досить популярним та доступним датчиком, його можна знайти в багатьох інтернет-магазинах. Він також досить простий у використанні та монтажі, що робить його популярним вибором для проектів на базі *Arduino Nano*.

2.5 Вибір датчика газу

Дуже важливу роль у збереженні майна несе на собі захист від пожежі, тож в процесі розробки було прийнято рішення додати датчик газу до нашого пристрою.

Один з бюджетних і хороших варіантів датчика газу для системи сигналізації - *MQ-2 Gas Sensor Module.*

Рис. 2.7 - MQ-2 Gas Sensor Module.

Основні характеристики датчика:

- *Напруга живлення: 5 В;*
- *Сигнальний вивід: аналоговий;*
- *Діапазон вимірювання: 300 - 10 000 ppm (партіями на мільйон);*
- *Чутливість: > 3,3 R / ppm;*
- *Робоча температура: -10 ℃ - 50 ℃;*
- *Розмір: 32 мм x 22 мм x 27 мм;*
- *Вага: 5 г.*

Датчик *MQ-2* може виявляти різні гази, включаючи метан, пропан, бутан, ліквідний газ, дим, водень і багато інших. Для підключення датчика до *Arduino Nano* необхідно підключити живлення (+5 В) і землю, а також підключити аналоговий вивід датчика до одного з аналогових входів *Arduino.*

Одним з недоліків цього датчика є те, що він може виявляти не тільки гази, а й дим, що може призвести до хибних спрацювань системи сигналізації. Також необхідно враховувати, що датчик не може розрізняти між різними видами газів, тому якщо в помешканні використовується певний тип газу, необхідно використовувати датчик, який спеціалізується на виявленні саме цього газу.

Модуль давача газу *MQ-2* дійсно є хорошим варіантом для системи сигналізації на базі Arduino Nano, забезпечуючи детекцію газів, таких як *LPG*, і метан, серед інших. Цей модуль використовує чутливий елемент, який реагує на зміни концентрації газу в повітрі і надсилає сигнал на мікроконтролер.

MQ-2 має невеликі габарити та вагу, що дозволяє легко встановити його в будь-якому місці. Він також має широкий діапазон робочих напруг, що дозволяє підключати його до різних типів мікроконтролерів, в тому числі Arduino Nano. Бібліотека для програмування, що надається розробником, дозволяє легко і швидко інтегрувати модуль *MQ-2* в систему сигналізації та налаштувати його параметри.

2.6 Вибір датчика руху

Звісно ж невід’ємною частиною кожної системи сигналізації є датчики руху.

Один з бюджетних варіантів датчика руху для системи сигналізації на базі Arduino Nano - *PIR-датчик руху HC-SR501*.

Рис. 2.8 - HC-SR501.

Характеристики датчика:

- *Напруга живлення: 5 В;*
- *Струм споживання: менше 50 мА;*
- *Дальність виявлення руху: до 7 метрів;*
- *Кут виявлення руху: 120 градусів;*
- *Частота звернення: 1 Гц;*
- *Вихідний сигнал: TTL сигнал з високим рівнем (3,3 В) при виявленні руху.*

Давач називається **PIR** (Passive Infra-Red). Пасивний він тому, що для виявлення руху не використовується будь-яка додаткова енергія, крім тієї, що випускається самими об'єктом

Цей модуль *PIR HC-SR501* досить популярний і доступний, його можна придбати в різних магазинах електроніки за прийнятну ціну. Він працює з принципом виявлення інфрачервоного випромінювання, що випромінюється рухомого об'єкта.

Модуль має дві потенціометри, які можна налаштувати: "Czas opóźnienia" - затримка, після якої сигнал буде вимкнено, та "Czułość" - чутливість датчика. Модуль має три виводи: VCC, GND та OUT. VCC

Щоб підключити датчик до *Arduino Nano*, потрібно підключити вивід *VCC* до піну *5V*, *GND* - до *GND*, а вивід *OUT* - до піна *D2* або іншого піна з можливістю зчитування сигналу високого рівня *(HIGH)* у програмі.

Для зчитування стану датчика використовують функцію *digitalRead ()*, яка поверне значення *HIGH*, якщо датчик виявив рух, та *LOW*, якщо руху немає.

Датчик *HC-SR501* має хорошу чутливість та швидкість реакції, що дозволяє виявляти рух швидко та ефективно. Також він має компактний розмір та просту схему підключення. Однак, його дальність виявлення руху може бути недостатньою для деяких вимог, а кут виявлення руху може бути недостатньо широким для деяких випадків застосування тож при його монтуванні варто зважити на його характеристики.

2.7 Вибір датчика температури та вологості

Один з найкращих бюджетних варіантів датчиків вологості та температури для системи сигналізації - це *DHT11*.

DHT11 - це цифровий датчик вологості та температури, який має досить дешеву ціну, але при цьому досить точно вимірює вологість та температуру. Він має цифровий інтерфейс та може підключатися безпосередньо до плати *Arduino nano.*

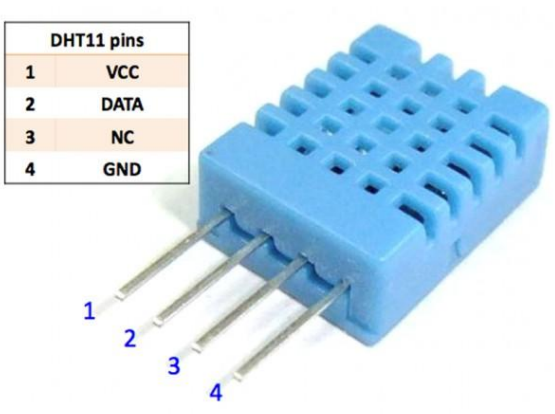

Рис. 2.9 - Датчик температури та вологості DHT11.

