ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

75

регистрационный номер

Робототехника и комплексная автоматизация
название факультета

Системы автоматизированного проектирования
название кафедры

Автоматизация деятельности в труднодоступных или опасных для человека условиях
название работы

Автор: ***Марадудина Вероника Валерьевна***
фамилия, имя, отчество
МБОУ «Лицей № 65», 11А
наименование учебного заведения, класс

Научный руководитель: **Берчун Юрий Валерьевич**
фамилия, имя, отчество
МГТУ им. Н. Э. Баумана
место работы
Старший преподаватель
звание, должность

_____
подпись научного руководителя

Москва - 2019

Аннотация

Целью данной работы является предложение образцов элементов человекоподобного робота для решения задачи автоматизации действий в условиях, непригодных для человека. Для достижения данной цели был выполнен ряд шагов:

- Анализ перспективы применения образцов на практике;
- Поиск оптимальных механических решений;
- Поиск и разработка управляющих схем;
- Разработка программ для микроконтроллеров;
- Печать корпуса на 3D-принтере;
- Осуществление сборки корпуса;
- Сборка аппаратной части;
- Написание и тесты кода.

В разработке и сборке образцов использовались следующие методы и приёмы: проектирование цифровых моделей, перевод цифровых моделей в .stl формат, склейка пластиковых деталей при помощи ацетона, пайка деталей на макетных платах, программирование в Arduino IDE, прошивка микроконтроллеров, соединение микроконтроллеров по Bluetooth, использование сервоприводов для точной передачи движений, разделение управляющей части и модели для избавления от ограничений передвижения. Полученный результат представляет собой действующие модели манипуляторов, способных воспроизводить движения рук. Это позволяет без вреда для здоровья оператора использовать образцы в опасных условиях, например, таких как химическое загрязнение, вредные выбросы, низкие или высокие температуры. Применение результатов данной работы минимизирует риски нанесения вреда человеку при чрезвычайных ситуациях или ситуациях, где необходимо, но невозможно срочное вмешательство.

Оглавление

ВВЕДЕНИЕ 4

1. ТЕОРЕТИЧЕСКИЙ РАСЧЁТ 5

1.1. РАЗРАБОТКА КОРПУСА 5

1.2. ОПТИМИЗАЦИЯ ДЕТАЛЕЙ ПОД ПРЕДЛОЖЕННЫЕ РЕШЕНИЯ 5

1.3. ОПИСАНИЕ ПРИНЦИПА УПРАВЛЕНИЯ 7

1.4. РАЗРАБОТКА УПРАВЛЯЮЩИХ СХЕМ 7

1.5. РАСЧЁТ ПИТАНИЯ 8

2. РЕАЛИЗАЦИЯ 10

2.1. НАПИСАНИЕ ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ 10

2.2. ВНЕДРЕНИЕ BLUETOOTH-МОДУЛЕЙ 10

2.3. СРАВНЕНИЕ ПРОВОДНОГО И БЕСПРОВОДНОГО ТИПА УПРАВЛЕНИЯ 11

2.4. ПЕЧАТЬ КОРПУСА 12

3. СЕБЕСТОИМОСТЬ РАЗРАБОТКИ 12

4. СРАВНЕНИЕ С АНАЛОГАМИ 13

5. ПРЕДЛОЖЕНИЯ ПО ПРАКТИЧЕСКОМУ ВНЕДРЕНИЮ 13

ЗАКЛЮЧЕНИЕ 14

СПИСОК ИСТОЧНИКОВ 14

ПРИЛОЖЕНИЯ 14

Введение

Согласно словарю, автоматизация – одно из направлений научно-технического прогресса, использующее саморегулирующие технические средства и математические методы с целью освобождения человека от участия в процессах получения, преобразования, передачи и использования энергии, материалов, изделий или информации, либо существенного уменьшения степени этого участия или трудоёмкости выполняемых операций.

По прогнозам экспертов, к 2030 году современные технологии могут обеспечить частичную автоматизацию до 60% трудовой деятельности [1]. В большей степени это коснётся профессий, связанных с физическим или монотонным трудом. Под группу риска не попадают исследовательские сферы деятельности, но что же делать в чрезвычайных ситуациях, где работа человека не представляется возможной, например, в токсичной среде или безвоздушном пространстве? В данной работе рассматривается один из возможных способов решения этой проблемы.

