НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____2015 р.

Дипломна робота

_____першого (бакалаврського)_____**рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності 7.050103, 8.050103 Системне проектування
(код та назва спеціальності)

на тему: Клієнт-серверна модель керування мікроконтролерними системами.

Виконав: студент IV курсу, групи ДА-12
(шифр групи)

_____Мельничук Микола Михайлович_____ ____
(прізвище, ім'я, по батькові) (підпис)

Керівник _____доцент., к.т.н., н.с., Харченко К.В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант охорона праці доцент, канд. біол. наук Гусєв А. М._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____доцент., к.т.н., Стіренко С.Г._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____ст.. викладач Бритов О.А._____ ____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2015 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський)

Спеціальність 7.050103, 8.050103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2015 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Мельничуку Миколі Михайловичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Клієнт-серверна модель керування мікроконтролерними системами.
керівник проекту (роботи) Харченко Костянтин Васильович, доцент, к.т.н. н.с. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «02» квітня 2015 р. № №30/1-ст

2. Строк подання студентом проекту (роботи) 08.06.2015

3. Вихідні дані до проекту (роботи)
Мікроконтролерна система Arduino Yun, робо-платформа. Датчики: гіроскоп-акселерометр MPU-6050, ультразвуковий датчик відстані HC-SR04. Драйвер двигунів L298N. Середовища розробки NetBeans IDE 8.0.2 та Arduino 1.6.4. Джойстик.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)
1. Огляд мікроконтролерних систем.
2. Програмні засоби створення клієнт-серверної моделі.
3. Клієнт-серверна модель керування мікроконтролерними системами.
4. Використання моделі на прикладі колісного робота.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Схема роботи клієнт-серверної моделі – плакат.
2. Схема підключення периферійних пристроїв Arduino – плакат.
3. Схема процесу керування ПІД-ом – плакат.
4. Комунікативна схема Arduino YUN – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	к.б.н., доц. Гусєв А.М.		

7. Дата видачі завдання 01.02.2015

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2015	
2	Збір інформації	15.02.2015	
3	Вивчення варіантів реалізації та вибір варіанту для розробки	28.02.2015	
4	Вивчення периферійних пристроїв	10.03.2015	
5	Розробка користувацького інтерфейсу	15.03.2015	
6	Розробка веб серверу на основі мікроконтролера	18.04.2015	
7	Розробка плану тестування програми	27.04.2015	
8	Тестування згідно розробленої методи	30.04.2015	
9	Оформлення дипломної роботи	31.05.2015	
10	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2015	

Студент _____ М.М. Мельничук
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ К.В. Харченко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

АНОТАЦІЯ

бакалаврської роботи Мельничука Миколи Михайловича

на тему: «Клієнт-серверна модель керування мікроконтролерними системами»

Дипломна робота присвячена дослідженню мікроконтролерних систем та способів їх управління за клієнт-серверною архітектурою. Результатом роботи був створений діючий прототип клієнт-серверної моделі управління платою Arduino YUN. Для демонстрування її роботи був створений колісний робот, який може виконувати ряд функцій на допомогою віддаленого керування. Дану роботу рекомендовано використовувати у якості основи для подальшого вивчення технологій та розробки нових систем.

Загальний обсяг роботи 76 сторінок, з них основна частина – 58 сторінок, 17 рисунків, 11 таблиць, 10 посилань, 1 додаток на 15 сторінок.

Перелік ключових слів: ардуїно, віддалене керування, колісний робот, клієнт-серверна модель, Bridge, Arduino IDE, драйвер двигунів L298N, Arduino YUN.

АННОТАЦИЯ

бакалаврской дипломной работе Мельничука Миколи Михайловича

на тему:« Клиент-серверная модель управления микроконтроллерными системами»

Дипломная работа посвящена исследованию микроконтроллерных систем и способов их управления с клиент-серверной архитектурой. Результатом работы был создан действующий прототип клиент-серверной модели управления платой Arduino YUN. Для демонстрации ее работы был создан колесный робот, который может выполнять ряд функций на помощью удаленного управления. Данную работу рекомендуется использовать в качестве основы для дальнейшего изучения технологий и разработки новых систем.

Общий объем работы 76 страниц, из них основная часть - 58 страниц, 17 рисунков, 11 таблиц, 10 ссылок, 1 приложение на 15 страниц.

Перечень ключевых слов: ардуино, удаленное управление, колесный робот, клиент-серверная модель, Bridge, Arduino IDE, драйвер двигателей L298N, Arduino YUN.

ANNOTATION

for the bachelors work of Melnychuk Mykola Mikhailovich

«The client-server model of microcontroller's systems control»

Diploma work is devoted to the study of microcontroller systems and methods for their control on client-server architecture. There was created a working prototype of a client-server model control of board Arduino YUN. To demonstrate it work there was created wheeled robot that can perform a number of functions via remote control. This work is recommended to be used as a basis for further study of technology and the development of new systems.

The total amount of work 76 pages, the main part - 58 pages, 17 figures, 11 tables, 10 links, one application on 15 pages.

Keyword list: Arduino, remote management, wheeled robot, the client-server model, Bridge, Arduino IDE, driver engines L298N, Arduino YUN.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Arduino - апаратна обчислювальна платформа, основними компонентами якої є плата вводу/виводу та середовище розробки на мові Processing/Wiring.

Arduino YUN - плата мікроконтролера на базі ATmega32u4 і Atheros AR9331. Процесор Atheros підтримує дистрибутив, заснований на OpenWrt. Плата має вбудований Ethernet і Wi-Fi підтримку, в USB-порт, слот для мікро-SD карти, 20 цифрових входу / вихідних контактів.

Сервер - будь-яка система, процес, комп'ютер, які володіють довільним обчислювальним ресурсом (пам'яттю, часом тощо).

Клієнт - будь-яка система, процес, комп'ютер, користувач, які запитують у сервера який-небудь ресурс, які користуються будь-яким ресурсом або обслуговуються сервером іншим способом.

Клієнт-сервер - це вид розподіленої системи, в якій є сервер, що виконує запити клієнта, причому сервер і клієнт спілкуються між собою з використанням того або іншого протоколу.

Скетч – в Ардуіно це програма, написана на мовах C/C++, що завантажується на плату.

ВСТУП

Комп’ютери вже давно стали невід’ємною частиною нашого повсякденного та професійного життя. Та іноді ми забуваємо, що крім них, нас оточують дуже маленькі комп’ютери, і мова йде не про телефони а про мікроконтролери.

Мікроконтролерні системи зараз повсюди, вони у пральних машинах, автомобілях, холодильниках і інших повсякденних речах. Також вони часто використовуються у різних інноваційних пристроях, таких як квадрокоптер, таких систем, як «Розумний будинок» чи, певно, найцікавіших з них – роботах.

Наразі існують певні системи зв’язку з цими пристроями, але більшість з них можуть використовуватись тільки напряму людиною, наприклад, натискаючи кнопки.

В даній роботі пропонується розглянути модель віддаленого керування мікроконтролерними системами, розроблену на основі клієнт-серверної архітектури, яка дає багато нових можливостей для управління пристроями.

Основним матеріалом для дослідження було взято Arduino YUN на основі якого побудовано колісний робот з віддаленим керуванням. Він передбачає декілька режимів роботи, що демонструють можливості цієї моделі управління.

На основі такої моделі можуть бути спроектовані на розроблені десятки пристроїв для полегшення роботи інженерам, лікарям, водіям, будівельникам, інвалідам. Також ця модель може бути використана при створенні «розумного» будинку.

1 ОГЛЯД МІКРОКОНТРОЛЕРНИХ СИСТЕМ

1.1. Мікроконтролерні системи

Мікроконтролер (MCU) - мікросхема, призначена для керування електронними пристроями. Типовий мікроконтролер поєднує в собі функції процесора і периферійних пристроїв, може містити ОЗУ і ПЗУ. По суті, це однокристальний комп'ютер, здатний виконувати прості завдання. Використання однієї мікросхеми, замість цілого набору, як у випадку звичайних процесорів, що застосовуються в персональних комп'ютерах, значно знижує розміри, енергоспоживання і вартість пристроїв, побудованих на базі мікроконтролерів. Мікроконтролери є основою для побудови вбудованих систем, їх можна зустріти в багатьох сучасних приладах, таких, як телефони, пральні машини і т. п.

Якщо представити всі типи сучасних мікроконтролерів (МК), то можна дивуватися величезною кількістю різноманітних приладів цього класу, доступних споживачеві. Проте всі ці пріори можна розділити на наступні основні типи:

- Вбудовувані (embedded) 8-розрядні МК;
- 16 і 32-розрядні МК;
- Цифрові сигнальні процесори.

Промисловістю випускаються дуже широка номенклатура вбудованих МК. У них всі необхідні ресурси (пам'ять, пристрої введення-виведення і т.д.) розташовуються на одному кристалі з процесорним ядром. Якщо подати харчування і тактові імпульси на відповідні входи МК, то можна сказати, що він як би «оживе» і з ним можна буде працювати. Зазвичай МК містять значну кількість допоміжних пристроїв, завдяки чому забезпечується їх включення в

реальну систему з використанням мінімальної кількості додаткових компонентів. До складу цих МК входять:

- Схема початкового запуску процесора (Reset);
- Генератор тактових імпульсів;
- Центральний процесор;
- Пам'ять програм (E(E)PROM) і програмний інтерфейс;
- Засоби введення/виводу даних;
- Таймери, що фіксують кількість командних циклів.

Більш складні вбудовані МК можуть додатково реалізовувати наступні можливості:

- Вбудований монітор/налагодження програм;
- Внутрішні засоби програмування пам'яті програм (ROM);
- Обробка переривань від різних джерел;
- Аналоговий введення/виводу;
- Послідовний ввід/вивід (синхронний та асинхронний);
- Паралельний ввід/вивід (включаючи інтерфейс з комп'ютером);
- Підключення зовнішньої пам'яті (мікропроцесорний режим).

Всі ці можливості значно збільшують гнучкість застосування МК і роблять більш простим процес розробки систем на основі. Деякі МК (особливо 16 - і 32 біт) використовують тільки зовнішню пам'ять, яка включає в себе як пам'ять програм (ROM), так і деякий обсяг пам'яті даних (RAM), необхідний для даного застосування [1]. Вони застосовуються в системах, де потрібен великий обсяг пам'яті і відносне не велика кількість пристроїв (портів введення/виводу. Типовим прикладом застосування такого МК з зовнішньої пам'яттю є контролер жорсткого диску (HDD) з буферної кеш-пам'яттю, який забезпечує проміжне зберігання і розподіл великих обсягів даних (порядку декількох мегабайт). Зовнішня пам'ять дає можливість

такого мікроконтролеру працювати з більш високою швидкістю, ніж вбудований МК.

Цифрові сигнальні процесори (DSP) - відносно нова категорія процесорів. Призначення DSP полягає в тому, щоб отримувати поточні дані від аналогової системи, обробляти дані і формувати відповідний відгук у реальному масштабі часу. Вони зазвичай входять до складу систем, використовуючись як пристроїв керування зовнішнім обладнанням, і не призначені для автономного використання.

1.1.1. NUY від 8Devices

Nuy це шилд, який додає Wi-Fi і Linux на платформу Arduino. Він побудований навколо Carambola2 - провідний промисловий модуль, заснований на Atheros AR9331 SoC. Шилд працює OpenWRT-YUN вбудований дистрибутив, заснований на OpenWRT створений командою Arduino. Це відкритий апаратний продукт виробляється компанією 8Devices. Його гарною стороною є те, що він сумісний з Arduino YUN та може використовували його скетчі. Але цей продукт не набув популярності в Україні, тому його вкрай важко знайти.

Рисунок.1.1 – Зовнішній вигляд плати NUY

Специфікація

Таблиця 1.1 – Специфікація NUY

Wireless module	**Carambola 2 (AR 9331)**
Interfaces	• **Wi-Fi 802.11** • **Ethernet 100 Mbit** • **USB host** • **TTL Serial** • **Standard external antenna connector (U.FL / IPEX)**
Wi-Fi Frequency	**2.4 GHz**
Max output power	**21 dBm**
Wireless standard	**802.11 bgn**
Power supply	**5V, powered from arduino**
Size	**54 by 69 mm**
Software	**OpenWrt – YUN**

1.1.2. Arduino YUN

Arduino Yun є мікроконтролерна плата на базі ATmega32u4 і Atheros AR9331. Процесор Atheros підтримує дистрибутив, заснований на OpenWrt з назвою OpenWrt-YUN.

