МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт Институт Радиоэлектроники и Информационных Технологий
Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника
Профиль образовательной программы Вычислительные машины, комплексы, системы и сети
Кафедра Вычислительные системы и технологии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра
(бакалавра, магистра, специалиста)

Студента Зуевского Ивана Максимовича группы 15-В-2
(Ф.И.О.)
на тему Программно-аппаратная система управления манипулятором
(наименование темы работы)

СТУДЕНТ:
_____ Зуевский И. М.
(подпись) (фамилия, и., о.)
_____
(дата)

РУКОВОДИТЕЛЬ:
_____ Гай В.Е.
(подпись) (фамилия, и., о.)
_____
(дата)

РЕЦЕНЗЕНТ:
_____ _____
(подпись) (фамилия, и., о.)
_____
(дата)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ
_____ Мякиньков А.В.
(подпись) (фамилия, и., о.)
_____
(дата)

КОНСУЛЬТАНТЫ:
1. По _____
_____ _____
(подпись) (фамилия, и., о.)
_____
(дата)

2. По _____
_____ _____
(подпись) (фамилия, и., о.)
_____
(дата)

3. По _____
_____ _____
(подпись) (фамилия, и., о.)
_____
(дата)

ВКР защищена _____
(дата)
протокол № _____
с оценкой _____

Наименование Содержимого	Выпускная квалификационная работа	
Название (тема) ВКР	Программно-аппаратная система управления манипулятором	
ФИО студента (полностью)	Зуевский Иван Максимович	
институт, факультет	Институт радиоэлектроники и информационных технологий	
Выпускающая Кафедра	Вычислительные системы и технологии	
Направление подготовки/специальность	09.03.01	Информатика и вычислительная техника
Профиль подготовки/ магистерская программа/ специализация	Вычислительные машины, комплексы, системы и сети	
Форма обучения	очная	Группа:15-ИВТ-2
ФИО руководителя	Гай Василий Евгеньевич	
Год защиты ВКР	2019	

СОДЕРЖАНИЕ

ВВЕДЕНИЕ……5

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ……..6
1.1. Назначение разработки и область применения…..…..……6
1.2. Технические требования…….…….…...6
1.3. Требования к реализации……..……..……..6
2. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ.....7
2.1. Выбор варианта реализации……..7
2.2. Выбор аппаратных компонентов системы…….…….....…7
2.3. Выбор среды разработки…….10
3. РАЗРАБОТКА СТРУКТУРЫ СИСТЕМЫ…….....11
3.1. Структура манипулятора……...11
3.2. Структура робототехнического устройства манипулятора…….12
3.3. Реализация подсистем…….....13
3.3.1. Исполнительный модуль…….13
3.3.2. Управляющий модуль…….13
3.3.3. Модуль связи…….13
3.4. Структурная схема взаимодействия модулей…….13
4. РАЗРАБОТКА АППАРАТНОЙ ЧАСТИ…….…….…15
4.1. Шина I2C……..……15
4.2. Описание и назначение компонентов системы……..……...16
4.2.1. Arduino Uno……16
4.2.2. PCA9685 и Tower Pro 9g SG90……18
4.2.3. Wemos d1 mini и GY-521……..19
4.3. Соединение аппаратных компонентов систем…….…....20
4.3.1. Модуль связи…….20
4.3.2. Управляющий модуль…….21
4.3.3. Исполнительный модуль……21
4.4. Структурная схема соединения аппаратной части……...22
5. РАЗРАБОТКА ПРОГРАММНОЙ ЧАСТИ…….……....24
5.1. Конфигурирование среды разработки…….24
5.2. Реализация программной части……...…25
5.2.1. Организация беспроводного соединения…….25
5.2.2. Создание UDP-сервера и UDP-клиента……..…….....26
5.2.3. Организация подключения гироскопа к клиенту...……..28
5.2.4. Определение ориентации в пространстве……...…32
5.2.5. Организация перемещения суставов манипулятора……37
6. ТЕСТИРОВАНИЕ…….……41
ЗАКЛЮЧЕНИЕ……..44
СПИСОК ЛИТЕРАТУРЫ……45
ПРИЛОЖЕНИЕ А……..46
ПРИЛОЖЕНИЕ Б……..50

Введение

В данной работе разработана программно-аппаратная система управления манипулятором. Манипулятор - это робот, который воспроизводит функции руки человека. Манипуляторы используют в различных видах промышленности начиная от горной и заканчивая атомной. Все промышленные манипуляторы обладают автоматическим управлением, т.е управляются определённой установленной программой, что позволяет быстро переналадить программное обеспечение, для выполнения других задач.

1. Техническое задание

1.1 Назначение разработки и область применения

Разрабатываемая система предназначена для перемещения суставов манипулятора под определенным углом, благодаря данным, полученных от акселерометра и гироскопа. Данные должны передаваться беспроводным путём, по Wi-Fi.

Для работы системы нужен управляющий модуль, благодаря которому распределяются данные в самом манипуляторе, и модули, один из которых будет выступать в качестве подсистемы связи, который будет отправлять и принимать данные, которые, после этого, поступят на управляющий модуль манипулятора, и исполняющий, с помощью которого будет выполняться перемещение суставов. Область применения данной системы:

- Перемещение, под средством бесконтактного управления.
- Выполнение функций руки человека.

1.2. Технические требования

Разрабатываемая система должна выполнять следующие функции:

На модуле связи:

- Формирование сигнала о перемещении.
- Отправка сигнала на сервер.
- Приём сигнала о перемещении.
- Отправка сигнала на модуль управления.

На управляющем модуле:

- Приём сигнала с сервера.
- Отправка сигнала о перемещении на исполнительный модуль

На исполнительном модуле:

- Приём сигнала с управляющего модуля
- Осуществление перемещения суставов манипулятора

1.3. Требования к реализации.

Требование к технической стороне реализации системы:

- Возможность беспроводного обмена информацией между компонентами системы.
- Возможность переналадки системы для других задач.

2. Анализ технического задания

2.1. Выбор варианта реализации

Разработанная система управляется непосредственно человеком и косвенно взаимодействует с окружающей средой, включает в себя подсистемы(модули):

- Управляющая
- Исполнительная
- Модуль связи

Данный вариант реализации облегчает управление манипулятором, так как для управления не нужен компьютер, а будут нужны только данные с акселерометра и гироскопа.

2.2. Выбор аппаратных компонентов системы.

На данный момент, на рынке существует множество микросхем, с помощью которых реализуется концепция, предложенная мною в данном проекте, перечислю только некоторые из них, между которыми осуществлялся выбор:

- Arduino Uno- это микроконтроллеры, а не полноценные компьютеры. На них нет операционной системы как таковой, Arduino просто выполняет код, интерпретируемый прошивкой. В данном случае вы не имеете в распоряжении базовых инструментов, предоставляемых операционной системой, но, с другой стороны, такое непосредственное выполнение несложного кода протекает проще, а при работе не возникает никаких издержек, связанных с операционной системой. Основное назначение платы Arduino – взаимодействие с сенсорами и устройствами, поэтому Arduino отлично подходит для аппаратных проектов, где требуется просто реагировать на различные сигналы сенсоров и ручной ввод. Может показаться, что в этом нет ничего особенного, однако на деле Arduino – сложная выверенная система, значительно облегчающая управление устройствами. Она отлично подходит именно для сочленения других устройств и исполнительных механизмов, где полновесная операционная система просто не требуется, так как речь идет просто о регистрации действий и реагировании на них. (Стоимость 2270р.)
- Raspberry Pi- является полнофункциональным компьютером. Он обладает всеми атрибутами настоящего компьютера: выделенным процессором, памятью и графическим драйвером для вывода через HDMI. На нем даже работает специальная версия операционной системы Linux. Поэтому на Raspberry Pi легко установить большинство программ для Linux. Хотя в Pi и отсутствует внутреннее хранилище данных, на этом компьютере можно использовать смарт-карты в качестве флэш-памяти, обслуживающей всю систему. Таким образом, можно быстро выгружать для отладки различные версии операционной системы или программных обновлений. Поскольку это устройство обеспечивает независимую соединяемость по сети, его можно

настраивать и для доступа по SSH, либо пересылать на него файлы по протоколу FTP. (Стоимость 4000 рублей).

Рисунок 1. «Arduino Uno»

Рисунок 2. «Raspberry Pi»

С учётом перечисленных технических возможностей и цены, была выбрана платформа Arduino Uno, так как в проекте требуется сочленение устройств(суставов), а данная платформа лучше всего подходит для данной задачи.

Данная платформа обладает недостатком. В отличии от Raspberry Pi, у Arduino Uno нет встроенного Wi-Fi модуля, поэтому нужно было найти на рынке подходящие микросхемы, благодаря которым, возможна организация небольшой сети «Клиент-Сервер». Под такую задачу подходят микросхемы серии ESP8266. В данном проекте используется Wemos D1 mini.

WeMos D1 mini построена на базе 32 разрядного микроконтроллера ESP8266 (он входит в сборку ESP-12F установленную на плате) с интегрированным WiFi модулем (802.11 b/g/n 2.4 ГГц). Так же на плате присутствуют стабилизатор напряжения на 3,3 В, разъем USB типа Micro-B и USB-UART преобразователь на базе чипа CH340G. Микроконтроллер ESP8266 работает на тактовой частоте 80 МГц и обладает оперативной памятью RAM данных на 80 КБ (для хранения значений переменных), и памятью RAM инструкций на 32 КБ. Программы хранятся в flash памяти объемом 4 МБ.

Рисунок 3. «WeMos D1 mini»

Для осуществления перемещения суставов манипулятора понадобятся сервоприводы и расширитель ШИМ. В данной работе используются сервопривод «Tower Pro 9g SG90» и расширитель ШИМ «PCA9685».

Tower Pro 9g SG90 – сервопривод, угол поворота которого ограничен диапазоном от 0 до 180°.

PCA9685 - это 16-ти канальный 12-разрядный контроллер с настраиваемой частотой ШИМ-а в пределах от 24 до 1526 Гц.

Рисунок 4. «Tower Pro 9g SG90»

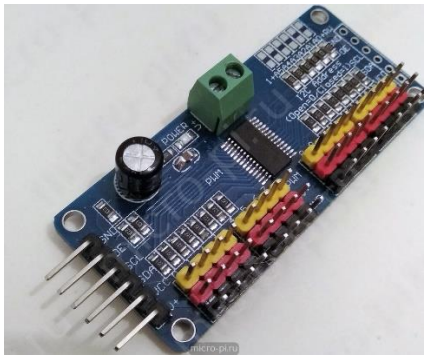

Рисунок 5. «PCA9685»

Для передачи данных о перемещении манипулятора используется акселерометр и гироскоп.

Акселерометр - определяет угол наклона устройства относительно поверхности Земли.

Гироскоп - это приспособление, которое служит для определения ориентации устройства в пространстве, для отслеживания его перемещения.

Существует микросхема, которая выступает одновременно в качестве акселерометра и гироскопа - GY-521. Это модуль с гироскопом, акселерометром и термометром на базе микросхемы MPU-6050 используется для определения положения в пространстве.

Модуль GY-521 построен на базе микросхемы MPU6050. На плате модуля также расположена необходимая обвязка MPU6050, включая подтягивающие резисторы интерфейса I^2C. Гироскоп используется для измерения линейных ускорений, а акселерометр – угловых скоростей. Совместное использование акселерометра и гироскопа позволяет определить движение тела в трехмерном пространстве.

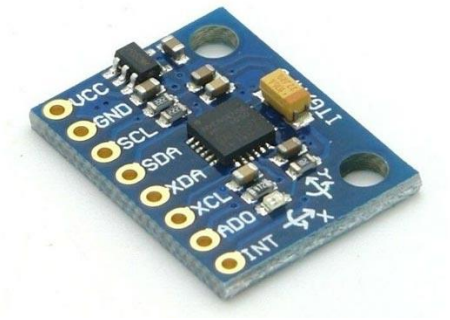

Рисунок 6. «GY-521»

2.3. Выбор среды разработки.

Для плат «Arduino» существует собственная среда разработки «Arduino IDE», данная программа используется для написания программной части системы. Язык программирования C++.

