§4 Заключительный этап: командная часть

Максимальное количество баллов за финальную часть - 100 баллов. На этом этапе участникам необходимо было создать систему управления бионическим протезом кисти человека с обратной связью (на основе сигналов электромиограммы и электроэнцефалограммы) и сравнить ее с системой управления на основе датчиков изгиба. Бионические протезы бывают обычными и высокофункциональными. Высокофункциональными считаются протезы, умеющие делать различные хваты, тогда как обычные бионические протезы делают одно основное движение — хват в щепоть. Чем больше будет реализовано жестов, тем больше участников могли набрать баллов.

Выполнение заданий из частей 1, 2, 3 и 4 предполагается на рабочем месте команды. Для сдачи заданий из перечисленных частей необходимо позвать Организаторов. Участники из других команд имеют право присутствовать при сдаче заданий других команд. Если команда неправильно выполнила задание, то в следующий раз команда может сдавать задание не менее, чем через 10 минут.

Выполнение заданий из части 5 предполагается на специальном стенде. Время, когда можно будет продемонстрировать выполнение заданий, лимитировано (5 минут), а очередность команд будет определяться жеребьевкой. Время для демонстрации задания 5 будет озвучено Организаторами. Учитывается результат лучшей попытки.

Задания из одной части надо выполнять и сдавать в той последовательности, в которой они идут. Части 1, 2, 3 допускается выполнять параллельно.

В случае возникновения сложностей, у вас есть возможность задать вопрос Организатору (по три вопроса на команду в день). Вопрос должен быть сформулирован так, чтобы на него можно было ответить “Да” или “Нет”. Ответ на вопрос объявляется всей аудитории.

4.1 Часть 1 - “Перчатка”

Ознакомьтесь с общей схемой установки, представленной в Приложении 1.

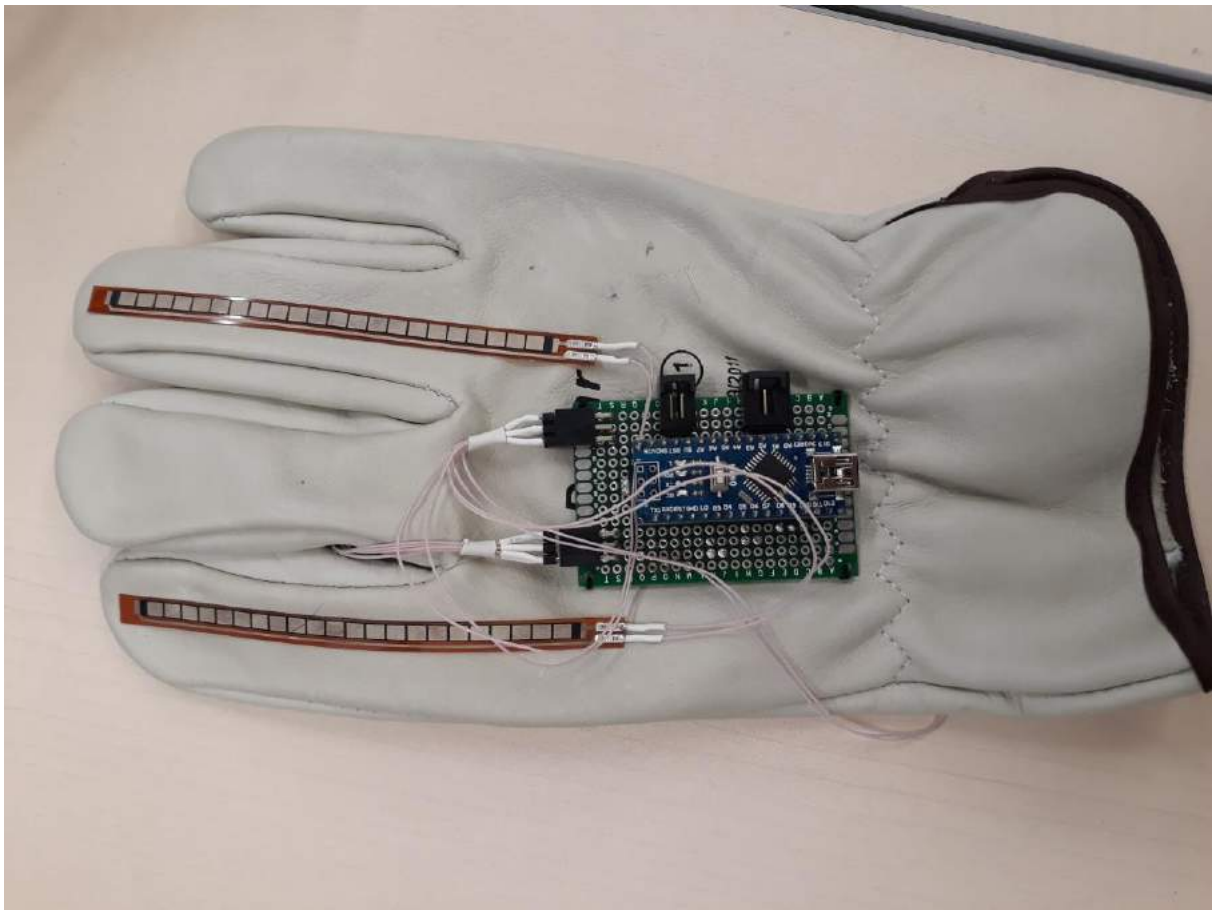
Задание 1.1 (*3 балла*)

