

"Електронна система
автоматизованого арбітражу командних ігор".

Анотація

У науково-дослідному проєкті розглянуто розробку електронної системи автоматизованого арбітражу командних ігор (розглянуто на прикладі гри в футбол), яка базується на використанні акселерометрів для фіксації рухів гравців та суддів.

Актуальність дослідження зумовлена необхідністю підвищення об'єктивності, оперативності та прозорості суддівських рішень у спортивних командних іграх. Застосування сучасних електронних засобів дозволяє мінімізувати вплив суб'єктивних факторів і знизити ймовірність помилкових рішень, що безпосередньо впливає на якість і справедливість змагального процесу.

Мета роботи полягає у створенні доступної та технологічно оптимізованої системи електронного арбітражу, здатної забезпечити високоточний контроль за діями учасників гри та підтримку прийняття рішень у режимі реального часу.

Для досягнення поставленої мети були визначені такі завдання дослідження: провести аналіз існуючих технічних рішень у сфері автоматизованого арбітражу та виявити їхні ключові недоліки, розробити конструктивно й економічно оптимізовану модель пристрою з урахуванням зменшення маси та вартості, побудувати електричну принципову схему апаратної частини системи, реалізувати програмне забезпечення, що виконує функціональний цикл арбітра гри, провести експериментальну перевірку працездатності та ефективності прототипу.

Практичне значення отриманих результатів полягає у можливості застосування розробленої системи в різних видах командних ігор для підвищення об'єктивності суддівства та вдосконалення процесу змагань. Запропоноване рішення характеризується зниженою вартістю й малою масою, що розширює перспективи його практичного впровадження у спортивних організаціях, навчальних закладах та аматорських змаганнях.

За матеріали даної роботи також були опубліковані тези доповіді на конференції [28]

Зміст

| | |
|--|----|
| ВСТУП..... | 6 |
| РОЗДІЛ I АКТУАЛЬНІСТЬ ПРОБЛЕМИ | 8 |
| I.I Актуальність проблеми | 8 |
| I.II Переваги та недоліки | 8 |
| I.III Технічний розвиток роботів-арбітрів | 8 |
| I.IV Висновок | 9 |
| РОЗДІЛ II НОВИЗНА ТА ОРИГІНАЛЬНІСТЬ ІДЕЇ..... | 10 |
| II.I Огляд існуючих систем..... | 10 |
| II.II Запропоноване рішення..... | 10 |
| II.III Новизна даної системи | 10 |
| II.IV Що нового має система | 10 |
| РОЗДІЛ III. ВИКОРИСТАНІ МЕТОДИ ДОСЛІДЖЕННЯ..... | 12 |
| III.I Характеристики існуючих систем..... | 12 |
| III.II Вирішення задачі за допомогою акселерометра..... | 12 |
| III.II.I Реалізація рішення з використанням акселерометра..... | 12 |
| III.II.II Перевірка адекватності використання рішення..... | 12 |
| III.II.III Апаратна реалізація | 13 |
| III.II.V Основні функції ПЗ..... | 14 |
| III.II.VI Структурна схема пристрою | 16 |
| Рисунок III.II.VI - Структурна схема пристрою | 16 |
| III.II.VII Алгоритмів обміну даними між датчиками та центральним вузлом по бездротовому інтерфейсу..... | 17 |
| III.II.VIII Алгоритм ухвалення рішень на основі розпізнавання зображень | 17 |
| III.II.IX Усунення помилок та налаштування запропонованого рішення. | 18 |
| III.III Висновок | 19 |
| РОЗДІЛ IV. ТЕОРЕТИЧНІ НАУКОВІ РЕЗУЛЬТАТИ..... | 20 |
| IV.I Нові теоретичні результати..... | 20 |
| IV.I.I. Визначення моменту удару по м'ячу за допомогою акселерометрів | 20 |
| IV.I.II. Аналіз рухів суддів за допомогою акселерометрів | 20 |
| IV.I.III. Аналіз фізіологічних та кінематичних показників суддів | 21 |

| | |
|---|----|
| IV.I.IV. Розпізнавання жестів суддів за допомогою акселерометрів | 21 |
| IV.I. V. Контроль руху робота-арбітра за допомогою акселерометрів | 22 |
| IV.I. VI. Основні принципи системи контролю руху | 23 |
| IV.I. VII. Аналіз технічних дій гравців за допомогою акселерометрів | 24 |
| РОЗДІЛ V. ЕЛЕКТРИЧНА ПРИНЦИПОВА СХЕМА | 25 |
| РОЗДІЛ VI. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ | 26 |
| ВИСНОВОК | 28 |
| ДЖЕРЕЛА | 30 |
| ДОДАТОК А | 32 |
| ДОДАТОК Б..... | 33 |
| ДОДАТОК В..... | 34 |
| ДОДАТОК Г | 35 |
| ДОДАТОК Д..... | 36 |
| ДОДАТОК Ж..... | 37 |
| ДОДАТОК З | 38 |
| ДОДАТОК Л..... | 40 |
| ДОДАТОК М..... | 41 |
| ДОДАТОК Н | 43 |
| ДОДАТОК О | 44 |
| ДОДАТОК П | 45 |

ВСТУП

Сьогодні футбол є не лише популярним видом спорту серед чоловіків і жінок, але й потужною індустрією, яка вимагає високого рівня точності, об'єктивності та технологічної підтримки. Попри всі досягнення у сфері відеоповторів та автоматизованих рішень, таких як система VAR, проблема людського фактору в арбітражі залишається актуальною. Арбітри продовжують помилятися в динамічних ситуаціях, що безпосередньо впливає на результат матчу.

Інженерні рішення на основі вбудованих систем, сенсорних модулів та цифрової обробки сигналів дають можливість створити повністю автоматизовану систему арбітражу, яка частково або повністю може замінити людину на полі. У межах даного дослідження акцент зроблено на створення прототипу інтелектуального пристрою, який буде здатний відслідковувати рух м'яча на футбольному полі, визначати момент його виходу за межі та приймати рішення у режимі реального часу.

Тому за мету було поставлено розробити електронну систему з вбудованим програмним забезпеченням, що автоматизує функції футбольного арбітра шляхом фіксації руху м'яча, аналізу просторових подій на полі та взаємодії з інтерфейсом судді.

Основна ідея полягає у використанні комплексу електронних модулів, до яких входять:

- ✓ акселерометри та гіроскопи для фіксації динаміки руху об'єктів,
- ✓ RFID-мітки та бездротові передавачі для визначення положення гравців та м'яча,
- ✓ камера з комп'ютерним зором для візуальної верифікації координат,
- ✓ мікроконтролер (STM32, ESP32 або аналогічний), який виконує збір, обробку та аналіз усіх даних за заданими алгоритмами.

Система повинна відповідати низці технічних критеріїв:

1. безперервна робота протягом усього матчу (90+ хв),
2. точне визначення координат м'яча з урахуванням тривимірного положення (X, Y, Z),
3. ігнорування незначних об'єктів (наприклад, комах або листя),
4. відображення подій на дисплеї судді в реальному часі через бездротовий зв'язок,
5. створення стоп-кадрів для суперечливих моментів із можливістю прокрутки,
6. автоматичне визначення типу події: вихід, гол, офсайд, дотик рукою, порушення.

Крім того, розроблена система має можливість:

- 1) зчитувати номери та приналежність гравців до команди,
- 2) визначати останнього, хто торкнувся м'яча,

3) фіксувати ключові сигнали під час гри та зберігати їх в лог-файлах для подальшого аналізу.

Основними задачами, що можна визначити є:

1. Розробити програму на мові C++ для вбудованої системи, яка забезпечить повноцінну обробку сенсорних та візуальних даних.
2. Інтегрувати апаратну частину, що включає акселерометри, бездротові модулі та мікроконтролер (STM32 або Arduino/ESP32), із програмною логікою.
3. Оптимізувати систему для польових умов, мінімізуючи затримки, збої та перешкоди для учасників гри.
4. Забезпечити простий та зрозумілий вивід на дисплей судді, включаючи ключові події та координати.

РОЗДІЛ І АКТУАЛЬНІСТЬ ПРОБЛЕМИ

I.1 Актуальність проблеми

Сучасний футбол вимагає максимальної точності, об'єктивності та оперативності в суддівстві. Через людський фактор часто виникають суперечки, помилки та недовіра до рішень арбітра. Саме тому актуальним стає створення робота-арбітра — автоматизованої системи, здатної повністю або частково замінити людину у прийнятті суддівських рішень під час гри.

Для вдосконалення суддівства вже пройшли певні етапи: початкові впровадження (VAR був вперше використаний на чемпіонаті світу 2018 року), розвиток SAOT (на чемпіонаті світу 2022 року вперше застосовано напів-автоматизовану систему визначення офсайду), інтеграція ІІІ (системи на кшталт VARS демонструють потенціал ІІІ у розпізнаванні фолів).

I.2 Переваги та недоліки

Проте з впровадженням даних технологій організатори стикаються як з позитивними результатами так і з труднощами реалізації. Перевагами даних технологій є підвищення точності рішень, зменшення впливу людського фактору, пришвидшення прийняття рішень та можливість використання в аматорському футболі. Але є і існують такі недоліки: високі витрати на впровадження, необхідність адаптації правил гри, потенційне зменшення ролі людських суддів та ще є питання довіри до рішень ІІІ.

I.3 Технічний розвиток роботів-арбітрів

На сьогоднішній день сформувались певні шляхи розвитку автоматизованих систем:

1. Мікромініатюризація та датчики

Сучасні системи суддівства (наприклад, SAOT, VAR, Hawk-Eye) вже містять десятки датчиків на гравцях і м'ячі. Йде постійне зменшення розмірів сенсорів, що дозволяє встановлювати їх у бутси, форму, м'яч або навіть вмонтовувати у покриття поля.

2. Бездротові інтерфейси передачі даних

Сучасні бездротові технології (Wi-Fi 6, 5G, UWB) забезпечують миттєву передачу даних із сенсорів до серверів аналізу.

3. Системи позиціонування

Технології GPS, локальні трекінг-системи та технології відносного позиціонування дозволяють точно знати розташування кожного об'єкта (гравця, м'яча, арбітра) з точністю до сантиметра.

4. Обробка зображень та розпізнавання

Із розвитком алгоритмів комп'ютерного зору системи вже можуть самостійно: розпізнавати фоли, фіксувати порушення правил руками, аналізувати емоції або небезпечну поведінку, будувати 3D-картину гри в реальному часі.

5. Штучний інтелект і прийняття рішень

AI-алгоритми дозволяють системі самостійно робити висновки. Крім того, системи можуть навчатися на основі мільйонів ігор та удосконалювати свої рішення.

Хоча ідея створення робота-арбітра виникла в контексті футболу, самі технології, які при цьому розробляються, мають ширше застосування. Це приклад мультифункціональної системи, що може використовуватися в охороні (автоматичний контроль периметра та виявлення порушників), будівництві (моніторинг переміщення персоналу, дотримання техніки безпеки), промисловості (автоматичне сортування деталей, облік дій працівників, розпізнавання небезпечних ситуацій).

I.IV Висновок

Отже можемо вважати, що робот-арбітр — це більше, ніж суддя для гри. Це універсальний інтелектуальний модуль, що поєднує досягнення у сфері електроніки, бездротових мереж, штучного інтелекту й комп'ютерного зору. Його розробка — це шлях до нових стандартів точності, безпеки та автоматизації в різних сферах життя.

Дана задача є актуальною тому, що вона є прикладом поєднання всіх вищезгаданих технологій разом, їх спільного інтенсивного використання, а також, тому що вона не тільки про ігровий, а і про інші сфери використання.

РОЗДІЛ II НОВИЗНА ТА ОРИГІНАЛЬНІСТЬ ІДЕЇ

II.I Огляд існуючих систем

Якщо зробити структурований огляд кожної існуючої системи [ДОДАТОК А], можна вказати на такі недоліки як: залежність від камер (обмежене поле зору, можливі "сліпі зони"), затримки в обробці (час на прийняття рішень може бути значним), висока вартість (дорогі в установці та обслуговуванні), обмежена мобільність (складно адаптувати до інших видів спорту або умов).

