Національний технічний університет України

“ КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ”

ЛАБОРАТОРНА РОБОТА

ПО КУРСУ: ” Електронні системи ”

ТЕМА: “ Дослідження ПІД-регулятора ”

Виконали :Студенти IV курсу

Групи ДЕ-91

Рєзнік Олександр Сергійович

КИЇВ 2022 Р.

Мета роботи: ознайомитися з процедурою налаштування параметрів ПІД-регулятора та набути основних навичок його налаштування.

Теоретичні відомості:

ПІД-регулятор

Пропорційно-інтегрально-диференціювальний (ПІД) регулятор – пристрій в керуючому контурі зі зворотним зв'язком. Використовується в системах автоматичного управління для формування сигналу керуючого з метою отримання необхідної точності і якості перехідного процесу. ПІД-регулятор формує керуючий сигнал, що є сумою трьох доданків, перший з яких пропорційно різниці вхідного сигналу і сигналу зворотного зв'язку (сигнал неузгодженості), друге – інтегралу сигналу неузгодженості, третє – похідної сигналу неузгодженості.

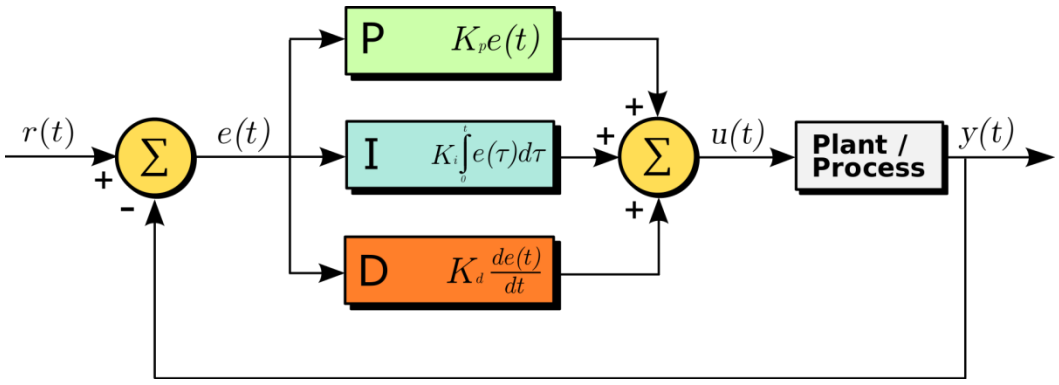

Рис. 1 Схематичне представлення ПІД-регулятора

Пропорційна складова

Пропорційна складова виробляє вихідний сигнал, протидіє відхиленню регульованої величини від заданого значення, що спостерігається в даний момент часу. Він тим більший, чим більше це відхилення. Якщо вхідний сигнал дорівнює заданому значенню, вихідний дорівнює нулю.

Однак за використання тільки пропорційного регулятора значення регульованої величини ніколи не стабілізується на заданому значенні. Існує так звана статична помилка, яка дорівнює такому відхилення регульованої величини, що забезпечує вихідний сигнал, що стабілізує вихідну величину саме на цьому значенні. Наприклад, регулятор температури вихідний сигнал (потужність нагрівача) поступово зменшується при наближенні температури

до заданої, і система стабілізується при потужності, що дорівнює тепловим втрат. Температура не може досягти заданого значення, тому що в цьому випадку потужність нагрівача дорівнюватиме нулю, і він почне остигати.

Чим більший коефіцієнт пропорційності між вхідним і вихідним сигналом (коефіцієнт посилення), тим менша статична помилка, проте при занадто великому коефіцієнті посилення за наявності затримок (запізнення) в системі можуть початися автоколивання, а при подальшому збільшенні коефіцієнта система може втратити стійкість.

Інтегруюча складова

Інтегруюча складова пропорційна інтегралу за часом відхилення регульованої величини. Її використовують для усунення статичної помилки. Вона дозволяє регулятору згодом врахувати статичну помилку.

Якщо система не відчуває зовнішніх збурень, то через деякий час регульована величина стабілізується на заданому значенні, сигнал пропорційної складової дорівнюватиме нулю, а вихідний сигнал повністю забезпечуватиметься інтегруючої складової. Тим не менш, інтегруюча складова також може призводити до автоколивань при неправильному виборі коефіцієнта.

Диференціююча складова

Диференціююча складова пропорційна темпу зміни відхилення регульованої величини та призначена для протидії відхиленням від цільового значення, які прогнозуються у майбутньому. Відхилення можуть бути викликані зовнішніми збуреннями або запізненням впливу регулятора на систему.

Вихідний сигнал регулятора визначається трьома складовими:

$$u(t) = P + I + D = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt},$$

де K_p, K_i, K_d – коефіцієнти посилення пропорційної, інтегруючої та диференціюючої складових регулятора відповідно.

Платформа Arduino

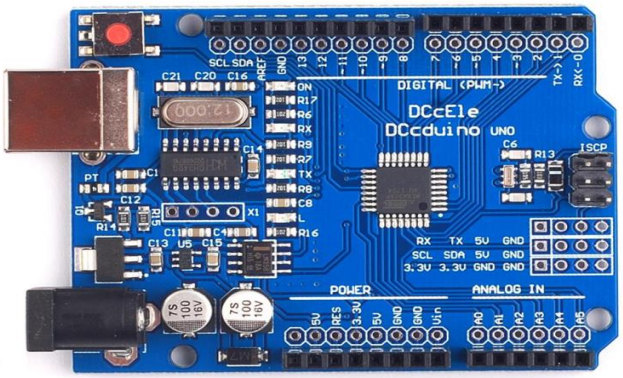

Рис. 2 Arduino UNO

Arduino – це відкрита програмована апаратна платформа для роботи з різними фізичними об'єктами і є простою платою з мікроконтролером, а також спеціальне середовище розробки для написання програмного забезпечення мікроконтролера.

Arduino може використовуватися для розробки інтерактивних систем, керованих різними датчиками та перемикачами. Такі системи, у свою чергу, можуть керувати роботою різних індикаторів, двигунів та інших пристроїв. Проекти Arduino можуть бути як самостійними, так і взаємодіяти з програмним забезпеченням, що працює на персональному комп'ютері (наприклад, програмами Flash, Processing, MaxMSP). Будь-яку плату Arduino можна зібрати вручну або купити готовий пристрій; середовище розробки для програмування такої плати має відкритий вихідний код і є повністю безкоштовним.

Ультразвуковий датчик відстані

Рис. 3 Сонар

Модуль HC-SR04 використовує акустичне випромінювання для визначення відстані до об'єкта. Цей безконтактний датчик забезпечує високу точність та стабільність вимірювань. Діапазон вимірювань становить: від 2 см до 400 см. На показання датчика практично не впливають сонячне випромінювання та електромагнітні шуми.

Драйвер двигуна

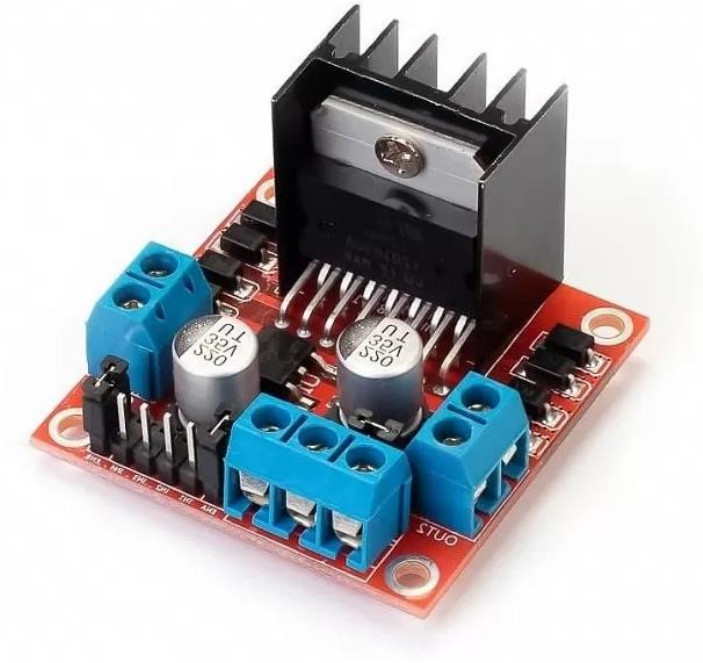

Рис. 4 Драйвер L298N

Драйвер L298N використовується радіоаматорами для багатофункціонального керування двигунами постійного струму. Схема модуля, що складається з двох Н-мостів, дозволяє підключати до нього один біполярний крокový двигун або одночасно два щіткові двигуни постійного струму. При цьому є можливість змінювати швидкість та напрямок обертання моторів. Управління здійснюється шляхом подачі відповідних сигналів на командні входи, виконані у вигляді контактів штирю.

Кожен Н-міст виконаний у вигляді збирання з чотирьох транзисторних ключів з включеним у центрі навантаженням у вигляді обмотки двигуна. Такий підхід дозволяє змінювати полярність в обмотці і як наслідок напрямок обертання двигуна шляхом чергування пар відкритих та закритих ключів.