3. Разработка структуры системы

3.1. Структура манипулятора

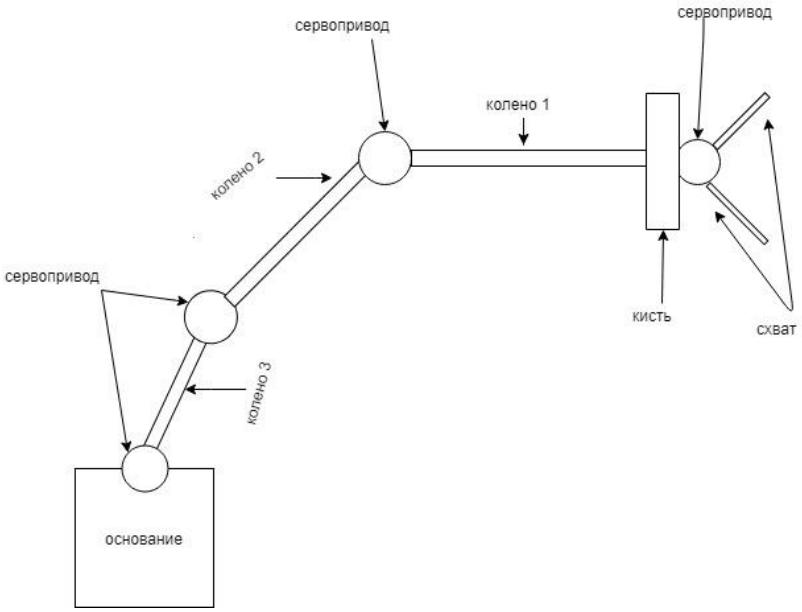

Рисунок 7. «Структура манипулятора»

Манипулятор- устройство, которое воспроизводит функции руки человека. У каждого манипулятора существуют подвижные звенья(суставы), кинематические пары и степени свободы.

Сустав манипулятора- подвижная часть манипулятора, которая обеспечивает позиционирование стрелы относительно плоскости x, y, z.

Кинематическая пара- пара подвижных суставов манипулятора.

Степень свободы- число управляемых осей манипулятора.

На рисунке 8 показана конструкция манипулятора.

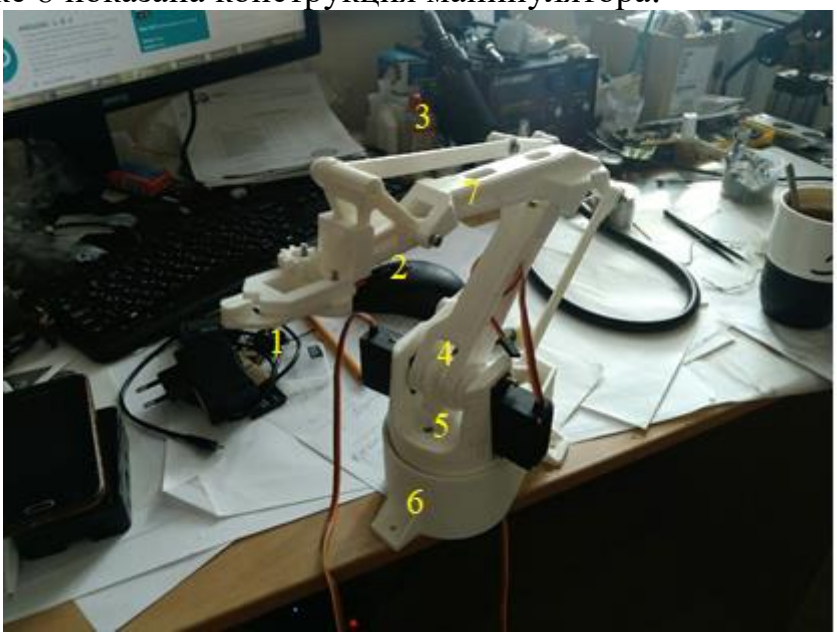

Рисунок 8. «Конструкция манипулятора»

1. Захватное устройство (схват) - конечный узел манипулятора, обеспечивающий захватывание и удержание в определённом положении объекта манипулирования.
2. Кисть.
3. Стрела
4. Привод руки- приводит в движение руку манипулятора.
5. Колонна – служит для поворота руки манипулятора.
6. Основание- конструкция, на которое крепится манипулятор.
7. Рука манипулятора

Кисть и схват являются вторым звеном, рука и привод - первым звеном, колонна третьим звеном.

Звенья манипулятора-последовательное-соединённые твёрдые тела, входящее в состав манипулятора и выполняющие его функциональное назначение.

3.2. Структура системы робототехнического устройства манипулятора

Устройство системы состоит из трёх частей(модулей):

- Управляющей
- Исполнительной
- Модуля связи

Управляющий модуль – это модуль, который на основании поступающих на него команд и заранее запрограммированных в программе, формирует движение суставов манипулятора. Он представляет из себя контроллер, который способен принимать сигналы с других микросхем и преобразовывать их в сигналы управления движением суставов манипулятора.

Исполнительный модуль – это подсистема, которая принимает сигналы от управляющего модуля и приводит в движении суставы манипулятора.

Он должен состоять из сервомоторов и платы расширения.

Модуль связи – модуль, который отвечает за передачу и приём данных в системе. Должен включать в себя наличие беспроводных и проводных каналов связи. Модуль должен использовать Wi-Fi модули, акселерометр, гироскоп.

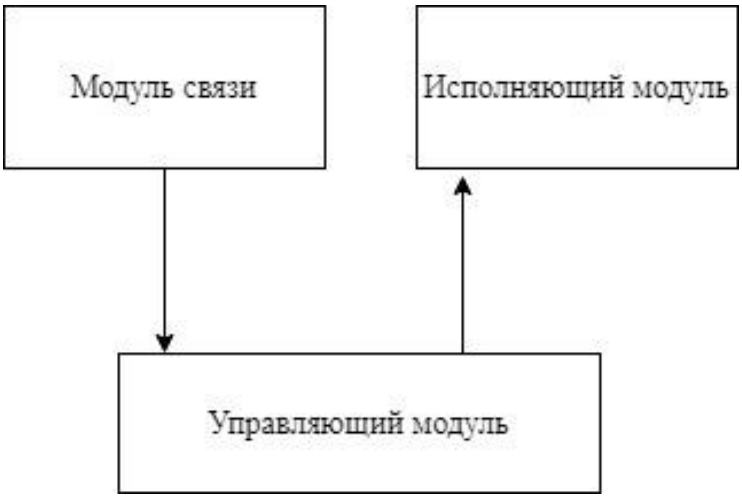

Рисунок 9. «Структурная схема робототехнического устройства»

3.3. Реализация подсистем

3.3.1. Исполнительный модуль

Исполнительная подсистема должна принимать сигнал с управляющего модуля и приводить в движение суставы манипулятора(звенья).

3.3.2. Управляющий модуль

В программно-аппаратной системе в роли управляющего устройства выступает контроллер, выполняющего функцию моста между модулем связи и исполнительной подсистемы, то есть, связывает все подсистемы воедино.

3.3.3. Модуль связи

В данной системе, модуль связи выполняет беспроводную отправку, приём и обработку сигнала, после чего, его отправляет на управляющий модуль.

3.4. Структурная схема взаимодействия модулей

В связи с приведённым выше описанием взаимодействия модулей, структурная схема системы будет иметь следующий вид:

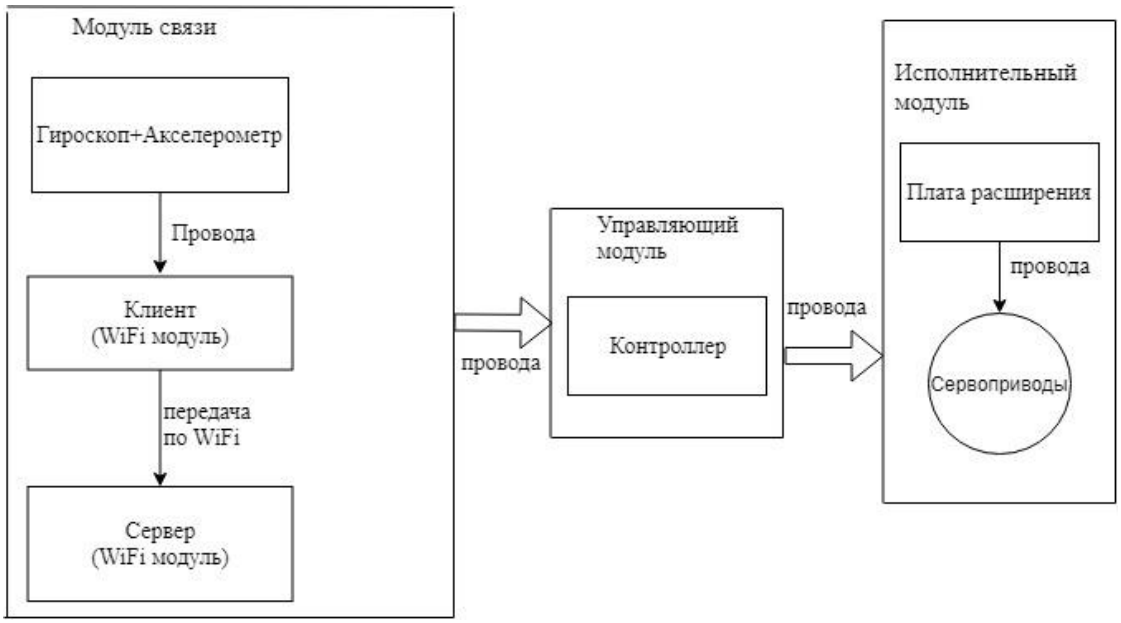

Рисунок 10. «Схема Взаимодействия модулей»

4. Разработка аппаратной части системы

4.1. Шина I2C

Шина I2C состоит из 2х проводов: для данных (SDA — Serial DAta) и для тактов (SCL — Serial CLock).

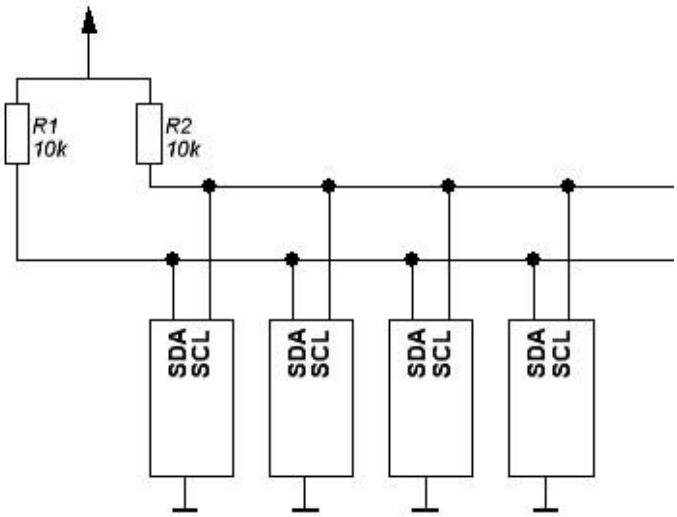

Рисунок 11. «Шина I2C»

Передача данных по шине I2C происходит следующим образом. Сначала идёт проверка- свободна ли линия, то есть равно ли значение SDA и SCL единице, за тем передается адрес устройства, к которому требуется обратиться. При передаче по шине I2C данные считываются при SCL=1, но изменяются при SCL=0. Передача данных проходит, начиная от старшего бита и заканчивая младшим. Далее передается 7 бит адреса, 8й бит R/W определяет чтение 1 или запись 0. После принятия 8го бита от slave должно придти подтверждение (ack, acknowledge), если сигнала нет, то можно сформировать стоп и повторить операцию. На 9й такт master принимает подтверждение, и ждёт, когда SDA будет выставлено в 1. Далее master отправляет байт данных устройству slave, или получает байт данных от этого устройства и отправляет подтверждение. Пример «Общения» между master и slave приведен на рисунке 12.

