УНИВЕРЗИТЕТ “Св. КИРИЛ И МЕТОДИЈ” - СКОПЈЕ

ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

- ДИПЛОМСКА РАБОТА -

по предметот

Моделирање, симулација и идентификација

Тема

ПРОЕКТИРАЊЕ И УПРАВУВАЊЕ НА ЕДНОКРАК ЛОСТ СО ПИД УПРАВУВАЧ ПРЕКУ КОРИСНИЧКИ ГРАФИЧКИ ИНТЕРФЕЈС

Ментор:
доц. д-р Душко Ставров

Изработил:
Мелани Цветковска, индекс бр. 13/2020
e-mail:melanicvetkovska01@gmail.com

Скопје, јуни 2024

Содржина

Листа на слики

Абстракт

Овој труд се фокусира на дизајн и имплементација на PID (Пропорционално - Интегрално – Диференцијално) управување на еднокрак лост со помош на развиен кориснички графички интерфејс (GUI – Graphical User Interface). Целта е да се обезбеди практично решение кое овозможува лесна конфигурација и оптимизација на PID (Proportional – Integral – Differential) параметрите, како и визуелизација на однесувањето на системот во реално време. За цели на реализација на проектот, беа користени различни методи за моделирње и анализа на системот, вклучувајќи симулации и експерименти на физички модел. Резултатите покажуваат дека PID управувачот обезбедува стабилно и прецизно управување, додека графичкиот интерфејс значително го олеснува процесот на експериментирање и учење. Овој проект може да послужи како ефикасна образовна алатка која може да инспирира идни истражувања и развој во областа на системите на автоматско управување и автоматизацијата.

Клучни зборови:

PID управување, стабилност, одзив на систем, графички кориснички интерфејс, алгоритми

Вовед

Напредокот на алгоритмите за управување на системите направило револуција во многу области, од индустриска автоматизација до роботика и воздухопловно инженерство. Основите на оваа револуција ги поставува токму PID алгоритмот за управување (PID – Proportional-Integral-Derivative), познат по својата едноставности и ефикасност во различни управувачки системи. Потполното разбирање и оптимизирање на параметрите на PID управувачите – односно пропорционалната константа K_p, интегралната константа K_i и диференцијалната константа K_d - се клучни за достигнување на посакуваните перформанси и стабилност на системот. За таа цел, во рамките на овој дипломски труд е проектирана и реализирана физичка макета, која обезбедува опиплива платформа за експериментирање и валидација на теориските концепти.

Физичката макета е мост помеѓу теориските симулации, кои ги тестираат нашите теории, и апликација на ваквите модели во реалниот свет. Додека симулациите овозможуваат да имаме управувана средина за предвидување на одѕивите на системите, тие често не успеваат да ги доловат вистинските реални ситуации и нивната сложеност која зависи од безброј надворешни фактори. Факторите како механичко триење, шум, немоделирана динамика, можат значително да влијаат на перформансите на системот и затоа најдобро можат да се разберат нивните влијанија преку практично експериментирање. Користејќи 3Д печатен модел, на кој ќе имаме интегриран еднонасочен мотор без четкички (brushless DC motor – BLDC), а кој ќе биде управуван преку Arduino Uno со помош на PID алгоритмот, ќе создадеме достапна и ефтина платформа за тестирање и усовршување на стабилноста и одѕивот на системот.

Во дипломски труд, имаме 3Д печатена основа за макетата и осовина која ќе овозможува подигање на една ПВЦ цевка, која игра улога на лост. На другиот крај од цефката се наоѓа позициониран и зацврстен BLDC мотор, кој е актуаторот во системот, а ќе биде управуван со директно посредство на ESC (Електронски контролер на брзина), а кое добива управувачки команди од PID управувачкиот алгоритам. Arduino Uno е мозокот на системот и управува со ESC-то врз основа на повратните информации од потенциометарот, поставен во центарот на ротација на лостот – осовината, кој обезбедува податоци за позицијата на лостот во реално време. Со тоа системот се става во затворена јамка и е стабилизиран. Реализацијата на PID алгоритмот во оваа макета овозможува прецизно управување на аголот на кој ќе се наоѓа прачката, покажувајќи го директно влијанието на PID-овите параметри K_p, K_i и K_d врз поведението на системот. Главна цел при изборот на вредности на овие параметри најпрво е гарантирање на стабилноста на затвореният систем, а потоа е постигнување на солидна брзина во преоден режим и задоволителна прецизност во стационарен режим. Затоа практичното искуство е круцијално за комплетно разбирање на PID алгоритмот и како секоја негова компонента придонесува за севкупната стабилност и перформанси на системот.

Дополнително, мора да се нагласи и важноста на графички интефејс во еден ваков систем, особено за демонстративни цели и корисници кои немаат голема програмска позадина. Интутивниот и лесен за користење HMI (анг. Human Machine Interface) го олеснува пристапот до комплексни системи за управување и овозможува на поширока публика пристап да се вклучи во разбирање на принципите на алгоритми за управување, како што е PID алгоритмот во овој случај. Овој графички интерфејс ќе биде изработен во Windows Forms и ќе овозможи комуникација со Arduino преку сериска комуникација на USB портата со UART (анг. Universal Asynchronous Receiver – Transmitter) протоколот. Дополнително, ќе има можност за подесување на сите три PID компоненти и поставување на посакувана вредност (анг. Setpoint) – посакуван агол на кој ќе балансира прачката. Воедно ќе има и копчнија за стартување и сопирање на макетата што ќе додаде и можност за поголема сигурност за корисниците. Со ова оваа макета од едукативна алатка за разбирање на алгоритми за управување прераснува во интерактивен начин на нивно учење и им овозможува на корисниците на експериментираат и веднаш да ги согледаат резултатите од различните влијанија и прилагодувања.

Способноста да се визуелизираат и манипулираат PID параметрите и нивните влијанија поттикнува подлабоко нивно разбирање и теоријата на автоматско управување. Зголемувањето на пропорционалната компонента ќе внесе побрзи одзиви на системите, инегралното дејство ќе ги елиминира стационарните грешки и диференцијалното ќе се погрижи за придушување на осцилациите. Таква директна интеракција помага за опипливо разбирање на апстрактните концепти од теорија на автоматско управување. Покрај тоа, комплетното разбирање на ваквите системи за управување ја зголемува и довербата во таквите, а со тоа и нивната употреба во реалниот свет за реални проблеми и ситуации.

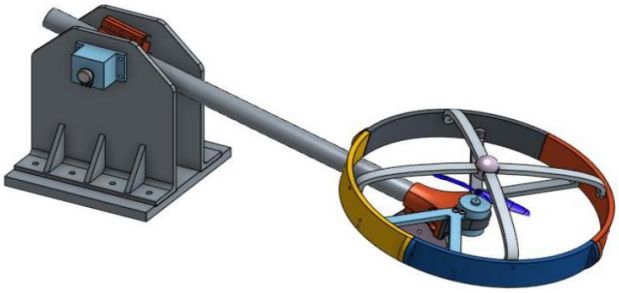
Во продолжение, дипломскиот труд е организиран така што најпрво се претставени неговите хардверски компоненти и нивниот начин на функционирање. Потоа е дискутиран применетиот алгоритам за негово управување и неговите различни одзиви. На крајот е опишан софтверскиот дел од трудот, односно корисничкиот графички интерфеј и е претставено успешното управување на системот. Дополнително се дадени и идните можни решенија и подобрувања како подлога да продолжување на трудот.

Дизајн на макетата

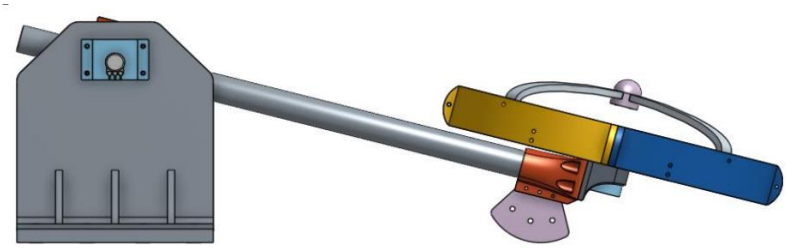
Физичко моделирање на макетата

Во оваа секција ќе го дискутираме физичкото моделирање на макетата. Најпрво, во 3Д софтвер е дизајнирана конструкција и тоа како физички ќе биде изведена макетата. Беше одлучено за 3Д печатење бидејќи е најлесниот, најпристапен и најефтин начин за изведба.

По дизајнирањето беше одлучено макетата да се изведе во структурна форма:



Слика 1 Изометриски преглед на моделот на макетата



Слика 2 Страничен преглед на моделот на макетата

Во нашиот случај заштитната обиколка околу перката не беше изведена, но постои како идно можно решение кое може да се воведе за зголемување на сигурноста на макетата при работа со нејзе.

Покрај 3Д принтаните делови, е искористен и потенциометар, поставен во оската на ротација на прачката, преку кој се реализира повратната врска во овој систем. Употребен е наједноставен, нелинеарен ротационен потенциометар, од 0.5 Watts и 1000 Ω. Може да се искористи и оптички ротационен енкодер, но е поскапа варијанта. Избраниот потенциометар сосема задоволително ја реализира поставената задача за обезбедување повратна информација за аголната позиција на лостот. Дополнително, претставува клучна алатка во реализација на ефективно управување на позицијата на прачката.