Кроковий двигун

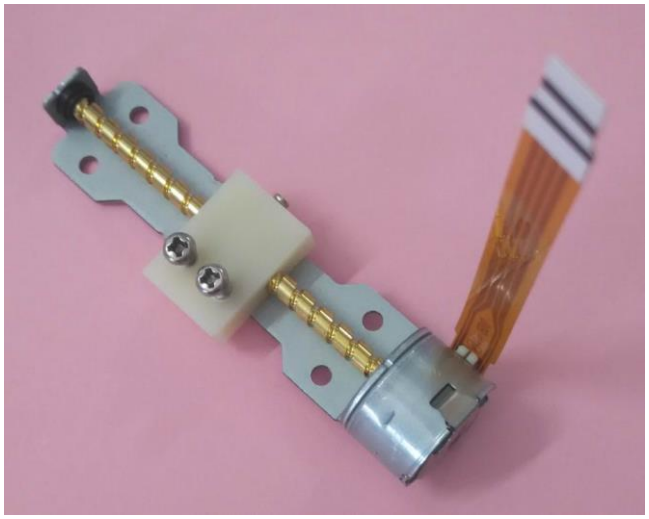

Рис. 5 Кроковий двигун на 4 pin

Кроковий двигун — електричний двигун, в якому імпульсне живлення електричним струмом призводить до того, що його ротор не обертається неперервно, а щоразу виконує обертальний рух на заданий кут. Завдяки цьому, кут повороту ротора залежить від числа поданих імпульсів струму, а кутова швидкість ротора точно дорівнює частоті імпульсів помножень на кут повороту ротора за один цикл роботи двигуна.

Кут повороту двигуна під впливом одного імпульсу може мати різні значення, залежні від конструкції двигуна, — як правило, це значення в діапазоні від декількох градусів до декількох десятків градусів. Крокові двигуни, залежно від призначення пристосовані до виконання від частки обертів на хвилину, до декількох тисяч обертів на хвилину.

Схема установки:

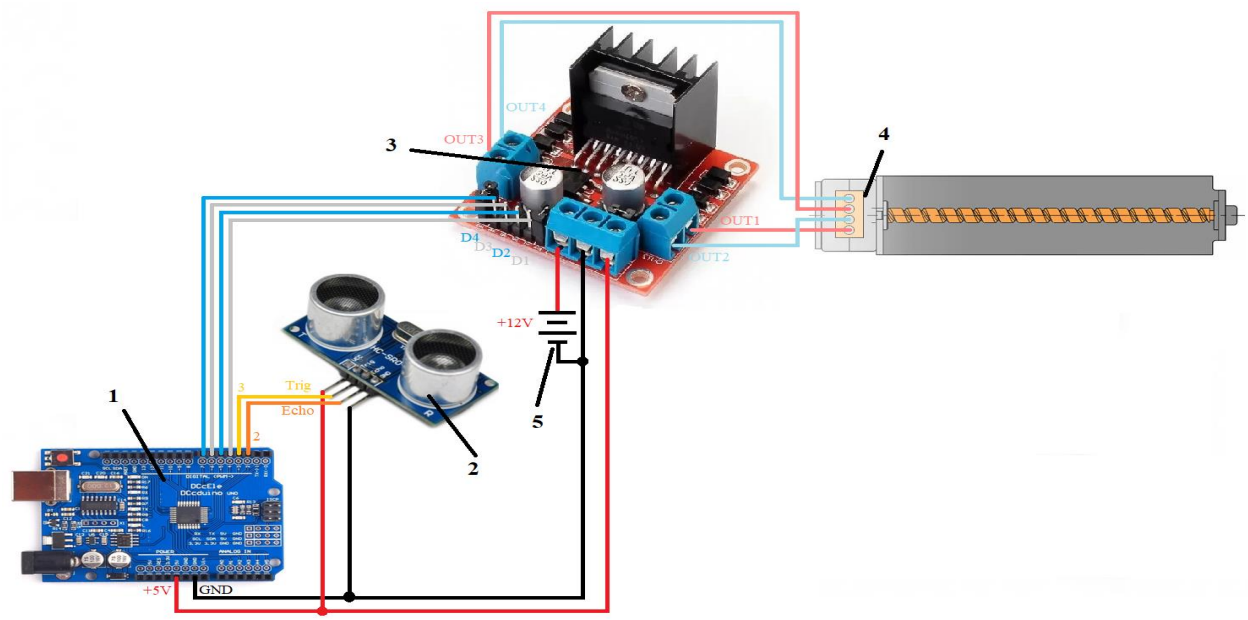

Рис. 6 Схематичне представлення робочої семи

1. Arduino UNO;
2. Ультразвуковий датчик відстані;
3. L298N (драйвер для крокового двигуна);
4. Крокковий двигун;
5. Джерело постійної напруги;

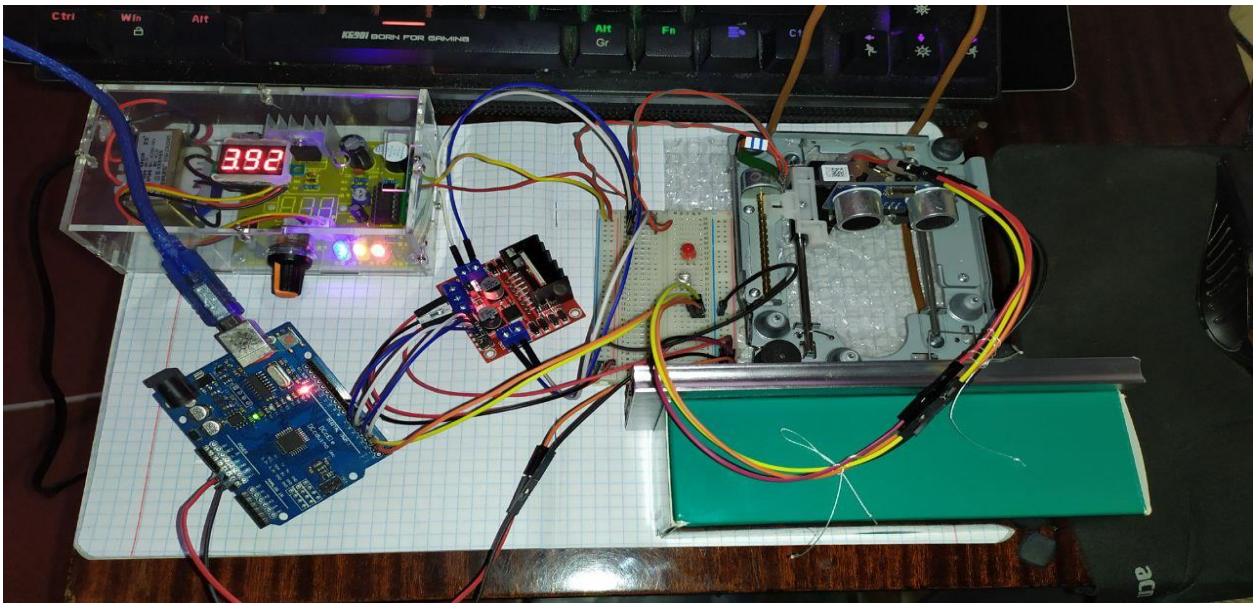

Рис. 7 Реально зібраний пристрій

Хід роботи

На початку роботи зібрали вище наведену схему, і написали код, для прошивки мікроконтролера Arduino, перевіривши виконання поставленої задачі. Задача роботи полягає в тому, щоб реалізувати ПІД-регулятор. В даній схемі воно працює так, ми маємо кроковий двигун за допомогою якого ми можемо переміщувати ультразвуковий датчик (сонар) на певну обмежену відстань від перешкоди. В результаті ми отримуємо функціонал, коли при постійному переміщенні на основі виміряних даних з сонару, ми можемо переміщувати сонар на певну задану відстань від перешкоди. То ж вирішуємо наступну задачу, в підібрані трьох коефіцієнтів ПІД-регулятора, щоб дана схема як найшвидше стабілізувалась, на певній заданій відстані від перешкоди.

Через не ідеальність сонару, додатково використовується алгоритм в поєднанні медіанного фільтра і рухомого середнього, це потрібно для того, щоб якнайкраще вимірювати відстань до перешкоди, щоб не отримувати непотрібні і не контрольовані похибки.

В графіках, які будуть наведені нижче, ми побачимо три сигнали, наведені трьома кольорами, синім буде позначено значення відстані до перешкоди на якій повинен зупинитись сонар, червоним – значення відстані отримане з сонару без обробки, зеленим – значення отримане з сонару, після обробки фільтрами.

Вісь Oy відповідає відстані до перешкоди, вісь Ox це не пряме значення часу, а відповідає кількості виміряних точок за певні інтервали часу, і кожна точка будувалась на графіку раз 0.5с.