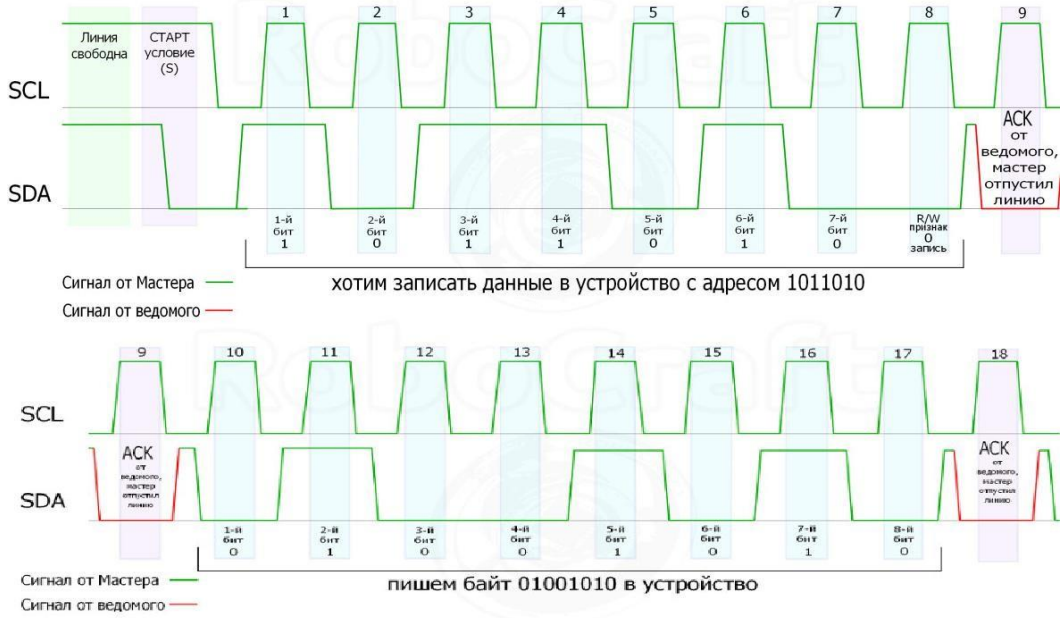

Рисунок 12. «Общение устройств master и slave по шине I2C»

Этот процесс можно описать пошагово: старт, адрес (чтение/запись), подтверждение, данные, стоп. Наглядно этот алгоритм представлен на рисунке 13.

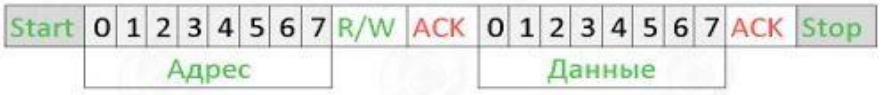

Рисунок 13. «Обмен данными по шине I2C»

В случае, когда данные принимает master, для большинства устройств slave после последнего байта данных master передает команду nack что бы завершить диалог. Такой процесс представлен на рисунке 14.

Рисунок 14. «Чтение данных устройством master»

4.2. Описание и назначение аппаратных компонентов системы

4.2.1. Arduino Uno

Как уже говорилось выше, данная система состоит трёх модулей, каждый из которых выполняет свою функцию, а взаимодействие между ними осуществляет микроконтроллер.

Управление данной системой выполняет плата «Arduino Uno». Приведу описание данной платформы. Информация взята из технической документации.

Таблица 1. «Технические характеристики Arduino Uno»

Микроконтроллер	ATmega328
Рабочее напряжение	5 В
Входное напряжение	7-12 В
Входное напряжение(предельное)	6-20 В
Цифровые входы/выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 Кб (ATmega328) из которых 0.5 Кб используются для загрузчика
ОЗУ	2 Кб (ATmega328)
EEPROM	1 Кб (ATmega328)
Тактовая частота	16 МГц

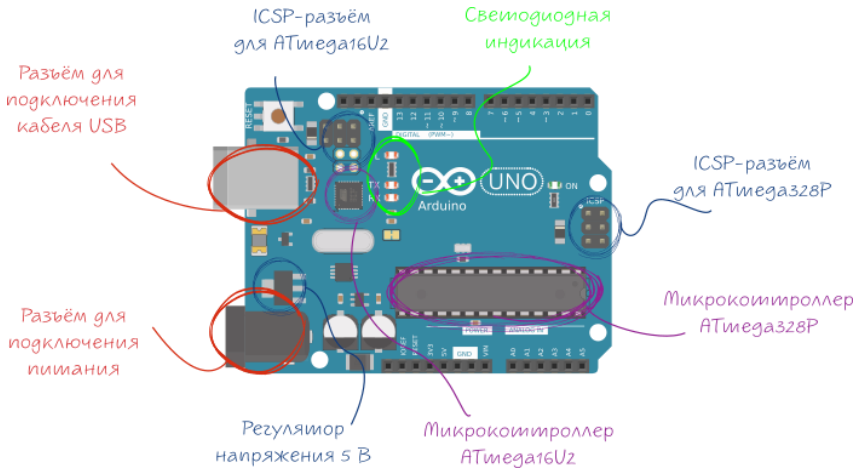

Рисунок 15. «Элементы платы Arduino Uno»

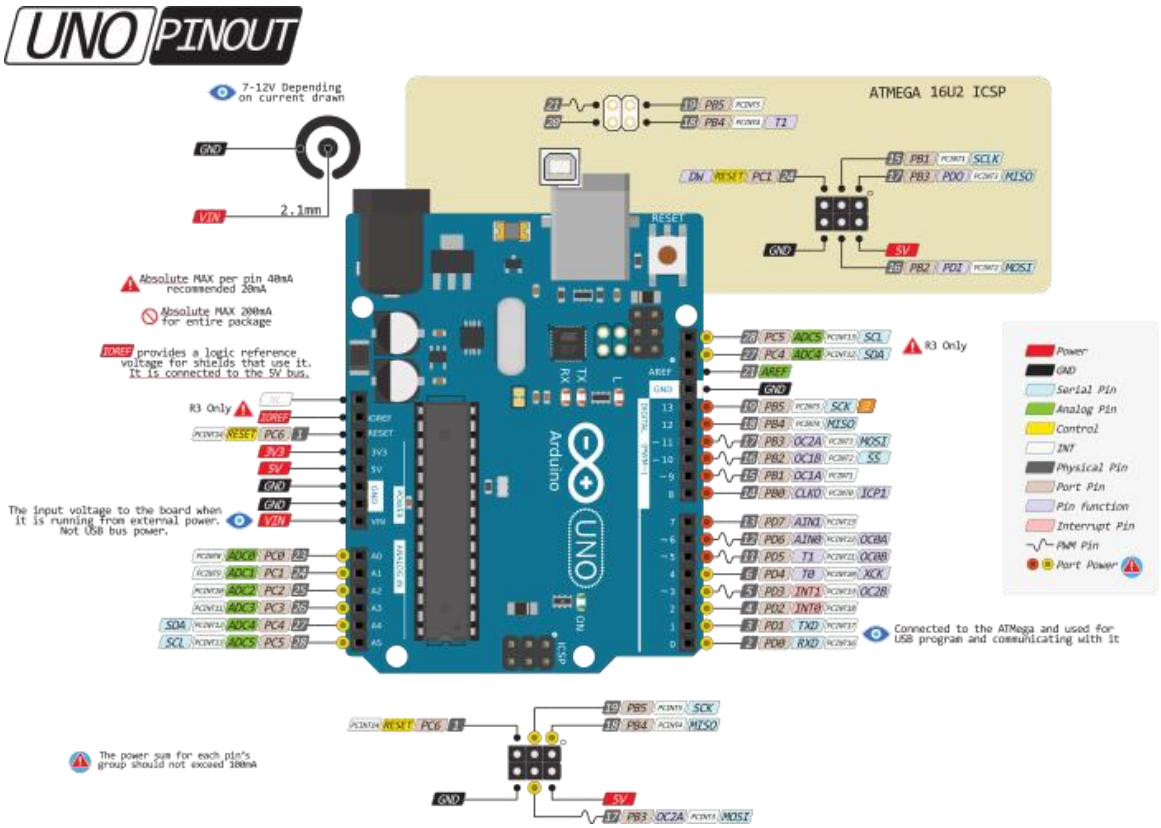

Рисунок 16. «Выводы Arduino Uno»

4.2.2. PCA9685 и Tower Pro 9g SG90

Исполнительная система состоит из 4-х сервоприводов серии Tower Pro 9g SG90 и расширителя ШИМ PCA9685.

PCA9685 — это 16-ти канальный 12-разрядный контроллер с настраиваемой частотой ШИМ-а в пределах от 24 до 1526 Гц. В качестве управляющего сигнала применяется ШИМ сигнал. Для управления PCA9685 используется шина I2C.

Таблица 2. «Выводы PCA9685»

Вывод	*Описание*
GND	общий (минус питания)
OE	разрешение работы выходов модуля
SCL	линия тактирования (интерфейс I2C)
SDA	линия данных (интерфейс I2C)
VCC	плюс питания чипа
V+	плюс питания периферии
PWM (0…15)	выходы ШИМ (широтно-импульсная модуляция)
A0…A5	состояния 0…5 битов адреса чипа на шине I2C

Сервопривод SG 90.

Технические характеристики SG90:

- Скорость отработки команды 0,12с/60 градусов;
- Питание 4,8В;
- Рабочие температуры от -30С до 60 С;
- Размеры 3,2 х 1,2 х 3 см;
- Вес 9 г.
- Угол поворота от 0 до 180 градусов

4.2.3. Wemos d1 mini и GY-521

В модуле связи, за отправку и приём данных о движении, в программно-аппаратной системе отвечают две платформы серии «Wemos d1 mini» и платформа «MPU6050 GY-521». Wemos d1 mini работает на базе микросхемы ESP8266 серии «ESP-12E». Приведу данные о платформах из технической документации:

Таблица 3. «Выводы WeMos d1»

Вывод	*Функция*	*ESP8266 Вывод*
TX	TXD	TXD
RX	RXD	RXD
A0	Аналоговый вход, максимум 3,3 В	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Земля	GND
5V	5V	-
3V3	3.3V	
RST	Reset	RST

MPU6050 GY-521.

Таблица 4. «Технические характеристики GY-521»

Напряжение питания	3 .. 5В (DC)
АЦП	16 бит
Гироскоп (диапазон)	± 250 500 1000 2000 ° / с
Ускорение (диапазон)	± 2 ± 4 ± 8 ± 16g
Интерфейс	I2C
Размеры платы	15x20мм
Вес	20 гр

4.3. Соединение аппаратных компонентов системы

4.3.1. Модуль связи

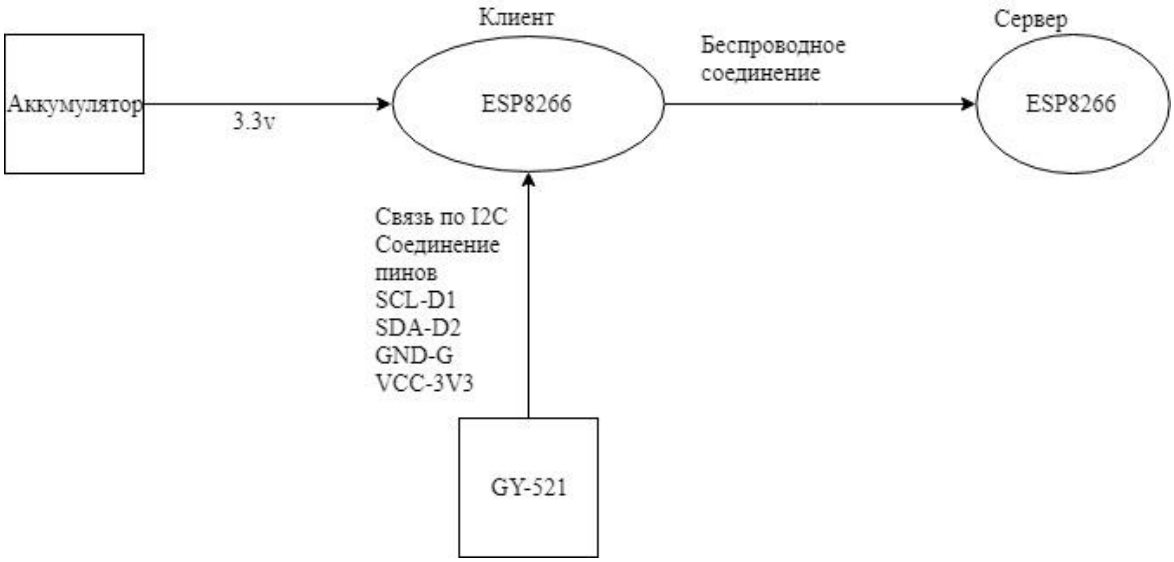

Рисунок 17. «Структурная схема соединения компонентов модуля связи»

В представленной структурной схеме соединения сигнал поступает по шине I2C от гироскопа на esp8266, который, в свою очередь, беспроводным путём поступит на сервер.

Схема не будет работать, если не подать питания. В данном случае используется переносной аккумулятор, можно использовать питание от Micro USB.

Выводы соединения GY-521 и ESP8266(клиент):

- D1(ESP8266)-SDA(GY-521)
- D2(ESP8266)-SCL(GY-521)
- 3V3(ESP8266)-VCC(GY-521)

•G(ESP8266)-GND(GY-521)

Клиентская ESP8266 подключается к серверной ESP с помощью IP адреса, логина и пароля. (подробнее процесс будет описан в пункте 5. Разработка программной части системы).