II.II Запропоноване рішення

Аналізуючи дані критерії можна зробити висновок, що використання однієї з існуючих систем не є достатньо дієвою, а використання їх сукупності є досить дорогим. Тому було розглянуто декілька варіантів, що розглядались і обрано рішення: використання акселерометра. Це дозволить збільшити кількість об'єктів спостереження та його різноманітність, а саме: використати не складні частини (акселерометри, бездротові модулі), врахували всі об'єкти (всі гравці, м'яч, арбітри), швидкість та точність рішень (до 0,5 секунди, до 1 мм), використати можливі спостереження та простий алгоритм роботи налаштування (акселерометри, відео, GPS, та автоматичне, адаптивне).

II.III Новизна даної системи

Безперервне відстеження: Дані про прискорення та рух кожного гравця та м'яча в реальному часі.

Незалежність від візуального контакту: Система не залежить від якості відео чи освітлення.

Швидка обробка даних: Миттєве виявлення порушень або подій.

Масштабованість: Легко адаптується до різних видів спорту або умов.

Дана система дозволить підвищити точність: Комбінація даних з акселерометрів та відео забезпечує більш точні рішення.

Зменшити витрати: Менша кількість камер та обладнання знижує загальні витрати. Та забезпечить можливість використання в різних умовах та для різних цілей (наприклад, моніторинг безпеки).

II.IV Що нового має система

1. Використання акселерометрів для кожного об'єкта гри

Існуючі системи покладаються переважно на візуальне відстеження (камери, сенсори в м'ячі). Наша система використовує мініатюрні акселерометри, які вбудовуються в одяг гравців, у взуття, в м'яч, у форму арбітра. Це дозволяє відслідковувати точні прискорення, напрямки руху, зіткнення та інші параметри у реальному часі з високою точністю.

2. Автономність від візуального контакту

У випадках поганої видимості, затемнення камери або затінення об'єктів на полі — візуальні системи (VAR, Hawk-Eye, SAOT) втрачають точність або не працюють. Акселерометри працюють незалежно від освітлення та видимості, що забезпечує стабільність і надійність системи.

3. Миттєва обробка і передача даних через бездротові інтерфейси.

Дані з акселерометрів передаються в режимі реального часу через Wi-Fi,

Bluetooth, UWB (Ultra-Wideband) з високою пропускнуою здатністю та мінімальною затримкою (<10 мс). Це дозволяє приймати автоматичні рішення за долі секунди без втручання людини.

4. Інтеграція з AI-алгоритмами для аналізу ситуацій

Комбінація зображення з відеокамер (при наявності) та даних прискорення дозволяє AI: відрізнити симуляцію від реального порушення (наприклад, фол чи "пірнання"), автоматично розпізнавати порушення правил, зокрема поза грою, удари після свистка, агресивну поведінку тощо.

5. Адаптація до різних ігрових ситуацій та неігрових сценаріїв тобто система може бути переналаштована під інші види спорту, контроль доступу або пересування персоналу на підприємствах, охорону територій, сортування рухомих об'єктів або деталей на виробництві.

6. Підвищена точність рішень

Завдяки прямому вимірюванню фізичних параметрів, очікується: точність визначення моменту дотику, удару або падіння — до 1 мілісекунди, позиціонування — до 1 мм при використанні комбінованих даних з GPS/UWB.

РОЗДІЛ III. ВИКОРИСТАНІ МЕТОДИ ДОСЛІДЖЕННЯ

III.1 Характеристики існуючих систем

Переглянувши характеристики існуючих систем [ДОДАТОК Б] та аналізуючи рішення для вирішення поставлених системі завдань, що вже існують для реалізації нашої системи обираємо систему CMU Smart Referee. А саме для реалізації системи робота-арбітра пропонується використання акселерометра MPU-6050, який поєднує в собі трьохосьовий акселерометр та гіроскоп.

Переглянувши, критерії такого рішення [ДОДАТОК В], можна виділити деякі переваги: економічно вигідним та ефективним рішенням, висока точність, низьке енергоспоживання, та незалежність від умов освітлення. Дані факти, роблять його ідеальним для вбудованих систем, які потребують швидкої та надійної обробки даних.

III.2 Вирішення задачі за допомогою акселерометра

III.2.1 Реалізація рішення з використанням акселерометра

Для розробки системи робота-арбітра, яка відслідковує події на полі (наприклад, вихід м'яча за межі, фіксація удару, пасу тощо), було обрано акселерометр як ключовий сенсор.

Використовується модуль MPU-6050, що включає: трьохосьовий акселерометр (фіксує прискорення по X, Y, Z), гіроскоп (реєструє кутові швидкості), інтерфейс I2C (для зв'язку з мікроконтролером (Arduino або ESP32)). Також MPU-6050 інтегрується в корпус м'яча або кріпиться на корпус робота-арбітра;

Дані з акселерометра передаються через Bluetooth/Wi-Fi до центрального блоку обробки;

Програма обробляє дані, розраховуючи: швидкість руху м'яча/гравця, зміну напрямку, силу удару, фіксацію падіння, удару або виходу за межі.

Можемо виділити основні етапи роботи системи: ініціалізація пристрою, читання даних акселерометра, обчислення загального прискорення, порівняння з пороговим значенням, якщо поріг перевищено — подія зафіксована збереження події в пам'ять / передача на сервер, повернення до зчитування Ці етапи представимо у вигляді алгоритму [ДОДАТОК Г].

III.2.2 Перевірка адекватності використання рішення

Спосіб 1: Відеоспостереження тобто візуальне підтвердження:

Встановлюється відеокамера, яка записує рух м'яча або гравця. У реальному часі фіксуються події: удар, вихід м'яча за лінію, зіткнення тощо. Відео порівнюється з графіками прискорення з MPU-6050 (наприклад: у момент удару фіксується пік на графіку прискорення $> 3g$, який співпадає з відео). Результат позитивний коли події, зареєстровані акселерометром, чітко співпадають з подіями на відео.

Спосіб 2: Лабораторне тестування на стенді

М'яч із вбудованим MPU-6050 кидається з відомої висоти під певним кутом на амортизуючу поверхню. Обчислюються очікувані прискорення за законами фізики. Зчитується реальний сигнал з датчика та порівнюється з

теоретичними розрахунками. Результат можна вважати вдалим, якщо похибка між очікуваним і фактичним прискоренням не перевищує 5–7%.

III.II.III Апаратна реалізація

На основі порівняльного аналізу існуючих рішень для електронних асистентів арбітра, запропоновано розділити розроблювану електронну систему на дві підсистеми:

1. Стаціонарна спостережна підсистема, яка має забезпечувати централізоване відеоспостереження за всім ігровим полем, а також збирати та інтегрувати дані з датчиків, накладаючи їх на відеозображення для зручності моніторингу арбітром.

2. Розподілена датчикова підсистема, яка включає модулі, розташовані на рухомих учасниках гри та на м'ячі. Кожен модуль відповідає за вимірювання механічної динаміки об'єкта, з яким він взаємодіє.

Таким чином, розроблювана система являє собою розподілений апаратно-програмний комплекс, що працює в просторі та з'єднаний бездротовою мережею, з високим рівнем автономії складових її компонентів. Важливим завданням цієї системи є забезпечення синхронізованої роботи окремих електронних пристроїв.

Для досягнення функціональної повноти роботи системи та повноцінного моніторингу, необхідні наступні компоненти:

- Відеокамери на приводах: Камери повинні бути оснащені механізмами для переміщення, що дозволяє охопити всі ділянки поля, з використанням двохосьових гіроскопічних підвісів і крокових двигунів.
- Пристрій збору даних з датчиків: Це пристрій, який отримує дані від датчиків на гравцях та м'ячі за допомогою бездротових інтерфейсів.
- Пристрій первинної обробки відеозображення: Цей пристрій обробляє відео та дані з датчиків, з можливістю автоматичного контролю ситуацій на полі, а також впливає на приводи відеокамер для кращого огляду.
- Пристрій виведення композитного відеозображення: Пристрій виводить на екран арбітра та глядачів відео з усіма позначками, такими як траєкторія руху м'яча, позиції гравців і межі поля, а також автоматично визначає важливі події (вихід м'яча за межі поля або фол).

Розподілена датчикова підсистема

Ця підсистема призначена для точного відслідковування та вимірювання руху об'єктів (м'яча і гравців) у реальному часі, з подальшою передачею цих даних до центральної обробної системи для аналізу.

- Модулі датчиків на учасниках гри: Вони повинні вимірювати такі параметри, як прискорення, швидкість і напрямок руху. Для цього можуть використовуватися акселерометри, гіроскопи, GPS-датчики для трекінгу.
- Модуль для м'яча: Датчик для визначення положення м'яча на полі в реальному часі, з використанням таких технологій, як RFID, GPS.
- Бездротові інтерфейси: Для передачі даних від датчиків до центральної системи повинні використовуватися швидкі й надійні бездротові технології, що гарантують передачу даних у реальному часі.

Інтеграція підсистем та узгоджена робота

Обидві підсистеми мають ефективно обмінюватися даними та працювати синхронізовано для безперервного моніторингу гри.

- Комунікація між підсистемами: Передача даних між відеокамерами, датчиками та обробними пристроями повинна бути швидкою і безперебійною.

- Керування привідними механізмами: Система автоматично налаштовує камери для забезпечення найкращого огляду, з урахуванням даних, отриманих від датчиків.

- Сигналізація: У разі виявлення важливих подій система повинна передавати сигнали на головний дисплей для арбітра та глядачів.

Пристрій виведення відеозображення

- Індикація: Пристрій виводить відео з усіма необхідними позначками, такими як траєкторія м'яча, положення гравців і межі поля, для арбітра та глядачів.

- Автоматичне визначення особливих положень гри: Система повинна автоматично визначати важливі ситуації, наприклад, вихід м'яча за межі поля або фол, і демонструвати ці події на дисплеї.

Всю рухому систему (для гравця), можна представити у вигляді структурної схеми [ДОДАТОК Д]

III.II.V Основні функції ПЗ

1. Відслідковування виходу м'яча за межі поля:

Алгоритми відеообробки повинні визначати межі ігрового поля (стінки, лінії воріт) та фіксувати момент, коли м'яч виходить за ці межі.

Для цього використовуються методи детекції контурів (наприклад, за допомогою градієнтних методів та канта у OpenCV), а також геометричні фільтри для розпізнавання різних об'єктів.

Висота польоту м'яча враховується за допомогою тригерів по висоті в кожному кадрі, а також вимірювання глибини в кадрі для обчислення висоти м'яча за допомогою стереозображення або глибини об'єктів.

2. Класифікація об'єктів на полі:

Для класифікації об'єктів на полі за кольором та типом використовуються методи машинного навчання на основі нейронних мереж чи класифікації за ознаками кольору.

Кольори м'яча та гравців можна визначити за допомогою моделей кольору (наприклад, RGB, HSV), що дозволяє розділяти об'єкти за кольоровими відмінностями.

Використовуються методи класифікації (наприклад, KNN, SVM) для визначення типу об'єкта на основі текстур, форми або розміру.

3. Супроводження спостереження за грою за допомогою відеокамер:

Враховуючи рух м'яча, система повинна направляти відеокамери за допомогою інтерфейсів для управління приводами (двохосьові гіроскопічні підвіси).

Алгоритми повинні враховувати шлях м'яча на полі, його швидкість і напрямок, а також записувати дані про координати у реальному часі для подальшої обробки.

Алгоритм слідування за об'єктами включає визначення напрямку руху м'яча через послідовні кадри та корекцію положення камери відповідно до траєкторії м'яча.

4. Запис і відтворення відео:

Програмне забезпечення повинно мати функцію запису відео, яка дозволить зберігати всі важливі моменти гри.

Для відтворення використовуються функції повторного перегляду з можливістю сповільнення і повторення частини відео. Це може бути корисним для арбітрів, які хочуть детально переглянути моменти гри, такі як вихід м'яча за межі поля або порушення.

5. Персоналізація гравців:

Щоб ідентифікувати, від якого гравця м'яч покинув поле, можна використовувати датчики, розташовані на формах гравців, а також відеоаналіз для розпізнавання гравців.

Функція відслідковування передачі м'яча між гравцями реалізується на основі зважених датчиків та алгоритмів на основі машинного навчання для визначення гравця, який зробив передачу.

6. Статистика володіння м'ячем:

Для відстежування володіння м'ячем кожним гравцем або командою використовуються алгоритми для відстеження об'єктів та аналітика по часу володіння м'ячем.

ПЗ повинно автоматично вести статистику, записуючи дані про час володіння м'ячем кожним гравцем за допомогою інтерфейсів датчиків та відеообробки.

Це дозволяє обчислювати статистику передач між гравцями, а також визначати контроль над м'ячем.