Целью работы является создание робота, управляемого оператором (далее – ZEN). ZEN может работать без непосредственной близости оператора, что позволяет использовать его без риска для человеческого здоровья

В работе ZEN представляет собой пару рук–манипуляторов, способных к воспроизведению действий оператора.

Состоит ZEN из управляющей и исполнительной части. К управляющей части относятся перчатки с резисторами изгиба, переключатель для запястий, две Arduino Mega, соединённые по Bluetooth. К исполнительной части относятся руки-манипуляторы, конструктивно состоящие из кисти, запястья и предплечья, аппаратно состоящие из десяти сервоприводов SG90, двух сервоприводов MG995, блока питания мощностью 90 Вт.

1. Теоретический расчёт

1.1. Разработка корпуса

При поиске оптимальных механических решений для корпуса ZEN был найден InMoov [2] – open source проект французского дизайнера, запущенный в 2012 году. Любой желающий может скачать цифровые модели расширения .stl (расширение файла, позволяющее слайсерам[1] воспринимать его) и распечатать их сам. В состав проекта InMoov входит много блоков, например, такие, как руки, плечи, голова, шея и челюсть, глазной механизм. Для данного проекта решено было взять только детали рук до локтя.

Но после изучения предлагаемых решений мною были внесены некоторые коррективы: управляющие пальцами сервоприводы MG995 были заменены на SG90 (подробное сравнение приводится в Таблица 1). Это позволило удешевить сборку и уменьшить массу конструкции без потери эффективности.

Таблица 1 – Сравнение сервоприводов SG90 и MG995

Характеристики	SG90	MG995
Масса	9 г	55 г
Крутящий момент	1,2 кг/см – 1,4 кг/см	8,5 кг/см – 10 кг/см
Напряжение	4,8 В – 6 В	4,8 В – 7,2 В
Стоимость	100 рублей	250 рублей

1.2. Оптимизация деталей под предложенные решения

Так как сервоприводы были заменены, то для корректной работы проекта внешние шестерни были переделаны под SG90 (Рисунок 1), а для самих сервоприводов в программе Meshmixer [3] созданы держатели для крепления

[1] Слайсер – программа для 3D печати, разбивающая трёхмерную модель на слои, тем самым подготавливающая её к печати на 3D-принтере.

(Рисунок 2). В новой шестерне диаметр внутреннего круга уменьшен с 8 мм до 7 мм, тогда как внешний круг остался таким же.