4.3.2. Управляющий модуль

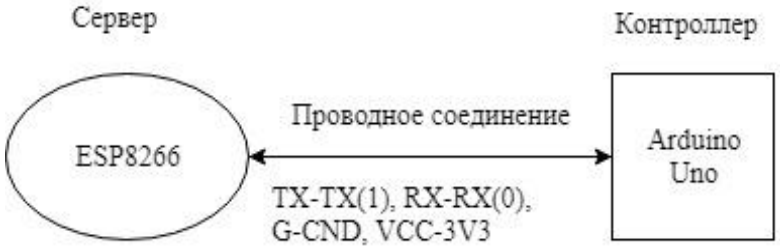

Рисунок 18. «Структурная схема соединения компонентов управляющего модуля системы»

Для корректной работы и передачи сигнала на контроллер с сервера(ESP8266) нужно правильно соединить выводы ESP8266 и Arduino Uno. Соединение будет таким:

- TX(ESP8266) - Digital 1(Arduino)
- RX(ESP8266)- Digital 0(Arduino)
- G(ESP8266) - GND(Arduino)
- VCC(ESP8266) - 3V3(Arduino)

4.3.3. Исполнительный модуль

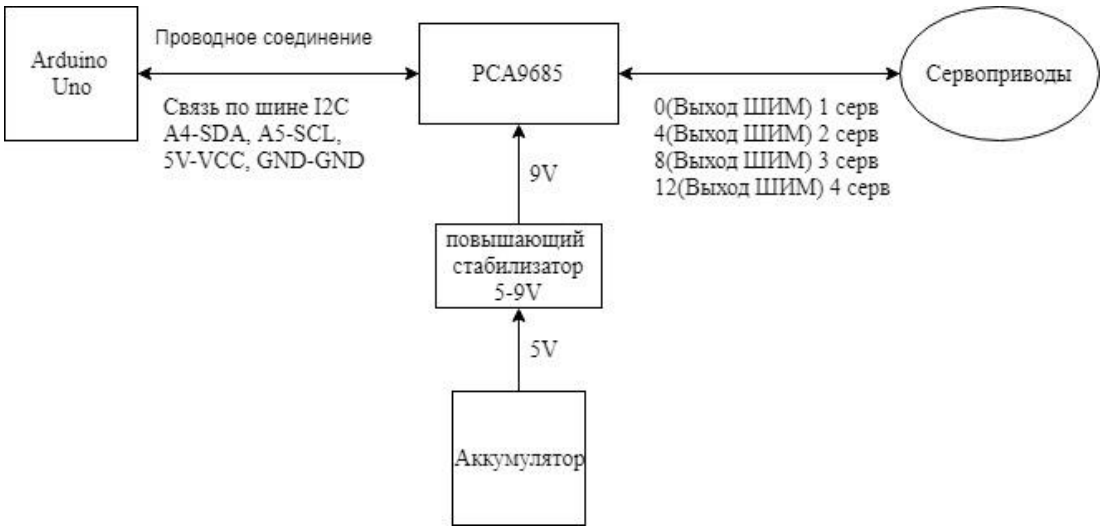

Рисунок 19. «Структурная схема соединения компонентов исполнительного модуля системы»

Связь между Расширителем ШИМ(PCA9685) и Arduino Uno обеспечивает шина I2C. Для того, чтобы питания хватало для всего модуля понадобился дополнительный аккумулятор и повышающий стабилизатор.

Выводы Arduino, отвечающие за линию такта и линию данных, располагаются на пинах A5 и A4, соответственно соединение с выводами PCA9685 будет таким:

- A4-SDA
- A5-SCL
- Digital GND-GND
- 5V-VCC

Сервоприводы присоединяем к выводам 0,4,8,12 расширителя ШИМ.

4.4. Структурная схема аппаратной части

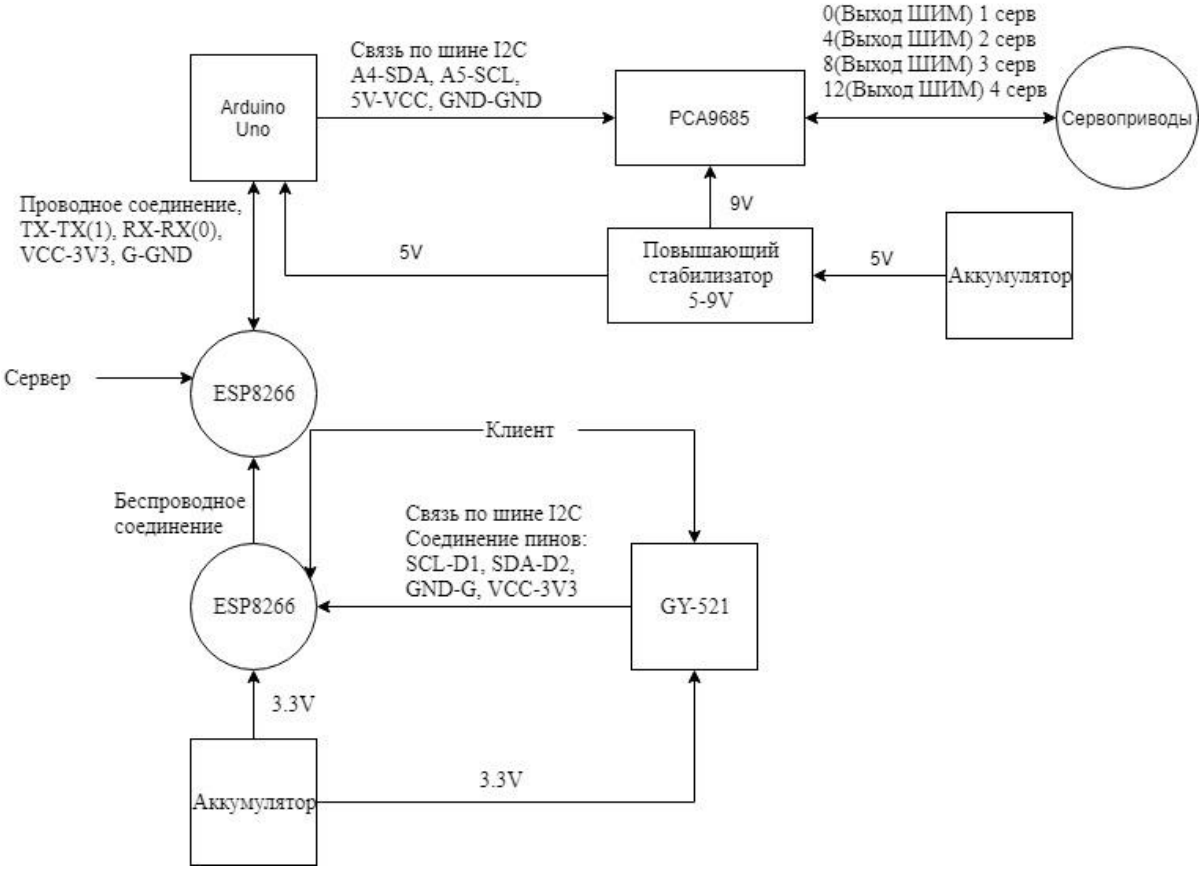

Рисунок 20. «Структурная схема соединения аппаратной части системы»

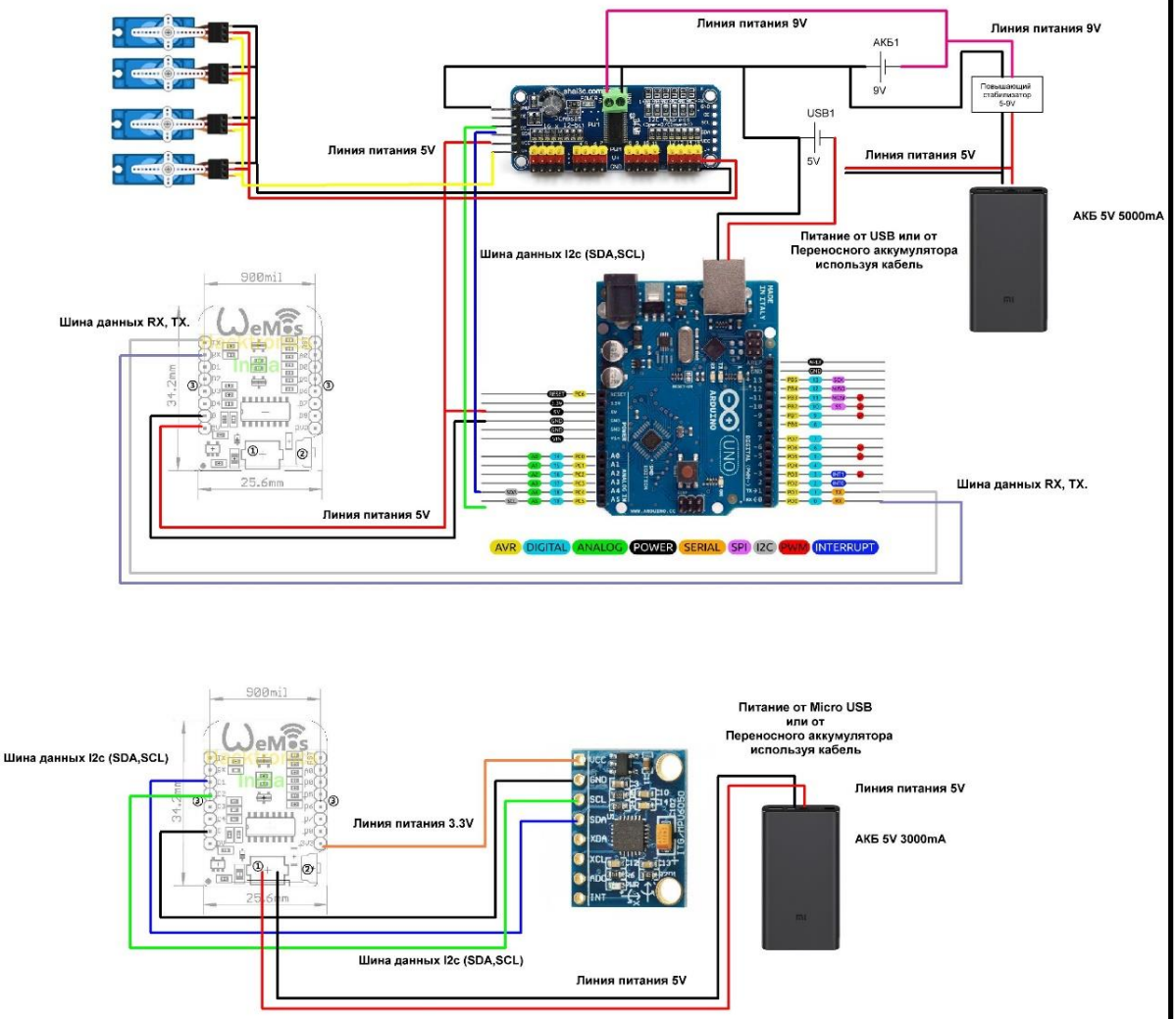

Рисунок 21. «Схема соединения аппаратной части системы»

5. Разработка программной части системы

5.1. Конфигурирование среды разработки

Для разработки программы системы управления манипулятором я использовал Arduino IDE. Программа была скачана с официального сайта Ардуино(https://www.arduino.cc/).

При запуске программы видно следующее:

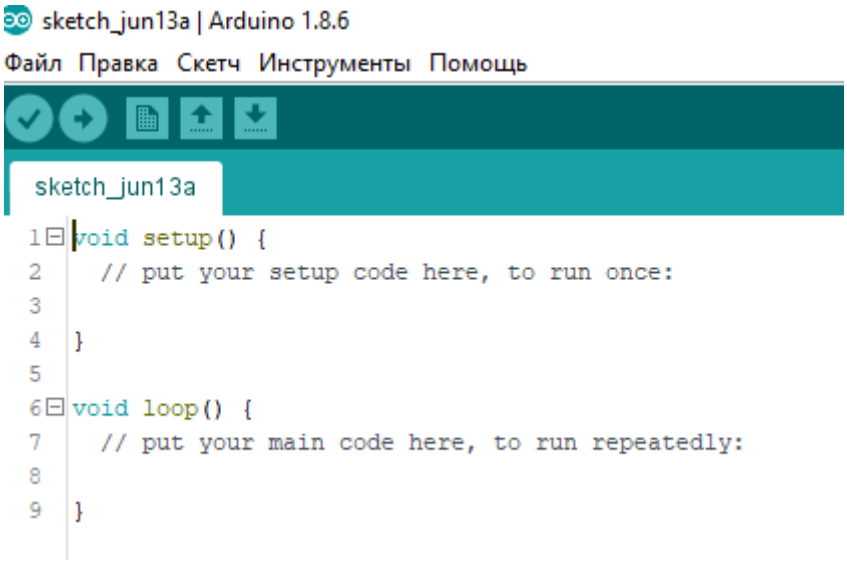

Рисунок 22. «Вид Arduino IDE»

Функция «void setup ()» вызывается при запуске скетча. Её используют для инициализации переменных, режимов выводов, запуска библиотек и т.д. Функция «void setup()» устанавливает начальные значения и запускается только один раз, после каждого включения питания или сброса платы ARDUINO.