Мапирањето на вредноста на потенциометарот, која го преставува падот на напон, со соодветниот агол, во опсегот од -60 до 30 степени, е направено експериментално, преку изведување на функција (статичка карактеристика) во Excel така што беа земани семплирачки точки блиску една до друга и запишување на вредноста на потенциометарот

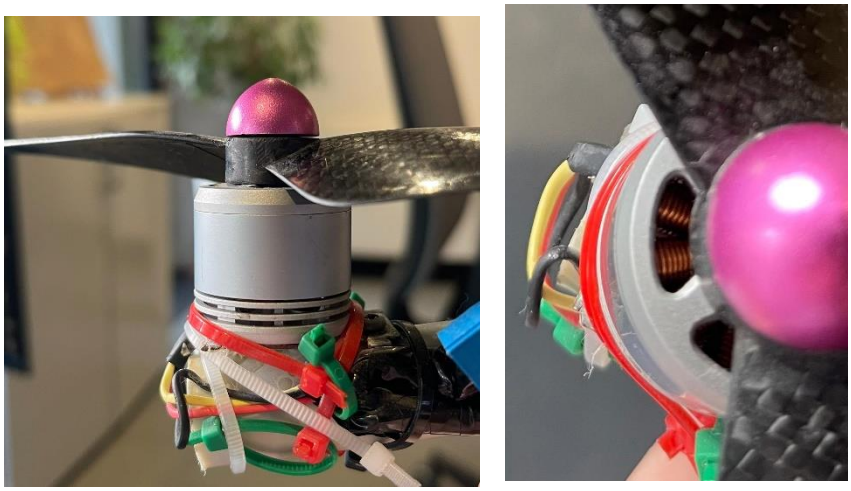
токму во овие точки. Откако го направивме ова мапирање овозможивме корисникот на макетата да внесе посакуван агол, а програмата која се напоѓа во Ардуиното да ја пресмета соодветната вредност на напон на која треба да се најде потенциометарот.

$$V = IR$$

$V =$ напон
$I =$ јачина на струја
$R =$ отпорност

Слика 3 Нелинеарен потенциометар

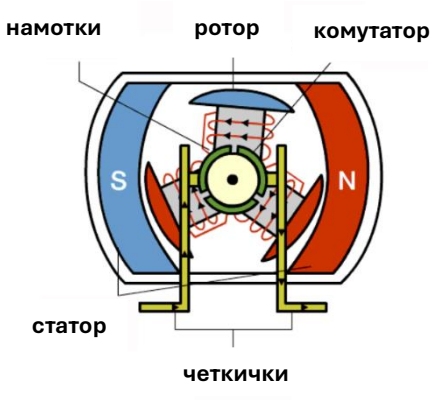
На крајот од прачката се наоѓа еднонасочен мотор без четкички (BLDC – brushless direct current) кој е актуатор во системот. Тој работи на 12V и постигнува 17 200 вртежи во минута. Управувачката команда од PID управувачот која доаѓа од драјверот ја конвертира во соодветна брзина на перките.



Слика 4 Еднонасочен мотор без четкички употребен во макетата

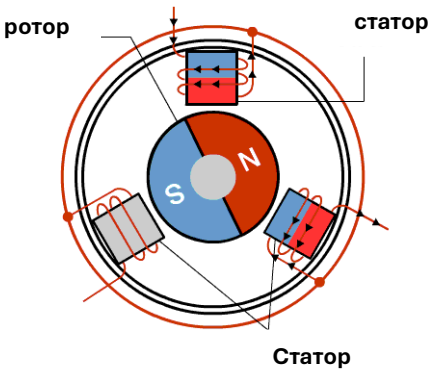
Кај еднонасочните мотори со четкици, четкиците ја пренесуваат струјата низ намотки кои се поставени во фиксно магнетно поле. Струјата генерира магнетно поле во намотките и со тоа тие ротираат така што се одбиваат од статорот кој е исто поларизиран. Дополнително

поларитетите на намотките постојано се менуваат за ова ротирање да е константно. Струјата стига до намотките преку проводните четкички кои се во постојан контакт со ротирачкиот комутатор чија ротација овозможува обратна струја во намотките.



Слика 5 Страничен преглед на моделот на макетата

Додека пак, кај еднонасочните мотори без четкички, намотките се статички поставени на статорот, не се наоѓаат на роторот. Роторот во овој вид на мотори го игра постојано поларизиран магнет. Со промена на брзината на ротација се менува напонот на намотките. Ротацијата се управува со промена на големината и насоката на струјата која минува низ намотките.



Слика 6 Еднонасочен мотор без четкички

Но, бидејќи не можеме директно да го управуваме моторот, посредник на управувачкиот сигнал ќе го игра Електронскиот регулатор на брзина (ESC- Electronic Speed Controller). Користиме Turnigy Multistar ESC, кој ја менува брзината на моторот преку промена на напојувањето на моторот. Бидејќи имаме еднонасочен мотор без четкички тоа се постигнува

со вклучување и исклчучување на моторот во секвенца, преку методот на модулација на ширина на импуслот (PWM – Pulse Width Modulation). Ова ESC се напојува со помош на ЛИПО батерија или во овој случај преку адаптер за напојување, Speedport W 503V Type A, Type: FW7576/EU/12, кој се карактеризира со излез од 12V напон и 1A струја, а влез од 230V напон, 50-60 Hz и 130mA проток на струја.

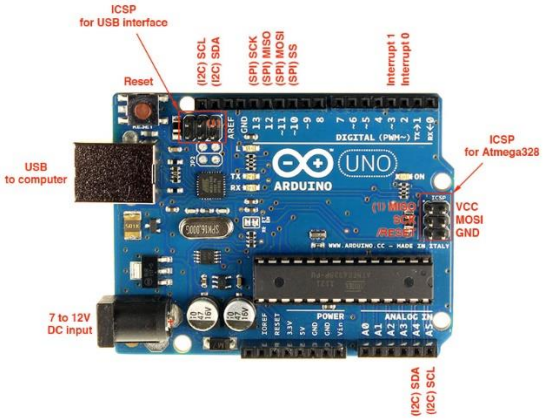
Слика 7 Електронски регулатор на брзина

Микроуправувачи и Arduino Uno

Микроуправувач е компактно интегрирано коло кое ја претставува управувачката единица во системите. За разлика од микропроцесоорите, кои се користат во персоналните компјутери, Микроуправувачите се оптимизирани да управуваат и контролираат специфични задачи и системи. Денес можеме да ги сретнеме во широк спектар на апликации, во секојдневни електронски уреди почнувајќи од телефоните до индустриски машини.

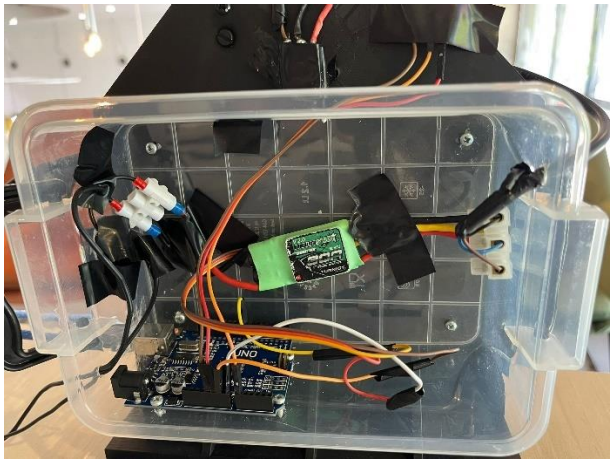
Мозокот позади управувањето на нашиот проект е микроуправувачот Arduino Uno, кој во оваа ситуација не беше специфично биран по тип као тип на управувач од серијата на Ардуино управувачи. Има доволен број на пинови за влез и излез за нашиот систем, доволно силно напојување и неговата физичка големина не играше улога во бирањето бидејќи е поставено во безбојна кутија прикачена на макетата.

Од микроуправувачот се користи дигиталниот пин 8 за испраќање на PID сигналот до ESC-то, кој добиената управувачка команда, која е во определен опсег, ја трансформира во PWM сигнал со кој се регулира брзината на моторот. Преку аналогниот пин A0 се чита повратната информација од потенциометарот за тоа на кој агол се наоѓа прачката. Потенциометарот исто така е поврзан на пинот 5V и GND за да добие напон за работа. А ESCто е поврзано на пиновите V_{in} и GND за да се напојува преку Arduino. Ќе ја користиме библиотеката <Servo.h> за иницијализација на ESC-то во Arduino програмата и потоа преку наредбата (value) ќе испраќаме соодветна ширина на пуслот на PWM сигналот кој се испраќа до ESC-то за тоа потоа да го управува вртежниот момент на DC моторот.



Слика 8 Arduino Uno

Прозирната кутија ни игра улога на заштитно куќиште за електронските компоненти на системот, при што би се спречиле како физички поместувања на компонентите, така и надворешни влијанија како влага, вода, физички контакт.



Слика 9 Прозирна кутија во која се сместени електронските компоненти

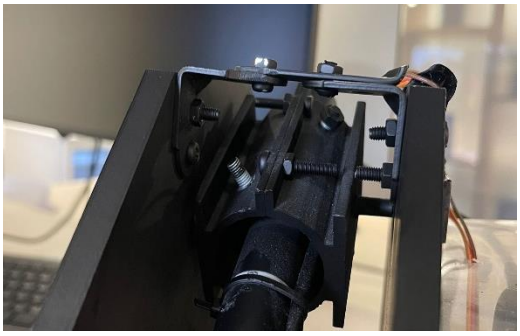
Во системот додадовме и стандарден прекинувач кој ќе се погрижи системот да стартува и да се стопира кога корисникот ќе посака, што воведува уште една мерка на сигурност.

Слика 10 Прекинувач употребен во системот

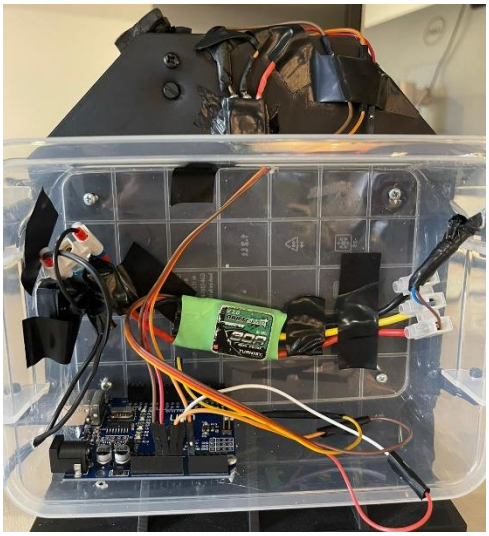
На врвот на DC моторот се наоѓа перка која е извадена од дрон.

