ОЛИМПИАДА ШКОЛЬНИКОВ “ШАГ В БУДУЩЕЕ”
НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ
“ШАГ В БУДУЩЕЕ, МОСКВА”

1745

регистрационный номер

Радиоэлектроника и лазерная техника

название факультета

Радиоэлектронные системы и устройства

название кафедры

Использование датчиков мышечной активности для опознания

напряжения мышц

название работы

Автор: Мифтахов Руслан Борисович

фамилия, имя, отчество

ГБОУ Инженерная школа №1581

наименование учебного заведения, класс

Научный руководитель: Круглов Геннадий Валерьевич

фамилия, имя, отчество

МГТУ им. Баумана кафедра РЛ-1

место работы

звание, должность

подпись научного руководителя

Москва - 2019

1 Аннотация

В работе проведено исследование методики считывания сигналов мышечной активности предплечья.

Целью работы является проектирование устройства, которое может осуществлять управление механическим объектом посредством мышечных сигналов. Основной подход к считыванию - измерение разности потенциалов в различных точках мышц. Наиболее распространены измерения сигналов с помощью миографических датчиков.

Рассмотрены поверхностные, неинвазивные электроды, проводился анализ существующих реализаций и обоснование необходимости в разработке собственной конструкции электродов и системы считывания.

Существующие решения нацелены,преимущественно, на получение единичных сигналов, и не предназначены для комплексного анализа пальцевых жестов. Спроектированы электрические схемы отдельных элементов и модульная схема устройства, разведены платы, изготовлен прототип.

Прототип представляет собой манжету с массивом электродов, датчиками, модулями питания, аналого-цифровыми преобразователями и центрального процессорного устройства на базе ESP-32.

Разработаны алгоритмы сбора сигналов и распознавания пальцевых жестов, программа для процессора.

Применимость данного решения продемонстрирована в виде стенда по управлению роботизированным манипулятором - пятипальцевой механической руки, реализованной по клиент серверной архитектуре с использованием произвольного протокола на базе http. Данная система может быть применена для управления исполнительными силовыми элементами в агрессивных средах, при протезировании, в виртуальной и дополненной реальности.

2 Содержание

Аннотация 2

Содержание 3

Введение 4

Биологическое обоснование 5

Миоэлектрический сенсор 7

Устройство электрода 10

Устройство датчика 11

Сборка и изготовление датчика 13

Структурная схема устройства 16

Программа контроллера миографических датчиков 18

Демонстрационный стенд 20

Взаимодействие стенда и основного устройства. 23

Заключение **Ошибка! Закладка не определена.**

Список используемых источников 25

3 Введение

В связи с развитием радиоэлектроники и робототехники, усложнением механизмов, которые требуют управления, в данное время проводится большое количество исследований, связанных с поиском дополнительных интерфейсов управления.

Одним из таких интерфейсов является прямое управление механизмами посредством прямого считывания миографических сигналов.

До недавнего времени большинство решений в этой области было связано с простейшим распознаванием состояния мышц. Как правило такие интерфейсы выдавали только два состояние - “ДА” и “НЕТ”.

Только недавно стали предприниматься попытки расширить функционал такого интерфейса. В большинстве случаев миографический интерфейс оснащался дополнительными источниками данных, например: аудиоканалом, различными датчиками перемещения в пространстве (гироскопы, акселерометры, магнитометры). Безусловно, данные решения заслуживают внимания, но в данной работе была предпринята попытка расширения функционала, основанного именно на чтении мышечных электрических управляющих воздействий.

В работе доказано, что на современной элементной базе возможно решение этой задачи с хорошим качеством.

Описана теория измерения сигналов мышечного тонуса, разработана схема и конструкция датчика. Выполнен рабочий экземпляр устройства, считывающего пальцевые жесты. Воплощен стенд демонстрирующий работоспособность.

В разделе “Биологическое обоснование” приведена краткая теория считывания биоэлектрических сигналов мышц пальцев.

Раздел “Миоэлектрический сенсор” содержит комплекс исследований, связанный с исследованием аналогов и прототипов электромиографических (ЭМГ) датчиков, выявлением их достоинств и недостатков.

Подраздел “Устройство электрода” описывает комплекс электродов, применяемый в данной работе.

Подраздел “Устройство датчика” содержит описание и расчет электрической схемы датчика.

“Сборка и изготовление датчика” - показывает этапы работы над датчиком.

“Структурная схема устройства” - разъясняет электрическую конструкцию устройства в целом.

“Программа контроллера миографических датчиков” - алгоритм и принцип работы микропроцессорного ядра устройства.

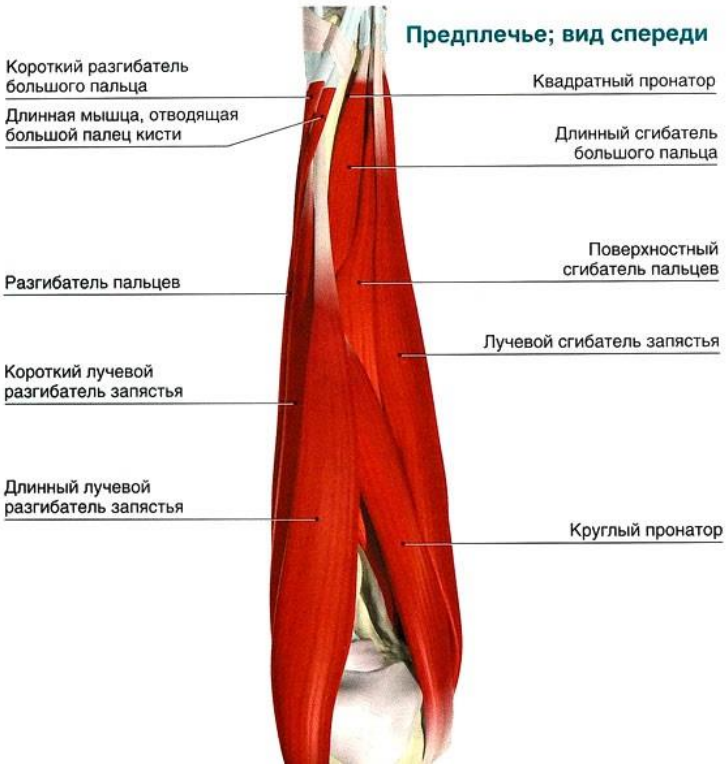
“Демонстрационный стенд” описывает структурную схему устройства визуализации.

“Взаимодействие стенда и основного устройства” описывает протокол связи.

4 Биологическое обоснование

Мышечное сокращение – это реакция мышечных тканей на нейромедиаторы. Нейромедиатор - биологически активные химические вещества, с помощью которых производится передача электрического импульса от нервной клетки. Такой импульс можно определить при замере разности потенциалов возникающей в мышце. Замеряя и обрабатывая разность потенциалов мышцы, можно определить в какой момент палец сгибается, напряжен или разгибается.

Мышцы предплечья состоят из двух групп мышц — передних и задних. Передние мышцы — сгибатели, а задние — разгибатели и супинаторы. Одни из этих мышц отвечают за движения запястья, а другие — за движения пальцев. Сокращающаяся мышца передает костям усилие через сухожилия. При сокращении мышц, если усилия достаточно, то пальцы перемещаются (1).



“Рис. 1”

В движении пальцев участвуют, по крайней мере, 2 мышцы, одна из которых отвечает за сгибание, а другая- за разгибание соответственно (Расположение мышц - “Рис. 1”). Мышечный комплекс предплечья не имеет отдельных групп мышц для управления каждым пальцем. Движение пальцев осуществляется вариацией изгибов одной и той-же группы мышц. В работе выдвигается гипотеза, в соответствии с которой, в независимости от усилия, для конкретного типа движения пальца, общая картина сигналов снятых с группы мышц будет неизменна. Основываясь на этом, снятие множества сигналов с различных участков группы мышц предплечья позволит однозначно определить сформированный жест, причём чем больше количество точек измерения

сигналов, тем точнее можно определить жест и тем больше будет общее количество распознаваемых жестов.

5 Миоэлектрический сенсор

В данной части произведено рассмотрение имеющихся реализаций датчиков и возможности их использования в проекте. При подборе датчиков пришлось учитывать: доступность, стоимость, надежность, размеры и сложность интеграции.

1)Датчик компании SICHIRAY OY motion “GForce”, стоимостью 86 401 рублей.

“Gforce”

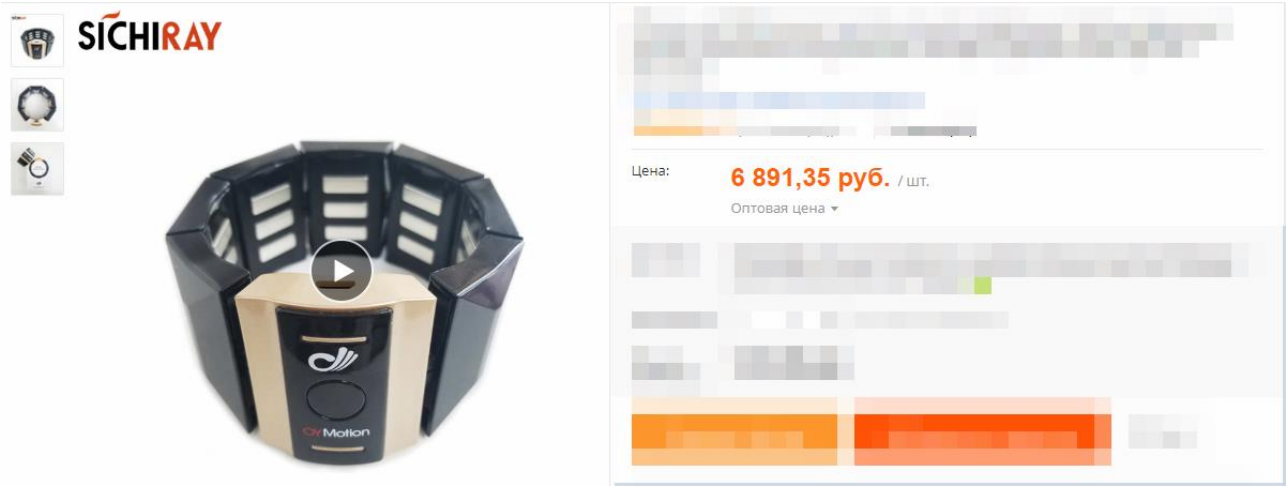
Датчик имеет ряд недостатков, такие как: высокая стоимость, количество распознаваемых жестов (8 жестов). Датчик не позволяет определить какие-либо жесты, кроме предусмотренных изначально.

2)Датчик “Thalmic Labs Myo Gesture Control”, стоимостью 20 990 рублей.