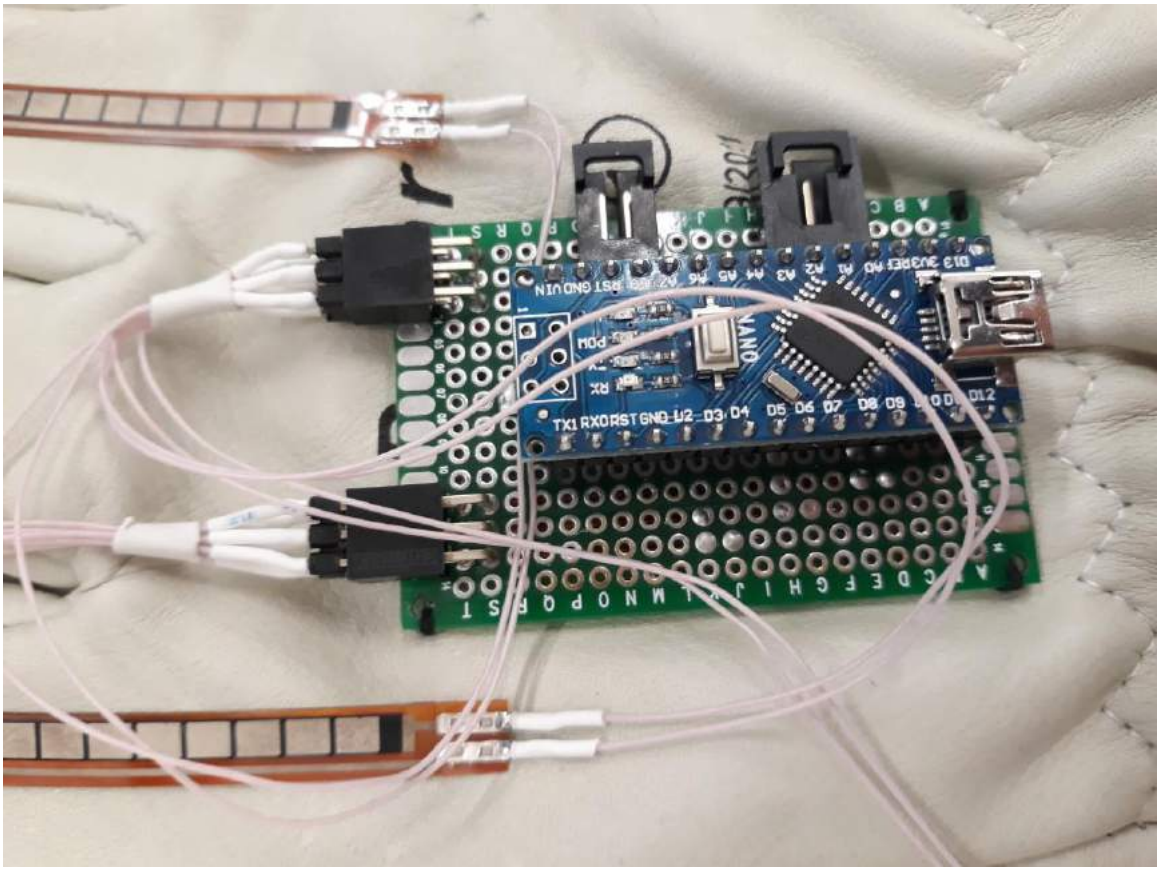
Приклейте (с помощью двухстороннего скотча) 3 вибромотора к кончикам пальцев перчатки - большой, указательный и безымянный пальцы.
Пришейте/закрепите плату (зеленую) к тыльной стороне перчатки так, чтобы длины проводов от вибромоторов и датчиков изгиба хватило до платы.
Пришейте/закрепите 3 датчика изгиба на пальцы перчатки (на те пальцы, куда прикреплены вибромоторы).
Подключите датчики изгиба и вибромоторы к плате согласно схеме, представленной в Приложении 2.
Вставьте Arduino Nano в плату так, чтобы USB-разъем оказался дальше от кончиков пальцев перчатки.

Комментарий: при необходимости вы можете пришить соединительные провода к перчатке. Кабель для подключения Arduino Nano к компьютеру выдается после правильно выполненного задания 1.1.

Решение задачи 1.1

В данном задании необходимо было продумать расположение датчиков изгиба, вибромоторов и платы, способ их крепления и подключить датчики изгиба и вибромоторы, согласно схеме из Приложения 1. Элементы необходимо было расположить так, чтобы было возможно выполнение последующих заданий. Пример выполненной задачи 1.1 представлен на фото ниже.





На фото: возможное решение задачи 1.1

Задание 1.2 (*3 балла*)

Напишите программу для Arduino Nano, которая будет оцифровывать значения с датчиков изгиба и отсылать их на компьютер. Данные должны отправляться согласно Протоколу (Приложение 3).

Комментарий: Блок-схема работы скетчей для Arduino представлена в Приложении 4.

Решение задачи 1.2

Возможное решение (программа для Arduino Nano) представлено ниже.

```
 #define FOREFINGER_FLEX A5
#define MIDDLEFINGER_FLEX A6
#define LITTLEFINGER_FLEX A7
const byte finger_sensors[3] = {FOREFINGER_FLEX,
                                MIDDLEFINGER_FLEX,
                                LITTLEFINGER_FLEX};

#define MAP_MIN 750
#define MAP_MAX 300

void setup() {
  Serial.begin(115200);
}
void loop() {
  while(1) {
    processSensors();
  }
}
void processSensors() {
   byte i;
   int tmp = 0;
   static uint8_t out_buf[6];

   for(i = 0; i < 6; i++) {
     if(i%2 == 0) {
       out_buf[i] = i/2;
     }
     else {
       tmp = analogRead(finger_sensors[i/2]);
       delay(1);
       out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
     }
    }

    Serial.write(out_buf, 6);
 }
```

Задание 1.3 (*2 балла*)

Напишите программу на языке Python, которая будет принимать данные от Arduino Nano и выводить их на экран. Каждому каналу соответствует свой столбец значений при выводе на экран.

Решение задачи 1.3

В данной задаче для передачи данных от Arduino используется программа из задания 1.2. Программа на языке Python, принимающая данные от Arduino Nano и выводящая их на экран, представлена ниже.

```
import serial
```

```
import time
import struct
glove = serial.Serial("COM4", 115200, timeout = None) #port number, baude rate
while (1):
    glove.flushInput()
    time.sleep(0.1)
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    if len(list_dat) >= 6:
        print(str(list_dat[1])+" "+str(list_dat[3])+" "+str(list_dat[5]))
raw.close()
```

Задание 1.4 (*2 балла*)

Напишите программу на языке Python, которая будет включать и выключать вибромоторы.