Основні характеристики *DHT11:*

- *Діапазон вимірювання температури: від 0 до 50 градусів Цельсія;*
- *Діапазон вимірювання вологості: від 20% до 80%;*
- *Точність вимірювання температури: +/- 2 градуси Цельсія;*
- *Точність вимірювання вологості: +/- 5% відносної вологості;*
- *Робоча напруга: 3.3-5V;*
- *Поточний споживання: менше 2.5 мА;*
- *Інтерфейс: 1-Wire;*
- *Розміри: 23мм х 12мм х 5мм;*

DHT11 є досить надійним та точним датчиком вологості та температури за свою ціну. Він є хорошим варіантом для бюджетної системи сигналізації на базі *Arduino nano.*

2.8 Вибір зумера для звукового оповіщення

Один з найкращих бюджетних варіантів модулів активного зумера для системи сигналізації - це модуль *KY-012.*

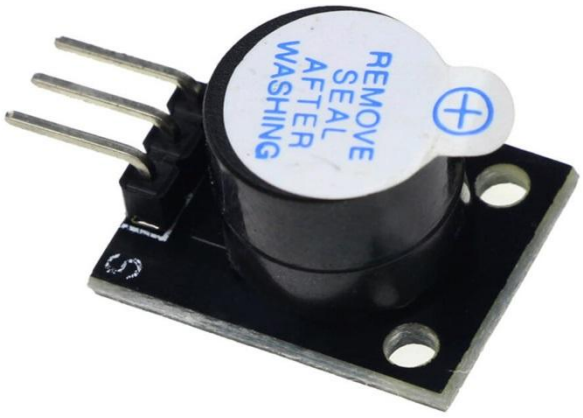

Рис 2.10 - KY-012.

KY-012 - це невеликий модуль активного зумера з вбудованим транзистором, який може видаляти звукові сигнали з досить високою гучністю за свою вартість. Його можна легко підключити до плати *Arduino nano.*

Основні характеристики *KY-012:*

- *Напруга живлення: 3.5-5V*
- *Сигнал зумера: активний (видає звукові сигнали)*
- *Гучність: до 80 дБ*
- *Розміри: 18мм х 15мм х 5мм*

KY-012 є надійним та простим модулем зумера за свою ціну. Він є хорошим варіантом для бюджетної системи сигналізації на базі Arduino nano. Завдяки своїм малим розмірам, доволі гучному сигналу та низькій ціні *KY-012* є чи не найкращим з доступних варіантів

2.9 Вибір контролюючого модулю заряду акумуляторної батареї

В процесі перегляду та аналізу доступних варіантів було обрано модуль *TP4056*

Рис. 2.11 - контролюючий модуль заряду акумуляторної батареї TP4056

TP4056 - це інтегральна схема зарядки літій-іонних акумуляторів, яка має наступні основні характеристики:

- *Сумісність з літій-іонними та літій-полімерними акумуляторами на 3.7 В*
- *Максимальний струм зарядки 1 А*
- *Основний вхідний інтерфейс microUSB*
- *Діапазон вхідної напруги від 4.5 В до 5.5 В*
- *Розміри 25x19x6 мм*

- *Діапазон робочих температур від -10 ° C до +85 ° C*

Модуль на базі чіпу *TP4056*, який зображений на *Рисунку 2.11*, є досить надійним та бюджетним варіантом для зарядки літій-іонних та літій-полімерних акумуляторів на *3,7 В*. Цей модуль забезпечує струм зарядки до *1А,* що дозволяє заряджати акумулятор досить швидко. Діапазон вихідної напруги становить *4,5 В - 5,5 В*, що повністю сумісно з напругою живлення плати *Arduino Nano* та датчиків сигналізації.

Основний вхідний інтерфейс модуля - *microUSB,* що дозволяє легко підключати модуль до зарядного пристрою або до будь-якого пристрою з підтримкою *USB*-підключення. Модуль має невеликі розміри - *25x19x6 мм*, що дозволяє легко інтегрувати його в систему сигналізації.

Модуль *TP4056* має хороший профіль *CC/CV* та може бути адаптований до багатьох різних конфігурацій зарядки та типів літій-іонних акумуляторів, що дозволяє його використовувати для різних проектів, які вимагають зарядки акумуляторів.

Варто зауважити що модуль з *TP4056* також має вбудований захист від перевантаження, короткого замикання та перевернення полярності, що забезпечує безпеку під час зарядки акумулятора

Отже, модуль на базі чіпу *TP4056* є хорошим варіантом для зарядки літій-іонних та літій-полімерних акумуляторів у системі сигналізації на базі *Arduino Nano*, оскільки він має необхідні характеристики та досить надійний.

2.10 Вибір резистора

В ролі резисторів в ланцюгах постійного, змінного та імпульсного струму ми будемо використовувати вметалоплівкові резистори з дротяними аксіальними виводами. З оглядом на схему було обрано резистор з опором *10 кОм* і розсіювальною потужністю *0.125 Вт*

Рис 2.12 - Зовнішній вигляд резистора

Резистор може працювати в діапазоні температур від -60°С до +70°С з допуском 5%. Такий резистор є досить надійним і з легкістю вмнотовується до схеми сигналізації.

2.11 Вибір конденсатора

В схемі використаний алюмінієвий електролітичний конденсатор .