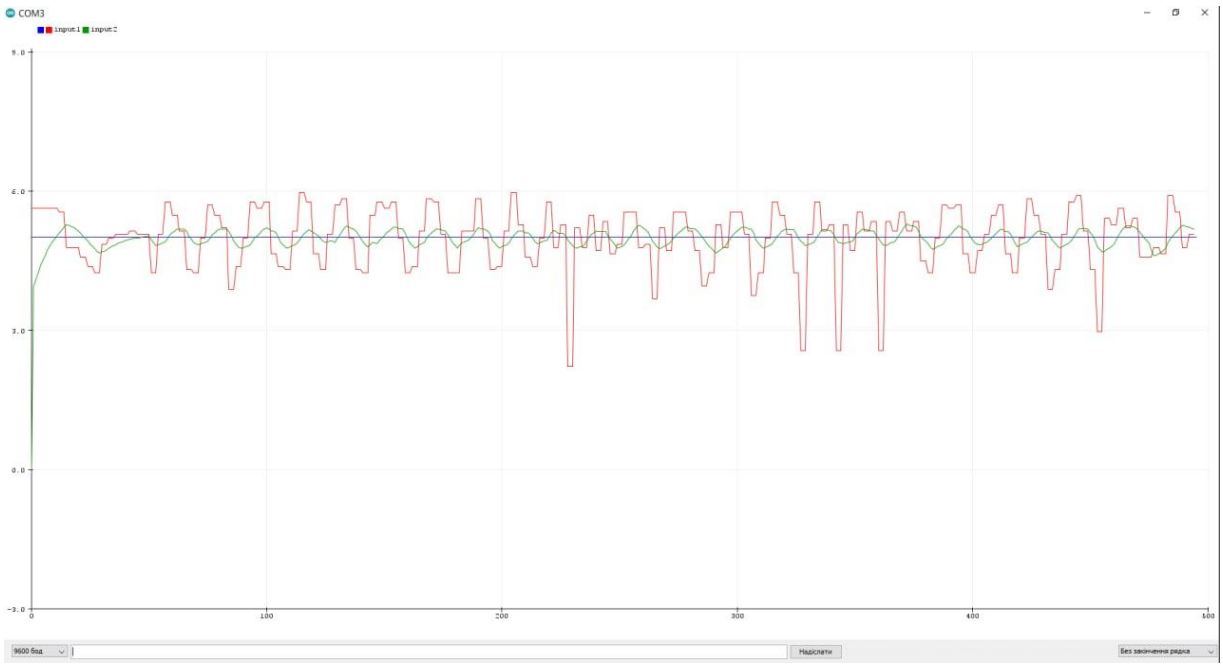

Рис. 8 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1, $К_i$=0, $К_d$=0

На цьому графіку чудово видно роботу алгоритму фільтрації сигналу отриманого з сонару, це є необхідна річ, також видно, що при коефіцієнту пропорційності, який дорівнює одному, система ніколи не стабілізується.

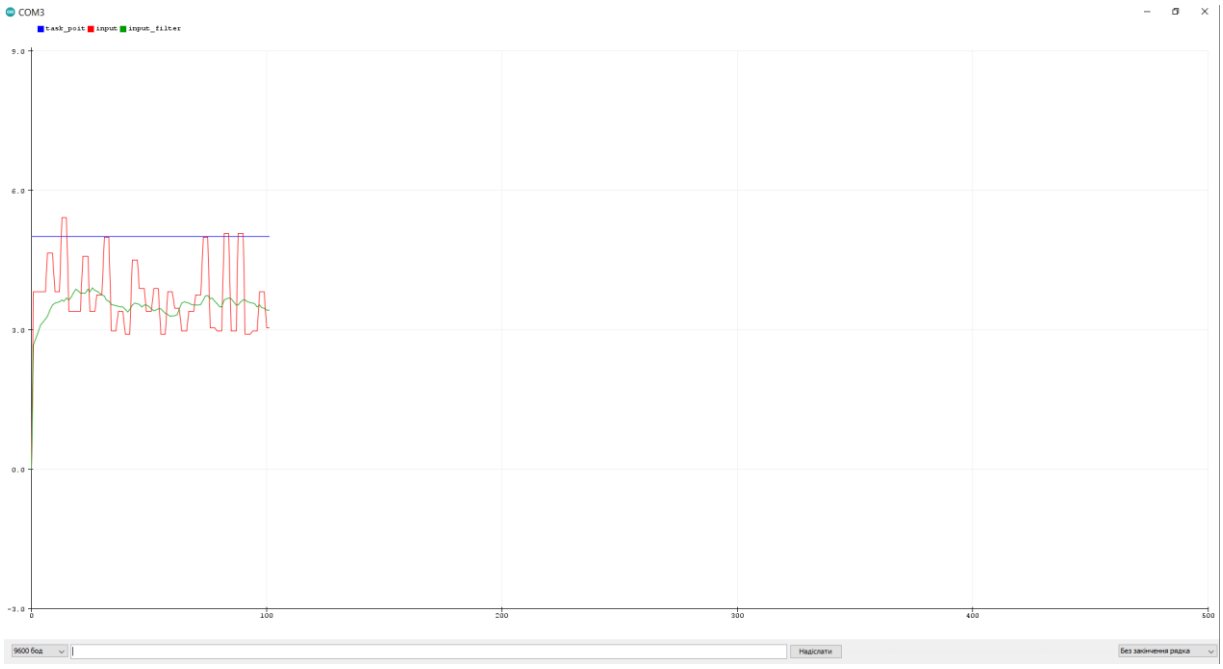

Рис. 9 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=0.5, $К_i$=0, $К_d$=0

При занадто малому коефіцієнті пропорційності, система зовсім не виходить на потрібний рубіж.

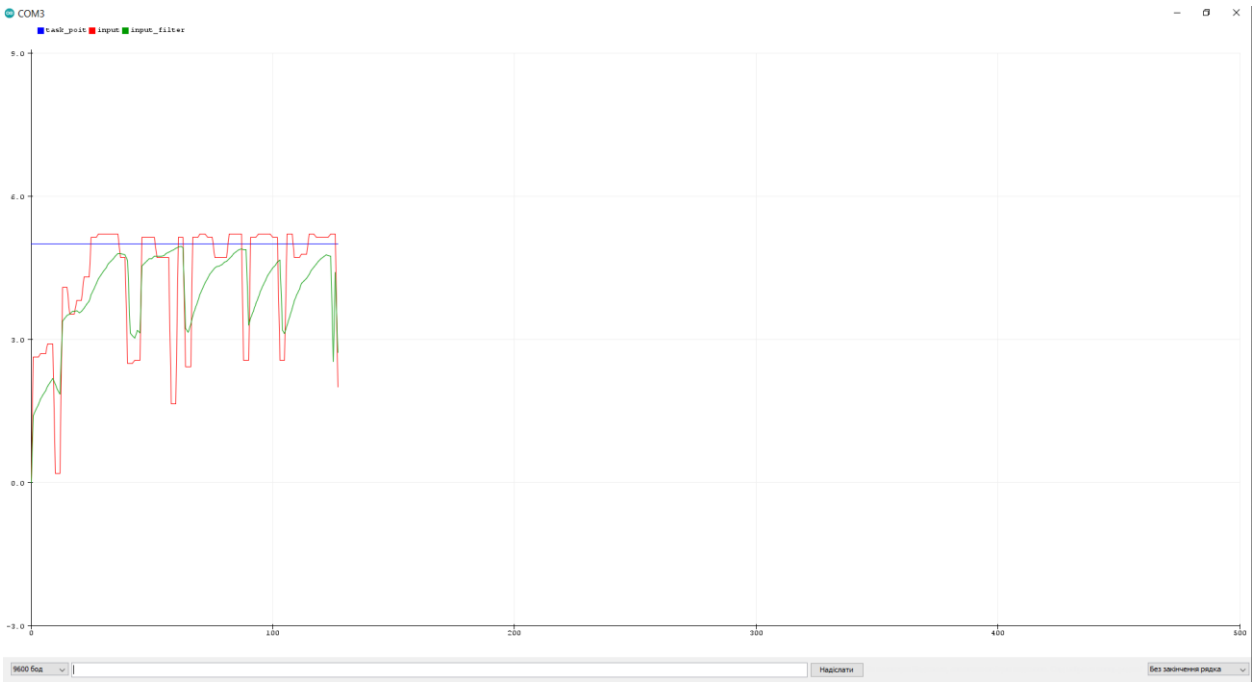

Рис. 10 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=0.5, $К_i$=0.5, $К_d$=0

Також було потрібно перевірити, що при малих значеннях пропорційності і інтегруючої складової, ніяких стабілізаційних процесів не відбувається.