Решение задачи 1.4

Для выполнения данного задания помимо написания программы на языке Python, необходимо модифицировать программу для Arduino Nano:

```
#define FOREFINGER_VIBRO 9
#define MIDDLEFINGER_VIBRO 6
#define LITTLEFINGER_VIBRO 3
const byte finger_vibros[3] = {FOREFINGER_VIBRO,
                         MIDDLEFINGER_VIBRO,
                         LITTLEFINGER_VIBRO};
#define FOREFINGER_FLEX A5
#define MIDDLEFINGER_FLEX A6
#define LITTLEFINGER_FLEX A7
const byte finger_sensors[3] = {FOREFINGER_FLEX,
                                MIDDLEFINGER_FLEX,
                                LITTLEFINGER_FLEX};
#define MAP_MIN 750
#define MAP_MAX 300
void setup() {
  byte i;
  for(i = 0; i < 5; i++) {
    pinMode(finger_vibros[i], OUTPUT);
    analogWrite(finger_vibros[i], 0);
  }

  Serial.begin(115200);
}
void loop() {
  while(1) {
    processInput();
    processSensors();
  }
}
void processInput() {
  static byte inp_buf[2];
  if(Serial.available() > 1) {
    Serial.readBytes((char *)inp_buf, 2);
    if(inp_buf[0] < 5)
      analogWrite(finger_vibros[inp_buf[0]], inp_buf[1]);
     else
       Serial.write('\x05\xff');
  }
}
```

```
void processSensors() {
   byte i;
   int tmp = 0;
   static uint8_t out_buf[10];
   for(i = 0; i < 6; i++) {
     if(i%2 == 0) {
       out_buf[i] = i/2;
     }
     else {
       tmp = analogRead(finger_sensors[i/2]);
       delay(1);
       out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
     }
    }
    Serial.write(out_buf, 6);
 }
```

Программа на языке Python для данного задания:

```
import serial
glove = serial.Serial("COM4", 115200, timeout = None) #port number, baude rate
while(1):
    pressure_num = int(input())
    pressure_state = int(input())
    arr = bytearray([pressure_num, pressure_state])
    glove.write(arr)
```

4.2 Часть 2 - “Макет протеза руки человека”

Ознакомьтесь с блок-схемой электроники макета протеза руки человека, представленной в Приложении 5.

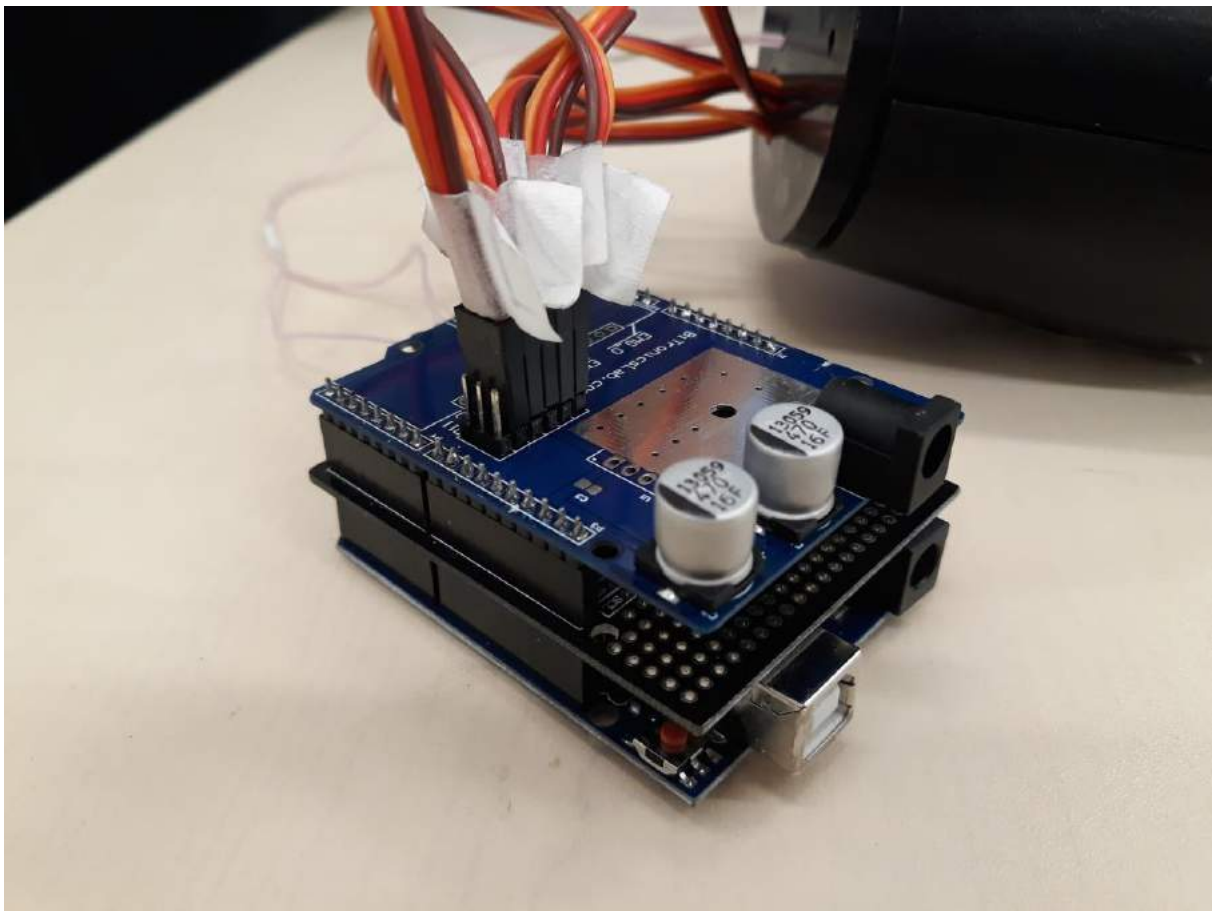
Задание 2.1 (2 балла)