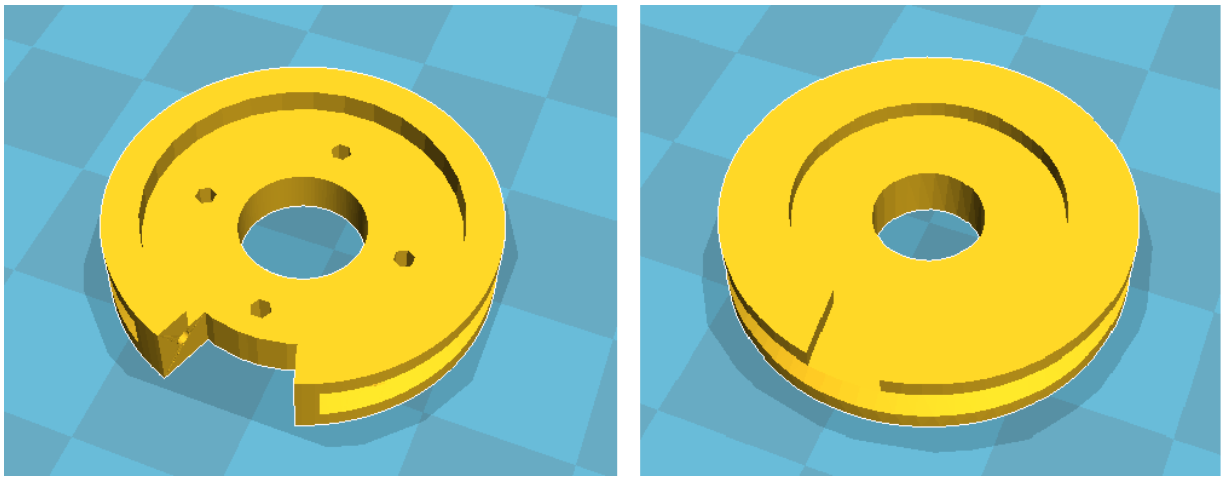

Рисунок 1 - старая и новая шестерни

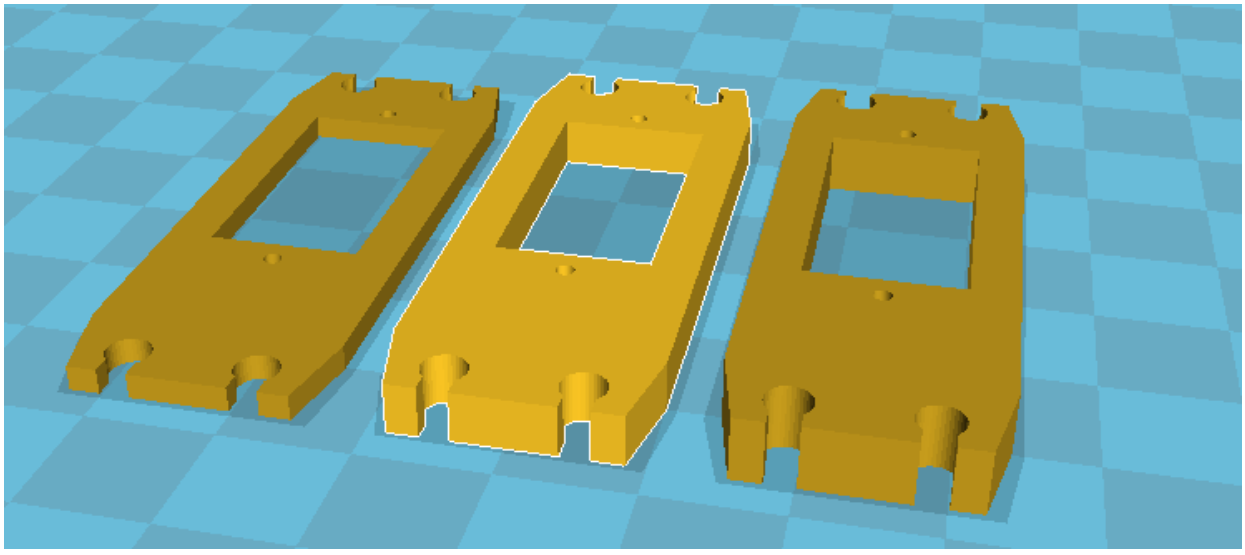

Рисунок 2 - различные держатели для SG90

Как можно увидеть из рисунка 2, не все держатели имеют одинаковую толщину. Это обусловлено тем, что деталь, на которую крепятся сервоприводы, имеет разную высоту, дабы избежать столкновения колёс. Так как SG90 меньше, чем MG995, это удаётся сделать, устанавливая держатели попеременно разными сторонами.

1.3. Описание принципа управления

Принцип работы устройства таков: шестерня сервопривода в начальном положении и конец пальца соединены нейлоновой леской. При повороте шестерни леска тянет за собой палец, способствуя его сгибу. Общая длина перемещения лески равна половине окружности шестерни (формула (1)).

$$l = \pi r, \tag{1}$$

где l – длина полуокружности шестерни, см;

r – радиус полуокружности, см.

Итого леска перемещается на 4,5 см, а палец сгибается полностью.

Запястье же управляется тумблером и имеет два положения: лицевое и тыльное.

1.4. Разработка управляющих схем

Для управления пальцами используются перчатки с самодельными резисторами изгиба (далее – РИ) [4]. РИ состоит из фоторезистора, светодиода и непрозрачной соединяющей трубки. В сравнении с готовыми резисторами изгиба РИ практически не уступают в точности, но в десятки раз уступают в цене.

Принцип работы РИ таков: включается светодиод, фоторезистор передаёт аналоговый сигнал, значение которого зависит от освещённости, на Arduino. При сгибании трубки на фоторезистор попадает меньшее количество света, он меняет своё сопротивление, Arduino считывает его и меняет позицию сервопривода. Схема РИ представлена на Рисунок 3.