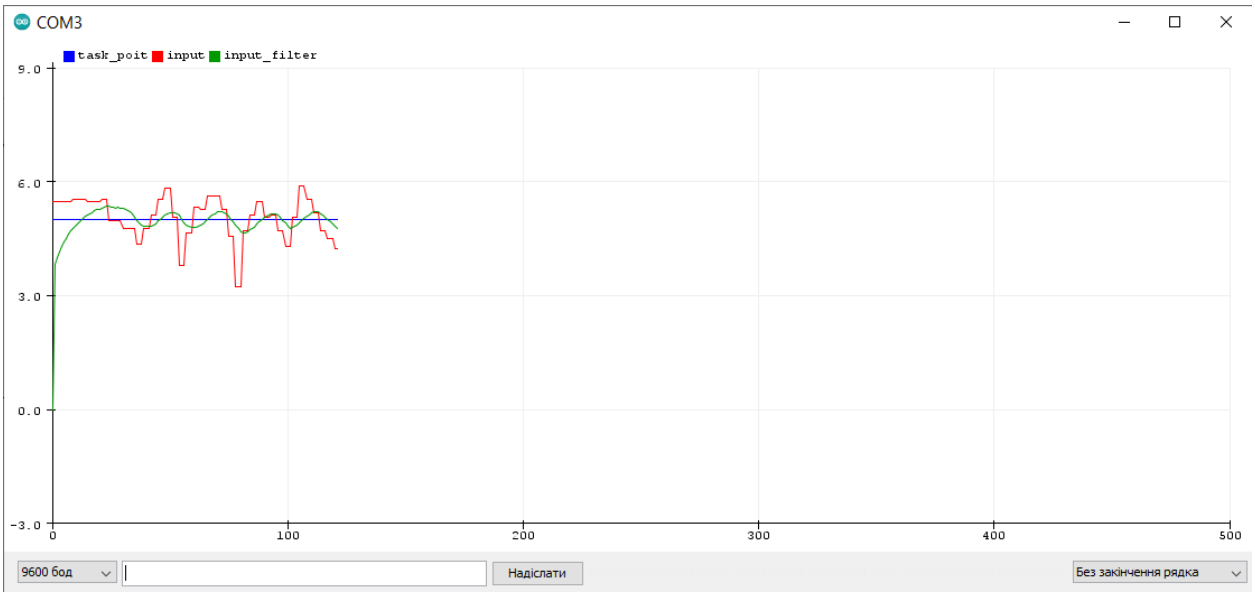

Рис. 11 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1, $К_i$=1, $К_d$=0

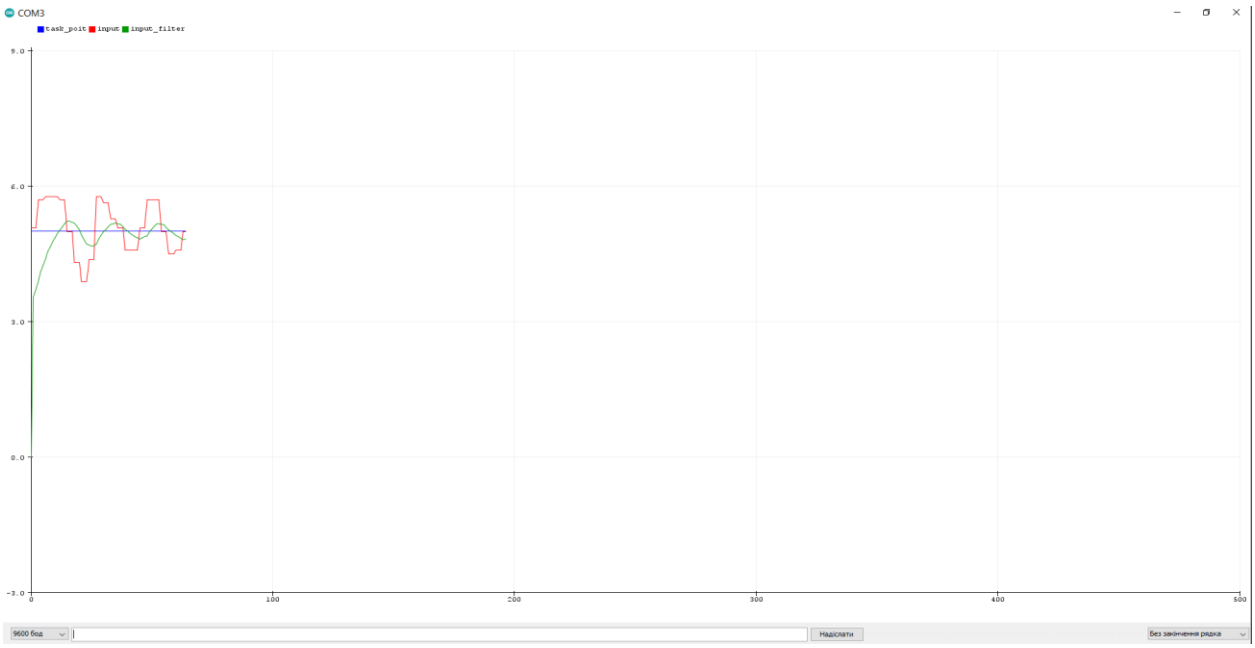

Рис. 12 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1.5, $К_i$=1.5, $К_d$=0

Графік стає виглядати більш гармонійно, але явної стабілізації системи немає.

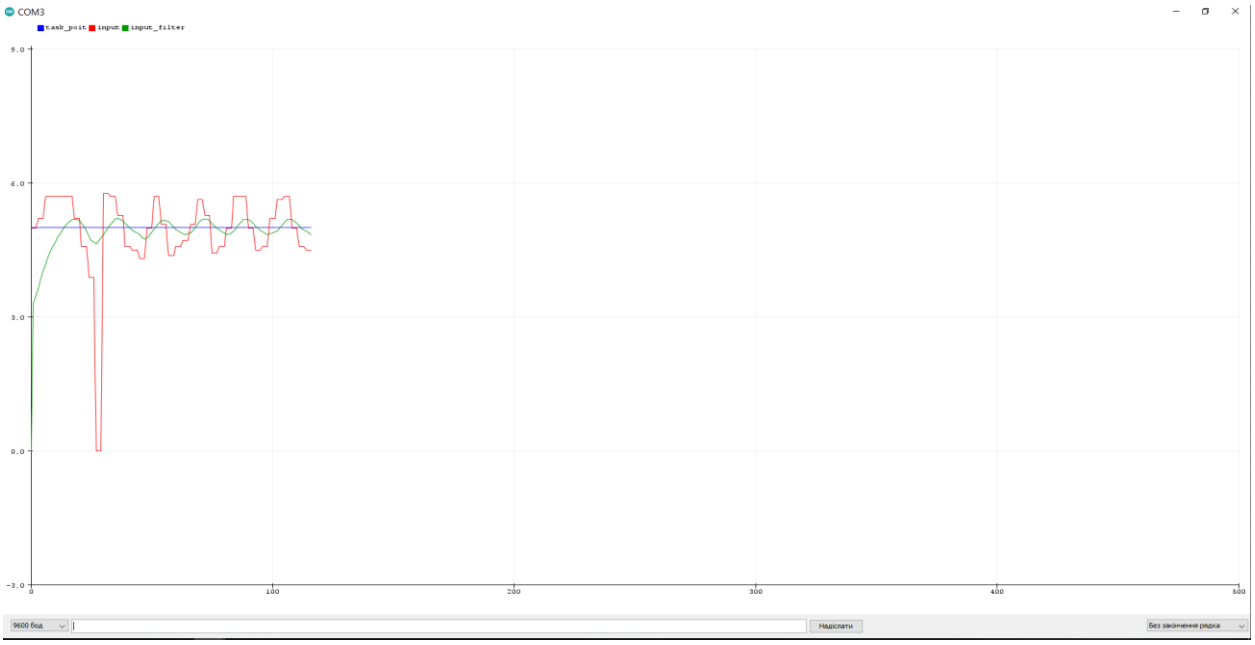

Рис. 13 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=0.5, $К_i$=0, $К_d$=0.5

З цього графіку бачимо, що при малих значеннях пропорційності і диференційної складової, достатньо лише, щоб отримати гармонійні коливання.

