ШИФР «ПЛАНЕРНИЙ СПОРТ»

РОЗРОБКА ЦИФРОВОГО ВАРІОМЕТРА ПЛАНЕРА ДЛЯ СУМІСНОГО ВИКОРИСТАННЯ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ ПОЛЬОТНОГО КОМП'ЮТЕРА XCSOAR

2021

ЗМІСТ

ВСТУП

В сучасному планерному спорті, як і в інших технічні видах спорту активне використовуються інформаційні технології. Ринок навігаційного обладнання для планерів складається з низки, так званих, польотних комп’ютерів, вартість яких досить висока. Але існує безкоштовне програмне забезпечення (ПЗ) для смартфонів. В Україні найбільшою популярність користується ПЗ XCSOAR. Основою рішення навігаційних задач безумовне є вбудований в смартфон модуль GPS. Основним недоліком використання XCSOAR на смартфоне є спізнення розрахункової вертикальної швидкості. А саме, оперативність отримання інформації про зміну вертикальної швидкості є головний фактор при наборі висоти в термічному потоку. Тому актуальною є розробка на основі популярної концепції «інтернет речей». Інтернет речей(IoT) – концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані передавачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку. Окрім датчиків, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Основною концепцією інтернет речей є можливість підключення об'єктів (речей), які людина може використовувати в повсякденному житті, наприклад, холодильник, кондиціонер, автомобіль, велосипед і навіть кросівки. Всі ці об'єкти (речі) повинні бути оснащені вбудованими передавачами або сенсорами, які мають можливість обробляти інформацію, що надходить з навколишнього середовища, обмінюватися нею і виконувати різні дії в залежності від отриманої інформації.

Вже зараз інтернету речей приділяється увага на найвищому рівні, зокрема

починаючи з 2009 року у Брюселі при підтримці Єврокомісії проходять конференції Annual Internet of Things, на який виступають з доповідями єврокомісари, науковці та керівники провідних IT-компаній. За прогнозами аналітиків у найближчі роки очікується справжній бум інтернету речей. Так, за прогнозами Gartner, до 2022 року кількість підключених до всесвітньої мережі пристроїв становитиме 35 мільярдів, а дохід від продажу устаткування, програмного забезпечення та послуг становитиме 2,9 трлн дол. Деякі інші аналітичні агентства висловлюють ще більш оптимістичні прогнози. Найбільші світові IT компанії вже почали перегони за лідерство на цьому ринку. Так корпорація Intel у 2014 році після випуску «SoC Edison» оголосила конкурс «Make it Wearable» з призовим фондом $1,3 млн на найкраще застосування своєї системи для концепції IoT та створила власний підрозділ «Internet of Things Solutions Group» для розвитку цього напрямку. Компанія «Google» на початку 2014 року за 3,2 млрд доларів купила невелику фірму «Nest Labs», яка займається випуском інтелектуальних термостатів. Спеціалісти цієї компанії займались впровадженням на американському ринку технологій IoT.

В даному проекті пропонується використовувати апаратні та програмні рішення технології Arduino. Arduino – торгова марка апаратно-програмних засобів для побудови систем автоматики і робототехніки. Програмна частина складається з безкоштовної програмної оболонки для написання програм, їх компіляції та програмування апаратури. Апаратна частина являє собою набір змонтованих друкованих плат, що продаються як офіційним виробником, так і сторонніми виробниками. Повністю відкрита архітектура системи дозволяє вільно копіювати або доповнювати лінійку продукції Arduino. Отже, Arduino є однією з найкращих платформ для розробки пристроїв інтернету речей. Тому для розробки проекту було обрано дану технологію.

РОЗДІЛ І. ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУ

Варіометр — пілотажний контрольно-вимірювальний прилад, яким вимірюють швидкість підйому й зниження (вертикальну швидкість) літального апарата.

У планеризмі вважається, що для нормального польоту апарат повинен мати достатню кількість механічної енергії. Енергія планера, що називається повною енергією, складається з потенціальної

$$E_{\text{П}} = mgH,$$

(1.1)

де m — маса планера, g — прискорення вільного падіння, H — висота польоту і кінетичної

$$E_K = \frac{mV^2}{2},$$

(1.2)

де V — швидкість польоту, m — маса планера, тобто

$$E = E_{\text{П}} + E_K,$$

(1.3)

де E— енергія планера, $E_{\text{П}}$— повна енергія, E_K— кінетична енергія.

Ці види енергії можуть взаємно перетворюватись — зменшувати швидкість за рахунок збільшення висоти і збільшувати швидкість за рахунок зменшення висоти. Більшість варіометрів, враховували лише зміну потенційної енергії літального апарату використовуючи тільки значення статичного тиску, незалежно чи ця зміна викликана зміною швидкості, чи в результаті впливу зовнішніх факторів (наприклад, висхідних теплових потоків). Цей факт є істотним у польоті планера, оскільки його пілота цікавить чи у місці простору перебування планера є висхідні потоки чи зміна висоти викликана за рахунок кінетичної енергії. Для вирішення цієї проблеми у планерах використовують варіометри повної енергії або скомпенсовані варіометри, які компенсують

реакцію на зміну висоти у результаті маневру, а реагують лише на зміну повної енергії.

Проте електронний варіометр, який часто поєднується з іншими покажчиками (наприклад, висотоміром) для безмоторної авіації стає основним. У такому приладі немає повітряних ємностей і трубопроводів, а є лише чутливі сенсори тиску, на основі даних з яких обчислювальні пристрої видають потрібну інформацію, в тому числі і про вертикальну швидкість.

Метою роботи є розробка інформаційної системи (варіометра) для вимірювання висоти та розрахунку вертикальної швидкості, а також передачі інформації у програмне забезпечення смартфона XCSOAR за допомогою технології Bluetoth та протоколу OpenVario

$$\$POV,E,\{V\}*\{CH\}, \tag{1.4}$$

де {V} — вертикальна швидкість, {CH} — геш-сума попередніх символів.