Слика 11 Перка од дрон

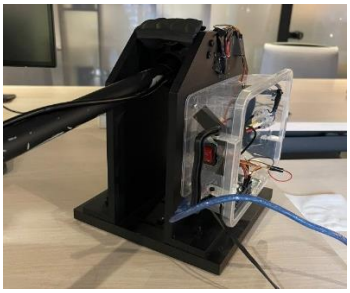
Во прилог, физички приказ на макетата која беше изработена и врз која ќе го вршиме управувањето:



Слика 12 Приказ на 3Д принтана осовина



Слика 13 Приказ на кутијата со електронски делови

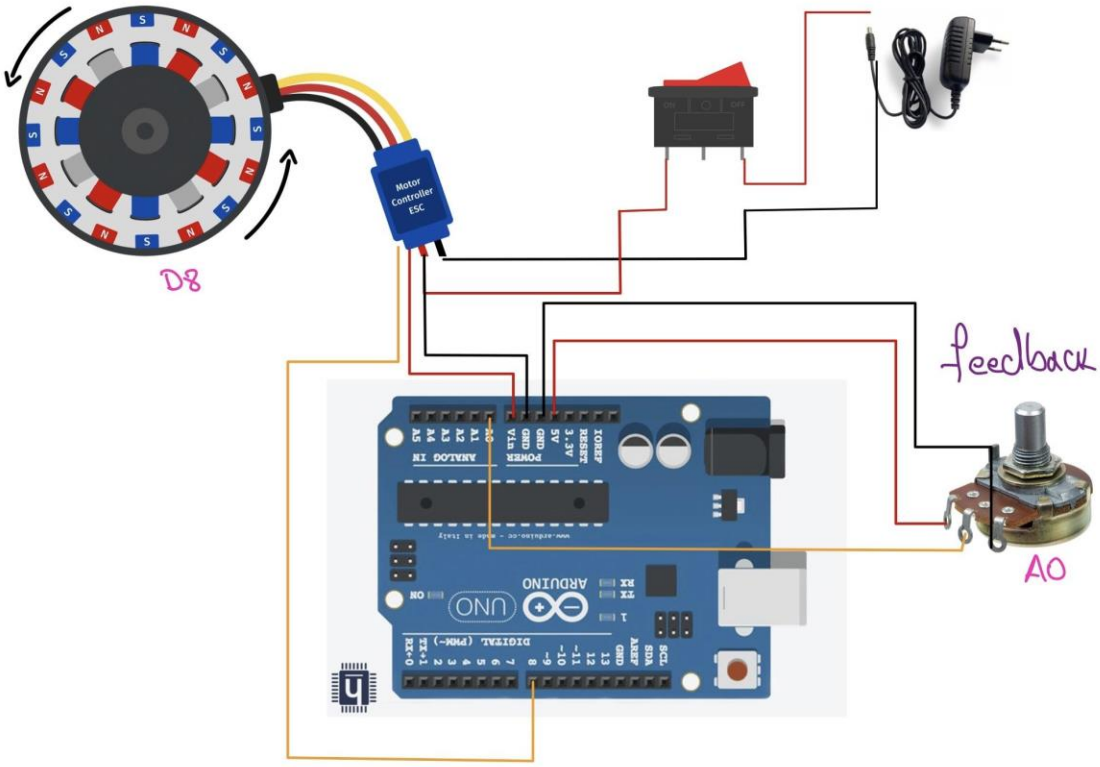


Слика 14 Бочен приказ на макетата

Слика 15 Приказ на DC моторот и перка

Слика 16 Приказ на 3Д принтана основа на макетата

Електрична шема на системот:



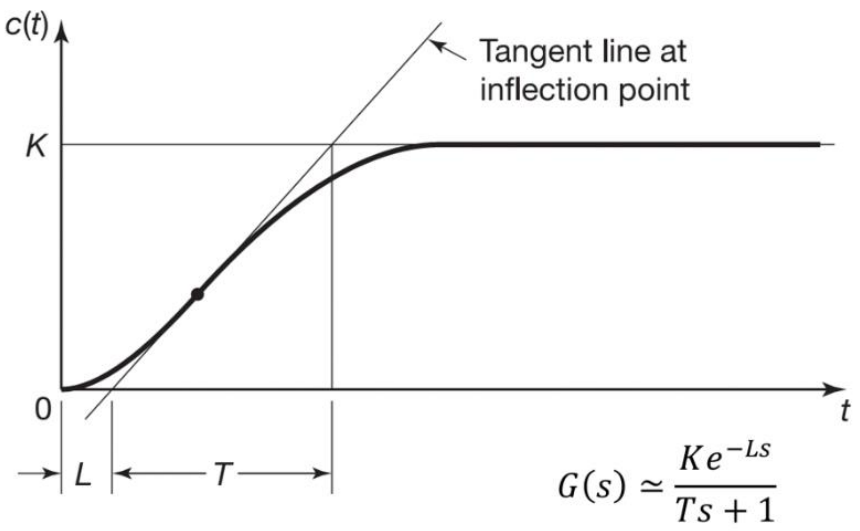
Слика 17 Електрична шема на системот

- D8- сигнал до ESC
- Vin+GND- за напојување на ардуино преку ESC
- A0-сигнал од потенциометар
- 5V+GND- напојување на потенциометарот

Идентификација на системот

Во нашиот случај, не направивме директна идентификација на системот, бидејќи се работи за нестабилен отворен систем. Но доколку симулациски го разгледуваме системот, можеме да направиме апроксимација на преносна функција со модел од прв ред.

Потребно е од преодната карактеристика на управуваниот систем да се пронајдат следните параметри како што се прикажани на слика 17.



Слика 18 Определување на параметрите од преодната карактеристика на управуваниот систем

Потребно е да се одреди точка во која кривата на одзивот има најголем раст и во неа да се повлече тангента, и понатаму параметрите што се одредуваат се :

- K - процесното засилување.
- L - временското доцнење.
- T - временската константа.

Така со одредените параметри може да се апроксимира преодната карактеристика со следната преносна функција.

$$G(s) = \frac{Ke^{-Ls}}{1+Ts}.$$

PWM (Pulse Width Modulation)

PWM е техника за енкодирање информации за управување на енергија (напон, струја) со која се снабдуваат електрични уреди. Во нашиот конекст ESCто управува со DC мотор. PWM овозможува ефикасна и прецизно управување на вртежниот момент на моторот преку менување на работниот циклус (duty cycle) на дигиталниот сигнал кој го испраќа ESC-то до него.

Duty cycle

Работниот циклус на PWM сигналот е еден период во кој тој сигнал е на високо напонско ниво (high), односно е активен. Се дефинира како однос помеѓу ширината на пулсот (време за кое е активен) и вкупниот период на сигналот.

- Пример: 50% работен циклус значи дека сигналот е висок за половина од периодот, а оснатата половина низок

Фреквенција

Фреквенцијата на PWM сигналот е број на циклуси во секунда измерен во Херци (Hz). Овој број ни одредува колку брзо сигналот се префрла од висока во ниска состојба и обратно.

Амплитуда

Амплитудата на PWM сигналот е нивото на напон на високата состојба. Во повеќето дигитални системи изнесува 3,3V или 5V.

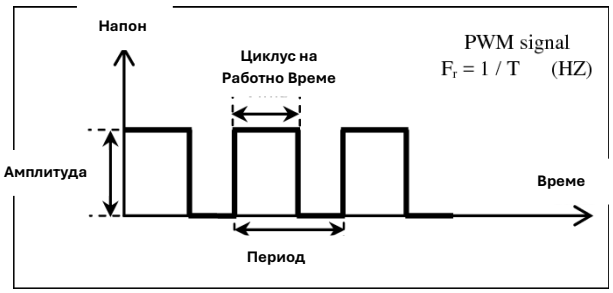
ESC и управување на DC мотор

Во стандардните системи за управување користиме некаков управувачки алгоритам директно врз управуваниот објект, како што во овој случај е DC моторот. Но, со оглед на тоа дека се работи за еднонасочен мотор без четкички, најдобар и најефикасен начин на негово управување е преку PWM сигнал. Тука во игра влегува ESC-то кое ја игра улогата на посредник меѓу управувачкиот сигнал и моторот.. Фактички во нашиот систем, PID алгоритмот ќе праќа вредности до ESC-то, кое потоа ќе произведува сигнал така што го варира работниот циклус на пулсовите кои се праќаат до моторот.

Начин на работа на ESC (Електронски контролер на брзина) : Прима влезен сигнал кој потоа го процесира во својот миктоконтролер (MCU – Microcontroller unit). По процесирањето се генерира PWM сигнал кој се испраќа до управувачките транзистори. Овие транзистори ја модулираат моќноста која се испраќа до соодветниот мотор, со што се овозможува прецизно управување на неговата брзина и насока. Дополнително ESC воведува заштитни механизми кои се се грижат моторот никогаш да не работи во критични случаи и корисникот да биде соодветно информиран за нив преку кратки звучни пораки, кои си имаат свое значење.

Со зголемување на duty cycle се зголемува брзината со која се врти моторот. На пример 100% duty cycle значи дека моторот ќе работи со полна брзина константно, додека при 0% ќе запре комплетно.

Просечниот напон кој се доставува до моторот е пропорционален со работниот циклус на PWM сигналот. Додека пак напонот е пропорционален со вртежниот момент. Тоа значи дека поголем работен циклус резултира со поголем напон до моторот, а тоа дава поголем вртежен момент, односно побрзо вртење на моторот.

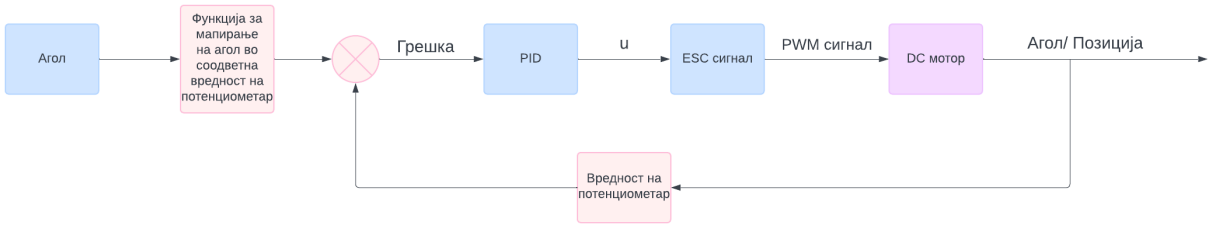


Слика 19 График на PWM сигнал

Дизајн на Arduino Uno програмата

Arduino Uno во нашиот систем е микроуправувачот кој го управува системот. Затоа е потребно да се осмисли алгоритам кој навремено ќе ја чита состојбата на потенциометарот. Овој алгоритам треба да ја преведува состојбата на потенциометарот во агол и да испраќа сигнал до ESC-то за тоа колкава треба да биде неговата PWM вредност до еднонасочниот мотор без четкички. За таа цел ќе го користиме PID алгоритмот за управување и регулација на системот до негово стабилизирање.

Најпрво ќе ја поставиме блок шемата на системот:



Слика 20 Блок шема на системот

Компоненти:

1. **Поставување на агол (Setpoint)**: Саканиот агол кој треба да го достигне системот.
2. **Функција која ќе направи конверзија на аголот во вредност која е соодветна на потенциометарот**
3. **PID управувач**: Управувачот кој ја процесира грешката помеѓу поставениот агол и повратната врска, и испраќа контролен сигнал за да се достигне саканиот поставен влезен агол.
4. **ESC (Electronic Speed Controller)**: Управувач на брзината која се регулира преку напонот на еднонасочниот мотор.
5. **Еднонасочен Мотор**: Моторот во системот кој го подига кракот.
6. **Потенциометар**: Сензорот кој дава повратна информација за тековниот агол на системот.

Целовкупната програма се состои од неколку етапи.