Рисунок.1.2 – Зовнішній вигляд плати Arduino YUN

Плата має вбудований Ethernet і Wi-Fi адаптери, USB-A порт, слот для мікро-SD карти, 20 цифрових вхідних / вихідних контактів, 16 МГц кристалічний в генератор, мікро USB-з'єднання.

Yun відрізняється від інших плат Arduino в тому, що він може спілкуватися з дистрибутивом Linux на платі, пропонуючи потужний мережевий комп'ютер з легкістю Arduino. На додаток до Linux команд, як cURL, ви можете написати свої власні оболонки і пітоновські скрипти для надійних взаємодій [2].

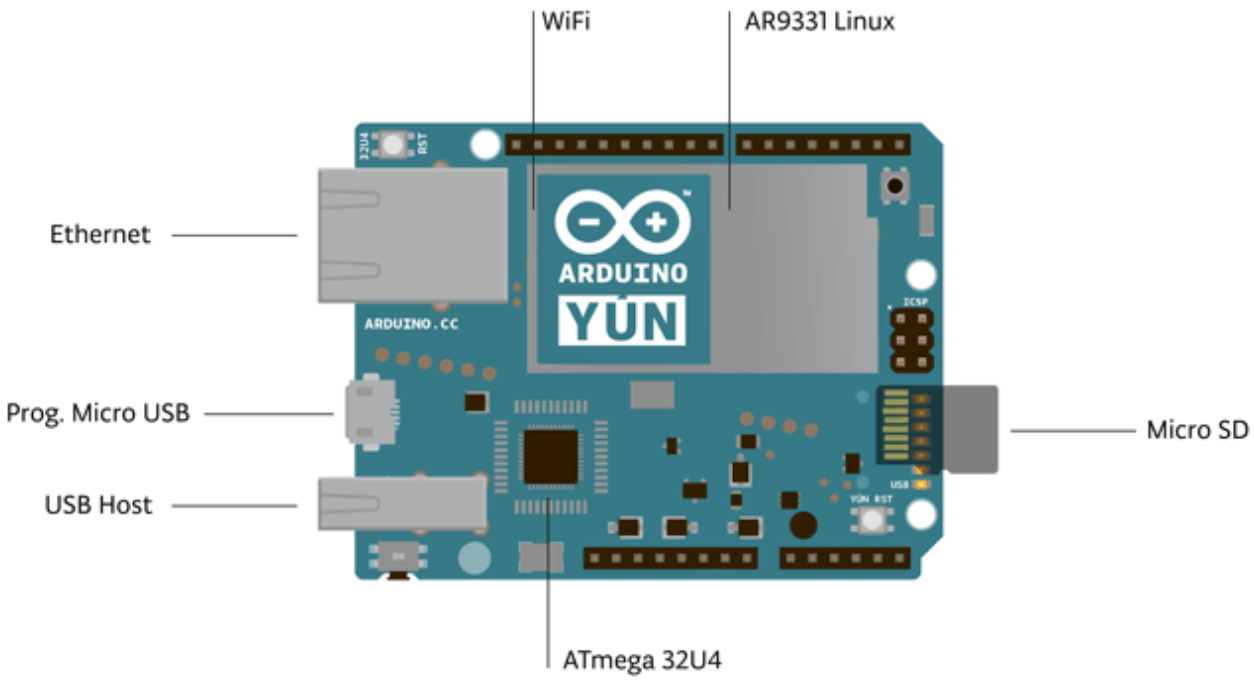

Рисунок.1.3 – Розташування модулів плати Arduino YUN [1]

Yun схожий на Leonardo в тому, що ATmega32u4 має вбудований USB зв'язок, що усуває необхідність другого процесора. Це дозволяє Yun з'являтися

на підключений комп'ютер як миша і клавіатура, та на додаток, до віртуального (CDC) послідовного / COM порта.

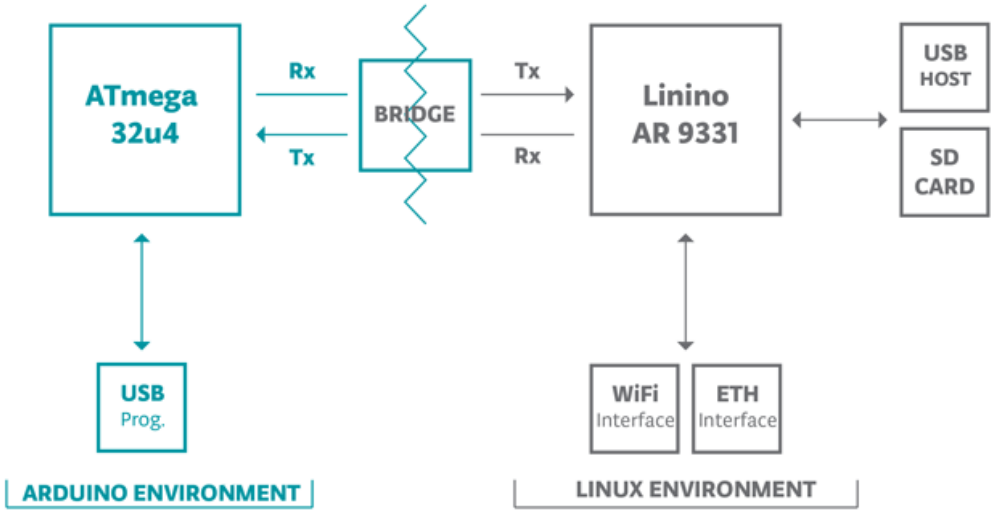

Рисунок.1.4 – Комунікативна схема плати Arduino YUN [2]

Бібліотека Bridge полегшує спілкування між двома процесорами, даючи скетчам Arduino можливість запуску скриптів на Linux, спілкуватися з мережевими інтерфейсами, та отримувати інформацію від процесора AR9331.

Специфікація

Таблиця 1.2 – Специфікація Arduino YUN

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage	5V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 4 KB used by bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 Hz

USB хост, мережеві інтерфейси і SD-карта не підключена до 32U4, але бібліотека Bridge дозволяє Arduino взаємодіяти з периферією.

1.2. Драйвер двигунів

Драйвер (англ. Driver - керуючий пристрій, водій) - електронний пристрій, призначений для перетворення електричних сигналів, метою якого є управління чимось. Драйвером зазвичай називається окремий пристрій або окремий модуль, мікросхема в пристрої, що забезпечують перетворення електричних сигналів в електричні або інші впливи, придатні для безпосереднього управління виконавчими або сигнальними елементами.

Під визначення драйвера підпадають численні пристрої:

- Шинні формувачі, призначені для передачі сигналів з одного рівня цифрового пристрою на інший з перетворенням рівня, посиленням здатності навантаження та іншими особливостями. Такі пристрої забезпечують передачу даних між різними логічними блоками по загальним лініях зв'язку всередині обчислювальних машин.
- Формувачі сигналів інтерфейсів цифрових електронних пристроїв, призначені для перетворення, прийому і передачі цифрових сигналів і узгодження електричних параметрів з особливостями лінії зв'язку. Найбільш відомими представниками таких драйверів вважаються формувачі інтерфейсів RS-232 (COM - порт), RS-485, RS-422, CAN, LIN, Ethernet, USB, IEEE 1394 і т. Д.
- Пристрої управління різними типами виконавчих пристроїв, такими як електромагніти, електродвигуни (у тому числі крокові), сигнальні лампи, дозатори (у тому числі друкують голівки принтерів), сервоприводи, звукові сигнали і т. Д. [3]
- Модулі живлення і управління пристроями, що вимагають дотримання певних робочих параметрів в процесі включення,

виключення і роботи. Яскравим прикладом можна вважати драйвери світлодіодів, оскільки до харчування світлодіодних пристроїв пред'являються підвищені вимоги [4].

- Драйвери силових транзисторів, MOSFET і IGBT-транзисторів. Затвори потужних польових силових транзисторів мають велику електричну ємність (тисячі пікофарад), для зарядки яких на високій частоті потрібен великий струм (ампер). Драйвер забезпечує великий струм для швидкої зарядки затвора транзистора для його відкриття. А також швидко розряджає затвор, коли транзистор потрібно закрити.

1.2.1. Драйвер двох двигунів на L298N

Можна використовувати цей модуль, щоб керувати двома двигунами постійного струму або чотирьох провідним двофазним кроковим двигуном. Плата також може бути використана в якості платформи драйвера постійного струму.

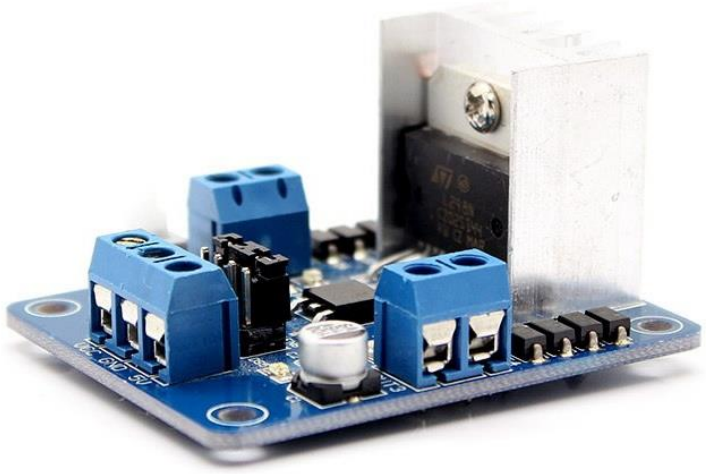

Рисунок.1.5 – Зовнішній вигляд драйвера на L298N [2]

Вимикач S1 служить для перемикання живлення логічної частини мікросхеми. Тобто при включеному S1 живлення логічної частини береться від внутрішнього перетворювача модуля. При вимкненому S1 харчування береться від зовнішнього джерела.

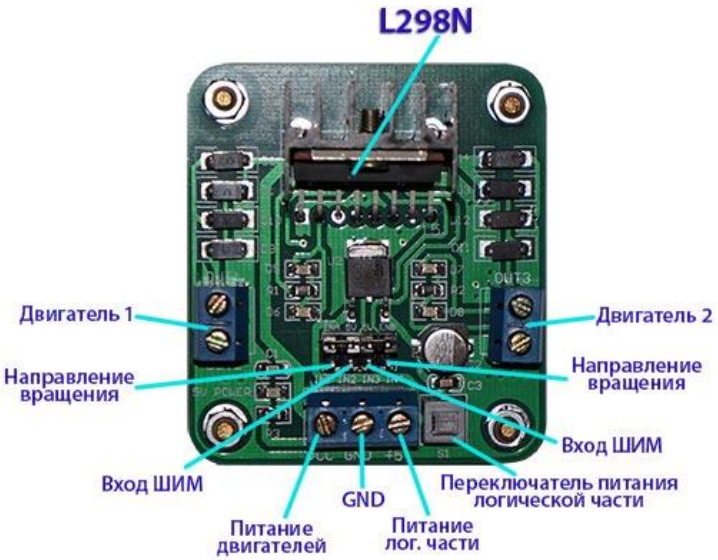

Рисунок.1.6 – Розташування елементів драйвера на L298N [3]

У всьому іншому цей драйвер повторює аналогічні конструкції. При напрузі живлення драйвера від 7 до 20 вольт можна використовувати вбудований стабілізатор напруги 5В. При меншій напрузі рекомендується використовувати напругу 5В контролера. При більшому - зовнішній стабілізатор або окреме джерело живлення 5В.