Функция «void loop()» это место, куда мы должны поместить команды, которые будут выполняться все время, пока включена плата Arduino. Начав выполнение с первой команды, микроконтроллер дойдет до конца и сразу же перепрыгнет в начало, чтобы повторить ту же последовательность. И так бесконечное число раз (до тех пор, пока на плату будет подан электричество). В данной работе использованы следующие библиотеки:

- ESP8266WiFi- для организации беспроводного соединения (программной точки доступа «сервера» и станции «клиента».)
- WiFiUDP- для беспроводной передачи данных между клиентом и сервером.
- Wire, MPU6050, I2Cdev- для организации работы I2C устройств (GY-521 и PCA9685).
- HPCA9685- для организации работы расширителя ШИМ «PCA9685 и Сервоприводов.
- Kalman-для фильтрации значений с гироскопа и акселерометр.

5.2. Реализация программной части системы

5.2.1. Организация беспроводного соединения

Микросхема ESP8266 может работать в двух режимах:

- Точки доступа(AP)
- Станции(STA)

Точка доступа (AP) представляет собой устройство, которое обеспечивает доступ к Wi-Fi сети с другими устройствами (станции) и соединяет их дальше к проводной сети. ESP8266 может обеспечить аналогичную функциональность, за исключением того, что у него нет интерфейса к проводной сети. Такой режим работы называется программной точкой доступа (soft-AP). Максимальное количество станций, которые могут быть одновременно подключены к программной точке доступа, может быть установлено от 0 до 8.Режим станции (STA) используется для подключения модуля ESP к сети Wi-Fi, установленной точкой доступа. Для создания точки доступа, нужно:

- Глобально объявить имя сети(ssid) ипароль(password),
- Назначить статический IP адрес
- Задать режим работы как «AP-Aplication Point».

Выдержка из кода:

```
const char* ssid = "Lola";
const char* password = "01024567";
……
IPAdrress Server(192,168,4,1);
……
void setup(){
WiFi.softAP(ssid, password)
……
}
```

Функция «WiFi.softAP()» задаёт режим работы программной точки доступа. «const char* ssid = "Lola"», «const char* password = "01024567"»-логин и пароль точки доступа, «IPAdrress Server(192,168,4,1)»- IP-адрес создаваемой точки доступа.

Чтобы сделать станцию достаточно обычного подключения к существующему IP адресу точки доступа, выдержка из кода:

```
const char   *ssid      = "Lola";
const char   *pass      = "01024567";

void initWiFi() {

  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
```

```
  delay(500);
  Serial.print(".");
 }
 Serial.println("");

 Serial.print("Connected to ");

 Serial.println(ssid);
 Serial.print("IP address ");
 Serial.println(WiFi.localIP());
……..
}
```

Для того, чтобы подключится к программной точки доступа:

- Глобально объявил логин и пароль (ssid and password) точки доступа.
- Подключился к заданной точки доступа (WiFi.begin()), используя логин и пароль.

5.2.2. Создание UDP клиента и UDP сервера

Для создания UDP сервера надо:

- Глобально объявить массив, который будет сохранять данные, считанные с клиента.
- Объект класса WiFiUDP, для запуска сервера.
- Локальный порт, по которому будет происходить соединение с сервером.

Пример реализации из программы:

```
unsigned int localport = 2000;
……
WiFiUDP Udp;
……
char buff [24];
……..
void setup() {
……
Serial.println("Starting UDP");
 Udp.begin(localport);
……
}

void loop() {
```

```
  int cb = Udp.parsePacket();
  if (cb) {

Udp.read(buff, 24);

……

}

}
```

Функцией «Udp.begin()» я инициализировал объект, который отвечает за приём UDP, по локальному порту.

Функцией «Udp.parsePacket()» начинается обработка входящего UDP пакета, проверяется наличие пакета и его размер.

Функцией «Udp.read()» считывается пакет и записываются данные в массив.

Для создания UDP клиента нужно:

- Объявить объект класса WiFiUDP, который будет передавать Udp пакеты на сервер
- Массив, в котором будут сохранятся переданные данные
- Локальный порт UDP сервера.

Реализация UDP клиента:

```
……

unsigned int  localport = 2000;

…….

WiFiUDP udp;

char buff [24];

…….

void initWiFi() {

……

udp.begin(localport);

…….

}
```

```
void sendBuffer() {
……..
udp.beginPacket(ServerIP, 2000);
  udp.write(buff, 24);
  udp.endPacket();
}
```

В данном отрывке программного кода показана реализация UDP клиента.

Функция «udp.beginPacket()» запускает соединение для записи данных UDP на удалённое соединение(сервер).

В функции «udp.write()» происходит запись данных на удалённое соединение.

Функцией «udp.endPacket()» завершается запись пакета, после чего, пакет отправляется на сервер.

5.2.3. Организация подключения гироскопа к клиенту

В данной работе гироскоп «GY-521» должен передавать данные о перемещении на сервер, следовательно, он должен являться частью клиента. Для этого:

- Надо задать адрес устройства.
- Глобально объявить переменные осей гироскопа и акселерометра.
- Инициализировать I2C устройство.
- Создать функцию считывания данных с устройства.
- Создать функцию калибровки значений.
- Создать функцию, которая будет заносить в массив данных UDP клиента откалиброванные значения.

Пример программной реализации:

```
……..
const int MPU_addr = 0x68; // I2C address of the MPU-6050
int32_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
float base_x_accel, base_y_accel, base_z_accel, base_x_gyro, base_y_gyro, base_z_gyro;
……..
void calibrate_sensors() {
  int num_readings = 10;
  float x_accel = 0;
```

```
  float          y_accel = 0;
  float          z_accel = 0;
  float          x_gyro = 0;
  float          y_gyro = 0;
  float          z_gyro = 0;
  for (int i = 0; i < num_readings; i++) {
    readData();
    x_accel += AcX;
    y_accel += AcY;

    z_accel += AcZ;
    x_gyro += GyX;
    y_gyro += GyY;
    z_gyro += GyZ;
   delay(100);

}

  x_accel /= num_readings;
  y_accel /= num_readings;
  z_accel /= num_readings;
  x_gyro /= num_readings;
  y_gyro /= num_readings;
  z_gyro /= num_readings;
  base_x_accel = x_accel;
  base_y_accel = y_accel;
  base_z_accel = z_accel;
  base_x_gyro = x_gyro;
  base_y_gyro = y_gyro;
  base_z_gyro = z_gyro;
  Serial.print("base_x_accel = "); Serial.print(base_x_accel);
  Serial.print("base_y_accel = "); Serial.print(base_y_accel);
  Serial.print("base_z_accel = "); Serial.print(base_z_accel);
  Serial.print("base_x_gyro = ");  Serial.print(base_x_gyro);

  Serial.print("base_y_gyro = ");  Serial.print(base_y_gyro);
  Serial.print("base_z_gyro = ");  Serial.println(base_z_gyro);
}
void readData() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true);
  AcX = Wire.read() << 8 | Wire.read();
  AcY = Wire.read() << 8 | Wire.read();
```

```
  AcZ = Wire.read() << 8 | Wire.read();
  Tmp = Wire.read() << 8 | Wire.read();
  GyX = Wire.read() << 8 | Wire.read();
  GyY = Wire.read() << 8 | Wire.read();
  GyZ = Wire.read() << 8 | Wire.read();
}

void fillBuffer() {

  readData();
  GyX = GyX - base_x_gyro;
  GyY = GyY - base_y_gyro;
  GyZ = GyZ - base_z_gyro;
  buff[0] = (AcX & 0xFF000000) >> 24;
  buff[1] = (AcX & 0xFF0000) >> 16;
  buff[2] = (AcX & 0xFF00) >> 8;
  buff[3] = AcX & 0xFF;
  buff[4] = (AcY & 0xFF000000) >> 24;
  buff[5] = (AcY & 0xFF0000) >> 16;
  buff[6] = (AcY & 0xFF00) >> 8;
  buff[7] = AcY & 0xFF;
  buff[8] = (AcZ & 0xFF000000) >> 24;
  buff[9] = (AcZ & 0xFF0000) >> 16;
  buff[10] = (AcZ & 0xFF00) >> 8;
  buff[11] = AcZ & 0xFF;
  buff[12] = (GyX & 0xFF000000) >> 24;
  buff[13] = (GyX & 0xFF0000) >> 16;
  buff[14] = (GyX & 0xFF00) >> 8;
  buff[15] = GyX & 0xFF;
  buff[16] = (GyY & 0xFF000000) >> 24;
  buff[17] = (GyY & 0xFF0000) >> 16;
  buff[18] = (GyY & 0xFF00) >> 8;
  buff[19] = GyY & 0xFF;
  buff[20] = (GyZ & 0xFF000000) >> 24;
  buff[21] = (GyZ & 0xFF0000) >> 16;
  buff[22] = (GyZ & 0xFF00) >> 8;
  buff[23] = GyZ & 0xFF;
}
void initWire() {
  Wire.begin();
  Wire.beginTransmission(MPU_addr);

        Wire.write(0x6B);

  Wire.write(0);    // set to zero (wakes up the MPU-6050)
```

```
  Wire.endTransmission(true);
}

void sendBuffer() {

  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" AcY = "); Serial.print(AcY);
  Serial.print(" AcZ = "); Serial.print(AcZ);
  Serial.print(" GyX = "); Serial.print(GyX);
  Serial.print(" GyY = "); Serial.print(GyY);
  Serial.print(" GyZ = "); Serial.println(GyZ);
  udp.beginPacket(ServerIP, 2000);
  udp.write(buff, 24);
  udp.endPacket();
}
```

Таблица 5. «Назначение переменных»

Переменная	Назначение
AcX	Неоткалиброванные значения акселерометра по оси X
AcY	Неоткалиброванные значения акселерометра по оси Y
AcZ	Неоткалиброванные значения акселерометра по оси Z
GyX	Неоткалиброванные значения гироскопа по оси X
GyY	Неоткалиброванные значения гироскопа по оси Y
GyZ	Неоткалиброванные значения гироскопа по оси Z
base_x_accel	Калиброванные значения акселерометра по оси X
base_y_accel	Калиброванные значения акселерометра по оси Y
base_z_accel	Калиброванные значения акселерометра по оси Z
base_x_gyro	Калиброванные значения гироскопа по оси X
base_y_gyro	Калиброванные значения гироскопа по оси X
base_z_gyro	Калиброванные значения гироскопа по оси X

В функции «void calibrate_sensors()» организован процесс калибровки значений.

Функцией, которая отвечает за инициализацию GY-521, является «void initWire()». В этой функции указывается адрес устройства с которым мы будем говорить: «Wire.begintransmission(mpu_addr)». Чтобы запросить информацию с регистра 0, отправляют 0 с помощью «Wire.write(0x6B)» эта функция посылает запрос на ведомое устройство и записывает полученные от него данные. Если вызывается между «beginTransmission()» и «endTransmission()», то записывается последовательность байт, подготовленная для передачи на ведомое устройство. Возвращает количество записанных байт.

Функция «void readData()» отвечает за считывание не откалиброванных значений с гироскопа и акселерометра. В данной функции аналогично с функцией «initWire()» начинаем обмен данными между устройствами:

«Wire.beginTransmission(MPU_addr)», «Wire.write(0x3B)»,

«Wire.endTransmission(false)». Затем запрашиваются показания с 14 адресов с помощью функции «Wire.requestFrom(MPU_addr,14,true)»,

и указываются каким переменным присвоить значения, полученные от этих адресов:

```
AcX = Wire.read() << 8 | Wire.read();
AcY = Wire.read() << 8 | Wire.read();  AcZ = Wire.read() << 8 | Wire.read();
Tmp = Wire.read() << 8 | Wire.read();
GyX = Wire.read() << 8 | Wire.read();
GyY = Wire.read() << 8 | Wire.read();
GyZ = Wire.read() << 8 | Wire.read();
```

Каждое значение складывается из показаний от двух адресов: больших и малых значений, малые значения, с 8 битным сдвигом, складываются с большими. Таким образом получают сырые данные, которые в таком виде не дают информацию об ориентации тела в пространстве.