Најпрво се дефинираат сите променливи кои се потребни низ текот на целата програма во соодветен формат. На вредностите на PID константите се поставуваат почетни вредности, кои беа експериментално одредени, а при нив PID алгоритмот работи онака како што е посакувано, односно балансира на поставениот влезен агол со задоволителна стабилност. Потоа е дефинирана функцијата којашто ќе прави конверзија агол – вредност на потенциометар. Таа се користи секогаш кога ќе сакаме да внесеме агол од графичкиот интерфејс, бидејќи е потребно тој да се конвертира во вредност која ја разбира потенциометарот за да знае на која позиција се наоѓа прачката и да прати повратна информација до микроуправувачот и PID алгоритмот. Дополнително, имаме и инверзна функција, која моменталната вредост на потенциометарот, која ја претставува положбата на прачката, ќе ја мапира во соодветниот агол. Постои и функција за стопирање, која е ставена од безбедносни причини. Таа ќе овозможи при стопирање на системот да не настане нагло запирање на моторот и со паѓање на лостот, туку ESC степенасто ќе го намалува PWM сигналот кој ќе го испраќа до DC моторот. Иако постои PID Arduino библиотека, во овој систем рачно беше напишана PID функцијата која што ќе ги пресметува сите три PID компоненти и грешката и соодветно ќе дава сигнали кон ESC-то. Бидејќи Arduino програмата е нужно да е во постојана комуникација со GUI-то имаме и функција која што ќе ја регулира оваа комуникација. Arduino програмата ќе ја препознава соодветната наредба која му ја праќа GUI-то преку совпаѓање на низи од карактери и ќе враќа соодветна реакција. Со тоа овозможуваме три сценарија:

1. Да се прати наредба “Start” од GUI што ќе резултира со почеток на PID алгоритмот и моторот со почетни вредности.
2. Да се прати наредба “Stop” од GUI што ќе резултира со престанок на моторот, но преку функцијата која ќе обезбеди негово постепено исклучување.
3. Да се прати наредба “GetStatus” од GUI што ќе резултира со испраќање на моменталните вредности на сите три PID константи и поставениот влезен агол.
4. Да се прати наредба “GetError” од GUI што ќе резултира со праќање на моменталната грешка пресметана во агли.
5. Да се прати наредби “SetKp” , “SetKi” , “SetKd” , “SetAngle” од GUI што ќе резултира со поставување на нови вредности во PID компонентите или нов посакуван агол на кој ќе настојува системот да балансира.

Во loop делот од програмата се проверува дали сериската порта праќа некакви команди кои стигаат од GUI и соодветно се спроведуваат. Доколку се добие повратна инфромација дека програмата е стартувана се влегува во PID алгоритмот.

Битно е да се каже дека системот е нестабилен додека е отворен, затоа 2е потребна повратната врска од потенциометарот која го стабилизира.

PID алгоритам

Пропорционално-интегрално-диференцирачкиот (PID) управувач е најкористениот управувачки алгоритам во системите со повратна врска. Проценките се дека околу 95% од управувањето на системите со повратна врска, на ниско ниво, во процесите од индустриската автоматика се одвиваат со PID управувач. Самото име е акроним од математичките операции од кој е составен овој управувач со цел генерирање на управувачки сигнал. Овој сигнал се сигналот на грешка, која се дефинира како разликата меѓу посакуваната вредност и измерената вредност. Притоа, управувачкиот ПИД сигнал се пресметува како сума од три поединечни фактори кои зависат од тековната грешка, минатите вредности на грешката и изводот од грешката.

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad ,$$

Каде со *e(t)* се обележува грешката која всушност е разликата во тој временски момент од посакуваната вредност и измерената вредност, со K_p е означен коефициентот на пропорционалната компонента, со K_i е означен коефициентот за интегралната компонента, K_d е коефициентот за дифенцирачката компонента, τ е променливата по која се интерира и е во опсегот од 0 до t.

Во нашиот случај, PID управувачот управува со еднонасочниот мотор преку ESC, споредувајќи ја поставената вредност (агол) со реалната вредност добиена од потенциометарот (повратната врска).

Компоненти на PID алгоритмот

1. Пропорционална (P) компонента – произведува сигнал кој ќе биде пропорционален на влезниот, со што ќе внесе негово слабеење или зголемување. Помага за намалување на вкупната грешка на системот со внесување на корекција која е директно пропорционална на грешката. Оваа компонента се карактеризира преку пропорционалното засилување K_p, кое преставува бројна вредност.

$$P = K_p\, e(t)$$

2. Интегрална (I) компонента – ја пресметува акумулацијата на грешки со текот на времето. Претставува сума од минатите грешки и помага за да се остстрани некоја заостаната грешка која пропорционалната P компонента не може сама да ја отстрани. Оваа компонента се карактеризира преку интегралното засилување K_i. Во овој труд ќе ја користиме дискретната вредност на PID алгоритмот, па затоа соодветно интегралот во интегралната компонента преминува во сума.

$$I = K_i \sum e(t)$$

3. Диференцијална (D) компонента – ја предвидува тенденцијата на грешката (да се зголемува или намалува) врз база на нејзината стапка на промена. Тоа помага за намалување на прескокот во системот и внесените осцилации доколку има такви. Оваа компонента се карактеризира преку интегралното засилување K_d.

$$D = K_d \frac{de}{dt}$$

⇨ Од тука ја добиваме равенката на PID управувачот: $PID = P + I + D$

Откако ќе се постави PID алгоритмот, треба да се нагодат сите три параметри за соодветно работење на системот. Целта на овој систем е да овозможи прачката да стои на посакуваниот агол кој ќе биде внесен од корисникот, а при внесување на некое нарушување што побрзо и ефикасно да се врати на посакуваната аголна положба.

Одзив	Време на искачување	Прескок	Време на смирување	Грешка во стац.состојба
k_p	Намалува	Зголемува	Мала промена	Намалува
k_I	Намалува	Зголемува	Зголемува	Елиминира
k_D	Мала промена	Намалува	Намалува	Мала промена

Слика 21 Влијанија на PID параметрите врз различни параметри во системите

Потребно е да го познаваме ефектите на секој параметар врз одзивот.

- K_p ја зголемува брзината на одзивот, но може да внесе прескок и осцилации на сметка на тоа.
- K_i ја намалува грешката во стационарна состојба, но доколку е превисока воведува преспор одзив и може да предизвика нестабилност.
- K_d го намалува прескокот и ја зголемува стабилноста, но доколку е преголема може да внесе засилување на шумот во системот.

Постојат повеќе начини за нагодување на PID параметрите.

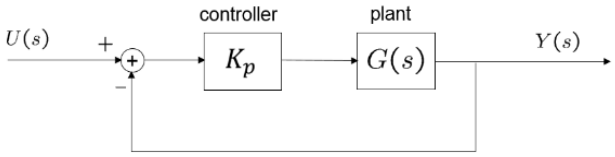
Мануелно нагодување

Мануелното нагодување е преку проба и грешка додека се постигне посакуваниот одзив.

Нагодувањето се започнува така што се поставуваат K_i и K_d на нула, а K_pсе зголемува се додека не се појават осцилации во затворениот систем. Потоа се зголемува Ki се додека не

се елиминира стационарната грешка и на крај се зголемува K_p се додека не се елиминираат осцилациите и максимизира стабилноста на системот.

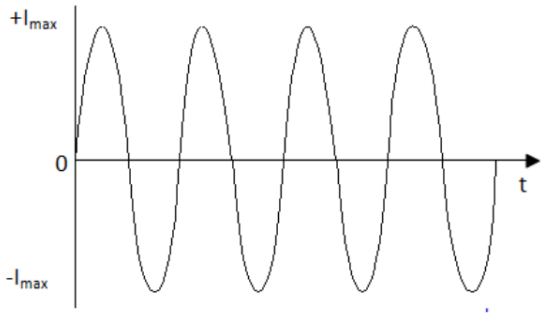
Методот на Зиглер Николс



Слика 22 Блок дијаграм на управуван систем

1.1 Метод на континуиран (затворен) циклус – се врши врз затвореният систем

– се поставуваат K_i и K_d на нула, а K_p се зголемува се додека системот не добие критично периодично поведение, односно осцилира со константна амплитуда.



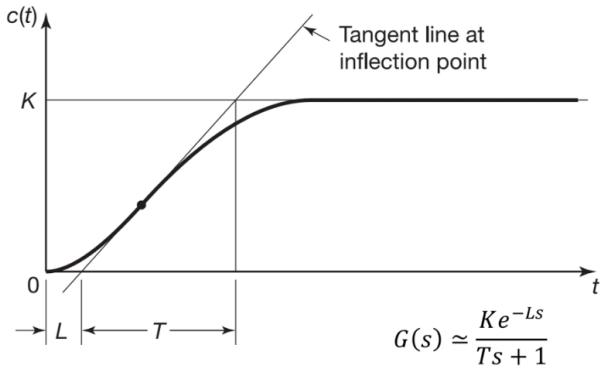
Слика 23 Одзив на систем при критично засилување

Потоа при ваквиот одзив се зема засилувањето K_{cr} и периодот на осцилации T_u и се пресметуваат PID компонентите според следната табела:

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$P_{cr}/1.2$	0
PID	$0.6K_{cr}$	$P_{cr}/2$	$P_{cr}/8$

Слика 24 Равенки за пресметка на PID според Зиглер Николс - метод на засилување

1.2 Метод на реакција на крива (отворен циклус) – преку графикот на кривата на одзивот ќе се види времето на доцнење L и време на пораст T и преку емпириските формули ќе се пресметаат PID параметрите.



Слика 25 Крива при која се пресметуваат PID параметрите на метод на реакција на крива

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{KL}$	∞	0
PI	$\frac{0.9T}{KL}$	$3.3L$	0
PID	$1.2\frac{T}{KL}$	$2L$	$0.5L$

Слика 26 Равенки за пресметка на PID според Зиглер Николс - метод на реакција на крива

$$K_i = \frac{Kp}{\mathrm{T}_i}, \qquad K_d = K_p T_d,$$

$$P = K_p e(t),\ I = K_i \sum e(t),\ D = K_d \frac{de}{dt},$$

$$PID = P + I + D$$

Секако постојат и други техники и методи на нагодување на параметрите. Но, во овој случај се користи емпириските метод на проба и грешка. По поголем број на итерации беа постигнати вредности за PID параметерите со кои задоволително се управува со лостот.

Усвоени вредности на параметрите се:

$$K_p = 0.04,\ K_i = 0.005 \text{ и } K_d = 0.1$$

Овој макета - проект е совршена алатка која може да се употребува во многу предмети каде што се изучува PID алгоритмот и неговите примени. Затоа како следен чекор е изработка на графички интерфејс кој овозможува менување на PID параметрите во реално време и обезедува приказ на нивното влијание врз стабилноста, брзината и прецизноста на одзивот на системот.