Для створення апаратної частини інформаційної системи було обрано:

- плата мікроконтролера Arduino Nano V3;
- датчик тиску Gy-68;
- LCD — дисплей 1602;
- резистори 4к7, 1К, 330 Ом;
- елемент живлення «Крона»;
- вимикач; макетна плата з провідниками;
- bluetooth модуль HC-06.

Для розробки програмної частини системи використані так інструменти:

- Arduino IDE;
- бібліотека Adafruit BMP280;
- бібліотека Adafruit Sensor;
- бібліотека Wire;
- бібліотека Tone;
- бібліотека LiquidCrystal;
- мова програмування с++.

РОЗДІЛ ІІ. РОЗРОБКА ЦИФРОВОГО ВАРІОМЕТРА ПЛАНЕРА ДЛЯ СУМІСНОГО ВИКОРИСТАННЯ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ ПОЛЬОТНОГО КОМП'ЮТЕРА XCSOAR НА ОСНОВІ ARDUINO

Задача даного проекту щодо розробки цифрового варіометра планера для сумісного використання з програмним забезпеченням польотного комп'ютера є задачею створення відповідної інформаційної системи. Тому доцільне проаналізувати її саме як задачу розробки інформаційної системи.

Інформаційна система — сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів. Інформаційні системи здавна знаходять (в тому чи іншому вигляді) досить широке застосування в життєдіяльності людства. Це пов'язано з тим, що для існування цивілізації необхідний обмін інформацією — передача знань, як між окремими членами і колективами суспільства, так і між різними поколіннями.

Варте підкреслити, те що інформаційні системи існують з моменту появи суспільства, оскільки на кожній стадії його розвитку існує потреба в управлінні. Місією інформаційної системи є переробка інформації, потрібної для ефективного управління всіма ресурсами організації, створення інформаційного та технічного середовища для управління її діяльністю.

Інформаційна система може існувати і без застосування комп'ютерної техніки — це питання економічної доцільності. В будь-якій інформаційній системі управління вирішуються задачі трьох типів:

- задачі оцінки ситуації (деколи їх називають задачами розпізнавання образів);
- задачі перетворення опису ситуації (розрахункові задачі, задачі моделювання);
- задачі прийняття рішень (в тому числі і оптимізаційні).

Найдавнішими і найпоширенішими ІС слід вважати бібліотеки. І, дійсно, здавна в бібліотеках збирають книжки (або їх аналоги), зберігають їх,

дотримуючись певних правил, створюють каталоги різного призначення для полегшення доступу до книжкового фонду. Видаються спеціальні журнали та довідники, що інформують про нові надходження, ведеться облік видачі.

Найстаріші (у моральному і у фізичному розумінні) ІС повністю базувалися на ручній праці. Пізніше їм на зміну прийшли різні механічні пристрої для обробки даних (наприклад, для сортування, копіювання, асоціативного пошуку тощо). Наступним кроком стало впровадження автоматизованих інформаційних систем (АІС), тобто систем, де для забезпечення інформаційних потреб користувачів використовується ЕОМ зі своїми носіями інформації. В наш час — епоху інформаційної революції — розробляється і впроваджується велика кількість найрізноманітніших АІСів з дуже широким спектром використання.

Інформаційні системи включають в себе: технічні засоби обробки даних, програмне забезпечення і відповідний персонал. Чотири складові частини утворюють внутрішню інформаційну основу:

- засоби фіксації і збору інформації;
- засоби передачі відповідних даних та повідомлень;
- засоби збереження інформації;
- засоби аналізу, обробки і представлення інформації.

Класифікація інформаційних систем. За ступенем автоматизації. В залежності від ступеня (рівня) автоматизації виділяють ручні, автоматизовані й автоматичні інформаційні системи:

- Ручні ІС - характеризуються тим, що всі операції з переробки інформації виконуються людиною.
- Автоматизовані ІС - частина функції (підсистем) керування або опрацювання даних здійснюється автоматично, а частина — людиною.
- Автоматичні ІС - усі функції керування й опрацювання даних здійснюються технічними засобами без участі людини (наприклад, автоматичне керування технологічними процесами).

Класифікація інформаційних систем за місцем діяльності.

- Наукові ІС — призначені для автоматизації діяльності науковців, аналізу статистичної інформації, керування експериментом.
- ІС автоматизованого керування — призначені для автоматизації праці інженерів-проектувальників і розроблювачів нової техніки (технології).
- ІС організаційного керування — призначені для автоматизації функції адміністративного (управлінського) персоналу. До цього класу відносяться ІС керування як промисловими (підприємства), так і непромисловими об'єктами (банки, біржа, страхові компанії, готелі і т. д.) і окремими офісами (офісні системи).
- ІС керування технологічними процесами — призначені для автоматизації різноманітних технологічних процесів (гнучкі виробничі процеси, металургія, енергетика тощо).

Інформаційна система, як система управління, тісно пов'язується, як з системами збереження та видачі інформації, так і з іншої — з системами, що забезпечують обмін інформацією в процесі управління. Вона охоплює сукупність засобів та методів, що дозволяють користувачу збирати, зберігати, передавати і обробляти відібрану інформацію.

Типи взаємодії інформаційних систем:

- Довільна взаємодія між двома окремими комп'ютерами, наприклад по модему. Обов'язкова участь оператора на приймаючої і передавальної стороні. Можливий обмін в довільному, але заздалегідь обумовленому форматі.
- Інтерактивна віддалена взаємодія комп'ютера з інформаційною системою, наприклад по протоколу http. Оператор на передавальній стороні. Як правило використовується певна форма HTML документа. Прийняті документи обробляються автоматично.
- Контрольована потокова обробка, наприклад прийом з e-mail, файл містить HTML форму, запуск якої ініціює процес обробки документа

або прийом оператором по e-mail електронних документів в обумовленому форматі і далі запуск програми обробки. Вимагає обов'язковий контроль оператора на прийнятої стороні.