Даний конденсатор призначений для кіл за постійним і пульсуючим струмом .

Даний конденсатор має полярний тип конструкції, все маркування присутнє на самому корпусі (ємність, номінальна напруга та полярність виводів)

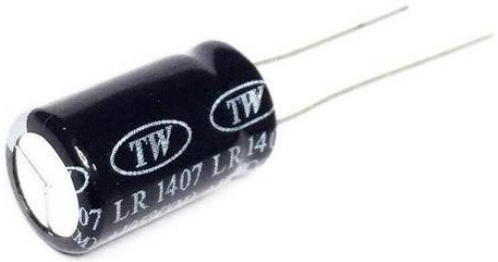

Рис. 2.13 - Зовнішній вигляд конденсатора

Характеристики :

- *ємність - 1000 мікрофарад*
- *максимальна напруга - 16 вольт*
- *робоча температура - від -40 до +105 градусів Цельсія*
- *розміри - 8 на 16,5 міліметрів.*

3 РОЗРОБКА ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Мікроконтролери, на яких працюють прилади *Arduino*, мають дуже обмежені ресурси - мало пам'яті та обмежені швидкості процесора. Це ставить певні вимоги до мов програмування, які використовуються для розробки програмного забезпечення для *Arduino*. Мови програмування *Assembler*, *C* і *C++* найбільше підходять для таких умов, оскільки дозволяють писати багатофункціональні програми, які працюють досить швидко і ефективно в умовах обмежених ресурсів.

Оскільки мова програмування *Assembler* досить складана та вимагає окремих навичок та вмінь, було обрано мову програмування *C++*.

Мова *C++* є надбудовою над мовою *C* і дозволяє писати складні програми з використанням об'єктно-орієнтованого програмування. Для розробки програм для *Arduino* зручно використовувати крос-платформний додаток *Arduino IDE*, який дозволяє легко прошивати код на мові *C++* на мікроконтролери.

Як мову програмування для даного диплому обрано мову C++ оскільки однією з основних цілей є забезпечення простоти програмування.

3.2 Налаштування сигналізації програмно

Оскільки дуже важливо донести продукт до кінцевого користувача простим у налаштуванні та прошивці, є потреба розробити скетч по налаштуванню системи сигналізації. Розглянемо лістинг програми детальніше.

Підключення бібліотек датчиків

```
#define WID_ENABLE   0 // Використовуємо watchdog
#define IR_ENABLE    0 //Бібліотека для ІФ приймача
#define BEEP_ENABLE  1 //зумер
#define DHT_ENABLE   1 //Бібліотека датчика DHT11
#define TERM_ENABLE  1 //Бібліотека для датчика температури
#define MOVE_ENABLE  1 //БІбліотека для датчика руху
#define RADAR_ENABLE 1 //Бібліотека мікрохвильового датчика
#define FIRE_ENABLE  0 //Датчик вогню
#define GAS_ENABLE   1 //Датчик газу
```

Рис. 3.1 - Бібліотеки датчиків

На рисунку 3.1 перераховано всі датчики, які можуть бути використані у системі, але насправді нам потрібні лише ті, які використовуються у сигналізації. Тому непотрібні рядки коду можна закоментувати, щоб не використовувати зайву

пам'ять. Для відключення бібліотеки можна замість "1" поставити "0", або навпаки.

Налаштування GPRS, тобто надсилання SMS

```
#define OTCHET_INIT  SET_FLAG_ONE(GPRS_ENABLE); SET_FLAG_ONE(SMS_ENABLE); SET_FLAG_ONE(CONNECT_ALWAYS);
```

Рис 3.2 - Налаштування SMS звітності

Якщо встановлені обидва прапора, пріоритет буде надаватися *GPRS*. Але, якщо *GPRS* з'єднання не встановлено, повідомлення будуть відправлені через *SMS*. Щоб увімкнути або вимкнути обидва прапори, можна використовувати *DTMF*-команди "GPRS_ON_OFF" та "SMS_ON_OFF". Щоб вимкнути будь-який з прапорів, можна записати "SET_FLAG_ZERO".

Перерахунок пінів

На першому місці є пін *RING* модему, який відповідає піну 2 на платі *Arduino*. Далі йде простий перелік пінів, де номер кожного наступного піна на один більший, ніж попереднього. Якщо якийсь модуль не використовується, то його пін не враховується в перерахунку. На рисунку 3.3 наведено перелік пінів. Якщо деякі датчики не використовуються, то відповідні рядки коду можна видалити.

```
enum pins {
  RING_PIN = 2, //2 відслідковує виклики з модему
  POWER_PIN,    //3 відслідковує наявність живлення
  DOOR_PIN,     //4 датчик відкриття дверей
  MOVE_PIN,     //5 всі датчики руху на одному піні
  RADAR_PIN,    //6 RCWL-0516
  FIRE_PIN,     //7 Датчик вогню
  DHT_PIN,      //8 DHT11, датчик температури та вологи
  BEEP_PIN,     //9 зумер
  RECV_PIN =11, //10 ІЧ приймач
  BOOT_PIN      //11 перезавантаження модему
  };
```

Рис. 3.3 - Перерахунок пінів датчиків

Якщо користувач використовує давач, якого немає у списку, то його можна додати, доповнивши SENSORS_INIT. Для цього потрібно вказати наступну інформацію про давач:

- *Пін Arduino, до якого він підключений;*
- *Тип давача, який відповідає переліку типів давачів у файлі "sensor.h";*
- *Унікальне ім'я давача, яке буде відображатися у звітах;*
- *Рівень на цифровому піні давача у стані спокою (LOW або HIGH);*
- *Час на підготовку давача до роботи при старті у секундах. Якщо давач не можна опитувати протягом цього часу, щоб уникнути отримання некоректних даних, то можна вказати більшу кількість секунд. За замовчуванням цей час дорівнює 10 секундам;*
- *Поріг спрацювання аналогового давача. За замовчуванням цей поріг дорівнює 200.*

Додавання датчиків до масиву

```
# define SENSORS_INIT Sensor sensors[5]={ \
    Sensor(DOOR_PIN,    DIGITAL_SENSOR,       "DOOR",    HIGH,   0)      \
    Sensor(MOVE_PIN,    DIGITAL_SENSOR,       "MOVE",    LOW),           \
    Sensor(A0,          ANALOG_SENSOR,        "GAS",     LOW,    120),   \
    Sensor(A1,          TERMISTOR,            "TERM",    LOW,    10, 45),\
    Sensor(DHT_PIN,     DHT11,                "DHT",     LOW,    10, 45);\
```

Рис. 3.4 - Масив датчиків

Налаштування поштової скрині GSM

Для того, щоб отримувати повідомлення на електронну пошту, необхідно налаштувати поштову скриню на *GSM* модулі та вказати адресу отримувача повідомлень. Першим кроком є встановлення *SMTP* сервера пошти, до якого буде підключатися модуль SIM800L. Зручним варіантом є використання *google* пошти, з якою цей модуль чудово працює.

SMTP (Simple Mail Transfer Protocol) - протокол передачі електронної пошти, він відповідає за передачу та доставку повідомлень. Для налаштування *SIM800L* потрібно знати адресу *SMTP*-сервера, порт, логін та пароль від електронної скрині. Для відправлення повідомлень на пошту необхідно також знати адресу отримувача повідомлення.

Також варто зазначити, що в деяких випадках провайдери мобільного зв'язку можуть блокувати вихідні *SMTP*-запити на порт 25, який зазвичай

використовується для відправлення пошти. У цьому випадку необхідно використовувати альтернативний порт, наприклад, 587 або 465.

```
#  define SMTP_SERVER                F("\"smtp-devices.google.com\",465") //поштовий сервер гугл та порт
#  define SMTP_USER_NAME_AND_PASSWORD  F("\"login\",\"password\"")  //логін та пароль від поштової скриньки
#  define SENDER_ADDRESS_AND_NAME      F("\" "login@google.com\",\SIM800L\"") //пошта відправника
#  define RCPT_CC_ADDRESS_AND_NAME     F("\" "login@gmail.com\",\Slava\"") //адреса та ім'я отримувача
```

Рис. 3.5 - Налаштування поштової скриньки

Налаштування на які варто звернути увагу :

```
# define SLEEP_MODE_ENABLE  1 // Дозволяємо режим сну за для економії заряду батареї
# define SERIAL_RATE    115200 // швидкість послідовного порту Serial
# define RESET_COUNT    3   // Скільки раз модем може не відповідати до перезавантаження
# define ALARM_MAX_TIME 60  // тривалість тривоги в секундах, після чого лічильники спрацьовувань обнуляються
```

Рис. 3.5 - Додаткові налаштування

В рядку define SLEEP_MODE_ENABLE можна визначити, чи дозволено переходити в режим економії енергії, який дозволяє продовжити час роботи на батареї, але при цьому система працює повільніше.

Рядок define SERIAL_RATE визначає швидкість роботи GSM модему з оператором, яку можна знайти в Інтернеті.

У рядку define RESET_COUNT потрібно вказати кількість невдалих запитів, яку потрібно зробити GSM модему, перед тим як перезавантажити систему.

В рядку define ALARM_MAX_TIME можна вказати, скільки секунд буде працювати режим тривоги після спрацювання датчиків.

3.3 Підключення додаткового датчика

Припустимо що в кінцевого користувача буде потреба додати до системи сигналізації новий датчик який не використовується у даному прокті.

В такому випадку потрібно перейти до програмного рішення розміщеного у файлі «sensor.h» та доробити код, файл знаходиться в дереві папки “Arduino/labraries/Sensor”

```
// Сюди додаємо типи датчиків
enum {  // датчик з одним цифровим виходом
        DIGITAL_SENSOR,
        // датчик з одним цифровим виходом,
        // з перевіркою від помилкового спрацьовування
        CHECK_DIGITAL_SENSOR,
        // датчик з аналоговим виходом
        ANALOG_SENSOR,
        // бездротовий датчик з іф діодом
        IR_SENSOR,
        // датчик температури - термістор
        TERMISTOR
        //DHT11, DHT21, DHT22 - датчики температури
        // та вологості, оголошені в stDHT.h
```

Рис. 3.6 - Види датчиків

На малюнку 3.6 показані види датчиків які можуть бути додані у разі потреби, в такому випадку також треба допрацювати програмний код.

```
class Sensor
{
  public:

    // volatile означає вказівку компілятора не оптимізувати код її читання,
    // оскільки її значення може змінюватися всередині обробника переривання
    volatile uint8_t start_time;
    volatile uint8_t end_time;
    uint8_t count; // кількість спрацьовувань
    uint8_t type;  // тип датчика
    uint8_t pin;   // пін
    bool level;    // високий та низький рівень піна
    bool prev_pin_state; // попередній стан піна

    Sensor(uint8_t _type, char* sens_name, uint32_t ir_code);
    Sensor(uint8_t _pin, uint8_t _type, char* sens_name, uint8_t pinLevel = LOW, uint8_t start_time_sec = 10, uint32_t alarm_val = 200);
    ~Sensor();

    bool get_pin_state();       // Повертає state = digitalRead(pin). Якщо стан змінилося, збільшує лічильник спрацьовувань на 1.
    uint8_t get_count();        // Повертає лічильник спрацьовувань датчика.
    void get_info(TEXT *str);   // Повертає рядок з ім'ям датчика та числом спрацьовувань
    void get_name_for_type(TEXT *str);

  private:
    bool check;
    int32_t alarm_value; // значення спрацьовування аналогового датчика
    char *name;
```