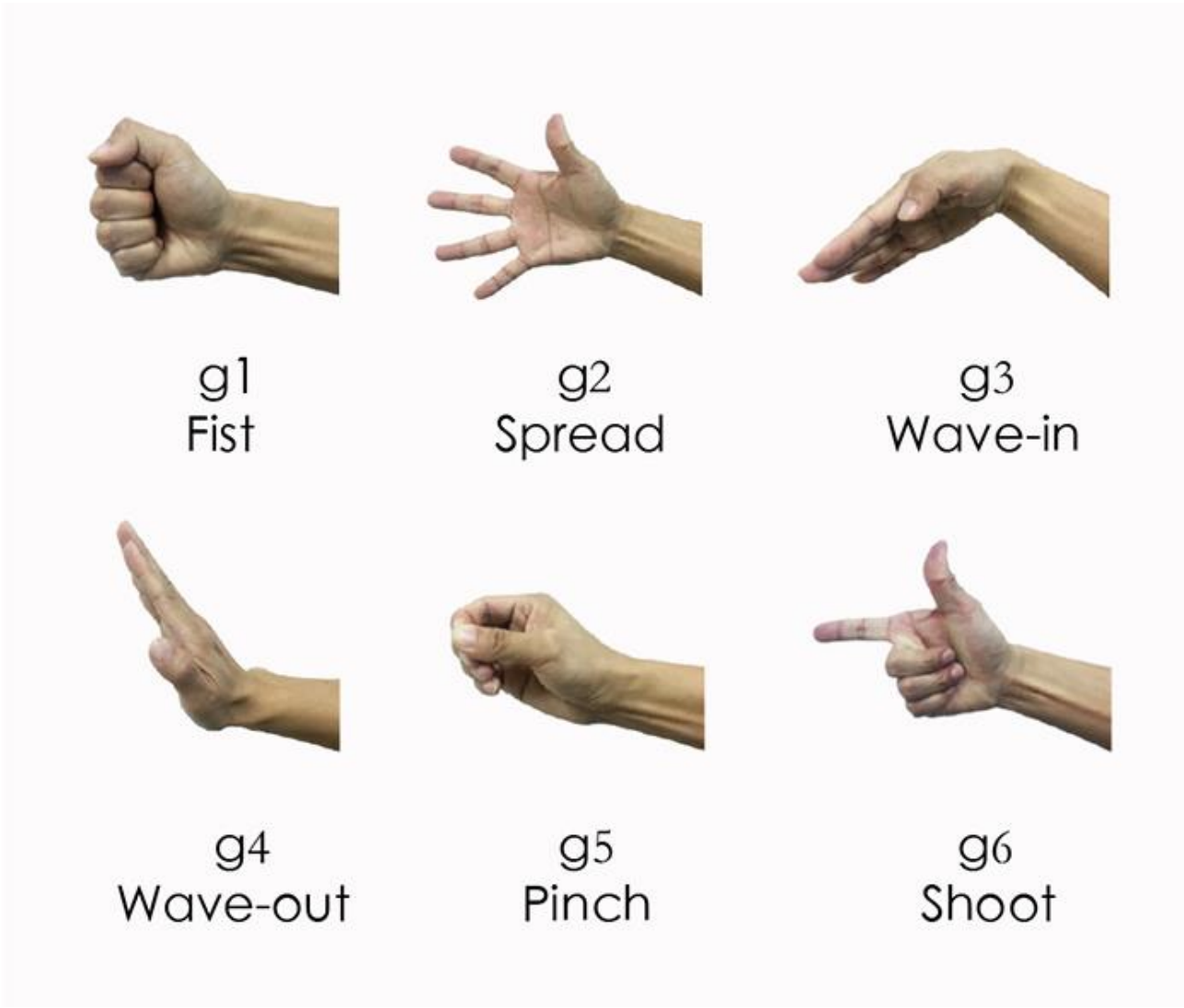
“Thalmic Labs Myo Gesture Control”

“Браслет Myo позволяет использовать электрическую активность в мышцах для беспроводного управления компьютером, телефоном и другими цифровыми приборами.” В датчик встроен гироскоп, но он также не позволяет определять большое количество жестов.

3) Датчик компании SICHIRAY OY motion “GForce”, стоимостью 6 891. 35 рублей.



“GForce используется для определения жестов в основном через сигналы EMG. Gforce позволяет определять 6 жестов, имеет модуль Bluetooth, что позволяет управлять смарт-системами.” Браслет имеет те же недостатки, которые были описаны ранее (2).



“Жесты, поддерживаемые браслетом”

Приведенные миографические сенсоры имеют малое количество считываний разности потенциалов с мышц, также скорость считывания низкая. Чтобы восполнить недостаток количества измерений, используется гироскоп. Используя подобную схему измерений, суть снимаемых показаний меняется, исключая возможность увеличения количества распознаваемых жестов при использовании пальцев руки. Гироскоп возможно использовать в датчиках, но как дополнительный источник данных.

4) ЭМГ-датчик компании “BitronicsLab”, стоимостью 25 000 рублей (3).

Данный ЭМГ-сенсор, имеет всего один датчик - два электрода. Скорость измерений датчика низка. Датчик необоснованно дорогой.

На основе произведенного анализа параметров доступных датчиков, было принято решение, спроектировать и изготовить собственный сенсор.

5.1 Устройство электрода

Электрод представляет собой контактную площадку, с минимальным электрическим сопротивлением, которая должна максимально плотно прилегать к участку кожи, с которой будет снята разность потенциалов. Электрод состоит из двух шайб, одна из которых имеет 3-х миллиметровый внутренний диаметр и 9-ти миллиметровый внешний диаметр, а другая - 5-ти миллиметровый внутренний и 15-ти миллиметровый внешний, которая и является основной проводящей частью прилегающей площадки. Разность потенциалов снимается между двумя электродами, расстояние между которыми 83мм.

Сами электроды закреплены на пластмассовом, диэлектрическом основании. Всего имеется 10 комплектов электродов, по три на датчик. Два

электрода - измерительные, третий - заземляющий. Как показали эксперименты, такая конструкция не требует контактных гелей, которые обычно используются.

5.2 Устройство датчика

На вход датчик получает 3 сигнала с электродов, между двумя из которых измеряем разность потенциалов, усиливаем и фильтруем ее, а третий сигнал является “заземлением”.

Первый каскад схемы построен на инструментальном (измерительном) усилителе U1, марки AD8220. В соответствии с руководством по применению, изменяя сопротивление R1, можно изменить коэффициент усиления G по формуле (4) :

$$R1=49.4k\Omega /(G-1),$$

где R1 - сопротивление резистора R1;

G - коэффициент усиления.

При проведении экспериментов оказалось, что оптимальный коэффициент усиления должен быть 200, проведя расчеты, можно вычислить сопротивление R1=248Ω. Ближайший резистор из таблицы номиналов 240Ω, его и используем, при этом коэффициент усиления первичного каскада G составил 206.8, что вполне соответствует требованиям.

Конденсатор C3 предназначен для фильтрации постоянной составляющей сигнала, кроме того R-C-цепочка, работающая как фильтр высоких частот; образует завал амплитудно-частотной характеристики в области нижних частот. Емкость C3 [мкФ] была рассчитана по формуле (5):

$$C3=1/(2*Pi*R7*f),$$

где R7 - сопротивление резистора R7;

f-частота аналогового сигнала равная 100 Гц.

Такая частота обоснована проведенными исследованиями диапазона частот входного сигнала от 100 Гц до 1000 Гц.

Последующие каскады схемы устроены на счетверенном операционном усилителе TL084 (6). Второй каскад использует один из операционных усилителей U2:B, включенный с коэффициентом усиления 1,фактически инвертор (7), рассчитан по формуле

$$G=R8/R7,$$

где R8 и R7 - сопротивления резисторов R7 и R8 соответственно.

Следующим каскадом схемы являются диоды: D1, скомпонованные в одном корпусе - BAS40W-04. Диоды предназначены для разделения сигнала на положительный и отрицательный полупериоды. Сигналы попадают на неинвертирующий и инвертирующий входы операционного усилителя U2:C соответственно. Данный ОУ складывает сигналы, без усиления.

Схема на U2:D является фильтром низких частот, пропуская сигнал до 1 кГц.

$$C4=1/(2*Pi*R13*f),$$

где R13 - сопротивление резистора R13;

f-частота переменного тока.

Ближайшие к расчетным значения: R13=82kΩ, C4=1мкФ.

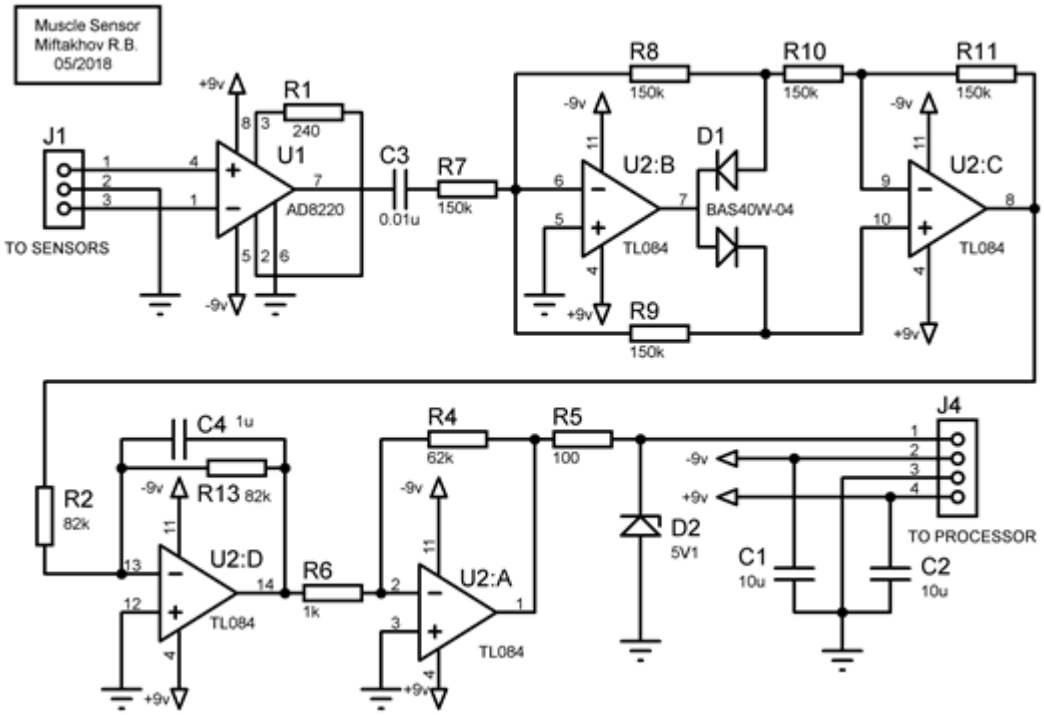
Следующий каскад имеет коэффициент усиления G

$$G=R4/R6=62,$$

где R4 и R6 - сопротивления резисторов R4 и R6 соответственно.

Сигнал на нем инвертируется. Данный каскад предназначен для усиления сигнала до уровня возможного для оцифровки (от 0 до 5 вольт) .

Резистор R5 является гасящим резистором и предназначен для согласования уровня, который будет ограничен (не выше 5.1 вольт) стабилитроном D2. Данное ограничение призвано защитить последующие блоки схемы.