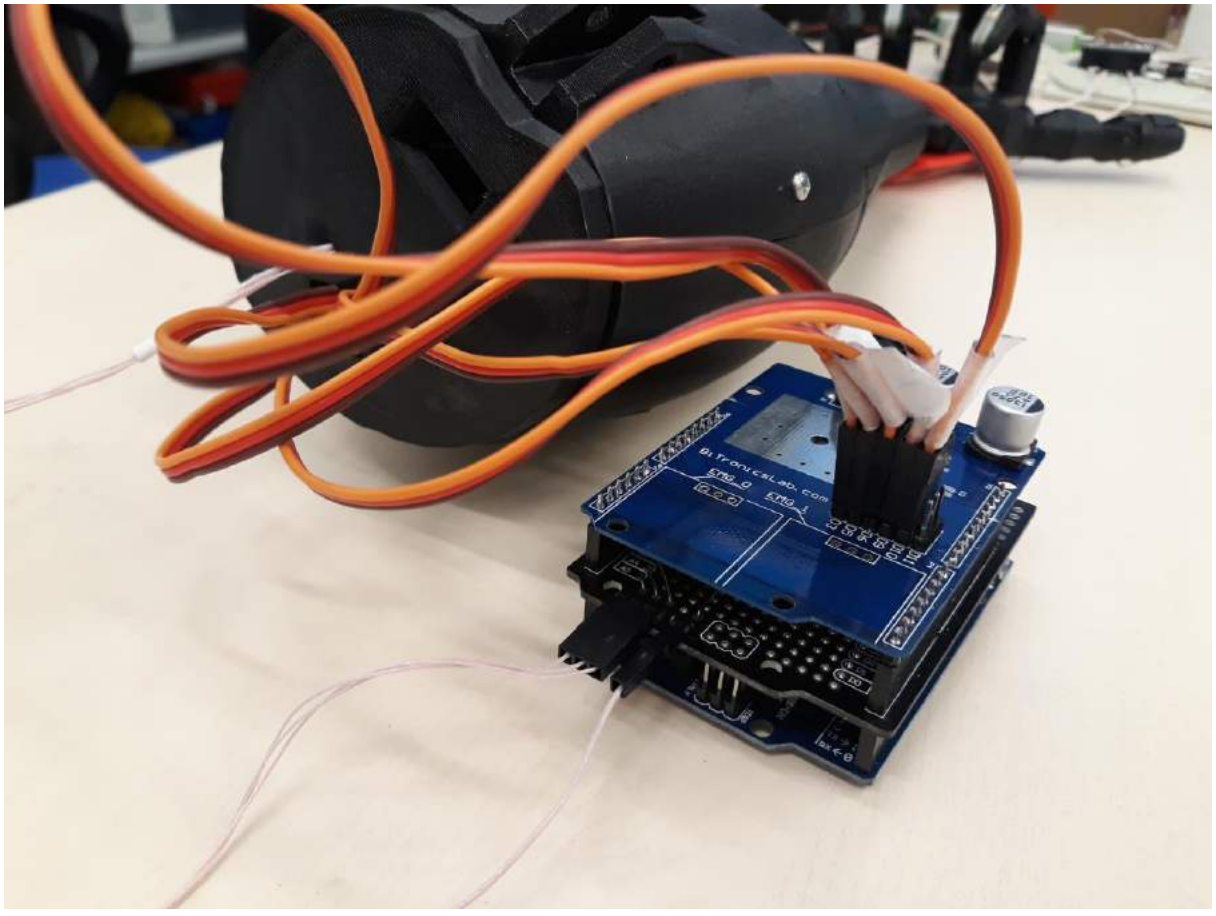
Подсоедините 2 Шилда к Arduino Uno. Подключите провода от датчиков давления (бледно-розовые) к шилду (черный). Подключите провода от сервоприводов к шилду №2 (цветные провода). Провода с отметкой "1" соответствуют большому пальцу механической руки, провода с отметкой "5" соответствует мизинцу механической руки. Коричневый провод от сервопривода должен быть расположен ближе к разъему питания шилда.

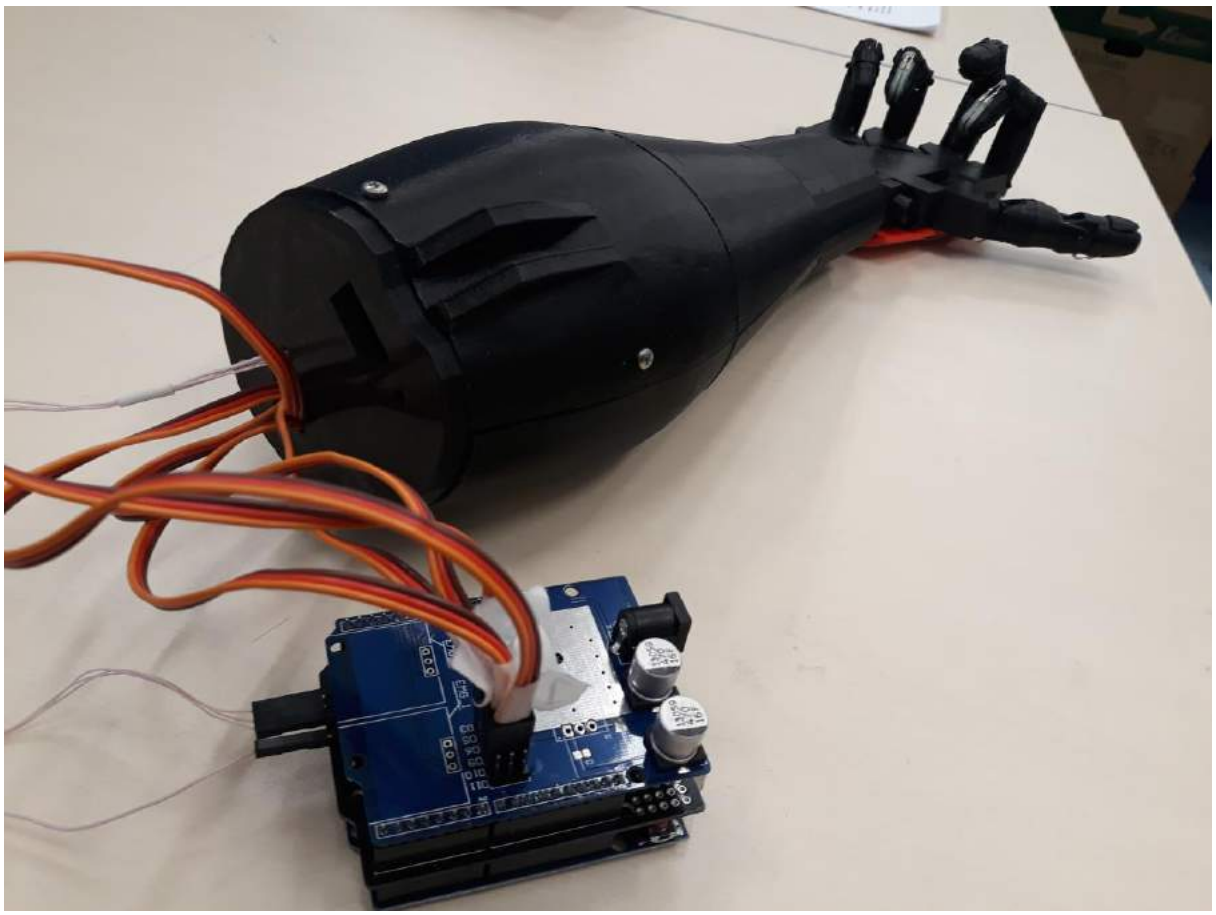
Комментарий: блок питания и провод miniUSB будет выдан только после правильного выполнения задания 2.1.