7. Алгоритмічний підхід:

Обробка відео за допомогою OpenCV:

Використовуються методи виявлення контурів для визначення границь ігрового поля.

Алгоритми перетворення кольору (наприклад, перехід у модель HSV) для точного визначення кольору м'яча і гравців.

Методи фільтрації і порогового значення для виділення об'єктів на полі (м'яч, гравці).

8. Трекінг об'єктів:

Для відслідковування м'яча та гравців можна використовувати методи оптичного потоку або фільтри Калмана, які дозволяють точно передбачати рух об'єкта в наступних кадрах, а також компенсувати шви зображення через камеру.

9. Розпізнавання і класифікація:

Для розпізнавання гравців можна застосувати методи глибокого навчання: навчання нейронних мереж для класифікації гравців за їхніми унікальними характеристиками (форма, рух, позиція).

Сукупність даних датчиків і відеоаналізу дозволяє точніше відстежувати, від якого гравця м'яч покинув поле.

10. Інтерфейс користувача:

ПЗ повинно забезпечувати інтерактивний інтерфейс, який дозволяє арбітру бачити онлайн-дані про гру, змінювати перегляд (наприклад, вибрати види відтворення відео або статистику володіння м'ячем).

III.II.VI Структурна схема пристрою

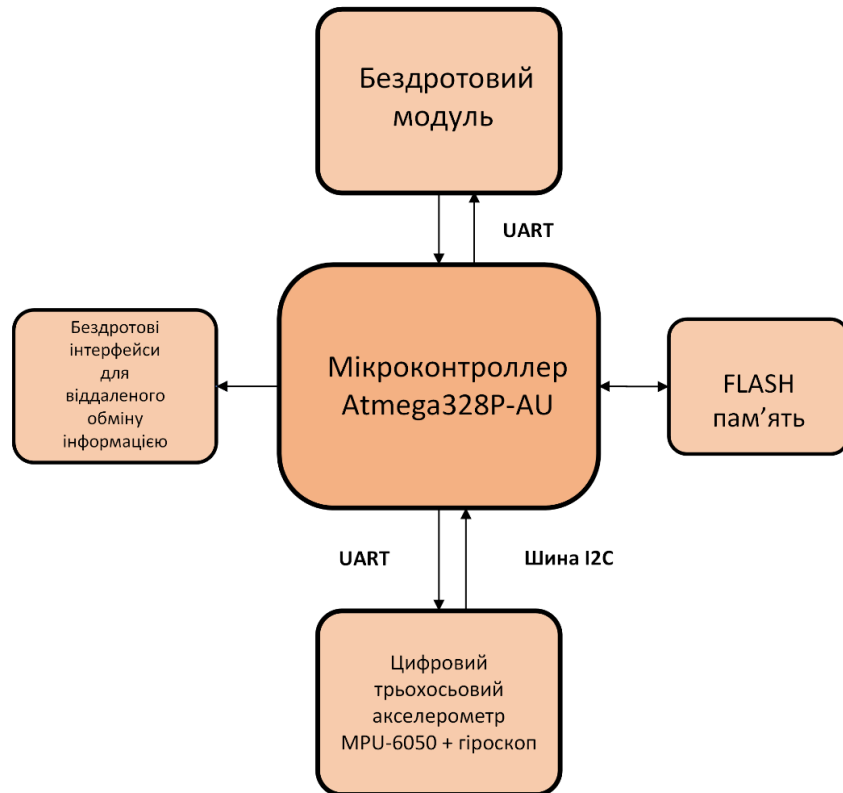


Рисунок III.II.VI - Структурна схема пристрою

Пояснення до схеми:

Розроблена структурна схема наведена на Рисунку III.II.VI та включає такі складові:

Цифровий 3-осьовий акселерометр MPU-6050+гіроскоп, використовується в якості давача

Бездротовий модуль CC2541 AT-09 Bluetooth 4.0 BLE

Мікроконтролер (МК) ATmega328P-AU, який призначений для збору і опрацювання даних з давача (акселерометра);

Флеш (FLASH) пам'ять для встановлення ОС, драйверів, ПЗ і збереження даних;

Інтерфейси бездротової комунікації для віддаленого обміну інформацією з ПК.

Побудована структура системи базується на модульному принципі, що дає змогу швидко та ефективно модифікувати систему в процесі вдосконалення.

Окрім того, низька ціна пристрою, що розробляється, забезпечується використанням дешевих апаратних складових загального призначення та вільновикористовуваного ПЗ.

Всі елементи під'єднані паралельно до основної плати з мікроконтроллером Atmega325, що подає сигнали на інші датчики для

виконання завдань.

III.II.VII Алгоритмів обміну даними між датчиками та центральним вузлом по бездротовому інтерфейсу

Основним завданням системи є збір даних з різних датчиків, обробка цих даних та передача результатів до центрального вузла в реальному часі. Основні компоненти схеми: датчики (акселерометри, гіроскопи, магнітометри), мікроконтролери Arduino (напр. Arduino Nano/Uno/MKR WiFi 1010), Бездротові модулі (напр. ESP8266, NRF24L01, Bluetooth, Wi-Fi), Центральний вузол обробки (напр. Raspberry Pi або ПК), Програмне забезпечення для обробки даних

Основні функції, що описано в схемі:

1.Збір даних із сенсорів

Акселерометри (наприклад, MPU6050) встановлюються на гравців або м'яч. Кожен сенсор вимірює прискорення по трьох осях (X, Y, Z). Дані зчитуються мікроконтролером Arduino.

2. Обробка на локальному рівні (на Arduino)

Arduino виконує попередню обробку сигналу: фільтрацію шуму, нормалізацію даних.

При потребі обраховується швидкість, напрям руху або виявлення зіткнень.

Формується пакет даних для передачі.

3. Бездротова передача до центрального вузла

За допомогою модуля Wi-Fi або NRF24L01 дані надсилаються до центрального вузла. Обирається протокол обміну: MQTT, HTTP або власний формат пакетів. Дані передаються з урахуванням захисту від втрат (контроль CRC/ACK-підтвердження).

4. Прийом та обробка на центральному вузлі

Центральний вузол отримує пакети від усіх пристроїв. Синхронізація часу між пристроями (на основі NTP або програмної логіки). Дані зберігаються, обробляються та аналізуються: визначається траєкторія м'яча, обраховується статистика по гравцях, виявляються ігрові події (наприклад, м'яч вийшов за межі поля)

Всі функції можна представити в схемі [ДОДАТОК Ж]. В даній схема використовуються доступні, дешеві та енергоефективні компоненти для реалізації складних завдань в режимі реального часу. Обмін між акселерометрами та центральним вузлом через бездротовий інтерфейс забезпечує гнучкість, точність і адаптивність системи робота-арбітра.

III.II.VIII Алгоритм ухвалення рішень на основі розпізнавання зображень

1. Ініціалізація системи (після запуску пристрій перевіряє, чи все підключено та готове до роботи)

Що відбувається: Завантаження програмного забезпечення на Arduino. Налаштовується зв'язок із камерою, акселерометром (наприклад, MPU6050) та іншими датчиками.

Інструменти: Arduino IDE, бібліотеки I2C, MPU6050.

2. Отримання відео та даних руху (якщо акселерометр «відчув» сильний поштовх, то система аналізує кадр у цей момент).

Що відбувається: Камера передає зображення гри, а акселерометр реєструє будь-які різкі рухи (наприклад, удар м'яча).

Приклад: Удар м'ячем викликає зміну прискорення — акселерометр це фіксує.

3. Обробка зображення (це як "підсвітити" тільки важливе на зображенні — поле, м'яч, гравці)

Що відбувається: Фото/відео кадр очищується від шуму, виділяються контури, кольори.

Методи: Перехід до HSV формату (для визначення кольорів м'яча або форми гравців). Виділення об'єктів через фільтри

4. Розпізнавання об'єктів

Що відбувається: За допомогою моделей машинного навчання або простих алгоритмів система розуміє, де м'яч, а де гравець.

Методи: TensorFlow Lite (нейромережа). Haar Cascades (простий вбудований детектор в OpenCV). Пояснення: Система "бачить" м'яч, форму, або людину й може сказати про це повідомити.

5. Визначення координат і руху (Якщо м'яч рухається швидко вправо — значить його хтось ударив, акселерометр це підтвердить.)

Що відбувається: Система обчислює, де на полі знаходиться об'єкт, у який бік він рухається.

Акселерометр: Визначає момент удару, силу, напрямок м'яча.

6. Аналіз ситуації згідно правил гри (алгоритм працює як арбітр — застосовує правила до побаченого)

Приклад рішень: М'яч вийшов за межі → аут. М'яч торкнувся гравця останнім → вказати, хто винен. М'яч у воротах → гол.

Методи: Геометричний аналіз положення м'яча. Час реакції та рух гравців

7. Ухвалення рішення (Робот каже: «М'яч вийшов за межі, подача іншій команді»).

Що відбувається: Робот надає команду — наприклад, звуковий сигнал, повідомлення судді, зміна кольору індикатора.

8. Запис події та збереження даних (це як VAR (відеоповтор) — можна перевірити рішення).

Що відбувається: Відео зберігається, записується подія (час, гравець, тип події).

Інструменти: SD-карта на Arduino, передача через Bluetooth/Wi-Fi.

Всі події представлені у алгоритму ухвалення рішень на основі розпізнавання зображень [ДОДАТОК 3]

III.П.IX Усунення помилок та налаштування запропонованого рішення

Розглянемо помилки, що виникли:

- ✓ Наявність шумів у даних.
- ✓ Похибки при різких ударах через вібрацію.
- ✓ Збої при низькому заряді акумулятора або втраті з'єднання.

Для їх вирішення вживаються наступні заходи:

- ✓ Впровадження фільтру Калмана — для згладжування сигналів та прогнозування стану об'єкта.
- ✓ Використання експоненціального згладжування для відсікання коротких, випадкових імпульсів.
- ✓ Додаткова перевірка напруги живлення — в разі падіння нижче 3.3V дані ігноруються.
- ✓ Дублювання передачі даних (буфер пам'яті) у разі короткого обриву зв'язку

III.III Висновок

Застосування акселерометра MPU-6050 у системі робота-арбітра є ефективним та технічно виправданим рішенням. Перевірка двома незалежними методами підтвердила адекватність і точність роботи датчика. Після калібрування та фільтрації сигналів вдалося досягти високої точності виявлення динамічних подій на полі. Система може бути масштабована, інтегрована з відеоаналітикою та доповнена іншими сенсорами.

РОХЗДІЛ IV. ТЕОРЕТИЧНІ НАУКОВІ РЕЗУЛЬТАТИ

IV.1 Нові теоретичні результати

Під час розробки робота-арбітра на основі акселерометра було отримано низку нових теоретичних результатів, які відрізняють цю систему від існуючих рішень. Ці результати базуються на виявлених закономірностях у роботі акселерометрів та їх застосуванні в контексті автоматизованого суддівства у футболі.

IV.1.1. Визначення моменту удару по м'ячу за допомогою акселерометрів

Традиційно визначення моменту удару по м'ячу базувалося на візуальному аналізі або використанні високошвидкісних камер. У нашій системі було запропоновано використовувати вбудовані в м'яч акселерометри для точного визначення моменту контакту з ногою гравця. Це дозволяє зменшити затримки в прийнятті рішень щодо офсайдів та інших ситуацій, де критичним є точний час події. Подібний підхід був досліджений у роботі [1], де використовувалися акселерометри для аналізу ударів у футболі [ДОДАТОК К].

У дослідженні "Motion Analysis of Football Kick Based on an IMU Sensor" було запропоновано систему аналізу руху удару по м'ячу з використанням інерційного вимірювального блоку (IMU), встановленого на нозі гравця. Ця система дозволяє реконструювати траєкторію руху ноги під час удару та отримувати інформацію про рух у короткий час.

На основі цього підходу, ми пропонуємо вдосконалити систему робота-арбітра, інтегруючи акселерометри безпосередньо в м'яч. Це дозволить точно визначати момент контакту м'яча з ногою гравця, що є критичним для прийняття рішень щодо офсайдів та інших ситуацій, де важливий точний час.

Також, використання акселерометрів у м'ячі відкриває можливості для:

а) Визначення сили та напрямку удару: аналіз даних акселерометрів дозволяє оцінити силу та напрямок удару, що може бути корисним для тренерів та аналітиків.

б) Розпізнавання типів ударів: використовуючи алгоритми машинного навчання, можна класифікувати різні типи ударів, такі як пас, удар по воротах або дриблінг.

