УДК 004.93’1 HTTPS://DOI.ORG/10.15407/PP2022.02.003

С. О. Безпалько, В. М. Шимкович, А. Ю. Дорошенко

МОДЕЛЬ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНЕРЦІЙНОГО ВИМІРЮВАЛЬНОГО ПРИСТРОЮ

Проблеми автоматичного регулювання в сучасних технічних системах вимагають вирішення таких задач, як дотримання високої точності регулювання і здатності працювати в реальному часі. В даній роботі ці задачі вирішуються на прикладі розробки моделі та програмного забезпечення модуля стабілізації кута нахилу площини з трьома ступенями свободи, а також програмного забезпечення для отримання даних з акселерометра-гіроскопа MPU-6050 з використанням протоколу I2C. Розроблено програмну реалізацію цифрового пропорційно-інтегрально-диференціального регулятора з алгоритмом автоматичного підбору коефіцієнтів. Побудовано модель гіроскопічного пристрою та проведено тестування створеного рішення.

Ключові слова: гіроскопічна система, модуль стабілізації, інерційний вимірювальний пристрій, регулятор, алгоритми автоматичного підбору коефіцієнтів.

Вступ

Сучасний етап розвитку людської цивілізації передбачає застосування різних механізмів, які потребують можливості точно регулювати кут нахилу своїх елементів у різних площинах. Наприклад, від танків нового покоління вимагається досконалі системи наведення гармат, а отже можливість стабілізації ствола. До даних систем постійно висуваються нові вимоги: такі як збільшення точності стабілізації та швидкості, поліпшення завадостійкості та ін.

Для виконання всіх цих вимог заводами-виробниками і розробниками постійно удосконалюються конструкції і робочі процеси в стабілізаторах озброєння. На даний момент створено нове покоління високоточних датчиків та обчислювальних комплексів, яке об’єднує результати науково-технічного прогресу і практичного досвіду.

Такі системи керування кутом нахилу необхідні не тільки у військовій сфері, а й мають безліч застосувань у комерції. Зокрема, вдало застосовуються для вирішення проблеми стабілізації камери або фотоапарату для фото- чи відеозйомки. Метод, який не вимагає будь-яких додаткових характеристик комплекту «корпус-об’єктив», полягає у стабілізації всього корпусу камери — на противагу використанню внутрішнього механізму стабілізації зображення. Цього досягають завдяки гіроскопу, приєднаного до корпусу камери. Це дозволяє зовнішньому гіроскопу стабілізувати камеру. Використовується в основному для фотографування із транспортних засобів під час руху в тому випадку, коли об’єктиви чи камери із іншим типом стабілізації зображення недоступні або виявляються неефективними.

Нині дуже великої популярності набули квадрокоптери, для яких контроль кута нахилу відносно землі є першочерговою задачею для якісного та контрольованого польоту. Також подібні технології можуть полегшувати щоденне життя людей із хворобою Паркінсона, а також застосовуватись у хірургії для полегшення роботи лікаря та уникнення механічних помилок.

Якісні електронні датчики-гіроскопи стали доступними в роздрібній торговельній мережі зовсім недавно. Електронні гіроскопи – це перетворювачі нахилу і кутової швидкості в електричний струм. Вони служать для контролю положення і кутових швидкостей із коротким часом відгуку. Відмінністю електронних гіроскопів від їхніх механічних побратимів є компактність і мала вага, відсутність зношування елементів, висока швидкість відгуку, низька напруга живлення та малий струм живлення. Діапазон робочих температур дозволяє здійснювати контроль па-

раметрів у широкому спектрі кліматичних умов і географічних поясів.

Під час автоматичного регулювання кута виникають наступні задачі: необхідно отримати високу точність регулювання (< 18 мрад), система має працювати в реальному часі, тобто встигати здійснювати обрахунки в детермінований проміжок часу, необхідний для нормальної роботи об'єкту(час реакції < 0.001 секунди). Наприклад (це особливо важливо), якщо вирішується питання стабілізації польоту квадрокоптера, коли занадто довгий час реакції призведе до падіння. Також необхідно, щоб перерегулюваня було не більше 15%, адже це може впливати на стабільність системи.

1. Апаратна платформа та модель з інерційним вимірювальним пристроєм

В якості інерційного вимірювального пристрою в даній роботі був використаний модуль акселерометра і гіроскопа MPU6050, що підключається до мікроконтролера через шину I2C. Збір даних із MPU6050 можливий двома способами: збір сирих даних прямо з гіроскопічного датчика та акселерометра або використання вбудованого процесора руху, що називається виробником Digital Motion Processor (DMP)[1]. Для розробки коду та тестування отриманих даних було створено схему на макетній платі, що містить Arduino Nano та MPU6050. Це необхідно для дослідження різних способів отримання даних, перевірки працеспроможності інерційної вимірювальної системи та проведення необхідних налаштувань. Завдяки підключенню до персонального комп'ютера через універсальну послідовну шину з'являється можливість будувати графіки залежності положення від часу і, таким чином, візуалізувати бачити різні проблеми роботи датчика.