- Повністю автоматизований процес прийому та обробки електронних документів в обумовленому форматі. Участь операторів не потрібно.

Отже, дослідження дозволяє зробити висновок про те, що XCSoar - це програмне забезпечення, яке встановлене в смартфоні для польотів на планері спочатку було розроблено для платформи Pocket PC. У 2005 році комерційне програмне забезпечення було надано спільноті з відкритими вихідними кодами для подальшого розвитку, і з тих пір вона постійно вдосконалюється. На даний час це багатоплатформенна програма, яка працює на пристроях Windows, Windows Mobile, Unix і навіть Android. Це програмне забезпечення випускається відповідно до Загальної публічної ліцензії GNU. Інтерфейс XCSoar надано на рис. 2.1., 2.2.

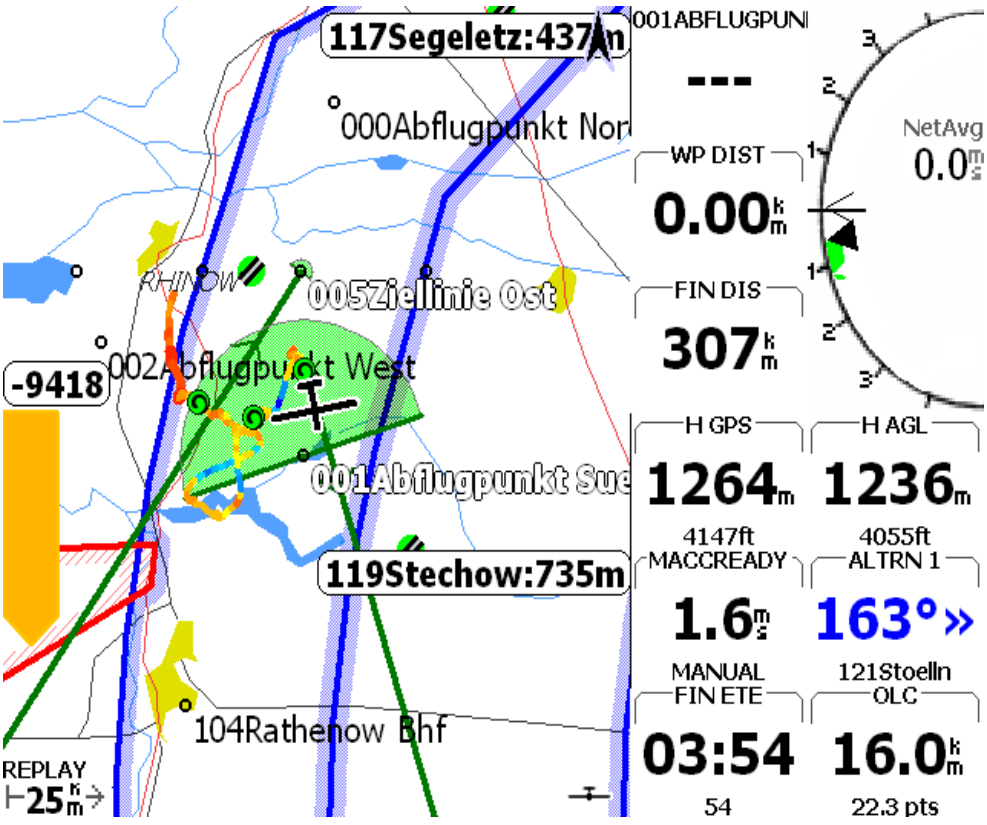

Рисунок 2.1 – Інтерфейс XCSoar

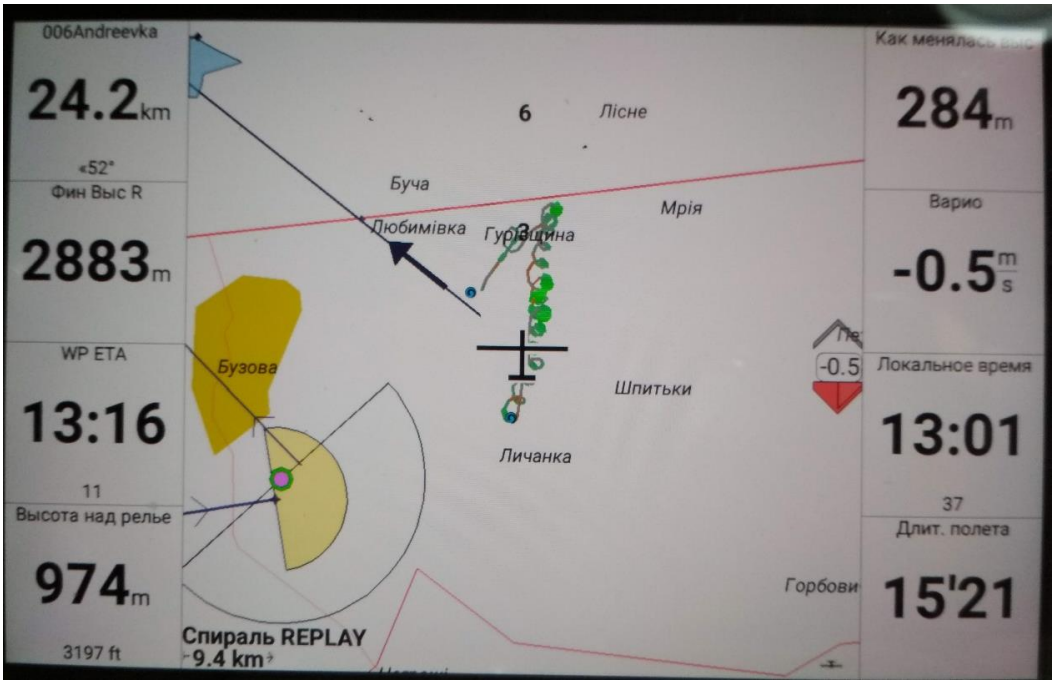

Рисунок 2.2 – Інтерфейс XCSoar в реальному польоті на висоті 974 м в режимі «Спіраль»

В даному проекті пробується наступна схема варіометра (рис. 2.3).