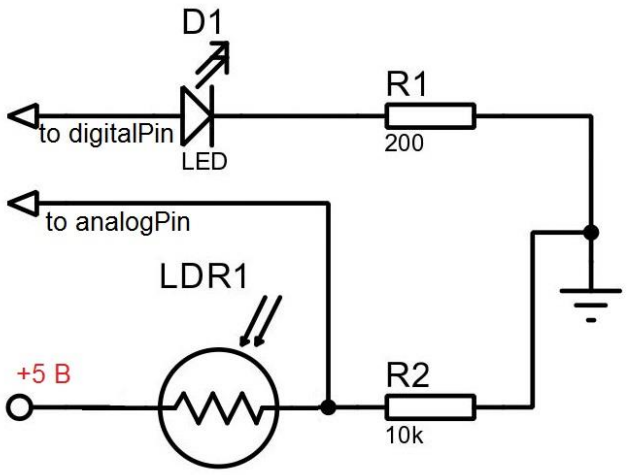

Рисунок 3 - принципиальная схема РИ

Для реализации такого принципа программа Arduino использует функцию map(), которая пропорционально преобразует аналоговое значение датчика в диапазон позиций сервопривода 0..180. Аналоговое значение каждого РИ неидеально, а поэтому диапазон начальных значений каждого датчика проверялся экспериментально. В среднем максимальное значение равно 900, а минимальное 300.

Для управления сервоприводом запястья используется переключатель. Он демонстрирует переключение между двумя положениями сервопривода – 0° и 180°, что для демонстрации вполне достаточно.

1.5. Расчёт питания

Для обеспечения питания исполнительной части взят блок питания модели LRS-100-5 на 5 В мощностью 90 Вт. Такой выбор обусловлен тем, что сервоприводы SG90 стабильно работают при напряжении 5 В, а при пиковой нагрузке могут выдавать до 1 А. Сервоприводы MG995 также работают при напряжении 5 В, при пиковой нагрузке могут выдавать до 1,5 А.

$$I_{общ} = I_1 + I_2 + .. + I_{12}, \quad (2)$$

где $I_{общ}$ – общий максимальный ток нагрузки, А;

I_1, I_2, … I_{12} – максимальный ток нагрузки каждого из сервоприводов, А.

$$U_{общ} = U_1 = U_2 .. = U_{12}, \quad (3)$$

где $U_{общ}$ – напряжение питания схемы, В;

U_1, U_2, … U_{12} – напряжение питания каждого сервопривода, В.

$$P = UI, \quad (4)$$

где P – мощность, Вт;

I – потребляемый схемой ток, А;

U – общее напряжение схемы, В.

Так как сервоприводы подключены параллельно, то при помощи вычислений по формулам (2), (3) и (4) можно получить максимально потребляемый ток, равный 13 А, и достаточную мощность, равную 65 Вт.

Помимо сервоприводов, к исполнительной части так же подключена одна управляющая перчатка (о причинах этого ниже, в пункте 2.3), и одна Arduino Mega 2560. Для стабильной работы платформе требуется напряжение питания, равное 9 В, а поэтому был установлен DC/DC преобразователь AM2D-0509SH40Z, преобразующий напряжение из (5,0±0,5) В в 9 В.

Для обеспечения питания управляющей части взята батарейка типа «крона» на 9 В и стабилизатор напряжения КР142ЕН5В, предоставляющий фиксированное выходное напряжение 5 В, рассчитанный на ток до 1,5 А. К стабилизатору подключены РИ. Arduino питается напрямую от батарейки, к платформе подключён HM-10.

2. Реализация

2.1. Написание программ для микроконтроллеров

Для программного управления используются две платы Arduino Mega 2560. Выбор именно этих микроконтроллеров обусловлен тем, что на каждой имеется 16 аналоговых выхода и 54 цифровых, что позволит легко расширить функционал в будущем. Объём flash-памяти равен 256 кБ. Обе платы программировались в Arduino IDE. [5]

Первоначально в проекте была только одна Arduino, управляющая одной рукой-манипулятором по следующему принципу: в переменную записывается значение с РИ, которое потом преобразовывается в диапазон значений угла поворота сервопривода.

2.2. Внедрение Bluetooth-модулей

При создании второй руки принято решение реализовать управление по Bluetooth. Для этого были взяты два модуля HM-10, преобразователь USB-TTL UART и ещё одна Arduino Mega.

