РЕФЕРАТ

Пояснювальна записка до дипломної роботи бакалавра: 86с., 10 рис., 11 табл., 17 джерел, 3 додатки.

Об’єкт розробки: металодетектор, здатний до пересування під впливом дистанційного керування.

Мета роботи: розробка доступного варіанту мобільного металодетектора з високою прохідністю для широкого споживання.

В проекті виконано:

- Аналіз існуючих видів металодетекторів;
- Підбір найбільш придатних електронних компонентів;
- Розробку індикатора металевих предметів;
- Реалізацію відповідного програмного забезпечення.

Отримані наступні результати: розроблено прототип мобільного металодетектора на базі апаратно обчислювальної платформи Arduino з ґрунтовим індикатором металевих предметів, працюючий під дистанційним керуванням додатку на OS Android, використовуючи бездротову технологію Wi-Fi.

Практичне значення, галузь застосування роботи:

- Пошук металевих предметів у важкодоступних місцях;
- Використання в якості міношукача;
- При видобутку корисних копалин;
- Пошук найбільш перспективних місць для детальних розкопок.

ANDROID, WI-FI, ARDUINO, L298N, ESP8266, XML, JAVA, МЕТАЛОДЕТЕКТОР, TCP, СЕРВЕР, КЛІЄНТ, SOCKET

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АЦП – Аналого-цифровий перетворювач

IDE – Інтегроване середовище розробки

OS – Операційна система

TCP – Протокол керування передачею

IP – Інтернет протокол

SDK – Набір із засобів розробки, утиліт і документації

ПЗ – Програмне забезпечення

ВСТУП

На сьогоднішній день підтверджено, що в 70 країнах існує проблема мін та боєприпасів котрі не розірвалися. Ситуація залишається вкрай серйозною. Міжнародне співтовариство докладає великих зусиль для того, щоб зменшити небезпеку, яку представляють міни та боєприпаси котрі не розірвалися. Подібна робота ведеться по всьому світу, включаючи і територію колишнього Радянського Союзу.

Жити на території, де є небезпека підірватися на міні, дуже важко. Люди змушені змінювати свою поведінку, щоб пристосуватися до нових умов життя. Часто цілі співтовариства не мають доступу до води, до сільськогосподарських угідь і пасовищ. Простіше кажучи, люди втрачають можливість отримувати їжу і дохід, за рахунок якого вони жили.

На Балканах, наприклад, жителі, навіть знаючи, що певна територія небезпечна, все одно туди йдуть. У них немає інших варіантів. Вони ризикують життям, але по-іншому просто не зможуть забезпечити свої сім'ї. Економічна необхідність штовхає людей на те, щоб наражати на небезпеку життя і здоров'я.

В даній роботі розглядається створення доступного металодетектора з високим показником прохідності під дистанційним керуванням на основі клієнт-серверної архітектури, яка дає багато нових можливостей для управління пристроями. Для створення використовуються найдешевші електроні компоненти та широко поширена технологія бездротового зв’язку Wi-Fi, яка є у більшості сучасних гаджетів, працюючих на OS Android.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз металодетекторів

Металодетектор - це електронний пристрій, який визначає присутність металу і інформує нас про це. Металевий предмет, скажімо монета, що знаходиться в землі, сам по собі нічого не випромінює і не видає своєї присутності. Щоб його виявити, необхідно опромінити його радіохвилями і вловити вторинний сигнал. Усі металодетектори засновані на цьому принципі

Коли ви включаєте металодетектор, в котушці протікає змінний електричний струм, що створює навколо котушки електромагнітне поле. Це поле проходить в навколишнє середовище, чи то повітря, ґрунт, вода, камінь, дерево і т. д. Якщо на шляху цього поля виявляється металевий предмет, то на його поверхні виникають так звані вихрові струми. Ці струми утворюють своє електромагнітне поле, яке послаблює поле передавальної котушки. Електронна схема приладу з допомогою котушки вловлює це ослаблення поля, викликане присутністю під котушкою металу, та інформує вас про це тим або іншим способом. Більш складні електронні схеми забезпечують краще розуміння більш слабких вторинних сигналів, більш точно їх обробляють. Тому такі прилади трудомісткі в виготовленні і коштують дорожче. Однак вони, як правило, здатні знаходити об'єкти найбільшої глибині.

В літературі розрізняють наступні підходи до побудови схемотехніки металодетекторів:

1) **BFO - beat frequency oscillation (метод биття).** Вимірюваним параметром є частота LC -генератора, що включає котушку пошукової

голівки. Частота порівнюється з еталонною і отримана різницева частота биття виводиться на звукову індикацію. Схемотехніка приладів достатня проста, котушка не вимагає прецизійного виконання. Робоча частота 40 - 500 кГц. Чутливість BFO - приборів невисока при низькій стабільності роботи і слабкої можливості відбудовуватися від вологого і мінералізованого ґрунту.

Метод BFO застосовувався в серійних іноземних приладах в 60-70 роки. Нині цей метод популярний у радіоаматорів і зустрічається в недорогих приладах російських виробників. Сюди ж можна віднести прилади з прямим виміром частоти, що добре реалізовуються на мікропроцесорах.

2) **TR/VLF - transmitter - reciver / very low frequency (передавач-приймач / дуже низька частота).** Пошукову голівку утворюють дві котушки, розташованих в одній площині і збалансованих так, що при подачі сигналу в передавальну котушку на виходах приймальнею присутній мінімальний сигнал. Передавальна котушка часто включається в контур LC - генератора. Вимірюваним параметром є амплітуда сигналу на приймальній котушці і фазове зрушення між переданим і прийнятим синусоїдальними сигналами. VLF - різновид цього методу, коли робоча частота зменшена до 2 - 10 кГц.

VLF - метод дозволяє побудувати високочутливі прилади з хорошим розрізненням металів за рахунок аналізу фазових характеристик. Схемотехніка приладів досить складна, котушки вимагають прецизійного балансування. По цьому методу зараз будуються більшість серійних приладів, у тому числі і комп'ютеризованих. Дискримінація об'єктів і настроєння від грунту в таких приладах робиться порівняно просто за допомогою фазозсувних ланцюгів.

Принцип TR (чи його різновид TR/VLF) передбачає аналіз фазових характеристик сигналу, тому усі вони легко розрізняють чорні і кольорові метали, відбудовуються від сміття і грунту. Ці прилади мають високу чутливість і роздільну здатність, яка залежить від діаметру голівки, - чим голівка більша, тим глибше виявлення, але тим важче шукати дрібні

предмети. Під терміном TR -дискриминация зазвичай розуміється розпізнавання металів в статиці.

3**) RF - radio frequency (радіо частота)** - високочастотний варіант TR, де передавальна і приймальна котушки утворюють не плоский трансформатор, а рознесені в просторі і розташовані перпендикулярно один до одного. Приймальна котушка приймає відбитий від металевої поверхні сигнал, що випромінюється передавальною котушкою. Цей метод використовується в глибинних приладах і характеризується нечутливістю до дрібних об'єктів і відсутністю розрізнення металів.

4) **PI - pulse induction (імпульсна індукція).** У приладах цього типу котушка пошукової голівки не є частиною коливального контура. У неї від запускаючого генератора подається імпульсний сигнал. Аналізованим параметром є час закінчення перехідного процесу (положення заднього фронту імпульсу напруги). До конструкції котушки не пред'являється особливих вимог. Відмінними рисами цього методу є: низька робоча частота дотримання імпульсів (5-600 Гц), велике споживання енергії, нечутливість до грунту, погане розпізнавання металів. PI -метод часто використовується в підводних приладах для послаблення впливу води.

5) **OR - off resonance (зрив резонансу).** Аналізованим параметром є амплітуда сигналу на котушці коливального контура, налагодженого близько до резонансу з сигналом, що подається на нього, від генератора. Поява металу в полі котушки викликає або досягнення резонансу або відхід від нього, залежно від виду металу, що призводить до збільшення або зменшення амплітуди коливань на котушці. Цей метод також як і BFO розроблявся радіоаматорами, але відомостей про його використання в серійних приладах для пошуку скарбів не виявлено.

У міру ускладнення конструкції приладу і збільшення його вартості покращується здатність приладу розпізнавати металевий предмет без викопування. При відмінності вартості у декілька разів чутливість детекторів збільшується трохи (найчастіше вона складає 20 - 45 см для монет і близько 1

- 2.0 м для великих знахідок). Проте і складні прилади, оснащені процесорами, можуть дати дуже приблизне ув'язнення про метал і глибину знахідки.

1.2 Аналіз існуючих аналогів

1.2.1 RC Moblile Metal

RC Moblile Metal – це не велика модель з дистанційним керуванням (до 15 метрів) та доступною ціною (близько 35 $). Ця модель має всюдихідні колеса та маленькі габарити. RC Moblile Metal має не великий детектор металу низького класу. Основні характеристики RC Moblile Metal наведні у таблиці 1.1

Таблиця 1.1 – Характеристики RC Moblile Metal

Особливість	Звуковий сигнал при виявленні металу
Дальність керування	15 м
Вага	1,72 кг.
Габарити	419 x 273 x 178 мм.
Середня вартість	34 $

1.3 Цільова OS Android

Android – операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA).

Платформа Android містить багатий набір мережевих засобів. У таблиці 1.2 наведені деякі пакунки пов'язані з мережевим можливостям, які присутні в SDK Android.

Таблиця 1.2 – Мережеві пакети SDK Android

Пакет	**Опис**
java.net	Містить класи, пов'язані з мережевими функціями, в тому числі сокети потоків і датаграмм, протокол IP, а також спільні засоби для роботи з HTTP. Це багатоцільовий ресурс для роботи з мережами. Користуючись цим знайомим пакетом, досвідчені Java-розробники зможуть відразу ж приступити до створення додатків.
java.io	Цей пакет є надзвичайно важливим, хоча і не відноситься безпосередньо до мереж. Його класи використовуються сокетами та сполуками, що містяться в інших пакетах Java. Вони використовуються також для обміну з локальними файлами (що часто відбувається при взаємодії з мережею).
java.nio	Містить класи, які служать буфером для певних типів даних. Зручний для організації мережевого зв'язку між двома кінцевими точками засобами Java.
org.apache.*	Набір пакетів, які забезпечують точний контроль і функції для HTTP-комунікацій. Це Apache - знайомий і популярний Web-сервер з відкритим вихідним кодом.
android.net	Містить додаткові сокети доступу до мережі на додаток до основних класів java.net. *. Цей пакет включає в себе клас URI, який часто використовується в розробці додатків Android, що виходять за рамки традиційних мережевих функцій.
android.net.http	Містить класи для роботи з сертифікатами SSL.
android.net.wifi	Містить класи для реалізації всіх аспектів WiFi (802.11 Wireless Ethernet) на платформі Android. Не всі пристрої оснащені можливостями WiFi, особливо з огляду на, що Android пробиває собі дорогу в сегмент "розкладачок" від таких виробників стільникових телефонів, як Motorola і LG.
android.telephony.gsm	Містить класи, необхідні для управління (текстовими) повідомленнями SMS і для їх передачі. Згодом, ймовірно, з'являться додаткові пакети, що надають аналогічні функції в не-GSM мережах, таких як CDMA, щось на зразок android.telephony.cdma.

У 84 % смартфонів, проданих у 3-ому кварталі 2014 року, була встановлена операційна система Android [17].

У березні 2017 року ОС Android стала найпопулярнішою ОС, з якої виходили в інтернет. Так з 37,93% користувачів заходили в інтернет із Android'a, а з Windows лише 37,91% користувачів. В Азії показники ще вищі — 52,2% і 29,2% відповідно. [17].

1.4 Технічне завдання на розробку

Розробити металодетектор з можливістю самостійного пересування, хорошою прохідністю та невеликими габаритами. Для реалізації прототипу детектора метала використовувати метод BFO. Обчислювальна платформа повинна бути на базі Arduino.

Для керування пристроєм необхідно створити додаток на базі операційної системи Android. Використовувати клієнт-серверну архітектуру взаємодії між сервером (мобільним детектором) та клієнтом (додатком на OS Android) з використанням широко поширеної бездротової технологією Wi-Fi на базі стеку протоколів TCP/IP.

Головною вимогою до апаратних компонентів є вартість та доступність на міжнародному ринку.

1.5 Висновки

У цьому розділі були проаналізовані існуючі методи індикації наявності металу та знайдені аналоги пристрою. Як можна побачити, знайдено лише один відомий аналог для широкого споживання, працюючий під дистанційним керуванням. Також у цьому розділі розглядались мережеві пакети, широко відомої операційної системи Android.