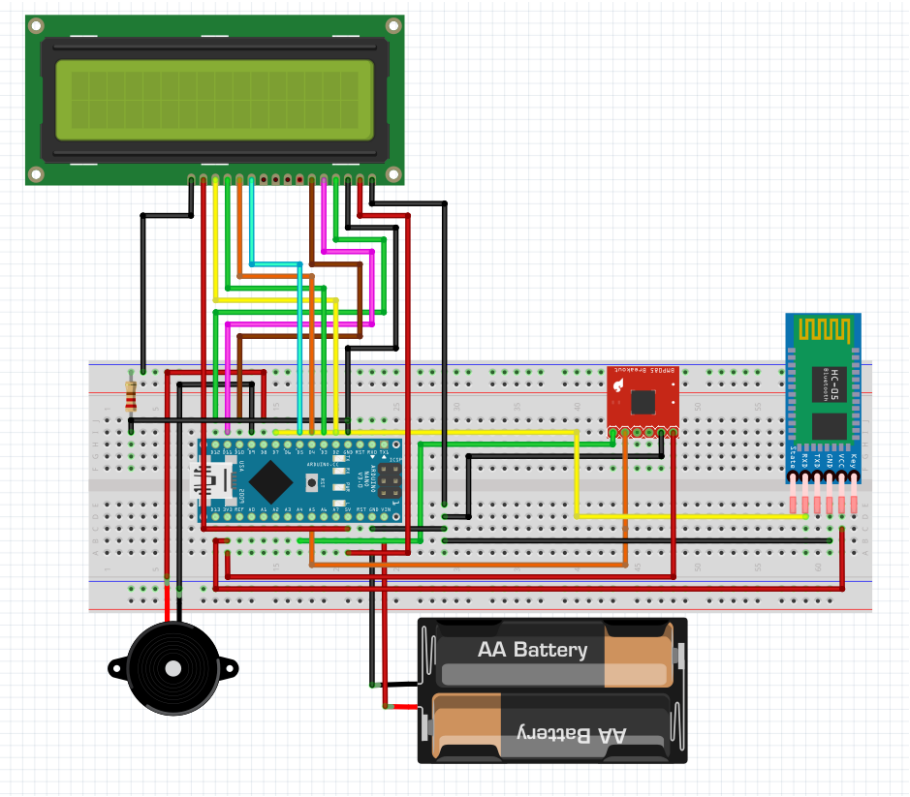

Рисунок 2.3 – Схема варіометра

Елементна база, яка використана надана на рис. 2.4

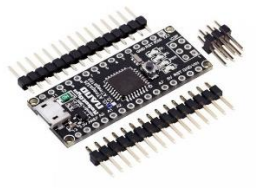	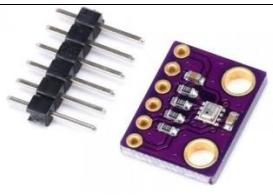		
Arduino Nano V3.0	**Барометр BMP280**	**Bluetooth HC-06**	**LCD**

Рисунок 2.4 – Елементна база варіометра

Після виготовлення дослідницький зразок варіометра має наступний вигляд (рис. 2.5).

Рисунок 2.4 – Дослідницький зразок варіометра

РОЗДІЛ ІІІ. ДОСЛІДЖЕННЯ ТА ОБРАННЯ МОВИ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРА

Програмне забезпечення було розроблено за допомогою мови програмування с++.

С ++ - компільований, статично типізований мова програмування загального призначення. Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає в себе поширені контейнери і алгоритми, введення-виведення, регулярні вирази, підтримку багатопоточності і інші можливості. С ++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником - мовою С, - найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування.

Синтаксис С ++ успадкований від мови С. Одним з принципів розробки було збереження сумісності з С. Проте, С ++ не є в строгому сенсі надмножиною С; безліч програм, які можуть однаково успішно транслюватися як компіляторами С, так і компіляторами С ++, досить велике, але не включає всі можливі програми на С.

Мова виникла на початку 1980-х років, коли співробітник фірми Bell Labs Бйорн Страуструп придумав ряд удосконалень до мови С під власні потреби. [11] Коли в кінці 1970-х років Страуструп почав працювати в Bell Labs над завданнями теорії черг (в додатку до моделювання телефонних викликів), він виявив, що спроби застосування існуючих в той час мов моделювання виявляються неефективними, а застосування високоефективних машинних мов занадто складно через їх обмеженою виразності. Так, мова Симула має такі можливості, які були б дуже корисні для розробки великого програмного забезпечення, але працює дуже повільно, а мова BCPL досить швидкий, але дуже близький до мов низького рівня і не підходить для розробки великого

програмного забезпечення.

Згадавши досвід своєї дисертації, Страуструп вирішив доповнити мову C (наступник BCPL) можливостями, наявними в мові Симула. Мова C, будучи базовою мовою системи UNIX, на якій працювали комп'ютери Bell, є швидким, багатофункціональним і стерпним. Страуструп додав до нього можливість роботи з класами та об'єктами. В результаті практичні задачі моделювання виявилися доступними для вирішення як з точки зору часу розробки (завдяки використанню Симула-подібних класів), так і з точки зору часу обчислень (завдяки швидкодії C). В першу чергу в C були додані класи (з інкапсуляцією), успадкування класів, сувора перевірка типів, inline-функції і аргументи за умовчанням. Ранні версії мови, спочатку іменувався «C with classes» («Сі з класами»), стали доступні з 1980 року.

Розробляючи C з класами, Страуструп написав програму cfront [en] - транслятор, що переробляє вихідний код C з класами в вихідний код простого C. Це дозволило працювати над новою мовою і використовувати його на практиці, застосовуючи вже наявну в UNIX інфраструктуру для розробки на C. Нова мова, несподівано для автора, набув великої популярності серед колег і незабаром Страуструп вже не міг особисто підтримувати його, відповідаючи на тисячі запитань.