Основні параметри:

- **Мікросхема драйвера**: L298N здвоєний Н-мостовий драйвер двигунів постійного струму.
- **Напруга живлення силової частини драйвера Vs**: +5 V ~ +35 V
- **Піковий струм Io**: 2A
- **Напруга живлення логічної частини Vss:** +5 V ~ +7 V (може бути запитана від основної плати + 5V)

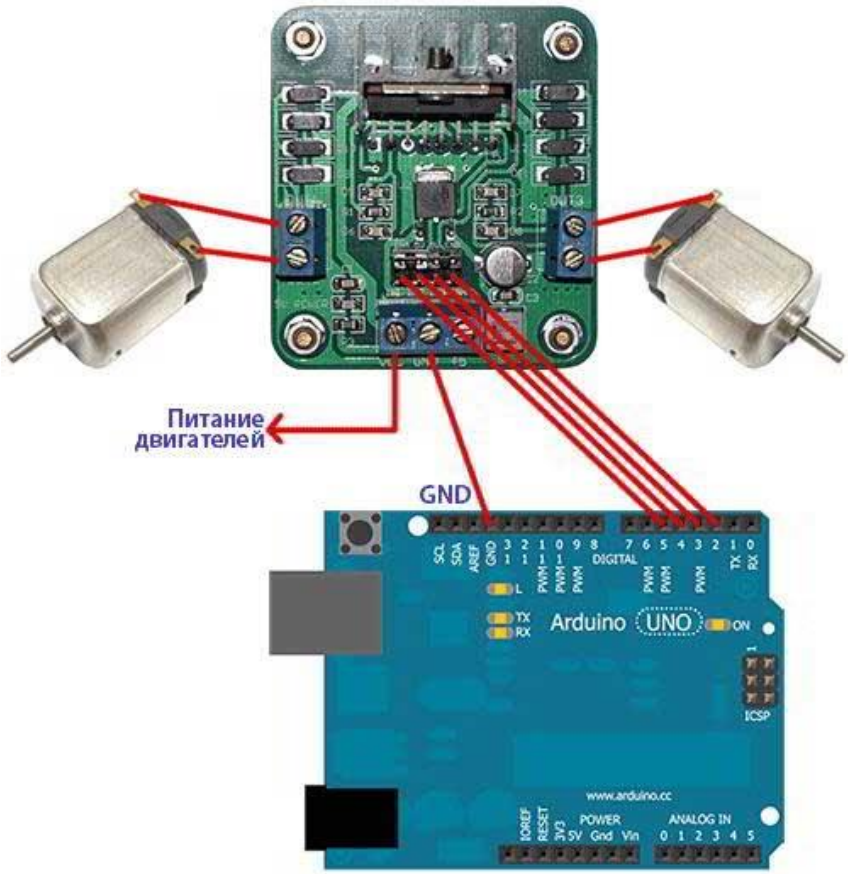

Рисунок.1.7 – Приклад підключення драйвера на L298N [4]

- **Струм логічної частини:** 0 ~ 36mA
- **Діапазон напруг керуючих сигналів і входів дозволу:**
  - **Низький рівень:** -0.3V ≤ Vin ≤ 1.5V
  - **Високий рівень**: 2.3V ≤ Vin ≤ Vss
- **Максимальна споживана потужність**: 20W (для температури T = 75°)
- **Температура зберігання:** -25 ° ~ +130 °
- **Габаритні розміри**: 48мм * 43мм * 33мм (з урахуванням висоти радіатора мікросхеми)
- **Вага**: 33 грами

Особливості: наявність світлодіодних індикаторів керуючих сигналів і напруги живлення логічної частини драйвера.

1.2.2. Flyduino-A контролер 12-ти серводвигунів

Це напевно самий маленький Arduino-контролер спеціально спроектований для управління сервами. Дозволяє управляти дванадцятьма стандартними сервами, має XBee сокет, який може бути використаний в моделі вертольота або UAV. Маленький вага (всього 7.5 грам) робить його ідеальним для проектів з обмеженою корисним навантаженням. Вбудований регулятор дозволяє підключатися до вхідному напрузі 3.5-8В забезпечується більшістю джерел живлення.

Рисунок.1.8 – Зовнішній вигляд драйвера Flyduino [4]

Працює з Arduino IDE як Arduino Mini (потрібно перехідник для підключення до USB з виходом живлення 3.3В. Не використовуйте перехідники з напругою живлення 5В інакше ви можете пошкодити контролер!). В якості бібліотеки рекомендується використовувати MegaServo.

Характеристики:

- контролер Atmega328P
- робоча напруга 3.3В
- вхідна напруга 3.5-8В
- 12 канальне управління серво-двигунами

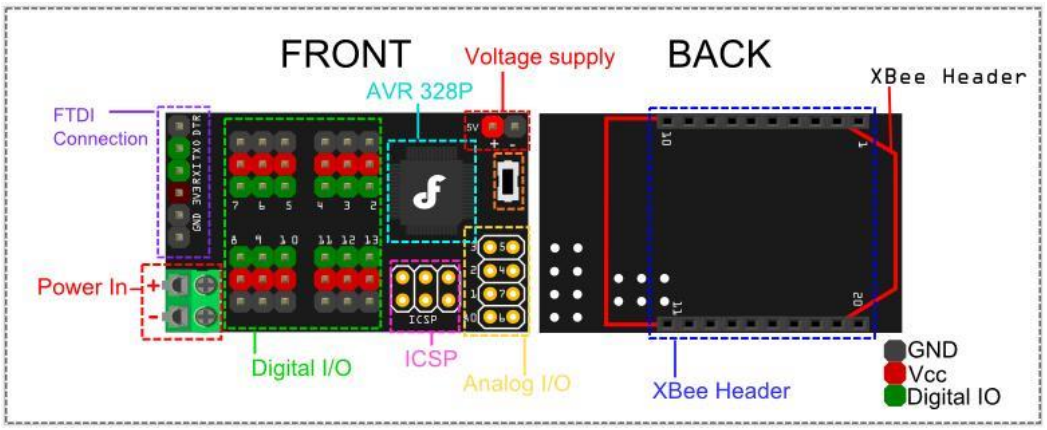

Рисунок.1.9 – Схема елементів драйвера Flyduino [5]

- Xbee сокет
- аналогових входів: 8
- SRAM: 2кб
- EEPROM: 1кб
- Вага: 7.5 гр

1.3. Датчики

Дава́ч (у багатьох джерелах — да́тчик, се́нсор) (рос.датчик, англ. sensing element, detector, transducer sensor; нім. Meßgeber m, Geber m, Fühler m) — вимірювальний пристрій у вигляді конструктивної сукупності одного або декількох вимірювальних перетворювачів величини, що вимірюється і контролюється, у вихідний сигнал для дистанційної передачі та використання в системах керування і має нормовані метрологічні характеристики.

Датчики є елементом технічних систем, призначених для вимірювання, сигналізації, регулювання, управління приладами і процесами. Датчики перетворюють величину, яка контролюється (тиск, температура, витрата, концентрація, частота, швидкість, переміщення, електрична напруга, електричний струм і т.д.) в сигнал (електричний, оптичний, пневматичний),

зручний для вимірювання, передачі, перетворення, зберігання і реєстрації інформації про стан об'єкта вимірювання.

Історично і логічно датчики пов'язані з технікою вимірювання та вимірювальними приладами, наприклад: термометри, витратоміри, прилади для вимірювання тиску і т.д.; узагальнювальний термін «давач (датчик)» закріпився у зв'язку з розвитком автоматичних систем керування, як елемент узагальненої логічної концепції «давач — система керування — виконавчий пристрій — Об'єкт керування». Спеціальний випадок представляє використання датчиків в автоматичних системах реєстрації параметрів, наприклад, в системах наукових досліджень.

Останнім часом стосовно давачів застосовуються терміни: «багатофункційний давач» чи «інтелектуальний давач», що відбиває тенденції розвитку сучасних давачів. Під цими термінами крім функцій первинного вимірювального перетворення, маються на увазі додаткові можливості вимірювання декількох фізичних величин та використання вбудованих аналого-цифрових перетворювачів з мікроконтролерами, що суттєво розширює функціональний діапазон давачів, а саме:

- попередня обробка сигналів (лінеаризація, фільтрування, корекція похибок);
- самодіагностування;
- дистанційне конфігурування (діапазону вимірювань, одиниць вимірювань, узгодження частотних характеристик);
- окремі елементи керування;

1.3.1. Гіроскоп-акселерометр MPU 6050.

Перш ніж приступити до розгляду модуля гіроскопа і акселерометра, думаю, буде не зайвим коротко розібратися що це таке. Гіроскоп являє собою пристрій, що реагує на зміну кутів орієнтації контрольованого тіла. У

класичному уявленні це якийсь інерційний предмет, який швидко обертається на підвісах. Як результат обертається предмет завжди буде зберігати свій напрямок, а по положенню підвісів можна визначити кут відхилення. Насправді ж електронні гіроскопи побудовані за іншою схемою і влаштовані трохи складніше. Акселерометр - це пристрій, який вимірює проекцію удаваного прискорення, тобто різниці між істинним прискоренням об'єкта і гравітаційним прискоренням. На простому прикладі така система являє собою деяку масу, закріплену на підвісі, що володіє пружністю (пружина для доброго прикладу). Так от якщо таку систему повернути під якимось кутом, або кинути, або зрадити лінійне прискорення, то пружний підвіс відреагує на рух під дією маси і відхилиться і ось по цьому відхиленню визначається прискорення. Таким чином, гіроскоп реагує на зміну в просторі незалежно від напрямок руху, за допомогою акселерометра ж може вимірювати лінійні прискорення предмета, а так само і штучно розраховується розташування предмета в просторі. Кожен пристрій має свої переваги і недоліки.

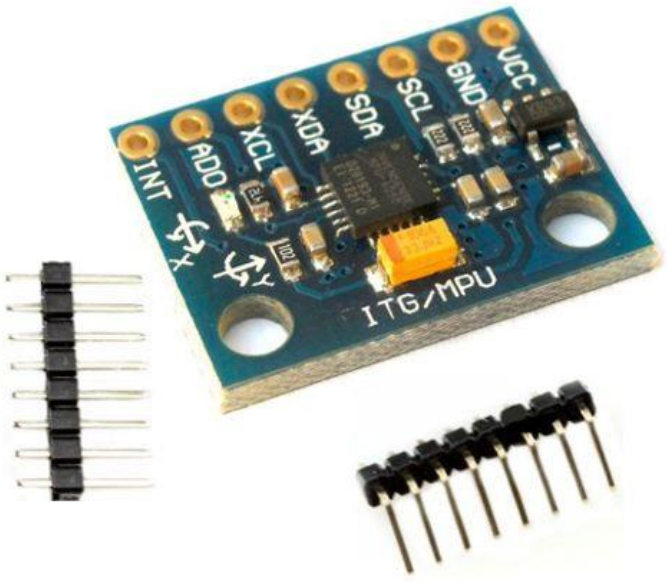

Рисунок.1.10 – Зовнішній вигляд MPU6050 [4]

Мікросхема MPU6050 містить на борту як акселерометр, так і гіроскоп, а крім цього ще і температурний сенсор. MPU6050 є головним елементом модуля GY-531. Крім цієї мікросхеми на платі модуля розташована необхідна обв'язка MPU6050, в тому числі підтягує резистори інтерфейсу I2C, а також стабілізатор напруги на 3,3 вольта з малим падінням напруги (при харчуванні вже в 3,3 вольта на виході стабілізатора буде 3 рівно вольта) з фільтруючими конденсаторами [6]. Ну і бонусом на платі розпаяний SMD світлодіод з обмежуючим резистором як індикатор напруги живлення. Розмір плати модуля GY-521 10 х 20 мм.

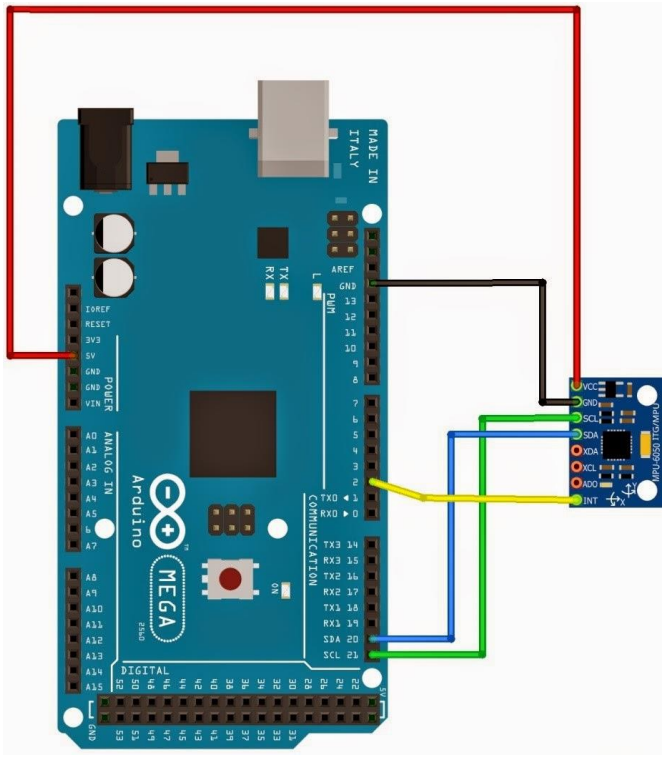

Рисунок.1.11 – Приклад підключення MPU6050[6]

Функції MPU6050:

- трьох осьової MEMS гіроскоп з 16 бітовим АЦП
- трьох осьової MEMS акселерометр з 16 бітовим АЦП

- Digital Motion Processor (DMP)
- slave I2C для підключення до мікроконтролеру
- master I2C для підключення до мікросхеми додаткового датчика
- регістри даних датчиків
- FIFO
- Переривання
- температурний сенсор
- самоперевірка гіроскопа і акселерометра
- регістр ідентифікації пристрою

1.3.2. Ультразвуковий датчик відстані HC-SR04

Цей далекомір може служити прекрасним датчиком для робота, завдяки якому він зможе визначати відстані до об'єктів, об'їжджати перешкоди, або будувати карту приміщення. Його можна також використовувати в якості датчика для сигналізації, що спрацьовує при наближенні об'єктів.