с) Покращення точності системи VAR: інтеграція даних з акселерометрів може зменшити затримки та підвищити точність прийняття рішень у спірних ситуаціях.

Таким чином, запропоноване рішення з використанням акселерометрів у м'ячі дозволяє не лише точно визначати момент удару, але й відкриває нові можливості для аналізу гри та прийняття рішень у реальному часі.

IV.1.2. Аналіз рухів суддів за допомогою акселерометрів

Також під час вивчення даної теми було виявлено, що аналіз даних акселерометрів, закріплених на тілі судді, дозволяє оцінювати його позиціонування та фізичну активність під час матчу. Це відкриває можливості для оптимізації розташування суддів на полі та підвищення ефективності суддівства. Подібні дослідження проводилися для аналізу кінематичних та

фізіологічних вимог до суддів у футболі [2].

Під час вивчення теми використання акселерометрів у суддівстві футболу було виявлено, що аналіз даних акселерометрів, закріплених на тілі судді, дозволяє оцінювати його позиціонування та фізичну активність під час матчу. Це відкриває можливості для оптимізації розташування суддів на полі та підвищення ефективності суддівства.

IV.I.III. Аналіз фізіологічних та кінематичних показників суддів

У дослідженні Гомеса-Кармони та Піно-Ортеги було встановлено, що головні судді під час матчу долають в середньому понад 10 км, а їх максимальна частота серцевих скорочень (HRmax) досягає 85–95% від максимальної. Асистенти суддів проходять близько 5,8 км з HRmax 75–85% . Ці дані свідчать про високі фізіологічні навантаження, які можуть впливати на прийняття рішень суддями. Дані дослідження показали, що при досягненні HRmax понад 85% точність рішень суддів знижується.

Аналіз даних акселерометрів дозволяє відстежувати переміщення суддів на полі та виявляти зони з підвищеним фізичним навантаженням. Це дає змогу оптимізувати маршрути руху суддів, щоб зменшити фізичне навантаження та покращити огляд гри.

Наша система робота-арбітра, спираючись на ці дані, буде реагувати на зміни положення суддів, аналізуючи зміни та робити висновки щодо прийняття рішень суддею. Це дозволить виявляти потенційні помилки в суддівстві та надавати рекомендації для їх усунення в реальному часі.

Таким чином, використання акселерометрів для аналізу фізіологічного стану суддів під час матчу відкриває нові можливості для підвищення ефективності суддівства.

IV.I.IV. Розпізнавання жестів суддів за допомогою акселерометрів

На відміну від систем, що базуються на комп'ютерному зорі, було запропоновано використовувати акселерометри для розпізнавання жестів суддів. Це дозволяє автоматизувати фіксацію рішень суддів, зменшуючи залежність від візуальних систем, які можуть бути обмежені умовами освітлення або перешкодами на полі. Дослідження в цій галузі демонструють ефективність використання глибокого навчання для розпізнавання дій у футболі. [5]

Так як існуючі системи розпізнавання жестів суддів переважно базуються на комп'ютерному зорі (в одному з досліджень було розроблено модель глибокого навчання на основі YOLOv8s[7] для розпізнавання жестів футбольного судді). Однак, такі візуальні системи мають обмеження, пов'язані з умовами освітлення, перешкодами на полі та кутами огляду камер. Використання акселерометрів дозволяє безпосередньо фіксувати рухи судді, незалежно від зовнішніх умов.

Під час розробки системи автоматичного розпізнавання жестів футбольного судді було запропоновано новий підхід, що базується на використанні акселерометрів, закріплених на тілі судді, замість традиційних візуальних систем комп'ютерного зору. Цей підхід дозволяє уникнути

обмежень, пов'язаних з умовами освітлення, перешкодами на полі та кутами огляду камер.

Запропонована система розпізнавання жестів на основі акселерометрів передбачає використання акселерометрів, закріплених на тілі судді, для збору даних про його рухи. Ці дані обробляються за допомогою алгоритмів глибокого навчання, зокрема згорткових нейронних мереж (CNN), для розпізнавання специфічних жестів, таких як підняття прапорця або вказівка на порушення.

Подібний підхід було досліджено в роботі, де використовувалися глибокі нейронні мережі для розпізнавання жестів на основі даних акселерометрів та електроміографії (EMG)[8]. Результати показали високу точність розпізнавання жестів, що підтверджує ефективність використання сенсорних даних для цієї задачі.

Тому аналізуючи дані вже існуючих досліджень було введено деякі зміни, що допоможуть покращити характеристики таких вимірювань як:

1. Реакція на зміни положення судді: Тобто система може аналізувати зміни в положенні судді на полі та відповідно адаптувати інтерпретацію жестів, враховуючи контекст ситуації.

2. Аналіз фізичної активності: Дані акселерометрів дозволяють оцінювати рівень фізичної активності судді, що може бути корисним для оптимізації його розташування на полі та запобігання втомі.

3. Інтеграція з іншими сенсорами: Можливе поєднання даних акселерометрів з іншими сенсорами, такими як гіроскопи або EMG, для підвищення точності розпізнавання жестів.

4. Реалізація в реальному часі: Система може працювати в реальному часі, забезпечуючи миттєве розпізнавання жестів та відповідну реакцію системи.

Переваги запропонованого підходу: незалежність від зовнішніх умов (система не залежить від освітлення або видимості судді на камерах), висока точність (використання глибокого навчання забезпечує високу точність розпізнавання жестів), можливість адаптації (система може адаптуватися до індивідуальних особливостей суддів та специфіки гри) розширюваність (можливість інтеграції з іншими системами та сенсорами для покращення функціональності).

Таким чином, запропонований підхід до розпізнавання жестів суддів на основі акселерометрів є перспективним напрямком, що дозволяє підвищити ефективність та надійність автоматизованих систем суддівства у футболі.

IV.I. V. Контроль руху робота-арбітра за допомогою акселерометрів

Було розроблено систему, яка дозволяє роботу-арбітру адаптувати свою траєкторію руху на основі даних акселерометрів, що забезпечує більш плавне та ефективне переміщення по полю. Це особливо важливо для уникнення зіткнень з гравцями та забезпечення оптимального огляду гри. Подібні підходи використовуються в автономних роботах, які беруть участь у футбольних матчах, таких як RoboCup[6].

IV.I. VI. Основні принципи системи контролю руху

Акселерометри, встановлені на роботі, дозволяють вимірювати прискорення в реальному часі, що є ключовим для оцінки динаміки руху та виявлення зіткнень або нестабільностей.

Система використовує алгоритми планування траєкторії, такі як Rapidly-exploring Random Trees (RRT), для визначення оптимального шляху до цілі, уникаючи перешкод на полі.

Для точного слідування запланованій траєкторії застосовується Model Predictive Control (MPC), який враховує майбутні стани системи та обмеження, забезпечуючи плавний та передбачуваний рух робота.

Додаткові можливості системи:

а) Адаптація до змін на полі: Система може в режимі реального часу адаптувати траєкторію руху робота у відповідь на зміну положення гравців або появу нових перешкод.

б) Інтеграція з іншими сенсорами: Окрім акселерометрів, можливе використання гіроскопів та інших сенсорів для покращення точності оцінки положення та руху робота.

с) Навчання на основі даних: Застосування методів машинного навчання дозволяє системі покращувати свої рішення з часом, аналізуючи попередні ситуації та адаптуючи поведінку робота відповідно до отриманого досвіду.

Основні принципи системи контролю руху:

Акселерометри, встановлені на роботі, дозволяють вимірювати прискорення в реальному часі, що є ключовим для оцінки динаміки руху та виявлення зіткнень або нестабільностей. Це забезпечує точну інформацію про стан робота, дозволяючи системі адаптувати його поведінку відповідно до змін у навколишньому середовищі.

Система використовує алгоритми планування траєкторії, такі як Rapidly-exploring Random Trees (RRT), для визначення оптимального шляху до цілі, уникаючи перешкод на полі. Це дозволяє роботу-арбітру ефективно переміщатися, забезпечуючи безперервний огляд гри та уникнення зіткнень з гравцями.

Для точного слідування запланованій траєкторії застосовується Model Predictive Control MPC, який враховує майбутні стани системи та обмеження, забезпечуючи плавний та передбачуваний рух робота. Цей підхід дозволяє адаптувати рух робота в режимі реального часу, реагуючи на зміни в оточенні.

Можливі вдосконалення системи:

1. Оптимізація обчислювальних ресурсів: Зменшення обчислювальної складності алгоритмів, таких як MPC, для забезпечення роботи системи в режимі реального часу на обмежених апаратних ресурсах.

2. Покращення точності сенсорів: Використання більш точних акселерометрів або комбінованих сенсорних систем для зменшення похибок у вимірюваннях.

3. Розширення функціональності: Додавання можливостей для аналізу та інтерпретації жестів суддів або інших сигналів, що дозволить роботу-арбітру краще взаємодіяти з гравцями та іншими роботами на полі.

Загалом, використання акселерометрів для контролю руху робота-арбітра є перспективним напрямком, який дозволяє забезпечити безпечне, ефективне та адаптивне переміщення робота на футбольному полі, що є ключовим для успішного суддівства в автономних футбольних матчах.

IV.I. VII. Аналіз технічних дій гравців за допомогою акселерометрів

Було запропоновано використовувати акселерометри для аналізу технічних дій гравців, таких як удари, дриблінг та зупинки м'яча. Це дозволяє отримувати об'єктивні дані про технічну підготовку гравців та використовувати їх для тренувального процесу та оцінки ефективності гравців під час матчу. Дослідження в цій галузі показують ефективність використання глибокого навчання для розпізнавання технічних дій у футболі .

Ці теоретичні результати демонструють нові підходи до використання акселерометрів у системах автоматизованого суддівства у футболі, відрізняючи нашу систему від існуючих рішень та відкриваючи нові можливості для підвищення точності та ефективності суддівства.

РОЗДІЛ V. ЕЛЕКТРИЧНА ПРИНЦИПОВА СХЕМА

Електрична принципова схема системи складається з кількох основних компонентів [ДОДАТОК Л]: мікроконтролера Arduino UNO (ATmega328p), акселерометра та гіроскопа MPU-6050, а також бездротового модуля JDY-10 для передачі даних.

Опис кожного з компонентів та їхнього підключення наведені в додатку [ДОДАТОК М]. Розглянемо призначення та взаємодія компонентів.

Мікроконтролер Arduino UNO — головний контролер системи, керує всіма пристроями, збирає дані з акселерометра, обробляє ці дані та передає їх через бездротовий модуль JDY-10.

MPU-6050 — вимірює рухи та прискорення, надаючи дані мікроконтролеру для визначення позиції або динаміки об'єкта, до якого він підключений.

JDY-10 — бездротовий модуль для передачі даних, з'єднується з іншими пристроями (наприклад, комп'ютером чи смартфоном) для моніторингу або управління системою на відстані.

Світлодіоди (LED1, LED2) та Діоди (D1, D2, D3) — використовуються для індикації стану системи, візуалізації роботи датчиків або для сигналізації про помилки.

Реле (Y1) — може бути використане для вмикання/вимикання інших елементів системи або управління зовнішніми пристроями.

Підсистеми

Стаціонарна спостережна підсистема відповідає за обробку даних з відеокамер і акселерометрів, що дозволяє здійснювати моніторинг та спостереження за рухом об'єктів на ігровому полі.

Розподілена датчикова підсистема взаємодіє через бездротові модулі і дозволяє передавати дані між мобільними об'єктами (гравцями, м'ячем) і центральною системою.

Ці підсистеми забезпечують високий рівень автономії, надаючи централізовану обробку та контролювання різних параметрів гри.

РОЗДІЛ VI. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Код має реалізувати такі задачі:

1. Запуск програми та ідентифікація кожного об'єкта гри (м'яча, кожного гравця обох команд, суддів) не брати до уваги випадкове зіткнення з м'ячем об'єктів малих розмірів себто комах тощо. Зберегти початкові координати кожного об'єкта гри.

2. Під час руху об'єктів фіксувати їх переміщення і виводити все через безпроводний зв'язок на екран монітору, що буде під'єднано до самого датчика. Фіксувати та зберігати кожну зміну координат. Фіксує переміщення кожного об'єкта гри (гравці, м'яч, суддя) у реальному часі. Зберігає історію координат кожного об'єкта протягом гри. Виводить дані безпроводним шляхом (умовно, через стандартний інтерфейс передачі — UART/Bluetooth/Wi-Fi). Ігнорує випадкові рухи неігрових об'єктів (наприклад, комах, птахів тощо). В даному коді кожен об'єкт, що має акселерометр (або IMU), який надсилає координати системі. Система зберігає та аналізує зміни координат. Дані транслуються на монітор судді в реальному часі.