До 1983 року в мову були додані нові можливості, такі як віртуальні функції, перевантаження функцій і операторів, посилання, константи, призначений для користувача контроль над управлінням вільною пам'яттю, поліпшена перевірка типів і новий стиль коментарів (//). Одержаний мову вже перестав бути просто доповненої версією класичного C і був перейменований з C з класами в «C ++». Його перший комерційний випуск відбувся в жовтні 1985 року.

До початку офіційної стандартизації мова розвивалася в основному силами Страуструпа у відповідь на запити співтовариства програміста. Функцію стандартних описів мови виконували написані Страуструпом друковані роботи по C ++ (опис мови, довідкове керівництво і так далі). Лише в 1998 році був

ратифікований міжнародний стандарт мови C ++: ISO / IEC 14882: 1998 «Standard for the C ++ Programming Language»; після прийняття технічних виправлень до стандарту в 2003 році - наступна версія цього стандарту - ISO / IEC 14882: 2003.[12]

З одного боку, C ++ є нащадком Симула, яку Алан Кей визначив [13] як «Алгол з класами», і тому буде актуальною оцінка C ++ в порівнянні з іншими мовами з сімейства нащадків Алгола (Pascal, Java, C #, Visual Basic, Delphi, D, Oberon і ін.). З іншого боку, C ++ претендує на мультіпарадігменность і універсальну застосовність (на відміну від Сі, орієнтованого на дуже вузьке коло завдань), і використовується в промисловості набагато ширше інших нащадків Алгола, і тому буде актуальною оцінка C ++ в порівнянні з усім різноманіттям вживаних мов, включаючи і Сі. Щоб уникнути повторень, оцінки зазвичай поєднуються.

C ++ - мову, що складається еволюційно. На відміну від мов з формальним визначенням семантики, кожен елемент C ++ запозичувався з інших мов окремо і незалежно від інших елементів (ніщо з запропонованого C ++ за всю історію його розвитку не було нововведенням в Computer Science), що зробило мову надзвичайно складним, з безліччю дублюються і взаємно суперечливих елементів, блоки яких засновані на різних формальних базах. В цьому відношенні C ++ повторює шлях PL / 1, але, на відміну від останнього, тривалий повсюдне використання C ++ забезпечив вибір мови Сі в якості відправної точки.

Переваги. C ++ містить засоби розробки програм контрольованої ефективності для широкого спектра задач, від низькорівневих утиліт і драйверів до вельми складних програмних комплексів. Зокрема:

- Висока сумісність з мовою Сі: код на Сі може бути з мінімальними переробками скомпільовано компілятором C ++. Внешнеязиковой інтерфейс є прозорим, так що бібліотеки на Сі можуть викликатися з C ++ без додаткових витрат, і більш того - при певних обмеженнях код на C ++ може експортуватися зовні не отличимо від коду на Сі (конструкція extern "C").

- Як наслідок попереднього пункту - обчислювальна продуктивність. Мова спроектований так, щоб дати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми. Один з базових принципів C ++ - "не платиш за те, що не використовуєш» (див. Філософія C ++) - тобто жодна з мовних можливостей, що призводить до додаткових накладних витрат, не є обов'язковою для використання. Є можливість роботи з пам'яттю на низькому рівні.
- Підтримка різних стилів програмування: традиційне імперативне програмування (структурний, об'єктно-орієнтоване), узагальнене програмування, функціональне програмування, що породжує метапрограмування.
- Автоматичний виклик деструкторів об'єктів в адекватному порядку (зворотному виклику конструкторів) спрощує і підвищує надійність управління пам'яттю і іншими ресурсами (відкритими файлами, мережевими з'єднаннями, сполуками з базами даних і т. П.).
- Перевантаження операторів дозволяє коротко і ємке записувати вирази над користувацькими типами у природному алгебраїчній формі.
- Є можливість управління константністю об'єктів (модифікатори const, mutable, volatile). Використання константних об'єктів підвищує надійність і служить підказкою для оптимізації. Перевантаження функцій-членів за ознакою константності дозволяє визначати вибір методу в залежності мету виклику (константний для читання, неконстантний для зміни). Оголошення mutable дозволяє зберігати логічну константность побачивши ззовні коду, що використовує кеші і ледачі обчислення.
- Шаблони C ++ дають можливість побудови узагальнених контейнерів і алгоритмів для різних типів даних. Попутно шаблони дають можливість виробляти обчислення на етапі компіляції.
- Можливість вбудовування предметно-орієнтованих мов програмування в основний код. Такий підхід використовує, наприклад бібліотека Boost.Spirit, що дозволяє задавати EBNF-граматику парсеров прямо в коді

C ++. Boost.Spirit реалізує рекурсивно-спадний алгоритм, що накладає відповідні обмеження (такі як неприпустимість лівої рекурсії).

- Доступність. Для C ++ існує величезна кількість навчальної літератури, перекладеної на всілякі мови. Мова має високий поріг входження, але серед всіх мов такого роду має найбільш широкі можливості.

Недоліки

До числа недоліків можна віднести:

- Відсутність системи модулів. C ++ використовує заголовки, які сповнені недоліків:
  - змушує двічі писати одну і ту ж функцію (визначення в файлі з вихідним кодом і оголошення в заголовки).
  - Збільшує час компіляції. Можна оптимізувати, використовуючи предкомпільовані заголовки.
- Складний синтаксис і складна специфікація мови.