Рис. 3.7 - Налаштування логіки датчиків

Ця програма є бібліотекою, яка містить інформацію про різні типи датчиків. У бібліотеці можна знайти інформацію про рівні пінів, кількість спрацювань датчиків та час, який потрібен на підготовку датчика до старту в секундах. Якщо час не вказано, то за замовчуванням він дорівнює 10 секундам.

3.4 *DTMF* команди

DTMF команди - це двотональний багаточастотний аналоговий сигнал, який використовується для набору телефонного номера та управління з'єднанням між аналоговим обладнанням, таким як телефонні апарати та АТС. Також вони застосовуються при ручному введенні абонентом команд для різних інтерактивних систем, наприклад, голосового автовідповідача.

У цьому проекті *DTMF* команди використовуються для керування сигналізацією дистанційно за допомогою *SIM800L* модулю. Якщо на *SIM*-карті вже є номери, то їх власники можуть включати та відключати сигналізацію. Адміністратором є номер з ім'ям ADMIN. Якщо на *SIM*-карті немає такого номера, то перший незареєстрований користувач, який подзвонить на нову *SIM*-карту, стане адміністратором, а його номер буде занесений в телефонну книгу. Адміністратор є єдиним користувачем який має можливість відправляти *DTMF* та *SMS* команди. На наступному малюнку будуть зображені *DTMF* команди які використовуються у даній системі сигналізації.

```
enum {
  GUARD_ON = 1,    // 1# - постановка на охорону
  GUARD_OFF,       // 2# - зняття з охорони
  GPRS_ON_OFF,     // 3# - увімкнути/вимкнути GPRS
  SMS_ON_OFF,      // 4# - увімкнути/вимкнути SMS
  TEL_ON_OFF,      // 5# - увімкнути/вимкнути дзвінок при тривозі
  GET_INFO,      // 6# - збирання та відправлення всіх даних датчиків
  EMAIL_ADMIN_PHONE,  // 7# - надсилаємо на пошту номер адміну
  EMAIL_PHONE_BOOK, // 8# - надсилання на пошту телефонної книги
  ADMIN_NUMBER_DEL, // 9# - адмін більше не адмін
  SM_CLEAR,      // 10# - видалити всі номери з симкарти
  MODEM_RESET,     // 11# - перезавантаження модему
  BAT_CHARGE,      // 12# - показує заряд батареї у вигляді рядка
                   //  +CBC: 0,100,4200
                   // де 100 - відсоток заряду
                   // 4200 - напруга на батареї у мВ.
  CONNECT_ON_OFF    // 13# - Інвертує флаг CONNECT_ALWAYS
  };
```

Рис. 3.8 - DTMF команди

Модем налаштований таким чином, що автоматично піднімає трубку при дзвінку з номера адміністратора. Це зроблено з метою можливості використання *DTMF* команд, які дозволяють виконувати певні дії через клавіші на телефоні. Дзвінки з інших номерів будуть автоматично скидатися, щоб не витрачати час на зайві розмови. Адміністратор може використовувати *DTMF* команди, які вказані у списку вище, для виконання певних завдань. Якщо команда прийнята, то сигналізація скине дзвінок і звіт про виконання команди буде надісланий на пошту. Для введення *DTMF* команди потрібно ввести будь-яку цифру та знак #, що вказує на кінець введення команди. Після цього модем завершить дзвінок та виконає отриману команду.

Логіка DTMF команд

```
switch (DTMF[0])
{
  case GUARD_ON:
    if(!GET_FLAG(GUARD_ENABLE))
    {
      if(sensors = new MY_SENS())
      {
        SET_FLAG_ONE(GUARD_ENABLE);

        DEBUG_PRINT(F("RAM free:"));
        DEBUG_PRINTLN(memoryFree()); // друк кількості вільної оперативної пам'яті
      }
    }
    flags_info();
    break;
  case GUARD_OFF:
    if(GET_FLAG(GUARD_ENABLE))
    {
      SET_FLAG_ZERO(GUARD_ENABLE);
      delete sensors;

      DEBUG_PRINT(F("RAM free:"));
      DEBUG_PRINTLN(memoryFree()); // друк кількості вільної оперативної пам'яті
    }
    flags_info();
    break;
  case TEL_ON_OFF:
    INVERT_FLAG(RING_ENABLE);
    flags_info();
    break;
  case GET_INFO:
    if(digitalRead(POWER_PIN))
    {
      if(GET_FLAG(GUARD_ENABLE)) sensors->TimeReset();
      email_buffer->AddText_P(PSTR(" Svet ON."));
    }
```

Рис. 3.9 - частина логіки DTMF команди

Кодї є досить великим і складним, тому для його написання використовується конструкція *switch-case*. Вона дозволяє замінити довгі *if-else* конструкції, які порівнюють змінну з багатьма константними значеннями. У *switch-case* змінна в порівнянні порівнюється зі значеннями, які описані після ключового слова case. Якщо значення змінної відповідає одному з цих значень, то виконується код, який знаходиться після двокрапки у відповідному *case*.