HM-10 - модуль Bluetooth 4.0, основанный на архитектуре приемопередатчика CC2541 с беспроводной технологией Bluetooth с низким энергопотреблением (BLE). Он имеет компактный размер и может выступать как в роли инициатора соединения (master), так и в роли принимающего соединение (slave). [6]

Сначала необходимо было связать модули между собой по модели взаимодействия «master – slave» (ведущий – ведомый), прошив их при помощи Arduino IDE и преобразователя. Для этого к компьютеру подключается «slave» модуль и производится сеанс связи с модулем через монитор последовательного порта для выяснения его имени, PIN-кода, скорости работы и MAC-адреса. Далее «slave» модуль отключается, к компьютеру подключается «master» модуль. Ведущему задаётся PIN, адрес и скорость ведомого, и при удачном соединении при каждом включении модули будут образовывать пару между собой. [7]

«Master» платформа включает светодиоды, принимает показания с РИ, конвертирует их и по serial-порту передаёт на «slave» платформу. Так как последовательный порт передаёт данные побайтово, был реализован протокол передачи (Приложение 1 – программный код управляющей части).

В свою очередь, «slave» плата принимает поток данных, декодирует его при помощи протокола и задаёт углы поворота сервоприводов (Приложение 2 – программный код исполнительной части

Принцип работы протокола передачи таков: на ведущем модуле отправляются последовательно: маркер начала пакета, длина переменной, отвечающей за номер сервопривода, сама переменная, маркер середины пакета, длина значения данного РИ, значение РИ, маркер конца пакета. На ведомом модуле пакет расшифровывается, значения РИ записываются в подходящие переменные значений сервоприводов. [8]

Сделано это для предотвращения присваивания переменным посторонних значений. Контроллер ждёт и записывает данные именно в таком порядке, если какая-то часть пакета потеряна, происходит полный сброс протокола.

2.3. Сравнение проводного и беспроводного типа управления

Соединение по Bluetooth выгодно тем, что осуществлять управление можно без ограничений передвижения на расстоянии до 5 м. Но недостатков при таком способе управления явно больше. Задержка около секунды, риск потери части пакета из-за большого объёма принимаемых данных, даже на минимальной скорости 9600 Бод. В начальном положении сервопривода это менее заметно, так как при потере части пакета протокол сбрасывает все параметры, но в конечном сброс происходит довольно часто, вследствие чего сервоприводы стремятся вернуться в начальное положение, что может привести к механическим повреждениям.

Также было экспериментально установлено, что связь между модулями нестабильна и может теряться вблизи других источников сигналов, а соединение не всегда происходит с первого раза.

Опираясь на вышенаписанное, можно сделать вывод, что проводной тип управления более выгоден как в плане точности, так и в плане скорости.

2.4. Печать корпуса

Все детали печатались при температуре пластика 235℃, температуре стола принтера 101℃, с заполнением 25% на 3D-принтере модели Prusa i3. При печати был использован пластик ABS. Если сравнивать два наиболее ходовых типа пластика – ABS и PLA, то хоть PLA является более экологичным и менее подвержен усадке при печати, но данный вид пластика начинает плавиться при 90℃, что делает его восприимчивым к перепадам температур. ABS же, напротив, имеет высокую температуру плавления (220℃), легок для шлифовки и склейки. Детали из ABS клеятся при помощи ацетона, тогда как для PLA требуется дихлорметан либо дихлорэтан. ABS также обладает более высокой прочностью и не подвержен разложению. [9]

На печать всех деталей ушло около 10 суток в сумме. Для лучшей адгезии[2] поверхность принтера смазывалась смесью ацетона и растворённого в нём пластика. Детали обрабатывались при помощи шлифовальной бумаги и инструментов.

3. Себестоимость разработки

Пластик для печати – 2500 рублей

Пальцевые сервоприводы – 1000 рублей

Кистевые сервоприводы – 500 рублей

Блок питания – 1000 рублей

Электронные компоненты – 1000 рублей

Arduino Mega – 1800 рублей

[2] Адгезия – сцепление поверхностей разнородных твёрдых и/или жидких тел.

Прочие расходы – 1000 рублей

Всего: 7800 рублей

4. Сравнение с аналогами

Безусловно, прямым аналогом является робот InMoov. Ниже приведены положительные и отрицательные отличия ZEN:

Преимущества:

- Стоимость ниже в 2 раза из-за замены сервоприводов;
- Реализовано управление путём механического копирования движений рук, тогда как в InMoov оно осуществляется при помощи технологии Leap Motion[3];
- Облегчённая конструкция.