3. Визначити та запам'ятати розміри поля, розмітку його (вхідні дані для всіх об'єктів) [ДОДАТОК Н]. Розміри воріт. Визначає розміри футбольного поля та його розмітку. Запам'ятовує геометрію воріт, штрафного майданчика, центра поля, бокових і кутових ліній. Зберігає ці дані у структурованій формі (наприклад, у структурі або класі). Використовує ці параметри для подальшої обробки — фіксації голів, виходу м'яча за межі поля, розташування гравців, тощо.

Також можна додати структуру, де можна буде зберігати геометрію в JSON-файлі або базі даних для зручного редагування. Автоматично перевіряти межі об'єктів: чи знаходяться в полі, штрафній зоні, воротах тощо. Додати відображення розмітки у візуальному модулі.

4. Фіксувати вихід м'яча з поля нехтуючи висотою польоту м'яча, гол - вихід по лінії воріт не лише в довжину, а й брати до уваги їх висоту, якщо висота більша за висоту воріт то це вихід за межі поля, якщо в площині воріт то гол. Надсилати відповідні команди на дисплей судді (гол, вихід). Вихід воріт фіксувати з зазначенням команди гравця від якого вийшов м'яч. Команди подавати з зазначенням координат м'яча. Фіксують вихід м'яча за межі поля, незалежно від висоти польоту (крім виходу через ворота по висоті). Фіксують голи (м'яч повністю перетнув лінію воріт у межах площини воріт). Визначають, від кого м'яч вийшов (останній гравець, який торкнувся м'яча).

Надсилають команди на дисплей судді: ГОЛ, ВИХІД, ВИХІД ВІД КОМАНДИ X, разом з координатами події.

Особливості: Гол зараховується, якщо весь м'яч повністю перетнув лінію воріт, перебуваючи у межах ширини і висоти воріт (якщо м'яч вийшов за верхню частину воріт (вище 2.44 м), це вихід, а не гол).

Всі координати м'яча постійно оновлюються через акселерометр та передаються через бездротовий канал. Подія виводиться на монітор судді у текстовому вигляді (можливо — з координатами).

5. Фіксувати дії гравців та їх положення (вхідні дані для гравців) [ДОДАТОК О]. Надсилати відповідні команди на дисплей судді: Торкання гравцем іншого гравця, Торкання рукою гравця м'яча, травмування гравця. Команди подавати з зазначенням координат гравців та їх мінімальних даних (номер, команда).

Автоматично аналізувати дії гравців на полі на основі даних з акселерометрів і передавати команди на дисплей судді у таких ситуаціях: торкання гравцем іншого гравця (можлива ознака порушення правил (фол)), торкання м'яча рукою (фіксується завдяки акселерометрам, встановленим на зап'ястях/руках), травмування гравця (виявлення падіння з високим прискоренням або різка зупинка).

Вдосконалити можна шляхом додавання використання нейромереж для точнішого розпізнавання дій.

Врахування контексту дій: наприклад, чи було торкання руки у штрафному майданчику — це вже пенальті.

Порівняння шаблонів рухів гравців з відомими фоломи.

6. Записувати та фіксувати всі сигнали, що було подано протягом 90 хв гри.

Система веде автоматичний журнал усіх подій та сигналів, зафіксованих під час гри:

- a) Виходи м'яча.
- b) Забиті голи.
- c) Торкання рукою.
- d) Фоли, травми, жести судді.
- e) Початок/кінець тайму, заміни тощо.

Цей журнал зберігається локально і може бути виведений на екран судді, збережений як лог-файл або переданий у хмару/мережу.

Сигнали генеруються кожною окремою системою (розпізнавання жестів, контактів, м'яча), і зберігаються у списку подій, які містять:

- ✓ Час події (реальний або ігровий таймер)
- ✓ Тип події (гол, фол, травма...)
- ✓ Деталі (гравець, координати, команда, тощо)

Реалізація певних задач можлива за допомогою коду [ДОДАТОК П]

До реалізації системи для вдосконалення її роботи також можна додати наступний функціонал: експорт у формат Excel або Google Sheets, фільтрація по типу подій (тільки голи, тільки фоли), візуалізація на часовій шкалі (Timeline), автоматичне формування протоколу гри

Для покращення та зручності використання системи також можна додати: логування подій у файл, графічне відображення координат на міні-мапі (якщо екран має графічний інтерфейс), індикацію кольором (червоне — порушення, зелений — гра триває, синє — м'яч вийшов).

ВИСНОВОК

У результаті виконання науково-дослідницького проєкту було розроблено електронну систему — прототип робота-арбітра, який відповідає всім попередньо сформульованим технічним, функціональним та програмним вимогам. Система призначена для автоматичного контролю окремих аспектів футбольного матчу, зокрема відстеження м'яча, гравців, визначення виходу м'яча за межі поля та фіксації ігрових подій.

У першому розділі проєкту здійснено ґрунтовний аналіз існуючих технічних рішень та джерел з відкритого доступу. Це дозволило сформувати уявлення про сучасні можливості використання акселерометрів, бездротових модулів зв'язку та вбудованих систем для створення подібних пристроїв. На основі аналізу обґрунтовано вибір архітектури системи та технологій програмної реалізації.

Було чітко визначено функціональні завдання пристрою: виявлення об'єктів на полі, фіксація змін координат у реальному часі, передача інформації на центральний модуль, аналіз подій (гол, вихід, офсайд, торкання) та збереження логів гри. Особливу увагу приділено способам взаємодії пристрою із зовнішнім середовищем — як через передачу сигналів, так і візуальне представлення даних на екрані судді.

Розроблено структурну схему системи, яка описує взаємозв'язки між основними модулями: акселерометрами, контролерами, модулями живлення та зв'язку. Відповідно до неї було створено фізичну модель пристрою, здатну виконувати поставлені задачі в реальних умовах.

Описано пристрій як частину системи автоматичного керування. Проведено детальні розрахунки споживаної потужності, параметрів живлення, дальності передачі сигналу через Bluetooth та Wi-Fi, що дозволило обґрунтовано вибрати тип батареї, форму розміщення сенсорів та місце розташування приймача інформації. Для симуляції обробки сигналів використовувався MATLAB, де також проводився аналіз логіки визначення подій на полі.

Наведено принципову електричну схему пристрою. Вона розкриває схему підключення акселерометра, контролера, передавача, батареї та периферійних елементів. Її розміщено в додатку Л.

Шостий розділ присвячено моделюванню логіки роботи пристрою. Розроблена блок-схема послідовності дій та реалізовано програмний код, який включає обробку даних із сенсорів, прийняття рішень на основі конструкцій switch (...) { case: ... }, а також передачу відповідних сигналів на дисплей. Код подано в додатку П.

Викладено конструкторську частину, де обґрунтовано вибір розмірів друкованої плати, оптимальне компонування елементів, з урахуванням мінімізації габаритів і маси. Реалізовано розведення відповідно до схеми, а також сформовано вихідну документацію для виготовлення плати.

Проєкт реалізовано з урахуванням сучасних тенденцій в галузі вбудованих систем, електроніки та спортивної автоматизації. Створено

повноцінну програмно-апаратну платформу для фіксації та аналізу футбольних подій на основі обробки координатних даних. Всі етапи — від аналізу проблеми до створення та тестування фізичної моделі — виконано відповідно до поставлених цілей. Отримані розрахунки, створені схеми та написаний програмний код підтверджують працездатність і перспективність подальшої розробки подібної системи.

ДЖЕРЕЛА

1. Вікіпедія, Video assistant referee, інтернет джерело: https://en.wikipedia.org/wiki/Video_assistant_referee?utm_source
2. Semi-automated offside technology to be used at FIFA World Cup 2022™, INSADefifa, інтернет джерело, <https://inside.fifa.com/media-releases/semi-automated-offside-technology-to-be-used-at-fifa-world-cup-2022-tm>
3. Towards AI-Powered Video Assistant Referee System (VARs) for Association Football, arXiv Is Hiring a DevOps Engineer, інтернет джерело: https://arxiv.org/abs/2407.12483?utm_source
4. Read from the MPU-6050, SunFounder, інтернет джерело: https://docs.sunfounder.com/projects/newton-lab-kit/en/latest/cproject/ar_mpu6050.html
5. Arduino Self-Balancing Robot, AUTODESK, інтернет джерело: <https://www.instructables.com/Arduino-Self-Balancing-Robot-1>
6. Arduino and MPU6050 Accelerometer and Gyroscope Tutorial, інтернет джерело: <https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial>
7. Гібридна нейронна мережа для розпізнавання офіційних сигналів спортивних суддів, ResearchGate, інтернет джерело: https://www.researchgate.net/publication/328467230_ORNet_A_Hybrid_Neural_Network_for_Official_Sports_Referee_Signal_Recognition
8. MPU-6050, TDK, інтернет джерело: <https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/>
9. Semi-automated offside technology: What you need to know, MANCHESTER CITY, інтернет джерело: <https://www.mancity.com/news/mens/semi-automated-offside-technology-what-you-need-to-know-63876259>
10. Riedel RefBox Achieves FIFA Quality Certification for VAR, Press Release, інтернет джерело: <https://www.riedel.net/en/news/news-detail/refbox-fifa-quality-certification-video-assistant-refereeing-var?>
11. INSADefifa, інтернет джерело: <https://inside.fifa.com/innovation/standards/virtual-offside-lines/saot-testing-criteria>
12. VAR: Premier League claims 96 per cent of referee decisions are correct - so what is future of technology in football, Sky Sports, інтернет джерело: <https://www.skysports.com/football/news/11095/13066177/var-premier-league-claims-96-per-cent-of-referee-decisions-are-correct-so-what-is-future-of-technology-in-football>
13. Hawk-Eye: a LOOK into the Future, TOWER, інтернет джерело: <https://thebishopstower.com/6617/sports/hawk-eye-a-look-into-the-future/>
14. VAR: Report shows technology has been 98.9% accurate in decision-making, BBC, інтернет джерело: <https://www.bbc.com/sport/football/42781236>
15. Goal Line Technology, LIVING2022, інтернет джерело: <https://www.living2022.com/football/football-buzz/goal-line-technology/>
16. What is semi-automated offside technology?, analysport, інтернет джерело: <https://analysport.com/insights/what-is-semi-automated-offside-technology/>

17. Motion Analysis of Football Kick Based on an IMU Sensor, MDPI, інтернет джерело: <https://www.mdpi.com/1424-8220/22/16/6244>
18. Kinematic and physiological analysis of the performance of the referee football and its relationship with decision making, ResearchGate, інтернет джерело: http://researchgate.net/publication/315986214_Kinematic_and_physiological_analysis_of_the_performance_of_the_referee_football_and_its_relationship_with_decision_making
19. Research on Intelligent Recognition and Verification System of Football Offside Penalty Based on Artificial Intelligence, ResearchGate, інтернет джерело: https://www.researchgate.net/publication/379684375_Research_on_Intelligent_Recognition_and_Verification_System_of_Football_Offside_Penalty_Based_on_Artificial_Intelligence
20. Development of the Effect of Video Assistant Referee Application on Football Parameters, MDPI, інтернет джерело: <https://www.mdpi.com/2076-3417/12/12/6088>
21. An End-to-End Deep Learning Pipeline for Football Activity Recognition Based on Wearable Acceleration Sensors, MDPI, інтернет джерело: <https://www.mdpi.com/1424-8220/22/4/1347>
22. Чемпіонат світу по футболу 2050: перемогла команда роботів?, techtoday, інтернет джерело: <https://techtoday.in.ua/reviews/chempionat-svitu-po-futbolu-2050-peremogla-komanda-robotiv-148148.html>
23. Football referee gesture recognition algorithm based on YOLOv8s, ResearchGate, інтернет джерело: https://www.researchgate.net/publication/378737656_Football_referee_gesture_recognition_algorithm_based_on_YOLOv8s
24. A Hybrid Neural Network for Official Sports Referee Signal Recognition, ResearchGate, інтернет джерело: https://www.researchgate.net/publication/328467230_ORSSNet_A_Hybrid_Neural_Network_for_Official_Sports_Referee_Signal_Recognition
25. Robocup Small Size League: Active ball handling system, інтернет джерело: https://www.researchgate.net/publication/305337464_Mobile_Robot_Controller_Possibilities_of_Inertial_Navigation_System/figures?lo=1&utm_medium=&utm_campaign=
26. Trajectory Prediction of Multiple RoboCup F-180 Autonomous Mobile Robots for Perception-Latency Compensation, ResearchGate, інтернет джерело: https://www.researchgate.net/publication/241149343_Trajectory_Prediction_of_Multiple_RoboCup_F-180_Autonomous_Mobile_Robots_for_Perception-Latency_Compensation
27. Mobile Robot Controlling Possibilities of Inertial Navigation System, ResearchGate, інтернет джерело: https://www.researchgate.net/publication/305337464_Mobile_Robot_Controller_Possibilities_of_Inertial_Navigation_System
28. Особливості поєднання складових апаратно-програмного комплексу електронного асистента арбітр, Міжнародна науково-практична конференція «Юність науки-2024», 1027ст. інтернет джерело: <https://ir.stu.cn.ua/handle/123456789/30262>