Після налаштувань запускається головний цикл програми і починається процес зчитування даних. Оскільки гіроскоп надає дані у вигляді кутової швидкості в даний момент, а не власне значення кута, необхідно здійснити розрахунки. Рівняння кутів на момент часу буде мати наступний вигляд:

$$\begin{aligned} x_i &= |x_{i-1} + w_x \times \Delta T|, \\ y_i &= |y_{i-1} + w_y \times \Delta T|, \\ z_i &= |z_{i-1} + w_z \times \Delta T|, \end{aligned} \quad (1)$$

де x_i, y_i та z_i - 3 кута Ейлера в даний момент часу; x_{i-1}, y_{i-1} та z_{i-1} – кути в попередній проміжок часу, w_x, w_y та w_z – кутова швидкість у відповідних площинах, ΔT – час між двома замірами;

Отримання часу здійснюється за допомогою функції millis() та розраховується кількість секунд із моменту минулого обрахунку. Дані датчика гіроскопа розміщені за адресами 0x43, 0x44 для осі x, 0x45, 0x46 для y та 0x47, 0x48 для z (рисунок 3). 2 байти для кожної з осей.

Для отримання всіх даних у коді запрошуються 6 байт з адреси 0x43. Після цього вони записуються у відповідні змінні. Поділ на значення 131 необхідне для переведення даних у градуси/с. Це значення відрізняється в залежності від виставленого діапазону на гіроскопі. Стандартне значення діапазону - 250 град/с. Після цього вираховується стала похибка датчика з заміряних кутових швидкостей та обраховуються кути за формулою (1).

Щоб збільшити точність показників, до розрахунку можна додати кути, розраховані за допомогою акселерометра. З роботи [4] відомо, що:

$$x = \operatorname{atan}\left(\frac{b}{\sqrt{a^2 + c^2}}\right),$$
$$y = \operatorname{atan}\left(\frac{-a}{\sqrt{b^2 + c^2}}\right),$$

де x, y – кути в радіанах, a, b, c – рух у трьох координатах, заміряний акселерометром; за цими формулами розраховуються кути, що переводяться в градуси. Відтак також віднімається статична похибка. Статична похибка обраховується експериментально, поклавши датчик на рівну поверхню та зробивши декілька сотень розрахунків і порахувавши їх середнє значення.

Після обрахування кутів гіроскопу та акселерометра можна отримати кінцевий результат, застосувавши «підсумову-

ючий фільтр», який підсумовує дані в певній пропорції. В даному випадку беруться 96% даних із гіроскопа та 4% даних із акселерометра, далі отримані дані виводяться в послідовний порт для відображення їх на персональному комп'ютері.

Проте, цей метод збору інформації не є ідеальним. Статична похибка досі присутня, хоча і є дуже малою. Під час руху відбуваються певні стрибки, що унеможливлює отримання кута обертання навколо осі Z з акселерометра, бо вона вирівняна з вектором гравітаційної сили. Однак головна проблема в тому, що після руху датчика з'являється велика похибка, яку неможливо усунути. Це пов'язано з тим, що мікроконтролер не встигає обробити всі дані. Тому якщо після руху покласти датчик у те ж положення, в якому він знаходився, буде добре видно, що дані спокою будуть зовсім іншими.

Для усунення таких помилок застосовується процесор руху, вбудований в MPU6050 [5]. Отриманий пакет даних повертає положення сенсора у вигляді кватерніона, який одразу виводиться в послідовний порт для відображення даних на комп'ютері. Виведення положення через кватерніон є перевагою даної бібліотеки, адже проводити обрахунки з використанням кватерніонів набагато легше, ніж з кутами Ейлера. Кватерніони можна перевести в кути Ейлера за допомогою формули:

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \begin{bmatrix}\arctan\left(\frac{2(q_0q_1 + q_2q_3)}{1-2(q_1^2+q_2^2)}\right)\\ \arcsin\left(2(q_0q_2 - q_3q_1)\right)\\ \arctan\left(\frac{2(q_0q_3 + q_1q_2)}{1-2(q_2^2+q_3^2)}\right)\end{bmatrix},$$

де α, β, γ – кути Ейлера, q_i - елементи кватерніона [6].