Решение задачи 2.1

В задаче 2.1 участникам необходимо было понять как устроена электроника макета протеза руки человека, подсоединить датчики давления и сервоприводы к Arduino Uno, используя Шилд. Пример выполненной задачи 2.1 представлен на фото ниже:







На фото: пример сборки/подключения задания 2.1

Задание 2.2 (*3 балла*)

Калибровка. Выясните рабочие диапазоны сервоприводов руки (минимальный и максимальный углы поворота каждого сервопривода, обеспечивающие анатомически верное воспроизведение движения соответствующего пальца макета протеза) сервоприводов.

Решение задачи 2.2

В данной задаче участникам необходимо было подобрать рабочие диапазоны сервоприводов руки, написав скетч и визуально отслеживая работу сервоприводов руки. Необходимо было подобрать такие значения параметров (см. скетч ниже), при которых минимальное и максимальное значение угла поворота каждого из сервопривода соответствовало анатомически верному воспроизведению движения соответствующего пальца макета руки. Вариант калибровочного скетча представлен ниже.

```
#include <Servo.h>
#define THUMB_PIN 10                //pins
#define FOREFINGER_PIN 9
#define MIDDLEFINGER_PIN 6
#define RINGFINGER_PIN 5
#define LITTLEFINGER_PIN 3
const byte fingers[5] = {THUMB_PIN,
                         FOREFINGER_PIN,
                         MIDDLEFINGER_PIN,
                         RINGFINGER_PIN,
                         LITTLEFINGER_PIN};
//подбираемые значения
#define THUMB_CLENCHED 10
#define FOREFINGER_CLENCHED  240
#define MIDDLEFINGER_CLENCHED  190
```

```
#define RINGFINGER_CLENCHED  160
#define LITTLEFINGER_CLENCHED  10
#define THUMB_RELAXED 150
#define FOREFINGER_RELAXED 10
#define MIDDLEFINGER_RELAXED 10
#define RINGFINGER_RELAXED 5
#define LITTLEFINGER_RELAXED 170
//конец подбираемых значений
const byte relaxed[5] = {THUMB_RELAXED,
                         FOREFINGER_RELAXED,
                         MIDDLEFINGER_RELAXED,
                         RINGFINGER_RELAXED,
                         LITTLEFINGER_RELAXED};
const byte clenched[5] = {THUMB_CLENCHED,
                         FOREFINGER_CLENCHED,
                         MIDDLEFINGER_CLENCHED,
                         RINGFINGER_CLENCHED,
                         LITTLEFINGER_CLENCHED};

#define DELAY_A 10
Servo thumb;
Servo foreFinger;
Servo middleFinger;
Servo ringFinger;
Servo littleFinger;
Servo servos[5] = {thumb, foreFinger, middleFinger, ringFinger, littleFinger};
void setup() {
  byte i;
  for(i = 0; i < 5; i++) {
    (servos[i]).attach(fingers[i]);
    delay(DELAY_A);

  }

}
void loop() {
  byte i;
  for(i = 0; i < 5; i++) {
    servos[i].write(clenched[i]);
    delay(1000);
    servos[i].write(relaxed[i]);
    delay(1000);
  }
}
```

Задание 2.3 (*2,5 балла*)

Напишите программу для Arduino Uno, которая будет осуществлять сжатие кисти макета руки, пока не будет достигнут максимум сжатия кисти макета руки или пока пальцы макета руки не упрутся в препятствие (при включении Arduino Uno выполняется инициализация: ладонь раскрывается, а затем из данного положения сжимается до тех пор, пока полностью не сомкнется (определяется программно согласно рабочим диапазонам из п.2.2), либо пока не упрется в препятствие (по показанию датчиков давления). В качестве препятствия может быть, например, сжимаемый предмет или подставленный палец человека.

Решение задачи 2.3

Пример скетча для Arduino Uno, реализующий условия задания 2.3 представлен ниже:

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

```
#define THUMB_PIN 10                //pins
#define FOREFINGER_PIN 9
#define MIDDLEFINGER_PIN 6
#define RINGFINGER_PIN 5
#define LITTLEFINGER_PIN 3
const byte fingers[5] = {
  THUMB_PIN,
  FOREFINGER_PIN,
  MIDDLEFINGER_PIN,
  RINGFINGER_PIN,
  LITTLEFINGER_PIN};
#define THUMB_CLENCHED 10           //servo positions
#define FOREFINGER_CLENCHED  240
#define MIDDLEFINGER_CLENCHED  190
#define RINGFINGER_CLENCHED  160
#define LITTLEFINGER_CLENCHED  10
#define THUMB_RELAXED 150
#define FOREFINGER_RELAXED 10
#define MIDDLEFINGER_RELAXED 10
#define RINGFINGER_RELAXED 5
#define LITTLEFINGER_RELAXED 170
const byte clenched[5] = {
  THUMB_CLENCHED,
  FOREFINGER_CLENCHED,
  MIDDLEFINGER_CLENCHED,
  RINGFINGER_CLENCHED,
  LITTLEFINGER_CLENCHED};
const byte relaxed[5] = {
  THUMB_RELAXED,
  FOREFINGER_RELAXED,
  MIDDLEFINGER_RELAXED,
  RINGFINGER_RELAXED,
  LITTLEFINGER_RELAXED};
#define DELAY_A 10
Servo thumb;
Servo foreFinger;
Servo middleFinger;
Servo ringFinger;
Servo littleFinger;
Servo servos[5] = {thumb, foreFinger, middleFinger, ringFinger, littleFinger};
#define FOREFINGER_PRES A3
#define MIDDLEFINGER_PRES A4
#define LITTLEFINGER_PRES A5
const byte sensors[3] = {
  FOREFINGER_PRES,
  MIDDLEFINGER_PRES,
  LITTLEFINGER_PRES};
#define MAP_MIN 0
#define MAP_MAX 700
#define THRSH 500
void setup() {
  byte i;
  for(i = 0; i < 5; i++) {
    (servos[i]).attach(fingers[i]);
    delay(DELAY_A);
  }
}
void loop() {
  byte i, j, k = 0;
  for(i = 0; i < 5; i++) {
    (servos[i]).write(relaxed[i]);
    delay(15);
  }
  delay(1000);
```

```
  for(i = 0; i < 180; i++) {
    if(processSensors())
      break;
    for(j = 0; j < 3; j++) {
      if(((k <= relaxed[j]) && (k >= clenched[j])) || ((k >= relaxed[j]) && (k
<= clenched[j])))
        (servos[j]).write(k);
    }

    k++;
    delay(15);
  }
}
bool processSensors() {
  byte i;
  int tmp = 0;
  for(i = 0; i < 3; i++) {
    tmp = analogRead(sensors[i]);
    delay(1);
    tmp = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
    if(tmp > THRSH)
      return true;
  }
  return false;
}
```