Кроссплатформеність. C ++ заявляється як багатоплатформовий: стандарт мови накладає мінімальні вимоги на ЕОМ для запуску двійкові програми. На практиці для написання портування коду на C ++ потрібна величезна майстерність і досвід, і «недбалі» коди на C ++ з високою ймовірністю можуть виявитися непортованими [14]. Тонке володіння C ++ в принципі може зробити код на C ++ настільки ж портіруемость, що і код на Сі (хоча, на думку Лінуса Торвальдса, C ++ при цьому фактично скоротиться до свого підмножини Сі [15]). Однак, критики C ++ стверджують, що вивчення і використання одночасно всіх мов, що протиставляються C ++ (що не викликають серйозних проблем при портуванні), в сумі вимагає приблизно тих же інтелекту, зусиль і тимчасових витрат, що і вивчення і використання одного тільки C ++ на висококласному рівні - в зв'язку з чим стає актуальною також оцінка порогу входження і результативності (продуктивності і якості праці програмістів).

РОЗДІЛ IV. АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для розробки програмного забезпечення інформаційної системи було обрано об'єктно-орієнтоване програмування як основну парадигму програмування.

Об'єктно-орієнтоване програмування — це метод програмування, заснований на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування [16]. Програмісти спочатку пишуть клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів.

На думку Алана Кея, розробника мови Smalltalk, якого вважають одним з «батьків-засновників» ООП, об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:

- Все є об'єктами.
- Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, при якій один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення — це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії.
- Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів.
- Кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів.

У класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії. Класи організовані у єдину деревоподібну структуру з загальним корінням, яка

називається ієрархією успадкування. Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Таким чином, програма являє собою набір об'єктів, що мають стан та поведінку. Об'єкти взаємодіють використовуючи повідомлення. Будується ієрархія об'єктів: програма в цілому — це об'єкт, для виконання своїх функцій вона звертається до об'єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об'єктів програми. Звісно, щоб уникнути безкінечної рекурсії у зверненнях, на якомусь етапі об'єкт трансформує запит у повідомлення до стандартних системних об'єктів, що даються мовою та середовищем програмування. Стійкість та керованість системи забезпечуються за рахунок чіткого розподілення відповідальності об'єктів (за кожну дію відповідає певний об'єкт), однозначного означення інтерфейсів міжоб'єктної взаємодії та повної ізольованості внутрішньої структури об'єкта від зовнішнього середовища (інкапсуляції).

В результаті дослідження Дебори Дж. Армстронг [17] комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень об'єктно-орієнтованого програмування. До них належить:

- Клас. Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості). Наприклад, клас Собака може характеризуватись рисами, притаманними всім собакам, зокрема: порода, колір хутра, здатність гавкати. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Як правило, клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

- Об'єкт. Окремий екземпляр класу (створюється після запуску програми і ініціалізації полів класу).
- Метод. Можливості об'єкта. В межах програми, використання методу має впливати лише на один об'єкт;
- Обмін повідомленнями. «Передача даних від одного процесу іншому, або надсилання викликів методів» [18].
- Успадкування (наслідування). Клас може мати «підкласи», спеціалізовані, розширені версії надкласу. Можуть навіть утворюватися цілі дерева успадкування. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть вводити свої власні. Успадкування може бути одиничне (один безпосередній батьківський клас) та множинне (кілька батьківських класів). Це залежить від вибору програміста, який реалізовує клас та мови програмування. Так, наприклад, в Java дозволене лише одинарне успадкування, а в C++ і те і інше.
- Приховування інформації (інкапсуляція). Приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення. Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт представляє кожному іншому класу певний інтерфейс — члени, доступні іншим класам. Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дозволить полегшити внесення змін, тобто, без потреби змінювати і користувачів інтерфейсу. Часто, члени класу позначаються як публічні (англ. public), захищені (англ. protected) та приватні (англ. private), визначаючи, чи доступні вони всім класам, підкласам, або лише до класу в якому їх визначено. Деякі мови програмування йдуть ще далі: Java використовує ключове слово private для обмеження доступу, що буде дозволений лише з методів того самого класу, protected — лише з методів того самого класу і його нащадків та з класів із того ж самого пакету, C# та VB.NET відкривають деякі члени лише для

класів із тієї ж збірки шляхом використання ключового слова internal (C#) або Friend (VB.NET), а Eiffel дозволяє вказувати які класи мають доступ до будь-яких членів;

- Абстрагування. Спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми.
- Поліморфізм. Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. На практиці - це реалізовується шляхом реалізації ряду підпрограм (функцій, процедур, методи тощо) з однаковими іменами, але з різними параметрами. В залежності від того, що передається і вибирається відповідна підпрограма.

Концепції. Поява в ООП окремого поняття класу закономірно випливає з бажання мати безліч об'єктів з подібним поведінкою. Клас в ООП - це в чистому вигляді абстрактний тип даних, який створюється програмістом. З цієї точки зору об'єкти є значеннями даного абстрактного типу, а визначення класу задає внутрішню структуру значень і набір операцій, які над цими значеннями можуть бути виконані. Бажаність ієрархії класів (а значить, успадкування) випливає з вимог до повторного використання коду - якщо кілька класів мають подібну поведінку, немає сенсу дублювати їх опис, краще виділити загальну частину в загальний батьківський клас, а в описі самих цих класів залишити тільки що розрізняються елементи.

Необхідність спільного використання об'єктів різних класів, здатних обробляти однотипні повідомлення, вимагає підтримки поліморфізму - можливості записувати різні об'єкти в змінні одного і того ж типу. В таких умовах об'єкт, відправляючи повідомлення, може не знати в точності, до якого класу належить адресат, і одні і ті ж повідомлення, відправлені змінним одного типу, що містить об'єкти різних класів, викличуть різну реакцію.

Окремої пояснення вимагає поняття обміну повідомленнями. Спочатку (наприклад, в тому ж Smalltalk) взаємодія об'єктів уявлялося як «справжній»

обмін повідомленнями, тобто пересилання від одного об'єкта іншому спеціального об'єкта-повідомлення. Така модель є надзвичайно загальною. Вона прекрасно підходить, наприклад, для опису паралельних обчислень за допомогою активних об'єктів, кожен з яких має власний потік виконання і працює одночасно з іншими. Такі об'єкти можуть вести себе як окремі, абсолютно автономні обчислювальні одиниці. Посилка повідомлень природним чином вирішує питання обробки повідомлень об'єктами, присвоєними поліморфним змінним - незалежно від того, як оголошується змінна, повідомлення обробляє код класу, до якого належить присвоєний змінної об'єкт. Даний підхід реалізований в мовах програмування Smalltalk, Ruby, Objective-C, Python.