2 АПАРАТНА ЧАСТИНА

У цьому розділі розглядається апаратна частина проекту. Відповідно до технічного завдання на розробку, були обрані і розглянуті необхідні електронні компоненти та модулі. Основними критерієм вибору стали вартість та доступність компонентів на ринку. Також у розділі представлені схеми підключення компонентів і конструкція мобільного металодетектора.

2.1 Огляд апаратно обчислювальної платформи і модулів

2.1.1 Arduino Leonardo

Arduino Leonardo – апаратно обчислювальна платформа на базі мікроконтролеру ATmega32u4. Платформа має 20 цифрових входів / виходів (7 з яких можуть використовуватися як виходи ШІМ і 12 як аналогові входи), кварцовий генератор 16 МГц, роз'єм мікро-USB, силовий роз'єм, роз'єм ICSP і кнопку перезавантаження. Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера AC / DC або батареї. Платформа Arduino Leonardo зображена на рисунку 2.1

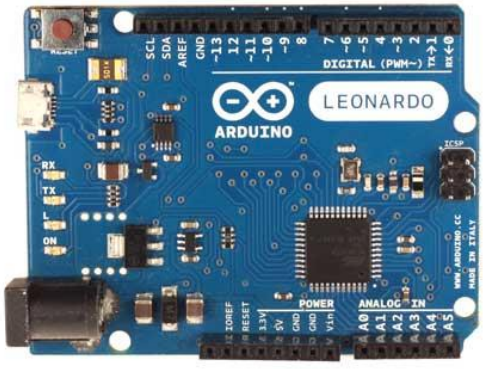

Рисунок 2.1 – Arduino Leonardo з роз'ємами

У таблиці 2.1 наведені основні характеристики Arduino Leonardo.

Таблиця 2.1 – Характеристики Arduino Leonardo

Мікроконтролер	ATmega32u4
Робоча напруга	5 В
Вхідна напруга (рекомендована)	7-12 В
Вхідна напруга (гранична)	6-20 В
Цифрові входи / виходи	20 (7 з яких можуть використовуватися як виходи ШІМ)
Аналогові канали	12
Постійний струм через вхід / вихід	40 мА
Постійний струм для виведення 3.3В	50 мА
Флеш-пам’ять	32 КБ з яких 4 КБ використовуються для завантажувача
ОЗП	2.5 КБ
Тактова частота	16 МГц
Габарити	69 х 54 мм.
Середня ціна	20 $

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

Зовнішнє живлення може подаватися через перетворювач напруги AC / DC або акумуляторною батареєю. Перетворювач напруги підключається за допомогою роз'єму 2.1 мм з центральним позитивним полюсом. Проводи від батареї підключаються до виводу GND і VIN роз'єму живлення.

Платформа може працювати при зовнішньому живленні від 6 В до 20 В. При напрузі живлення нижче 7 В, вихід 5V може видавати менше 5 В, при цьому платформа може працювати нестабільно. При використанні напруги вище 12 В регулятор напруги може перегрітися і пошкодити плату. Рекомендований діапазон від 7 В до 12 В.

Виводи живлення:

- VIN. Використовується для подачі живлення від зовнішнього джерела;
- 5V. Джерело напруги, що використовується для живлення мікроконтролера і компонентів на платі;

- 3v3. Напруга 3.3 В, яка генерується вбудованим регулятором на платі;
- GND. Загальний контакт;
- IOREF. Вихід з робочою напругою входів / виходів плати. Для Leonardo ця напруга дорівнює 5 В.

Входи і виходи:

- Послідовна шина: 0 (RX) і 1 (TX). Виводи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виводи підключені до відповідних виводів мікросхеми послідовної шини ATmega32u4 USB-to-TTL. У Leonardo клас Serial відноситься до послідовного з'єднання USB CDC. Послідовне з'єднання через виводи 0 і 1 здійснюється через клас Serial1;
- TWI: 2 (SDA) і 3 (SCL). За допомогою цих виводів здійснюється зв'язок I2C (TWI);
- ШІМ: 3, 5, 6, 9, 10, 11 і 13. Будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт;
- LED: 13. Вбудований світлодіод, підключений до цифрового виводу 13. Якщо значення на виводі має високий потенціал, то світлодіод горить;
- Аналогові входи: A0-A5, A6-A11. Leonardo має 12 аналогових входів, помічених від A0 до A11. Всі аналогові входи можуть працювати в режимі цифрових входів / виходів. Роздільна здатність аналогових входів - 10 біт, тобто 1024 різних значень. За замовчуванням значення на аналогових входах вимірюється від землі (0) до 5 Вольт, верхня межа діапазону може бути змінена за допомогою AREF входу.

2.1.2 Драйвер двох двигунів L298N

Модуль драйвера двигунів L298N (рис.2.2) дозволяє управляти двома моторами постійного струму, або кроковим двигуном до 2 Ампер.

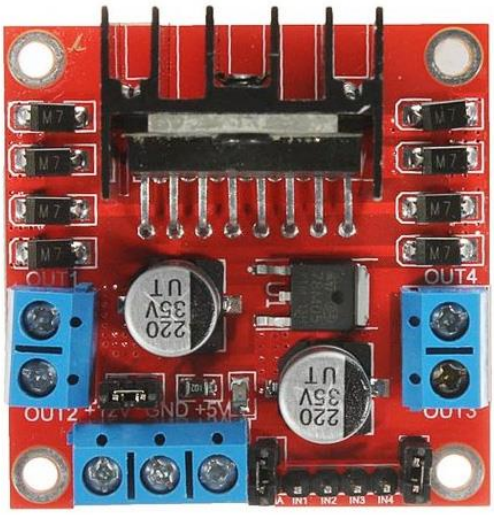

Рисунок 2.2 – Модуль драйвера двигунів L298N

Драйвером двигуна управляють за допомогою контролера Arduino або іншого мікропроцесорного керуючого пристрою. Джампери на платі призначені для вмикання і вимикання вихідних сигналів модуля. Джампер ENA вмикає і вимикає живлення виходів out1 і out2, а джампер ENB – виходів out3 і out4. Якщо встановити джампер ENA, виходами out1 і out2 управлятимуть входи In1 і In2. Якщо встановити джампер ENB, то виходами out3 і out4 управлятимуть входи In3 і In4. Входи широтно-імпульсної модуляції In2 і In3 дозволяють плавно змінювати швидкість обертання двигуна. Входи In1 і In4 визначають напрям обертання двигуна. Також плата має джампер для перемикання напруги живлення 5В вбудованої логіки. Коли цей джампер ввімкнений, логічна частина отримує живлення від внутрішнього перетворювача модуля. Якщо джампер вимкнений, живлення постачає зовнішнє джерело.

У L298N є такі інтерфейси для підключення мікроконтролера, живлення і пристроїв, якими він повинен управляти:

- інтерфейс з трьома контактами-затискачами використовується для підключення живлення. Контакти позначені: +5V (напруга живлення вбудованої логіки), +12V (напруга для керованих пристроїв), GND (загальний контакт);
- інтерфейс із чотирма штировими контактами призначений для підключення мікроконтролера. Контакти позначені: In1, In2, In3, In4;
- інтерфейс A і B для підключення керованих пристроїв з двома контактами-затискачами. У інтерфейсу A контакти позначені як OUT1, OUT2, а у інтерфейсу B – OUT3 і OUT4.

Модуль отримує живлення від контролера Arduino, іншого мікропроцесорного керуючого пристрою або зовнішнього джерела (батареї, блоку живлення).

Характеристики L298N наведені у таблиці 2.2

Таблиця 2.2 – Характеристики L298N

Напруга живлення вбудованої логіки	5 В
Струм споживання вбудованої логіки	0-36 мА
Напруга живлення драйвера	5-35 В
Робочий струм драйвера	2 А (піковий струм – 3 А)
Максимальне споживання енергії	25 Вт
Габарити	43,5 х 43,2 х 29,4 мм.
Вага	26 г.
Середня вартість	5 $

2.1.3 Wi-Fi модуль ESP8266

ESP8266 (рис.2.3) це дешеві широко розповсюджені модулі Wi-Fi. Вони складаються з самодостатнього мікроконтролера з GPIO (дискретними входами-виходами), аналоговим входом, портами паралельного зв'язку, I2C, SPI, та блоком Wi-Fi зв'язку.

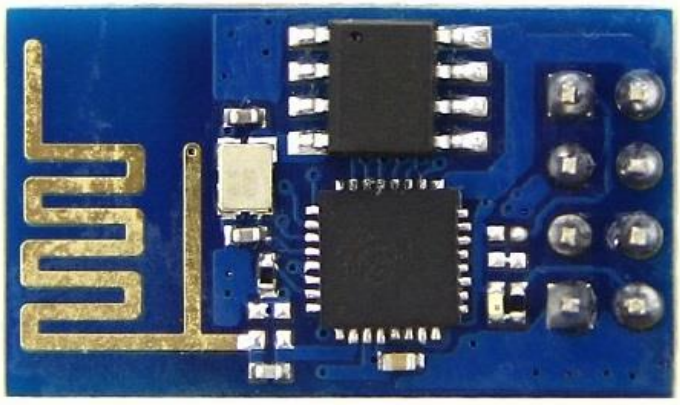

Рисунок 2.3 –Wi-Fi модуль ESP8266

ESP8266 може працювати як в ролі точки доступу так і кінцевої станції. При нормальній роботі в локальній мережі ESP8266 конфігурується в режим кінцевої станції. Для цього пристрою необхідно задати SSID Wi-Fi мережі і, в закритих мережах, пароль доступу [13]. Для початкового конфігурування цих параметрів зручний режим точки доступу. У режимі точки доступу пристрій видно при стандартному пошуку мереж в планшетах і комп'ютерах. Залишається підключитися до пристрою, відкрити HTML сторінку конфігурації і задати параметри мережі. Після чого пристрій штатно підключиться до локальної мережі в режимі кінцевої станції.

У разі виключно місцевого використання можливо завжди залишати пристрій в режимі точки доступу, що знижує необхідні зусилля користувача для його налаштування.

Керуючий пристрій може спілкуватися з ESP8266 через UART (Serial-порт) за допомогою набору AT-команд. Тому робота з модулем тривіальна для будь-якої плати з UART-інтерфейсом.

Робота над прийомом і передачею даних виглядає, як взаємодія з TCP-сокетом або з Serial-портом.

Характеристики ESP8266 наведені у таблиці 2.3

Таблиця 2.3 – Характеристики ESP8266

Бездротовий інтерфейс	Wi-Fi 802.11 b/g/n 2,4 ГГц
Режими	P2P (клієнт), soft-AP (точка доступу)
Номінальна напруга	3,3 В
Максимальний струм споживання	220 мА
Портів вільного призначення	2
Габарити	21 x 13 мм.
Середня вартість	2.5 $

2.2 Індикатор металевих предметів

2.2.1 Принцип дії

Принцип роботи металодетектора заснований на властивості металевих предметів вносити загасання в LC-контур автогенератора. Режим автогенератора встановлюють поблизу точки зриву генерації, і наближення до його контуру металевих предметів (в першу чергу феромагнітних) помітно знижує амплітуду коливань або призводить до зриву генерації.

Якщо сигналізувати наявність або відсутність генерації, то можна визначати місце розташування цих предметів.

Такий тип металодетектора вважається ґрунтовим. Параметри по виявленню металевих предметів наведені у таблиці 2.4 [12].

Таблиця 2.4 – Параметри по виявленню предметів металодетектором [12]

великі металеві предмети	10 см
труба діаметром 15 мм	8 см
гвинт М5 × 25	4 см
гайка М5	3 см
гвинт М2.5 × 10	1.5 см

2.2.2 Принципова схема

Схема пристрою наведена на рисунку 2.4. Пристрій має звукову та світлову індикацію виявленого предмета. На транзисторі VT1 зібраний ВЧ автогенератор з індуктивним зв'язком. Контур L1C1 визначає частоту генерації (близько 100 кГц), а котушка зв'язку L2 забезпечує необхідні умови для самозбудження. Резисторами R1 і R2 можна встановлювати режими роботи генератора.