Типови на поведение

Системите може да се карактеризииираат со различен тип на одзив врз основа на неколку клучни параметри кои ја диктираат положбата на половите на затворениот систем. Станува збор за параметрите параметрите: фактор на релативно придушување и природна фрекфенција на осцилирање кои се директна последица на моделирањето на системите со преносна функција од втор ред, од обликот:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

Врз основа на ζ и ω_n може да се реализираат неколку видови:

1. Придушен апериодичен одзив - се карактеризира со чисто реални и различни полови, нема осцилации и има превојна точка.

$$s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}, \varepsilon > 1$$

2. Придушен осцилаторен одзив – се карактеризира со комплексно конјугирани полови, има осцилации кои се придушуваат со тек на времето.

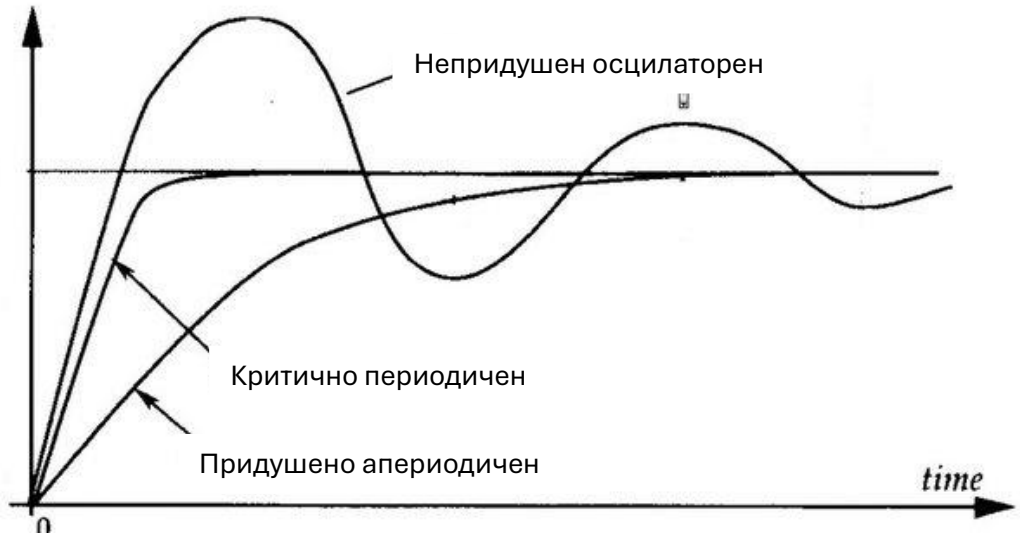
$$s_{1,2} = -\zeta\omega_n \pm \mathrm{j}\omega_n\sqrt{1 - \zeta^2}\,, 0 < \varepsilon < 1\ \omega_d = \omega_n\sqrt{1 - \varepsilon^2}$$

3. Непридушен осцилаторен одзив – се карактеризира со два чисто имагинарни пола и осцилациите никогаш не исчезнуваат.

$$s_{1,2} = -j\omega_n\ , \varepsilon = 0$$

4. Критично придушен одзив – се карактеризира со два еднакви реални пола, кои тргнуваат од различни локации и при нивно спојување се одметнуваат и стануваат комплексно конјугирани. Точката на нивен спој е всушност границата меѓу придушено апериодично поведение и придушено осцилаторно поведение.

$$s_{1,2} = -\delta, \varepsilon = 1$$



Слика 27 Графички приказ на различни поведенија

Начини на сериска комуникација

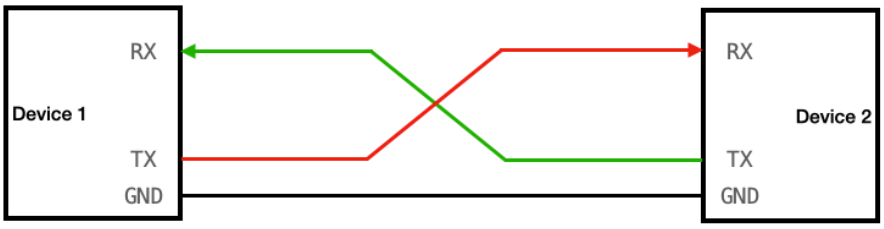
Сериска комуникација подразбира пренесување на податоци бит по бит преку канал за комуникација (жица). Претставува ефикасен и добар начин за пренос на податоци, собено кога треба да се праќаат на поголема далечина.

UART (Universal Asynchronous Receiver – Transmitter)

Еден од наједноставните и најкористени протоколи за комуникација. Користи 2 транспортни линии TX (transmit) – пренос и RX (receive) – приемник. Претставува асинхрон начин на пренос на податоци. Тоа значи дека не е потребен такт сигнал, туку и приемникот и испраќачот се договараат на број на испратени бити во секунда (bits per second).

Податоците се испраќаат во рамки, во кои има почетен бит (start bit), 5 – 9 бити за податоци , опционален бит за парност (parity bit) и еден или повеќе стоп бити (stop bit).

Грешките се детектираат преку битот за парност.

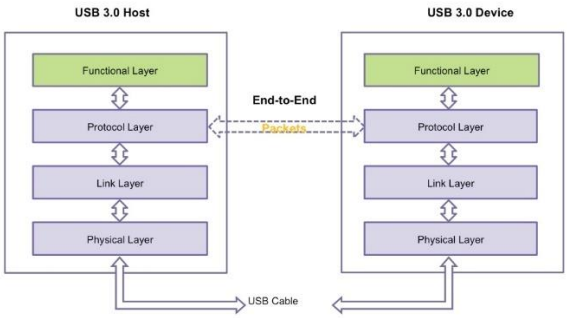


Слика 28 UART протокол за сериска комуникација

USB (Universal Serial Bus)

Протокол од широка употреба кој се користи бидејќи поддржува поголема стапка на пренос на податоци и овозможува напојување на конектираните уреди.

Има можност за plug - and – play – односно да се поврзат уредите без да има потреба да се рестартираат, а исто така и hot – swapping што овозможува да се конектираат и дисконектираат уреди без да треба да се пали и гаси носечкиот систем. Поддржува различни брзини на пренос – Low, High и SuperSpeed.

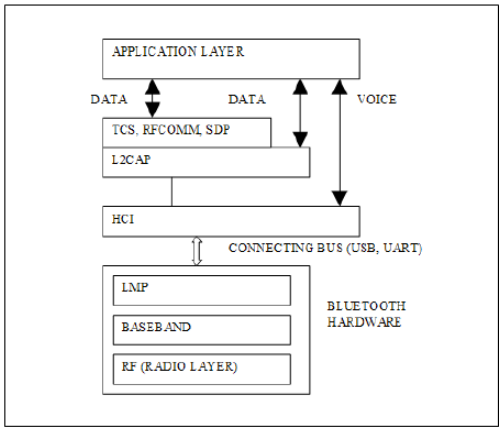


Слика 29 Преглед на USB протоколот

Bluetooth

Претставува бежична технологија за пренос на податоци на кратко растојание преку UHF радио бранови со кратка бранова должина.

Има можност и за Ad-Hoc поврзување.



Слика 30 Протоколен преглед на Bluetooth

Во нашиот проект го користиме UART протоколот преку USB порта, со што овозможуваме асинхрона сериска комуникација помеѓу макетата и графичкиот интерфејс. Но, доколку размислуваме за бежичан начин на комуникација, како опција која ја овозможува моменталната конфигурација и употреба на микроуправувачот Arduino Uno е поставување на Bluegiga WT11 модул.

Слика 31 Bluegiga WT11 модул

Графички интерфејс

Потребата од графички интерфејс за оваа макета да може да се користи во демонстративни цели нè наведе на изработка на Windows Forms апликација.

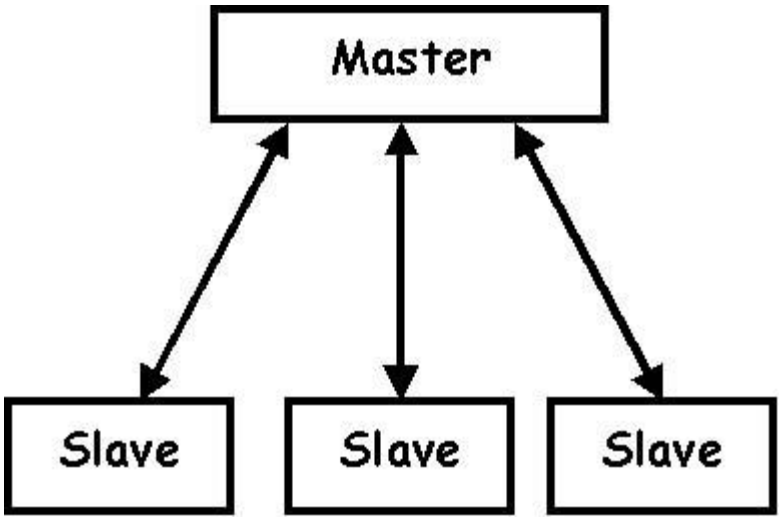
Клучни делови на апликацијата од кои се состои:

- Копче за скенирање на активни USB порти
- Копче за конектирање, односно овомножување на сериска комуникација со избраната USB порта
- Копче за старт / стоп на макетата
- Места за внес на PID параметрите и соодветно копчиња за нивно испраќање до Arduino-то
- Место за внес на посакуван агол на кој ќе балансира прачката и соодветно копче за негово испраќање до Arduino-то
- График кој ќе ја исцртува грешката во реално време
- Лог на комуникацијата помеѓу GUI и Arduino Uno

Слика 32 Приказ на графичкиот интерфејс

Начин на работа

Најпрво потребно е да се осигураме дека и Arduino-то и GUI-то праќаат податоци со иста брзина. Затоа се посветуваме да параметарот baud rate кој ни претставува број на пратени промени на сигнали или симболи во секунда. Кај сериска комуникација најчесто се зема baud rate да е иста брзина колку брзината на трансмисија на бити во секунда. Во овој случај избираме и двете страни, кои играат master - slave сценарио да се поставени на baudrate = 9600. Битно е да се постави оваа master - slave комуникација затоа што графичкиот интерфејс ни дозволува целосно управување на микроуправувачот на многу поедноставен начин, без потреба од целосно препрограмирање на Arduino-то. Дополнително ова дава многу појасна слика за комуникацискиот протокол, како се разменуваат податоците и со тоа се обезбедува навремена и сигурна комуникација без губиток на податоци или колизии на пакети.



Слика 33 Графички приказ на master - slave комуникација

Секое копче или поле кое се наоѓа на екранот има позадинска функција која ја превзема соодветната акција кога ќе е стиснато или на некаков начин иницирано. Најважно низ сите функции ќе ја користиме класата SerialPort која што ќе овозможува сериска комуникација меѓу компјутерот, односно GUI-то и миктроуправувачот, односно макетата и управувањето на моторот.