В функции «void fillBuffer()» организован процесс занесения в массив данных UDP клиента откалиброванных значений, которые потом будут пересланы серверу.

Отправка откалиброванных значений акселерометра и гироскопа организована в функции «void sendBuffer()».

5.2.4. Определение ориентации в пространстве

С клиента на сервер поступают «сырые данные» от «GY-521».

Гироскоп измеряет и выводит значения мгновенной угловой скорости с разрешением, заданным в настройках. Показания будут приходить только при повороте, при наличии линейной скорости или отсутствии воздействия, гироскоп будет выдавать нулевые значения. Для установления ориентации объекта в пространстве нужно мгновенное значение угловой скорости умножить на промежуток времени между опросами датчика гироскопа. Далее каждое полученное значение поворота нужно сложить с предыдущим. Таким

образом, при возвращении датчика в исходное положение значение отклонения будет равно нулю. При отклонении от исходного положения, выполнив вышеперечисленные действия для трех осей, можно получить ориентацию тела в пространстве. Так как гироскоп не идеален и имеет некоторую погрешность, эта погрешность интегрируется вместе с показанием угловой скорости и при длительном использовании погрешность определения ориентации тела становится значительной, поэтому гироскоп используют с акселерометром.

Фильтр Калмана имеет возможность сгладить несколько показаний датчиков с балансом шумоподавления и приспособляемости фильтра

к изменениям внутри системы. Для этого необходимо знать состояние системы, основанное на некоторых шумных датчиках. Имея показания датчиков, система может предсказывать собственное состояние, исходя из данных, полученных на последнем шаге, и, используя скорректированные данные, сделать новые прогнозы для следующего состояния системы:

$$\widehat{X_k} = \widehat{X_{\overline{k}}} + K_k\left(y_k - h\left(\widehat{X_{\overline{k}}}\right)\right)$$

где $\widehat{X_k}$ – новая исправленная оценка состояния системы, $\widehat{X_k}$ – предыдущая оценка состояния системы, K_k – коэффициент усиления (на данный момент, это просто некоторая константа), y_k – измерения датчика, h – функция измерения преобразования вектора состояния системы, $h\left(\widehat{X_{\overline{k}}}\right)$ – предсказание измерения, $(y_k - h(\widehat{X_k}))$ – ошибка предсказания измерения или вектор инноваций.

Следующий шаг – это оценка, основанная на новой исправленной оценке и измерении показаний датчика:

$$\widehat{X_{\overline{k+1}}} = f\left(\widehat{X_{\overline{k}}}, u_k\right)$$

где u_k – фактическое измерение с датчика, $\widehat{X_{\overline{k}}}$, – получаем из уравнения, $\widehat{X_{\overline{k+1}}}$ – новое прогнозируемое состояние системы, f – функция, которая связывает предыдущее состояние системы и нынешние показания датчика.

Фильтр Калмана используется для того, чтобы избавиться от дрейфа нуля гироскопа. Для этого необходимо знать дрейф нуля на выходе, который нужно скорректировать сигналом на входе. Так же необходимо знать угол смещения системы, поэтому сигнал с выхода датчика необходимо перевести из радиан в градусы:

$$\Theta_{k+1} = \Theta_k + \omega_k\delta;$$
$$\omega_{k+1} = \omega_k + (y_k - \text{bias}_k)\,\delta;$$
$$\omega_{k+1} = \omega_k;$$
$$\omega_{k+1} = y_k - \text{bias}_k;$$
$$\text{bias}_{k+1} = \text{bias}_k;$$

где Θ – угол; ω – угловая скорость; δ – размер выборки; bias – угловая скорость смещение гироскопа; y – выходной сигнал с гироскопа. Получим

выходной сигнал в виде суммы угловой скорости смещения гироскопа и угловой скорости вращения:

$$y_k^{gyro} = \omega_k + \text{bias}_k$$

Видно, что есть как минимум два пути для модулирования Θ и ω. Тем не менее, если выбрать неправильный набор уравнений, фильтр Калмана может не работать. Использование в коде уравнений

$$\omega_{k+1} = \omega_k + (y_k - \text{bias}_k)\,\delta,$$

$$\omega_{k+1} = y_k - \text{bias}_k$$

поможет избежать нелинейности в системе, которая может привести к появлению расходящейся ошибки. Необходимо взять производную от коэффициента усиления по отношению к погрешности, чтобы подать нужное усиление и таким образом минимизировать ошибку. Получим выходной сигнал с гироскопа. Обновим H используя выходной сигнал и сигнал со входа:

$$K = PH^{`}(HPH^T + VRV^T)^{-1};$$

$$P = (I - KH)P;$$

$$x = x + K(y - h(x)).$$

Сохраним исправленные x для системы. Далее перепишем x с предсказанием следующего шага. Обновим F используя новые значения:

$$P = FPF^T + WQW^T,$$

$$x = f(x, y),$$

здесь K – коэффициент усиления Калмана, который имеет размерность m x n, где m – количество переменных состояния, а n – количество датчиков. K показывает число настроечных состояний системы, исходя из разницы между измерением и предсказанием измерения. Q – дисперсия шума в состоянии, матрица размером m x m показывает число ошибок на каждое состояние системы. R – дисперсия шума при изменении состояния системы, матрица размером $n\ x\ n$. P – матрица, которая приравнивается к матрице Q перед итерацией. H – матрица для преобразования из старого состояния к состоянию измерения с помощью функции $h(x)$. Если уравнение является нелинейным, то просто берётся его частная производная относительно измеряемых переменных. V – это частная производная от функции выходного сигнала по отношению к выходному шуму. Обычно это единичная матрица размерностью $n\ x\ n$. I – это единичная матрица размерностью $m\ x\ m$. F – матрица преобразования в новое состояние с учётом старого состояния системы. Это частная производная функции P по отношению к каждой переменной состояния, если уравнение не является линейным. W – частная производная состояния функции f по отношению к состоянию шума, как правило, это единичная матрица размерностью $m\ x\ m$. Важно применять эти уравнения в точной последовательности. Учитывая выходной сигнал, делается коррекция текущего состояния и затем используется исправленное состояние, чтобы сделать предсказание нового состояния. Матрицы H и F – это постоянные матрицы.

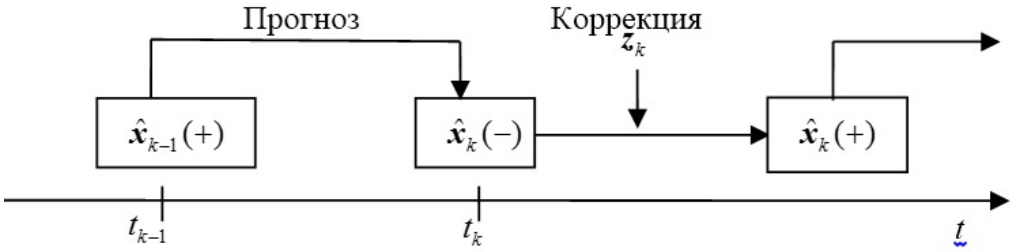

Рисунок 23. «Схема принципа работы фильтра Калмана»

В данной работе, для упрощения математических вычислений фильтра Калмана, используется библиотека «Kalman.h».

С помощью акселерометра можно получить ускорения трех осей датчика. Преобразуя данные, с помощью геометрии, можно также получить ориентацию объекта в пространстве. Ориентация объекта может искажаться при движении датчика в линейных направлениях. Данные, получаемые от акселерометра всегда достаточно точные, так как не зависят ни от времени, ни от характера воздействия, однако недостаток в том, что данные имеют шум в некотором диапазоне данных, то есть с точностью до десятых долей градуса измерять угол невозможно. Акселерометр используется для измерения линейных ускорений, следовательно, можно определить угол отклонения оси, зная значение проекции g на эту ось. Для получения положения тела с помощью акселерометра нужно найти угол между осью, ортогональной искомой оси и осью Z, сдвинутый на π. Этот угол будет измеряться в радианах и, если требуется, может быть переведен в градусы. Эти действия можно выполнить следующим образом:«AcYangle = (atan2(AcX, AcZ)+PI)*RAD_TO_DEG», где AcYangle – отклонение тела по оси Y. Так же можно применить другой метод. Известно, что коэффициент пропорциональности между градусами угла и значением, получаемым от соответствующих оси адресам будет 177, тогда можно разделить входной сигнал на 177 и получить значение наклона оси в градусах. Преимущества измерения акселерометром в простоте использования этого прибора и получении текущего положения тела в каждый момент времени. Недостаток заключается в низкой точности из-за шумов и погрешности при линейных перемещениях прибора. Таким образом определение ориентации тела с помощью акселерометра допустимо при не высоких требованиях к точности и отсутствии линейных перемещений. После преобразований показаний акселерометра следует сделать задержку в 1мкс, так как акселерометр работает с частотой 1 Мгц.

Для получения данных с гироскопа, сначала указать начальное положение тела, которое будет определять акселерометр, затем измерять угловую скорость ω через интервалы времени Δt. Тогда ω × Δt = изменение угла. Проблема с этим подходом заключается в том, что мы интегрируем. Многократное суммирование приращения ω × Δt приведет к увеличивающейся со временем ошибке. Это является причиной гироскопического дрейфа.

Деление сырых данных гироскопа на 131 дает угловую скорость в градусах в секунду. 131 – коэффициент чувствительности гироскопа в заданном режиме 250 град/с. Поскольку у него АЦП 16 бит, то модуль максимального необработанного значения равен 32767.

Теперь 32767 / 250 = 131 условных единиц на градус в секунду.

То есть, если необработанное значение равно 131, то угловая скорость равна 1 градус в секунду. С помощью этих данных можно получить положение объекта.

Для этого мгновенное значение угловой скорости умножим на промежуток времени между опросами датчика гироскопа. Например, разрешение 2000 градусов в секунду, время между опросами 0,1 с, значение мгновенной скорости 210, значит 210·0,1=21 – за это время произошел поворот на 21 градус. Далее каждое полученное значение нужно сложить с предыдущим. Такие действия можно реализовать в функции «readData» следующим образом:

«double gyroxrate = (double)GyX/131.0»,

«GyroskopX1angle = gyroxrate*((double)(micros()–timer)/1000000)», где gyroxrate – угловая скорость по оси *X*, GyroskopX1angle – угловое перемещение тела по оси,

«((double)(micros()– timer)/1000000)» – промежуток времени одного опроса.

Чтобы убрать шумы следует написать условие для данных, которые мы суммируем: «GyroskopX2angle = sqrt(pow(GyroskopX2angle,2))», «(GyroskopX2angle> 0.5)»

«{ xxangle += gyroXrate*((double)(micros()–timer)/1000000); }else»

«{xxangle += 0;}»

То есть, если значение угла по модулю менее 0.5 градусов, то оно считается шумом. Преимущества этого метода в том, что на измерения не влияют линейные перемещения и внешние воздействия, а также высокая точность на коротких промежутках времени. Недостатком является то, что измерения ориентации тела являются косвенными и ошибка накапливается с течением времени, что не позволяет длительно использовать этот прибор.

Исходя из вышеперечисленных свойств приборов следует что для повышения точности прибора следует использовать гироскоп и акселерометр совместно.

Начальное значение угла гироскопа следует задавать как значения, снимаемые акселерометром. Одним из способов такого использования является комплементарный фильтр. Принцип его работы в следующем:

Угол фильтра = α × (Угол от гироскопа) + (1 − α) × (Угол от акселерометра)

где $\alpha=\tau/(\tau\cdot\Delta t)$.

Угол от гироскопа = (Последний угол, измеренный фильтром) + $\omega\times\Delta t$, где Δt – время выборки, τ = постоянная времени превышающая интервалы между шумами, ω – угловая скорость, α – коэффициент фильтра.

Угол фильтра – отфильтрованный, результирующий угол наклона. Так как от гироскопа, с течением времени измерений, увеличивается погрешность, необходимо возвращать показания гироскопа к показаниям акселерометра, при большом расхождении показаний этих устройств.

Недостаток заключается в том, что при возврате значений к показаниям акселерометра происходит скачек в результирующем значении, что приводит к нестабильности отфильтрованных значений. Решение этих проблем заключается в использовании фильтра Калмана.