Задание 2.4 (*0,5 балла*)

Напишите программу для Arduino Uno, которая будет оцифровывать значения с датчиков давления и отправлять на компьютер согласно Протоколу (Приложение 3).

Решение задачи 2.4

Пример программы для Arduino Uno, которая оцифровывается значения с датчиков давления и отправляет на компьютер согласно Протоколу из Приложения 3:

```
#define FOREFINGER_PRES A3
#define MIDDLEFINGER_PRES A4
#define LITTLEFINGER_PRES A5
const byte sensors[3] = {FOREFINGER_PRES,
                         MIDDLEFINGER_PRES,
                         LITTLEFINGER_PRES};
#define MAP_MIN 0
#define MAP_MAX 700
void setup() {
  Serial.begin(115200);
}
void loop() {
  while(1) {
    processSensors();
  }
}
void processSensors() {
   byte i;
   int tmp = 0;
   static uint8_t out_buf[6];

   for(i = 0; i < 6; i++) {
     if(i%2 == 0) {
       out_buf[i] = i/2;
     }
     else {
       tmp = analogRead(sensors[i/2]);
       delay(1);
       out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
```

```
     }
    }
    Serial.write(out_buf, 6);
 }
```

Задание 2.5 (*2 балла*)

Напишите программу на языке Python, которая будет принимать данные с датчиков давления от Arduino Uno и осуществлять управление сервоприводами руки (управление сервоприводами осуществляется аналогично заданию 2.3).

Решение задачи 2.5

В данном решении необходимо было доработать скетч для Arduino Uno из предыдущих заданий, а также написать программу на языке Python.

Программа для Arduino Uno:

```
#include <Servo.h>
#define THUMB_PIN 10                //pins
#define FOREFINGER_PIN 9
#define MIDDLEFINGER_PIN 6
#define RINGFINGER_PIN 5
#define LITTLEFINGER_PIN 3
const byte fingers[5] = {THUMB_PIN,
                         FOREFINGER_PIN,
                         MIDDLEFINGER_PIN,
                         RINGFINGER_PIN,
                         LITTLEFINGER_PIN};
#define THUMB_CLENCHED 10           //servo positions
#define FOREFINGER_CLENCHED  240
#define MIDDLEFINGER_CLENCHED  190
#define RINGFINGER_CLENCHED  160
#define LITTLEFINGER_CLENCHED  10
#define THUMB_RELAXED 150
#define FOREFINGER_RELAXED 10
#define MIDDLEFINGER_RELAXED 10
#define RINGFINGER_RELAXED 5
#define LITTLEFINGER_RELAXED 170
#define DELAY_A 10
Servo thumb;
Servo foreFinger;
Servo middleFinger;
Servo ringFinger;
Servo littleFinger;
Servo servos[5] = {thumb, foreFinger, middleFinger, ringFinger, littleFinger};
#define FOREFINGER_PRES A3
#define MIDDLEFINGER_PRES A4
#define LITTLEFINGER_PRES A5
const byte sensors[3] = {FOREFINGER_PRES,
                         MIDDLEFINGER_PRES,
                         LITTLEFINGER_PRES};
#define MAP_MIN 0
#define MAP_MAX 700
void setup() {
  byte i;
  for(i = 0; i < 5; i++) {
    (servos[i]).attach(fingers[i]);
    delay(DELAY_A);

  }
```

```
  Serial.begin(115200);
}
void loop() {
  while(1) {
    processInput();
    processSensors();
  }
}
void processInput() {
  static byte inp_buf[2];
  if(Serial.available() > 1) {
    Serial.readBytes((char *)inp_buf, 2);
    if(inp_buf[0] < 5)
      (servos[inp_buf[0]]).write(inp_buf[1]);
     else
       Serial.write('\x05\xff');
  }
}
void processSensors() {
   byte i;
   int tmp = 0;
   static uint8_t out_buf[6];
   for(i = 0; i < 6; i++) {
     if(i%2 == 0) {
       out_buf[i] = i/2;
     }
     else {
       tmp = analogRead(sensors[i/2]);
       delay(1);
       out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
     }
    }
    Serial.write(out_buf, 6);
 }
```

Программа на языке Python, реализующая условия данного задания.

```
import serial
import time
import struct
hand = serial.Serial("COM16", 115200, timeout = None)
first = True
state = 10
calm_1 = 0
calm_2 = 0
calm_3 = 0
while (1):
    # hand.flushInput()
    time.sleep(0.06)
    in_queue = hand.in_waiting
    dat = hand.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    print(list_dat)
    if first and (len(list_dat) > 5):
        calm_1 = list_dat[1]
        calm_2 = list_dat[3]
        calm_3 = list_dat[5]
        first = False
    if (state<140)and(len(list_dat) > 5)and(calm_1 + 35 > list_dat[1])and(calm_2
+ 35 > list_dat[3])and(calm_3 + 35 > list_dat[5]):
```

```
for i in range(1, 4):
    arr = bytearray([i, state])
    hand.write(arr)
arr = bytearray([0, 160-state])
hand.write(arr)
arr = bytearray([4, 160 - state])
hand.write(arr)
state += 1
```