Соодветно, при клик на ScanPort се пребаруваат активните порти, односно соодветната која ќе овозможи комуникација со Arduino-то.

При клик на копчето Connect ќе иницираме да се овозможи комуникција токму низ избраната порта и со тоа таа ќе се отвори и ќе можеме да примаме и праќаме податоци меѓу графичкиот интерфејс и Микроуправувачот. Кога ќе е остварена комуникацијата копчето Connect се оневозможува, односно затскрива, а се овозможува копчето Disconnect кое ќе ни дозволи потоа да ја прекинеме комуникацијата, односно затвориме портата.

Копчето Start ќе овозможи да започне работата на макетата, односно Arduino ќе испраќа управувачки сигнал до ESC кое потоа со PWM сигнал ќе го контролира BLDC моторот. Програмата на Микроуправувачот е дизајнирана така што започнува со дефаултните вредности, па доколку графичкиот интерфеј испрати команда која нагласува нивна промена во следниот циклус ќе ги имплементира тие промени. При почеток на работата копчето за почеток се оневозможува, а се појавува копчето Stop. При клик на него макетата ќе запре постепено, што ќе ја осигура сигурноста на корисникот.

Постојат три соодветни прозорчиња за внес на трите PID константи, K_p, K_d и K_i, како и посакуван референтен агол Desired angle на кој системот би се балансирал. До секое од овие input полиња, кои примаат вредност која се зачувува како низа од карактери, постои копче Send за нивно испраќање до Arduino.

На графичкиот интерфејс се забележува и график Chart, кој ќе ја изцртува кривата на грешка. Функцијата во позадина на графикот работи така што GUI испраќа наредба до

Микроуправувачот да му ја врати грешката, која што при повратната информација ќе биде изразена во агли од страна на Arduino. Откакао ќе ја прими повратната вредност за состојбата на грешката функцијата го исцртува графикот. Во програмата постои тајмер кој ќе се погрижи оваа меѓусебна комуникација да е синхронизирана и исцртувањето глатко.

Под графикот имаме динамичко поле Status: (соодветна порака) . Соодветната порака ќе се прикажува при различни ситуации. Пример при успешно конектирање ќе добиеме „Connected“ или пак при почеток на програмата ќе добиеме “PID started”. Ова ни овозможува да го пратиме текот на разменетите информации и тоа дали стигнале соодветно. Ваквите пораки имаат за цел да помогнат во откривање на грешки при комуникација и нивно полесно отстранување.

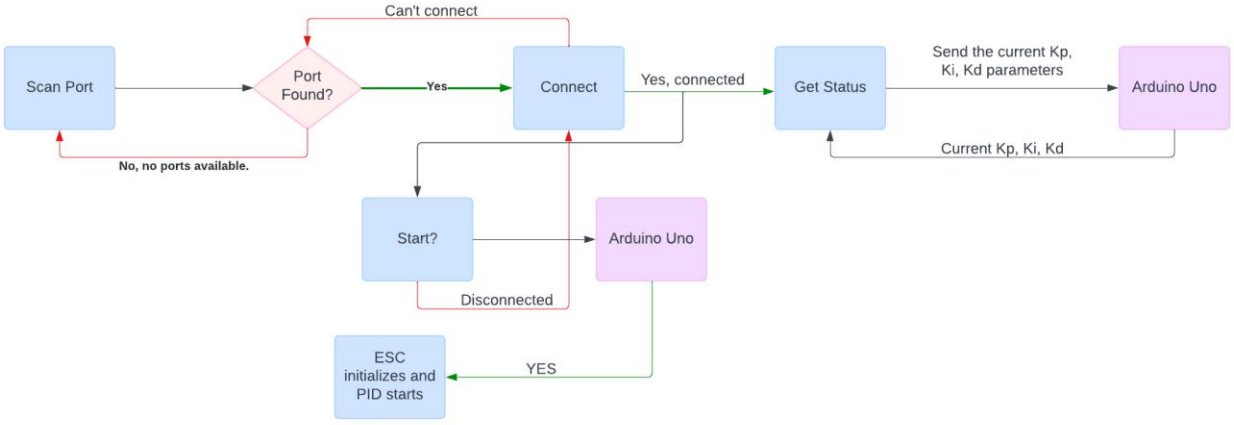
За крај имаме и прозорче за Log, кое ќе ни ја прикажува комуникацијата помеѓу GUI и Arduino. Неговата намена е стриктно за директен приказ на разменетите команди и поголемо разбирање на комуникацијата меѓу нив и комуникациските протоколи.

За да се долови начинот на комуникација и испраќање на податоците и сигналите помеѓу микроуправувачот и корисничкиот графички интерфејс ќе се разгледаат следните две сценарија.

Сценарио 1:

Се иницира конекција така што се скенира за активни порти и доколку се пронајде соодветната, се конектираат. Назад веднаш се добиваат информации за моменталните вредности на PID константите и поставениот агол на кој балансираме. Доколку конекцијата не е успешна, ќе се добие соодветна порака зошто, па ќе може да се превземат соодветни чекори.

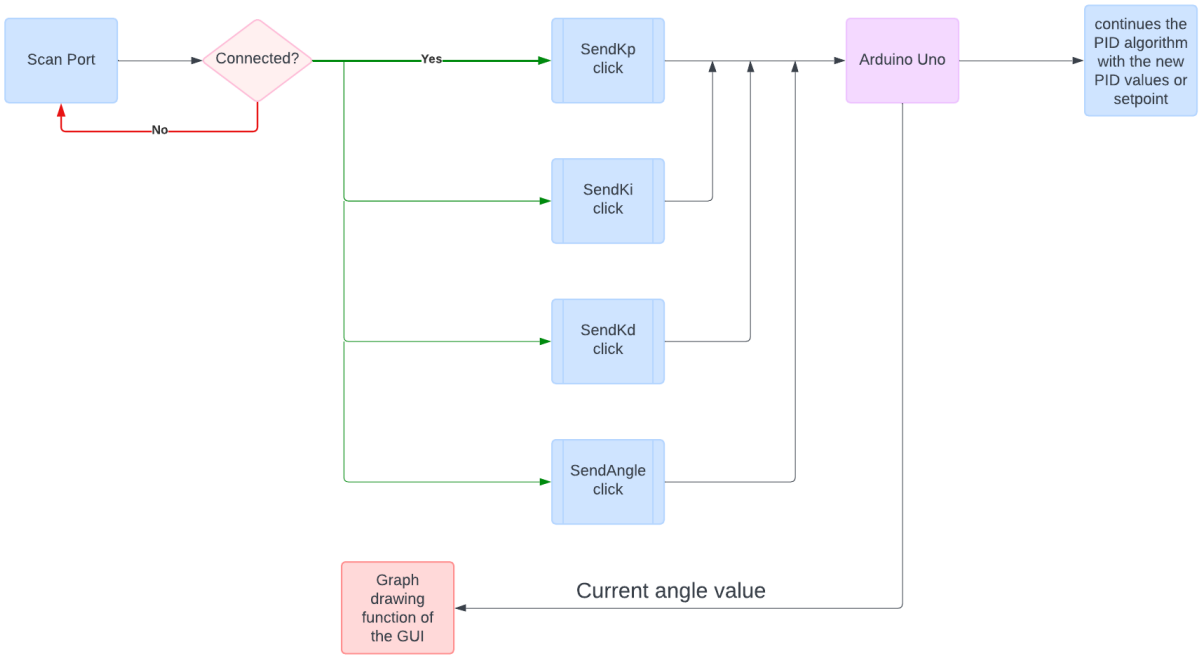
Ако се конектирани, може да се притисне копчето Start кое ќе го започне процесот на PID управување. Доколку се случи некаква грешка и не може да започне процесот, ќе се добие соодветна порака.



Слика 34 Flowchart претстава на сценарио 1

Сценарио 2:

Доколку конекцијата е воспоставена, има можност да се испраќаат наредби до Arduino за промена на параметрите. Промената ќе биде применета од самиот систем при наредниот скенирачки циклус, што е всушност период на земање примероци.