“Схема электрическая принципиальная”

5.3 Сборка и изготовление датчика

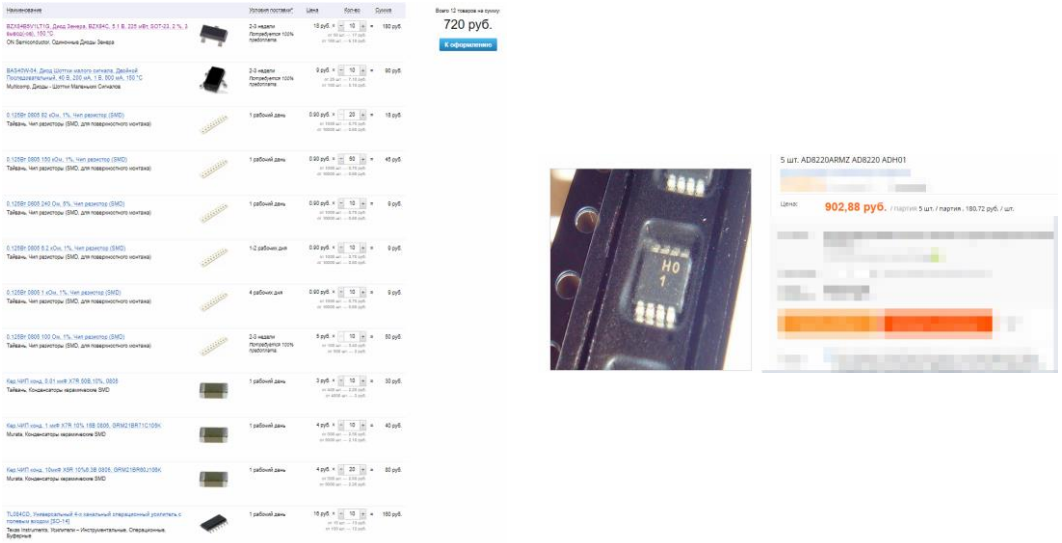
Для минимизации габаритов прототипа, было решено изготовить датчик используя поверхностные монтаж, для этого были подобраны детали соответствующих номиналов.

Таблица 1 - используемые радиодетали.

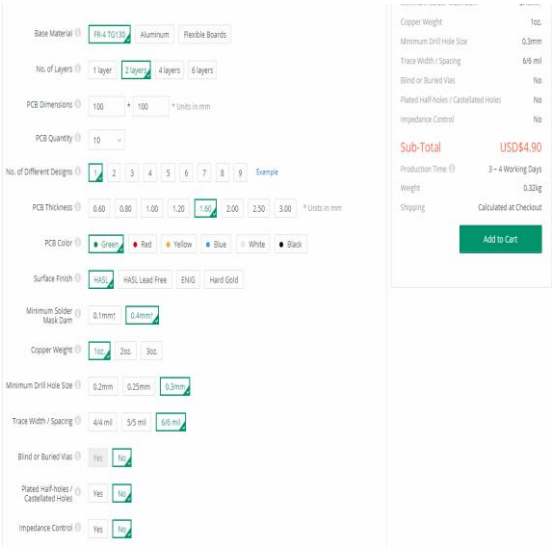
Наименование, тип	Номинал	Корпус	Количество	Обозначение на схеме
Инструментальный ОУ	AD8220	MSOP	1	U1
ОУ	TL084	S0-14	1	U2
Резистор	240Ω	0805	1	R1
Резистор	82kΩ	0805	2	R2, R13

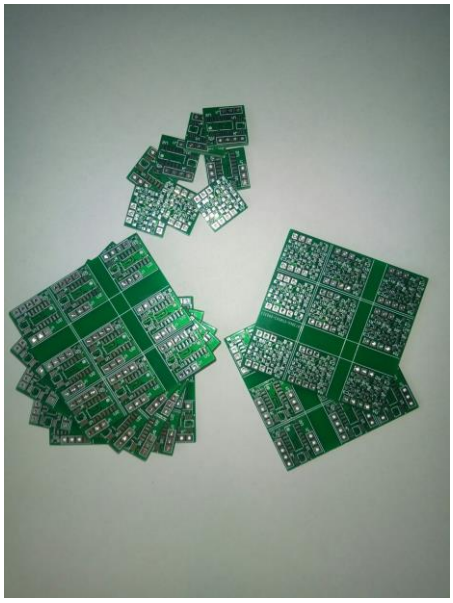
Резистор	150kΩ	0805	5	R7, R8, R9, R10, R11
Резистор	1kΩ	0805	1	R6
Резистор	62kΩ	0805	1	R4
Резистор	100Ω	0805	1	R5
Диодная сборка	BAS40W-04	SOT323	1	D1
Стабилитрон	BZX84B5V1	SOT23	1	D2
Конденсатор	0.01uF	0805	1	C3
Конденсатор	1uF	0805	1	C4
Конденсатор	10uF	0805	2	C1, C2

В программе “Proteus” была разведена плата, которую изготовили на заводе “Itead”. Комплектующие приобретены на “aliexpress.com” и в магазине “Чип и Дип”.



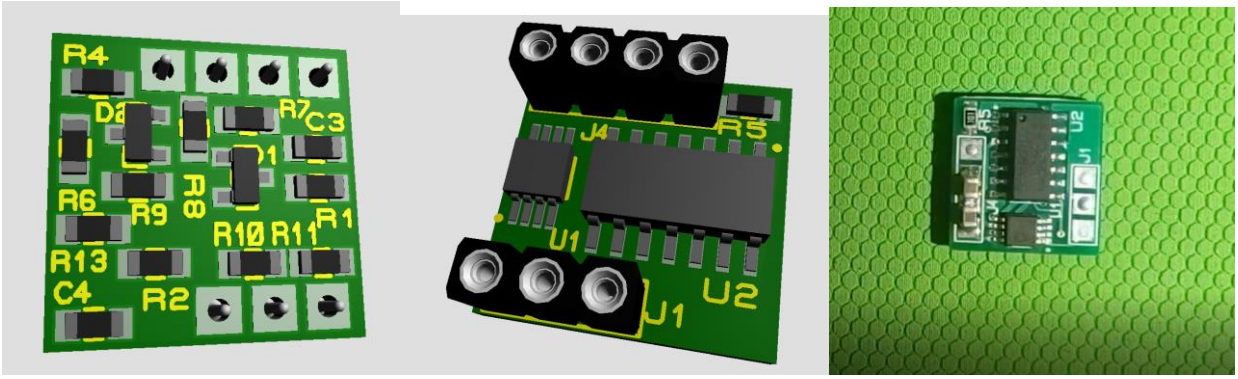
“Приобретение деталей”





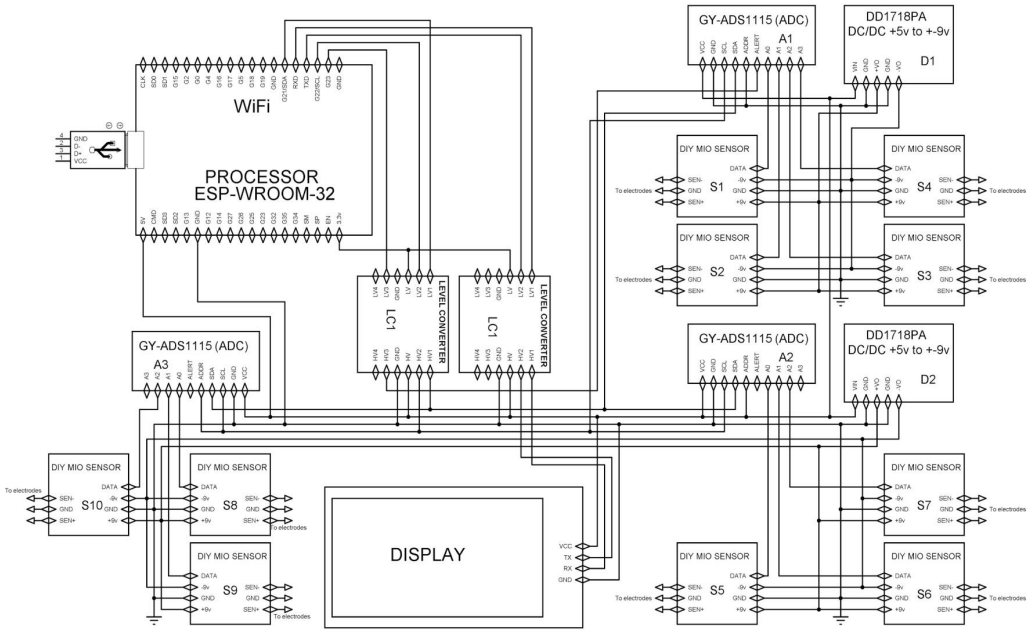
“Заказ и изготовленные платы датчика”

Сборка датчика произведена в домашних условиях. Не учитывая сборку, себестоимость собранной платы датчика составила 277 рублей.



“Датчик”

5.4 Структурная схема устройства



“Структурная схема”

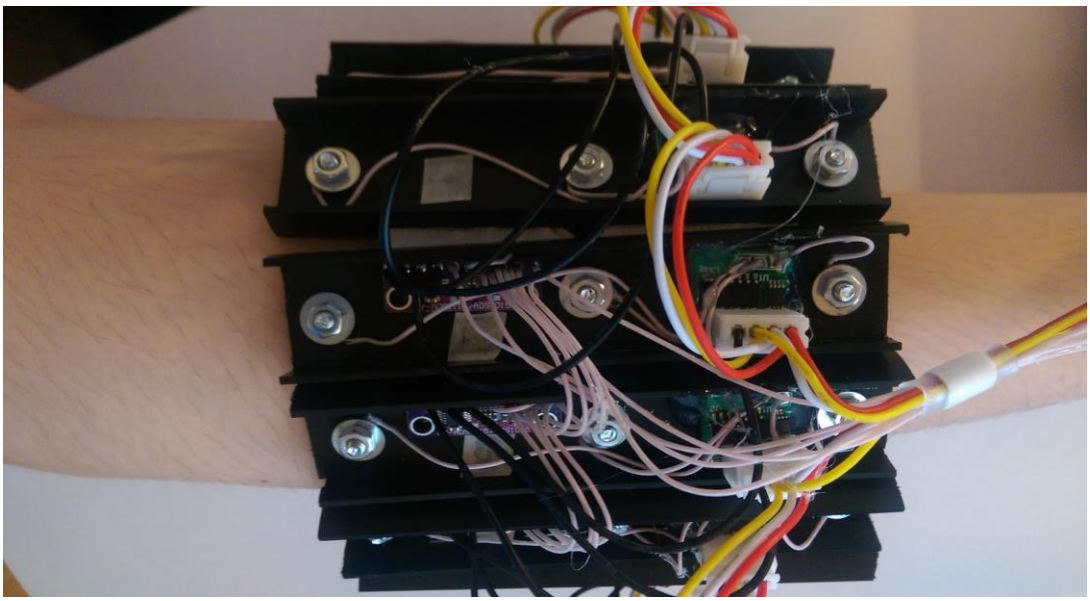
Более подробно Приложение А.

Сигналы полученный с электродов обрабатывается платами датчика (S1-S10). Выходы датчиков подключены к 4-х канальным аналого-цифровым преобразователям (АЦП), на схеме: A1-A3, каждый датчик к отдельному каналу. АЦП по шине I2C, передают оцифрованную информацию на процессорное устройство, по запросу процессора. Дисплей отображает информацию по команде ЦПУ. В связи с тем, что уровни сигналов дисплея и АЦП 0-5 вольт, а уровни процессора 0-3.3 вольта, необходимо применить согласующие устройства в виде преобразователей уровня (LC1, LC2).