Однак спільність механізму обміну повідомленнями має й іншу сторону - «повноцінна» передача повідомлень вимагає додаткових накладних витрат, що не завжди прийнятно. Тому в багатьох сучасних об'єктно-орієнтованих мовах програмування використовується концепція «відправка повідомлення як виклик методу» - об'єкти мають доступні ззовні методи, викликами яких і забезпечується взаємодія об'єктів. Даний підхід реалізований у величезній кількості мов програмування, в тому числі C ++, Object Pascal, Java, Oberon-2. Однак, це призводить до того, що повідомлення вже не є самостійними об'єктами, і, як наслідок, не мають атрибутів, що звужує можливості програмування. Деякі мови використовують гібридне представлення, демонструючи переваги одночасно обох підходів - наприклад, CLOS, Python.

Концепція віртуальних методів, підтримувана цими та іншими сучасними мовами, з'явилася як засіб забезпечити виконання потрібних методів при використанні поліморфних змінних, тобто, по суті, як спроба розширити можливості виклику методів для реалізації частини функціональності, забезпечується механізмом обробки повідомлень.

Вся логіка програми знаходиться у файлі Vario.ino де відбувається підключення бібліотек:

```
#include <Wire.h>                    //i2c бібліотека
#include <Adafruit_BMP280.h> //bmp280 бібліотека бародатчика
#include <Tone.h>                    //tone бібліотека динаміка
#include <LiquidCrystal.h>         //бібліотека для роботи з LCD
#include <SoftwareSerial.h>
```

Ініціалізація змінних портів

```
short speaker_pin1 = 8;
short speaker_pin2 = 9;
short bluetoothTX = 7;
short bluetoothRX = 6;
```

Створення об’єктів для роботи з бібліотеками

```
Adafruit_BMP280 bmp085;
SoftwareSerial btSerial = SoftwareSerial(bluetoothRX, bluetoothTX);
Tone     tone_out1;
Tone     tone_out2;
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);
```

Налаштування Arduino при запуску

```
void setup()
{
        Wire.begin();
        pinMode(4, INPUT);
        digitalWrite(4, HIGH);
        pinMode(13, OUTPUT);
        digitalWrite(13, LOW);
        lcd.begin(16, 2);
        bounseInput4O =  digitalRead(4);
        tone_out1.begin(speaker_pin1);
```

```
        tone_out2.begin(speaker_pin2);
        Serial.begin(9600);
        if (!bmp085.begin(0x76)) {
        Serial.println("Could not find a valid BMP085 sensor, check wiring!");
  }
  pinMode(bluetoothRX, INPUT);
  pinMode(bluetoothTX, OUTPUT);
  btSerial.begin(9600);
  lcd.begin(16, 2);
  Pressure = bmp085.readPressure();
 Alt0 = (float)44330 * (1 - pow(((float)Pressure/p0), 0.190295));
}
```

Розрахунок вертикальної швидкості

```
float tempo=millis();
    float vario=0;
    float N1=0;
    float N2=0;
    float N3=0;
    float D1=0;
    float D2=0;
    Pressure = bmp085.readPressure();
    Altitude = (float)44330 * (1 - pow(((float)Pressure/p0), 0.190295));
    for(int cc=1;cc<=maxsamples;cc++){
       alt[(cc-1)]=alt[cc];
       tim[(cc-1)]=tim[cc];
    };
    alt[maxsamples]=Altitude;
    tim[maxsamples]=tempo;
```

```
float stime=tim[maxsamples-samples];
for(int cc=(maxsamples-samples);cc<maxsamples;cc++){
   N1+=(tim[cc]-stime)*alt[cc];
   N2+=(tim[cc]-stime);
   N3+=(alt[cc]);
   D1+=(tim[cc]-stime)*(tim[cc]-stime);
   D2+=(tim[cc]-stime);
};
vario=1000*((samples*N1)-N2*N3)/(samples*D1-D2*D2);
```

РОЗДІЛ V. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Зібравши пристрій та запрограмувавши його отримуємо такі результати (рис.51-5.3).

Рисунок 5.1 - Отримання тиску для розрахунку висоти

Рисунок 5.2 - Висота та вертикальна швидкість при увімкненні

Рисунок 5.3 - Показники на висоті 1м

Польотні випробування на планері в повному обсязі підтвердили функціональність приладу при вимірювання висоти та розрахунку вертикальної швидкості. Показники відповідають значенням параметрів, які вимірювалися за допомогою штатних ароїдне-мембранних приладів планера L-13 «Бланік». Також розрахункова вертикальна швідкість, яка передавалась за протоколом Блютуз в смартфон покращила ефективність ПЗ при ноборі висоти в термічному потоки (рис. 5.4).