Проте, позаяк функції arctan та arcsin в комп'ютерних мовах програмування імплементовані так, що працюють тільки в межах від $-\pi/2$ до $\pi/2$, це накладає обмеження на всі ступені свободи [17]. Аби мати можливість згенерувати кути для всіх можливих орієнтацій платформи необхідно використовувати функцію atan2 замість arctan.

$$q = \begin{bmatrix}\cos\left(\frac{\gamma}{2}\right)\\ 0\\ 0\\ \sin\left(\frac{\gamma}{2}\right)\end{bmatrix}\begin{bmatrix}\cos\left(\frac{\beta}{2}\right)\\ 0\\ \sin\left(\frac{b}{2}\right)\\ 0\end{bmatrix}\begin{bmatrix}\cos\left(\frac{\alpha}{2}\right)\\ \sin\left(\frac{\alpha}{2}\right)\\ 0\\ 0\end{bmatrix} =$$

$$= \begin{bmatrix}\cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right)\cos\left(\frac{\gamma}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right)\sin\left(\frac{\gamma}{2}\right)\\ \sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right)\cos\left(\frac{\gamma}{2}\right) - \cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right)\sin\left(\frac{\gamma}{2}\right)\\ \cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right)\cos\left(\frac{\gamma}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right)\sin\left(\frac{\gamma}{2}\right)\\ \cos\left(\frac{\alpha}{2}\right)\cos\left(\frac{\beta}{2}\right)\sin\left(\frac{\gamma}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\sin\left(\frac{\beta}{2}\right)\cos\left(\frac{\gamma}{2}\right)\end{bmatrix},$$

де α, β,γ – кути Ейлера, q - кватерніон;[6]

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \begin{bmatrix}\text{atan2}\left(2(q_0q_1 + q_2q_3), 1-2(q_1^2+q_2^2)\right)\\ \text{asin}\left(2(q_0q_2 - q_3q_1)\right)\\ \text{atan2}\left(2(q_0q_3 + q_1q_2), 1-2(q_2^2+q_3^2)\right)\end{bmatrix},$$

де α, β,γ – кути Ейлера, q_i – елементи кватерніона. В [5] міститься програмна реалізація даної формули в структурі MPU6050. В даному випадку для розрахунку нахилу в осі Y та в осі X використовується вектор гравітації, який можна отримати з акселерометра.

2. Цифровий ПІД регулятор

ПІД регулятор можна виразити математично через наступне рівняння:

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt},$$

де K_p – пропорційний коефіцієнт, K_i – інтегральний коефіцієнт, K_d – диференціальний коефіцієнт, e(t) – помилка; це рівняння можна розділити на 3 окремі частини.

$$pOut(t) = K_p e(t),$$

$$iOut(t) = K_i \int_0^t e(t')dt',$$

$$dOut(t) = K_d \frac{de(t)}{dt}.$$

Де pOut – формула пропорційного впливу, iOut – інтегрального та dOut – диференціального[8]. Після цього формулу можна переписати наступним чином:

$$u(t) = pOut(t) + iOut(t) + dOut(t).$$

Підібравши коефіцієнти регулятора, можна отримати майже ідеальний перехідний процес системи. Тому програмна

реалізація має мати наступні особливості: можливість легкої зміни коефіцієнтів, вихід регулятора має регулюватись у певних межах, а також має бути можливість задавання різної позиції.

Нижче на рис.1 показано оголошення класу, який реалізує логіку роботи пропорційно-інтегрально-диференціального регулятора.

Програмний інтерфейс даного класу складається з трьох функцій та одного конструктора. В конструкторі визначаються межі регулювання, setK відповідає за встановлення коефіцієнтів регулятора. SetPos використовується для визначення необхідного положення, calcReg – функція, що прораховує керуючий вплив, на вхід вона приймає значення положення в даний момент часу, а на виході повертає значення впливу.

```
double PID::calcReg(double inp_){
    //calculate time interval
    auto currTime = millis();
    double dT = (double)(preTime - currTime);
    preTime = currTime;

    //get error
    double error = pos - inp_;

    //calculate every part of PID
    double pOut = kp * error;
    double iOut = ki * error * dT;
    double dOut = kd * ((error - preErr) / dT);
```

Рисунок 1. Програмна реалізація цифрового ПІД регулятора

Конструктор класу задає стандартне значення коефіцієнтам та визначає межі виходу регулювання. Для правильної роботи також необхідно визначити час початку роботи. Для цього використовується функція millis, яка є частиною ядра Arduino. Щоб розрахувати керуючий вплив спочатку треба отримати попередні дані. Тому в calcReg, на рис. 1, спачатку здійснюється обрахунок часу, що минув. Для цього через функцію millis отримується поточний час і від нього віднімається preTime. Після обрахування dT в змінну preTime записується оновлене значення теперішнього часу. Для отримання значення помилки від заданого значення необхідного положення віднімається реальне значення, зняте з датчиків, яке ця функція отримує через змінну inp_. Після цієї підготовки починається розрахунок впливу для кожної частини.

3. Керування двигунами гіроскопічної системи

ПІД регулятор не матиме ніякої користі за відсутності можливості керування електромоторами, які безпосередньо впливають на зворотній зв'язок системи. Для керування обраними драйверами двигунів необхідно генерувати сигнал на виведених пінах мікроконтролера. Щоб мати можливість міняти момент обертання електромоторів необхідно міняти напругу на виході. Для вирішення цієї задачі використовується широтно-імпульсна модуляція (ШІМ) [9].