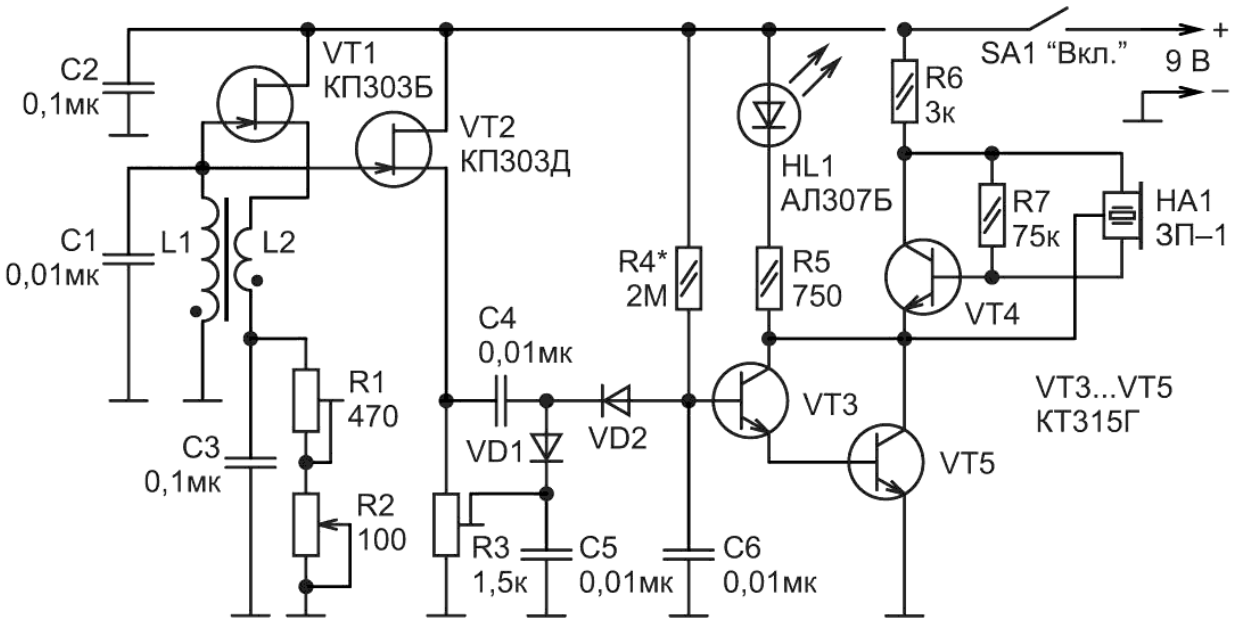

Рисунок 2.4 – Принципова схема індикатора металевих предметів [12]

На транзисторі VT2 зібраний повторювач, на діодах VD1, VD2 - випрямляч, на транзисторах VT3, VT5 - підсилювач струму, а на транзисторі VT4 і BF1 - звуковий сигналізатор.

При відсутності генерації струм, що протікає через резистор R4, відкриває транзистори VT3 і VT5, тому світлодіод HL1 буде світити, а BF1 видавати звуковий сигнал на резонансній частоті (2-3 кГц).

Якщо ВЧ автогенератор буде працювати, то його сигнал з виходу повторювача випрямляється, і мінусова напруга з виходу випрямляча закриє транзистори VT3, VT5. Світлодіод згасне, звучання сигналізатора припиниться.

При наближенні контуру до металевого предмету амплітуда коливань в ньому буде зменшуватися, або генерація зірветься. В цьому випадку мінусова напруга на виході детектора буде знижуватися і через транзистори VT3, VT5 почне протікати струм.

Світлодіод запалиться, пролунає звуковий сигнал, що вкаже на наявність поблизу контуру металевого предмета.

Паралельно до світлодіода були розведені проводи, для їх подальшого підключення до загально заземлення і аналового входу A0 платформи Arduino Leonardo. Напруга при генерації становить близько 0.4 В, а при зриві генерації не перевищує 2.5 В. Завдяки аналого-цифровому перетворювачу Arduino, можна зчитувати значення напруги на цих проводах і проводити подальші маніпуляції.

Пристрій живиться від батареї із загальною напругою 9 В. Струм становить 3-4 мА, коли світлодіод не горить, і зростає приблизно до 20 мА, коли він запалюється.

Якщо приладом користуватися не часто, то вимикач SA1 можна не встановлювати, подаючи напругу на пристрій підключенням батареї живлення.

2.2.4 Конструкція котушок індуктивності

Конструкція котушки індуктивності автогенератора наведена на рисунку 2.5. На круглий стрижень 1 з фериту діаметром 8-10 мм і проникністю 400-600 надягають паперові гільзи 2 (2-3 шари щільного паперу), на них намотують виток до витка проводом ПЕВ-2 0,31 котушки L1 (60 витків) і L2 (20 витків) - 3.

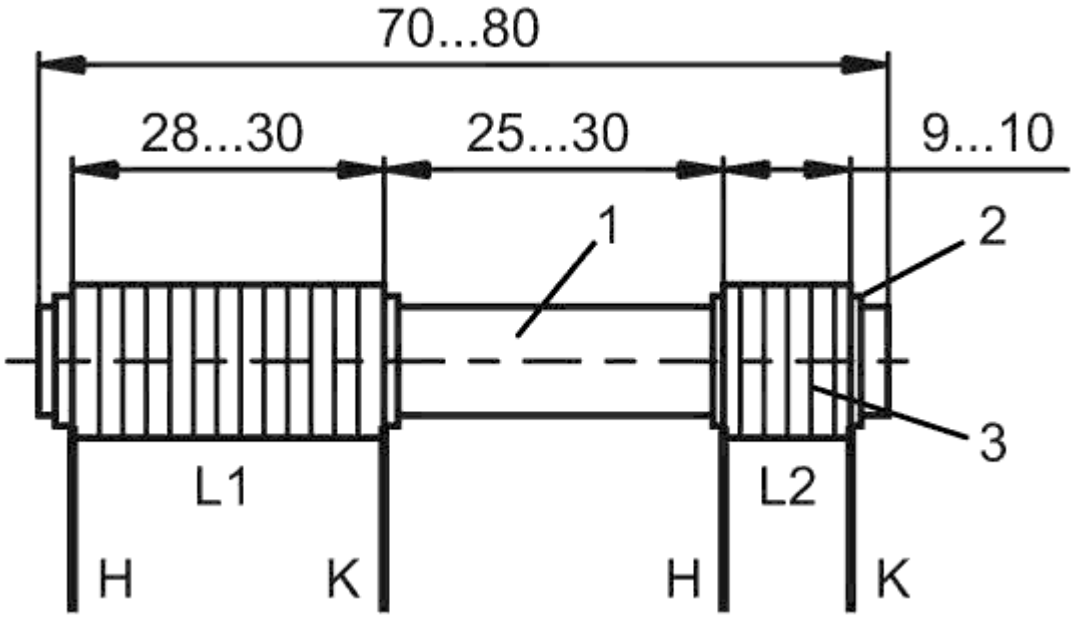

Рисунок 2.5 – Конструкція котушки індуктивності [12]

Крім того, котушка L2 повинна переміщатися по стрижні з невеликим тертям. Обмотку на паперовій гільзі можна закріпити скотчем.

2.2.5 Налагодження

Налагоджувати пристрій рекомендується в наступній послідовності:

– підібрати резистор R4 (для цього тимчасово видалити із виводів діода VD2 і встановити резистор R4 такого максимально можливого

опору, щоб на колекторі транзистора VT5 була напруга 0.8-1 В, при цьому світлодіод повинен світити, а звуковий сигнал лунатиме.

- встановити движок резистора R3 в нижнє за схемою становище і встановити діод VD2, а котушку L2 видалити, після цього транзистори VT3, VT5 повинні закритися (світлодіод згасне);
- акуратно переміщаючи движок резистора R3 вгору за схемою, домогтися відкриття транзисторів VT3, VT5 і включення сигналізації;
- встановити движки резисторів R1, R2 в середнє положення і встановити котушку L2.
- котушку L2 віддалити від L1 і домогтися моменту зриву генерації, а резистором R1 її відновити.
- встановити генератор на межі зриву і перевірити чутливість пристрою.

На цьому налаштування вважається завершеним. Опис пристрою наводиться в [11].

2.3 Конструкція мобільного металодетектора

Основою розробленого мобільного металодетектора є рухома гусенична платформа (рис.2.6). Гумові гусениці мають сильне зчеплення з дорогою, що підвищує прохідність мобільного металодетектора.

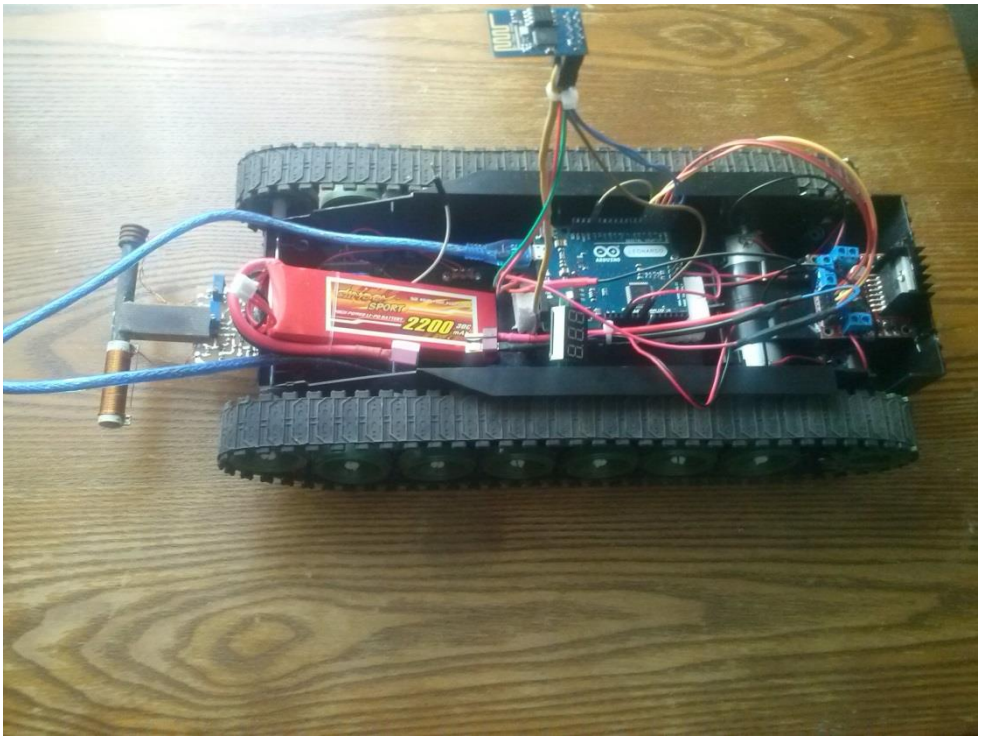

Рисунок 2.6 – Зовнішній вигляд мобільного металодетектора

Характеристики гусеничної платформи мобільного металодетектора наведені у таблиці 2.5

Таблиця 2.5 – Характеристики гусеничної платформи

Двигун	RE-260RA
Номінальна напруга	5-7 В
Габарити	300 х 150 х 76 мм.
Вага	550 г.

2.4 Загальні схеми з'єднання

2.4.1 Макетна схема

Для наочності на рисунку 2.7 представлена макетна схема. Така схема полегшує з'єднання електронних компонентів в єдину систему.

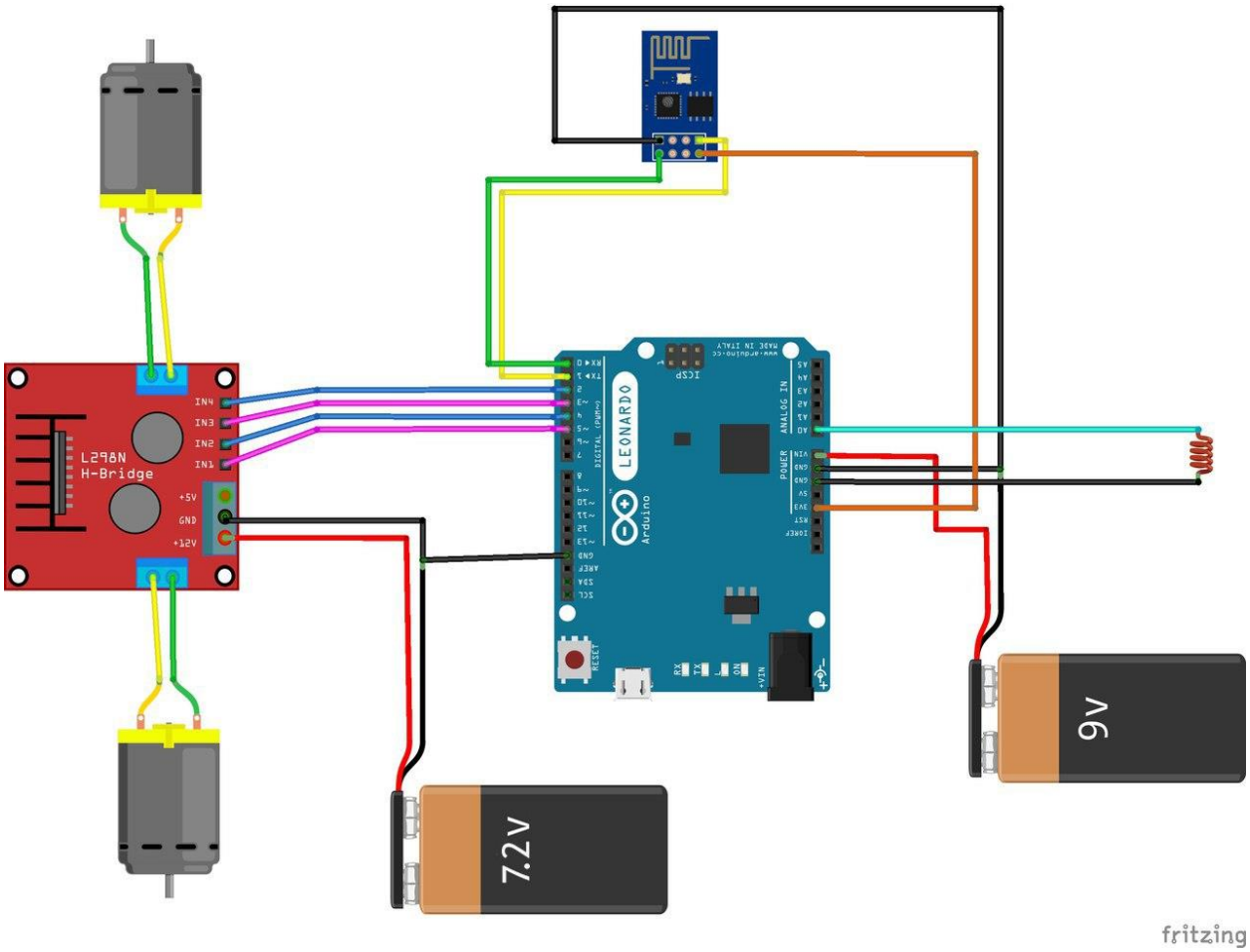

Рисунок 2.7 – Макетна схема з’єднання

2.4.2 Принципова схема

На рисунку 2.8 представлена принципова схема, яка визначає повний склад елементів і зв'язок між ними і дає детальне уявлення про принципи роботи мобільного металодетектора.