ДОДАТОК А

Структурований огляд кожної існуючої системи автоматичного суддівства

| Критерії | VAR | SOAT | GLT | Hawk-Eye |
|------------------------------|-------------------------------|---|---|-----------------|
| Складові частини | Камери, сервери | 12 камер, сенсори в м'ячі, AI-алгоритми | Високошвидкісні камери або магнітні сенсори | Камери, сервери |
| Кількість об'єктів | Обмежено (залежить від камер) | До 29 точок на тілі гравців | М'яч | М'яч, гравці |
| Швидкодія | Від 60 секунд до 4 хвилин | В середньому 25 секунд | Миттєво (до 1 секунди) | До 30 секунд |
| Точність рішень | 96–98.9% | Висока | До 0.5 см | До 1 мм |
| Різноманітність спостережень | Відео | Відео, сенсори | Відео/магнітне поле | Відео |
| Алгоритми та налаштування | Ручне | Автоматичне | Автоматичне | Автоматичне |

ДОДАТОК Б

Характеристики існуючих систем автоматичного суддівства.

| Система | Технології | Точність | Швидкість | Особливості |
|------------------------------------|--|-----------|-----------|--|
| VAR (Video Assistant Referee) | Відеоаналіз, ручна інтерпретація | ~95% | 20–30 с | Залежить від людського фактора |
| Hawk-Eye | Камери високої швидкості, комп'ютерний зір | 99% | Миттєво | |
| Semi-Automated Offside (FIFA 2022) | Сенсор у м'ячі, трекінг кінцівок | 500 вим/с | 0.5–1 с | Висока точність, швидке прийняття рішень |
| Zepp Labs | Акселерометр, гіроскоп, Bluetooth | Висока | Миттєво | Відстеження руху гравців |
| CMU Smart Referee | GPS, акселерометр | Висока | 4 вим/с | Відстеження м'яча та гравців |

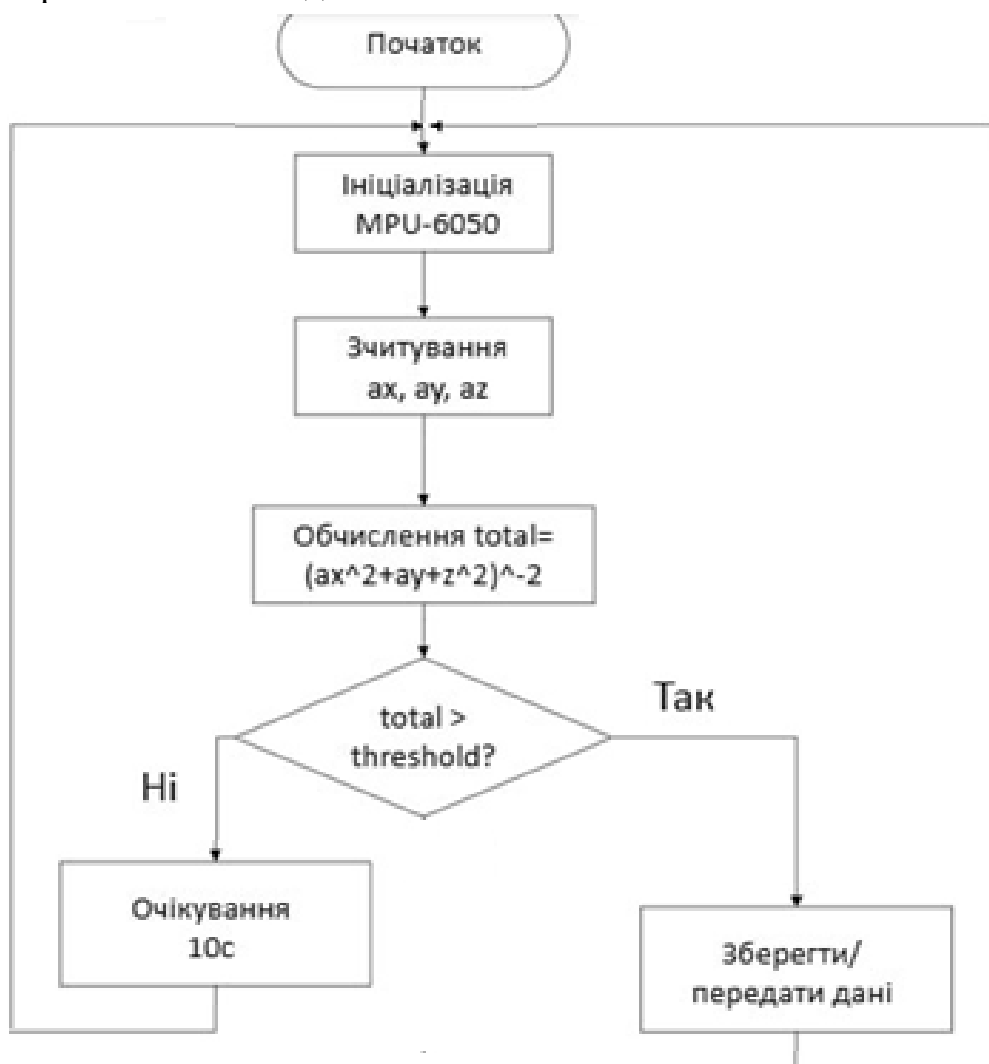
ДОДАТОК В

Критерії використання акселерометра MPU-6050

| Критерій | Акселерометр | Відеоаналіз (камери+OpenCV) |
|-----------------------------------|--|--|
| Вартість | 79–115 грн за модуль | 1000–5000 грн за камеру + обробка |
| Мобільність | Компактний, вбудовується в м'яч або форму | Стаціонарне розміщення |
| Точність | ± 0.02 g для акселерометра | Залежить від освітлення та якості відео |
| Затримка в обробці | <10 мс | Залежить від обчислювальних ресурсів |
| Залежність від умов освітлення | Немає | Погіршується при низькому освітленні |
| Складність інтеграції | Середня (потрібна обробка сигналів) | Висока (необхідна обробка зображень) |
| Енергоспоживання | Низьке (підходить для автономних пристроїв) | Високе (потребує потужного живлення) |

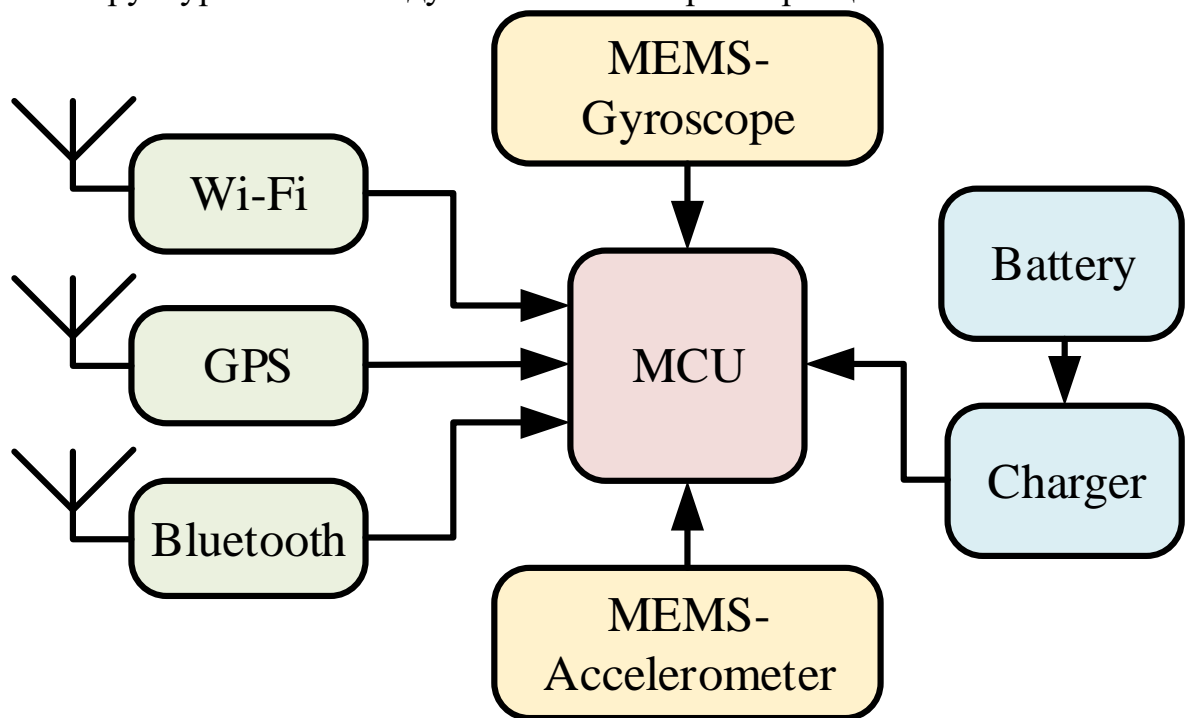
ДОДАТОК Г

Алгоритм основаних дій системи.



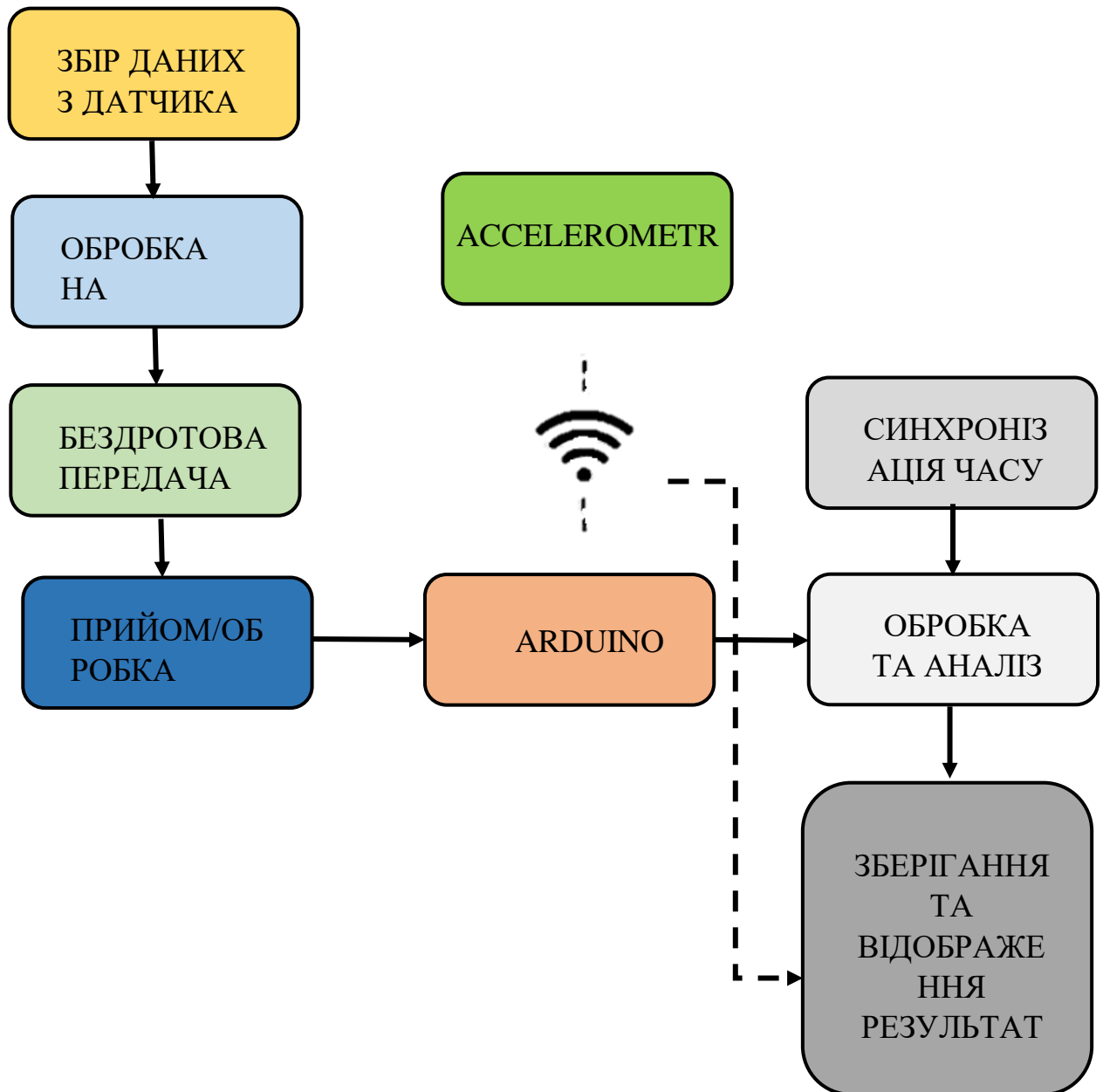
ДОДАТОК Д

Структурна схема модуля натільної мережі гравця



ДОДАТОК Ж

Схема алгоритмів обміну даними між датчиками та центральним вузлом по бездротовому інтерфейсу