Змінюючи коефіцієнт заповнення можна отримати сигнал із різною середньою напругою. Це добре видно на рис. 2,

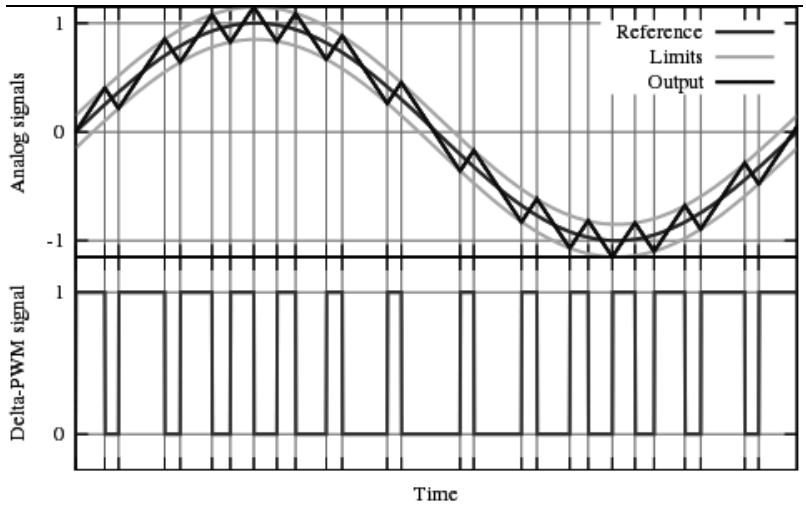

Рисунок 2. Моделювання синусоїдального сигналу в ШІМ[10]

де модулюється синусоїдальний сигнал. Щоб вирівняти вихідний сигнал можна застосувати фільтр низьких частот у вигляді додаткової ємності, проте, позаяк дана система дуже інертна, це не потрібно.

Середню напругу сигналу можна отримати за формулою:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)dt,$$

де $\bar{y}$ – середня напруга, $f(t)$ – функція ШІМ сигналу з періодом T. Оскільки $f(t)$ – це імпульсна хвиля, можна сказати, що її значення буде рівне y_{min}, в періоді, де час $0 < t < D \cdot T$ і y_{max}, коли $D \cdot T < t < T$ відповідно, де D – коефіцієнт заповнення. Тому дане рівняння можна переписати так:

$$\bar{y} = \frac{1}{T}\left(\int_0^{D\cdot T} y_{max}dt + \int_{D\cdot T}^{T} y_{min}dt\right) =$$

$$= \frac{1}{T}(D \cdot T \cdot y_{max} + T \cdot (1-D) \cdot y_{min}) =$$

$$= D \cdot y_{max} + (1-D) \cdot y_{min}.$$

У випадку, якщо y_{min} = 0, то дане рівняння можна спростити навіть більше:

$$\bar{y} = D \cdot y_{max}.$$

Мікроконтролер має можливість генерувати ШІМ сигнал своїх вихідних портах. Оскільки двигуни можуть бути підключені по-різному в залежності від фізичної реалізації, програма має мати можливість легко присвоювати піни для кожного електромотора. Головною метою цього модуля є можливість легко змінювати напругу на входах електродвигунів. Для реалізації цього функціоналу було створено 2 класи – GimbalMot та Motor.

Клас Motor відповідає за керування двигуном. У ньому зберігаються піни, що відведені на підключення цього мотору, напрям обертання та швидкість. Інтерфейс цього класу складається з функції та конструктора. В конструкторі ініціалізуються відведені піни, а функція setO відповідає за встановлення вихідного сигналу на пінах мікроконтролера.

Клас GimbalMot у свою чергу відповідає за керування трьома двигунами гіроскопічної системи. Для цього використовуються три об’єкти вище згаданого класу Motor. Для повороту в осі x, для обертання в осі y та для обертання в осі z. Для ініціалізації необхідно передати на вхід номера шести пінів, по 2 для кожного мотору. Є можливість керувати кожним мотором окремо через відповідні функції setVel[x/y/z].

Збільшення частоти ШІМ необхідне щоб прибрати шум з електромоторів, адже нижчі частоти може почути людське вухо. Після налаштування ШІМ починається налаштування переданих пінів як виходів. Потім із даних виходів створюються по три об’єкти класу Motor для кожної осі. Оскільки в даному випадку зберігається посилання на об’єкт, а не сам об’єкт, в деструкторі необхідно здійснити очищення пам’яті. Тому клас GimbalMot містить деструктор для звільнення пам’яті. Через це відбувається вивільнення пам’яті для кожного з моторів. Таким чином з′являється можливість налаштування потрібного двигуна без зміни роботи інших.