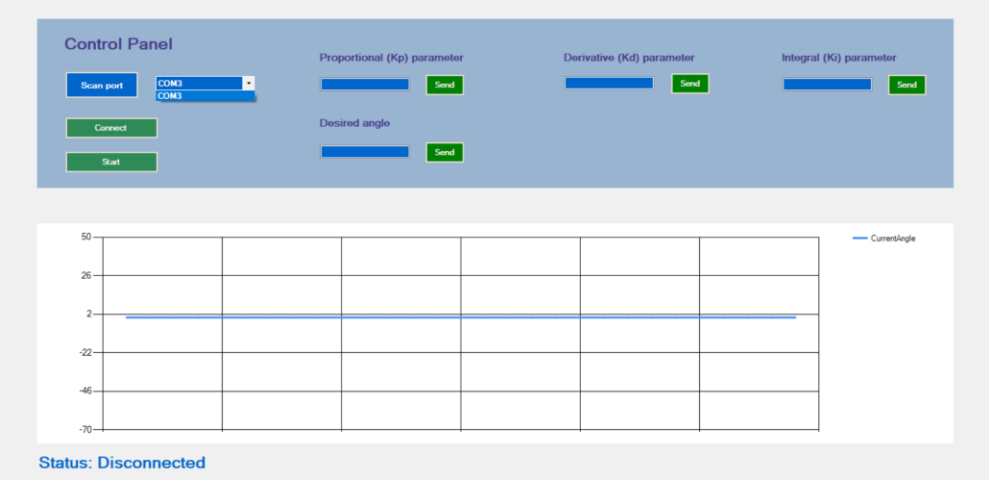


Слика 35 Flowchart претстава на сценарио 2

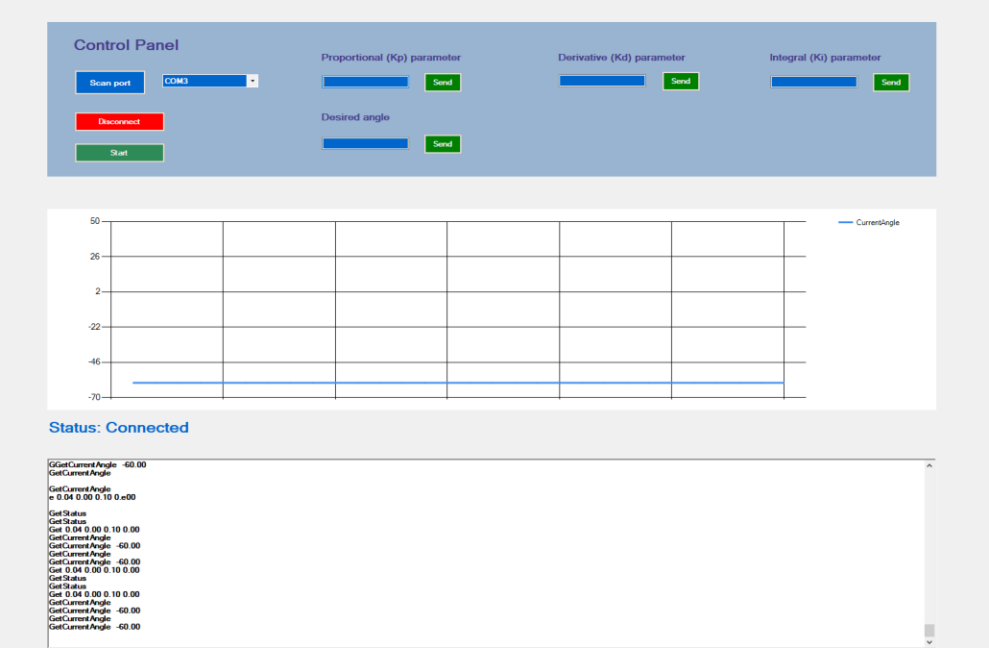
Демонстрација на работата на системот

Со демонстрацијата ќе се претставуваат ефикасноста на управувањето и неговата способност да го балансира лостот на поставениот влезен агол. Се разгледуваат неколку различни сценарија при што, најпрво почетната состојба во која се јавува макетата при нејзино вклучување. Потоа ќе внесеме различни промени преку GUI-то и на крај ќе внесеме и физички нарушувања што ќе предизвикаат шум.

Сценарио 1 – За почеток е потребно да се конектираат GUI и Arduino Uno. Се бара соодветната порта и притиска копчетко „Connect“. По успешна конекција може да се премине на некое од сценаријата. Потоа се притиска копчето „Start“.



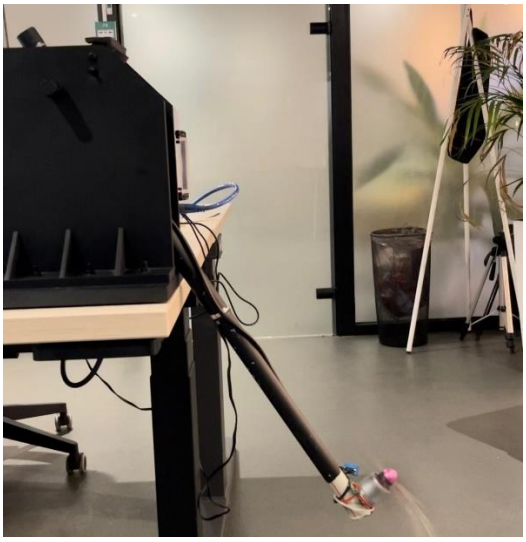
Слика 36 Барање на соодветна порта



Слика 37 Демонстрација на успешно конектирање

Макетата започнува да работи со своите пред дефинирани вредности, односно ќе балансира на позиција 0 степени, со PID константи $K_p = 0.04$, $K_i = 0.005$ и $K_d = 0.1$.

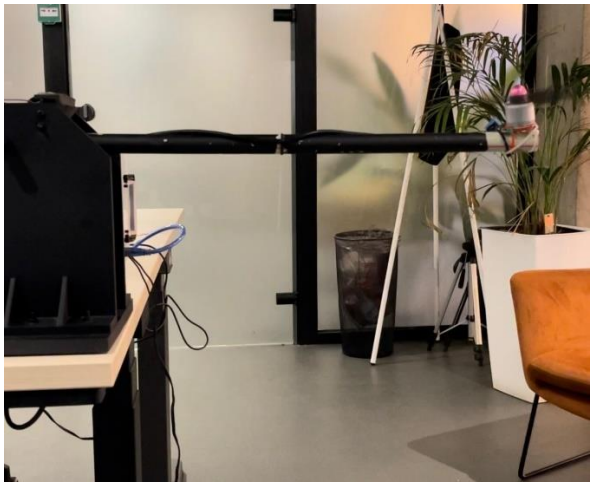
На слика 37 е пикажана почетната положба при самото иницирање на ESC-то со кое тоа дава PWM сигнал до моторот на минимален вртежен момент. Слика 38 го покажува почетното придвижување на прачката. Слика 39 го илустрира придвижувањето до стационарната положба, без да има осцилации во системот, а слика 40 го прикажува достигнувањето на истата.



Слика 38 Почетна положба на лостот

Слика 39 Приказ на придвижување на лостот

Слика 40 Приближување до стационарна положба



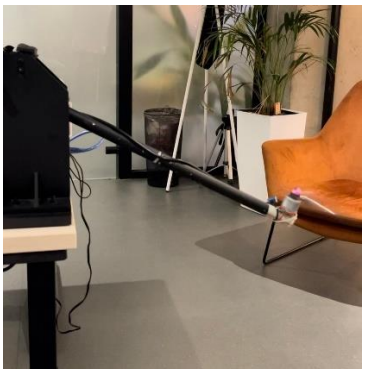
Слика 40 Лостот ја достигнува поставената положба

Слика 40 го прикажува графичкиот интерфејс и исцртувањето на кривата на моменталната положба на прачката, односно аголната положба од притискање на копчето „Start“ до достигнување на стационарната положба од 0 степени на прачката.

Слика 41 Симулациски приказ на моменталниот агол на кој се наоѓа лостот

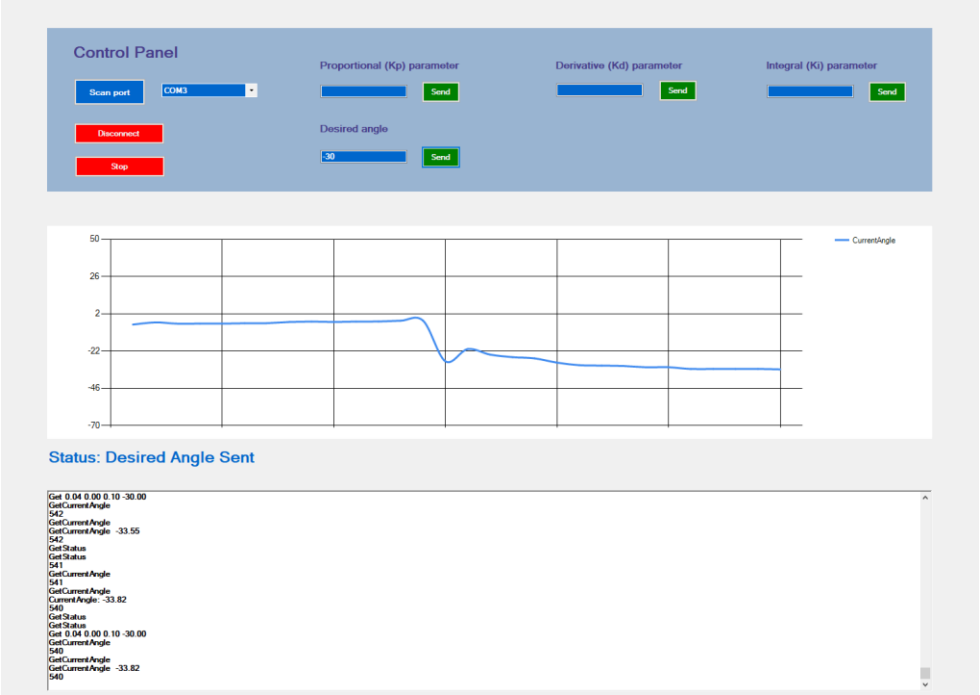
Сценарио 2 – со внесување на различна сакана положба, односно агол на кој би требало да балансира лостот, ќе резултира со промена на управувачкиот сигнал. Во зависност од тоа дали ќе поставиме поголем или помал агол, ESC-то ќе испраќа подолг или пократок PWM сигнал до DC моторот.

Слика 41 ја претставува физичката претстава на промената на аголот, односно неговото спуштање за -30 степени во однос на референтниот нулти агол.



Слика 42 Макетата поставена на -30 степени

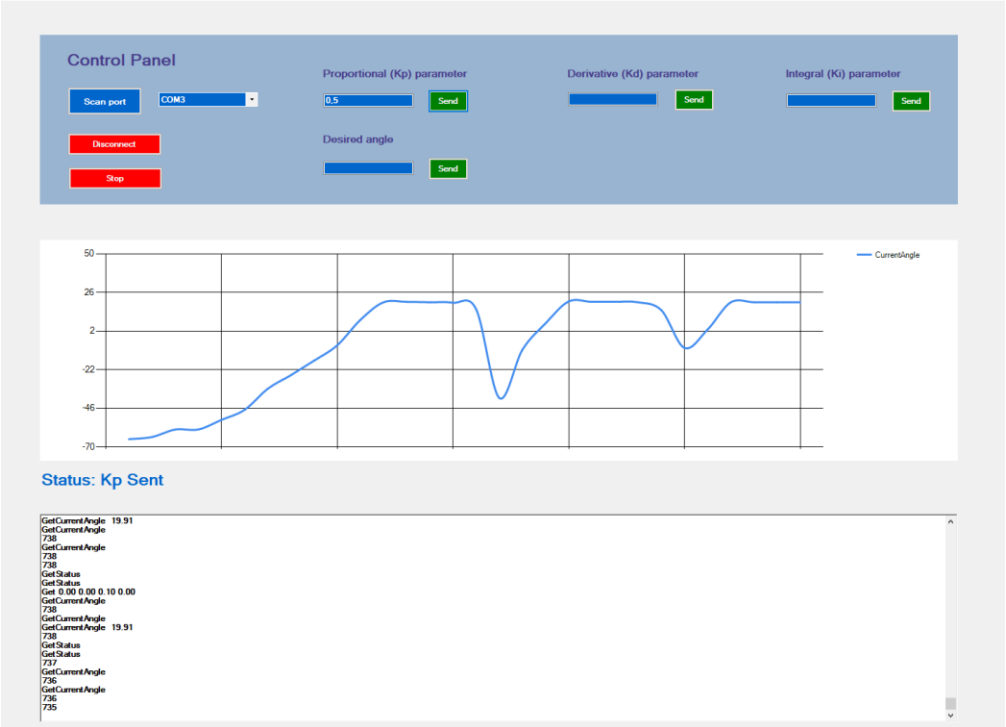
На слика 42 се забележува дека по внесување на промената од почетент агол 0 степени на на новиот внесен агол од -30 степени, ESC-то го намалува PWM сигналот и системот по краток временски период го управува мотрот на новата посакувана положба.