Недостатки:

- Нет датчиков нажима на пальцах;
- InMoov способен удерживать более тяжёлые предметы.

Стоит отметить, что в сравнительной характеристике не рассмотрены дополнительные функции InMoov, такие как поворот туловища или голосовое управление, так как они не входят в тематику данной работы.

5. Предложения по практическому внедрению

ZEN может быть использован:

- в химических исследовательских лабораториях, при работе в которых имеются риски для здоровья;
- в космической отрасли, что позволит избавить космонавтов от вынужденных выходов в открытый космос для ремонтирования обшивки или иных внешних проблем;
- при взаимодействии с буйными пациентами;

[3] Leap Motion – разрабатываемая технология, основанная на захвате движения, для человеко-компьютерного взаимодействия. Средняя цена Leap Motion Controller – 7500 рублей.

- в областях, связанных с животным миром;
- в чрезвычайных ситуациях, например, при ликвидации последствий пожаров и землетрясений.

Заключение

В результате проделанной исследовательской и практической работы была создана рабочая модель робота ZEN, включающего в себя два манипулятора, управляемых оператором; написан протокол передачи данных с управляющих элементов на ZEN; предложены оптимизационные механические решения; проведено сравнение управления напрямую и по Bluetooth, исследована перспектива внедрения устройства в различные сферы человеческой жизни.

В будущем возможны модификации данной работы. К ним может относиться добавление в проект других блоков, таких как локтевые и плечевые механизмы, голосовое управление, система распознавания лиц. Также возможно подключение Wi-Fi модуля для обеспечения стабильного соединения на расстоянии, камеры для передачи изображения на ПК.

Список источников

1. https://hitech.newsru.com/article/30nov2017/robots800mln
2. http://inmoov.fr/
3. http://3dprintstory.org/tutorial-po-meshmixer-dlya-nachinayuschih
4. https://habr.com/ru/post/225111/
5. http://arduino.ru/
6. https://voltiq.ru/hm-10-as-cheap-ibeacons/
7. https://howtomechatronics.com/tutorials/arduino/how-to-configure-pair-two-hc-05-bluetooth-module-master-slave-commands/
8. http://robocraft.ru/blog/1090.html
9. https://rusabs.ru/blogs/blog/razlichie-mezhdu-abs-i-pla-dlya-3d-pechati

Приложение 1 – программный код управляющей части

```
class Flex
{
  public:
  int flexPin;
  int servoNum;
  int flex;
  int value;
  int maxFlex;
  int minFlex;

  void ReadSend()
  {
    flex = analogRead(flexPin);
    value = map(flex, maxFlex, minFlex, 0, 180);
    value = constrain(value, 0, 180);
    String servoStr = String(servoNum);
    String valueStr = String(value);

    Serial1.print("$");
    Serial1.write(servoStr.length());
    Serial1.print(servoNum);
    Serial1.print("*");
    Serial1.write(valueStr.length());
    Serial1.print(value);
    Serial1.print("@");
  }
};

Flex index;
Flex middle;
Flex ring;
Flex little;
Flex thumb;

int lightPin1 = 30;
int lightPin2 = 31;
int lightPin3 = 32;
int lightPin4 = 33;
int lightPin5 = 34;
int wristPin = 35;

int toggle = 36;


void setup()
{
  Serial1.begin(9600);
  index.flexPin = A1;
  middle.flexPin = A2;
  ring.flexPin = A3;
  little.flexPin = A4;
  thumb.flexPin = A5;
```

```
  pinMode(index.flexPin, INPUT);
  pinMode(middle.flexPin, INPUT);
  pinMode(ring.flexPin, INPUT);
  pinMode(little.flexPin, INPUT);
  pinMode(thumb.flexPin, INPUT);
  pinMode(toggle, INPUT);

  pinMode(lightPin1, OUTPUT);
  pinMode(lightPin2, OUTPUT);
  pinMode(lightPin3, OUTPUT);
  pinMode(lightPin4, OUTPUT);
  pinMode(lightPin5, OUTPUT);

  index.servoNum = 1;
  middle.servoNum = 2;
  ring.servoNum = 3;
  little.servoNum = 4;
  thumb.servoNum = 5;

  index.maxFlex = 610;
  middle.maxFlex = 920;
  ring.maxFlex = 930;
  little.maxFlex = 910;
  thumb.maxFlex = 780;

  index.minFlex = 15;
  middle.minFlex = 290;
  ring.minFlex = 420;
  little.minFlex = 435;
  thumb.minFlex = 340;

  delay(300);
}

void loop()
{
  digitalWrite(lightPin1, HIGH);
  digitalWrite(lightPin2, HIGH);
  digitalWrite(lightPin3, HIGH);
  digitalWrite(lightPin4, HIGH);
  digitalWrite(lightPin5, HIGH);

  index.ReadSend();
  middle.ReadSend();
  ring.ReadSend();
  little.ReadSend();
  thumb.ReadSend();

  int wristNum = 6;
  String wristStr = String(wristNum);
  boolean wristVal = digitalRead(toggle);
  String wristSV = String(wristVal);

  Serial1.print("$");
  Serial1.write(wristStr.length());
  Serial1.print(wristNum);
```

```
  Serial1.print("*");
  Serial1.write(wristSV.length());
  Serial1.print(wristVal);
  Serial1.print("@");
}
```