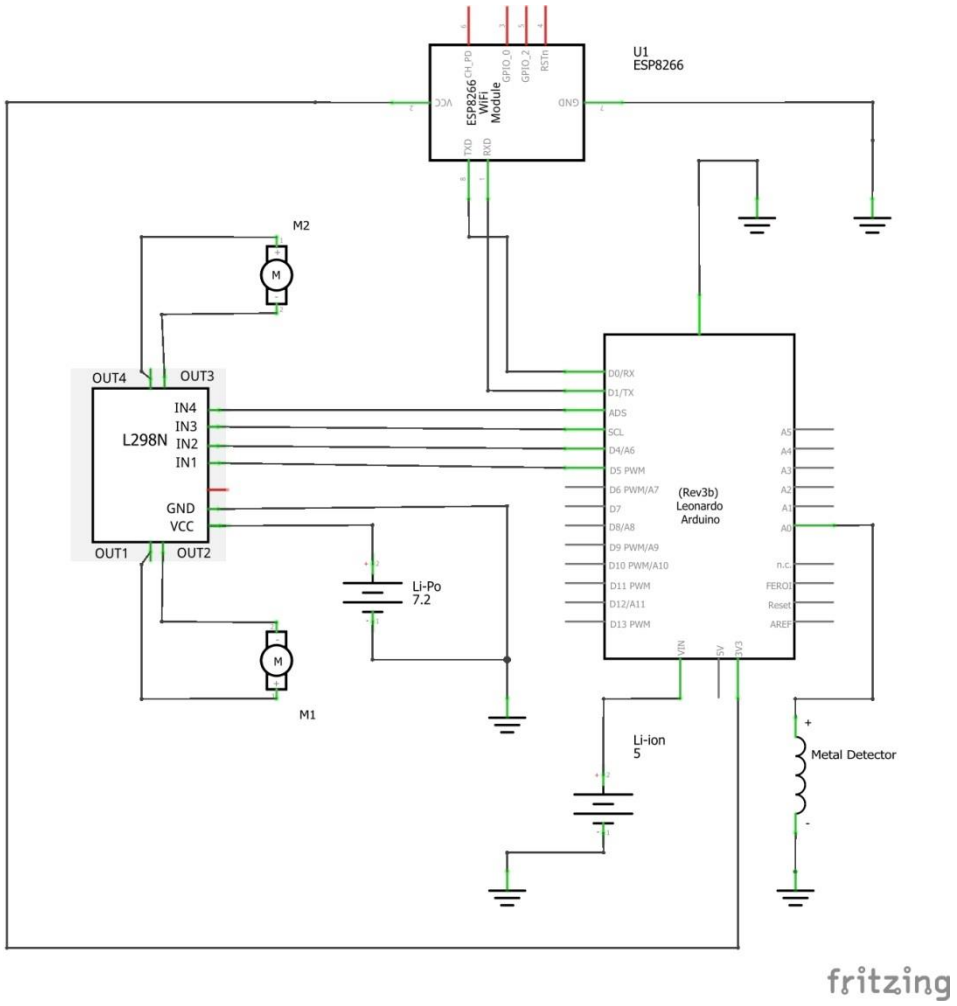

Рисунок 2.8 – Принципова схема з’єднання

2.5 Висновки

У даному розділі були детально розглянуті усі використовувані компоненти мобільного металодетектора. Компоненти системи підбирались за критерієм найнижчої вартості. Також було реалізовано ґрунтовий індикатор металевих предметів.

Завершальною стадією апаратної частини стала компоновка деталей у єдину систему та підготовка металодетектора до наступного етапу розробки.

3 ПРОГРАМНА ЧАСТИНА

При розробці різних додатків, часто стоять завдання щодо передачі даних з мобільного додатку на сервер або інший пристрій. Це можуть бути різні файли, такі як фотографія або відео, дані по датчикам або ж будь-яка інша інформація.

При такому завданні необхідно реалізувати якийсь механізм обміну даними. Найчастіше будуються архітектури, при яких заздалегідь є певний клієнт і сервер. Адже необхідно знати, куди ми будемо відправляти дані і хто їх буде приймати.

Крім вибору, хто буде сервером, а хто клієнтом, необхідно ще знати (або визначити), в якому вигляді будуть надходити дані і як їх можна інтерпретувати.

У цьому розділі розглядається написання програмного забезпечення для мобільного металодетектора (у якості сервера) та додатку для дистанційного керування (у якості клієнта). Також проводиться короткий огляд на засоби програмної розробки.

3.1 Огляд засобів розробки

3.1.1 Arduino IDE

Інтегроване середовище розробки Arduino це багатоплатформовий додаток на Java, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату [16].

Середовище розробки засноване на мові програмування Processing і спроектоване для програмування новачками.

Мова програмування Arduino є стандартним C++ з деякими особливостями, які полегшують написання простий працюючої програми.

Програми, написані на Arduino називаються скетчі і зберігаються в файлах з розширенням ino. Ці файли перед компіляцією обробляються препроцесором Arduino. Також існує можливість створювати і підключати до проекту стандартні файли C ++.

Обов'язкову в C ++ функцію main () препроцесор Arduino створює сам, вставляючи туди необхідні «чорнові» дії.

Програміст повинен написати дві обов'язкові для Arduino функції setup () і loop (). Перша викликається одноразово при старті, друга виконується в нескінченному циклі.

В текст своєї програми (скетчу) програміст не зобов'язаний вставляти заголовки використовуваних стандартних бібліотек. Ці заголовки додасть препроцесор Ардуіно відповідно до конфігурації проекту. Однак призначені для користувача бібліотеки потрібно вказувати.

Менеджер проекту Arduino IDE має нестандартний механізм додавання бібліотек. Бібліотеки у вигляді вихідних текстів на стандартному C ++ додаються в спеціальну папку в робочому каталозі IDE. При цьому назва бібліотеки буде додано до списку бібліотек в меню IDE. Програміст зазначає потрібні бібліотеки і вони вносяться до списку компіляції.

Arduino IDE не пропонує ніяких налаштувань компілятора і мінімізує інші настройки, що спрощує початок роботи для новачків і зменшує ризик проблем.

3.1.2 Android Studio IDE

IDE є необхідним компонентом для розробки програми під мобільні пристрої, так як зібрати і протестувати проект, використовуючи тільки редактор і компілятор стає неможливим. З'явилися системи автоматичного складання (Gradle, Maven), засоби тестування (JUnit, GreenHat), емулятори та багато іншого, що допомагає створити більш якісне ПЗ за коротший час.

Аж до 2012го року у Android не було своєї IDE, тільки набір засобів розробки. Однак на конференції Google IO 2012 була представлена перша бета-версія рідної IDE для Android - Android Studio [15]..

Android Studio створена на основі Intellij IDEA і Android Bundle, увібравши в себе переваги обох продуктів. Підтримує всі засоби роботи з синтаксисом IDEA. Android Studio стала офіційною підтримуваної Google IDE для Android.

Для прискорення розробки додатків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу додатків (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub.

До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.

3.1.3 Мова програмування Java

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems» [14]. В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. У мови Java є багато переваг перед іншими мовами програмування, що дозволяє вирішувати з його допомогою практично будь-які завдання.

Нижче перераховані основні переваги Java:

- Мова Java проста для вивчення. При розробці Java було приділено велику увагу простоті мови, тому програми на Java, в порівнянні з програмами на інших мовах, простіше писати, компілювати, налагоджувати і вивчати.
- Java - це об'єктно-орієнтована мова. Це дозволяє створювати модульні програми, вихідний код яких може використовуватися багаторазово.
- Мова Java не залежить від платформи. Одною з основних переваг мови Java є можливість перенесення програм з однієї системи в іншу. Оскільки програми на Java не залежать від платформи як на рівні вихідного коду, так і на бінарному рівні, їх можна запускати в різних системах.

Широкі можливості Java, простота застосування, незалежність від платформи і вбудовані функції захисту роблять цю мову програмування одною з кращих мов для створення додатків

3.1.4 Мова розмітки XML

XML (Extensible Markup Language) - це мова розмітки документів, що дозволяє структурувати інформацію різного типу, використовуючи для цього довільний набір інструкцій.

XML-документ являє собою звичайний текстовий файл, в якому за допомогою спеціальних маркерів створюються елементи даних, послідовність і вкладеність яких визначає структуру документа і його зміст. Основною перевагою XML документів є те, що при відносно простому способі створення та обробки (звичайний текст може редагуватися будь-яким тестовим процесором і оброблятися стандартними XML аналізаторами), вони дозволяють створювати структуровану інформацію, яку добре "розуміють" комп'ютери.

Сьогодні XML може використовуватися у будь-яких додатках, яким потрібна структурована інформація - від складних геоінформаційних систем, з гігантськими обсягами інформації, що передається до звичайних "однокомпьютерних" програм, що використовують цю мову для опису службової інформації.

3.2 Програмне забезпечення сервера

У якості сервера виступає апаратно обчислювальна платформа на Arduino Leonardo. Платформа Arduino Leonardo не має бездротової технології Wi-Fi, тому спілкування з клієнтом буде проводитись через послідовне з’єднання з модулем ESP8266. Для спілкування Arduino з модулем ESP8266 використовуються AT команди. З повним кодом програми можна ознайомитись у Додатку А.