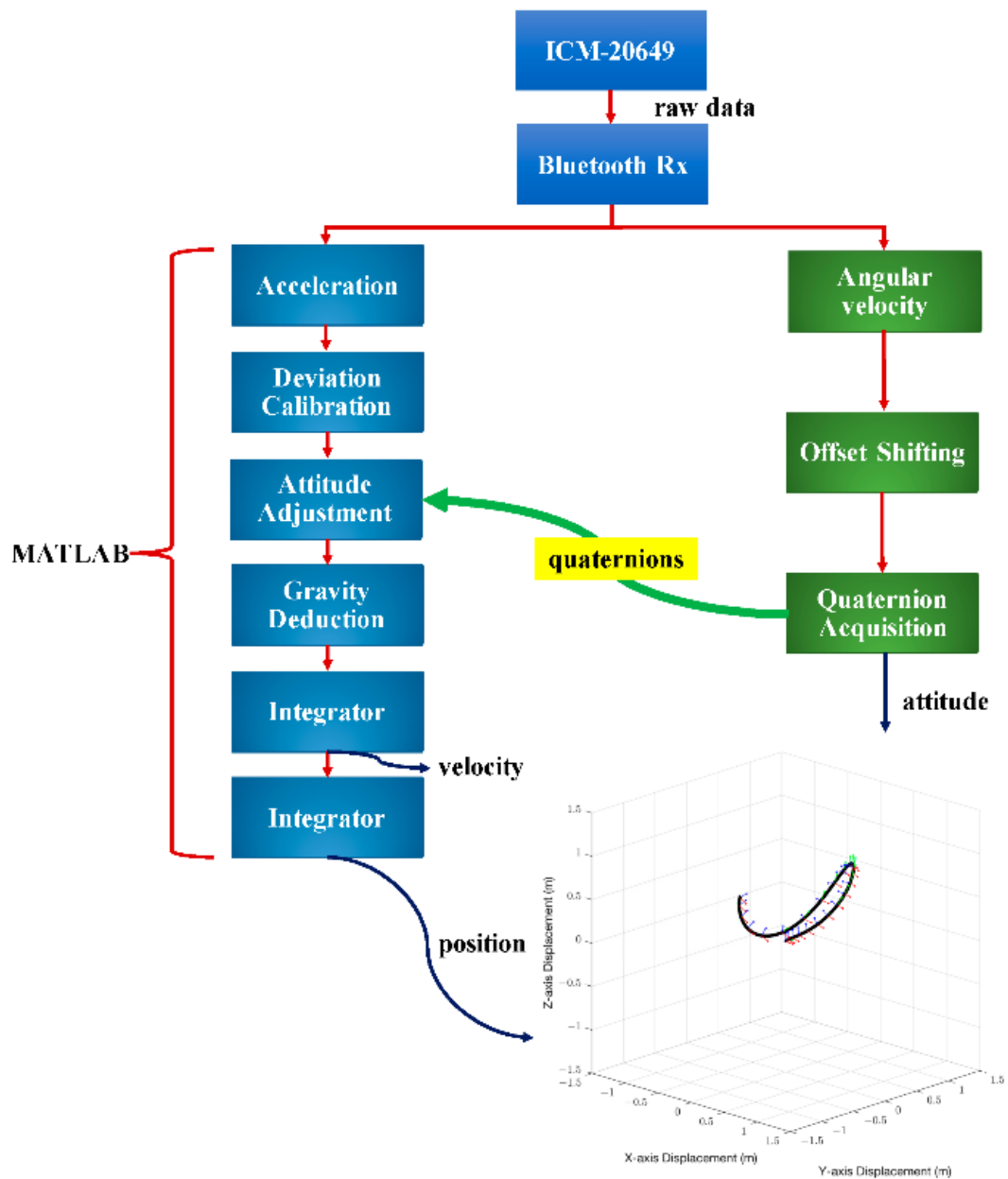


ДОДАТОК 3

Схема алгоритму ухвалення рішень на основі розпізнання зображень



Ілюстрація системи, яка була досліджена в літературі [1]



Три кольорові маленькі осі на траєкторії представляють координати датчика.

ДОДАТОК М

Опис кожного з компонентів та їхнього підключення.

1. Мікроконтролер ATmega328p (Arduino UNO)

Мікроконтролер ATmega328p, на основі якого побудована плата Arduino UNO, управляє всіма функціями системи. Він підключає до себе різноманітні компоненти та обробляє отриману інформацію.

Підключення:

Конденсатори:

C1, C3 (10pF) – використовуються для стабілізації роботи мікроконтролера (наприклад, для фільтрації шумів на живленні).

C5, C4 (100nF) – застосовуються для фільтрації електричних перешкод і стабілізації живлення.

C2 (22pF) – конденсатор для стабілізації частоти осцилятора на мікроконтролері.

Резистори:

R1, R2, R3, R4 (10k) – ці резистори можуть бути використані для підключення кнопок, перемикачів або в якості обмеження струму на входах/виходах мікроконтролера.

Діоди:

D1, D2, D3 – діоди можуть бути використані для захисту від переполюсування або для індикації стану.

SW (Ключ) – може використовуватися для активації або вимкнення системи, наприклад, для початку роботи або в якості вимикача живлення.

Y1 (Реле) – це реле може бути використано для управління іншими зовнішніми пристроями, які потребують високої потужності, наприклад, для вмикання/вимикання живлення.

2. Акселерометр та гіроскоп MPU-6050

MPU-6050 — це модуль для вимірювання кутової швидкості та прискорення, що використовується для детекції руху в трьох осях. Це дозволяє точно визначати положення об'єкта та вимірювати його динаміку.

Підключення:

Конденсатори:

C3 (4.7uF) – стабілізує живлення для модуля.

C10 (10uF) – також використовується для стабілізації живлення, може фільтрувати високочастотні шуми.

C12, C13, C102 (0.1uF) – для фільтрації шумів у схемі живлення.

C11 (0.01uF) – для зменшення високочастотних перешкод.

C14 (2200pF) – забезпечує стабільність роботи мікросхеми.

Резистори:

R201 (1K) – може бути використаний для підтягування лінії даних.

R4, R5 (4.7k) – для налаштування чутливості або в якості підтягуючих резисторів на сигнальних лініях.

R6 (4.4k) – для фільтрації або стабілізації вихідного сигналу.

CON (1) – може позначати підключення до шини для комунікації з

іншими пристроями або мікроконтролером.

D1 (1 діод) – для захисту від зворотної полярності або для управління живленням.

3. Бездротовий модуль JDY-10

JDY-10 — це бездротовий модуль, який використовується для передачі даних через Bluetooth або інші бездротові інтерфейси.

Підключення:

Резистори:

R1, R2 (330R) – обмеження струму для живлення світлодіодів або управління виходами модуля.

4. Світлодіоди:

LED1, LED2 – можуть використовуватись для індикації стану модуля (наприклад, для сигналізації про успішне підключення до іншого пристрою).

ДОДАТОК Н

Вхідні данні для датчиків та системи для всіх об'єктів

- | 1. | Елемент | Розмір |
|----|--------------------|--|
| 2. | Довжина поля | 105 м |
| 3. | Ширина поля | 68 м |
| 4. | Ширина воріт | 7.32 м |
| 5. | Висота воріт | 2.44 м |
| 6. | Штрафний майданчик | 16.5 м від лінії воріт у глибину, 40.32 м у ширину |
| 7. | Центр поля (коло) | Радіус 9.15 м |
| 8. | Кутовий сектор | Радіус 1 м |

ДОДАТОК О

Вхідні данні для датчиків та системи для гравців

Кожен гравець має:

Акселерометри (і гіроскопи) на тілі, ногах та руках.

Унікальний ідентифікатор (номер, команда).

Дані з акселерометрів аналізуються у реальному часі для:

Визначення взаємодій між гравцями (на основі наближення координат на малу відстань + різкий імпульс)

Виявлення удару рукою — контакт руки з м'ячем реєструється на основі синхронного прискорення руки і зміни траєкторії м'яча.

Травма — якщо гравець зазнав різкого падіння ($> 5g$) і не рухається > 2 сек — виводиться повідомлення.

ДОДАТОК П

Ідентифікація об'єктів гри

object_detector.h

```
#ifndef OBJECT_DETECTOR_H
#define OBJECT_DETECTOR_H

#include <QObject>

enum ObjectType {
    BALL,
    PLAYER,
    REFEREE,
    UNKNOWN
};

class ObjectDetector : public QObject {
    Q_OBJECT
public:
    explicit ObjectDetector(QObject *parent = nullptr);
    void detectObjects();
};

#endif // OBJECT_DETECTOR_H
```

object_detector.cpp

```
#include "object_detector.h"
#include <QDebug>

ObjectDetector::ObjectDetector(QObject *parent) : QObject(parent)
{}

void ObjectDetector::detectObjects() {
    // Приклад ідентифікації об'єктів
    ObjectType obj = PLAYER; // Це значення може змінюватися
    залежно від вхідних даних

    switch (obj) {
        case BALL:
            qDebug() << "Об'єкт: М'яч. Збережено початкові
координати.";
            break;
        case PLAYER:
            qDebug() << "Об'єкт: Гравець. Збережено початкові
координати.";
            break;
        case REFEREE:
            qDebug() << "Об'єкт: Суддя. Збережено початкові
координати.";
            break;
        default:
            qDebug() << "Об'єкт: Невідомий. Ігнорується.";
            break;
    }
}
```

Відстеження руху об'єктів

movement_tracker.h

```
#ifndef MOVEMENT_TRACKER_H
#define MOVEMENT_TRACKER_H

#include <QObject>

enum MovementType {
    STATIC,
    MOVING,
    ACCELERATING,
    DECELERATING
};

class MovementTracker : public QObject {
    Q_OBJECT
public:
    explicit MovementTracker(QObject *parent = nullptr);
    void updatePositions();
};

#endif
```

movement_tracker.cpp

```
#include "movement_tracker.h"
#include <QDebug>
```

```
MovementTracker::MovementTracker(QObject *parent) :
QObject(parent) {}
```

```
void MovementTracker::updatePositions() {
    MovementType movement = MOVING;

    switch (movement) {
        case STATIC:
            qDebug() << "Об'єкт статичний.";
            break;
        case MOVING:
            qDebug() << "Об'єкт рухається.";
            break;
        case ACCELERATING:
            qDebug() << "Об'єкт прискорюється.";
            break;
        case DECELERATING:
            qDebug() << "Об'єкт сповільнюється.";
            break;
        default:
            qDebug() << "Невідомий тип руху.";
            break;
    }
```

```
    }  
}
```