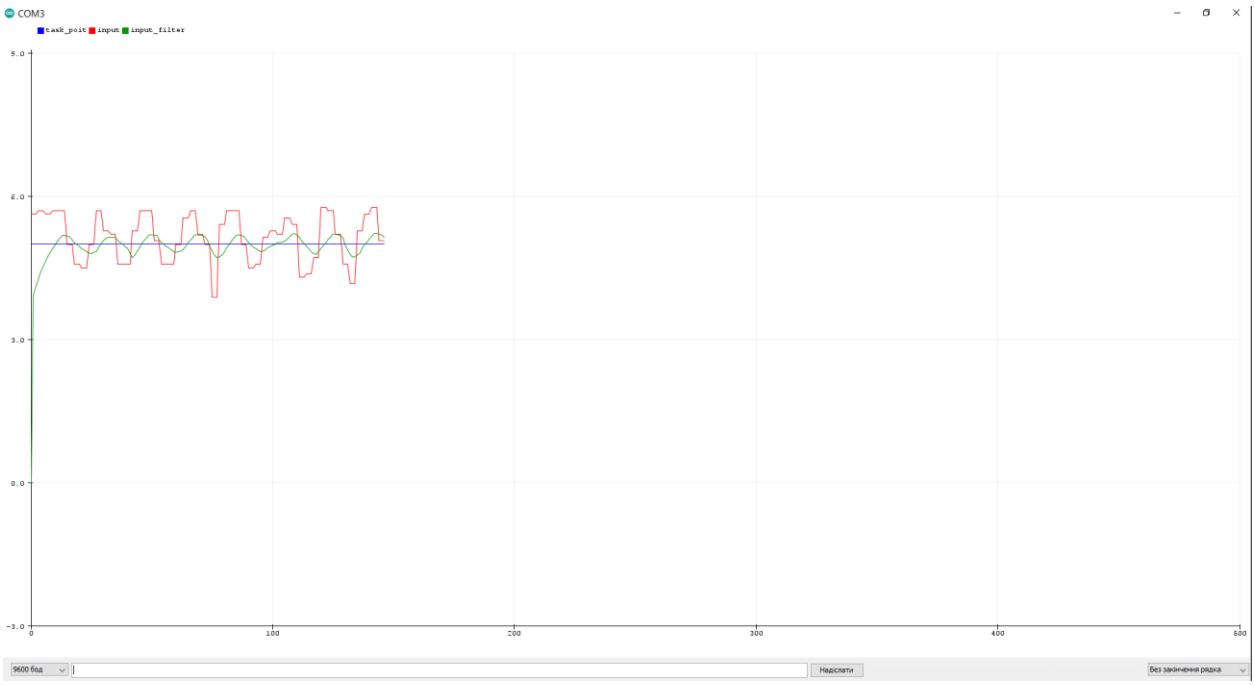

Рис. 14 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1, $К_i$=0, $К_d$=1

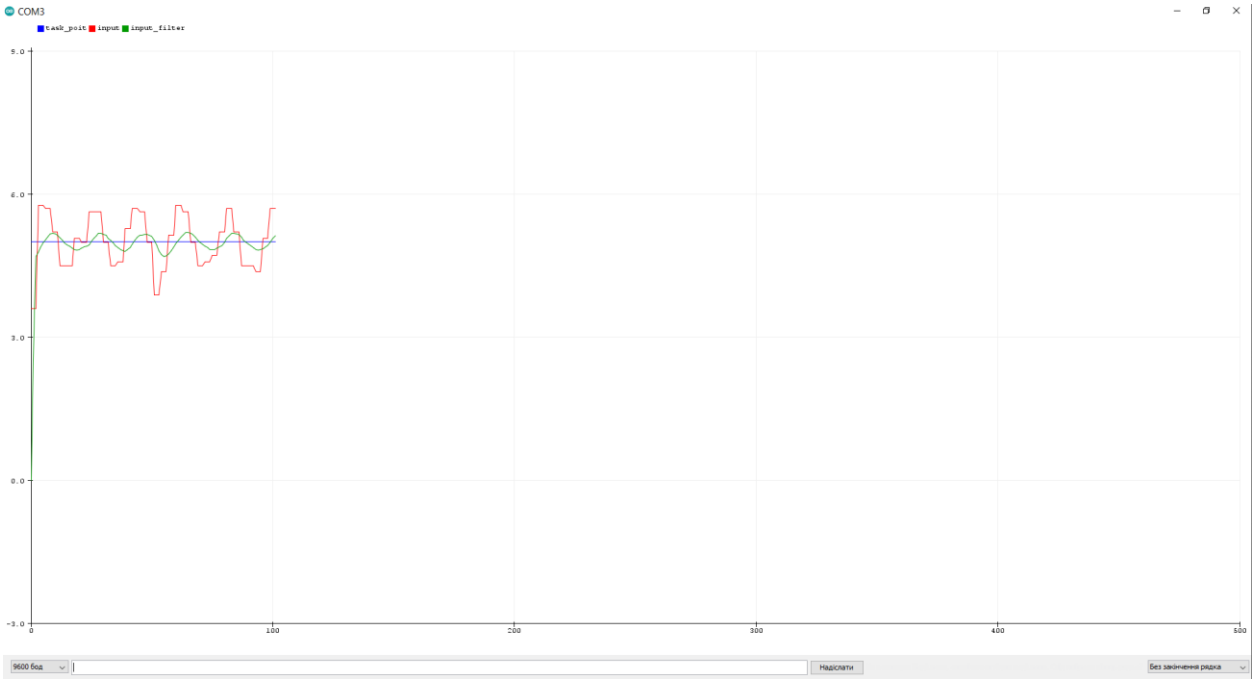

Рис. 15 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1.5, $К_i$=0, $К_d$=1

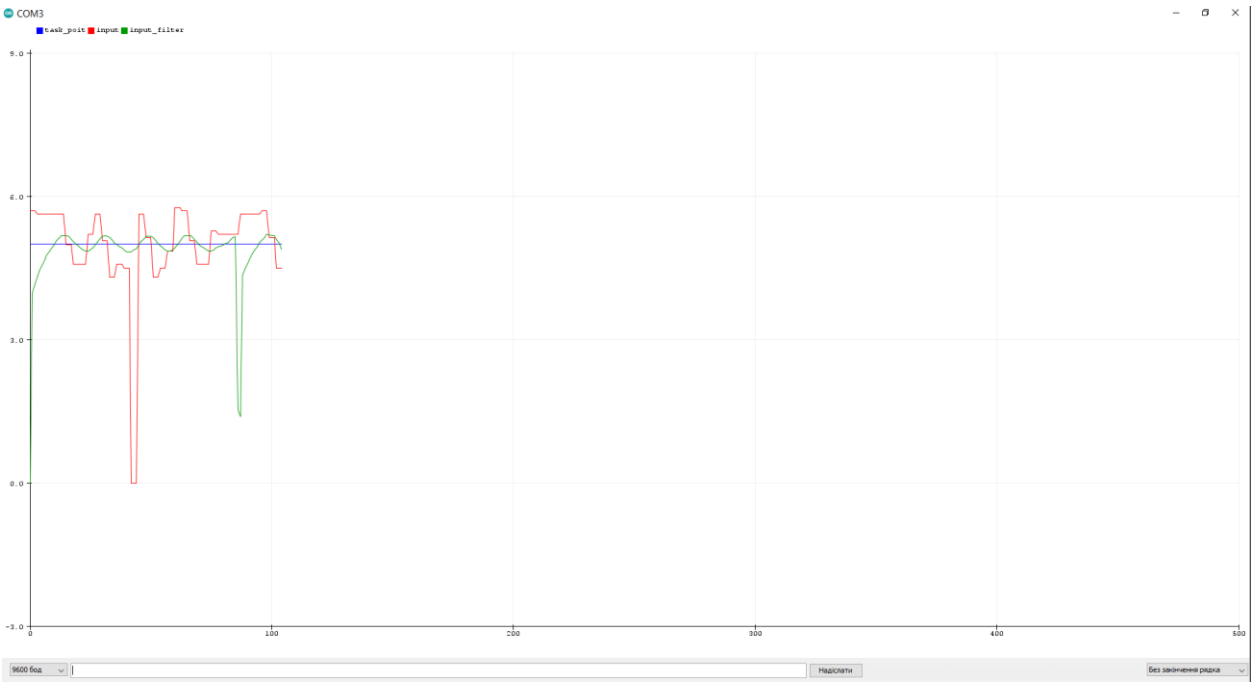

Рис. 16 Графік роботи ПІД-регулятора з коефіцієнтами K_p=1.5, K_i=0, K_d=1.5

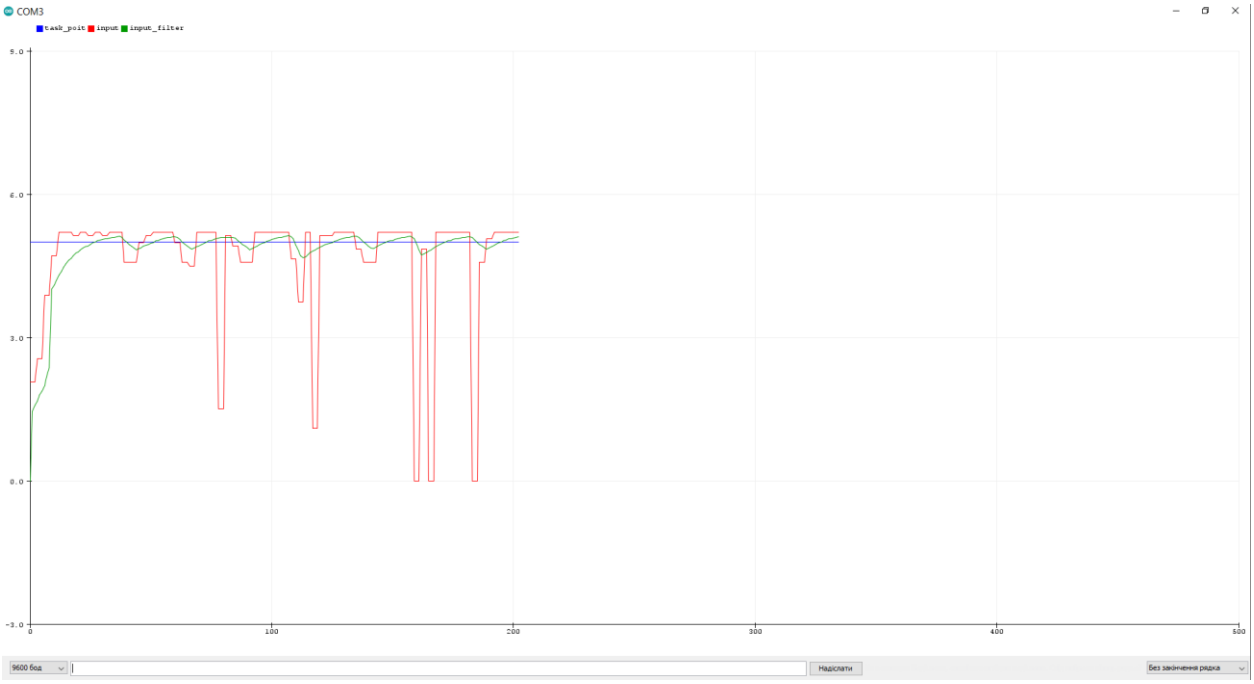

Рис. 17 Графік роботи ПІД-регулятора з коефіцієнтами K_p=0.25, K_i=0.25, K_d=0.25

За даних коефіцієнтів можемо отримати з малою амплітудою гармонійні коливання, але цього не достатньо.