Рисунок.1.12 – Зовнішній вигляд HC-SR04[6]

Принцип дії:

Ультразвуковий далекомір визначає відстань до об'єктів точно так само, як це роблять дельфіни або кажани. Він генерує звукові імпульси на частоті 40 кГц і слухає відлуння. За часом поширення звукової хвилі туди і назад можна однозначно визначити відстань до об'єкта [7].

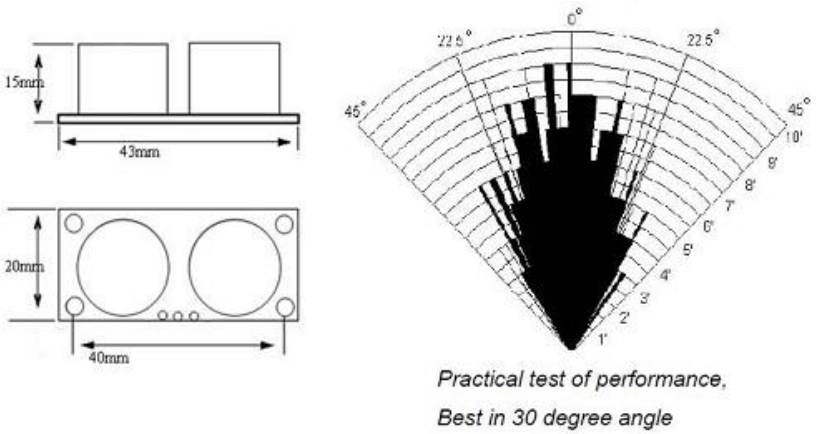

Рисунок.1.13 – Діаграма показників практичного використання HC-SR04[6]

На відміну від інфрачервоних далекомірів, на свідчення ультразвукового далекоміра не впливають засвічення від сонця або колір об'єкта. Але можуть виникнути труднощі з визначенням відстані до пухнастих або дуже тонких предметів.

1.4. Висновок

В даному розділі була розглянута мікроконтролерна частина дипломного проекту. А саме були розглянуті різні платформи, що служать основою для розробки клієнт серверної моделі управління. Також були розглянуті периферійні пристрої для використання їх у практичній частині роботи.

2. ПРОГРАМНІ ЗАСОБИ СТВОРЕННЯ КЛІЄНТ-СЕРВЕРНОЇ МОДЕЛІ

2.1 Arduino IDE 1.6.4

Arduino IDE це середовище розробки з відкритим вихідним кодом, написане спеціально для створення скетчів під Arduino платформу. Воно дозволяє легко написати код і завантажити його на плату. Arduino IDE працює на Windows, Mac OS X, Linux і. Середовище написане в Java на основі Processing та іншого програмного забезпечення з відкритим вихідним кодом.

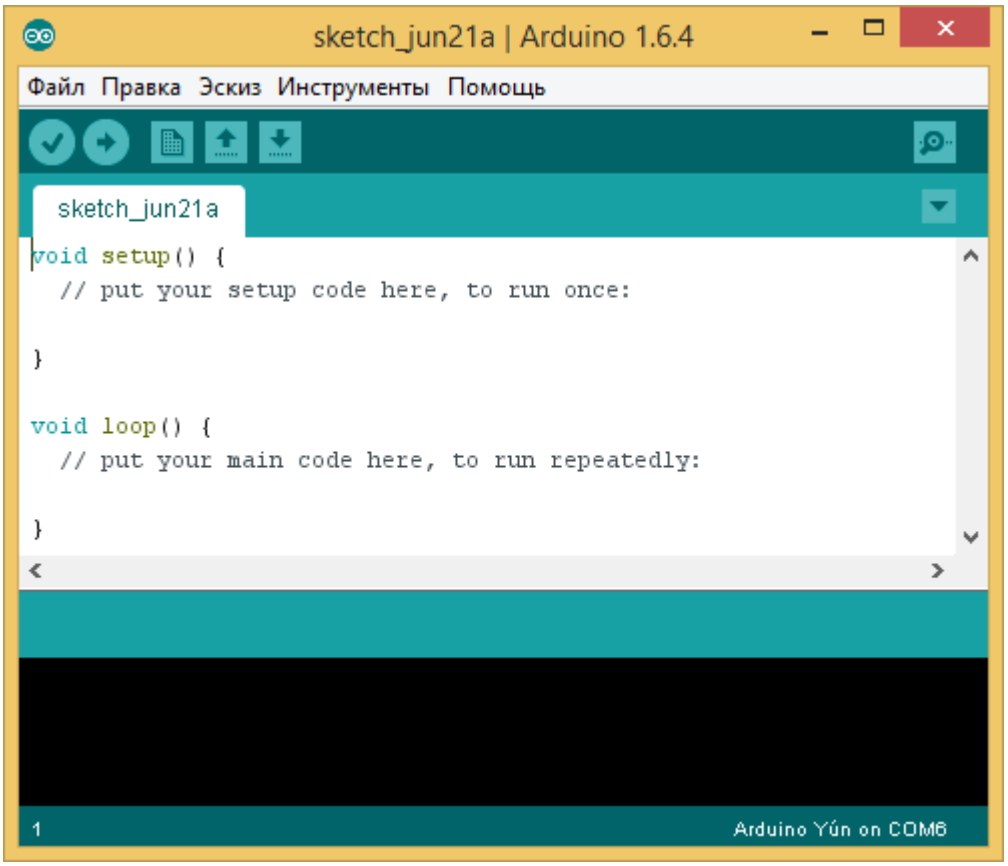

Рисунок.2.1Зовнішній вигляд Arduino IDE 1.6.4 [5]

Середовище розробки Arduino складається з вбудованого текстового редактору програмного коду, області повідомлень, вікна виведення тексту

(консолі), панелі інструментів з кнопками часто використовуваних команд і декількох меню. Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини Arduino [8].

Програма, написана в середовищі Arduino, називається скетч. Скетч пишеться в текстовому редакторі, що має інструменти вирізки / вставки, пошуку / заміни тексту. Під час збереження та експорту проекту в області повідомлень з'являються пояснення, також можуть відображатися виникли помилки. Вікно виводу тексту (консоль) показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дозволяють перевірити і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини:

 «Verify»

Перевірка програмного коду на помилки, компіляція.

 «Load»

Перевірка програмного коду на помилки, компіляція та загрузка коду на платформу

 «New»

Створення нового скетчу.

 «Open»

Відкриття меню доступу до всіх скетчам в блокноті. Відкривається натисканням в поточному вікні.

 «Save»

Збереження скетчу.

 «Serial monitor»

Відкриття моніторингу послідовної шини (Serial monitor).

Додаткові команди згруповані в п'ять меню: File, Edit, Sketch, Tools, Help. Доступність меню визначається роботою, виконуваної в даний момент.

Edit

- Copy for Discourse - Копіює в буфер обміну відповідний для розміщення на форумі код скетчу з виділенням синтаксису.
- Copy as HTML - Копіює код скетчу в буфер обміну як HTML код, для розміщення на веб-сторінках.

Sketch

- *Verify* - Перевірка скетчу на помилки.
- *Import Library* - Додає бібліотеку в поточний скетч, вставляючи директиву #include в код скетчу. Детальна інформація в описі бібліотек нижче (Libraries).
- *Show Sketch Folder* - Відкриває папку, яка містить файл скетчу, на робочому столі.
- *Add File...* - Додає файл в скетч (файл буде скопійований з поточного місця розташування). Новий файл з'являється в новій закладці у вікні скетчу. Файл може бути видалений з скетчу за допомогою меню закладок.

Tools

- *Auto Format* - Дана опція оптимізує код, наприклад, вибудовує в одну лінію по вертикалі відкриває і закриває дужки і поміщає між ними твердження.
- *Board* - Вибір використовуваної платформи. Список з описом платформ наводиться нижче.
- *Serial Port* - Меню містить список послідовних пристроїв передачі даних (реальних і віртуальних) на комп'ютері. Список оновлюється автоматично кожного разу при відкритті меню Tools.
- *Burn Bootloader* - Пункти даного меню дозволяють записати Завантажувач (Bootloader) в мікроконтролер на платформі Arduino. Дана

дія не потрібно в поточній роботі з Arduino, але стане в нагоді, якщо є новий ATmega (без завантажувача). Перед записом рекомендується перевірити правильність вибору платформи з меню. При використанні AVR ISP необхідно вибрати відповідний программатору порт з меню Serial Port [9].

Блокнот (Sketchbook)

Середовищем Arduino використовується принцип блокнота: стандартне місце для зберігання програм (скетчів). Скетчі з блокнота відкриваються через меню File> Sketchbook або кнопкою Open на панелі інструментів. При першому запуску програми Arduino автоматично створюється директорія для блокнота. Розташування блокнота змінюється через діалогове вікно Preferences.

Закладки, Файли та Компіляція

Дозволяють працювати з декількома файлами скетчів (кожен відкривається в окремій закладці). Файли коду можуть бути стандартними Arduino (без розширення), файлами C (розширення * .c), файлами C ++ (* .cpp) або головними файлами (.h).

Завантаження скетчу в Arduino

Перед завантаженням скетчу потрібно задати необхідні параметри в меню Tools> Board і Tools> Serial Port. Платформи описуються далі за текстом. В ОС Mac послідовний порт може позначатися як dev / tty.usbserial-1B1 (для плати USB) або /dev/tty.USA19QW1b1P1.1 (для плати послідовної шини, підключеної через адаптер Keyspan USB-to-Serial). В ОС Windows порти можуть позначатися як COM1 або COM2 (для плати послідовної шини) або COM4, COM5, COM7 і вище (для плати USB). Визначення порту USB проводиться в поле Послідовною шини USB Диспетчера пристроїв Windows. В ОС Linux порти можуть позначатися як / dev / ttyUSB0, / dev / ttyUSB1.

Після вибору порту і платформи необхідно натиснути кнопку завантаження на панелі інструментів або вибрати пункт меню File> Upload to I / O Board. Сучасні платформи Arduino перезавантажуються автоматично перед завантаженням. На старих платформах необхідно натиснути кнопку перезавантаження. На більшості плат під час процесу будуть мигати світлодіоди RX і TX. Середовище розробки Arduino виведе повідомлення про закінчення завантаження або про помилки.

При завантаженні скетчу використовується Завантажувач (Bootloader) Arduino, невелика програма, що завантажується в мікроконтролер на платі. Вона дозволяє завантажувати програмний код без використання додаткових апаратних засобів. Завантажувач (Bootloader) активний в перебігу декількох секунд при перезавантаженні платформи і при завантаженні будь-якого з скетчів в мікроконтролер. Робота завантажувач (Bootloader) розпізнається за миготінню світлодіода (13 пін) (наприклад .: при перезавантаженні плати).

Моніторинг послідовної шини (Serial Monitor)

Відображає дані посилаються в платформу Arduino (плата USB або плата послідовної шини). Для відправки даних необхідно ввести текст і натиснути кнопку Send або Enter. Потім вибирається швидкість передачі зі списку, відповідна значенню Serial.begin в скетчі. На ОС Mac або Linux платформа Arduino буде перезавантажена (скетч почнеться спочатку) при підключенні моніторингу послідовної шини.

2.2 Пропорційно – Інтегрально – Диференціальний регулятор (ПІД)

Пропорціонально-інтегрально-диференціальний (ПІД) закон регулювання - найскладніший алгоритм функціонування автоматичного регулятора, що включає вплив усіх розглянутих вище законів.

Реалізація цього закону пов’язана із застосуванням пружного зворотного зв’язку. На рис. 1 б подана перехідна функція ПІД-закону, де виділено області впливу складовими Д, П, І закону.

Регулятори з випередженням значно поліпшують якість регулювання, особливо якщо об’єкт володіє великим запізненням та інерційністю.