4. Фільтр вхідних даних

Оскільки датчики зазвичай не ідеальні, їхній вихідний сигнал буде під впливом різних шумів, тож необхідно мати фільтр для отримання точніших даних. Для розв′язання даної проблеми було прийнято рішення використати фільтр Калмана. Для того, щоби використовувати фільтр Калмана для оцінювання внутрішнього стану процесу, маючи лише послідовність зашумлених спостережень, необхідно змоделювати процес відповідно до моделі фільтру Калмана. Це означає задання наступних матриць: F_k - моделі переходу станів; H_k - моделі спостереження; Q_k - коваріації шуму процесу; R_k - коваріації шуму спостереження; та іноді B_k - моделі керування для кожного моменту часу, k[12].

$$x_k = F_k x_{k-1} + B_k u_k + w_k,$$

$$w_k \sim N(0, Q_k),$$

де F_k – модель переходу стану для попереднього стану x_{k-1}, B_k - модель впливів керування, що застосовується до вектора керування u_k, w_k – шум процесу, що має

нормальний розподіл з нульовим середнім значенням та коваріацією Q_k;

У момент вимірювання z_k справжнього стану x_k отримується за наступною формулою:

$$z_k = H_k x_k + v_k,$$
$$v_k \sim N(0, R_k),$$

де H_k є моделлю спостереження, що відображає простір справжнього стану у спостережуваний простір, і v_k є шумом спостереження, що, як вважається, є ґаусовим білим шумом з нульовим середнім значенням і з коваріацією R_k.

Фільтр Калмана є рекурсивним, а отже для роботи йому необхідний тільки минулий стан та поточні дані. Це надає перевагу над пакетними фільтрами, які для своєї роботи потребують збір та збереження масиву даних. Щоб зберегти стан фільтру використовуються дві змінні: $\hat{x}_{k|k}$ – оцінка стану в момент часу k при заданих спостереженнях до моменту часу k включно та $P_{k|k}$ – коваріаційна матриця помилок. Алгоритм фільтра Калмана можна розділити на дві частини – стадія передбачення та уточнення.

Стадія передбачення використовує оцінку стану з попереднього моменту часу для отримання оцінки стану на поточний момент часу.

$$\hat{x}_{k|k} = F_k \hat{x}_{k-1|k-1} + B_k u_k,$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k.$$

Розрахунки в цій стадії включають розрахунок передбаченої оцінки стану та коваріацію передбаченої оцінки.

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1},$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k,$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1},$$

$$\hat{x}_{k|k} = \hat{x}_{k-1|k-1} + K_k \tilde{y}_k,$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}.$$

У стадії уточнення необхідно розрахувати наступні значення: відхилення вимірювання, коваріацію відхилення, оптимальний передавальний коефіцієнт Калмана, оновлену оцінку стану, коваріацію оновленої оцінки [12].

Розроблений модуль має наступні функції: getAngle – повертає відфільтроване значення кута (для цього йому необхідно передати значення кута зібране з датчика), кутову швидкість в цій площині та пройдений час в секундах. Функція setAngle відповідає за встановлення початкового кута нахилу, необхідного для майбутніх розрахунків.

Завдяки функції getRate можна також отримати відфільтроване значення кутової швидкості. Для тонкого калібрування фільру використовуються функції setQangle, setQbias та setRmeasure. Q_angle - дисперсія шуму процесу для акселерометра, Q_bias - дисперсія шуму процесу для зміщення гіроскопа, R_measure - дисперсія шуму вимірювання.

У конструкторі класу Kalman здійснюється початкова ініціалізація змінних. Виставляються початкові значення для вищезгаданих Q_angle, Q_bias та R_measure, які в подальшому можна корегувати для кращої роботи. Також заповнюється матриця коваріації помилок P[2][2]. Позаяк на даний момент відоме точне положення та швидкість у початковий момент необхідно заповнити її нульовим значенням.

Алгоритм отримання даних можна умовно поділити на сім кроків. Стадія передбачення включає перші 2 кроки (рис. 3). Перший крок передбачає виконання дискретних рівнянь оновлення часу фільтра Калмана, обраховується змінна rate та angle. Наступним кроком буде обрахування нових значень для матриці коваріації помилок P. Після цього починається стадія уточнення, в якій обчислення здійснюються за формулами, згаданими вище.