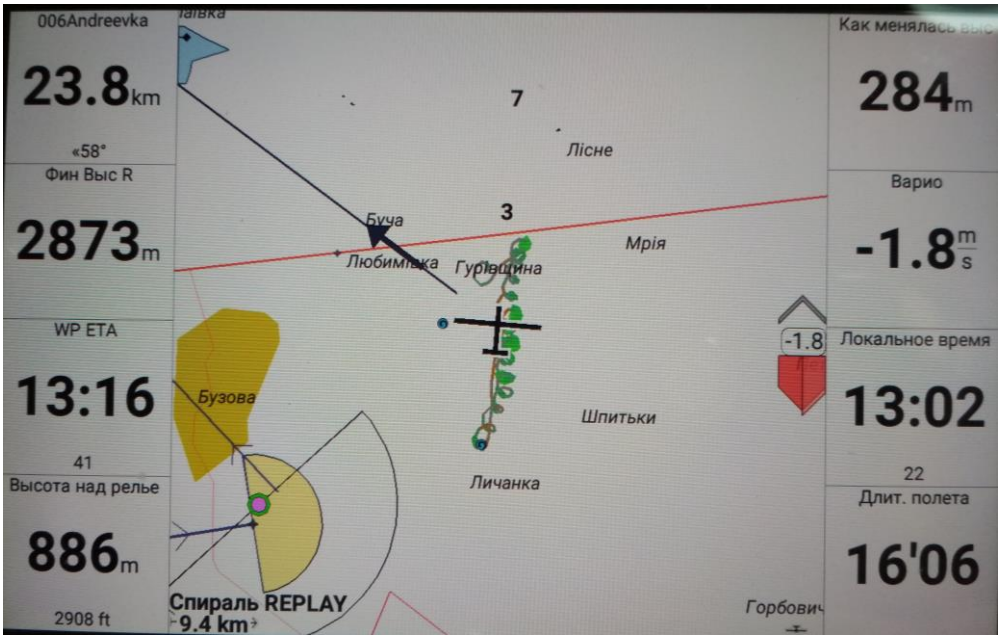

Рисунок 5.4 – Результат впливу з розробленого приладу

ВИСНОВКИ

1. Завдання проекту – розробка інформаційної системи для вимірювання висоти та розрахунку вертикальної швидкості та передача інформації у програмне забезпечення XCSOAR за допомогою технології Bluetoth та протоколу OpenVari виконане в повному обсязі. Дослідницький зразок варіометра виконує всі функції.
2. За час льотних випробувань на планері L-13 «Бланік» при виконанні учбових та спортивних польотів виявлене неступне:
   - інформація про вертикальну швидкість дійсне передоється з варіометра в смартфон з ПЗ XCSOAR та покращує якість формування як кількісних, так і якісних показників термічного потоку;
   - показники висоти на вертикальної швидкості відповідають значенням параметрів, які вимірювалися за допомогою штатних ароїдне-мембранних приладів планера L-13 «Бланік»;
   - звукова інформація про швидкість дозволяє вивільнити увагу пілота з приладної дошці на пілотування по принципу «капот-горизонт».
3. Подальшим шляхом покращення проекту є:
   - фільтрація вихідного сигналу на LCD за алгоритмом фільтра Калмана для зменшення швидкості зміни інформації на цифровому дисплеї;
   - створення схеми живлення на акумуляторі;
   - забезпечення вимкнення індикатора.
4. Вартість розробки на порядок менше аналогів існуючих планерних польотних комп’ютерів. На закупівлю комплектуючих затрачено близько 400 грн., а середньо вартістю аналога – 1000 евро.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Duffy Marsan, Carolyn. “IAB Releases Guidelines for Internet-of-Things Developers.” IETF Journal 11.1 (2015): 6-8. Internet Engineering Task Force.

2. Duffy Marsan, Carolyn. “IAB Releases Guidelines for Internet-of-Things Developers.” IETF Journal 11.1 (2015): 6-8. Internet Engineering Task Force.

3. Duffy Marsan, Carolyn. “IAB Releases Guidelines for Internet-of-Things Developers.” IETF Journal 11.1 (2015): 6-8. Internet Engineering Task Force.

4. Tschofenig, H. , et. al. — P. 6.

5. Tschofenig, H. , et. al. — P. 9.

6. Duffy Marsan, Carolyn. — P. 7.

7. Soltanian A., Van Dyck R.E. Performance of the Bluetooth system in fading dispersive channelsand interference // IEEE Global Telecommunications Conference, 2001 (GLOBECOM '01). — P. 3499—3503.

8. BLUETOOTH SIG Introduces BLUETOOTH Low Energy Wireless Technology, the Next Generation BLUETOOTH Wireless Technology.

9. Бителева А. Технологии мультимедийного доступа. Журнал «Теле-Спутник» 8(82) (август 2002).

10. Вишневский и др. Широкополосные беспроводные сети передачи данных. — М.: Техносфера, 2005. — 592 с. — ISBN 5-94836-049-0.

11. Страуструп, 1999, 1.4. Исторические замечания, — 46 с.

12. C++ — Standards.

13. Alan Kay's Definition Of Object Oriented Programming.

14. Андрей Карпов 20 ловушек переноса C++ - кода на 64-битную платформу. — RSDN Magazine #1-2007, 25.04.2007.

15. Открытая переписка gmane.comp.version-control.git от 06.09.2007.

16. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс, 2008. — 720 с.

17. Armstrong, «The Quarks of Object-Oriented Development.» In descending order of popularity, the «quarks» are: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, Abstraction.
18. Armstrong, «The Quarks of Object-Oriented Development.» In descending order of popularity, the «quarks» are: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, Abstraction.

АНОТАЦІЯ

Об’єкт дослідження: програмне забезпечення XCSOAR; технологія Bluetoth; протокол. OpenVario.

Мета роботи (проекту): розробка цифрового варіометру для вимірювання висоти та розрахунку вертикальної швидкості, а також передачі цієї інформації в програмне забезпечення XCSOAR за допомогою технологій Arduino, Bluetoth та протоколу OpenVario.

Методика дослідження: комплексне використання апаратних та програмних можливостей технологій Arduino, Bluetoth; протоколу OpenVario та ПЗ XCSOAR.

В роботі запропоноване та практичне реалізоване Завдання розробки апаратної та програмної частини інформаційної системи для вимірювання висоти та розрахунку вертикальної швидкості та передачі інформації у програмне забезпечення XCSOAR (аналог польотного комп’ютера планера на смартфоні) за допомогою технології Bluetoth та протоколу OpenVari виконане в повному обсязі. За час льотних випробувань на планері L-13 «Бланік» при виконанні учбових та спортивних польотів дослідницький зразок виконує всі функції.