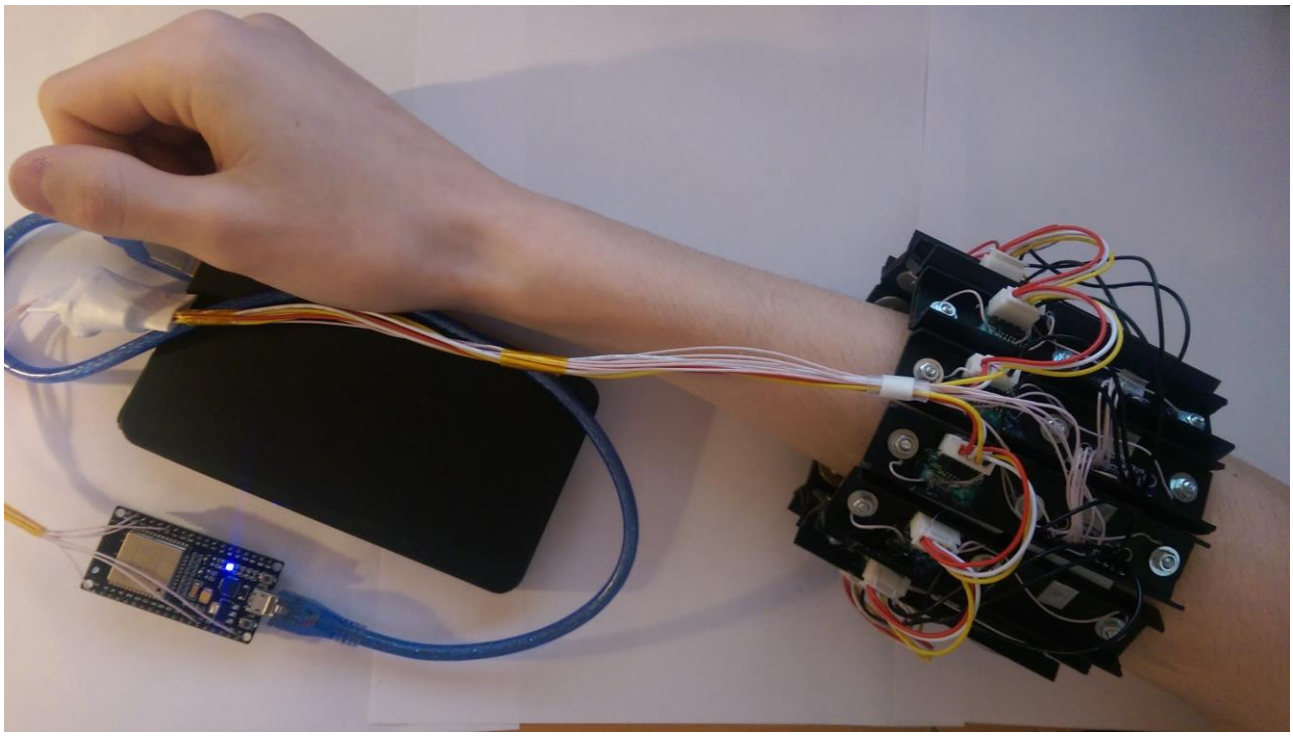
Основной источник питания схемы имеет напряжение в 5 вольт, плата процессора содержит встроенный преобразователь из 5 в 3.3 вольт. Модулям датчиков необходимо двуполярное питание 9 вольт. Поэтому в схеме используется 2 преобразователя питания (D1, D2) из 5 вольт в +-9 вольт.

Таблица 2 - блоки структурной схемы “Комплекс датчиков мышечной активности”.

Название блока	Назначение	Количество	Обозначение
ESP-WROOM-32	Центральный процессор	1	PROCESSOR
GY-ADS1115(ADC)	Аналого-цифровой преобразователь	3	A1, A2,A3
DIY MIO SENSOR	Измерение и первичная обработка сигнала	10	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10
NX4832T035_011	Дисплей	1	DISPLAY
LEVEL CONVERTER	Преобразователь уровней	2	LC1, LC2
DD1718PA	Преобразование напряжения +5 вольт в +-9 вольт	2	D1, D2



“Устройство на руке”



“Включенное устройство”

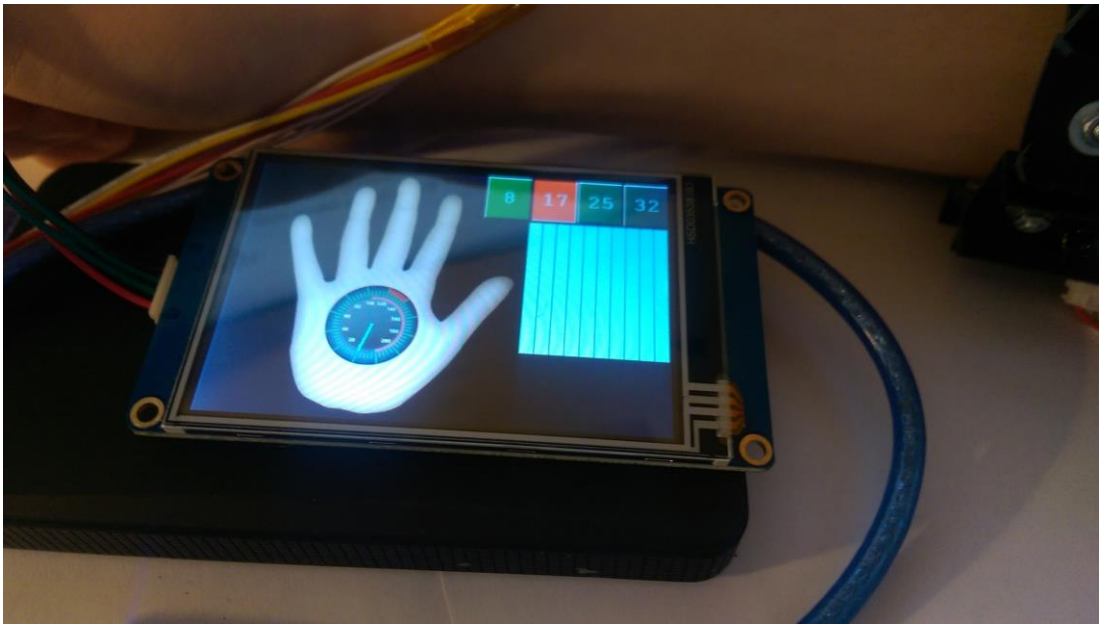
5.5 Программа контроллера миографических датчиков

Программа написана на языке C++ в среде разработки Arduino IDE (11). Приложение Б .Программа имеет два режима: режим настройки “калибровка” и основной режим “функционирование”. Микропроцессор, вне зависимости от режима работы, с частотой 30 Гц получает измерения с 10-ти сенсоров.

5.5.1 Этап калибровки

На этапе калибровки оператору требуется, в соответствии с сценарием, производить жесты пальцами. Показания датчиков при каждом жесте объединяются в n-мерный массив значений. Полученные данные проходят нормировку относительно максимального значения в последовательности, обработанный массив сохраняется в матрице как ее строка, где номер строки равен соответствует жесту в словаре. Среднеарифметическое значение показаний всех датчиков записывается в массив параллельно калибровочной матрице и будет использовано в дальнейшем для определения усилия сжатия. Словарь представляет из себя массив из 32 жестов (2^5=32 комбинации)

отсортированный по частоте использования. В конце калибровки получается матрица 32x10. В дальнейшем для удобства обработки, строка матрица воспринимается как 10-мерный вектор. Калибровка осуществляется с помощью дисплея и демонстрационного стенда. Во время калибровки желательно максимально напрягать мышцы для получения наилучшего распознавания.



“Дисплей на этапе калибровки”

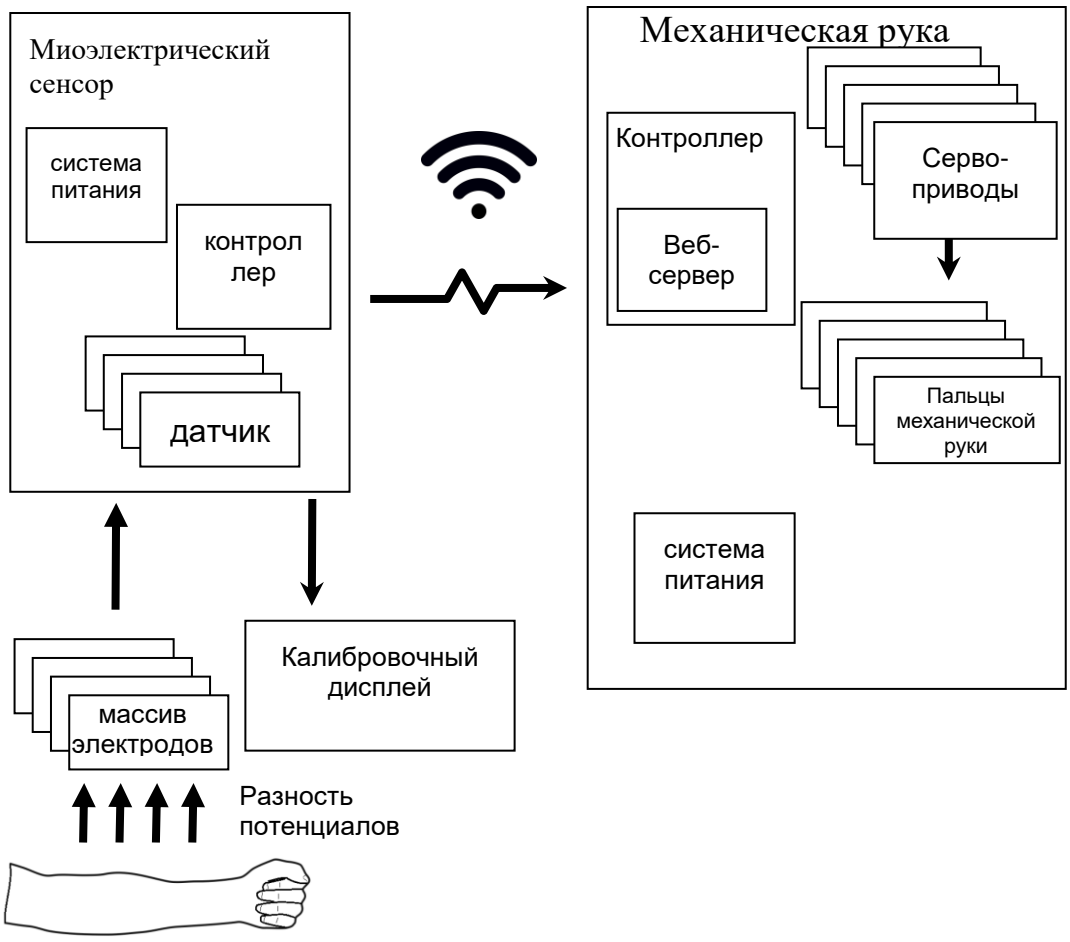
5.5.2 Основной цикл работы

На этом этапе программа циклически опрашивает каждый из датчиков и формирует текущий 10-ти мерный вектор, далее осуществляется поиск вектора с наименьшим отклонением от исходного. Графически процесс нормализации можно представить, как определение точки на поверхности гиперсферы 10-мерного пространства, соответствующей определенному вектору. Помимо текущего измерения на сфере можно также отобразить точки, соответствующие различным жестам, вычисленные на этапе калибровки. Найдя точку в калибровочной матрице, которая в совокупности с измеренной даст вектор с наименьшей длиной, можно определить какому жесту соответствует полученный вектор, а сила сжатия руки влияет на усредненные значения сигналов и соответственно угол отклонения пальцев механической руки.