Рівняння ПІД-закону мають вигляд:

$$u = \mathrm{K}_{1\varepsilon} + \mathrm{K}_2 \int \varepsilon dt + \mathrm{K}_3 \frac{d\varepsilon}{dt} = \mathrm{K}_1 (\varepsilon + \frac{1}{\mathrm{T}_\mathrm{i}} \int \varepsilon dt + \mathrm{T}_\mathrm{p} \frac{d\varepsilon}{dt}),$$

що реалізовується ізодромним регулятором з передуванням або ПІД-регулятором з параметрами налаштування *К1:*

$$\mathrm{T}_\mathrm{i} = \frac{\mathrm{K}_1}{\mathrm{K}_2}, \mathrm{T}_\mathrm{p} = \frac{\mathrm{K}_3}{\mathrm{K}_1}.$$

Настроювальні параметри кр, Ті і Тр регулятора зумовлюють вид і якість перехідного процесу системи регулювання, як і динамічні властивості об'єкта.

Найчастіше можна зустріти приклади, де ПІД-регулятор використовується для регулювання температури, і, на мій погляд, цей приклад прекрасно підходить для вивчення теорії і розуміння принципу роботи регулятора. Тому саме завдання регулювання температури і буде розглядатись.

Вихідні дані:

По-перше, об'єкт, температуру якого необхідно підтримувати на заданому рівні, крім того, цю температуру необхідно регулювати ззовні. По-друге, наш пристрій на базі мікроконтролера, за допомогою якого ми і будемо вирішувати поставлену задачу. Крім того, у нас є вимірювач температури (він повідомить контролеру поточну температуру) і який-небудь пристрій для управління потужністю нагрівача. Ну і оскільки необхідно якось задавати температуру, підключимо мікроконтролер до ПК.

Таким чином, у нас є вхідні дані - поточна температура і температура, до якої необхідно нагріти / остудити об'єкт, а на виході ми маємо отримати значення потужності, яке необхідно передати на нагрівальний елемент.

І для такого завдання, та й взагалі будь схожою завдання, відмінним рішенням буде використання пропорційно-інтегрально-диференціального регулятора.

Пропорційна складова.

Тут все просто, беремо значення потрібної нам температури (уставку) і віднімаємо з нього значення поточної температури. Отримуємо неузгодженість (невязку). Множимо отриману невязку на коефіцієнт і отримуємо значення потужності, яке і передаємо на нагрівач. Ось і все) Але при використанні тільки пропорційною складовою є два великих мінуса - по-перше, ефект від нашого впливу настає не моментально, а з запізненням, і, по-друге, пропорційна складова ніяк не враховує вплив навколишнього середовища на об'єкт. Наприклад, коли ми домоглися того, щоб температури об'єкта дорівнювала потрібного нам значенням, невязка стала дорівнювати нулю, а разом з нею і видавана потужність стала нульовою. Але температура не може просто так залишатися постійною, оскільки відбувається теплообмін з навколишнім середовищем і об'єкт охолоджується. Таким чином, при використанні тільки пропорційною складовою температура коливатиметься близько потрібного нам значення.

Давайте розбиратися, як ПІД-регулятор вирішує дві виявлені проблеми).

Для вирішення першого використовується диференціальна складова. Вона протидіє передбачуваним відхиленням регульованої величини, які можуть відбутися в майбутньому.

Отже, нехай у нас поточна температура менше потрібного нам значення. Пропорційна складова починає видавати потужність і нагрівати об'єкт. Диференціальна складова вносить свій внесок у потужність і представляє з

себе похідну температури, взяту також з певним коефіцієнтом (у програмі для контролера необхідно брати різницю між поточним значенням невязки і попереднім). Температура зростає і наближається до потрібного значення, а отже невязка в попередній момент більше поточного значення нев'язки, а похідна негативна. Таким чином, диференціальна складова починає поступово знижувати потужність до того, як температура досягла необхідного значення. З цим начебто розібралися, згадуємо про другу проблему регулятора.

А з нею нам допоможе впоратися інтегральна складова. Як нам в програмі отримати інтеграл? А легко - просто підсумовуванням (накопиченням) значень невязки, на те він і інтеграл) Повертаємося до нашого прикладу. Температура нижче значення уставки, починаємо підігрівати. Поки ми нагріваємо, значення нев'язки позитивне і накопичується в інтегральною складовою. Коли температура «дійшла» до потрібного нам значення, пропорційна і диференціальна складова стали рівні нулю, а інтегральна перестала змінюватися, але її значення не стало рівним нулю. Таким чином, завдяки накопиченому інтегралу ми продовжуємо видавати потужність і нагрівач підтримує потрібну нам температуру, не даючи об'єкту охолоджуватися. Ось так от просто і ефективно.

В результаті ми отримуємо наступну формулу ПІД-регулятора:

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

Тут u(t) - шукане вихідний вплив, а e(t) - значення нев'язки. Частенько формулу перетворюють до наступного вигляду, але суть від цього не змінюється:

$$u(t) = K_p(e(t) + K_{ip} \int_0^t e(\tau)d\tau + K_{dp} \frac{de}{dt})$$

Налаштування коефіцієнтів ПІД-регулятора.

Основні методи настройки і підбору його коефіцієнтів) Взагалі, за великим рахунком, при використанні ПІД-регулятора необхідно побудувати модель всієї системи в цілому і математично обчислити необхідні значення коефіцієнтів. Так робити правильно. Але, природно, так ніхто не робить. Насправді, математичний розрахунок коефіцієнтів задача далеко не тривіальна, вимагає глибоких знань теорії автоматичного управління, тому й використовуються інші, спрощені, методи настройки.

Найбільш часто використовується методом налаштування коефіцієнтів є метод Циглера-Нікольса. Полягає він в наступному ...

Метод Циглера-Нікольса.

- Для початку обнуляємо всі коефіцієнти регулятора (пропорційний, інтегральний і диференціальний)
- Поступово починаємо збільшувати пропорційний коефіцієнт і стежимо за реакцією системи. При певному значенні виникнуть незгасаючі коливання регульованої величини.
- Фіксуємо коефіцієнт, при якому це сталося. Крім того, заміряємо період коливань системи.

Власне, на цьому практична частина методу закінчується) З отриманого коефіцієнта розраховуємо пропорційний коефіцієнт ПІД-регулятора:

$$K_p = K \cdot 0.6$$

А з нього отримуємо і інші:

$$K_u = (2 \cdot K_p) / T,$$
$$K_g = (K_p \cdot T) / 8$$

Метод досить простий, але застосувати його можна далеко не завжди. Якщо чесно, мені ще жодного разу не доводилося налаштовувати регулятор

таким чином. Але тим не менш, цей метод є основним і, за великим рахунком, єдиним широко відомим. Просто підходить не всім і не завжди.

Що ж робити, якщо метод Циглера-Нікольса не спрацював? Тут прийде на допомогу «аналітичний» метод настройки.

Знову ж Обнуляємо всі коефіцієнти і починаємо збільшувати пропорційний. Але тепер не чекаємо появи коливань, а просто фіксуємо поведінку системи для кожного значення коефіцієнта (відмінним варіантом буде побудова графіка величини, яку необхідно стабілізувати, для кожного значення коефіцієнта). Якщо бачимо, що, наприклад, система дуже повільно виходить на потрібне значення, збільшуємо пропорційний коефіцієнт. Система починає сильно коливатися щодо потрібної величини? Значить, коефіцієнт занадто великий, зменшуємо і переходимо до налаштування інших складових.

Розуміючи, як працює ПІД-регулятор в цілому, і уявляючи, як повинна працювати настроюється система, можна досить-таки швидко і точно налаштувати коефіцієнти регулятора. Особливо, якщо є можливість побудувати графічні залежності і візуально стежити за поведінкою системи.

Ось деякі правила, які можуть допомогти при налаштуванні:

- Збільшення пропорційного коефіцієнта призводить до збільшення швидкодії, але зниження стійкості системи
- Збільшення диференціальної складової також призводить до значного збільшення швидкодії
- Диференціальна складова покликана усунути затухаючі коливання, що виникають при використанні тільки пропорційною складовою
- Інтегральна складова повинна усувати залишкове неузгодженість системи при налаштованих пропорційної і диференціальної складових

До речі, варто додати, що не завжди необхідно використовувати всі три складові ПІД-регулятора, часом вистачає пропорційної і диференціальної, наприклад (ПД-регулятор). Загалом, все зводиться до того, що для кожної системи необхідний свій власний підхід при налаштуванні і використанні ПІД-регулятора.

2.3 Висновок

У даному розділі розглянуті програмні засоби для створення клієнт-серверної системи на основі Arduino. Також було розглянуто і, в подальшому, впроваджено алгоритм роботи ПІД регулятора.

3. КЛІЄНТ-СЕРВЕРНА МОДЕЛЬ КЕРУВАННЯ МІКРОКОНТРОЛЕРНИМИ СИСТЕМАМИ

Технологія клієнт - сервер, яка широко застосовується при роботі з базами даних в мережі, відома вже давно і найчастіше застосовувалась у великих організаціях. Сьогодні, з розвитком INTERNET, ця технологія все частіше приваблює погляди розробників програмного забезпечення, оскільки в світі нагромаджено величезну кількість інформації по різноманітних питаннях і найчастіше ця інформація зберігається в базах даних. Технологію клієнт - сервер можна описати наступним алгоритмом:

- клієнт формує і посилає запит до бази даних серверу, вірніше - до програми, яка обробляє запити;
- ця програма проводить маніпуляції з базами даних, що знаходяться на сервері, у відповідності з запитом, формує результат і передає його клієнту;
- клієнт отримує результат, відображає його на дисплеї і чекає подальших дій користувача. Цикл повторюється до того часу, поки користувач не завершить роботу з сервером.

Стандартне програмне забезпечення, що реалізує технологію клієнт – сервер, має хорошу масштабованість (ефективне використання нарощеного апаратного забезпечення), стійкість в роботі, захист від несанкціонованого доступу і потужність при роботі з великими проектами в галузі баз даних.

Конкретно все залежить від того, де знаходиться клієнт та сервер, і як клієнт під’єднаний до серверу. Користувач на клієнтському комп’ютері в програмі перегляду заповнює запропоновану форму або вибирає подальшу дію. Броузер (програма пошуку) по натиску однієї з кнопок на формі пересилає дані із заповненої форми або відображає заново отримані в результаті деякої операції. Не важливо, до якої з мереж під’єднаний клієнт. Він навіть може бути

віддаленим користувачем і з’єднуватися по модему. Програма приймає дані, перевіряє їх і формує запит до монітора баз даних або отримує від нього результат. Отримавши запит, монітор опрацьовує його і тоді, якщо не сталося помилок обробляє і відправляє потрібні дані програмі. На диску сервера зберігається база даних, що модифікується по запиту клієнта. При такому режимі роботи забезпечується високий рівень безпеки бази даних як від збоїв обладнання і програм, так і від несанкціонованого доступу, висока продуктивність, навантаження на мережу падають, але зростають вимоги до серверу.

3.1 Сервер

Мікроконтролера плата Arduino YUN включає в собі два мікропроцесора, один з яких архітектури AVR ATmega 32u4, що займається посиланням сигналів на периферійні пристрої, діоди і т.п. Другий з них Linino AR 9331 містить на собі операційну систему Linux, через яку має доступ до Wi-Fi адаптера. Цей адаптер може працювати в двох режимах:

- Точка доступу – працює як звичайний маршрутизатор до якого можна підключитись на віддавати команди
- Підключення – режим у якому його можна конфігурувати для підключення до будь-якої точки доступу та працювати з ним по локальній мережі

В будь-якому з цих випадків, Arduino YUN можна використовувати, як сервер, що успішно використовується багатьма дослідниками світу.

З цією мікроконтролерною системою працюють за допомогою скетчів, які завантажують на платформу через програматор Arduino, тобто сам програмний код, що пишеться записується лише в пам’ять з якою працює ATmega 32u4, а для підтримання зв’язку з Linino AR 9331 використовується

бібліотека Bridge, яка містить ряд методів, для спілкування з лінуксовою частиною системи.

Перш за все треба запустити цю систему зв’язку через метод класу Bridge:

Bridge.begin();

Також ця бібліотека передбачує механізм сервера, що представляється, як клас YunServer, який потрібно налаштувати на роботу з клієнтами напряму через його власну мережу YunServer::listenOnLocalhost(), або на роботу через точку доступу YunServer::noListenOnLocalhost().

Звичайно після цього сервер варто запустити для початку прийняття клієнтів методом YunServer::begin().