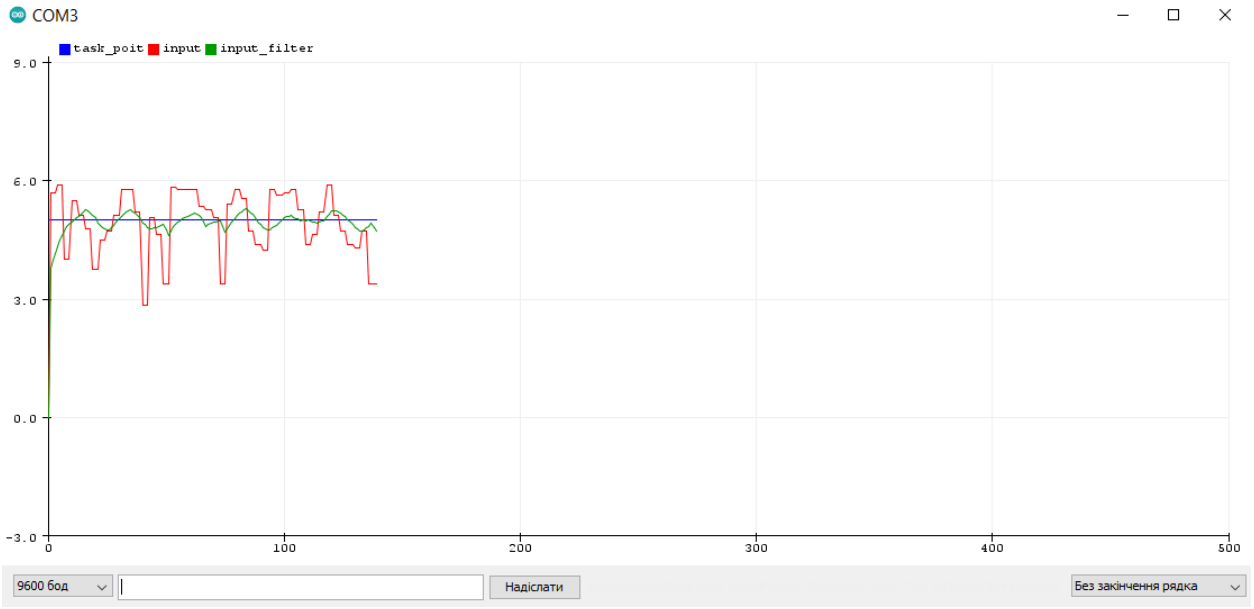

Рис. 18 Графік роботи ПІД-регулятора з коефіцієнтами K_p=0.5, K_i=0.5, K_d=0.5

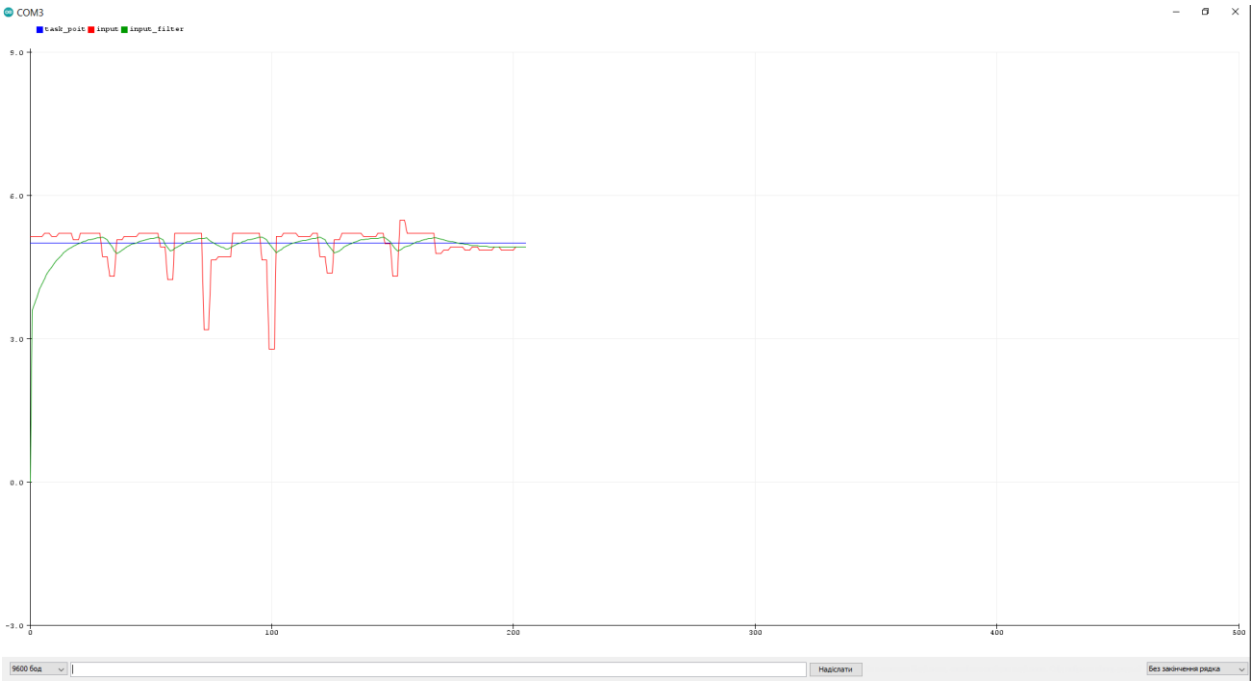

Рис. 19 Графік роботи ПІД-регулятора з коефіцієнтами K_p=1, K_i=0.5, K_d=0.5

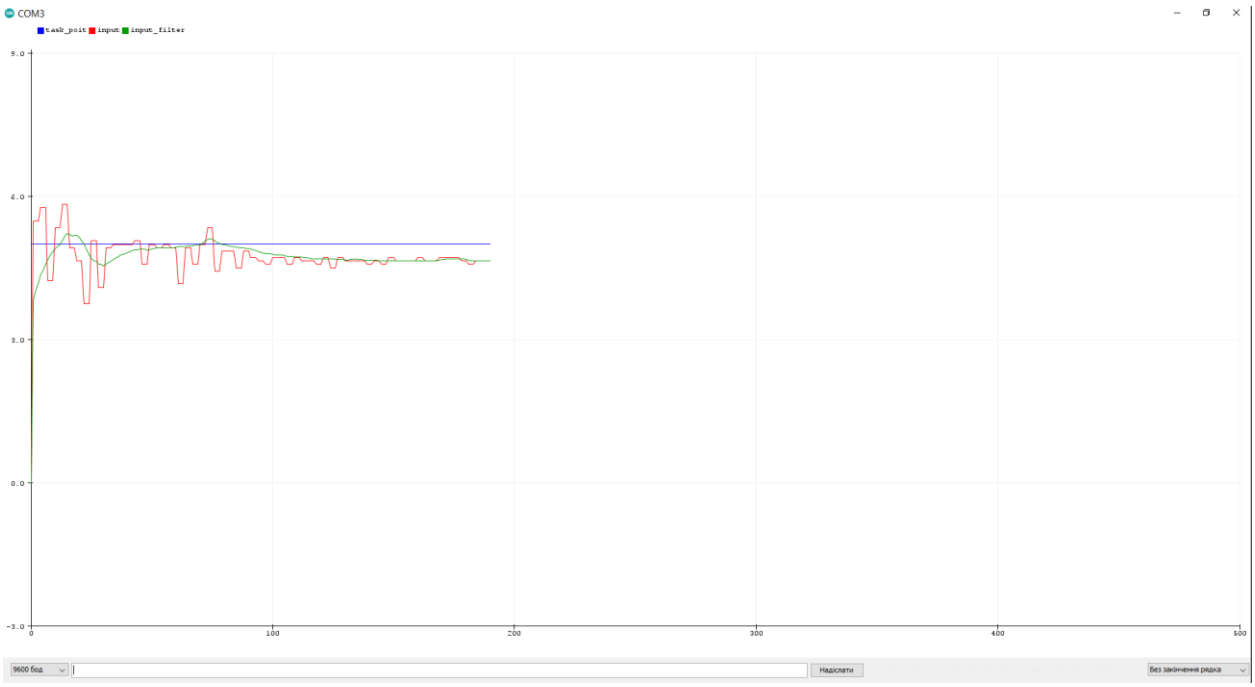

Рис. 20 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1, $К_i$=1, $К_d$=1

У підборі коефіцієнтів вже знаходимось близько, але явно можна отримати кращі результати в стабілізації системи.