Кожен case містить обробку конкретного запиту, що надійшов від користувача за допомогою *DTM*F команд. Після обробки запиту відправляються відповідні дані користувачу, а за допомогою ключового слова *break* виконання *switch-case* переривається.

3.5 **Код програми для Arduino**

```
#include "modem.h" // підключення бібліотеки модему
void setup() //функція в якій пишеться основа програми
{
#if WTD_ENABLE
  wdt_disable();
  wdt_enable(WDTO_8S);
#endif
  pinMode(RING_PIN, INPUT); //встановлюємо вивід як вхід
  digitalWrite(RING_PIN, LOW); //встановлюємо низький рівень напруги
  pinMode(POWER_PIN, INPUT);
  digitalWrite(POWER_PIN, LOW);
  pinMode(BOOT_PIN, OUTPUT); //встановлюємо вивід як вхід
  digitalWrite(BOOT_PIN, LOW);
  attachInterrupt(1, power, CHANGE); // переривання для POWER_PIN
  phone = new MODEM();
  phone->init();
}
void timer(uint16_t time) //функція виклику таймера
```

```
{
  if(millis() - msec >= time) //функція відліку таймера
  {
    msec = millis();
    DEBUG_PRINT('.');
    if(GET_FLAG(GUARD_ENABLE))
    { /// Опитування датчиків ///
      if(sensors->SensOpros())
      {
        ALARM_ON // режим тривоги вкл.
      }
    }
    if(GET_FLAG(ALARM))
    {
      if(AlarmTime) AlarmTime--;
      else // Після закінчення заданого часу ALARM_MAX_TIME
      {
        ALARM_OFF; // Вимикаємо режим тривоги та надсилаємо e-mail. Очищуємо статистику датчиків.
      }
    }
  }
}
```

Наведений код описує основний принцип роботи програми. Програма на Arduino прослуховує запити, які надсилаються до неї від користувачів або давачів, до яких підключена Arduino. Якщо запит знайдений, програма виконує відповідну функцію. Якщо давач спрацював, програма надсилає сигнал на GSM модуль, який повідомляє користувача про спрацювання давача.

4 ПРОЕКТУВАННЯ ПРИЛАДУ ТА ПЕРЕВІРКА ЙОГО НА ПРАЦЕЗДАТНІСТЬ ТА БЕЗПЕЧНІСТЬ

4.1 Опис макету

Макет системи сигналізації включає в себе декілька компонентів. Основним контролером системи є *Arduino Nano*, який взаємодіє з іншими компонентами за допомогою звичайних дротів для плат *Arduino*. Для передачі повідомлень про стан системи використовується *GSM* модуль *SIM800L*.

Для виявлення різних небезпек, таких як відкриті двері, протікання газу, рух людей та зміна температури та вологості використовуються відповідні датчики, а саме: *MC-38, MQ-2 Gas Sensor Module, HC-SR501, DHT11*. Якщо будь-який з цих датчиків зафіксує небезпеку, то *Arduino Nano* обробляє цю інформацію та активує зумер звукового оповіщення *KY-012*.

Для зарядки акумуляторної батареї використовується контролюючий модуль заряду *TP4056*, а для стабілізації роботи системи використовуються резистор з опором 10 кОм і розсіювальною потужністю 0.125 Вт, а також конденсатор на 1000 мікрофарад.

Для підключення компонентів між собою використовуються звичайні дроти для плат *Arduino*. Однак, не доцільно використовувати більш навантажені дроти, оскільки вся схема працює до 5 В.

Для збору системи сигналізації був використаний корпус з поліпропілену, який був обраний для розміщення модулів. Для монтажу готової системи у корпусі було використано термоклей, а під модулями датчиків були вирізані отвори у корпусі для зручного підключення.

Оскільки цей макет є лише прототипом, габаритні розміри не є обмеженням і елементи можна розмістити так, як буде зручно та в логічній послідовності згідно блок-схеми пристрою. Цей прототип дозволяє перевірити працездатність периферії та взаємодію програмного коду з макетом, що дає змогу зрозуміти, які можна внести зміни або що можна додати для покращення функціональності на базі прототипу.

4.2 Перевірка працездатності

Перевірка працездатності полягає у підключенні готової системи сигналізації та програмного забезпечення. Систему потрібно встановити у захищене місце, щоб уникнути пошкоджень та потрапляння вологи. Перевірка проводиться, щоб переконатися, що всі модулі працюють без помилок та надають коректну інформацію. Для перевірки використовується смартфон. При транспортуванні готового модуля потрібно бути обережним, щоб не пошкодити елементи. Підтверджено, що застосовані рішення задовольняють завдання до дипломного проекту, але для покращення роботи системи можна вдосконалити механізм підключення до додатку на смартфоні та підключення до *Wi-Fi* мережі.

4.3 Перевірка безпеки пристрою та рекомендації

Безпечність використання цієї сигналізації залежить від правильної установки та належного використання компонентів. Необхідно слідкувати за тим, щоб всі компоненти були належно замонтовані та підключені.

Компоненти, які пов'язані з електричним живленням, такі як резистор та конденсатор, повинні бути встановлені відповідно до їх розрахункових значень та забезпечені достатнім відведенням тепла.

Датчики, такі як датчик газу та датчик руху, повинні бути розташовані відповідно до інструкцій виробника та необхідно використовувати захисні кожухи, якщо вони є необхідними для захисту від пошкоджень.