Слика 43 Симулациски приказ на промена на аголот

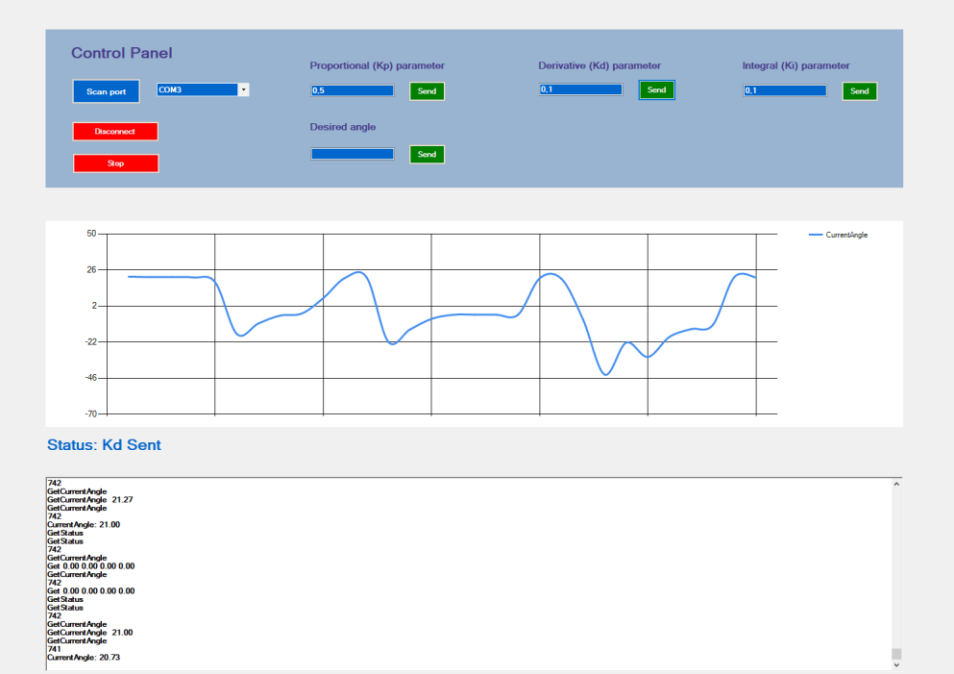
Сценарио 3 - се разгледува случајот на промена на PID параметрите.

Најпрво се зголеми пропорционалната константа K_p и веднаш беа забележани осцилациите од слика 43.



Слика 44 Симулациски приказ на промена на пропорционалната константа

Потоа со промена и на останатите константи, сепак осцилаторното поведение остана и не беше остварена нова стационарна положба. Симулацискиот приказ на поведението е прикажан на слика 44.



Слика 45 Осцилаторно поведение на системот при променети PID параметри

Заклучок

Во рамките на овој труд беше спроведено дизајнирање и имплементација на систем, составен од еднокрак лост – прачка чијашто аголна позиција сеуправува со PID алгоритам преку графички интерфејс. Со овој дипломски труд ја истакнуваме важноста на интегрирањето на теориските модели во практични експериментални примери при нивно изучување, за да се постигне сеопфатно разбирање на динамиката на системот како и самите управувачки алгоритми.

Процесот на развој и тестирање на физичкиот модел, како и софтверските аспекти за негово управување се состоеа од неколку клучни точки:

1. Истакнување на важноста на физичките модели – Конструкција и користење на физичка макета која обезбеди важни сознанија за системите, кои не можат да се согледаат комплетно преку симулациони средини. Факторите од реалниот свет, како што се механичко триење, шум, немоделирана динамика, физички оштетувања, беа подобро нагласени и разбрани, а со тоа и решени преку практично експериментирање.
2. Ефикасност на PID управување – PID алгоритмот се покажа како ефикасен начин на управување на овој систем, нудејќи прецизна регулација на положбата на прачката. Со нагодување на PID константите K_p, K_d и K_i ги забележавме различните однесувања на системот, покажувајќи ја флексибилноста и робусноста на ваквото управување и неговото влијание врз стабилноста и одзивот на системот.
3. Улога на графички кориснички интерфејс (GUI) – Неговата имплементација значително ја подобри употребливоста на системот за демонстративни цели. Градењето на интуитивен интерфејс овозможи прилагодување во реално време на PID параметрите и како и нивно следење, така и нивните влијание врз системот преку график. Со ова овозможивме подобро разбирање на нивните директни влијанија на перформансите на системот.
4. Образовна вредност: Проектот ќе биде одлична едукативна алатка, одврзувајќи го јазолот меѓу теоретските концепти и нивно практично применување. Ќе им служи на студентите како практично и опипливо средство за истражување и експериментирање со управувачките алгоритми. Со тоа ќе го зајакнат своето теоретско знаење преку непосредна визуелна информација.
5. Идни подобрувања: Проектот отвора врата за многу идни подобрувања, како што се воведување дополнителни безбедносни мерки, како заштита околу перката, која беше првично моделирана. Потоа подобрување на прецизноста на системот за повратни информации со што ќе може да се подобрува и PID управувањето. А воедно отвора и простор за имплементација и на други видови на управувања и нивни детални изучувања и нивните директни влијанија во системот. Како крајна точка, се остава простор и за физички подобрувања како интеграција на дополнителни компонени и сензори, обезбедувајќи негова приспособливост на различни експериментални постапки. Како предлог решение е воведување на Raspberry Pi, кој преносен компактен компјутер и ќе овозможи да биде сервер во системот. Ова е важно затоа

што ќе се отвори можноста за бежично поврзување во системот, отфрлајќи ја потребата од сериска комуникација.

Како заклучок, овој проект ја демонстрираше важноста од комбинацијата на теоријското знаење со негова практична имплементација. Детално ја објаснивме ефикасноста на PID управувањето, кое и покрај неговата едноставност може да претставува солидна подлога за идни едукативни истражувања и развивања во полето на автоматското управување. Преку континуирано експериментирање и усовршување, таквите системи може да се оптимизираат и развијат во апликации во реалниот свет, придонесувајќи за напредок на индустриската автоматика, роботиката, Internet of Things (IoT) и пошироко.

Референци

[1] Turnigy Multistar ESCs: https://turnigy.com/essential_grid/turnigy-multistar-escs/

[2] What are Brushless DC Motors: https://www.renesas.com/us/en/support/engineer-school/brushless-dc-motor-01-overview

[3] Arduin, Megan, "Low-Cost Control Engineering Experiments" (2017). Honors Theses. 2903.

[4] Norman S. Nise, “Control System Engineering”, 6th edition

[5] James F. Kurose and Keith W. Ross, “Computer Networks – A top down approach”, 6th edition

[6] Миле Станковски и Татјана Колемишевска – Гугуловска, Компјутерско Водење на Процеси

[7] GUI tutorial: https://github.com/fahimabrar353/ArduinoGUIVisualStudio

Додаток

Код имплементиран во Arduino Uno:

```
#include <Servo.h>
#include <math.h>

Servo esc;
float desired_angle = 0.0;
float setpoint = 0.0;
float pid_p = 0.0, pid_i = 0.0, pid_d = 0.0;
float kp = 0.04;
float ki = 0.005;
float kd = 0.1;
float sum = 1000.0;
float error = 0.0;
float angle_gui = -60.0;
float previous_error = 0.0;
float u = 0.0;
bool running = false;
bool dataReceived = false;
unsigned long lastSendTime = 0;     // To track the last time error was sent
unsigned long sendInterval = 1000;  // Interval to send error in milliseconds
//Functionn that does the conversion from the input angle to the coresponding potentiometer value
float convertInput(int input) {
  float a;
  a = (0.0757 * input * input) + (24.394 * input) + 2200;
  return a;
}
float calculateAngle(float x) {
  float y;
  float min = 445;
  float max = 775;
  float range = max - min;
  float current = x- min;
  float perc = current/ range;
  float total = 90;
  float angle = total*perc;
  return -60+angle;
}
void stopESC() {
  for (int i = u; i >= 1000; i -= 10) {
    esc.writeMicroseconds(i);
    delay(150);
  }
```

```
  esc.writeMicroseconds(1000);  // Ensure it stops at the minimum value
}
void pidFunction() {
  int potValue = analogRead(A0);
  Serial.println(potValue);
  float pwm = map(potValue, 446, 788, 1000, 3000);  // Map potentiometer value to PWM range

  error = desired_angle - pwm;

  // Calculate PID terms
  pid_p = kp * error;
  pid_i = sum + ki * error;
  pid_d = kd * (error - previous_error);

  if (u >= 1000 && u < 3000) {
    sum = sum + ki * error;
  }

  u = pid_p + pid_i + pid_d;

  // Ensure PWM values are within bounds
  if (u < 1000) {
    u = 1000;
  }
  if (u > 3000) {
    u = 3000;
  }

  esc.writeMicroseconds(u);
  previous_error = error;
  unsigned long currentTime = millis();
  if (currentTime - lastSendTime >= sendInterval) {
    angle_gui = calculateAngle(potValue);
    Serial.print("CurrentAngle: ");
    Serial.println(angle_gui);
    lastSendTime = currentTime;
  }
}
void processCommand(String command) {
  if (command.startsWith("Start")) {
    running = true;
  } else if (command.startsWith("Stop")) {
    running = false;
    stopESC();
  } else if (command.startsWith("SetKp ")) {
```

```
    kp = command.substring(6).toFloat();
  } else if (command.startsWith("SetKi ")) {
    ki = command.substring(6).toFloat();
  } else if (command.startsWith("SetKd ")) {
    kd = command.substring(6).toFloat();
  } else if (command.startsWith("SetAngle ")) {
    setpoint = command.substring(9).toFloat();
    desired_angle = convertInput(setpoint);
  } else if (command.startsWith("GetStatus")) {
    Serial.print("Get ");
    Serial.print(kp);
    Serial.print(" ");
    Serial.print(ki);
    Serial.print(" ");
    Serial.print(kd);
    Serial.print(" ");
    Serial.println(setpoint);
  } else if (command.startsWith("GetCurrentAngle")) {
    Serial.print("GetCurrentAngle ");
    Serial.print(" ");
    Serial.println(angle_gui);
  }
}
void setup() {
  esc.attach(8);               // Attach servo to pin 8
  esc.writeMicroseconds(1000);  // Initialize servo at 1000 microseconds
  Serial.begin(9600);          // Initialize Serial communication
  desired_angle = convertInput(setpoint);
}
void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    input.trim();  // Remove leading and trailing whitespace

    processCommand(input);
  }
  if (running) {
    pidFunction();
  }
  delay(150);  // Delay for stability
}
```

Кодот имплементиран при изработка на кориснички графички интерфејс може да се пристапи преку следниот линк:

https://github.com/cvetkovskamelani/Diplomska_GUI.git