Перш за все треба налаштувати всі виводи Arduino та запустити режим TCP-сервера модуля ESP8266. Для цього в функцію `void setup()` треба додати наступний код:

```
pinMode(ledData, OUTPUT);
digitalWrite(ledData, HIGH);

pinMode(A0, INPUT);

pinMode (L_IN0, OUTPUT);
pinMode (L_IN1, OUTPUT);
pinMode (R_IN0, OUTPUT);
pinMode (R_IN1, OUTPUT);

Serial.begin(115200);
Serial1.begin(115200);

delay(2000);
clearBuffer();
commandSend("AT+CIPMODE=0");

delay(2000);
clearBuffer();
commandSend("AT+CIPMUX=1");

delay(2000);
clearBuffer();
commandSend("AT+CIPSERVER=1,8888");

clearBuffer();
digitalWrite(ledData, LOW);
```

Для запуску TCP-сервера виконуються наступні команди:

1. Встановлюється режим передачі

AT + CIPMODE = <mode>

mode = 0 - not data mode (сервер може відправляти дані клієнта і може приймати дані від клієнта)

mode = 1 - data mode (сервер не може відправляти дані клієнта, але може приймати дані від клієнта)

2. Встановлюється можливість множинних з'єднань

AT + CIPMUX = <mode>

mode 0 - single connection

mode 1 - multiple connection

3. Запускається сервер на порту 8888:

AT + CIPSERVER = <mode> [, <port>]

mode 0 - to close server

mode 1 - to open server

Функція `clearBuffer()` виконує очистку буферу послідовного з'єднання, завдяки пустому зчитуванню даних.

```
void clearBuffer()
{
  while(Serial1.available() > 0) Serial1.read();
}
```

Функція `commandSend()` відправляє текстову строку та очікує завершення відправки вихідних даних.

```
void commandSend(String text)
{
  Serial1.println(text);
  Serial1.flush();
  delay(50);
}
```

Основна функція програми `void loop()` зчитує вхідний буфер послідовного порта та заносить його в свій буфер команд. Якщо у буфері є якісь існуючі команди, то вони виконуються. Також у функції реалізована відправка даних клієнту, за допомогою AT команди: AT+CIPSEND=0,4, яка відправляє трьохсимвольне значення з метелодетектору (перший символ відводиться під решітку, це робиться для того, щоб розділити потік даних). Відправка даних призводиться раз у 400 мс. Відправку можно деактивизувати / активізувати, якщо у вхідний буфер надійде решітка.

```
void loop()
{
  int buffer[64];
  int b = 0;

  while (Serial1.available() > 0)
  {
    int r = Serial1.read();
    if (((r > 96 && r < 123) || r == 35) && b < 64)
    {
      buffer[b++] = r;
    }
  }

  for (int i = 0; i < b; i++)
  {
    if (buffer[i] == DATA)
    {
      Data = !Data;
      tmpTimeout = 0;
      digitalWrite(ledData, (Data) ? HIGH : LOW);
    }

    if (buffer[i] == LEFT_FORWARD)
    {
      digitalWrite (L_IN0, HIGH);
      digitalWrite (L_IN1, LOW);
    }

    if (buffer[i] == LEFT_BACK)
    {
      digitalWrite (L_IN0, LOW);
      digitalWrite (L_IN1, HIGH);
    }

    if (buffer[i] == LEFT_STOP)
    {
      digitalWrite (L_IN0, LOW);
      digitalWrite (L_IN1, LOW);
    }

    if (buffer[i] == RIGHT_FORWARD)
    {
      digitalWrite (R_IN0, HIGH);
      digitalWrite (R_IN1, LOW);
    }

    if (buffer[i] == RIGHT_BACK)
    {
      digitalWrite (R_IN0, LOW);
      digitalWrite (R_IN1, HIGH);
    }

    if (buffer[i] == RIGHT_STOP)
    {
      digitalWrite (R_IN0, LOW);
      digitalWrite (R_IN1, LOW);
    }
```

```
  }

  if (Data && !tmpTimeout)
  {
    int val = analogRead(A0);
    if (val < 100) val = 100;
    if (val > 999) val = 999;
    Serial1.println("AT+CIPSEND=0,4");
    Serial1.flush();
    delay(100);
    Serial1.print("#");
    Serial1.print(val);
    Serial1.flush();
  }
  else
  {
    delay(100);
  }

  tmpTimeout++;

  if (tmpTimeout > 4)
  {
    tmpTimeout = 0;
  }
}
```

Для зручності, буфер вхідного потоку має цілочисельний тип, тому порівняння виконується з цілочисельним значеннями команд, відповідно до кодів ASCII (рис. 3.1).

0	NUL	16	DLE	32	SPC	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Рисунок 3.1 – Таблиця перетворень ASCII

Усі команди задекларовані наступним чином:

```
#define LEFT_FORWARD 113
#define LEFT_BACK 97
#define LEFT_STOP 122
#define RIGHT_FORWARD 119
#define RIGHT_BACK 115
#define RIGHT_STOP 120
#define DATA 35
```

3.3 Графічний інтерфейс додатку на OS Android

На цьому етапі створюється графічний інтерфейс додатка. Додаток має кнопки: з’єднання, закриття з’єднання, активації / дезактивації отримування даних с детектора метала та управляючі кнопки гусеничної платформи (вперед лівої гусениці, назад лівої гусениці, вперед правої гусениці, назад правої гусениці). Також додаток має поле для виводу логів та поле для виводу даних с металевого детектора. Графічний інтерфейс (рис.3.2) було створено у форматі XML. Розмітка графічного додатку наведена у додатку В.

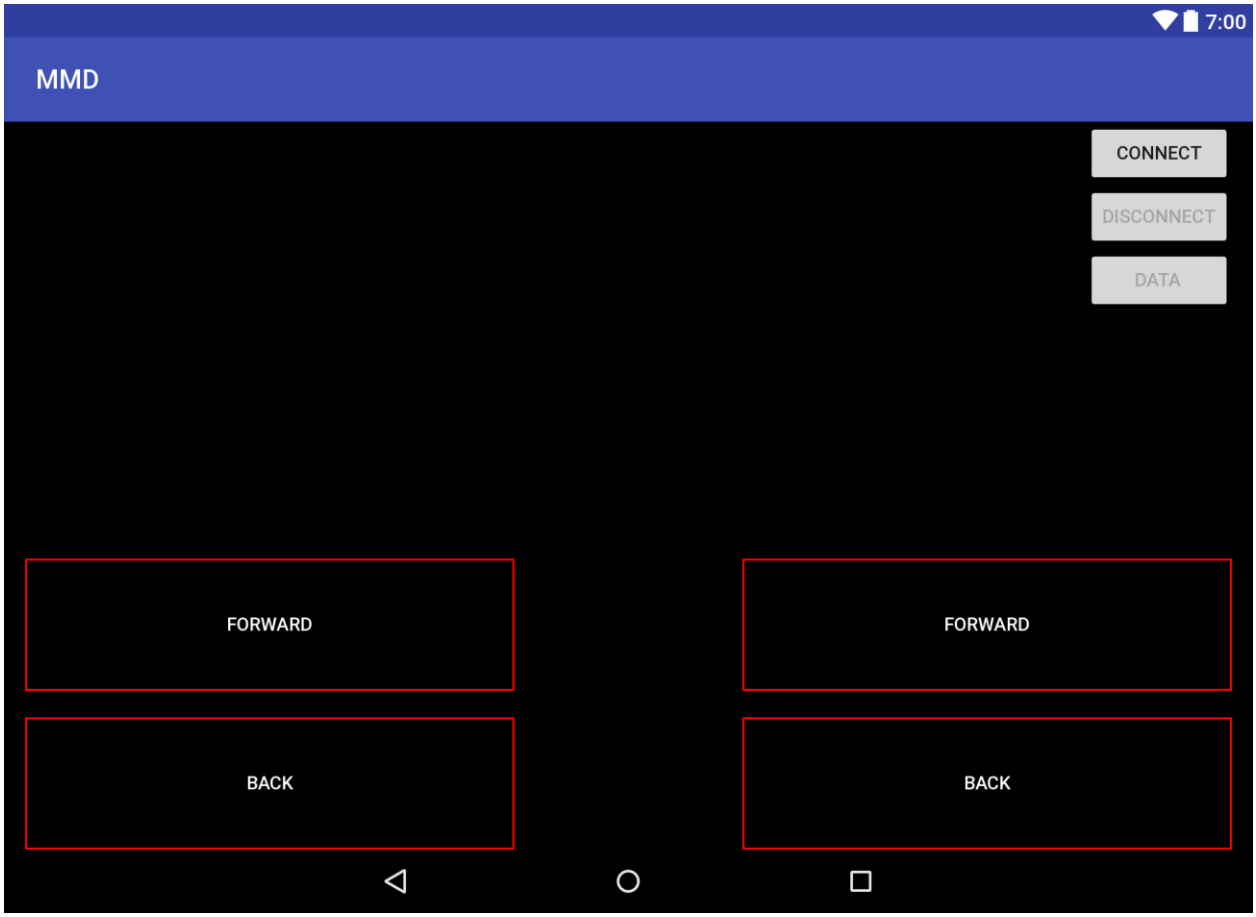

Рисунок 3.2 – Графічний інтерфейс додатку

3.4 Програмне забезпечення клієнта

Насамперед, створюємо порожній android додаток і даємо йому права на використання мережі для передачі. Для цього необхідно в маніфесті (AndroidManifest.xml) додати потрібні права.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Тепер програма має доступ до мережі. Для передачі даних необхідно використовувати механізм сокетів. Socket - це механізм обміну даними між процесами, при цьому ці процеси можуть знаходитись на різних машинах, які з'єднані в одну мережу. Для роботою с мережею використовуються бібліотеки `java.io` та `java.net`, розглянуті у розділі 1.3. Для того, щоб

відбувалося спілкування, необхідно знати ip адресу і номер порту, по якому буде відбуватися обмін.

Створимо клас mmdServer, в якому будемо описувати логіку підключення. І додамо в нього поля для імені сервера та номера порту, а також самого сокета, через який буде відбуватися спілкування:

```
private class mmdServer
{
    /* ip адрес сервера mmd */
    private String mmdServerName = "192.168.4.1";


    /* номер порта, на который сервер mmd принимает соединение */
    private int mmdServerPort = 8888;

    /* сокет, через который приложение общается с сервером mmd */
    private Socket mmdSocket = null;
}
```

Для подальшої роботи необхідно створити три методи: відкрити з'єднання, відправити дані і закрити з'єднання.

При відкритті з'єднання, необхідно переконатися, що раніше не було відкрито сокет, що б не засмічувати ресурси, і якщо він відкритий - закрити його. Також при відкритті з'єднання необхідно запустити окремий поток для прослуховування з'єднання.

```
    void openConnection() throws Exception
    {
        /* освобождаем ресурсы */
        closeConnection();

        try
        {
            /* создаем новый сокет */
            mmdSocket = new Socket(mmdServerName, mmdServerPort);
            disabledControlInterface = false;

            runOnUiThread(new Runnable()
            {
                @Override
                public void run()
                {
                    /* тут меняем граф. интерфейс*/
                    logWrite("Соединение установлено");
```

```
                    }
                });
            }
            catch (IOException e)
            {
                throw new Exception("Невозможно создать сокет: " + e.getMessage());
            }
        }
```

Для закриття з'єднання, у сокета необхідно викликати метод `close ()`:

```
<   /* метод для закрытия сокета */
        void closeConnection()
        {
            /* проверяем сокет, если он не закрыт, то закрываем его и освобождаем сое
динение */
            if (mmdSocket != null && !mmdSocket.isClosed())
            {
                try
                {
                    mmdSocket.close();
                }
                catch (IOException e)
                {
                    Log.e(LOG_TAG, "Невозможно закрыть сокет: " + e.getMessage());
                }
                finally
                {
                    mmdSocket = null;
                    disabledControlInterface = true;

                    runOnUiThread(new Runnable()
                    {
                        @Override
                        public void run()
                        {
                            detectorView.setText("");
                            logWrite("Соединение закрыто");
                        }
                    });
                }

            }
            mmdSocket = null;
        }
```

Відправка даних відбувається через OutputStream. Його можна обгорнути в різні обгортки, а можна і безпосередньо у сокета викликати `getOutputStream ()` метод `write ()`:

```
        /* метод для отправки данных по сокету */
        void sendData(byte[] data) throws Exception
        {
            /* проверяем сокет, если он не создан или закрыт, то выдаем исключение */
            if (mmdSocket == null || mmdSocket.isClosed())
            {
                throw new Exception("Невозможно отправить данные. Сокет не создан или
закрыт");
            }

            /* отправка данных */
            try
            {
                mmdSocket.getOutputStream().write(data);
                mmdSocket.getOutputStream().flush();
            }
            catch (IOException e)
            {
                throw new Exception("Невозможно отправить данные: " + e.getMessage())
;
            }
        }
```

Також нам необхідно перевизначити метод `finalize ()` і звільнити ресурс:

```
        @Override
        protected void finalize() throws Throwable
        {
            super.finalize();
            closeConnection();
        }
```

Тепер необхідно додати прослуховування з'єднання у метод `openConnection()`. Якщо перший символ вхідного потоку дорівнює решітці (код 35 згідно ASCII), то вхідні дані є значенням з АЦП Arduino (дані з детектора метала). Виводимо це значення у додаток, і якщо значення перевищує 119 то включаємо вібрацію пристрою (за наявністю). Якщо ніяких даних немає, то закриваємо з’єднання і блокуємо усі управляючі кнопки.

```
                /* запуск прослушивания соединения */
                 new Thread(new Runnable()
                 {
                     @Override
```

```
                    public void run()
                    {
                        /* входящий поток */
                        InputStream inputStream = null;
                        try
                        {


                            inputStream = mmdSocket.getInputStream();
                        }
                        catch (IOException e)
                        {
                            Log.e(LOG_TAG, "Невозможно получить входящий поток: " + e
.getMessage());
                        }

                        final byte[] buffer = new byte[1024*4];

                        while(true)
                        {
                            try
                            {
                                final int count = inputStream != null ? inputStream.r
ead(buffer, 0, buffer.length) : -1;

                                if (count > 0)
                                {
                                    runOnUiThread(new Runnable()
                                    {
                                        @Override
                                        public void run()
                                        {
                                            if (buffer[0] == 35)
                                            {
                                                String in = new String(buffer, 1, 3);
                                                detectorView.setText(in);
                                                int val = new Integer(in);
                                                if (val >= 120)
                                                {
                                                    vibrator.vibrate(350);
                                                }
                                            }
                                            else
                                            {
                                                logWrite(new String(buffer, 0, count)
);

                                            }
                                        }
                                    });
                                }
                                else
                                {
                                    if (count == -1)
                                    {
                                        mmdSocket.close();
                                        mmdSocket = null;
                                        disabledControlInterface = true;
                                        runOnUiThread(new Runnable()
                                        {
                                            @Override
```

```
                                        public void run() {
                                            btnConnect.setEnabled(true);
                                            btnData.setEnabled(false);
                                            btnDisconnect.setEnabled(false);
                                            btnLeftForward.setBackgroundResource(
R.drawable.button_style_disabled);
                                            btnLeftBack.setBackgroundResource(R.d
rawable.button_style_disabled);
                                            btnRightForward.setBackgroundResource
(R.drawable.button_style_disabled);
                                            btnRightBack.setBackgroundResource(R.
drawable.button_style_disabled);
                                            detectorView.setText("");
                                            logWrite("Соединение потеряно");
                                        }
                                    });
                                    break;
                                }
                            }
                        }
                        catch (IOException e)
                        {
                            Log.e(LOG_TAG, "Ошибка: " + e.getMessage());
                        }
                    }
                }
            }).start();
```

У зв'язку з тим що ми знаходимося в окремих потоках, то для оновлення елементів у графічному інтерфейсі використовується метод `runOnUiThread ()`.