Наступною буде стадія уточнення. Крок номер 3 - це обрахунок відхилення вимірювання. Крок 4 включає обрахування коваріації відхилення. Далі у кроці 5 обраховується коефіцієнт Калмана, що на виході являє собою вектор 2 на 1. Після цього необхідно оновити оцінку стану, що на виході дає шуканий кут. Останній крок - це розрахунок коваріації оновленої оцінки, що знову оновлює значення в матриці P для наступного циклу. В кінці функція повертає отриманий кут.

```
float Kalman::getAngle(float newAngle, float newRate, float dt) {

    // Discrete Kalman filter time update equations - Time Update ("Predict")
    /* Step 1 */
    rate = newRate - bias;
    angle += dt * rate;

    // Update estimation error covariance - Project the error covariance ahead
    /* Step 2 */
    P[0][0] += dt * (dt*P[1][1] - P[0][1] - P[1][0] + Q_angle);
    P[0][1] -= dt * P[1][1];
    P[1][0] -= dt * P[1][1];
    P[1][1] += Q_bias * dt;
```

Рисунок 3. Фрагмент коду фільтра Калмана, крок 1-2[7]

```
// Discrete Kalman filter measurement update equations - Measurement Update
("Correct")
    // Calculate angle and bias - Update estimate with measurement zk (newAngle)
    /* Step 3 */
    float y = newAngle - angle; // Angle difference

    // Calculate Kalman gain - Compute the Kalman gain
    /* Step 4 */
    float S = P[0][0] + R_measure; // Estimate error
    /* Step 5 */
    float K[2]; // Kalman gain - This is a 2x1 vector
    K[0] = P[0][0] / S;
    K[1] = P[1][0] / S;
    /* Step 6 */
    angle += K[0] * y;
    bias += K[1] * y;

    // Calculate estimation error covariance - Update the error covariance
    /* Step 7 */
    float P00_temp = P[0][0];
    float P01_temp = P[0][1];

    P[0][0] -= K[0] * P00_temp;
    P[0][1] -= K[0] * P01_temp;
    P[1][0] -= K[1] * P00_temp;
    P[1][1] -= K[1] * P01_temp;

    return angle;
```

Рисунок 4. Фрагмент коду фільтра Калмана, крок 3-7[7]

Тестування розробленого фільтра було проведено на генераторі шуму. Як видно з графіку на рис. 4, фільтр Калмана чудово справляється з даним шумом (на графіку – unfiltered) та легко отримує значення (на графіку – filtered), подібне до реального (на графіку – real).

Однак, як бачимо на рис. 5, фільтр може додавати затримку сигналу, тому для отримання точних даних необхідно здійснювати точне калібрування і підбір значень змінних Q_angle, Q_bias та R_measure. Оскільки обрано інерційний вимірювальний пристрій, датчик MPU6050 під час роботи із вбудованим цифровим процесором руху (DMP) сам фільтрує вихідні дані.

5. Тестування

Розроблена керуюча програма мікроконтролера складається з чотирьох відокремлених частин:

модуль зв’язку з інерційним вимірювальним пристроєм;

модуль, що містить реалізацію ПІД регулятора;

модуль керування електродвигунами гіроскопічної системи;

модуль фільтрування вхідних даних.

Для проведення тестування розробленої системи було зібрано робочий макет. Для спрощення створення макету прийнято рішення зробити для тестування систему з одним ступенем свободи.