Зумер звукового оповіщення повинен бути налаштований на відповідну гучність, щоб не заважати оточуючим.

Контролюючий модуль заряду акумуляторної батареї повинен бути підключений до джерела живлення, яке відповідає вимогам виробника.

З метою пожежної безпеки необхідно уникати підключення пристрою до перевантажених електричних мереж, а також уникати використання некоректних джерел живлення. Крім того, пристрій повинен бути розташований відповідно до вимог пожежної безпеки та не перешкоджати вихідним шляхам з будівлі.

Рекомендації для підвищення безпечності використання включають ретельну перевірку підключення та належного монтажу всіх компонентів, використання кільцевих клем для підключення дротів та захисних кожухів для датчиків, а також регулярну перевірку стану живлення та заряду.

ВИСНОВКИ

1. Було проведено аналіз існуючих рішень систем сигналізації від різних виробників. Оглянуто їх основні характеристики, переваги та недоліки.Основними недоліками можна назвати вартість пристроїв та неможливість їх удосконалення.
2. Було розроблену схему та обрано основні компонети з яких буде складатись система сигналізації, розглянуто їх переваги, особливості та недоліки.
3. Створено макет для демонстрації працездатності розробленого програмного забезпечення. Проведено роботу по тестуванню П.З. результати якаї виявились успішними.
4. Розроблена охоронна стаціонарна система з *GSM* модулем.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сигналізації Ajax StarterKit - Ajax StarterKit 2 Білий - купити в інтернет-магазині | Ціна | Київ, Дніпро, Харків, Одеса, Львів – PIPL.UA: Безпека для людей

2. Сигналізація *"PiPo WL-99CGST "* - Комплект бездротовий GSM+ Wi-Fi сигналізації PiPo WL-99CGST – низькі ціни, кредит, оплата частинами в інтернет-магазині ROZETKA | Купити в Україні: Києві, Харкові, Дніпрі, Одесі, Запоріжжі, Львові

3. Сигналізація *"CoVi Security GSM-200 Kit Wi-Fi Tuya Smart "* - Комплект сигналізації CoVi Security GSM-200Kit Wi-Fi Tuya Smart - купити в інтернет-магазині Фортер™: ціни, відгуки, фото, характеристики (forter.com.ua)

4.Atmega328 - Microchip Atmega328P: все, що вам потрібно знати про цей MCU | Безкоштовне обладнання (hwlibre.com)

5. Arduino Nano - Nano | Arduino Documentation

6. З'єднання Arduino Nano та SIM800L - In-Depth: Send Receive SMS & Call with SIM800L GSM Module & Arduino (lastminuteengineers.com)

7. SIM800L - GSM модуль на SIM800L купити в Києві та Україні (arduino.ua)

8. Датчик відкриття дверей MC-38 - Датчик відкриття дверей MC-38 купити в Києві та Україні (arduino.ua)

9. Датчик газу *MQ2* - Модуль датчика дыма MQ-2 купити в Києві та Україні (arduino.ua)

10. Датчик руху HC-SR501 - ІЧ датчик руху для Arduino HC-SR501 купити в Києві та Україні

11. Датчик вологості та температури DHT11 - Датчик вологості та температури DHT11 (V2) купити в Києві та Україні (arduino.ua)

12. Активний зумер - Модуль з динаміком KY-012 (активний) купити в Києві та Україні (arduino.ua)

13. Модуль TP4056 - Зарядний модуль TP4056 Type-C з функцією захисту акумулятора купити в Києві та Україні (arduino.ua)

14. Резистор 10k МЛТ-0.125 - Резистор МЛТ-0,125 10 кОм 5-10%: продаж, ціна у Луцьку. Резистори від "ЕлектроПриладТехСервіс" - 1110371019 (epts.com.ua)

15. Конденсатор - Конденсатор 1000 мкф 16в 105 °C KM 1000uf 16v Ø 10x16 mm CapXo... - 6 ₴, купити на ІЗІ (67197119) (izi.ua)

16. Програмування на Ардуіно - Программирование Ардуино (arduino.ua)

17. DTMF команди -

18. Скетч для прошивки Arduino -

https://drive.google.com/drive/folders/1i3GsW53dEzHK7rNx7Mw8vqQ9Ln6EIuGq?usp=share_link