Приложение 2 – программный код исполнительной части

```
#include <Servo.h>

String startMarker;
String stopMarker;
String middleMarker;
String positionString;
String servoString;
int startMarkerStatus;
int stopMarkerStatus;
int middleMarkerStatus;
int servoLength;
int positionLength;
boolean packetAvailable;

Servo index, middle, ring, little, thumb, finger1, finger2,
finger3, finger4, finger5, wristLeft, wrist;
int indexPin = 30;
int middlePin = 31;
int ringPin = 32;
int littlePin = 33;
int thumbPin = 34;

int servoPin1 = 35;
int servoPin2 = 36;
int servoPin3 = 37;
int servoPin4 = 38;
int servoPin5 = 39;
int flexPin1 = A1;
int flexPin2 = A2;
int flexPin3 = A3;
int flexPin4 = A4;
int flexPin5 = A5;
int lightPin1 = 40;
int lightPin2 = 41;
int lightPin3 = 42;
int lightPin4 = 43;
int lightPin5 = 44;

int wristLeftPin = 45;
int wristPin = 46;
int togglePin = 47;

int indexPos;
int middlePos;
int ringPos;
int littlePos;
int thumbPos;
```

```
void protoSetUp()
{
  startMarker = "$";
  stopMarker = "@";
  middleMarker = "*";
  servoString.reserve(4);
  positionString.reserve(64);
  protoResetAll();
}

void protoResetAll()
{
  servoString = "";
  positionString = "";
  protoReset();
}

void protoReset()
{
  startMarkerStatus = 0;
  stopMarkerStatus = 0;
  middleMarkerStatus = 0;
  servoLength = 0;
  positionLength = 0;
  packetAvailable = false;
}

void serialEvent()
{
 protoRead();
 if(packetAvailable)
 {
  ParseCommand();
  protoResetAll();
 }
}

void protoRead()
{
  while(Serial1.available()>0 && !packetAvailable)
  {
    int bufferChar = Serial1.read();
    if(startMarkerStatus < startMarker.length())
    {
     if(startMarker[startMarkerStatus] == bufferChar)
     {
       startMarkerStatus++;
     }
     else
     {
       protoResetAll();
     }
    }
    else
    {
       if(servoLength <= 0)
       {
```

```
        servoLength = bufferChar;
      }
    else
    {
      if(servoLength > servoString.length())
      {
        servoString += (char)bufferChar;
      }
      else
      {
    if(middleMarkerStatus < middleMarker.length())
    {
      if(middleMarker[middleMarkerStatus] == bufferChar)
      {
        middleMarkerStatus++;
      }
      else
      {
        protoResetAll();
      }
    }
    else
    {
      if(positionLength <= 0)
      {
        positionLength = bufferChar;
      }
      else
      {
        if(positionLength > positionString.length())
        {
          positionString += (char)bufferChar;
        }
        else
        {
          if(stopMarkerStatus < stopMarker.length())
          {
            if(stopMarker[stopMarkerStatus] == bufferChar)
            {
              stopMarkerStatus++;
              if(stopMarkerStatus == stopMarker.length())
              {
                protoReset();
                packetAvailable = true;
              }
            }
            else
            {
              protoResetAll();
            }
          }
        }
      }
    }
  }
 }
}
```

```
 }
}

void ParseCommand()
{
  if(positionString == "" || servoString == "")
  {
    int indexExtra = index.read();