5.2.5 Организация перемещения суставов манипулятора

После фильтрации значений гироскопа и акселерометра, были получены углы наклона по осям X и Y. Движение вперёд-назад будет происходить по оси Y, а движение влево-вправо по оси X. Если значение угла наклона по оси X будет отрицательным, то движение будет налево, а если положительным, то вправо. Та же и с осью Y, если значение угла наклона положительное, то движение вперёд, иначе-назад. Но, прежде чем присваивать значения углов для сервоприводов, их нужно откалибровать. Для этого нужно взять определённые промежутки и привести их к одному значению угла поворота сервомотора. То есть, организовать выборку положений манипулятора относительно осей.

Движение сервоприводов осуществляется с помощью ШИМ.

Широтно-импульсная модуляция- процесс, с помощью которого управляется мощность, подводимая к нагрузке, путём изменения скважности импульсов, при постоянной частоте. ШИМ представляет из себя периодический импульсный сигнал. Принцип работы заключается в сравнении двух видов сигналов:

- Uоп – опорное (пилообразное, треугольное) напряжение;
- Uупр – входное постоянное напряжение.

Сигналы поступают на компаратор, где они сравниваются, а при их пересечении возникает / исчезает (или становится отрицательным) сигнал на выходе ШИМ. Выходное напряжение Uвых ШИМ имеет вид импульсов, изменяя их длительность, регулируется среднее значение напряжения (Ud) на выходе ШИМ:

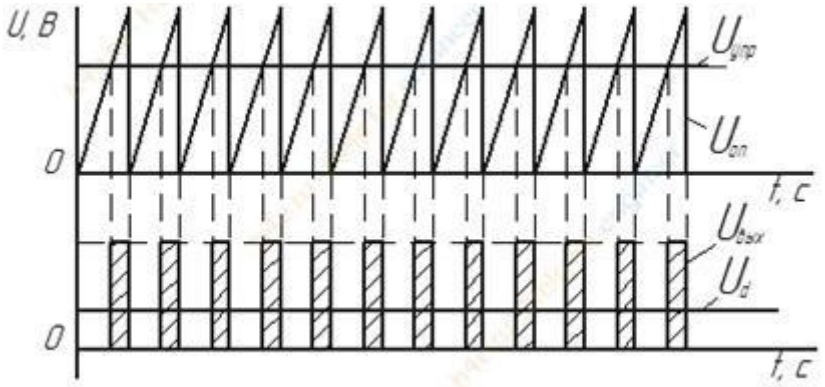

Рисунок 24. «Скважность сигнала при однополярной ШИМ»

Однополярная модуляция означает, что происходит формирование импульсов только положительной величины и имеет нулевое значение напряжения.

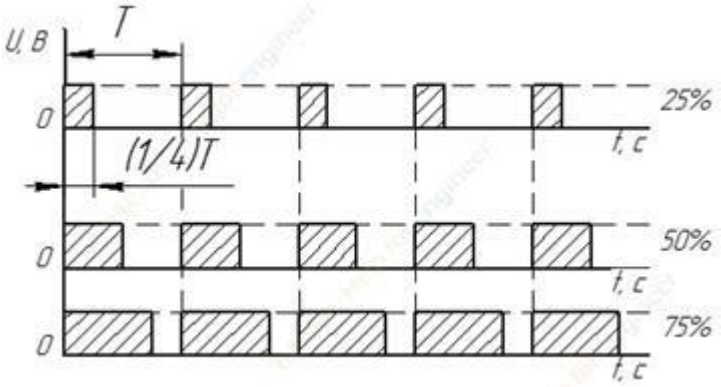

Рисунок 25. «Скважность импульсов»

Если сформированный таким образом сигнал подать на объект, обладающий фильтрующими свойствами, на сервопривод, то объект будет использовать среднюю мощность сигнала.Т.е. мощность, потребляемая объектом управления, пропорциональна скважности сигнала ШИМ, при условии, что период импульсов ШИМ на порядок меньше минимальной постоянной времени объекта. Преимущество использования ШИМ — это легкость изменения величины напряжения при минимальных потерях.

Реализация в программе:

```
#define I2CAdd  0x40// адрес расширителя ШИМ
……..
HCPCA9685 HCPCA9685(I2CAdd);
……
 void setup(){
HCPCA9685.Init(SERVO_MODE);
……..
}
 void readData(){
……
if(kalAngleX<=45 && kalAngleX>=0){
kalAngleX=45;
```

```
HCPCA9685.Servo(0, kalAngleX);

HCPCA9685.Servo(4,kalAngleX);
if(kalAngleX<=0){
kalAngleX=10;
HCPCA9685.Servo(0, kalAngleX);
HCPCA9685.Servo(4,kalAngleX);
if(kalAngleY<=45 && kalAngleY >=0){
kalAngleY=45;

HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);

if(kalAngleY<=0){
kalAngleY=10;
HCPCA9685.Servo(8, kalAngleY);

HCPCA9685.Servo(12,kalAngleY);
}
}
}
}


while(kalAngleX>45&&kalAngleX<=60){

kalAngleX++;

HCPCA9685.Servo(0, kalAngleX);
HCPCA9685.Servo(4,kalAngleX);
}

    while(kalAngleY>45&&kalAngleY<=60){

    kalAngleY++;

HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
}
if(kalAngleX>60 && kalAngleX<=90){
kalAngleX=90;
HCPCA9685.Servo(0, kalAngleX);
HCPCA9685.Servo(4,kalAngleX);
if(kalAngleX>90 && kalAngleX<=120){
kalAngleX=120;
HCPCA9685.Servo(0, kalAngleX);
```

```
HCPCA9685.Servo(4,kalAngleX);
}
}
if(kalAngleY>60 && kalAngleY<=90){
kalAngleY=90;
HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
if(kalAngleY>90 && kalAngleY<=120){
kalAngleY=120;
HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
}
}
if(kalAngleX>120 && kalAngleY>120){
kalAngleX=10;
kalAngleY=10;
HCPCA9685.Servo(0, kalAngleX);
HCPCA9685.Servo(4,kalAngleX);

HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
}
```

В функции «HCPCA9685.Servo(0, kalAngleX)» 0- это вывод расширителя ШИМ, к которому подключён сервопривод, а kalAngleX- угол, на который повернётся сервопривод.

В результате данной калибровки, суставы манипулятора будут перемещаться, относительно своей оси, на 45, 60, 90 и 120 градусов.

Если угол наклона для осей превысит 120 градусов, то суставы сбросятся в ноль. (нулевое положение для сервоприводов-10 градусов).

6. Тестирование

Тестирование программно-аппаратной системы управления - это процесс исследования, испытания данной системы, который имеет следующие цели:

- Демонстрация соответствия требованиям
- Выявление изъянов

Одной из задач, в процессе разработки программно-аппаратной системы, была фильтрация «сырых значений» полученных с гироскопа и акселерометра и получения стабильных показаний угла наклона по осям, так как для перемещения сустава нужны стабильные показания углов, без сильных отклонений. В процессе фильтрации нужно было достигнуть результата, который позволял снизить отклонение в значениях до 1. В результате тестирования, получены следующие показания:

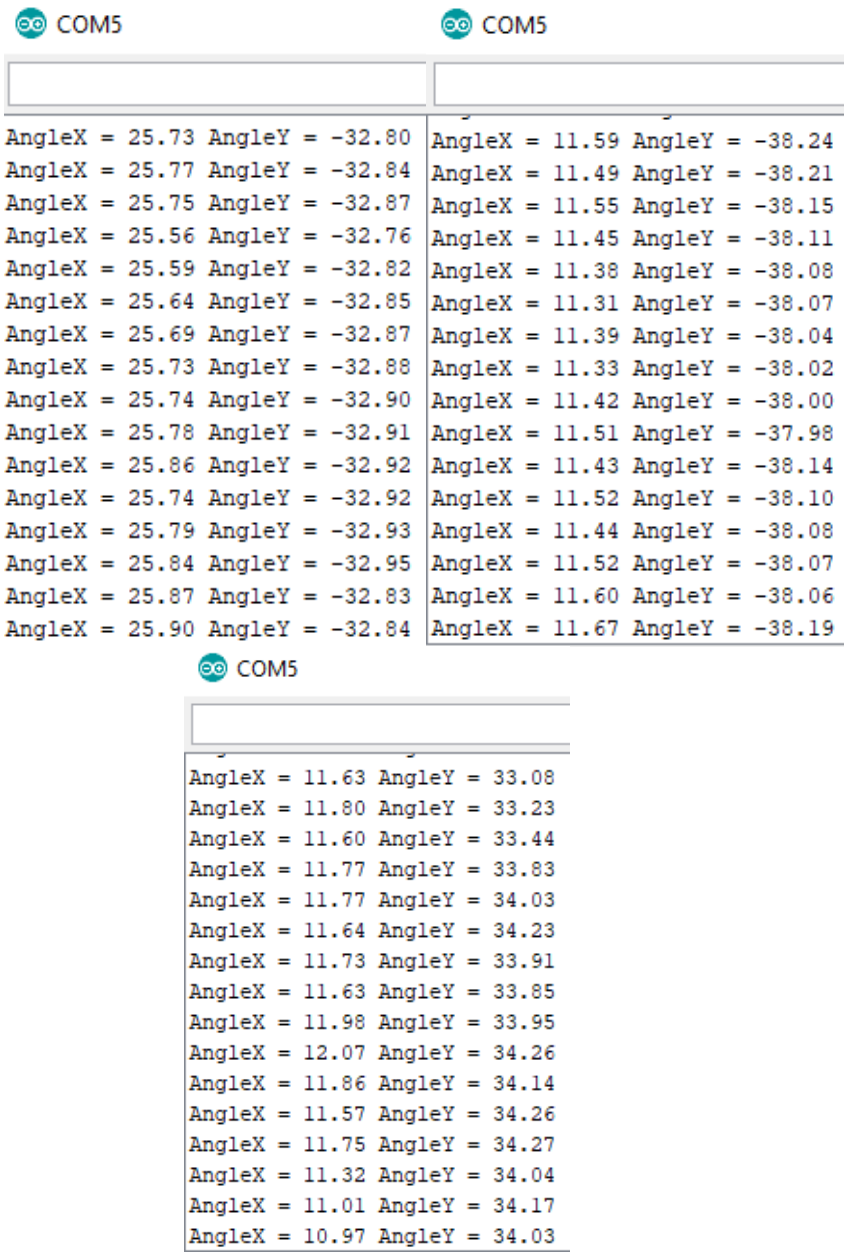

Рисунок 26. «Углы наклона»

Как видно из рисунка, показания углов стабильны, сильного разброса в значениях нет, максимальное отклонение приблизительно равно 1.

В процессе тестирования манипулятора, была задача откалибровать и зафиксировать оптимальные углы наклона для каждого из суставов.

Было произведено 2 тестирования. Первое- это калибровка, поиск лучших положений для оптимальной работы манипулятора. Второе- функциональное, тестирование зафиксированных рабочих положений.

Результатом начального тестирования является следующее:

1. При показании угла наклона больше 90 градусов, для колонны, манипулятор начинает крениться и терять равновесие.
2. При значении угла наклона свыше 120 градусов для руки и привода манипулятора, конструкция теряет равновесие, при этом происходит некорректное взаимное движение с колонной.

3. При угле наклона, равным 100 градусов и выше, происходит разрушение конструкции схвата.
4. При показании углов наклона, в диапазоне от 45 до 60 градусов для колонны, происходит свободное перемещение колонны относительно оси координат.
5. Показания углов наклона для привода и руки в диапазоне от 30 до 70 градусов позволяют свободно перемещаться звену манипулятора относительно оси, при этом, успешно совершать совместное движение с колонной.
6. Значение угла наклона, равное 75 градусам, позволяет успешно выполнять рассхват.

После проведения калибровки выявлены подходящие значения углов для рабочих положений. Результатом функционального тестирования являлось:

1. Зафиксированное 1 положение, при котором углы наклона, для каждого звена равны 10 градусам. Это начальное положение.
2. Зафиксированное 2 положение, при котором угол наклона для колонны равен 45 градусов, а для руки и привода 30.
3. Зафиксированное 3 положение, где угол наклона для колонны равен 75 градусов, а для руки и привода 60 градусов.
4. Зафиксированное 4 положение, при котором угол наклона колонны равен 90 градусов, руки и привода 40 градусов.

Как видно из результатов тестирования, разработанная программно-аппаратная система управления готова к использованию и выполнению своих задач.

Заключение

В результате выполнения выпускной квалификационной работы была разработана аппаратно-программная система управления. Основными задачами данной системы были- перемещения суставов манипулятора под определенным углом, благодаря данным, полученных от акселерометра и гироскопа по беспроводному соединению.