Далі у функції головного циклу програми void **loop**() в кожній ітерації циклу потрібно відловлювати клієнтів, які намагаються приєднатися до нашого YunServer. За це відповідає клас YunClient, екземпляри якого отримуються з серверу за допомогою метода YunServer::accept(). Якщо клієнта немає на вході, то результат такого методу **NULL**, в іншому випадку ми отримаємо об’єкт класу YunClient.

Основним механізмом передачі даних в Arduino через мережу є REST запити, що здійснюються через строку запиту в будь-якому браузері.

Для зчитування даних з клієнта є один основний метод YunClient::read(), що зчитує один байт інформації із рядка запиту. Але для зручності є методи YunClient::readStringUntil (char), YunClient::parseInt() та інші, що функціональністю відразу зрозумілі з назви.

ЦІ методи дозволяють парсити строку і діяти згідно з її параметрами.

Також досить важливим механізмом для сервера є запис та віддача даних клієнту. Це передбачено бібліотекою та виконується методами YunClient::print() та YunClient::println().

Але після зчитування зі строки запиту та відправки зворотного повідомлення обов’язково потрібно завершити з’єднання методом YunClient::stop().

3.2 Клієнт

Програма-клієнт взаємодіє з сервером, використовуючи певний протокол. Вона може запитувати з сервера будь-які дані, маніпулювати даними безпосередньо на сервері, запускати на сервері нові процеси і т. п. Отримані від сервера дані клієнтська програма може надавати користувачеві або використовувати як-небудь інакше, в залежності від призначення програми. Програма-клієнт і програма-сервер можуть працювати як на одному і тому ж комп'ютері, так і на різних. У другому випадку для обміну інформацією між ними використовується мережеве з'єднання.

Як було сказано вище, для комунікації з сервером, клієнти повинні використовувати REST запити, тому клієнтом може бути будь-який браузер. Але так як необхідно часто передавати запити, була розроблена програма на основі браузера, що допомагає в користуванні мікроконтролерною системою.

Програма написана в середовищі розробки NetBeans 8.0.2 на мові програмування Java 1.6. При створенні додатку, як оформлення зовнішнього вигляду була використана графічна бібліотека Java Swing.

Управління системою ведеться за допомогою джойстика (пояснення нижче), тому для написання клієнта також була використана бібліотека Jinput, що допомагає працювати з будь-якими, підключеними до комп’ютера маніпуляторами, клавіатурами і т.п.

3.3 Мережа

Як було описано раніше платформа Arduino YUN має в собі вбудований в собі Wi-Fi адаптер, і для робота саме з ним не вимагає ніяких додаткових драйверів чи бібліотек. При правильних його налаштуваннях, він може підключатися до будь-якої точки доступу, яка знаходиться на відстані, потрібній для передачі даних.

Другий спосіб створення мережі, це Ethernet-адаптер, вбудований в платформу, але цей спосіб є менш практичний, ніж Wi-Fi зв’язок, через те, що платформа не зможе вільно рухатись.

Третій спосіб, це GSM модуль. Його, нажаль немає в комплектації платформи, але його можна докупити та налаштувати за допомогою відповідних бібліотек. Такий спосіб з’єднання є найбільш практичним у відношенні автономності, але швидкість та надійність такого зв’язку погіршується.

3.4 Висновок

Описана клієнт-серверна модель дозволяє керувати мікроконтролерними системами на неймовірних відстанях. Головна вимога як до клієнта, так і до сервера це підключення до мережі Інтернет.

4. ВИКОРИСТАННЯ МОДЕЛІ НА ПРИКЛАДІ КОЛІСНОГО РОБОТА

У результаті проведеної роботи був створений прототип колісного робота, який використовує зазначене вище обладнання та програмне забезпечення. Перш ніж описати його, потрібно визначитися, що є робот.

Ро́бот (від чеськ. robota) — автоматичний пристрій, що призначений для виконання виробничих та інших операцій, які зазвичай виконувались безпосередньо людиною. Для опису автоматичних пристроїв дія яких, не має зовнішньої схожості з діями людини, переважно використовується термін «автомат».

У більшості випадків сучасні роботи промислового призначення — це «руки», маніпулятори, закріплені на основі і призначені для виконання одноманітної роботи типу складання, переміщення. До роботів також належать мобільні пристрої, що працюють у небезпечних для людини середовищах і керовані дистанційно, наприклад роботи, що працюють на великих водних глибинах, у космосі, пристрої військового призначення (ведення розвідки, розмінування, доставка боєприпасів тощо) та ін., а також роботизовані іграшки.

Робот може реалізуватися як керований системою керування електромеханічний, пневматичний, гідравлічний пристрій або їх комбінація, основне призначення якого — заміна людини на виробництві, небезпечних чи шкідливих середовищах, побуті тощо.

Робот може безпосередньо виконувати команди оператора, може працювати по заздалегідь складеній програмі або дотримуватись набору загальних вказівок з використанням технології штучного інтелекту. Ці завдання дозволяють полегшити або зовсім замінити людську працю на

виробництві, в будівництві, при роботі з важкими вантажами, шкідливими матеріалами, а також в інших важких або небезпечних для людини умовах.

Побутовий робот — робот, призначений для допомоги людині в повсякденному житті. Наразі поширення побутових роботів є невеликим, проте футурологи передбачають широке їх використання у найближчому майбутньому.

4.1 Конструкція робота

Основою розробленого робота є рухома платформа, що складається з двох палуб для монтування плат на датчиків з вирізьбленими в них дірками для закріплення периферії. Рухають платформу сервоприводи, що прикріплені внизу платформи з під'єднаними до них колесами. Як третя точка опори, використане паразитне колесо без приводу, що може рухатись у будь-який бік.

Рисунок. 4.1 – Зовнішній вигляд двопалубної платформи [7]

На даній платформі розташовані:

- Плата Arduino YUN
- Драйвер двох двигунів на L298N
- 4 гальванічні елементи по 1.5 V в корпусі (живлення двигунів)
- Ультразвуковий датчик відстані HC-SR04
- Гіроскоп-акселерометр MPU 6050
- Блок живлення Arduino YUN
- Вимикач живлення двигунів

4.2 Схема з’єднання

Схема, зображена на рисунку 4.2.1 є діючою схемою з’єднання Arduino YUN та периферійних пристроїв на даний момент.

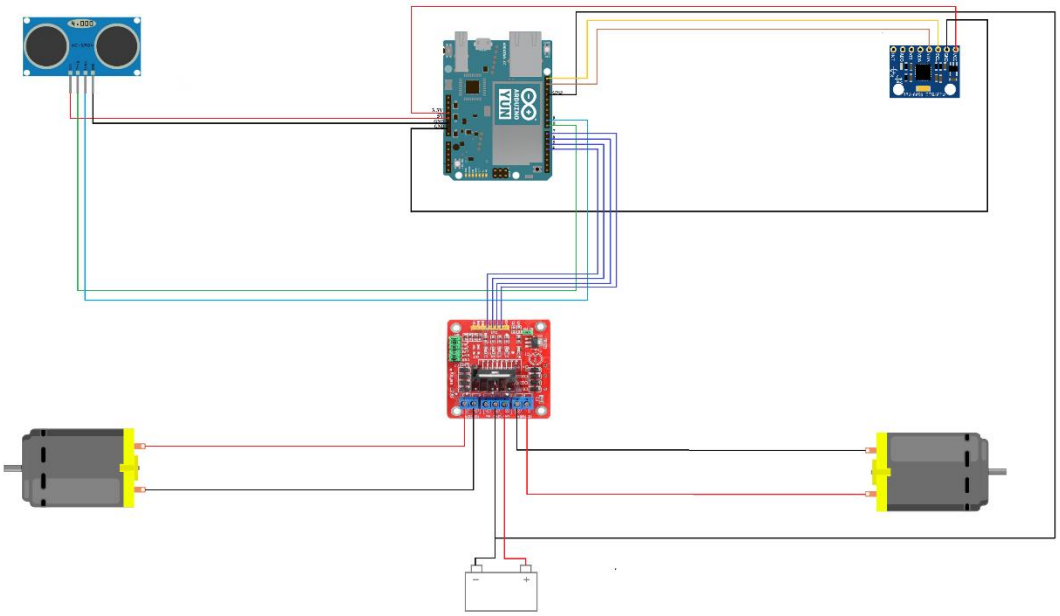

Рисунок. 4.2 – Схема з’єднання [8]

4.3 Опис режимів пристрою

Керування роботом здійснюється через REST API, і кожен режим має свою назву або ключове слово його увімкнення та певні параметри, які передбачає використання цього режиму. Всі запити починаються з введення слова “arduino”, що означає передачу запиту AVR процесору та “drive”, що вмикає модель поведінки саме колісного робота.

4.3.1 Режим «Ручне керування»

4.3.1.1 Опис

Приводиться в дію за допомогою ключового слова “manual”. Прикладом запиту являється рядок:

http://ip_address/arduino/drive/manual/P:0,Y:-1000,X:-7,Z:-7/

Параметри:

- P – параметр ввімкнення подачі сигналу на колеса
- Y – швидкість руху вперед (>0) та назад (<0) в шкалі від -1000 до 1000
- X – швидкість повороту вліво (>0) та вправо (<0) в шкалі від -1000 до 1000
- Z – параметр швидкості розвороту за годинниковою стрілкою (>0) та проти неї (<0) в шкалі від -1000 до 1000

Управління ведеться з джойстика оператором за допомогою розробленої клієнтської програми. Джойстик можна нахиляти вперед, назад, вправо, вліво та по колу. Також його можна обертати навколо його осі, імітуючи розвертання.

4.3.1.2 Практичне значення

Такий механізм роботи може бути використаний для створення віддаленого управління різними пристроями. Наприклад:

- управління будівним краном, для убезпечення роботи та кращого контролю
- управління автомобілем, де потрібні точні маніпуляції
- управління інвалідним візком з двигуном

4.3.2 Режим «Переслідування»

4.3.2.1 Опис

Приводиться в дію за допомогою ключового слова “pursuit”. Прикладом запиту являється рядок:

http://ip_address/arduino/drive/pursuit/D:30/

Єдиним його параметром D являється відстань у сантиметрах.

Даний режим дозволяє роботу триматися за перешкодою на певній відстані, що вказана параметром D. За допомого ПІД – регулятора швидкість робота регулюється згідно з показами ультразвукового датчика відстані. При наближенні до перешкоди він зменшує швидкість, при віддалені від неї більш, ніж на D він збільшує швидкість, якщо відстань до об’єкта рівна D, він зберігає поточну швидкість.

4.3.2.2 Практичне значення

За допомогою цього механізму ми отримуємо певну систему, що може бути використаною при створенні електронного помічника водія в автомобілі, або при створенні автомобіля з автономним керуванням. Також ця система може бути впроваджена, як паркувальна система автомобіля.

4.3.3 Режим «Поворот кута»

4.3.3.1 Опис

Приводиться в дію за допомогою ключового слова “angle”. Прикладом запиту являється рядок:

http://ip_address/arduino/drive/angle/A:30/

Єдиним його параметром А являється кут в градусах.

Цей режим дозволяє роботу розвернутись навколо його осі на певний кут за або проти часової стрілки. Вимірювання кута розвороту проводиться за допомогою гіроскоп-акселерометра MPU6050, та алгоритму ПІД який дозволяє плавно повертатися зменшуючи швидкість повороту у залежності від кута, на який залишилось повернутись.

4.3.3.2 Практичне значення

Така система може бути використана для встановлення на маніпуляційний робот, для його плавної та злагодженої роботи.

4.4 Висновки

На основі розробленої системи можна побудувати десятки, чи навіть, сотні пристроїв на систем, які будуть допомагати людям та компаніям виконувати їх роботу, або взагалі заберуть на себе частину марудної, складної або нецікавої роботи. Розроблений колісний робот може бути використаний як матеріал для майбутніх досліджень у цій сфері та принесе не мало корисної інформації.

5. Охорона праці

5.1. Вступ

Результатом дипломної роботи є клієнт-серверна система віддаленого керування. Система розроблена для використання в ВУЗах, та в інженерних бюро. Для використання розробленої системи мають бути дотримані наступні технічні вимоги:

- організація повинна створити робоче місце для користувача ПЕОМ;
- робоче місце повинно бути оснащене комп'ютером із встановленими необхідними програмами.

Організація повинна забезпечити виконання вимог безпеки праці таким чином, щоб вони відповідали фізіологічним, ергономічним і технічним вимогам відповідно до чинного законодавства України [11]. Дана дипломна робота виконана з урахуванням вимог охорони праці, пожежної та екологічної безпеки виробництва.