    int middleExtra = middle.read();
    int ringExtra = ring.read();
    int littleExtra = little.read();
    int thumbExtra = thumb.read();

    index.write(indexExtra);
    middle.write(middleExtra);
    ring.write(ringExtra);
    little.write(littleExtra);
    thumb.write(thumbExtra);
  }
  else
  {
    int servoNum = servoString.toInt();
    int space = positionString.toInt();
    if(servoNum == 1)
    {
      indexPos = space;
    }
    else if(servoNum == 2)
    {
      middlePos = space;
    }
    else if(servoNum == 3)
    {
      ringPos = space;
    }
    else if(servoNum == 4)
    {
      littlePos = space;
    }
    else if(servoNum == 5)
    {
      thumbPos = space;
    }
    else if(servoNum == 6)
    {
      if(space)
      {
        wristLeft.write(160);
      }
      else
      {
        wristLeft.write(0);
      }
    }
    else
    {
      protoResetAll();
```

```
    }
  }
}

void setup()
{
  protoSetUp();
  Serial1.begin(9600);
  delay(2000);

  index.attach(indexPin);
  middle.attach(middlePin);
  ring.attach(ringPin);
  little.attach(littlePin);
  thumb.attach(thumbPin);
  finger1.attach(servoPin1);
  finger2.attach(servoPin2);
  finger3.attach(servoPin3);
  finger4.attach(servoPin4);
  finger5.attach(servoPin5);
  wrist.attach(wristPin);
  wristLeft.attach(wristLeftPin);

  pinMode(indexPin, OUTPUT);
  pinMode(middlePin, OUTPUT);
  pinMode(ringPin, OUTPUT);
  pinMode(littlePin, OUTPUT);
  pinMode(thumbPin, OUTPUT);
  pinMode(servoPin1, OUTPUT);
  pinMode(servoPin2, OUTPUT);
  pinMode(servoPin3, OUTPUT);
  pinMode(servoPin4, OUTPUT);
  pinMode(servoPin5, OUTPUT);

  pinMode(lightPin1, OUTPUT);
  pinMode(lightPin2, OUTPUT);
  pinMode(lightPin3, OUTPUT);
  pinMode(lightPin4, OUTPUT);
  pinMode(lightPin5, OUTPUT);

  pinMode(flexPin1, INPUT);
  pinMode(flexPin2, INPUT);
  pinMode(flexPin3, INPUT);
  pinMode(flexPin4, INPUT);
  pinMode(flexPin5, INPUT);
}

void loop()
{
  serialEvent();
  ParseCommand();

  index.write(indexPos);
  middle.write(middlePos);
  ring.write(ringPos);
  little.write(littlePos);
  thumb.write(thumbPos);
```

```
  digitalWrite(lightPin1, HIGH);
  digitalWrite(lightPin2, HIGH);
  digitalWrite(lightPin3, HIGH);
  digitalWrite(lightPin4, HIGH);
  digitalWrite(lightPin5, HIGH);

  int flex1 = analogRead(flexPin1);
  int flex2 = analogRead(flexPin2);
  int flex3 = analogRead(flexPin3);
  int flex4 = analogRead(flexPin4);
  int flex5 = analogRead(flexPin5);
  boolean wristPos = digitalRead(togglePin);

  int pos1 = map(flex1, 610, 15, 0, 180);
  pos1 = constrain(pos1, 0, 180);
  int pos2 = map(flex2, 920, 290, 0, 180);
  pos2 = constrain(pos2, 0, 180);
  int pos3 = map(flex3, 930, 420, 0, 180);
  pos3 = constrain(pos3, 0, 180);
  int pos4 = map(flex4, 910, 435, 0, 180);
  pos4 = constrain(pos4, 0, 180);
  int pos5 = map(flex5, 780, 340, 0, 180);
  pos5 = constrain(pos5, 0, 180);

  finger1.write(pos1);
  finger2.write(pos2);
  finger3.write(pos3);
  finger4.write(pos4);
  finger5.write(pos5);

  if(wristPos)
  {
    wrist.write(160);
  }
  else
  {
    wrist.write(0);
  }
}
```