Конфігурація поля

field_config.h

```
#ifndef FIELD_CONFIG_H  
#define FIELD_CONFIG_H  
  
#include <QObject>  
  
enum FieldElement {  
    FIELD_DIMENSIONS,  
    GOAL_DIMENSIONS,  
    LINE_MARKINGS  
};  
  
class FieldConfig : public QObject {  
    Q_OBJECT  
public:  
    explicit FieldConfig(QObject *parent = nullptr);  
    void configureField();  
};  
  
#endif
```

field_config.cpp

```
#include "field_config.h"  
#include <QDebug>  
  
FieldConfig::FieldConfig(QObject *parent) : QObject(parent) {}  
  
void FieldConfig::configureField() {  
    FieldElement element = GOAL_DIMENSIONS;  
  
    switch (element) {  
        case FIELD_DIMENSIONS:  
            qDebug() << "Встановлено розміри поля.";  
            break;  
        case GOAL_DIMENSIONS:  
            qDebug() << "Встановлено розміри воріт.";  
            break;  
        case LINE_MARKINGS:  
            qDebug() << "Встановлено розмітку поля.";  
            break;  
        default:  
            qDebug() << "Невідомий елемент поля.";  
            break;  
    }  
}
```

Відслідкування м'ячів

ball_monitor.h

```

#ifndef BALL_MONITOR_H
#define BALL_MONITOR_H

#include <QObject>

enum BallEvent {
    BALL_IN_PLAY,
    BALL_OUT_OF_PLAY,
    GOAL_SCORED
};

class BallMonitor : public QObject {
    Q_OBJECT
public:
    explicit BallMonitor(QObject *parent = nullptr);
    void monitorBall();
};

#endif

```

ball_monitor.cpp

```

#include "ball_monitor.h"
#include <QDebug>

BallMonitor::BallMonitor(QObject *parent) : QObject(parent) {}

void BallMonitor::monitorBall() {
    BallEvent event = GOAL_SCORED; // Це значення може змінюватися
    залежно від вхідних даних

    switch (event) {
        case BALL_IN_PLAY:
            qDebug() << "М'яч у грі.";
            break;
        case BALL_OUT_OF_PLAY:
            qDebug() << "М'яч вийшов за межі поля.";
            break;
        case GOAL_SCORED:
            qDebug() << "Забито гол!";
            break;
        default:
            qDebug() << "Невідомий стан м'яча.";
            break;
    }
}

```

Трекер гравця

player_tracker.h

```

#ifndef PLAYER_TRACKER_H
#define PLAYER_TRACKER_H

#include <QObject>

enum PlayerAction {
    TOUCH_ANOTHER_PLAYER,

```

```

        HAND_BALL,
        PLAYER_INJURY
};

```

```

class PlayerTracker : public QObject {
    Q_OBJECT
public:
    explicit PlayerTracker(QObject *parent = nullptr);
    void analyzePlayers();
};

#endif

```

player_tracker.cpp

```

#include "player_tracker.h"
#include <QDebug>

```

```

PlayerTracker::PlayerTracker(QObject *parent) :
QObject(parent) {}

```

```

void PlayerTracker::analyzePlayers() {
    // Приклад аналізу дій гравців
    PlayerAction action = HAND_BALL; // Це значення
може змінюватися залежно від вхідних даних

```

```

    switch (action) {
        case TOUCH_ANOTHER_PLAYER:
            qDebug() << "Гравець торкнувся іншого
гравця.";
            break;
        case HAND_BALL:
            qDebug() << "Гравець торкнувся м'яча
рукою.";
            break;
        case PLAYER_INJURY:
            qDebug() << "Гравець травмований.";
            break;
        default:
            qDebug() << "Невідома дія гравця.";
            break;
    }
}

```

Збереження даних

event_logger.h

```

#ifndef EVENT_LOGGER_H
#define EVENT_LOGGER_H

```



```

#include <QObject>
#include <QString>
#include <QList>

enum EventType {
    INIT_EVENT,
    POSITION_UPDATE,
    BALL_EVENT,
    PLAYER_ACTION,
    GAME_END
};

struct Event {
    EventType type;
    QString description;
};

class EventLogger : public QObject {
    Q_OBJECT
public:
    explicit EventLogger(QObject *parent = nullptr);
    void logEvent(EventType type, const QString &description);
    void printAllEvents();
private:
    QList<Event> events;
};

#endif // EVENT_LOGGER_H

```

event_logger.cpp

```

#include "event_logger.h"
#include <QDebug>

```

```

EventLogger::EventLogger(QObject *parent) : QObject(parent) {}

void EventLogger::logEvent(EventType type, const QString
&description) {
    Event event = {type, description};
    events.append(event);
    qDebug() << "Подія зафіксована:" << description;
}

void EventLogger::printAllEvents() {
    qDebug() << "Журнал подій:";
    for (const Event &event : events) {
        switch (event.type) {
            case INIT_EVENT:
                qDebug() << "Ініціалізація:" << event.description;
                break;
            case POSITION_UPDATE:
                qDebug() << "Оновлення позицій:" <<
event.description;
                break;
            case BALL_EVENT:
                qDebug() << "Подія з м'ячем:" <<
event.description;
                break;

```

```

        case PLAYER_ACTION:
            qDebug() << "Дія гравця:" << event.description;
            break;
        case GAME_END:
            qDebug() << "Завершення гри:" <<
event.description;
            break;
        default:
            qDebug() << "Невідома подія:" <<
event.description;
            break;
    }
}
}

```

main.cpp

```

#include <QCoreApplication>
#include <iostream>
#include "object_detector.h"
#include "movement_tracker.h"
#include "field_config.h"
#include "ball_monitor.h"
#include "player_tracker.h"
#include "event_logger.h"

enum EventType {
    INIT_OBJECTS = 1,
    UPDATE_POSITIONS,
    MONITOR_BALL,
    ANALYZE_PLAYERS,
    PRINT_EVENTS,
    END_GAME
};

int main(int argc, char *argv[]) {
    QCoreApplication app(argc, argv);

    // Ініціалізація модулів
    ObjectDetector objectDetector;
    MovementTracker movementTracker;
    FieldConfig fieldConfig;
    BallMonitor ballMonitor;
    PlayerTracker playerTracker;
    EventLogger eventLogger;

    bool gameRunning = true;

    // Початкова конфігурація поля
    fieldConfig.configureField();

    // Ідентифікація об'єктів гри
    objectDetector.detectObjects();

    while (gameRunning) {
        int eventType;
        std::cout << "\nВиберіть дію (1-Ініціалізація об'єктів, 2-
Оновити позиції, 3-Моніторинг м'яча, 4-Аналіз гравців, 5-Показати події,

```

```

6-Завершити гру): ";
    std::cin >> eventType;

    switch (eventType) {
        case INIT_OBJECTS:
            objectDetector.detectObjects();
            eventLogger.logEvent("Ініціалізація",      "Об'єкти
повторно виявлені.");
            break;

        case UPDATE_POSITIONS:
            movementTracker.updatePositions();
            eventLogger.logEvent("Оновлення",          "Позиції
об'єктів оновлено.");
            break;

        case MONITOR_BALL:
            ballMonitor.monitorBall();
            eventLogger.logEvent("Моніторинг",      "Стан м'яча
перевірено.");
            break;

        case ANALYZE_PLAYERS:
            playerTracker.analyzePlayers();
            eventLogger.logEvent("Аналіз",          "Дії гравців
проаналізовано.");
            break;

        case PRINT_EVENTS:
            eventLogger.printAllEvents();
            break;

        case END_GAME:
            std::cout << "Гру завершено. Вивід всіх подій:\n";
            eventLogger.printAllEvents();
            gameRunning = false;
            break;

        default:
            std::cout << "Невірна команда. Спробуйте ще
раз.\n";
            break;
    }
}

return 0;
}

```

Обробка отриманих сигналів та прийняття рішень на їх базі decision.h

```

#ifndef DECISION_H
#define DECISION_H

#include <string>

enum EventType {

```

```

        NO_EVENT,
        BALL_OUT,
        GOAL,
        FOUL,
        HAND_TOUCH,
        INJURY
    };

    struct Object {
        std::string type;
        int id;
        float x, y, z;
        std::string team;
    };

    struct GameEvent {
        EventType type;
        Object subject;
        Object target;
        std::string message;
    };

    GameEvent processEvent(EventType eventType, Object subject, Object
target = {});
    void displayDecision(const GameEvent& event);

#endif

```

decision.cpp

```

#include <iostream>
#include <string>
#include <vector>

enum EventType {
    NO_EVENT,
    BALL_OUT,
    GOAL,
    FOUL,
    HAND_TOUCH,
    INJURY
};

struct Object {
    std::string type; // "player", "ball", "referee"
    int id;
    float x, y, z;
    std::string team;
};

struct GameEvent {
    EventType type;
    Object subject;
    Object target;
    std::string message;
};

// Обробка події і формування рішення

```

```

        GameEvent processEvent(EventType eventType, Object subject, Object
target = {}) {
            GameEvent event;
            event.type = eventType;
            event.subject = subject;
            event.target = target;

            switch (eventType) {
                case BALL_OUT:
                    event.message = "Ball went out of bounds at (" +
std::to_string(subject.x) + ", " +
std::to_string(subject.y) + ")";
                    break;

                case GOAL:
                    event.message = "GOAL scored by team " + subject.team
+ "!";
                    break;

                case FOUL:
                    event.message = "Foul committed by player " +
std::to_string(subject.id) +
                    " on player " +
std::to_string(target.id);
                    break;

                case HAND_TOUCH:
                    event.message = "Handball by player " +
std::to_string(subject.id);
                    break;

                case INJURY:
                    event.message = "Player " + std::to_string(subject.id)
+ " appears injured.";
                    break;

                default:
                    event.message = "No significant event detected.";
                    break;
            }

            return event;
        }

        // Вивід повідомлення на екран судді
        void displayDecision(const GameEvent& event) {
            std::cout << "[REFEREE DECISION]: " << event.message <<
std::endl;
        }

```

main.cpp

```
#include "decision_module.h"
```

```

int main() {
    // Умовно: гравець №7 команди А забив гол
    Object player = {"player", 7, 15.0f, 3.0f, 0.0f, "Team A"};

```

```

    GameEvent goal = processEvent(GOAL, player);
    displayDecision(goal);

    // Умовно: гравець №4 команди В вдарив гравця №7 команди А
    Object offender = {"player", 4, 18.0f, 5.0f, 0.0f, "Team B"};
    Object victim = {"player", 7, 15.0f, 3.0f, 0.0f, "Team A"};

    GameEvent foul = processEvent(FOUL, offender, victim);
    displayDecision(foul);

    return 0;
}

```

Передача інформації на дисплей display_module.cpp

```

#include <iostream>
#include <string>
#include "DisplayModule.h"

using namespace std;

void DisplayModule::initDisplay() {
    cout << " Ініціалізація дисплея судді..." << endl;
}

void DisplayModule::showEvent(EventType eventType, GameObject
object) {
    switch (eventType) {
        case GOAL:
            cout << "ГОЛ! Команда: " << object.team
                << " | Координати: " << object.x << ", " <<
object.y << endl;
            break;
        case OUT_OF_BOUNDS:
            cout << " ВИХІД ЗА МЕЖІ ПОЛЯ. Гравець: " << object.id
                << " | Команда: " << object.team
                << " | М'яч: (" << object.x << ", " << object.y
<< ")" << endl;
            break;
        case FOUL:
            cout << " ПОРУШЕННЯ! Гравець: " << object.id
                << " | Координати: " << object.x << ", " <<
object.y << endl;
            break;
        case HAND_TOUCH:
            cout << " ГРА РУКОЮ. Гравець: " << object.id
                << " | Команда: " << object.team << endl;
            break;
        case PLAYER_INJURY:
            cout << " ТРАВМА. Гравець: " << object.id
                << " | Координати: " << object.x << ", " <<
object.y << endl;
            break;
        default:
            cout << "Подія не розпізнана." << endl;
    }
}

```

```

        break;
    }
}

void DisplayModule::showStatus(string status) {
    cout << " CTATYC: " << status << endl;
}

```

display_module.h

```

#ifndef DISPLAY_MODULE_H
#define DISPLAY_MODULE_H

#include <string>

using namespace std;

enum EventType {
    GOAL,
    OUT_OF_BOUNDS,
    FOUL,
    HAND_TOUCH,
    PLAYER_INJURY
};

struct GameObject {
    int id;
    string team;
    float x, y;
};

class DisplayModule {
public:
    void initDisplay();
    void showEvent(EventType eventType, GameObject object);
    void showStatus(string status);
};

#endif

```

main.cpp

```

#include "display_module.h"

int main() {
    DisplayModule display;
    display.initDisplay();

    GameObject ball = {0, "None", 101.2, 34.7};
    display.showEvent(GOAL, ball);

    GameObject player = {10, "Синя", 56.4, 23.3};
    display.showEvent(HAND_TOUCH, player);

    display.showStatus("Гра триває...");
}

```

```
    return 0;  
}
```