Вычисленные данные визуализируются на дисплее и по http-протоколу отправляются на демонстрационный web-сервер.

5.6 Демонстрационный стенд

Демонстрационный стенд состоит из исполнительного устройства - механической руки с контроллером управления. Структурная схема системы представлена ниже.



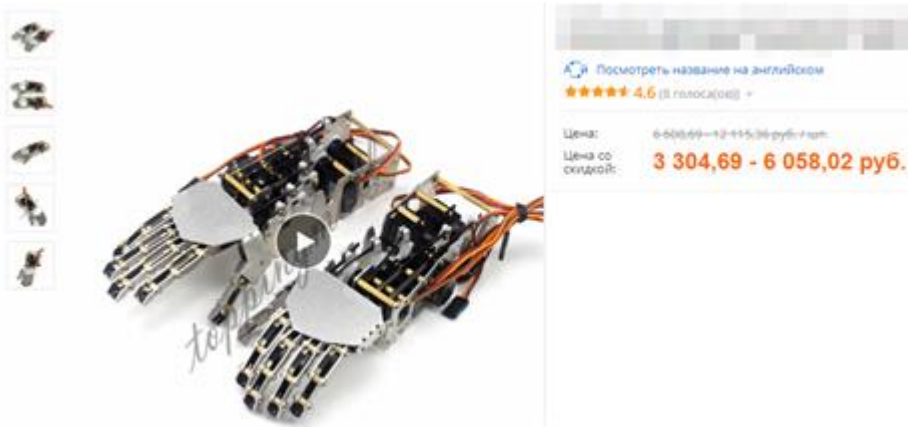
“Обобщенная блок-схема”

5.7 Описание механической руки

Механическая рука была заказана из Китая (2), но также полноразмерный протез или макет может быть напечатан на 3д-принтере. Помимо протезирования данная система может использоваться для жестового

управления другими машинами и механизмами, в виртуальной и дополненной реальности.

В движение пальцы механической руки приводятся с помощью сервоприводов.Они были заказаны отдельно, так как те, которые шли в комплекте оказались ненадлежащего качества.



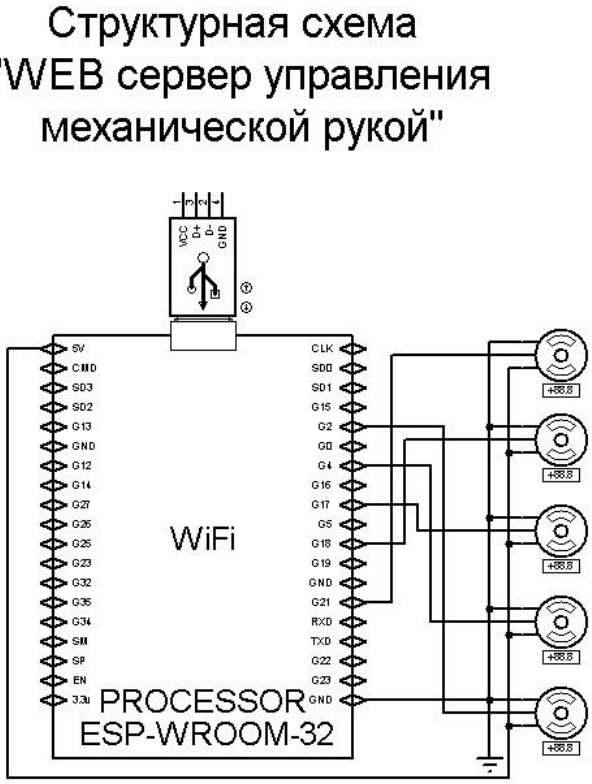
“Используемая механическая рука”

“Используемые сервоприводы”

Была написана программа в среде разработки Arduino IDE для контроллера EPS32/ESP8266 (9), которая принимает команды через web-интерфейс и приводит пальцы в движение.

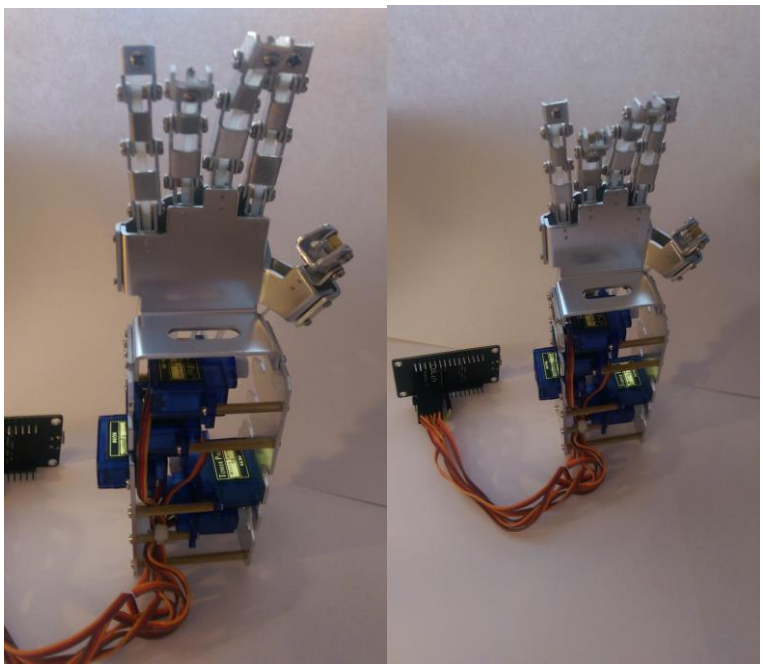
Сервопривод управляется с помощью библиотеки servo.h, прямым указанием угла отклонения пальца. Программа, работающая на сервере-руке,

является вспомогательной и не заслуживает глубокого описания в рамках данной научно-исследовательской работы.



С программой можно ознакомиться в Приложении В к данной работе. Назначение программы - получить управляющую команду от “клиента”, которым является основная программа и выполнить ее.

Кроме того, как дополнение к основной программе визуализации, используется дисплей компании “Nextion” (10), который принимает простейшие управляющие команды от основной программы по проводному последовательному интерфейсу и дублирует показания датчиков и действия механической руки. К сожалению, текст программы из-за особенности организации дисплея, не может-быть распечатан в читаемом виде.



“Собранная механическая рука”

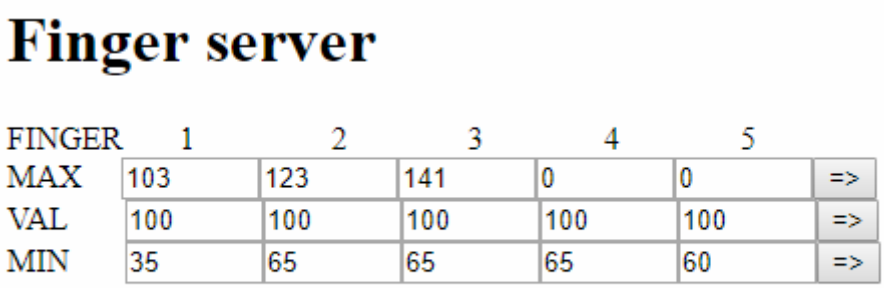
5.8 Взаимодействие стенда и основного устройства.

Основное устройство измеряет миографическое состояние мышц, производит расшифровку данных. Определяет жест и транслирует его на демонстрационный стенд в виде HTML POST запроса.

Формат запроса “POST/?N=XX...&N=XX”, где N-номер пальца (от 1 до 5), а XX - степень сгибания (от 0 до 100 %). Пальцы пронумерованы так, что 1 - мизинец, а 5 - большой. После получения запроса стенд преобразует его в управляющие воздействия и применяет их к системе сервоприводов, которые сгибают и разгибают пальцы механической руки.

```
l?1=100&2=100&3=100&4=100&5=100
```

“Запрос”



“Сервер”

6 Заключение

На основании проведенных работы и анализа можно утверждать, что использование ЭМГ датчиков, на современной радио-элементной базе, позволяет сконструировать недорогой высококачественный инструмент, дающий возможность создать “дружественный” и удобный интерфейс управления.

Проведенные исследования показали “жизнеспособность” решения на базе миографических датчиков. Невзирая на существующие сложности, однозначно можно утверждать, что применение этой разработки в ближайшем будущем рекомендовано и будет востребовано.

В результате получено уникальное устройство, способное корректно определять пальцевые жесты, степень напряжения мышц.

Дальнейшее исследование необходимо посвятить разработке более эргономичной конструкции устройства в целом и совершенствованию алгоритмов распознавания жестов. Идеально было бы достичь такого результата, когда обосновались-бы полусогнутые пальцы, что дало бы еще большее расширение сферы применения устройства.

7 Список используемых источников

1 - https://www.bsmu.by/downloads/kafedri/k_anatomia/stud/2017-2/mr13.pdf

2 - https://aliexpress.com/

3 - http://www.bitronicslab.com/

4 - Техническая документация на ОУ AD8220

©2006 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners. C03579-0-4/06(0)

5 - Р. Кофлин Ф. Дрискол ОПЕРАЦИОННЫЕ УСИЛИТЕЛИ И ЛИНЕЙНЫЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ Издательство “Мир”, Москва

6 - Техническая документация на ОУ TL084

2001 STMicroelectronics - Printed in Italy - All Right Reserved

STMicroeelectronics GROUP OF COMPANIES

7 - Г.И. Пухальский Т.Я. Новосельцева Справочник: ”Проектирование дискретных устройств на интегральных микросхемах”

8 - С.Т. Хвощ Н.Н. Варлинский Е.А. Попов “МИКРОПРОЦЕССОРЫ и МИКРОЭВМ в системах автоматического управления”