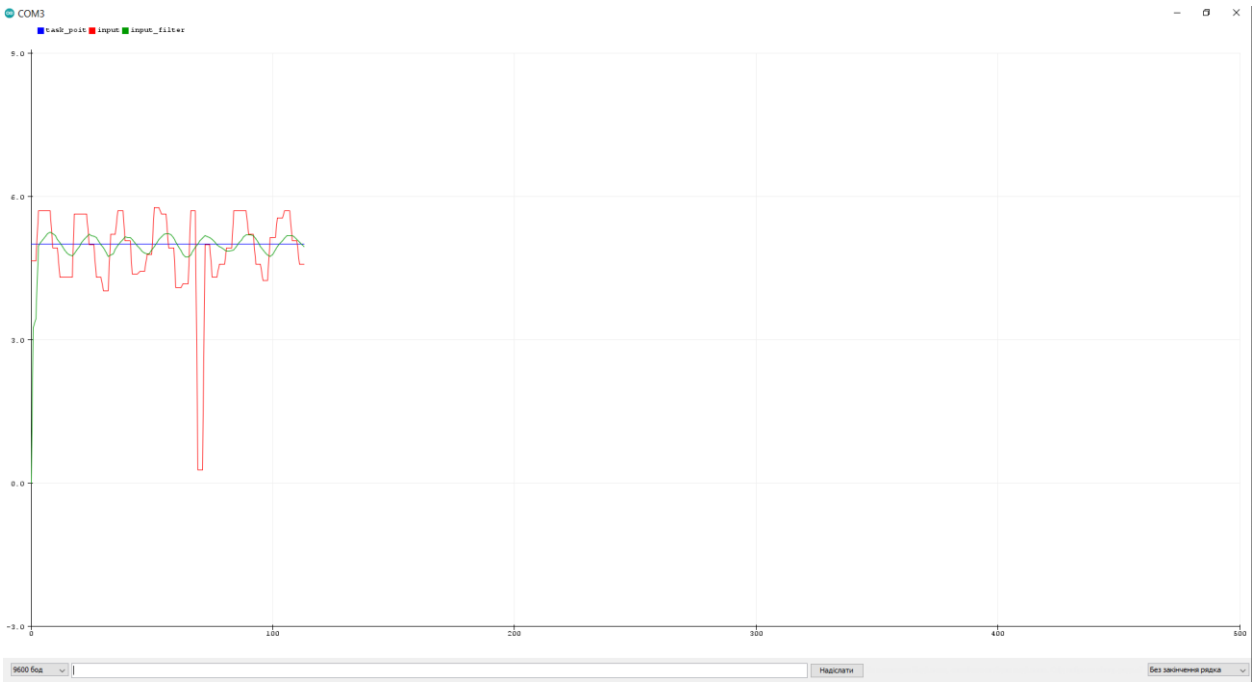

Рис. 21 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=2, $К_i$=1.5, $К_d$=1.5

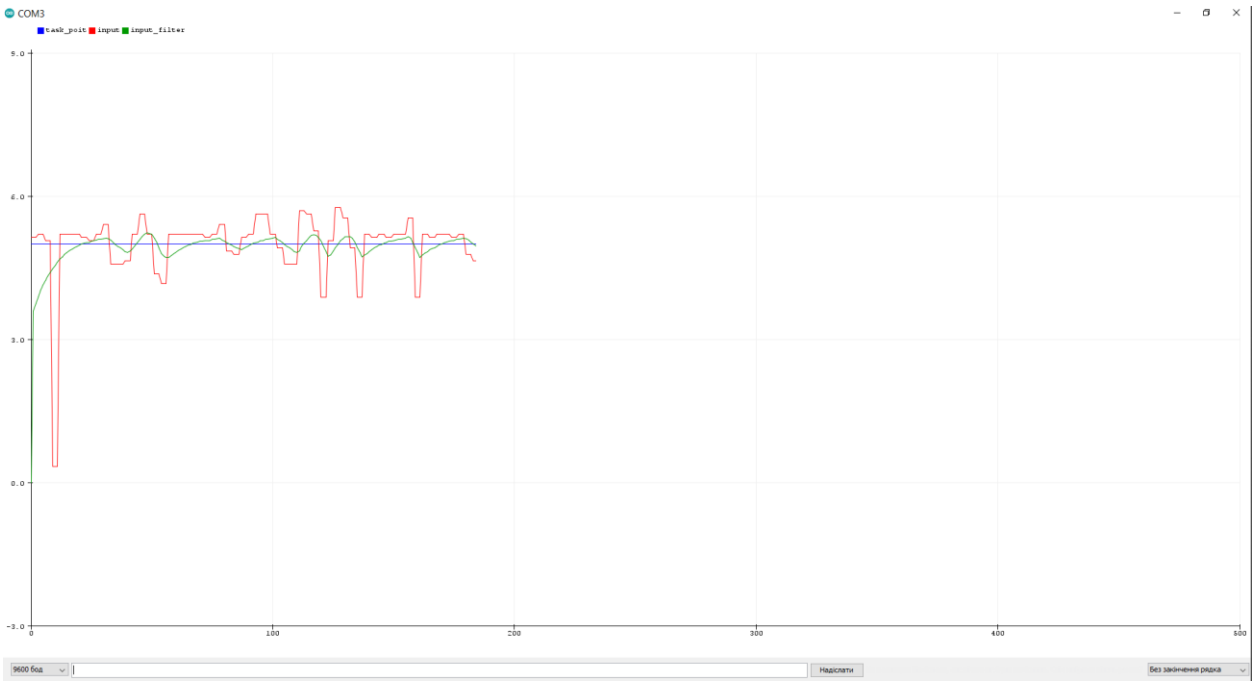

Рис. 22 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=2, $К_i$=2, $К_d$=2

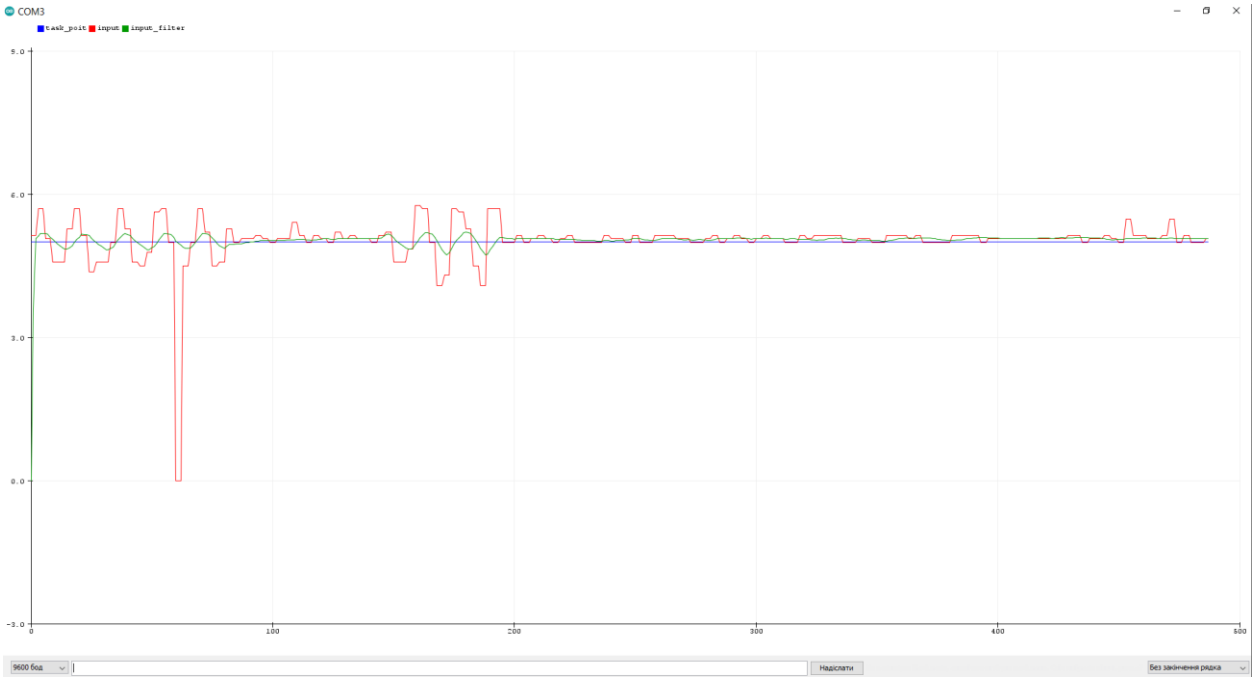

Рис. 23 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1, $К_i$=1.5, $К_d$=2

Тут можемо бачити, що система за даними коефіцієнтами може на певний час стабілізуватись, то ж потрібні коефіцієнти поруч.