5.2. Характеристика технології та організації виробництва, з точки зору охорони праці

5.2.1. План виробничого приміщення

План виробничого приміщення (об'єм і площа, кількість працюючих, розташування обладнання та робочих місць, освітлення (природне та штучне), наявність вентиляції, характеристика підлоги і стін) - рисунок;

Приміщення розташоване в 16-ти поверховому офісному приміщенні на третьому поверсі. Розрахунки умови праці розглянуті на прикладі офісу у

дослідницькому центрі. Офіс розрахований на 7 робочих місць. Основні показники даного приміщення:

* ширина = 6 м, довжина =9 м, висота = 3 м.

* площа приміщення = 54 м^2, об’єм = 162 м^3.

* площа обладнання = 7,75 м^2, об’єм обладнання = 5,996 м^3.

План робочої зони даного приміщення зображений на рис.1.

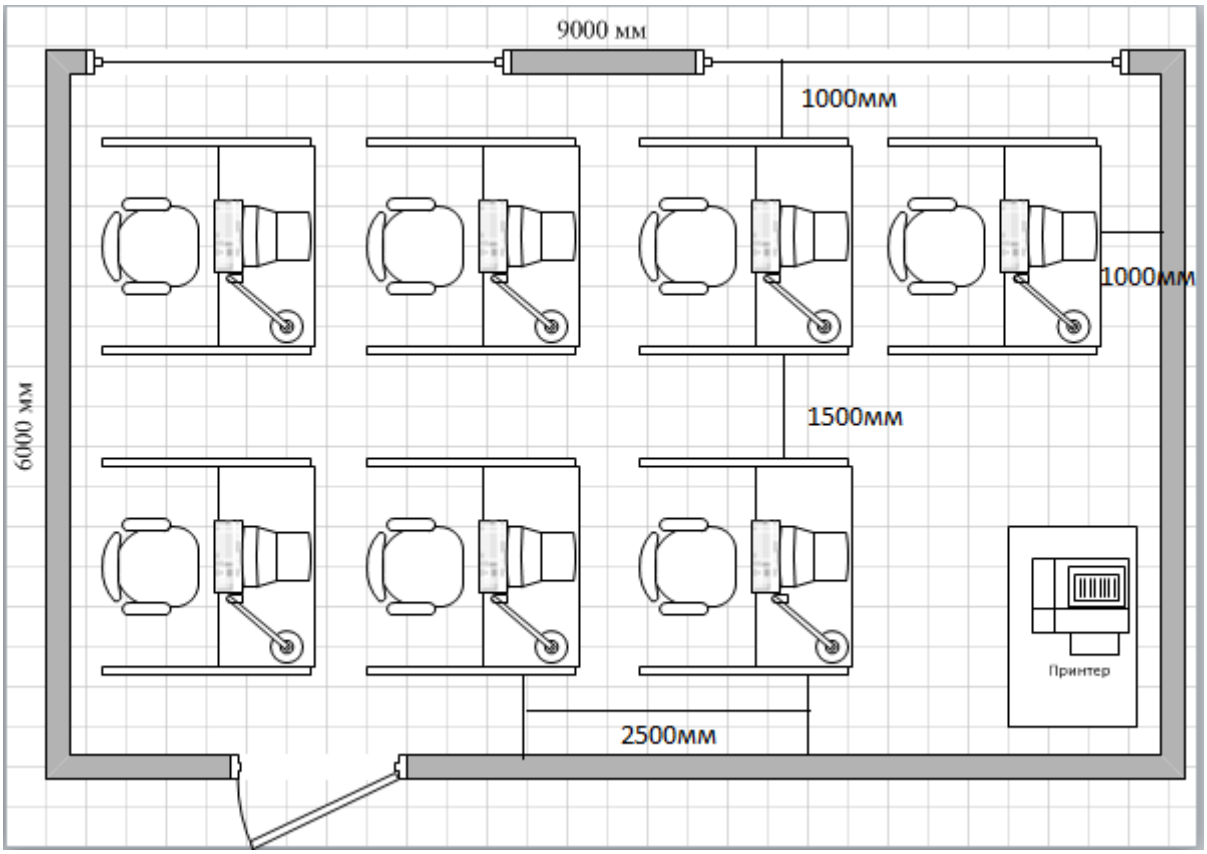

Рисунок 5.1 - Приміщення роботи користувача ПЕОМ [11]

1 – комп’ютерний стіл (75×65×140 см), 2 – комп’ютерний стільчик (37,5×45×100см), 3 – принтер (35,4×29,7×24,6 см), 4 – проектор (34,5×27×11,3 см).

Розраховуємо об’єм та площу приміщення на одного працівника відділу:

V1пр = Vпр–Vобл/n = (162 -5,996)/7= 22, 3 м^3

$S1пр = (54–7,75)/2 = 6,6$ м^2

Відповідність вимогам нормативних документів щодо площі та об’єму на одного працівника, а також правильність розташування обладнання, наводиться у табл. 5.1.

Табл. 5.1 Характеристики робочого місця користувача

Найменування параметра	Значення	
	Нормативне	Фактичне
Площа	не менше 6 м^2	6,6 м^2
Висота	не менше 3 м	3 м
Обсяг на одну людину	20 м^3	22,3 м^3
Висота робочої поверхні на рівнем підлоги	700 мм	700 мм
Оптимальні розміри робочої поверхні	1900 х 900 мм^2	1900 х 900 мм^2
Дальність клавіатури від краю стола	не більше 300 мм	300 мм
Відстань між очима користувача і екраном монітора	400 – 800 мм	600 мм
Регулювання висоти сидіння робочого стільця	400 – 500 мм	450 мм
Глибина сидіння	380 мм	380 мм
Ширина сидіння	400 мм	400 мм
Висота опорної поверхні спинки	300 мм	380 мм
Ширина опорної поверхні спинки	380 мм	380 мм

Отже, фактичні показники вважаються цілком задовільними.

5.2.2. Технологічний процес та робочі операції які виконуються (коротко охарактеризувати);

Технологічний процес являє собою експлуатацію системивіддаленого керування, до якого залучено 7 чоловік. Робочий персонал займає такі посади: інженер-проектувальник – 1 особа, інженер-технолог – 1 особа, програміст-схемотехнік – 2 особи, оператор – 2 людини, тестувальник – 1 особа.

В таблиці 5.2 наведено комп'ютерне обладнання, яке використовується робочим персоналом.

Таблиця 5.2 Комп'ютерне обладнання

Найменування	Кількість	Модель
Системний блок ПК	7	IntelCorei4 / ATI HD4670 2048Mb / Asus P5K Premium SATA II 320 Гб 7200 rpm 32 МбHitachi
Монітор	7	21" Samsung V193DOB
Принтер	1	Samsung ML-1660

5.3. Шкідливі та небезпечні фактори

Таблиця 5.3 Основні джерела небезпеки

Основними джерелами небезпеки шкідливих факторів для працівників в даному відділі є	Електронебезпека
	Низька (висока) температура повітря
	Висока відносна вологість повітря
	Нервово-емоційні перевантаження
	Перенапруження зорового аналізатора
	Пожежа

5.3.1. Несприятливі мікрокліматичні умови

Таблиця 5.4 – Шкідливі фактор - несприятливі мікрокліматичні умови

Шкідливі фактори	Вплив на організм людини	Заходи щодо усунення або зниження їх впливу на працюючих
Висока температура повітря та вологість	Перегрівання тіла, прискорене серцебиття, падіння артеріального тиску	Раціональне розміщення устаткування, встановлення систем вентиляції повітря, спеціальний питний режим
Низька температура	Переохолодження організму, що може призвести до застудного захворювання	Встановлення систем опалювання, спеціальний питний режим

5.3.2. Недостатня освітленість і підвищена яскравість світла

Таблиця 5.5 Характеристика зорової роботи

Параметр	Значення
Розподільна здатність екрану	1024x768
Величина 1 пікселя	0.3 мм
Розмір об’єктів на моніторі (в ср.)	7 – 10 пк (2 – 3мм)
Розмір мінімальних об’єктів	3 пк (0.9 мм)
Контраст	великий
Фон	світлий
Точність зорової роботи	середня точність

Таблиця 5.6 Шкідливі фактори - освітлення

Шкідливі фактори освітленості	Вплив на організм людини	Заходи щодо усунення або зниження їх впливу на працюючих
Недостатня освітленість робочого місця	Передчасне зорове стомлення, зменшення рівня працездатності, збільшення ймовірності механічних помилок	Збільшення кількості світильників, розповсюджених рівномірно по робочій кімнаті.
Підвищена яскравість світла	Знижує світлочутливість очей, зменшення рівня працездатності, збільшення ймовірності механічних помилок	Реорганізація світильників, встановлення жалюзі на вікнах

5.3.3. Шум та вібрація

Таблиця 5.7 Шкідливі фактори - шум

Шкідливі фактори	Вплив на організм людини	Заходи щодо усунення або зниження їх впливу на працюючих
Підвищений рівень шуму на робочому місці	Є джерелом погіршення слуху та зниження продуктивності праці людини, що може призвести до збільшення ймовірності виникнення помилок	Зниження шуму в джерелі створення за рахунок якісного монтажу вузлів комп’ютера, зменшення шуму на шляху поширення

5.3.4. Випромінювання при роботі з обчислювальною технікою

При роботі з комп'ютером людина піддається впливу ряду небезпечних і шкідливих виробничих факторів, приклади яких наведені в таблиці 5.8.

Таблиця 5.8 Шкідливі та небезпечні фактори при роботі з електронно-обчислювальною технікою

Небезпечні і шкідливі виробничі фактори	Влив на організм людини	Заходи щодо усунення або зниження їх впливу на працюючих
Електростатичне поле, магнітне поле, високоінтенсивне іонізуюче випромінювання, ультрафіолетове і м’яке рентгенівське випромінювання, змінне елктромагнітне поле	Значна напруга зорового апарату з проявом скарг на незадоволеність роботою,зменшення працездатності, головні болі, дратівливість, порушення сну, втома	додержуватись відстані від очей до екрану в межах 60 - 80 см; використання додаткових захисних екранів; дотримуватись вимог санітарних норм режиму праці та відпочинку;

5.3.5. Небезпека враження людини електричним струмом

Для захисту електрообладнання від короткого замикання застосовуються запобіжники. На шафах управління, розподільних коробках нанесені знаки електричної напруги. Джерела враження людини електричним струмом наведені у таблиці 5.9 [12].

Таблиця 5.9. Джерела враження людини електричним струмом

Джерела ураження струмом	Конструкторські заходи зменшення небезпеки	Організаційно-технічні заходи зменшення небезпеки
Пошкодження електрично-струменевого шнуру	Захисний кожух	Проведення інструктажу про безпечні методи роботи з електроустаткуванням
Електропроводка в приміщенні	Прихована (АПВ, 5,25 мм)	Проведення інструктажу
Підлога	Ізолююча (лінолеум)	Проведення інструктажу

Загальними рекомендаціями є обслуговування діючих електроустановок, проведення в них оперативних перемикань, ремонтних, монтажних, налагоджувальних робіт виключно електротехнічним персоналом.

5.3.6. Небезпека пожежі

Будинок виконано із залізобетонних плит, тобто він відноситься до II ступеня вогнестійкості [13]. Межа вогнестійкості конструкції 0,5-2,5 год. У приміщенні знаходяться важко горючі тверді й волокнисті речовини й матеріали. За рівнем вибухонебезпечної і пожежної небезпеки приміщення відноситься до категорії В: стелажі з дерева, підлога, касовий апарат.