9 - https://github.com/espressif/

10 - https://nextion.itead.cc/

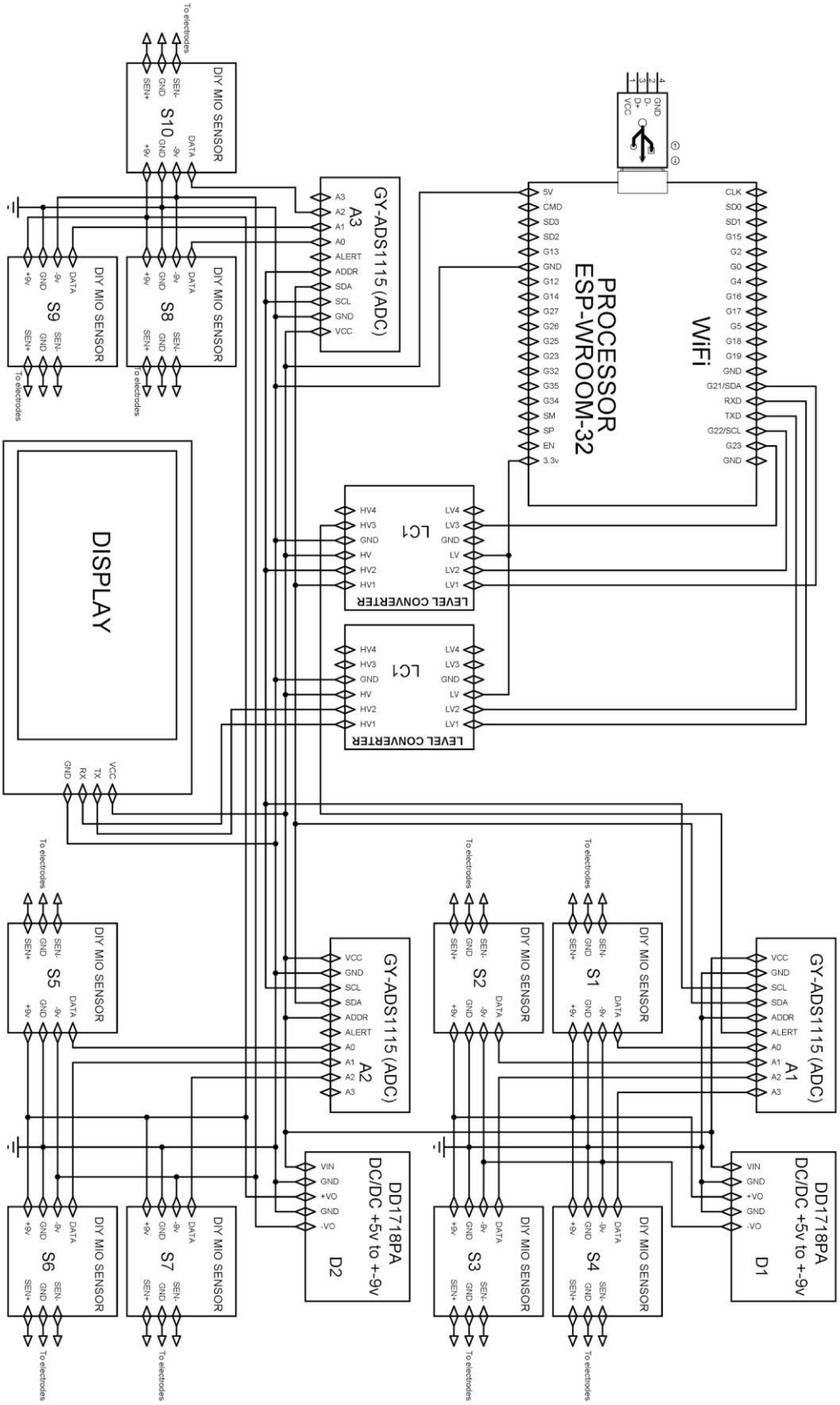
11 - https://www.arduino.cc/

12 - https://ru.wikipedia.org/

13 - http://www.lomonosov-fund.ru/enc/ru/encyclopedia/

Приложение А

“Блок схема основного



устройства”