Тепер залишається створити обробники і прив'язати їх до відповідних кнопок.

Код для обробника кнопки з'єднання з сервером. Для того, щоб все підключався, необхідно винести підключення в окремий поток:

```
btnConnect.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        /* создаем объект для работы с mmd */
        mmdS = new mmdServer();
```

```
                /* открываем соединение, открытие должно происходить в отдельном пото
ке */
                new Thread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        try
                        {
                            mmdS.openConnection();
                        }
                        catch (Exception e)
                        {
                            runOnUiThread(new Runnable()
                            {
                                @Override
                                public void run()
                                {
                                    logWrite("Ошибка соединения");
                                }
                            });
                            Log.e(LOG_TAG, e.getMessage());
                            mmdS = null;
                        }
                    }
                }).start();
            }
        });
```

Код для обробника кнопки активації / дезактивації отримування даних с детектора метала. При натисканні на кнопку ми передаємо мобільному металодетектору символ решітки, металодетектор отримує його і міняє стан дозволу передачі даних на протилежний:

```
        btnData.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                if (mmdS == null)
                {
                    Log.e(LOG_TAG, "Сервер не создан");
                }
                new Thread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        try
                        {
                            /* отправляем данныее */
                            mmdS.sendData("#".getBytes());
```

```
                    }
                    catch (Exception e)
                    {
                        Log.e(LOG_TAG, e.getMessage());
                    }
                }
            }).start();
        }
    });
```

Код для обробника кнопки закриття з’єднання:

```
    btnDisconnect.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            /* закрываем соединение */
            mmdS.closeConnection();
        }
    });
```

Код для обробника кнопки керування гусеничної платформи (ліва гусениця вперед). Для інших керуючих кнопок код обробника має однаковий вигляд, за винятком команд для передачі.

```
 btnLeftForward.setOnTouchListener(new OnTouchListener()
    {
        @Override
        public boolean onTouch(View v, MotionEvent event)
        {
            if (disabledControlInterface)
            {
                return false;
            }
            switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    btnLeftForward.setBackgroundResource(R.drawable.button_style_
focus);
                    new Thread(new Runnable()
                    {
                        @Override
                        public void run()
                        {
                            try
                            {
                                /* отправляем данныее */
                                mmdS.sendData("q".getBytes());
                            }
```

```
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_UP:
                        btnLeftForward.setBackgroundResource(R.drawable.button_style_
enabled);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("z".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_CANCEL:
                        btnLeftForward.setBackgroundResource(R.drawable.button_style_
disabled);
                        break;
                }
                return true;
            }
        });
```

На цьому створення додатка-клієнта для операційної системи Android вважається завершеним.

3.5 Висновки

У цьому розділі виконувався завершаючий етап розробки мобільного металодетектора, а саме – створення програмного забезпечення. Розроблене ПЗ дозволяє підключатися до точки доступу Wi-Fi (розгорнутому на мобільному металодетекторі) та призводити обмін даним, в результаті чого можна керувати пристроєм з додатку на OS Android та отримувати значення з його детектору метала.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даного проекту бакалавра була розробка мобільного металодетектора та додатку для дистанційного керування ним. Так як процес розробки виконувався у домашніх умовах, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для приміщення, де проводились роботи над дипломним проектом.

4.1 Аналіз стану умов праці

Робота над створенням мобільного металодетектора проходить в побутовому приміщенні. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

4.1.1 Вимоги до приміщення

Геометричні розміри приміщення зазначені у таблиці 4.1. Для зручності спільної роботи з іншими працівниками (обговорення ідей, з’ясування проблем і т.д.) в кімнаті є диван і журнальний стіл. Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення та кондиціонування.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі. Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

Таблиця 4.1 – Розміри робочого місця

Параметр	Значення
Довжина, м	6
Ширина, м	4
Висота, м	2,5
Площа, $м^2$	24
Об’єм, $м^3$	60

Згідно до санітарних норм мікроклімату виробничих приміщень [3] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм – не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

4.1.2 Вимоги до організації робочого місця

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця [4] (табл. 5.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 – Характеристика робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Робочий стіл на досліджуваному місці також містить достатньо простору для ніг. Крісло, що використовується в якості робочого сидіння, є підйємно-поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки, також воно м'яке і виконане з екологічної шкіри, що дає можливість працювати у комфорті. Екран монітору знаходиться на відстані 0.8 м, клавіатура має можливість регулювання кута нахилу 5-15°. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам. Приміщення кабінету знаходиться на першому поверсі двох поверхової будівлі і має об’єм 60 $м^3$, площу – 24 $м^2$. У цьому кабінеті обладнано одне робоче місце, яке укомплектовано 2 ПК, один з котрих сервер без наявності пристроїв І/О інформації.

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум у приміщенні знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна, а опалення — централізоване.

Розміщення вікон забезпечує природне освітлення з коефіцієнтом природного освітлення не менше 1,5%, а загальне штучне освітлення, яке здійснюється за допомогою восьми люмінесцентних ламп, забезпечує рівень освітленості не менше 200 Лк.

У кабінеті є електрична мережа з напругою 220 В, яка створює небезпеку ураження електричним струмом. ПК та периферійні пристрої можуть бути джерелами електромагнітних випромінювань, аерозолів та шкідливих речовин (часток тонеру, оксидів нітрогену та озону).

За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет оснащений переносним вуглекислотним вогнегасником ВВК-5 .

Наявна аптечка для надання долікарської допомоги, а також у кабінеті роблять вологе прибирання та щоденно провітрюють приміщення.

4.1.3 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до ІІ – ІІІ ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м’язово скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Тобто наявне психофізіологічні небезпечні та шкідливі фактори:

а) фізичного перевантаження:

- статичного;
- динамічного;

б) нервово-психічного перевантаження:

- розумового перенапруження;
- монотонності праці;
- перенапруження аналізаторів;
- емоційних перевантажень.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи [4].

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви тривалістю 15 хв через кожну годину роботи;

4.2 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.2.1 Аналіз небезпечних та шкідливих факторів при розробці виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із

забезпеченням виконання , які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп'ютери (ПК) та периферійні пристрої. Основними робочими характеристиками персонального комп'ютера є:

- робоча напруга U = +220В +-5%;
- робочий струм I = 2А;
- споживана потужність P = 600 Вт.

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Нормативні документи
Фізичні		
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, серверного обладнання для роботи	[3]
- підвищена або знижена вологість повітря	-//-	[3]
- підвищена або знижена рухливість повітря	-//-	[3]
- підвищений рівень напруги електричної мережі	-//-	[6] [7]
- підвищений рівень статичної електрики	-//-	[6]
- підвищена напруженість електромагнітного поля	-//-	[5]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	[5]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	[6]
Психофізіологічні		
-нервово-психічна перевантаження	Розумова робота над проектом	[1] [4]
- фізичні (статичне – сидіння)	порушення умов праці та організації робочого часу	[1]

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного

санітарного лікаря України від 10.12.98 [4]. За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

4.2.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання. Небезпека загоряння пов'язана з особливістю комп'ютерів – із значною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах. При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80…100°*С*). При проходженні електричного струму по провідниках і деталей виділяється тепло, що в умовах їх високої щільності може привести до перегріву, і може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання між струмоведучими частинами обладнання (шини, електроди). Також ймовірна небезпека внаслідок перевантаження напруги, розрядки зарядів статичної електрики,

пошкодження обладнання та електропроводки. Електростатичний розряд виникає під час тертя двох ізольованих матеріалів. Розряд статичної електрики може виникнути під час роботи вентилятора або комп'ютера. Кабельні лінії є найбільш пожежонебезпечними місцем. Наявність пального ізоляційного матеріалу, ймовірних джерел запалювання у вигляді електричних іскор і дуг, розгалуженість і недоступність роблять кабельні лінії місцем найбільш ймовірного виникнення і розвитку пожежі. Для зниження займистості і здатності поширювати полум'я кабелі покривають вогнезахисними покриттями.

Для гасіння пожеж в офісному приміщенні пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними.

Виникнення пожежі можливе, якщо на об'єкті є горючі речовини, окиснювач і джерела запалювання. Вірогідність пожежної небезпеки приймається значною, якщо ймовірна взаємодія цих трьох чинників. Горючими компонентами є: будівельні матеріали для акустичної і естетичної обробки приміщень, перегородки, підлоги, двері, ізоляція силових, сигнальних кабелів і т.д.

Горючими матеріалами в приміщенні, де розташовані ЕОМ, є:

1) поліамід – матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420°*C*,

2) полівінілхлорид – ізоляційний матеріал, горюча речовина, температура запалювання 335°*C*, температура самозаймання 530°*C*,

3) склотекстоліт ДЦ – матеріал друкарських плат, важкогорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання,

4) пластикат кабельний – матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1,

5) деревина – будівельний і обробний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255°*C*, температура самозаймання 399°*C*.

Для відводу теплоти від ЕОМ діє потужна система кондиціонування. Тому кисень, як окиснювач процесів горіння, є в будь-якій точці приміщень обчислювального центру.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до [9] відносяться до пожежонебезпечної зони класу П-IIа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється II ступень вогнестійкості.

Потенційними джерелами запалювання можуть бути:

- іскри і дуги короткого замикання;
- електрична іскра при замиканні і розмиканні ланцюгів;
- перегріви від тривалого перевантаження,
- відкритий вогонь і продукти горіння,
- наявність речовин, нагрітих вище за температуру самозаймання,
- розрядна статична електрика.

Причинами можливого загоряння і пожежі можуть бути:

- несправність електроустановки;
- конструктивні недоліки устаткування;
- коротке замикання в електричних мережах;
- запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, синильна кислота, аміак, ацетон та ін.

4.2.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.3 Гігієнічні вимоги до параметрів виробничого середовища

4.3.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт

Іа. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають нормам [3] і наведені в табл. 4.4.

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура *С°***	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [3]. Рівні позитивних і негативних іонів у повітрі мають відповідати [3]. Для забезпечення оптимальних параметрів мікроклімату в приміщенні проводяться перерви в роботі співробітників, з метою його провітрювання. Існують спеціальні системи кондиціонування, які забезпечують підтримання в приміщенні балансу оптимальних параметрів мікроклімату.

Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.3.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового

сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

Освітленість приміщення має велике значення при роботі на ПЕОМ. Вона багато в чому визначається колірною і мережевий обстановкою. Для зменшеного поглинання світла стеля і стіни вище панелей (1,5 – 1,7 м.). Якщо вони не облицьовані звукопоглинальним матеріалом, фарбуються білою водоемульсійною фарбою (коефіцієнт відбиття повинен бути не менше 0,7).

Для забарвлення стіни панелей рекомендується віддавати перевагу світлим фарбам.

Основний потік природного світла при цій повинен бути зліва. Не допускається спрямування основного світлового потоку природного світла праворуч, ззаду і спереду працівника на ПЕОМ.