Джерела виникнення пожежі наведені в таблиці 5.10

Таблиця 5.10. Джерела виникнення пожежі

Джерела виникнення пожежі	Конструкторські заходи зменшення небезпеки	Організаційно-технічні заходи зменшення небезпеки
Перевантаження проводів	2 вогнегасники: вогнегасник CO_2 (вуглекислотний) переносний ВВ-3 (ВВК 2) місткістю балона 3 літри (2 кілограми), призначений для гасіння загорянь різних речовин, горіння яких не може відбуватися без доступу повітря, на один вогнегасник = 107 м^2; датчики пожежної безпеки.	Дотримання технологічного режиму,контроль параметрів температури за допомогою термометра LaCrosse WS8005; використання кондиціонеру LG G24LHT (для кондиціонування і провітрювання) розвантаження електровузлів після виконання роботи, реконструкція електропроводки; ознайомлення з інструкціями по використанню електроприладів; узгоджений план евакуації
Коротке замикання		
Поява великого перехідного опору		

ПЕРЕЛІК ПОСИЛАНЬ

1. Іванов А. О. Теорія автоматичного керування: Підручник. — Дніпропетровськ: Національний гірничий університет. — 2003. — 250 с.
2. Енциклопедія кібернетики. тт. 1, 2. — К.: Головна редакція УРЕ, 1973.
3. Офіційний сайт Arduino. – Режим доступа: http://www.arduino.cc/. Дата доступа: 23.05.2015.
4. Офіційний сайт 8Devices. – Режим доступа: http://www.8devices.com. - Дата доступа: 29.05.2015.
5. Хабр-Хабр. – Режим доступу: http://habrahabr.ru. - Дата доступа: 30.05.2015.
6. Медведев В. С., Лесков А. Г., Ющенко А. С. Системы управления манипуляционных роботов. — М.: Наука, 1978. — 416 с.
7. Фу К., Гонсалес Р., Ли К. Робототехника / Пер. с англ. — М.: Мир, 1989. — 624 с.
8. Форум инновационных технологий – Режим доступу: http://innotech.kiev.ua/. - Дата доступа: 01.06.2015.
9. Client-Server Programming and Applications. — Department of Computer Sciences, Purdue University, West Lafayette, IN 47907: Prentice Hall, 1993.
10. Микушин А. Занимательно о микроконтроллерах. — М.: БХВ-Петербург, 2006
11. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98 (затверджено Постановою Головного державного санітарного лікаря України від 10.12.1998 р. № 7).

12. Державні санітарні норми та правила "Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу" ЗАТВЕРДЖЕНО Наказ Міністерства охорони здоров'я України 08 квітня 2014 року N 248.
13. Санітарні норми виробничого шуму, ультразвуку та інфразвуку. ДСН 3.3.6.037.99 (затверджено Постанова Головного Державного санітарного лікаря України від 1.12.1999 р. № 37).

ДОДАТОК А

Скетч програми Arduino Drive_version_1.3.ino

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
//#include <Servo.h>
#include <PID_v1.h>
//#include <Ultrasonic.h>

#define D1 4          // Направление вращение двигателя 1
#define M1 5          // ШИМ вывод для управления двигателем 1
#define D2 6          // Направление вращение двигателя 2
#define M2 7

#define Trig 8
#define Echo 9
#define ledPin 13

double Setpoint, Input, Output;

PID myPID(&Input, &Output, &Setpoint,15,5,1, DIRECT);//создаем ПИД-регулятор
bool autoMode;
bool autoReverse;

bool ON=false;
bool directionr = 0;
```

```
bool directionl = 0;
int valuel = 0;
int valuer = 0;

YunServer server;
//Servo servol;
//Servo servor;

//Ultrasonic ultrasonic(12, 13);

unsigned int impulseTime=0;
unsigned int distance_sm=0;

void setup() {
  autoMode = false;
  autoReverse = false;
  pinMode(Trig, OUTPUT); //инициируем как выход
  pinMode(Echo, INPUT); //инициируем как вход
  pinMode(ledPin, OUTPUT);

  Serial.begin(9600);

  digitalWrite(ledPin, HIGH);
  Bridge.begin();
  digitalWrite(ledPin, LOW);

  server.begin();
  Setpoint = 30;
```

```
  myPID.SetOutputLimits(0, 255);
  myPID.SetSampleTime(100);
  myPID.SetMode(AUTOMATIC);

  pinMode(D1, OUTPUT);
  pinMode(D2, OUTPUT);

  analogWrite(M1,0);
  analogWrite(M2,0);

}
int counter = 0;
YunClient client;
void loop() {
        if(client){
           client.connect("192.168.240.240", );
           client.print("Counter = "); client.println(counter);
        }
        YunClient tempClient = server.accept();

        counter++;

        if (tempClient) {
           client = tempClient;
           process(client);
           client.print("Counter = "); client.println(counter);
           client.stop();
```

```
}else if(autoMode){
   Serial.print("Setpoint: "); Serial.println(Setpoint);
   Input = distance_sm;
   Serial.print("Input: "); Serial.println(Input);
   // it can only go forward or backward at once
   if((distance_sm - Setpoint) < 2 || (distance_sm - Setpoint) > 2){
      if(distance_sm > Setpoint){
         if(!autoReverse){
            myPID.SetControllerDirection(DIRECT);
            Serial.print(myPID.Compute());
            directionl = 0;
            valuel = 255 - (int)Output;
            directionr = 0;
            valuer = 255 - (int)Output;
         }else{
            if(directionr == 1 && valuer > 220){
               autoReverse = false;
               myPID.SetControllerDirection(DIRECT);
               Serial.print(myPID.Compute());
               directionl = 0;
               valuel = 255 - (int)Output;
               directionr = 0;
               valuer = 255 - (int)Output;
            }else{
               myPID.SetControllerDirection(REVERSE);
               Serial.print(myPID.Compute());
               directionl = 1;
               valuel = (int)Output;
```

```
            directionr = 1;
            valuer = (int)Output;
        }
    }
}else{
    if(autoReverse){
        myPID.SetControllerDirection(REVERSE);
        Serial.print(myPID.Compute());
        directionl = 1;
        valuel = (int)Output;
        directionr = 1;
        valuer = (int)Output;
    }else{
        if(directionr == 0 && valuer < 20){
            autoReverse = true;
            myPID.SetControllerDirection(REVERSE);
            Serial.print(myPID.Compute());
            directionl = 1;
            valuel = (int)Output;
            directionr = 1;
            valuer = (int)Output;
        }else{
            myPID.SetControllerDirection(DIRECT);
            Serial.print(myPID.Compute());
            directionl = 0;
            valuel = 255 - (int)Output;
            directionr = 0;
            valuer = 255 - (int)Output;
```

```
                }
              }
            }
          }
            Serial.print("Output: "); Serial.println(Output);
            Serial.print("valuer: "); Serial.println(valuer);
            Serial.print("valuel: "); Serial.println(valuer);
        }
 //       float dist_cm=ultrasonic.Ranging(CM);
  //      Serial.print("Dist:");
  //      Serial.println(dist_cm);
//        if(dist_cm > 10) {
//           directionl = directionr = 0;
//           valuer = valuel = 0;
//        }

        digitalWrite(D1, directionl);  // Задаем направление вращения
        digitalWrite(D2, directionr);
        //servol.write(valuel);
        //servor.write(valuer);
        analogWrite(M1,valuel);
        analogWrite(M2,valuer);

     //          Serial.print("directr: "); Serial.println(directionr);
     //          Serial.print("directl: "); Serial.println(directionl);
           /*   Serial.print("valuer: "); Serial.println(valuer);
              Serial.print("valuel: "); Serial.println(valuel);*/
    //    client.println(valuel);
```

```
    //      Serial.println(valuel);
      //    client.println(valuer);
    //      Serial.println(valuer);


  getDistance();
  delay(100);
}

void getDistance(){
  digitalWrite(Trig, HIGH);
  /* Подаем импульс на вход trig дальномера */
  delayMicroseconds(10); // равный 10 микросекундам
  digitalWrite(Trig, LOW); // Отключаем
  impulseTime=pulseIn(Echo, HIGH); // Замеряем длину импульса
  distance_sm=impulseTime/58; // Пересчитываем в сантиметры
  if(autoMode && distance_sm > (Setpoint + 100)){
     distance_sm = Setpoint + 100;
  }
  Serial.println(distance_sm); // Выводим на порт
  if (distance_sm<30) // Если расстояние менее 30 сантиметром
  {
    digitalWrite(ledPin, HIGH); // Светодиод горит
  }
  else
  {
    digitalWrite(ledPin, LOW); // иначе не горит
  }
```

```
}

void process(YunClient client) {
  String command = client.readStringUntil('/');

  if (command == "digital") {
    digitalCommand(client);
  }

  if (command == "analog") {
    analogCommand(client);
  }

  if (command == "mode") {
    modeCommand(client);
  }
  if (command == "drive") {
    driveCommand(client);
  }

}


void driveCommand(YunClient client){
  String mode = client.readStringUntil('/');
  Serial.println(autoMode);
  if(mode == "power"){
      ON = (bool)client.parseInt();
```

```
    client.println(ON);
    Serial.println(ON);
    if(ON){
//        if(!servol.attached() || !servor.attached()){
          //servol.attach(M2);
          //servor.attach(M1);
          client.println("Servos were attached");
  //        Serial.println("Servos were attached");
     //   }
    }else{
        //servor.detach();
        //servol.detach();
        analogWrite(M1,0);
        analogWrite(M2,0);
    }
 }else if (mode == "pursuit"){
    int tempDist = client.parseInt();
     if(tempDist){
       Setpoint = tempDist;
       autoMode = true;
     }
 }else{
  if(client.readStringUntil(':') == "P"){
      if(client.parseInt() == 1){
        bool directionr = 0;
        bool directionl = 0;
        int valuel = 0;
        int valuer = 0;
```

```
    ON = ON ^ 1;
    if(ON){
      client.println("Power turned ON");
//      Serial.println("Power turned ON");
    }else{
      client.println("Power turned OFF");
//      Serial.println("Power turned OFF");
    }
   }
}
 if(ON){
  int X = 0, Y = 0, Z = 0;
    bool correct = false;
    client.read();
    if(client.readStringUntil(':') == "Y"){
      Y = client.parseInt();
      client.read();
      if(client.readStringUntil(':') == "X"){
        X = client.parseInt();
        client.read();
        if(client.readStringUntil(':') == "Z"){
          Z = client.parseInt();
          correct = true;
        }
      }
    }
    if(correct){
      Y = dataCorrect(Y);
```

```
X = dataCorrect(X);
Z = dataCorrect(Z);

if(Y >= 0){
   directionl = 0;
   valuel = Y*0.255;
   directionr = 0;
   valuer = Y*0.255;
}else{
   directionl = 1;
   valuel = 255 + Y*0.255;
   directionr = 1;
   valuer = 255 + Y*0.255;
}
if(X >= 0){
   if(Y >= 0){
      valuer = valuel - (int)((float)X*valuel/1000);
   }else{
      valuer = valuel + (int)((float)X*(255-valuel)/1000);
   }
}else{
   if(Y >= 0){
      valuel = valuer + (int)((float)X*valuer/1000);
   }else{
      valuel = valuer - (int)((float)X*(255-valuer)/1000);
   }
}
if(X == 0 && Y == 0){
```

```
                if(Z >= 0){
                    directionl = 0;
                    valuel = Z*0.255;
                    directionr = 1;
                    valuer = 255 - Z*0.255;
                }else{
                    directionr = 0;
                    valuer = 0 - Z*0.255;
                    directionl = 1;
                    valuel = 255 + Z*0.255;
                }

            }
        }
);

    }
  }
}

int dataCorrect(int val){
    if(val < 200 && val > -200){ return 0; }
    if(val > 1000) return 1000;
    if(val < -1000) return -1000;
    return val;
}

void digitalCommand(YunClient client) {
```

```
  int pin, value;

  pin = client.parseInt();

  if (client.read() == '/') {
    value = client.parseInt();
    digitalWrite(pin, value);
  }
  else {
    value = digitalRead(pin);
  }

  client.print(F("Pin D"));
  client.print(pin);
  client.print(F(" set to "));
  client.println(value);

  String key = "D";
  key += pin;
  Bridge.put(key, String(value));
}

void analogCommand(YunClient client) {
  int pin, value;

  pin = client.parseInt();

  if (client.read() == '/') {
```

```
    value = client.parseInt();
    analogWrite(pin, value);

    client.print(F("Pin D"));
    client.print(pin);
    client.print(F(" set to analog "));
    client.println(value);

    String key = "D";
    key += pin;
    Bridge.put(key, String(value));
  }
  else {
    value = analogRead(pin);

    client.print(F("Pin A"));
    client.print(pin);
    client.print(F(" reads analog "));
    client.println(value);

    String key = "A";
    key += pin;
    Bridge.put(key, String(value));
  }
}

void modeCommand(YunClient client) {
  int pin;
```

```
  pin = client.parseInt();

  if (client.read() != '/') {
    client.println(F("error"));
    return;
  }

  String mode = client.readStringUntil('\r');

  if (mode == "input") {
    pinMode(pin, INPUT);
    client.print(F("Pin D"));
    client.print(pin);
    client.print(F(" configured as INPUT!"));
    return;
  }
  if (mode == "output") {
    pinMode(pin, OUTPUT);
    client.print(F("Pin D"));
    client.print(pin);
    client.print(F(" configured as OUTPUT!"));
    return;
  }
  client.print(F("error: invalid mode "));
  client.print(mode);
}
```