Приложение Б

“Программа основного устройства”

```
#include "ads1115.h"    // чтение показаний 12 каналов датчиков
#include "radiosend.h"  // передача команд о состоянии пальцев на манипулятор
#include "screenobj.h"  // для подключения дисплея

#define SENSORS 10  // всего 10 датчиков
#define MATRSIZE 32 // размер матрицы пальцы - по два состояния 2 в степени 5

#define VECTORS // поиск осуществляется с помощью векторной математики
#define THRESHOLD_OK 80 // порог опознавания комбинации при поиске по матрице
#define THRESHOLD_DEV 15 // порог допустимых колебаний показаний, при удержании комбинации


uint8_t oldGauge[SENSORS]={0}; // на дисплее есть всего 10 градусникjd
uint8_t matr[MATRSIZE][SENSORS]; // матрица значений датчиков в процентах от показаний, 0,1,2,3 - показания датчиков, 100% -выбирается датчик с самыми мощными показаниями, строка матрицы соответствует пальцу
int16_t matrPow[MATRSIZE]; // в этом буфере сохраняем максимальные показания датчика для данного слепка, используем для вычисления усилия
uint8_t power=0; // ориентировочное усилие мышцы относительно эталонного
uint8_t oldPower=255; // предыдущее (старое) ориентировочное усилие мышцы относительно эталонного, для исключения повторной обработки
uint8_t sysStat=0; // системный статус сработки датчиков
uint8_t oldSysStat=0; // старый статус сработки датчиков
uint8_t calibrate=0; // статус калибровки
uint32_t calibrateDelay=0; // количество времени на текущую операцию калибровки
uint32_t calibrateHalfDelay=0; //буфер половинного времени стадии, время для сохранения измерений
uint32_t calibrateTimer=0; // буфер засечки этапов калибровки
uint8_t  maxDest=0; //количество жестов на которое настраиваем устройство
int16_t adc_data[12];      // полученные данные
int16_t adc_max[12];       // максимумы данных

uint8_t fingerStat[MATRSIZE]={         //карта состояния пальцев большой,указательный,средний,безымянный, мизинец
  B00001, // всего 8 простейших жестов
  B00010,
```

```
  B00100,
  B01000,
  B10000,
  B11111,
  B00000, // всего 17 простых жестов
  B01111,
  B00111,
  B00011,
  B11000,
  B11100,
  B01100,
  B01001,
  B10001,
  B11001,
  B01011,
  // сложные жесты
  B11110, // 25 простых+сложных
  B11101,
  B11011,
  B01101,
  B10011,
  B01110,
  B10111,
  B10101,
  // неудобные жесты
  B11010, // всегшо 32 комбинации пальцев
  B01010,
  B00101,
  B00110,
  B10010,
  B10100,
  B10110,

};

// управление показом стадий калибровки
void calibrateStage(byte stage){
  calibrate=stage; // установить новый этап калибровки
  calibrateTimer=millis(); //засечь таймер этапа
  byte st=128; // буфер статуса пальцев для показа на экране
```

```
word delayGaug=50; // шаг таймера градусника
byte stepGaug=1; // шаг градусника
String msg=""; //буфер сообщения для показа на экране
if (stage==1) { // стадия запуска калибровки
  powerGauge.setValue(256); // погасить спидометр
  powerTimer.enable(); // запустить таймер обработки спидометра
  for(byte i=0;i<SENSORS;i++){
    //shift[i]=0; // сброс смещения графика относительно ноля
    gr[i].setValue(0); // сброс проресс баров на экране
    oldGauge[i]=0;
    //_max[i]=1;
  }
  msg=F("Сожмите руку в кулак");
  calibrateDelay=4000; // установить продолжительность этапа 5 сек
  delayGaug=80; // шаг таймера градусника каждые 80 us
  stepGaug=2; // шаг градусника по два деления (80*100/2 = 4000)
  st+=1+2+4+8+16;
} else if (stage==2) { //
  msg=F("Полностью расслабьте руку");
  calibrateDelay=1500; // установить продолжительность этапа 1,5 сек
  delayGaug=60; // шаг таймера градусника каждые 60 us
  stepGaug=4; // шаг градусника по два деления (60*100/4 = 4000)
} else if (stage==3) {
  //for(byte i=0;i<SENSORS;i++){
    //shift[i]=adc_data[i]; // запоминаем смещение графиков относительно ноля
    //adc_data[i]=0; // применяем смещение к текущим данным
    //_max[i]=0; // чистим буфер максимумов
  //}
} else if (stage<(3+(maxDest*2))){
  stage-=3;
  if ((stage&1)==0){ // четные стадии (сгибания)
    msg=F("Согните пальцы выделенные красным");
    calibrateDelay=4000;
    delayGaug=80; // шаг таймера градусника каждые 80 us
    stepGaug=2; // шаг градусника по два деления (80*100/2 = 4000)
    st+=fingerStat[stage/2]; // картина пальцев из таблицы
  } else {
    msg=F("Разогните...");
    calibrateDelay=1500;
    delayGaug=60; // шаг таймера градусника каждые 60 us
```

```
      stepGaug=4; // шаг градусника по два деления (60*100/4 = 4000)
    }
  } else { // окончание калибровки
    msg=F("Калибровка окончена");
    calibrateDelay=1000;
    delayGaug=50; // шаг таймера градусника каждые
    stepGaug=100; // шаг градусника по два деления
    LearnButton8.setValue(1); // "отпустить кнопку обучения"
    LearnButton17.setValue(1); // "отпустить кнопку обучения"
    LearnButton25.setValue(1); // "отпустить кнопку обучения"
    LearnButton32.setValue(1); // "отпустить кнопку обучения"
    calibrate=0; // переход в нормальную работу
  }
  calibrateHalfDelay=calibrateDelay/2;//половина времени стадии
  UniToIso(&msg); // перекодировать сообщение в буфер
  ShowText.setText(msg.c_str()); // напечатать на дисплее
  fingerScreenStat.setValue(st); // установить статус показа пальцев
  metalHandSetFingers(st);
  nexClick(ShowText.getObjName(),0); //активировать статус
  gaugeStep.setValue(stepGaug);
  gaugeTimer.setCycle(delayGaug);
  gaugeTimer.enable();
}

// запуск калибровки
void startCalibrate8(void *ptr){
  if (calibrate==0) { //калибровка не запущена
    calibrateStage(1); //запускаем калибровку первая стадия калибровки
    maxDest=8; // учимся всего восьми комбинациям
  }
}
void startCalibrate17(void *ptr){
  if (calibrate==0) { //калибровка не запущена
    calibrateStage(1); //запускаем калибровку первая стадия калибровки
    maxDest=17; // учимся всего 17 комбинациям
  }
}
void startCalibrate25(void *ptr){
  if (calibrate==0) { //калибровка не запущена
    calibrateStage(1); //запускаем калибровку первая стадия калибровки
```

```
    maxDest=25; // учимся всего 25 комбинациям
  }
}
void startCalibrate32(void *ptr){
  if (calibrate==0) { //калибровка не запущена
    calibrateStage(1); //запускаем калибровку первая стадия калибровки
    maxDest=32; // учимся 32 комбинациям
  }
}

// расчет в процентах
uint8_t getProc(int16_t val, int16_t maxVal){
  if (val<0) { // отрицательные значения недопустимы, делить на 0 нельзя
    return 0;
  } else if (maxVal<1) {
    return val;
  }
  return (((long)val*100)/maxVal);
}

void setup(){
   Serial.begin(115200);
   Serial.println(F("Hallo"));
   nexInit();// инициализация дисплея
   String msg=F("Подключаемся к серверу !");
   ShowText.setText(msg.c_str());
   uint8_t test=1;
   while(test==1){ // запуск связи с сервером-рукой
     test=metallHandConnect();
     if(test==1) {
       msg=F("Попытка подключения к серверу");
       ShowText.setText(msg.c_str());
     }
   }
   if(test==2) {
     msg=F("Ошибка. Работаем без сервера !");
     delay(5000);
   } else {
     msg=F("Подключение успешно !");
     metalHandSetFingers((uint8_t)0); // выпрямить пальцы руки
```

```
    }
    ShowText.setText(msg.c_str());
    adcBegin(20); //запуск системы модулей ADC, с правильным коэфициентом измерений, для получения показаний
//    adc_data=getData();
//    adc_max=getMax();
    LearnButton8.attachPop(startCalibrate8);   // подключаем на виртуальную кнопку на дисплее подпрограмму запуска калибровки на 8 жестов
    LearnButton17.attachPop(startCalibrate17); // подключаем на виртуальную кнопку на дисплее подпрограмму запуска калибровки на 17 жестов
    LearnButton25.attachPop(startCalibrate25); // подключаем на виртуальную кнопку на дисплее подпрограмму запуска калибровки на 25 жестов
    LearnButton32.attachPop(startCalibrate32); // подключаем на виртуальную кнопку на дисплее подпрограмму запуска калибровки на 32 жеста
}


void loop() {

  // постоянно проверяем нет ли какого действия с элементами экрана
  nexLoop(nex_listen);

  // показываем прогресс бары на дисплее
  static uint8_t pbCount=0;
  long temp=getProc(adc_data[pbCount],adc_max[pbCount]); // расчитываем значение показаний датчика для вывода на экран в виде прогрессбара, в процентах от максимума
  // показываем показания на градуснике на экране
  if (oldGauge[pbCount]!=temp) {
     if (temp<=100) { // значения больше 100 воспринимаются монитором как ошибка
        gr[pbCount].setValue(temp);
        oldGauge[pbCount]=temp;
     }
  }
  if (pbCount<(SENSORS-1)){ // счет от 0 до 11
     pbCount++;
  } else {
     pbCount=0;
  }

  // обработка данных с датчиков
```

```
  while (adcAvailable()){ // данные готовы

    for(uint8_t i=0; i<10; i++){
      Serial.print(adc_data[i]);
      Serial.print(' ');
    }
    Serial.println();



     // если идет этап калибровки калибровки
     if (calibrate) {// идет калибровка
       if (millis()-calibrateTimer > calibrateDelay) {  // конец стадии калибровки
          calibrateStage(calibrate+1); // начало следующего этапа калибровки
       } else if ((calibrateHalfDelay!=0) && (millis()-calibrateTimer>calibrateHalfDelay)){ // середина стадии
          calibrateHalfDelay=0; // обнулить таймер сохранения данных, что бы не повторять операции
          // ищем максимумы измерений среди датчиков
          int16_t _maxBuff=1; // максимум стабилизации датчиков, не может быть меньше 1, поскольку на него
будем делить и это проценты
          int32_t _maxAbs=0;  // абсолютный максимум усилия для матрицы
          uint8_t k=(calibrate-3)/2; // вычисляем номер комбинации в таблице
          if (k<MATRSIZE) {
            for(uint8_t i=0;i<SENSORS;i++){ // перебираем показания датчиков
              if (_maxBuff<adc_data[i]) { // ищем максимальные
                _maxBuff=adc_data[i];
              }
              _maxAbs+=(adc_max[i]); // суммируем все максимумы показаний датчиков для усреднения
              adc_max[i]=0; // сбрасываем максимумы для сл. измерения
            }
            _maxAbs/=SENSORS; //получаем среднее среди всех показаний
            // заполняем матрицу измерений датчиков в процентах от максимума
            for (uint8_t i=0;i<SENSORS;i++){
              matr[k][i]=getProc(adc_data[i],_maxBuff);
            }
            // сохраняем показания соответствующие абсолютному максимуму данной матрицы пальца
            if (_maxAbs<1){
              matrPow[k]=1; // поскольку на это число будем делить и поскольку это проценты то не может быть
меньше 1
            } else {
              matrPow[k]=_maxAbs;
```

```
        }
      } else {
        Serial.println(F("Error calc matrix size !"));
      }
    }

  } else { // процесс работы (калибровка отключена)

    // ПОДБИРАЕМ НАИБОЛЕЕ ПОДХОДЯЩУЮ СТРОКУ МАТРИЦЫ ТЕКУЩЕМУ СЛЕПКУ ИЗМЕРЕНИЙ

    // ищем максимум измерений среди датчиков
    int16_t _maxBuff=1; // на это число будем делить, не может быть 0 или отрицательным, потому что это процентные вычисления
    int32_t _maxAbs=0; // тут будем вычислять средние показания датчиков
    for(uint8_t i=0;i<SENSORS;i++){
      if (_maxBuff<adc_data[i]) { // ищем максимум
        _maxBuff=adc_data[i];
      }
      _maxAbs+=adc_data[i]; // накапливаем сумму для вычисления среднего
    }
    _maxAbs/=SENSORS;
    // заполняем текущий слепок (матрицу) измерения датчиков в процентах от максимума
    uint8_t currMatr[SENSORS];
    for (uint8_t i=0;i<SENSORS;i++){
      currMatr[i]=getProc(adc_data[i],_maxBuff);
    }

    #ifdef VECTORS
      // считаем, что каждая строка матрицы n-мерный вектор, поскольку все вектора нормализованы (имеют одинаковую длинну,
      // связано это с тем, что в матрице хранятся показания в процентах от максимального) можно искать ближайший вектор
      // по наименьшему расстоянию между концами векторов, которое можно найти через корень суммы квадратов разностей координат
      // одноименных плоскостей. Раз мы ищем минимум, а сами показания нас не интересуют, можно не брать корень (это долгая операция)
      uint8_t num=maxDest; // буфер номера ближайшей комбинации
      uint32_t min=4294967295;
      for (uint8_t j=0;j<maxDest;j++) { // перебираем образцовые матрицы
        int32_t test=0;
```

```
          for(uint8_t i=0;i<SENSORS;i++){
            int16_t buf=currMatr[i]-matr[j][i];
            test+=buf*buf;
          }
          if (min>test){
            min=test;
            num=j;
          }
        }
      #else
        // расчитываем веса отклонения полученой матрицы от образцовых и ищем с минимальным отклонением
        uint16_t min=65535; // для поиска минимумов
        uint8_t num=maxDest; // буфер номера ближайшей комбинации
        for (uint8_t j=0;j<maxDest;j++) { // перебираем образцовые матрицы
          uint16_t test=0; // тут будем накапливать вес отклонения
          for(uint8_t i=0;i<SENSORS;i++){  // перебираем датчики в матрице и текущие
            if (currMatr[i]>matr[j][i]){ // суммируем отклонения всех датчиков от эталона в процентах
              test+=currMatr[i]-matr[j][i];
            } else {
              test+=matr[j][i]-currMatr[i];
            }
          }
          if ((test<THRESHOLD_OK) && (min>test)) { // ловим минимум отклонения и границу порога
чувствительности
            min=test;
            num=j; // при минимуме отклонений запоминаем подходящую строку матрицы
          }
        }
      #endif

      if(num<maxDest){ // был найден подходящий элемент матрицы
        sysStat=fingerStat[num] ; //устанавливаем соответствующий пальцевый статус
      } else {
        sysStat=0; // либо нет никакого статуса (рука разогнута)
      }

      // как дополнительный фильтр коротких выбросов используем порог силы, как порог изменения статуса
      power=getProc(_maxAbs,matrPow[num]); // вычисляем ориентировочное усилие
      if ((power<THRESHOLD_DEV) && (oldSysStat!=sysStat)) { // если это начало напряжения группы мышц,
отличной от предыдущей, но усилие не достигло 15% , учитывать будем (фильтр коротких выбросов)
```

```
  sysStat=0; // игнорируем измененение статуса
}

// показываем статус на экране
if (oldSysStat!=sysStat) { // статус изменился, нужно изменить данные на экране
  oldSysStat=sysStat; // уравниваем, что бы не повторять
  fingerScreenStat.setValue(sysStat); // установить статус показа пальцев
  metalHandSetFingers(sysStat,power); // отправить показания на сервер для работы манипулятора
  nexClick(ShowText.getObjName(),0); //активировать статус пальцев на экране
  // текстовое сообщение
  String msg=""; // создать правильное сообщение о согнутых пальцах
  if ((sysStat & (1+2+4+8+16))==(1+2+4+8+16)) {
    msg=F("КУЛАК");
  } else {
    if (sysStat & 1) {
      msg+=F("МИЗИНЕЦ, ");
    }
    if(sysStat & 2){
      msg+=F("БЕЗЫМЯННЫЙ, ");
    }
    if(sysStat & 4){
      msg+=F("СРЕДНИЙ, ");
    }
    if(sysStat & 8){
      msg+=F("УКАЗАТЕЛЬНЫЙ, ");
    }
    if(sysStat & 16){
      msg+=F("БОЛЬШОЙ, ");
    }
    if (msg.length()>2){ // если в строке что то есть, удалить два последних символа (запятую и пробел)
      msg.remove(msg.length()-2, 2);
    }
  }
  UniToIso(&msg); // перекодировать сообщение в буфер
  ShowText.setText(msg.c_str()); // напечатать на дисплее
}

// отображение усилия на спидометре
if (sysStat==0){ // если статус не опопзнан,
  power=0; // то и силу сжатия не показываем
```

```
      }
      if (oldPower!=power) { // только если усилие изменилось и не выходит за разумные пределы из-за очень маленького максимума
        metalHandSetFingers(sysStat,power); // отправить показания на сервер для работы манипулятора, с указанием мощности сжатия
        oldPower=power; // что бы не повторять дважды передачу одинаковых данных на экран
        powerGauge.setValue(power); // передать данные спидометру на экране, коэфициент введен для увеличения размаха показаний, просто так нагляднее
        powerTimer.enable(); // запустить таймер обработки спидометра
      }
    }
  }
```