4.3 Часть 3 - Модули электромиограммы (ЭМГ) и электроэнцефалограммы (ЭЭГ)

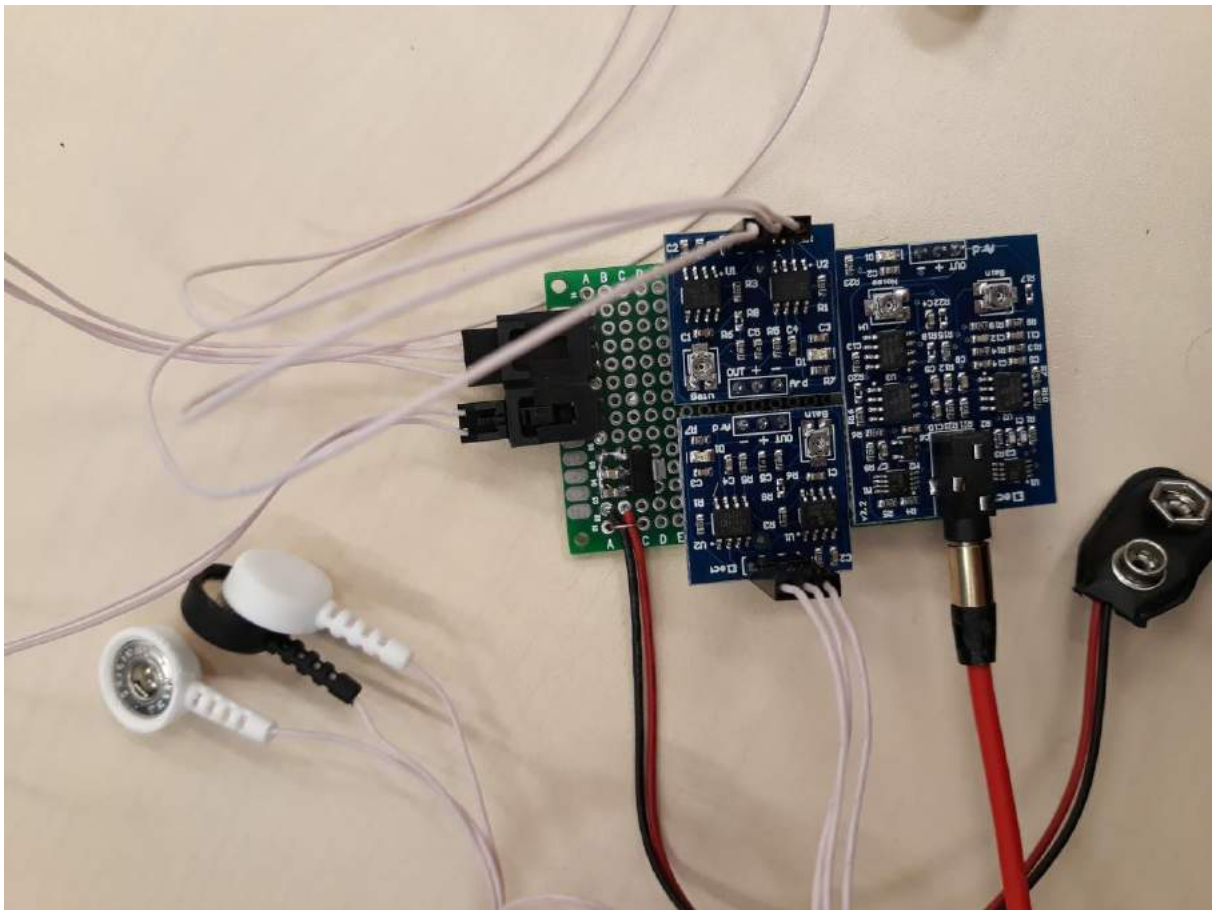
Задание 3.1 (*1 балл*)

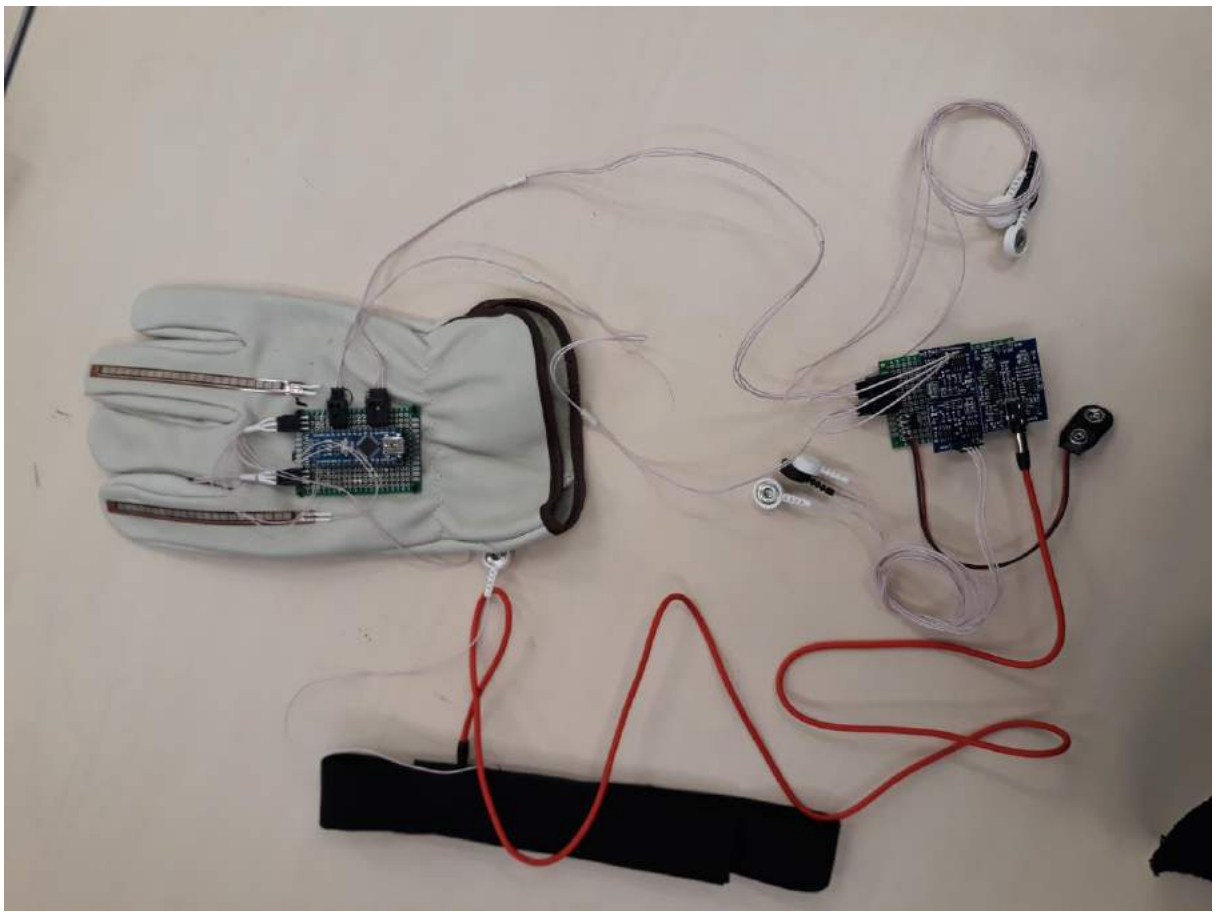
Подключите 2 модуля Электромиограммы (EMG/ECG) и модуль электроэнцефалограммы (EEG) к плате (см. фото в Приложении 2). Затем подключите эту плату к плате на перчатке, согласно схеме из Приложения 2.