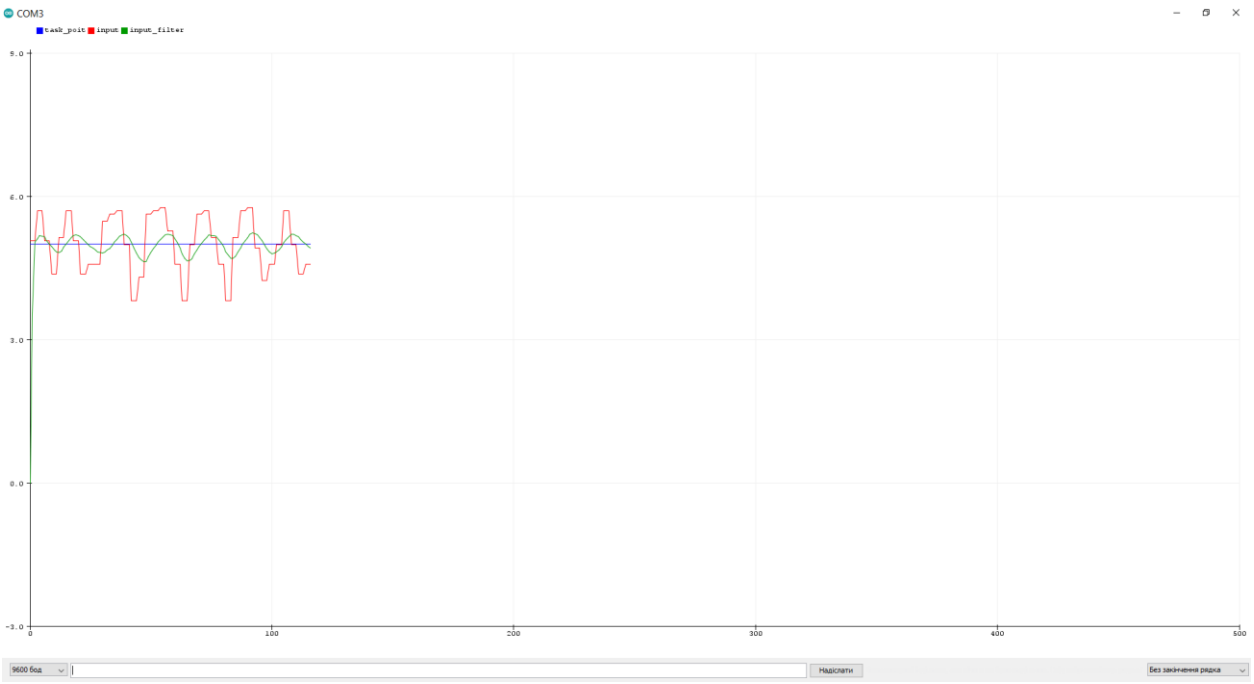

Рис. 24 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=2, $К_i$=1.5, $К_d$=1

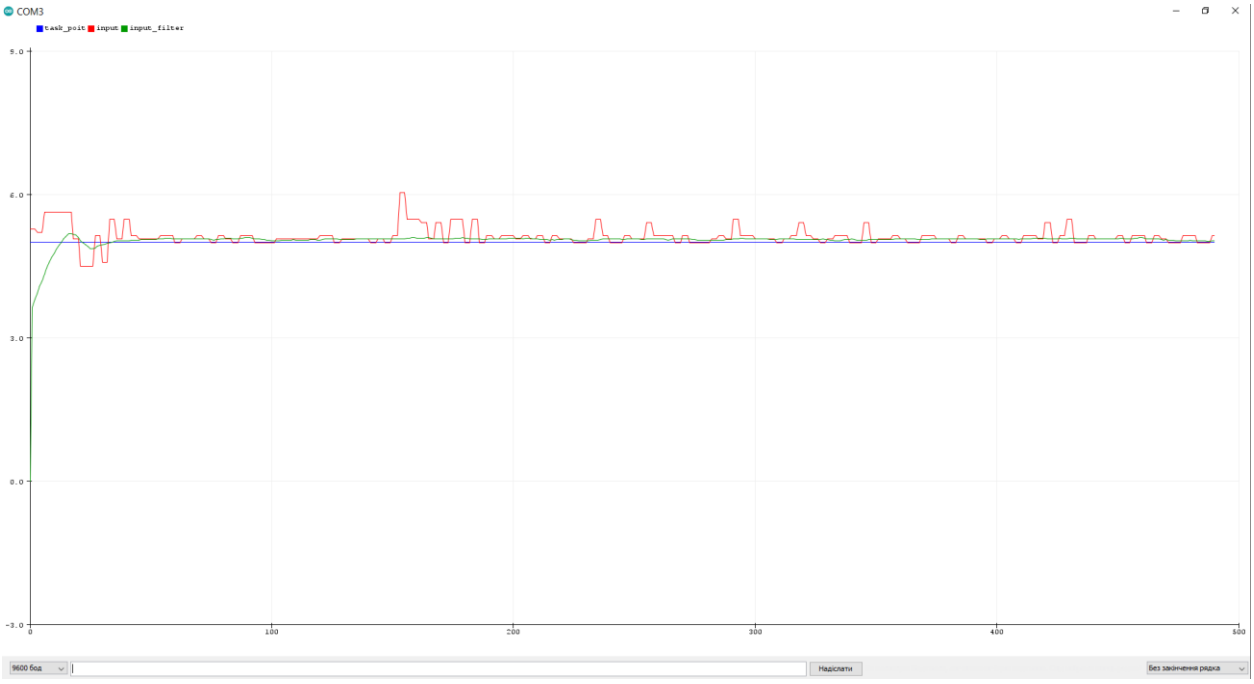

Рис. 25 Графік роботи ПІД-регулятора з коефіцієнтами $К_p$=1.5, $К_i$=1.5, $К_d$=1.5

На даному графіку бачимо майже ідеальну стабілізацію системи, і також добре видно, що певні шуми, дуже добре нівелюються програмними фільтрами, що система залишається стабільною на довгий проміжок часу.

Лістинг програми:

```
// Підключення бібліотеки для роботи з КД

#include <Stepper.h>

#define steps 10 // Кількість кроків на один оберт

// сонар

#define ECHO 2

#define TRIG 3

//time

#include <TimerOne.h>

//бібліотека сонара

#include <NewPing.h>

NewPing sonar(TRIG, ECHO, 400);


float point = 5;

float rezult = 0;

float dist_3[3] = {0.0, 0.0, 0.0}; // массив для зберігання трьох останніх вимірів

float middle, dist, dist_filtered, len;

float k;

byte i, delta;

unsigned long sensTimer;


// піни Arduino, до яких підключені обмотки двигуна

Stepper myStepper(steps, 7, 6, 5, 4);
```

```
void setup() {
     Serial.begin(9600);
     Serial.println("task_poit, input, input_filter");
     // встановлюємо швидкість обертання об./хв.
     myStepper.setSpeed(60);
     }

void loop() {
     // вимір і вивід кожні 500 мс
     if (millis() - sensTimer > 500) {
          Serial.print(point); //task_point
          Serial.print(",");
          creation_arr(dist_3);
          Serial.print(dist_3[1]); // input
          Serial.print(",");

          dist_filtered = middle_of_dist(dist_3[0], dist_3[1],
     dist_3[2]);
          Serial.println(dist_filtered); // input_filter

          rezult = computePID(dist_filtered, point, 1.5, 1.5, 0,
     0.05, 2, 6);
          // умова з визначенням руху сонару
          if (dist_filtered - point > 0.1 ){
```

```
                myStepper.step(rezult*steps);
                }
            else if(dist_filtered - point < -0.1 ) {
                myStepper.step(rezult*(-steps));
                }
            else {myStepper.step(0);}
            sensTimer = millis();    // скинути таймер
            }
      }


// (вхід, установка, п, і, д, період в секундах, мін.вихід, макс. вихід)
int computePID(float input, float setpoint, float kp, float ki, float kd, float dt, int minOut, int maxOut) {
      float err = setpoint - input;
      static float integral = 0, prevErr = 0;
      integral = constrain(integral + (float)err * dt * ki, minOut, maxOut);
      float D = (err - prevErr) / dt;
      prevErr = err;
      return constrain(err * kp + integral + D * kd, minOut, maxOut);
      }
void creation_arr(float arr[]){
      if (i > 1) i = 0;
```

```
      else i++;
      //отримати відстань до масиву
      arr[i] = (float)sonar.ping() / 57.5;
      }

// медіанний фільтр із трьох значень
float middle_of_3(float a, float b, float c) {
      if ((a <= b) && (a <= c)) {
            middle = (b <= c) ? b : c;
      }
      else {
            if ((b <= a) && (b <= c)) {
                  middle = (a <= c) ? a : c;
                  }
            else {
                  middle = (a <= b) ? a : b;
                  }
      }
      return middle;
}

float middle_of_dist(float a, float b, float c){
      // фільтрувати медіанним фільтром із трьох останніх вимірів
      dist = middle_of_3(dist_3[0], dist_3[1], dist_3[2]);
```

```
// розрахунок вимірів із попереднім
delta = abs(dist_filtered - dist);
// якщо більше то різкий коефіцієнт
if (delta > 1) k = 0.7;
// якщо менше то плавний коефіцієнт
else k = 0.1;
// фільтр "рухоме середнє"
dist_filtered = dist * k + dist_filtered * (1 - k);
return dist_filtered;
}
```

Висновок: в ході виконання лабораторної роботи ознайомились з процедурою налаштування ПІД-регулятора, і в результаті для зібраної схеми простим перебором коефіцієнтів знайшли значення , при яких система стабілізується доволі швидко. І як видно з вище наведених графіків система постійно коливається відносно заданої відстані, і відносно ідеальних коефіцієнтів. Також зібрана схема не є ідеальною, бо частково з'єднання елементів відбувалось, за допомогою макетної плати, і за допомогою багатьох невеличких проводів, дані фактори впливали на похибку вимірювань. Бо стандартна ситуація, провід десь відійшов, в результаті отримуєш графік з некоректними значеннями.

Також хочеться додати, що дана схема будувалась відносно двох простих, бо окремо складували схему для контролю сонару, і окремо – контролювання крокового двигуна за допомогою драйвера, бо Arduino не має такої потужності, щоб спокійно контролювати кроковий двигун. То ж в результаті цієї роботи, можна далі масштабувати в більш складний проект.