Как показало тестирование робототехнического устройства (см. пункт 6 «Тестирование»), весь требуемый функционал выполняется системой в полном объеме. Поэтому, можно сказать, что задачи, которые были поставлены перед разработкой системы, были успешно выполнены.

Данную программно-аппаратную систему управления можно без труда поставить на любой промышленный манипулятор, и, если потребуется, переналадить его, для выполнения других задач.

Список литературы

1. PCA9685 16 Channel 12 Bit PWM Servo Driver[Электронный ресурс].Режим доступа: **http://wiki.sunfounder.cc/index.php?title=PCA9685_16_Channel_12_Bit_PWM_Servo_Driver**
2. Википедия – Свободная энциклопедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/ .
3. Основы программирования Ардуино на языке [Электронный ресурс]. Режим доступа: http://mypractic.ru/urok-4-osnovy-programmirovaniya-arduino-na-yazyke-c.html.
4. Wemos D1 R2 и mini на основе esp8266 [Электронный ресурс].
5. Режим доступа: https://arduinomaster.ru/datchiki-arduino/esp8266-wemos-d1-mini-raspinovka/.
6. MPU-6050 Accelerometer + Gyro [Электронный ресурс]. Режим доступа: https://playground.arduino.cc/Main/MPU-6050/#short.
7. ESP8266:Прошивки/Arduino/Библиотеки/Библиотека ESP8266WiFi/Класс UDP/UDP-коммуникация между ESP8266 и внешней программой [Электронный ресурс]. Режим доступа:
8. ESP8266:Прошивки/Arduino/Библиотеки/Библиотека ESP8266WiFi/Класс UDP/UDP-коммуникация между ESP8266 и внешней программой[Электронный ресурс]. Режим доступа: http://wikihandbk.com/wiki/ESP8266:%D0%9F%D1%80%D0%BE%D1%88%D0%B8%D0%B2%D0%BA%D0%B8/Arduino/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_ESP8266WiFi/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_UDP/UDP-%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BC%D0%B5%D0%B6%D0%B4%D1%83_ESP8266_%D0%B8_%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B5%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BE%D0%B9.

Приложение А

Листинг клиентской программы «mpu6050_calman.ino»:

```
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

const int MPU_addr = 0x68; // I2C address of the MPU-6050
int32_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
float base_x_accel, base_y_accel, base_z_accel, base_x_gyro,  base_y_gyro, base_z_gyro;
const char   *ssid      = "Lola";
const char   *pass      = "01024567";
unsigned int  localport = 2000;
IPAddress ServerIP(192, 168, 4, 1);
WiFiUDP udp;
char buff[24];

void calibrate_sensors() {
  int               num_readings = 10;
  float              x_accel = 0;
  float              y_accel = 0;
  float              z_accel = 0;
  float              x_gyro = 0;
  float              y_gyro = 0;
  float              z_gyro = 0;
  for (int i = 0; i < num_readings; i++) {
    readData();
    x_accel += AcX;
    y_accel += AcY;
    z_accel += AcZ;
    x_gyro += GyX;
    y_gyro += GyY;
    z_gyro += GyZ;
    delay(100);
  }
  x_accel /= num_readings;
```

```
  y_accel /= num_readings;
  z_accel /= num_readings;
  x_gyro /= num_readings;
  y_gyro /= num_readings;
  z_gyro /= num_readings;
  base_x_accel = x_accel;
  base_y_accel = y_accel;
  base_z_accel = z_accel;
  base_x_gyro = x_gyro;
  base_y_gyro = y_gyro;
  base_z_gyro = z_gyro;
  Serial.print("base_x_accel = "); Serial.print(base_x_accel);
  Serial.print("base_y_accel = "); Serial.print(base_y_accel);
  Serial.print("base_z_accel = "); Serial.print(base_z_accel);
  Serial.print("base_x_gyro = ");  Serial.print(base_x_gyro);
  Serial.print("base_y_gyro = ");  Serial.print(base_y_gyro);
  Serial.print("base_z_gyro = ");  Serial.println(base_z_gyro);
}
void readData() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true);
  AcX = Wire.read() << 8 | Wire.read();
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();
  Tmp = Wire.read() << 8 | Wire.read();
  GyX = Wire.read() << 8 | Wire.read();
  GyY = Wire.read() << 8 | Wire.read();
  GyZ = Wire.read() << 8 | Wire.read();
}
void fillBuffer() {
  readData();
  GyX = GyX - base_x_gyro;
  GyY = GyY - base_y_gyro;
  GyZ = GyZ - base_z_gyro;
  buff[0] = (AcX & 0xFF000000) >> 24;
  buff[1] = (AcX & 0xFF0000) >> 16;
  buff[2] = (AcX & 0xFF00) >> 8;
  buff[3] = AcX & 0xFF;
  buff[4] = (AcY & 0xFF000000) >> 24;
  buff[5] = (AcY & 0xFF0000) >> 16;
  buff[6] = (AcY & 0xFF00) >> 8;
```

```
  buff[7] = AcY & 0xFF;
  buff[8] = (AcZ & 0xFF000000) >> 24;
  buff[9] = (AcZ & 0xFF0000) >> 16;
  buff[10] = (AcZ & 0xFF00) >> 8;
  buff[11] = AcZ & 0xFF;
  buff[12] = (GyX & 0xFF000000) >> 24;

  buff[13] = (GyX & 0xFF0000) >> 16;
  buff[14] = (GyX & 0xFF00) >> 8;
  buff[15] = GyX & 0xFF;
  buff[16] = (GyY & 0xFF000000) >> 24;
  buff[17] = (GyY & 0xFF0000) >> 16;
  buff[18] = (GyY & 0xFF00) >> 8;
  buff[19] = GyY & 0xFF;
  buff[20] = (GyZ & 0xFF000000) >> 24;
  buff[21] = (GyZ & 0xFF0000) >> 16;
  buff[22] = (GyZ & 0xFF00) >> 8;
  buff[23] = GyZ & 0xFF;
}
void sendBuffer() {
  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" AcY = "); Serial.print(AcY);
  Serial.print(" AcZ = "); Serial.print(AcZ);
  Serial.print(" GyX = "); Serial.print(GyX);
  Serial.print(" GyY = "); Serial.print(GyY);
  Serial.print(" GyZ = "); Serial.println(GyZ);
  udp.beginPacket(ServerIP, 2000);
  udp.write(buff, 24);
  udp.endPacket();
}
void initSerial() {
  Serial.begin(115200);
}
void initWiFi() {
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address ");
  Serial.println(WiFi.localIP());
```

```
  Serial.println("Starting UDP");
  udp.begin(localport);
  Serial.print("Local port ");
  Serial.println(udp.localPort());
}
void initWire() {
  Wire.begin();
 Wire.beginTransmission(MPU_addr);

  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0);    // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
}
void setup() {
  initSerial();
  initWiFi();
  initWire();
  calibrate_sensors();
}
void loop() {
  fillBuffer();
  sendBuffer();
  delay(100);
}
```

Приложение Б

Листинг серверной программы «serverkalmanmod.ino»

```
#include "Kalman.h"
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#define I2CAdd  0x40
const char* ssid = "Lola";
const char* password = "01024567";
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
double Xg, Yg, AcXangle, AcYangle, AcZangle, GyroskopX1angle,GyroskopY1angle,GyroskopY2angle,GyroskopX2angle,xxangle,yangle,kalAngleX,kalAngleY;
uint32_t timer;
unsigned int localport = 2000;
HCPCA9685 HCPCA9685(I2CAdd);
Kalman kalx;
Kalman kaly;
Kalman kalz;
IPAddress ServerIP(192, 168, 4, 1);
IPAddress ClientIP(192, 168, 4, 2);

  WiFiUDP Udp;

char buff[24];

  void setup() {

 HCPCA9685.Init(SERVO_MODE);
 Serial.begin(115200);
 Serial.println();
 WiFi.softAP(ssid, password);
 Serial.println("Starting UDP");
 Udp.begin(localport);
 Serial.print("Local port: ");
 Serial.println(Udp.localPort());
}
```

```
  void readData(){

 int cb = Udp.parsePacket();
 if (cb) {
  Udp.read(buff, 24);
  AcX = buff[0] << 24 | buff[1] << 16 | buff[2] << 8 | buff[3];
  AcY = buff[4] << 24 | buff[5] << 16 | buff[6] << 8 | buff[7];
  AcZ = buff[8] << 24 | buff[9] << 16 | buff[10] << 8 | buff[11];

  GyX = buff[12] << 24 | buff[13] << 16 | buff[14] << 8 | buff[15];
  GyY = buff[16] << 24 | buff[17] << 16 | buff[18] << 8 | buff[19];
  GyZ = buff[20] << 24 | buff[21] << 16 | buff[22] << 8 | buff[23];
 Xg=AcY/177 ;
Yg=AcX/177 ;
  xxangle=Xg;
  yangle=Yg;
AcYangle = (atan2(AcX, AcZ) + PI) * RAD_TO_DEG;
AcXangle = (atan2(AcY, AcZ) + PI) * RAD_TO_DEG;
AcZangle = (atan2(AcX, AcY) + PI) * RAD_TO_DEG;
double gyroXrate = (double)GyX / 131.0;
double gyroYrate = -((double)GyY / 131.0);
GyroskopX1angle = kalx.getRate() * ((double)(micros() - timer) / 1000000);
GyroskopY1angle = kaly.getRate() * ((double)(micros() - timer) / 1000000);
GyroskopX1angle = kalx.getRate() * ((double)(micros() - timer) / 1000000);
GyroskopX2angle=sqrt(pow(GyroskopX1angle,2));
GyroskopY1angle=kaly.getRate() * ((double)(micros() - timer) / 1000000);
GyroskopY2angle=sqrt(pow(GyroskopY1angle,2));
if(GyroskopX2angle>0.5){
 xxangle+=gyroXrate*((double)(micros() - timer)/1000000);
}
else{
xxangle+=0;
 }
if(GyroskopY2angle>0.5){
  yangle += gyroYrate*((double)(micros() - timer)/1000000);
}
else{
yangle+=0;
}
kalAngleX = kalx.getAngle(Xg, gyroXrate, (double)(micros() - timer) /
1000000);
kalAngleY = kaly.getAngle(Yg, gyroYrate, (double)(micros() - timer) /
1000000);
timer = micros();
```

```
Serial.print("AngleX = "); Serial.print(kalAngleX);
Serial.print(" AngleY = "); Serial.println(kalAngleY);
}
}
void loop()
{
  readData();
  delay(100);
}


      if(kalAngleX<=45 && kalAngleX>=0){

kalAngleX=45;

HCPCA9685.Servo(0, kalAngleX);
HCPCA9685.Servo(4,kalAngleX);
if(kalAngleX<=0){
kalAngleX=10;
HCPCA9685.Servo(0, kalAngleX);
HCPCA9685.Servo(4,kalAngleX);
if(kalAngleY<=45 && kalAngleY >=0){
kalAngleY=45;
HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
if(kalAngleY<=0){
kalAngleY=10;
HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
}
}
}
}

         while(kalAngleX>45&&kalAngleX<=60){

          kalAngleX++;

        HCPCA9685.Servo(0, kalAngleX);

         HCPCA9685.Servo(4,kalAngleX);
}
       while(kalAngleY>45&&kalAngleY<=60){
       kalAngleY++;
HCPCA9685.Servo(8, kalAngleY);
HCPCA9685.Servo(12,kalAngleY);
```

```
    }
    if(kalAngleX>60 && kalAngleX<=90){
    kalAngleX=90;
    HCPCA9685.Servo(0, kalAngleX);
    HCPCA9685.Servo(4,kalAngleX);
    if(kalAngleX>90 && kalAngleX<=120){
    kalAngleX=120;
    HCPCA9685.Servo(0, kalAngleX);
    HCPCA9685.Servo(4,kalAngleX);
    }
    }

    if(kalAngleY>60 && kalAngleY<=90){
kalAngleY=90;
    HCPCA9685.Servo(8, kalAngleY);
    HCPCA9685.Servo(12,kalAngleY);
    if(kalAngleY>90 && kalAngleY<=120){
    kalAngleY=120;
    HCPCA9685.Servo(8, kalAngleY);
    HCPCA9685.Servo(12,kalAngleY);
    }
    }
    if(kalAngleX>120 && kalAngleY>120){
    kalAngleX=10;
    kalAngleY=10;
    HCPCA9685.Servo(0, kalAngleX);
    HCPCA9685.Servo(4,kalAngleX);

        HCPCA9685.Servo(8, kalAngleY);
        HCPCA9685.Servo(12,kalAngleY);
    }
```