Приложение В

“Программа демонстрационного стенда”

```
#include "EEPROM.h"

#ifdef ESP8266 // для процессора esp8266
  #include <ESP8266WiFi.h>
  #include <WiFiClient.h>
  #include <ESP8266WebServer.h>
  #include <Servo.h>
  #define PIN_FINGER_0 D4 // пины подключения сервоприводов пальцев
  #define PIN_FINGER_1 D3
  #define PIN_FINGER_2 D2
  #define PIN_FINGER_3 D1
  #define PIN_FINGER_4 D0
  Servo serv[5];
#else  // для процессора ESP32
  #include <WiFi.h>
  #define DUTY_LOW 2000 // заполнение ШИМ при минимальном занчении сервопривода
  #define DUTY_HIGH 7800 //при максимальном
  #define PWM_FREQUENCY 50 //частота PWM
  #define PWM_RESOLUTION 16 //Разрешение PMW
  #define PIN_FINGER_0 19 // пины подключения сервоприводов пальцев
  #define PIN_FINGER_1 18
  #define PIN_FINGER_2 5
  #define PIN_FINGER_3 17
```

```
  #define PIN_FINGER_4 16
#endif

#define AP_WIFI // если эта строка раскоментирована, то наше устройство - точка доступа
#ifdef AP_WIFI
  #define AP_SSID   "FingerServer"  // параметры точки доступа
  #define AP_PASS   "fspassword"
#else
  #define STA_SSID   "Test"  // измените эти параметры на параметры вашей сети
  #define STA_PASS   "password"
#endif

#define EEPROM_SIZE 32 // максимальный размер сохраняемых в независимой памяти данных

WiFiServer server(80); //экземпляр класса веб сервер

byte f_max[5]={180,180,180,180,180}; //массив содержащий маскимальные значения положения серв пальцев (градусы)
byte f_min[5]={0,0,0,0,0};//массив содержащий минимальные значения положения серв пальцев (градусы)
byte fval[5]={50,50,50,50,50}; //массив содержащий текущие значения положения серв пальцев (проценты)
byte f_pin[5]={PIN_FINGER_0, PIN_FINGER_1, PIN_FINGER_2, PIN_FINGER_3, PIN_FINGER_4}; //массив содержаищй контакнты подключений сервомашин пальцев

// переменные для клиента:
char linebuf[80] ;// буффер получения данных от клиента
int charcount=0; // указатель на текущее место в буфере

void setFingerPos(byte n, byte pos){//подпрограмма установки пальца n в значение c (проценты)
  static uint16_t oldPos[5]={0};   // буфер хранения старых показаний
  if ((n<5) && (pos<=100)){
    #ifdef ESP8266
      uint8_t value=map(pos,0,100,f_min[n],f_max[n]);    //Текущее положение в градусах
      if (oldPos[n]!=value) {
        oldPos[n]=value;
        serv[n].attach(f_pin[n]);          // активируем серву
        serv[n].write(value);
      }
    #else
      uint16_t maximum=map(f_max[n],0,180,DUTY_LOW,DUTY_HIGH); //максимум в единицах PWM
      uint16_t minimum=map(f_min[n],0,180,DUTY_LOW,DUTY_HIGH); //Минимум в единицах PWM
```

```
      uint16_t value=map(pos,0,100,minimum, maximum);          //Текущее положение в единицах PWM
      if (oldPos[n]!=value) {
        oldPos[n]=value;
        ledcSetup(n,PWM_FREQUENCY,PWM_RESOLUTION); // настриваем канал PWM
        ledcAttachPin(f_pin[n], n); //подключаем каналы к выходам
        ledcWrite(n, value); //Устанавливаем серву в необоходимое положение
      }
    #endif
  }
}

void setup() {
  // инициализируем последовательную коммуникацию
  // и ждем открытия порта:
  Serial.begin(115200);
  while(!Serial) {
    ; // ждем подключения последовательного порта
      // (нужно только для штатного USB-порта)
  }
  EEPROM.begin(EEPROM_SIZE);
  for (byte i = 0; i < sizeof(f_max); i++){
    //настраиваем сервоприводы
    #ifdef ESP8266
      serv[i].attach(f_pin[i]);
    #else
      ledcSetup(i,PWM_FREQUENCY,PWM_RESOLUTION); // настриваем канал PWM
      ledcAttachPin(f_pin[i], i); //подключаем каналы к выходам
    #endif
    //считываем сохраненные данные привода из eeprom
    byte temp=EEPROM.read(i);
    if (temp<181){
     f_max[i]=temp;
    }
    temp=EEPROM.read(sizeof(f_max)+sizeof(fval)+i);
    if (temp<181){
     f_min[i]=temp;
    }
    setFingerPos(i,fval[i]);
  }
  // начинаем с подключения к WiFi-сети:
```

```
  Serial.println();
  Serial.println();
  WiFi.disconnect();
  WiFi.softAPdisconnect();

  #ifdef AP_WIFI
    // работа как точка доступа
    WiFi.mode(WIFI_AP); // режим как точка доступа
    Serial.print("Create WIFI net: ");  //  "Подключаемся к "
    Serial.println(AP_SSID);
    WiFi.softAP(AP_SSID,AP_PASS); // запускаем нашу сеть
    Serial.print("Password for connect: ");  //  "Подключаемся к "
    Serial.println(AP_PASS);
    Serial.print("IP address: ");    //  "IP-адрес: "
    Serial.println(WiFi.softAPIP());
  #else
    WiFi.mode(WIFI_STA); // режим как станция
    Serial.print("Connecting to ");  //  "Подключаемся к "
    Serial.println(STA_SSID);
    WiFi.begin(STA_SSID, STA_PASS); // если к открытой сети, то заменить пароль на пустой
    while(WiFi.status() != WL_CONNECTED) {// пытаемся подключиться к WiFi-сети:
      delay(500);
      Serial.print(".");
    }
    Serial.println();
    Serial.println("WiFi connected");  //  "Подключение по WiFi выполнено"
    Serial.print("IP address: ");    //  "IP-адрес: "
    Serial.println(WiFi.localIP());
  #endif
  server.begin(); // запуск веб сервера
}

void loop() {
  WiFiClient client = server.available(); // анализируем канал связи, ждем подключения клиента:
  if (client) {
    Serial.println("New client");  //  "Новый клиент"
    memset(linebuf,0,sizeof(linebuf));// чистим буфер для клиента
    charcount=0;
    // HTTP-запрос заканчивается пустой строкой:
    boolean currentLineIsBlank = true; // переменная для контроля пустой строки
```

```
   while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      linebuf[charcount]=c;// считываем HTTP-запрос, символ за символом в буфер
      if (charcount<sizeof(linebuf)-1) charcount++;
      // если добрались до конца строки (т.е. получили
      // символ новой строки) и строка пуста,
      // это значит, что HTTP-запрос закончился;
      // следовательно, можно отправлять ответ:
      if (c == '\n' && currentLineIsBlank) {
        // Формируем HTML страницу
        // отправляем стандартный заголовок HTTP-ответа:
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
              //  "Тип контента: text/html"
        client.println("Connection: close");
              //  "Соединение: отключено";
              //  после отправки ответа связь будет отключена
        client.println(); // ниже отправка содержимого HTML страницы
        client.println("<!DOCTYPE HTML><html><head>");
        client.println("<meta name=\"viewport\"content=\"width=device-width, initial-scale=1\"></head>");
        client.println("<h1>Finger server</h1>");
        client.println("<form>FINGER       1                 2
              3              &nbsp4 
              &nbsp5<BR>");
        client.print("MAX     <input type=\"number\"size=\"3\"name=\"1x\"min=\"0\"max=\"180\"value=\"");
        client.print(f_max[0]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"2x\"min=\"0\"max=\"180\"value=\"");
        client.print(f_max[1]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"3x\"min=\"0\"max=\"180\"value=\"");
        client.print(f_max[2]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"4x\"min=\"0\"max=\"180\"value=\"");
        client.print(f_max[3]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"5x\"min=\"0\"max=\"180\"value=\"");
        client.print(f_max[4]);
        client.println("\"><input type=\"submit\"value=\"=>\"title=\"send\"></form><form>");
        client.print("VAL       <input
type=\"number\"size=\"3\"name=\"1\"min=\"0\"max=\"180\"value=\"");
        client.print(fval[0]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"2\"min=\"0\"max=\"180\"value=\"");
```

```
        client.print(fval[1]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"3\"min=\"0\"max=\"180\"value=\"");
        client.print(fval[2]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"4\"min=\"0\"max=\"180\"value=\"");
        client.print(fval[3]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"5\"min=\"0\"max=\"180\"value=\"");
        client.print(fval[4]);
        client.println("\"><input type=\"submit\"value=\"=>\"title=\"send\"></form><form>");
        client.print("MIN       <input
type=\"number\"size=\"3\"name=\"1n\"min=\"0\"max=\"180\"value=\"");
        client.print(f_min[0]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"2n\"min=\"0\"max=\"180\"value=\"");
        client.print(f_min[1]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"3n\"min=\"0\"max=\"180\"value=\"");
        client.print(f_min[2]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"4n\"min=\"0\"max=\"180\"value=\"");
        client.print(f_min[3]);
        client.print("\"><input type=\"number\"size=\"3\"name=\"5n\"min=\"0\"max=\"180\"value=\"");
        client.print(f_min[4]);
        client.println("\"><input type=\"submit\"value=\"=>\"title=\"send\"></form></html>");
        break;
      }

      if (c == '\n') {// началась новая строка
        currentLineIsBlank = true; // начинаем новую строчку:
        if (strstr(linebuf,"GET /?") > 0){// если это строка GET запроса
          char* start=strstr(linebuf,"?"); // ищем в ней наличие данных
          while (start) {
            byte number=0; // сюда считаем номер пальца
            byte type=0; // тут будет признак данных
            int value=255; // сюда считаем значение параметра
            if ((start[0]>='1') && (start[0]<='5')){ // считываем номер пальца
              number=start[0]-'0'-1;
              if ((start[1]=='x') && (start[2]=='=')){ //если второй смвол 'x', то это максимум
                type=1; // тип данных максимум
                start+=2; //значит обработали два символа
              } else if (start[1]=='='){ //если второй смвол '=', то это положение пальца
                type=2; // тип данных текущее значение
                start+=1; //значит обработали один символ
              } else if ((start[1]=='n') && (start[2]=='=')) {//если второй смвол 'n', то это минимум
```

```
  type=3; // тип данных минимум
  start+=2; // обработали два символа
} else { // значит ошибочные данные
  break; // сбрасываем анализ
}
if (type>0){ // если тип был считан верно
  if (start[0]=='='){//считываем значение параметра
   start++; //указатель на символ
   if ((start[0]>='0') && (start[0]<='9') ){ //проверим что цифра
    value=start[0]-'0';
    start++;
    if ((start[0]>='0') && (start[0]<='9')) { //проверим что цифра
     value*=10;
     value+=start[0]-'0';
     start++;
     if ((start[0]>='0') && (start[0]<='9')) { //проверим что цифра
       value*=10;
       value+=start[0]-'0';
     }
    }
   }
  } else { // ошибка данных
   break;
  }
  if ((value<=180) && (type>0)){ // параметр не может быть больше 180 градусов
   if (type==1){ // если считывали максимум
    f_max[number]=value; // обновить данные максимума
    Serial.print(number);
    Serial.print(" MAX=");
    Serial.println(f_max[number]);
    EEPROM.write(number,f_max[number]); // сохранить параметр в епроме
    setFingerPos(number,fval[number]); // установить палец в соотвествии с новыми значениями
   } else if (type==2){ // положение пальца в процентах
    if (value>100){ // проценты не должны быть более 100
     break;
    }
    fval[number]=value;  // обновить данные текущего положения пальца
    Serial.print(number);
    Serial.print(" Value=");
    Serial.println(fval[number]);
```

```
                    // EEPROM.write(number+sizeof(f_max),fval[number]);
                    setFingerPos(number,fval[number]); // установить палец в соотвествии с новыми значениями
                   } else if (type==3){
                    f_min[number]=value;// обновить минимум
                    Serial.print(number);
                    Serial.print(" MIN=");
                    Serial.println(value);
                    EEPROM.write(number+sizeof(f_max)+sizeof(fval),f_min[number]);// сохранить параметр в епроме
                    setFingerPos(number,fval[number]); // установить палец в соотвествии с новыми значениями
                   }
                  } else { // если параметр ошибочный, далее не анализируем
                    break;
                  }
                 }
               }
               start++;
               if ((start[0]=='\n')|| (start[0]=='\r')|| (start[0]==0)){ // если строка закончилась
                 break; // больше параметры не ищем
               }
             }
             EEPROM.commit();//сохранить изменения епрома
           }
           currentLineIsBlank = true; // начинаем новую строчку:
           memset(linebuf,0,sizeof(linebuf)); // очищаем буфер приема
           charcount=0;
          } else if (c != '\r') { // в строке попался новый символ:
           currentLineIsBlank = false;
          }
         }
        }
        // даем веб-браузеру время, чтобы получить данные:
        delay(1);
        // закрываем соединение:
        client.stop();
        Serial.println("client disconnected");  //  "клиент отключен"
      }
    }
```