Робота на ПЕОМ може здійснюватися за таких видах освітлення:

- загальному штучному освітленні, коли відео монітори розташовуються по периметру приміщення або при центральному розташуванні робочих місць у два ряди по довжині кімнати з екранами, звернені в протилежні сторони;

- суміщене освітлення (природне + штучне) тільки при одному і трьох рядном розташуванні робочих місць, коли екран і поверхню робочого столу знаходяться перпендикулярно світла несучій стіні. При цьому штучне освітлення буде виконане стельовими або підвісними люмінесцентними світильниками, рівномірно розміщеними по стелі рядами паралельно світловим прорізам так, щоб екран відео монітора знаходився в зоні захисного кута світильника, і його проекції не доводилися на екран. Працюючі на ПЕОМ не повинні бачити відображення світильників на екрані. Застосовувати місцеве освітлення при роботі на ПЕОМ не рекомендується.

Природне освітлення, коли робочі місця з ПЕОМ розташовуються в один ряд по довжині приміщення на відстані 0,8 – 1,0 м від стіни з віконними прорізами, і екрани знаходяться перпендикулярно цієї стіни. Основний потік

природного світла при цій повинен бути зліва. Не допускається спрямування основного світлового потоку природного світла праворуч, ззаду і спереду працює на ПЕОМ. Оптимальна відстань очей до екрана відео монітора повинна становити 60-70 см, допустиме не менше 50 см. Розглядати інформацію ближче 50 см не рекомендується.

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації ЕОМ виконується зорова робота IVв розряду точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) рівна 200 лк. Джерелом природного освітлення є сонячне світло.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає нормам [8]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \times S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 6 \cdot 4 = 24$ м^2,

$S = 1/10 \cdot 24 = 2{,}4$ м^2.

Приймаємо 1 вікно площею S=2,4 м^2.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 3200 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \times S \times Z \times K}{F \times U \times M}, \quad (4.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 24 м^2;

Z – поправочний коефіцієнт світильника (1,1 для люмінесцентних ламп);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 3200лм.

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \times 24 \times 1.1 \times 1.5}{3200 \times 0.575 \times 2} = 3{,}2$$

Приймаємо освітлювальну установку, яка складається з 3-х світильників, які складаються з 2-х люмінесцентних ламп загальною потужністю 80 Вт, напругою – 220 В.

4.3.3 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в [10].

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об'єм приміщення 20 $м^3$, то потрібно подати не менш як 30 $м^3$/год повітря;

б) якщо об'єм приміщення у межах від 20 до 40 $м^3$, то потрібно подати не менш як 20 $м^3$/год повітря;

в) якщо об'єм приміщення становить понад 40 $м^3$, допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

- зниження рівня шуму та вібрації:

а) у джерелі виникнення, шляхом застосування раціональних конструкцій, нових матеріалів і технологічних процесів;

б) звукоізолювання устаткування за допомогою глушників, резонаторів, кожухів, захисних конструкцій, оздоблення стін, стелі, підлоги тощо;

в) використання засобів індивідуального захисту).

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;

- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

- не тягнути за мережевий кабель, щоб витягти вилку з розетки;

- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

- не залишати включені електроприлади без нагляду;

- не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

Від ураження струмом застосовують різні електричні захисні засоби:

а) Ізолюючі – ізолюють людини від струмоведучих або заземлених частин, а так-же від землі. Вони діляться на основні та додаткові.

б) Основні – володіють ізоляцією, здатної довго витримувати робоче напругу електроустановки і тому ними дозволяється стосуватися струмоведучих частин, знаходячи-трудящих під напругою.

в) Запобіжні – володіють ізоляцією нездатною витримати робоча напруга електроустановки, і тому вони не можуть самостійно захищати людину від ураження струмом під цим напругою. Їх значення - посилити захисні дії основних і ізолюючих засобів, разом з якими вони повинні застосовуватися, при чому при використання основних захисних засобів достатньо застосування одного запобіжного захисного засобу.

4.5.1 Розрахунок захисного заземлення

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [2], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового

заземлювача η – це відношення діючої провідності цього заземлювача до найбільш можливої його провідності за нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача $\eta_с$.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \quad (4.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.} = R_д$.

Підставивши числові значення у формулу (4.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ = 40 Ом·м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр.} = \Psi \cdot \rho \quad (4.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{\text{розр.в}}$ = 1,7 і горизонтальних $\rho_{\text{розр.г}}$ = 5,5 Ом·м.

$$\rho_{\text{розр.в}} = 1{,}7 \cdot 40 = 68 \text{ Ом} \cdot \text{м}$$

$$\rho_{\text{розр.в}} = 5{,}5 \cdot 40 = 220 \text{ Ом} \cdot \text{м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_{\text{в}}$, Ом, за (4.5).

$$R_B = \frac{\rho_{\text{розр.в}}}{2 \cdot \pi \cdot 1_B} \cdot \left(\ln \frac{2 \cdot 1_B}{d_{CT}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + 1_B}{4 \cdot t - 1_B} \right), \quad (4.5)$$

де $l_{\text{в}}$ – довжина вертикального заземлювача (для труб – 2 − 3 м; $l_{\text{в}}$ = 3 м);

$d_{\text{ст}}$ – діаметр стержня (для труб – 0,03 − 0,05 м; $d_{\text{ст}}$ = 0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (4.6):

$$t = h_E + \frac{1_E}{2}, \quad (4.6)$$

де $h_{\text{в}}$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді

$$t = 0{,}8 + \frac{3}{2} = 2{,}3 \text{ м};$$

$$R_{\text{в}} = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2{,}3 + 3}{4 \cdot 2{,}3 - 3} \right) = 18{,}5 \text{ Ом}$$

1) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_{\text{в}}$:

$$n = \frac{2R_E}{R_Д} = \frac{2\times18,5}{4} = 9,25 \quad (4.7)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ = 0,57 (табличне значення).

2) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $\eta_в$, шт:

$$n = \frac{2\cdot R_E}{R_Д\cdot{}^{n}_E} = \frac{2\cdot18,5}{4\cdot0,57} \approx 16 \quad (4.8)$$

3) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$1_C = 1,05 \cdot L_B \cdot (n_B - 1), \quad (4.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м); $n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05\cdot3\cdot(16-1) \approx 48 \text{ м}.$$

Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2\cdot\pi\cdot1_c} \cdot \ln\frac{2\cdot1_c^2}{d_{см}\cdot h_г}, \quad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см} = 0{,}95b$, $b = 0{,}15$ м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

$l_с$ – довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0{,}95 \cdot 0{,}15 \cdot 0{,}5} = 8{,}1 \text{ Ом}$$

4) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с = 0{,}3$.

Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг.} = \frac{R_E \cdot R_г}{R_E \cdot \eta_c + R_г \cdot n_E \cdot \eta_E} \leq R_д, \qquad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг} < 4$ Ом, а саме:

$$R_{заг} = \frac{18{,}5 \cdot 8{,}1}{18{,}5 \cdot 0{,}3 + 8{,}1 \cdot 16 \cdot 0{,}57} = 1{,}9 \leq R_д$$

При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. В приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися згідно з нормами. У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків та усуненню пожежі.

4.5 Висновки

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

У даній дипломній роботі було розроблено мобільний металодетектор. Також було створено додаток-клієнт під мобільну операційну систему Android, щоб надати користувачам простий і візуально зрозумілий інтерфейс управлінням мобільним детектором.

Передача даних від Arduino до мобільного додатку використовує клієнт-серверну архітектуру з використанням пакетної передачі даних на базі стеку протоколів TCP/IP згідно з поставленим технічним завданням на розробку.

Собівартість такого пристрою виявилась не значною, пристрій успішно функціонує, отже головну задачу дипломного проекту можна вважати досягненою.

ПЕРЕЛІК ПОСИЛАНЬ

1. НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин»;
2. НПАОП 0.00-4.15-98 «Положення про розробку інструкцій з охорони праці»;
3. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» ;
4. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»;
5. ГОСТ 12.1.044-89 «ССБТ. Вогнестійкість. Номенклатура показників і методи їх визначення»;
6. ГОСТ 12.1.030-81 «Електробезпека. Захисне заземлення, занулення».
7. ГОСТ 13109-97 «Електрична енергія. Сумісність технічних засобів. Норми якості електричної енергії в системах електропостачання загального призначення»;
8. ДБН В.2.5-28:2015 «Державні Будівельні Норми України. Природне і штучне освітлення»;
9. НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою».
10. ДБН В.2.5-67:2013. «Опалення, вентиляція та кондиціонування.»
11. Нечаев И. Индикатор металлических предметов. // Радио, 2003, №10, с. 56.
12. С. Л. Корякин-Черняк, А. П. Семьян. «Металлоискатели своими руками. Как искать, чтобы найти монеты, украшения, клады»

13. Работа с ESP8266: Первоначальная настройка, обновление прошивки, связь по Wi-Fi, отправка-получение данных на ПК: [Електронний ресурс] // Geektimes URL: https://geektimes.ru/post/241054/ (дата звернення: 10.05.2017).
14. Java: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення: 07.05.2017).
15. Android Studio: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення: 07.05.2017).
16. Arduino: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Arduino (дата звернення: 02.05.2017).
17. Android: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Android (дата звернення: 12.05.2017).

Додаток А

Програмне забезпечення для платформи Arduino

```
#define LEFT_FORWARD 113
#define LEFT_BACK 97
#define LEFT_STOP 122
#define RIGHT_FORWARD 119
#define RIGHT_BACK 115
#define RIGHT_STOP 120
#define DATA 35
#define L_IN0 5
#define L_IN1 4
#define R_IN0 3
#define R_IN1 2
#define ledData 13
bool Data = false;
int tmpTimeout = 0;
void clearBuffer()
{
  while(Serial1.available() > 0) Serial1.read();
}
void commandSend(String text)
{
  Serial1.println(text);
  Serial1.flush();
  delay(50);
}
void setup()
{
  pinMode(ledData, OUTPUT);
  digitalWrite(ledData, HIGH);
  pinMode(A0, INPUT);
  pinMode (L_IN0, OUTPUT);
  pinMode (L_IN1, OUTPUT);
  pinMode (R_IN0, OUTPUT);
  pinMode (R_IN1, OUTPUT);
  Serial.begin(115200);
  Serial1.begin(115200);
  delay(2000);
  clearBuffer();
  commandSend("AT+CIPMODE=0");
  delay(2000);
  clearBuffer();
  commandSend("AT+CIPMUX=1");
  delay(2000);
  clearBuffer();
  commandSend("AT+CIPSERVER=1,8888");
  clearBuffer();
  digitalWrite(ledData, LOW);
}
void loop()
{
  int buffer[64];
  int b = 0;
  while (Serial1.available() > 0)
  {
    int r = Serial1.read();
    if (((r > 96 && r < 123) || r == 35) && b < 64)
    {
      buffer[b++] = r;
    }
  }
  for (int i = 0; i < b; i++)
  {
```

```
    if (buffer[i] == DATA)
    {
      Data = !Data;
      tmpTimeout = 0;
      digitalWrite(ledData, (Data) ? HIGH : LOW);
    }
    if (buffer[i] == LEFT_FORWARD)
    {
      digitalWrite (L_IN0, HIGH);
      digitalWrite (L_IN1, LOW);
    }
    if (buffer[i] == LEFT_BACK)
    {
      digitalWrite (L_IN0, LOW);
      digitalWrite (L_IN1, HIGH);
    }
    if (buffer[i] == LEFT_STOP)
    {
      digitalWrite (L_IN0, LOW);
      digitalWrite (L_IN1, LOW);
    }
    if (buffer[i] == RIGHT_FORWARD)
    {
      digitalWrite (R_IN0, HIGH);
      digitalWrite (R_IN1, LOW);
    }
    if (buffer[i] == RIGHT_BACK)
    {
      digitalWrite (R_IN0, LOW);
      digitalWrite (R_IN1, HIGH);
    }
    if (buffer[i] == RIGHT_STOP)
    {
      digitalWrite (R_IN0, LOW);
      digitalWrite (R_IN1, LOW);
    }
  }
  if (Data && !tmpTimeout)
  {
    int val = analogRead(A0);
    if (val < 100) val = 100;
    if (val > 999) val = 999;
    Serial1.println("AT+CIPSEND=0,4");
    Serial1.flush();
    delay(100);
    Serial1.print("#");
    Serial1.print(val);
    Serial1.flush();
  }
  else
  {
    delay(100);
  }
  tmpTimeout++;
  if (tmpTimeout > 4)
  {
    tmpTimeout = 0;
  }
}
```