ДОДАТОК 1

```
#include "modem.h"
#if WTD_ENABLE
#include <avr/wdt.h>
#include <stdint.h>
uint8_t mcusr_mirror __attribute__ ((section (".noinit")));
void get_mcusr(void) \
__attribute__((naked))\
__attribute__((used)) \
__attribute__((section(".init3")));
void get_mcusr(void)
{
mcusr_mirror = MCUSR;
MCUSR = 0;
wdt_disable();
}
#endif

static uint32_t msec = 0;
static uint8_t AlarmTime = 0;
extern uint8_t flags;
MODEM *phone;
MY_SENS *sensors = NULL;

// Активуємо прапорець тривоги для збору інформації та відправки e-mail
```

```
#define ALARM_ON

if(!GET_FLAG(ALARM)){SET_FLAG_ONE(ALARM);AlarmTime=ALARM_MAX_TIM
E;p

hone->ring_to_admin(); \

phone->email_buffer->AddText_P(PSTR(" ALARM!"));sensors-

>GetInfo(phone->email_buffer);DEBUG_PRINTLN(phone->email_buffer-
>GetText());}


// Після закінчення часу ALARM_MAX_TIME обнулюємо прапорець тривоги та
відправляємо e-mail з показниками датчиків

#define ALARM_OFF

{SET_FLAG_ZERO(ALARM);if(GET_FLAG(GUARD_ENABLE)){phone-
>email_buffer-

>AddText_P(PSTR(" ALL:"));sensors->GetInfo(phone->email_buffer); \

sensors->Clear();DEBUG_PRINTLN(phone->email_buffer->GetText());}}


void power()

{

SET_FLAG_ONE(INTERRUPT);

}


void setup()

{

#if WTD_ENABLE

wdt_disable();

wdt_enable(WDTO_8S);

#endif

pinMode(RING_PIN, INPUT);
```

```
digitalWrite(RING_PIN, LOW);
pinMode(POWER_PIN, INPUT);
digitalWrite(POWER_PIN, LOW);
pinMode(BOOT_PIN, OUTPUT);
digitalWrite(BOOT_PIN, LOW);
// Переривання для POWER_PIN
attachInterrupt(1, power, CHANGE);
phone = new MODEM();
phone->init();
}

void timer(uint16_t time)
{
if(millis() - msec >= time)
{
msec = millis();
DEBUG_PRINT('.');
if(GET_FLAG(GUARD_ENABLE))
{ /// Опитування датчиків ///
if(sensors->SensOpros())
{
ALARM_ON // Режим тривоги увімкнено
}
}
if(GET_FLAG(ALARM))
{
if(AlarmTime) AlarmTime--;
```

```
else // Після закінчення заданого часу ALARM_MAX_TIME
{
ALARM_OFF; // Вимикаємо режим тривоги та відправляємо e-mail. Очищуємо статистику датчиків
        }
      }
   }
}

void loop()
{
   while(1)
   {
#if WTD_ENABLE
      wdt_reset();
#endif
      phone->wiring();
      timer(1000);
   }
}
```

ДОДАТОК 2

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "sms.h"
#include "call.h"
SMSGSM sms;
CallGSM call;
//==================Піни 2 та 3 для Підключення GSM модуля
int sensor1=0;
int flag1=0;
boolean started=false;
char smsbuffer[160];
char n[20];
String n1 = "+79170417032";
String input_string = "";
String smsContent = "";
char sirena[] = "Сирена включена!";
char PowerOFF[] = "Знято з охорони";
char smsDv[] ="Увага! Рух на об'єкті!";
char smsW[] ="Постановка на охорону!";
char pos;
char sendsms[160];

void setup() {
  pinMode(10, OUTPUT); // Пасивний зумер (сигнал)
  pinMode(7, OUTPUT);
  Serial.begin(9600);
```

```
 pinMode(4, INPUT); // датчик руху
 // digitalWrite(4, HIGH);
 pinMode(5, INPUT); // ще якийсь датчик
 // digitalWrite(5, HIGH);
 if (gsm.begin(4800)) {
  Serial.println("\nстатус=ГОТОВО");
  started=true;
 } else {
  Serial.println("\nстатус=ОЧІКУЄМО");
 }
}

void loop() {
 //============================Перше спрацювання датчика руху
 if ((digitalRead(5)==HIGH) && sensor1==1 && flag1==0){
  n1.toCharArray(n,20);
  sms.SendSMS(n, smsDv); // Відправляємо СМС про те, що є рух
  sms.DeleteSMS(1);memset(n,0,20);
  char smsbuffer[160]="";
  flag1++;
  delay(5000); // Чекаємо 5 секунд
 }
 //============================Повторне спрацювання датчика руху
 if ((digitalRead(5)==HIGH) && sensor1==1 && flag1==1){
  n1.toCharArray(n,20);
  sms.SendSMS(n, sirena); // Відправляємо СМС про те, що включена сирена
  sms.DeleteSMS(1);memset(n,0,20);
```

```
   char smsbuffer[160]="";
   flag1++;
  }
  if (flag1>=2){
   tone(10, 2780, 200); // Сирена
  }
  pos = sms.IsSMSPresent(SMS_UNREAD); // дивимось непрочитані СМС
  if (pos) { // Якщо непрочитані СМС є, то...
   getsms(); // отримуємо непрочитану СМС
   if (input_string=="0"){
    Serial.print("Знято з охорони!");
    n1.toCharArray(n,20);
    sms.SendSMS(n, PowerOFF);
   sms.DeleteSMS(1);
memset(n,0,20);
char smsbuffer[160]="";
flag1++;

if (flag1>=2){
  tone(10, 2780, 200); // Увімкнути сирену
}

pos = sms.IsSMSPresent(SMS_UNREAD); // Перевірте наявність непрочитаних повідомлень
if (pos) { // Якщо є непрочитані повідомлення, то...
  getsms(); // Отримати непрочитане повідомлення
  if (input_string=="0"){
```

```
   Serial.print(" Знято з охорони!");
   n1.toCharArray(n,20);
   sms.SendSMS(n, PowerOFF);
   sms.DeleteSMS(1);
   memset(n,0,20); // Команда "0" - зняти з охорони
   sensor1=0;
   digitalWrite(7,LOW);
   input_string="";
   char smsbuffer[160]="";
   flag1=0;
  }
  if (input_string=="1"){
   Serial.print("Поставлено на охорону!");
   n1.toCharArray(n,20);
   sms.SendSMS(n, smsW);
   sms.DeleteSMS(1);
   memset(n,0,20); // Команда «1» - поставити на охорону
   sensor1=1;
   input_string="";
   char smsbuffer[160]="";
   flag1=0;
  }
  sms.DeleteSMS(pos); // Видалити повідомлення з SIM-карти
}
```