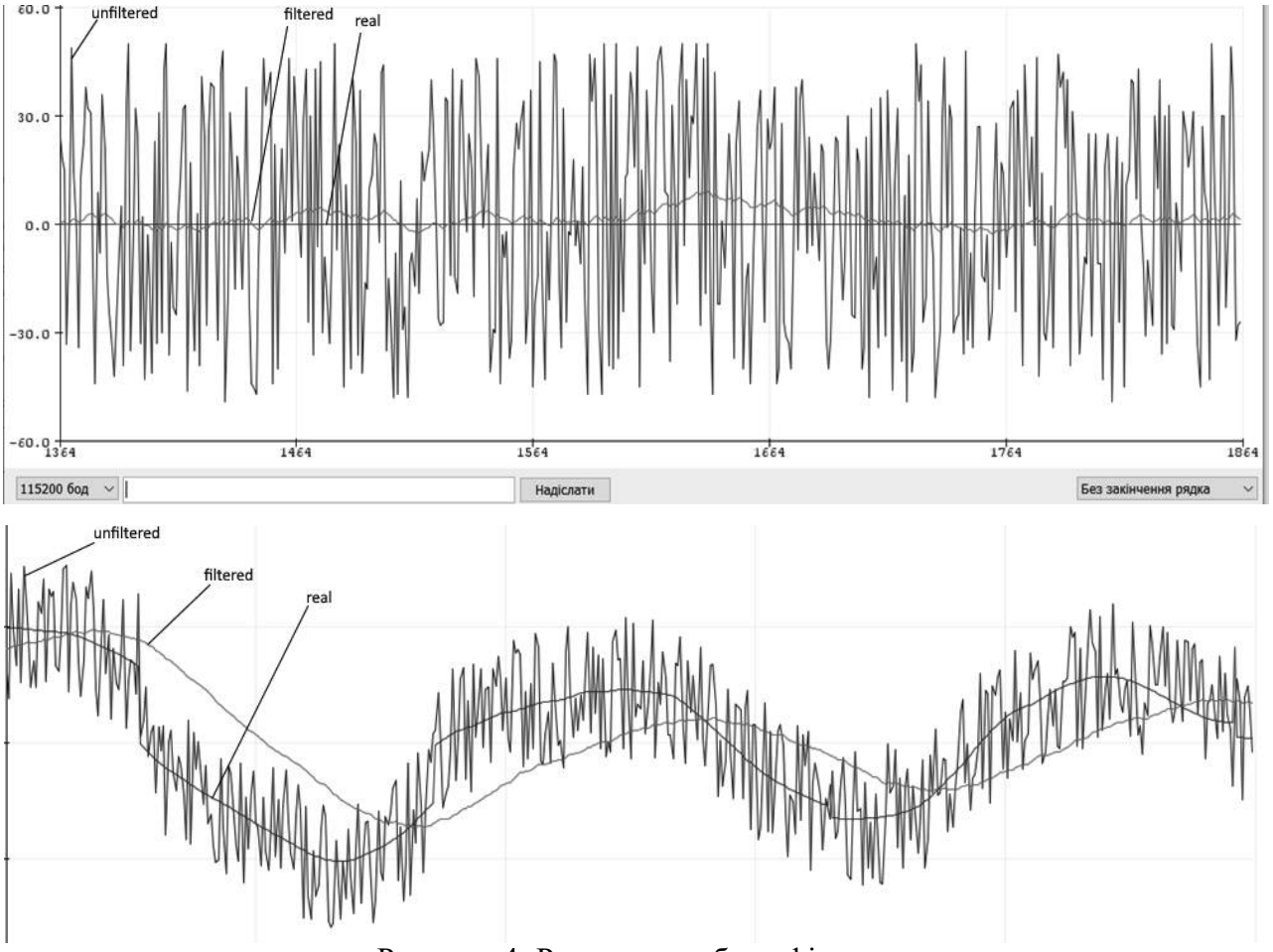

Рисунок 4. Результат роботи фільтру

Для обрахунку пропорційної, інтегральної та диференціальної частин регулятора було вирішено застосувати алгоритм автоматичного підбору. Для цієї задачі обрано релейний метод у парі з методом Зіглера-Нікольса[12]. Метод Зіглера-Нікольса передбачає використання таблиці для підбору коефіцієнтів.

Тип регулятора	K_p	T_i	T_d
Класичний ПІД	$0.6K_u$	$0.5T_u$	$0.125T_u$
Інтегральне правило Песена	$0.7K_u$	$0.4T_u$	$0.15T_u$
Невелике перерегулювання	$0.33K_u$	$0.5T_u$	$0.33T_u$
Без перерегулювання	$0.2K_u$	$0.4T_u$	$0.33T_u$

Підбір коефіцієнтів регулятора проходитиме за відносно простим алгоритмом, що включає поєднання двох методів. Спершу необхідно виставити максимальний вихід регулятора, та очікувати отримання заданого цільового положення, після чого вимкнути вихідний сигнал. Коли появиться похибка, ще раз налаштувати вихід контролера. Повторити ці кроки декілька разів. Далі необхідно обрахувати ультимативний множник, використовуючи значення амплітуди виходу регулятора, та обрахувати період коливань регулювання. Отримавши ці дані можна обрахувати коефіцієнти з використанням метода Зіглера-Нікольса.

Таким чином, було отримано наступний результат:

На даному графіку(рис. 5) зображено перехідний процес системи при наданні короткотривалого фізичного імпульсу на макет системи. З даного графіку можна визначити наступні характеристики перехідного процесу: час перехідного процесу - 0.44с, перерегулювання - 6.2%.

Результати тестування моделі задовільняють поставлені технічні вимоги до її роботи.