Додаток Б

Програмне забезпечення для додатку на OS Android

```
package ua.edu.snu.mmd;
import android.content.Context;
import android.os.Vibrator;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Layout;
import android.text.method.ScrollingMovementMethod;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.content.pm.ActivityInfo;
import android.widget.TextView;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Locale;
import static android.view.WindowManager.*;
public class MainActivity extends AppCompatActivity
{
    private final String LOG_TAG = "mmd";
    private Button btnConnect  = null;
    private Button btnData  = null;
    private Button btnDisconnect = null;
    private Button btnLeftForward = null;
    private Button btnLeftBack = null;
    private Button btnRightForward = null;
    private Button btnRightBack = null;
    private TextView detectorView = null;
    private TextView logView = null;
    private mmdServer mmdS = null;
    private boolean disabledControlInterface = true;
    private Vibrator vibrator = null;
    private void logWrite(String msg)
    {
        long date = System.currentTimeMillis();
        SimpleDateFormat sdf = new SimpleDateFormat("[hh:mm:ss] ", Locale.US);
        String dateString = sdf.format(date);
        logView.append(dateString + msg + "\n");
        final Layout layout = logView.getLayout();
        if(layout != null)
        {
            int scrollDelta = layout.getLineBottom(logView.getLineCount() - 1) -
logView.getScrollY() - logView.getHeight();
            if(scrollDelta > 0)
                logView.scrollBy(0, scrollDelta);
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        getWindow().setFlags(LayoutParams.FLAG_FULLSCREEN,
LayoutParams.FLAG_FULLSCREEN);
        getSupportActionBar().hide();
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        logView = (TextView)findViewById(R.id.logView);
        logView.setMovementMethod(new ScrollingMovementMethod());
```

```
        logWrite("Соединение отсутствует");
        detectorView = (TextView)findViewById(R.id.detectorView);
        btnConnect = (Button) findViewById(R.id.btnConnect);
        btnData = (Button) findViewById(R.id.btnData);
        btnDisconnect = (Button) findViewById(R.id.btnDisconnect);
        btnLeftForward = (Button) findViewById(R.id.btnLeftForward);
        btnLeftBack = (Button) findViewById(R.id.btnLeftBack);
        btnRightForward = (Button) findViewById(R.id.btnRightForward);
        btnRightBack = (Button) findViewById(R.id.btnRightBack);
        vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
        btnConnect.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                /* создаем объект для работы с mmd */
                mmdS = new mmdServer();
                /* открываем соединение, открытие должно происходить в отдельном
потоке */
                new Thread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        try
                        {
                            mmdS.openConnection();
                        }
                        catch (Exception e)
                        {
                            runOnUiThread(new Runnable()
                            {
                                @Override
                                public void run()
                                {
                                    logWrite("Ошибка соединения");
                                }
                            });
                            Log.e(LOG_TAG, e.getMessage());
                            mmdS = null;
                        }
                    }
                }).start();
            }
        });
        btnData.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                if (mmdS == null)
                {
                    Log.e(LOG_TAG, "Сервер не создан");
                }
                new Thread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        try
                        {
```

```
                            /* отправляем данныее */
                            mmdS.sendData("#".getBytes());
                        }
                        catch (Exception e)
                        {
                            Log.e(LOG_TAG, e.getMessage());
                        }
                    }
                }).start();
            }
        });
        btnDisconnect.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                /* закрываем соединение */
                mmdS.closeConnection();
            }
        });
        btnLeftForward.setOnTouchListener(new OnTouchListener()
        {
            @Override
            public boolean onTouch(View v, MotionEvent event)
            {
                if (disabledControlInterface)
                {
                    return false;
                }
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
btnLeftForward.setBackgroundResource(R.drawable.button_style_focus);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("q".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_UP:
btnLeftForward.setBackgroundResource(R.drawable.button_style_enabled);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("z".getBytes());
                                }
```

```
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_CANCEL:
btnLeftForward.setBackgroundResource(R.drawable.button_style_disabled);
                        break;
                }
                return true;
            }
        });
        btnLeftBack.setOnTouchListener(new OnTouchListener()
        {
            @Override
            public boolean onTouch(View v, MotionEvent event)
            {
                if (disabledControlInterface)
                {
                    return false;
                }
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
btnLeftBack.setBackgroundResource(R.drawable.button_style_focus);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("a".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_UP:
btnLeftBack.setBackgroundResource(R.drawable.button_style_enabled);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("z".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
```

```
                        break;
                    case MotionEvent.ACTION_CANCEL:
btnLeftBack.setBackgroundResource(R.drawable.button_style_disabled);
                        break;
                }
                return true;
            }
        });
        btnRightForward.setOnTouchListener(new OnTouchListener()
        {
            @Override
            public boolean onTouch(View v, MotionEvent event)
            {
                if (disabledControlInterface)
                {
                    return false;
                }
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
btnRightForward.setBackgroundResource(R.drawable.button_style_focus);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("w".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_UP:
btnRightForward.setBackgroundResource(R.drawable.button_style_enabled);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("x".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_CANCEL:
btnRightForward.setBackgroundResource(R.drawable.button_style_disabled);
                        break;
                }
                return true;
```

```
            }
        });
        btnRightBack.setOnTouchListener(new OnTouchListener()
        {
            @Override
            public boolean onTouch(View v, MotionEvent event)
            {
                if (disabledControlInterface)
                    return false;
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
btnRightBack.setBackgroundResource(R.drawable.button_style_focus);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("s".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_UP:
btnRightBack.setBackgroundResource(R.drawable.button_style_enabled);
                        new Thread(new Runnable()
                        {
                            @Override
                            public void run()
                            {
                                try
                                {
                                    /* отправляем данныее */
                                    mmdS.sendData("x".getBytes());
                                }
                                catch (Exception e)
                                {
                                    Log.e(LOG_TAG, e.getMessage());
                                }
                            }
                        }).start();
                        break;
                    case MotionEvent.ACTION_CANCEL:
btnRightBack.setBackgroundResource(R.drawable.button_style_disabled);
                        break;
                }
                return true;
            }
        });
    }
    private class mmdServer
    {
        /* ip адрес сервера mmd */
        private String mmdServerName = "192.168.4.1";
        /* номер порта, на который сервер mmd принимает соединение */
```

```
        private int mmdServerPort = 8888;
        /* сокет, через которий приложение общается с сервером mmd */
        private Socket mmdSocket = null;
        /* открытие нового соединения, если сокет уже открыт, то он закрывается */
        void openConnection() throws Exception
        {
            /* освобождаем ресурсы */
            closeConnection();
            try
            {
                /* создаем новый сокет */
                mmdSocket = new Socket(mmdServerName, mmdServerPort);
                disabledControlInterface = false;
                runOnUiThread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        btnConnect.setEnabled(false);
                        btnData.setEnabled(true);
                        btnDisconnect.setEnabled(true);
btnLeftForward.setBackgroundResource(R.drawable.button_style_enabled);
btnLeftBack.setBackgroundResource(R.drawable.button_style_enabled);
btnRightForward.setBackgroundResource(R.drawable.button_style_enabled);
btnRightBack.setBackgroundResource(R.drawable.button_style_enabled);
                        logWrite("Соединение установлено");
                    }
                });
                /* запуск прослушивания соединения */
                new Thread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        /* входящий поток */
                        InputStream inputStream = null;
                        try
                        {
                            inputStream = mmdSocket.getInputStream();
                        }
                        catch (IOException e)
                        {
                            Log.e(LOG_TAG, "Невозможно получить входящий поток: " +
e.getMessage());
                        }
                        final byte[] buffer = new byte[1024*4];
                        while(true)
                        {
                            try
                            {
                                final int count = inputStream != null ?
inputStream.read(buffer, 0, buffer.length) : -1;
                                if (count > 0)
                                {
                                    runOnUiThread(new Runnable()
                                    {
                                        @Override
                                        public void run()
                                        {
                                            if (buffer[0] == 35)
                                            {
```

```
                                                String in = new String(buffer, 1, 3);
                                                detectorView.setText(in);
                                                int val = new Integer(in);
                                                if (val >= 120)
                                                {
                                                    vibrator.vibrate(350);
                                                }
                                            }
                                            else
                                            {
                                                logWrite(new String(buffer, 0,
count));
                                            }
                                        }
                                    });
                                }
                                else
                                {
                                    if (count == -1)
                                    {
                                        mmdSocket.close();
                                        mmdSocket = null;
                                        disabledControlInterface = true;
                                        runOnUiThread(new Runnable()
                                        {
                                            @Override
                                            public void run() {
                                                btnConnect.setEnabled(true);
                                                btnData.setEnabled(false);
                                                btnDisconnect.setEnabled(false);
btnLeftForward.setBackgroundResource(R.drawable.button_style_disabled);
btnLeftBack.setBackgroundResource(R.drawable.button_style_disabled);
btnRightForward.setBackgroundResource(R.drawable.button_style_disabled);
btnRightBack.setBackgroundResource(R.drawable.button_style_disabled);
                                                detectorView.setText("");
                                                logWrite("Соединение потеряно");
                                            }
                                        });
                                        break;
                                    }
                                }
                            }
                            catch (IOException e)
                            {
                                Log.e(LOG_TAG, "Ошибка: " + e.getMessage());
                            }
                        }
                    }
                }).start();
            }
            catch (IOException e)
            {
                throw new Exception("Невозможно создать сокет: " + e.getMessage());
            }
        }
        /* метод для закрытия сокета */
        void closeConnection()
        {
            if (mmdSocket != null && !mmdSocket.isClosed())
            {
                try
```

```
                {
                    mmdSocket.close();
                }
                catch (IOException e)
                {
                    Log.e(LOG_TAG, "Невозможно закрыть сокет: " + e.getMessage());
                }
                finally
                {
                    mmdSocket = null;
                    disabledControlInterface = true;
                    runOnUiThread(new Runnable()
                    {
                        @Override
                        public void run()
                        {
                            btnConnect.setEnabled(true);
                            btnData.setEnabled(false);
                            btnDisconnect.setEnabled(false);
btnLeftForward.setBackgroundResource(R.drawable.button_style_disabled);
btnLeftBack.setBackgroundResource(R.drawable.button_style_disabled);
btnRightForward.setBackgroundResource(R.drawable.button_style_disabled);
btnRightBack.setBackgroundResource(R.drawable.button_style_disabled);
                            detectorView.setText("");
                            logWrite("Соединение закрыто");
                        }
                    });
                }
            }
            mmdSocket = null;
        }
        /* метод для отправки данных по сокету */
        void sendData(byte[] data) throws Exception
        {
            if (mmdSocket == null || mmdSocket.isClosed())
            {
                throw new Exception("Невозможно отправить данные. Сокет не создан или
закрыт");
            }
            try
            {
                mmdSocket.getOutputStream().write(data);
                mmdSocket.getOutputStream().flush();
            }
            catch (IOException e)
            {
                throw new Exception("Невозможно отправить данные: " +
e.getMessage());
            }
        }
        @Override
        protected void finalize() throws Throwable
        {
            super.finalize();
            closeConnection();
        }
    }
}
```

Додаток В

Розмітка графічного інтерфейсу для додатка на OS Android

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_dark"
    android:paddingLeft="16dp"
    android:paddingRight="16dp">
    <TextView
        android:id="@+id/logView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/holo_green_dark"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignBottom="@+id/btnData"
        android:layout_toLeftOf="@+id/btnConnect"
        android:layout_toStartOf="@+id/btnConnect" />
    <Button
        android:id="@+id/btnConnect"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Connect"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignLeft="@+id/btnDisconnect"
        android:layout_alignStart="@+id/btnDisconnect" />
    <Button
        android:id="@+id/btnDisconnect"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Disconnect"
        android:enabled="false"
        android:layout_below="@+id/btnConnect"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
    <Button
        android:id="@+id/btnData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/btnDisconnect"
        android:layout_alignRight="@+id/btnDisconnect"
        android:layout_below="@+id/btnDisconnect"
        android:text="DATA"
        android:enabled="false"
        android:layout_alignLeft="@+id/btnDisconnect"
        android:layout_alignStart="@+id/btnDisconnect" />
    <Button
        android:id="@+id/btnRightForward"
        android:layout_width="370dp"
        android:layout_height="100dp"
        android:layout_above="@+id/btnLeftBack"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:background="@drawable/button_style_disabled"
        android:text="FORWARD"
        android:textColor="#FFFFFF" />
    <Button
        android:id="@+id/btnRightBack"
        android:layout_width="370dp"
```

```
        android:layout_height="100dp"
        android:background="@drawable/button_style_disabled"
        android:text="BACK"
        android:textColor="#FFFFFF"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:id="@+id/btnLeftForward"
        android:background="@drawable/button_style_disabled"
        android:layout_width="370dp"
        android:layout_height="100dp"
        android:text="FORWARD"
        android:textColor="#FFFFFF"
        android:layout_above="@+id/btnLeftBack"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
    <Button
        android:id="@+id/btnLeftBack"
        android:layout_width="370dp"
        android:layout_height="100dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="20dp"
        android:background="@drawable/button_style_disabled"
        android:text="BACK"
        android:textColor="#FFFFFF" />
    <TextView
        android:id="@+id/detectorView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/logView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="51dp"
        android:textColor="#00FF00"
        android:textSize="32sp" />
</RelativeLayout>
```