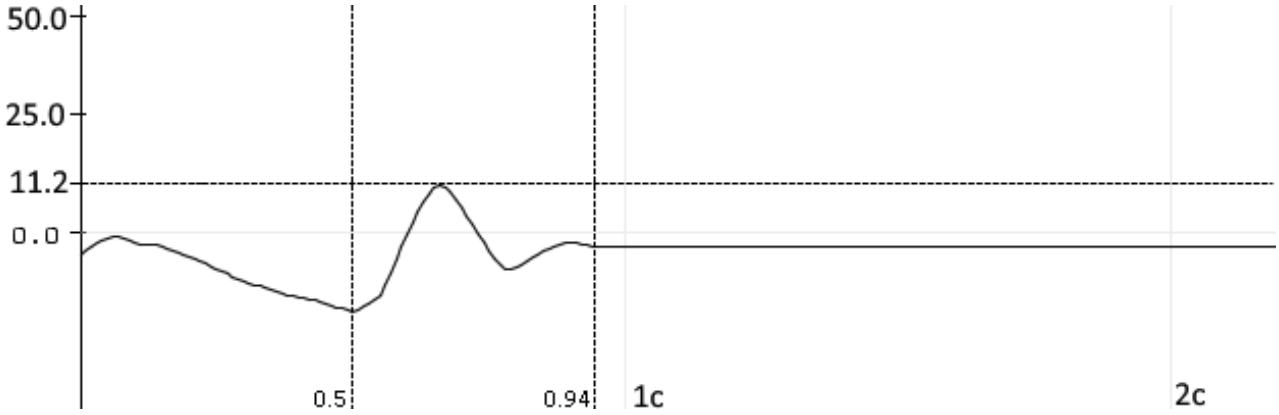

Рисунок 5. Перехідний процес при впливі на систему

Висновки

Розробка модулю стабілізації кута нахилу площини з трьома ступенями свободи була виконана із урахуванням новітніх досягнень у сфері збору даних про положення об'єкта в просторі. Даний модуль був розроблений так, щоб легко замінювати частин модуля та використовувати різні вузли, які відрізняються від описаних у роботі. Для того, щоб поміняти вузол, необхідно змінити модуль у програмі мікроконтролера, відповідальний за нього. За умови дотримання заданого інтерфейсу для зміни буде достаньо поміняти лише один файл. Дані характеристики уможливлюють гнучке застосування даного модуля в багатьох різних сферах.

Перевагами даного модуля є простота реалізації, легкість підключення, гнучкість програми мікроконтролера та можливість використання дешевих та поширених компонентів. Використання новітніх інерційних вимірювальних систем дозволяє легко отримувати точні дані про положення регульованої площини. Щоб мати можливість виконувати роботу навіть у випадку шуму показників був реалізований фільтр даних, що дозволяє працювати із більш давніми моделями датчиків. Для налаштування регулятора було запропоновано використання популярних методик підбору коефіцієнтів ПІД, що значно спрощує запуск системи, яка буде використовувати даний модуль.

У процесі роботи були розроблені структурна, принципова електрична схема, схема алгоритму програми мікроконтролера, обрані та описані основні вузли й елементи.

Для підтвердження дієвості даного модуля було створено розділ, присвячений розробці фізичної моделі. В ньому була сформована структура моделі та продемонстрована можливість стабілізації площини навіть у разі використання неідеальних компонентів.

References

1. *Nyberg, L., & Tjellander, M.* (2017). Camera stabilization (Dissertation). p. 29
2. *John Pardue* (2005) C Programming for Microcontrollers, Knoxville.: Smiley Micros. p. 300
3. MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2 – URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf (дата звернення: 05.04.2021)
4. *Welch, Greg & Bishop, Gary* (2006) An Introduction to the Kalman Filter. Proc. Siggraph Course. 8. pp. 1-16 jrowberg/i2cdevlib The I2C Device Library – URL: https://github.com/jrowberg/i2cdevlib (дата звернення: 21.04.2021)
5. *Hamilton, William Rowan* (2000) On quaternions, or on a new system of imaginaries in algebra. Philosophical Magazine. 1844. pp. 489-495
6. Pandanom/StabilizationModule Module for plane stabilization with three degrees of freedom – URL: https://github.com/Pandanom/StabilizationModule (дата звернення: 25.05.2021).
7. *Samotyy V., Telenyk S., Kravets P., Shymkovych V., Posvistak T.* (2018) A real time

control system for balancing a ball on a platform with FPGA parallel implementation. Technical Transactions. Vol. 5. pp. 109-118
8. *Yurkevich, Valery* (2009) PWM PI/PID/PIDF Control for Nonlinear Nonaffine Systems via Singular Perturbation. 2009 International Forum on Strategic Technologies. pp. 185-190.
9. Barr, Michael (2001) Pulse Width Modulation. Embedded Systems Programming. September. pp. 103-104
10. ESP8266 Arduino Core's documentation – URL: https://arduino-esp8266.readthedocs.io/en/latest/ (дата звернення: 07.05.2021)
11. *J. B. Ziegler and N. B. Nichols.* «Optimum settings for automatic controllers» ASME Transactions v64. 1942; pp. 759-768.

Отримано: 24.05.2022

***Про авторів*:**

Безпалько Станіслав Олегович, магістрант 1 року навчання Національного Технічного Університету України «КПІ імені Ігоря Сікорського». https://orcid.org/0000-0001-8550-1792
Шимкович Володимир Миколайович, кандидат технічних наук, доцент кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського». Кількість наукових публікацій в українських виданнях – понад 30. Кількість наукових публікацій в зарубіжних виданнях – понад 10. Індекс Хірша – 4. https://orcid.org/0000-0003-4014-2786
Дорошенко Анатолій Юхимович, доктор фізико-математичних наук, професор, завідувач відділу теорії комп'ютерних обчислень, професор кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського». Кількість наукових публікацій в українських виданнях – понад 190. Кількість наукових публікацій в зарубіжних виданнях – понад 80. Індекс Хірша – 6. http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», проспект Перемоги 37 та Інститут програмних систем НАН України, 03187, м. Київ-187, проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559
E-mail:
stas110922@gmail.com,
v.shymkovych@kpi.ua,
doroshenkoanatoliy2@gmail.com