Комментарий. Элемент питания будет выдан команде после правильно выполненного задания 3.1.

Решение задачи 3.1

В этом задании участникам необходимо было правильно подсоединить модули электроэнцефалограммы и электромиограммы к плате согласно схеме, представленной в приложении 2. На фото ниже - пример правильного подключения.





На фото: пример выполнения задания 3.1

Задание 3.2 (*1 балл*)

Допишите программу для Arduino Nano из п.1.2: программа должна также оцифровывать значения с модулей ЭМГ (порты A1, A2) и модуля ЭЭГ (порт A0) и передавать их на компьютер, согласно Протоколу (Приложение 3).

Решение задачи 3.2

Модернизированная программа из п.1.2 для Arduino Nano, оцифровывающая значения с модулей ЭМГ и модуля ЭЭГ, согласно условию задания:

```
#define FOREFINGER_FLEX A5
#define MIDDLEFINGER_FLEX A6
#define LITTLEFINGER_FLEX A7
const byte finger_sensors[3] = {
  FOREFINGER_FLEX,
  MIDDLEFINGER_FLEX,
  LITTLEFINGER_FLEX};
#define MAP_MIN 750
#define MAP_MAX 300
void setup() {
  Serial.begin(115200);
}
void loop() {
  while(1) {
    processSensors();
  }
```

```
}
void processSensors() {
  byte i;
  int tmp = 0;
  static uint8_t out_buf[12];
  for(i = 0; i < 6; i++) {
    if(i%2 == 0) {
      out_buf[i] = i/2;
    }
    else {
      tmp = analogRead(finger_sensors[i/2]);
      delay(1);
      out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
    }
  }
  for(i = 6; i < 12; i++) {
    if(i%2 == 0) {
      out_buf[i] = i/2;
    }
    else {
      switch(i) {
      case 7:
        tmp = analogRead(A1);
        delay(1);
        out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
        break;
      case 9:
        tmp = analogRead(A2);
        delay(1);
        out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
        break;
      case 11:
        tmp = analogRead(A0);
        delay(1);
        out_buf[i] = map(tmp, MAP_MIN, MAP_MAX, 0, 255);
        break;
      }
    }
    Serial.write(out_buf, 12);
  }
}
```

Задание 3.3 (*0,5 балла*)

Напишите программу на языке Python, которая будет принимать данные от Arduino Nano и выводить их на экран. Каждому каналу соответствует свой столбец значений при выводе на экран.

Решение задачи 3.3

Для Arduino Nano в данной задаче используется скетч из задания 3.2. Пример программы на языке Python, реализующая условия задачи, представлен ниже.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
```

```
while(1):
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    if len(list_dat) > 5:
        print(list_dat[1]+' '+list_dat[3]+' '+list_dat[5])
```

Задание 3.4 (*2 балла*)

Визуализация ЭМГ и ЭЭГ. Напишите программу на языке Python, визуализирующую “сырой” сигнал одновременно с двух модулей ЭМГ и модуля ЭЭГ. По оси абсцисс отложите время, по оси ординат - значение сигнала. При считывании сигналов (в т.ч. одновременном) сигналы не должны зашкаливать, а также сигналы не должны быть зашумлены.

Решение задачи 3.4

Для Arduino Nano в данной задаче используется скетч из задания 3.2. Пример программы на языке Python, реализующая условия задачи, представлен ниже.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
hand = serial.Serial("COM16", 115200, timeout = None)
max_points = 100   #number of points in frame
# create a figure with two subplots
fig, (ax1, ax2, ax3) = plt.subplots(3,1)
# intialize two line objects (one in each axes)
line1, = ax1.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line2, = ax2.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line3, = ax3.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line = [line1, line2, line3]
# the same axes initalizations as before (just now we do it for both of them)
for ax in [ax1, ax2, ax3]:
    ax.set_ylim(0, 260)
    ax.set_xlim(0, 100)
    ax.grid()
ax1.set_title("A0")
ax2.set_title("A1")
ax3.set_title("A2")
def init():
    return line
i = 0
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-25)/(200-25)*(150-10)))
        if res < 10:
            return 10
```

```
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 18) / (200 - 18)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 18) / (200 - 18)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state) / (160)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state) / (160)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
t = 0
def animate(data):
    global t
    glove.flushInput()
    hand.flushInput()
    time.sleep(0.03)  #could be changed
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    print(list_dat)
    if len(list_dat) > 5:
        corrected = mapping(0, list_dat[1])
        arr = bytearray([0, corrected])
        hand.write(arr)
        print("1")
        corrected = mapping(1, list_dat[3])
        arr = bytearray([1, corrected])
        hand.write(arr)
        print("2")
        corrected = mapping(2, list_dat[3])
        arr = bytearray([2, corrected])
        hand.write(arr)
        print("3")
        corrected = mapping(3, list_dat[5])
        arr = bytearray([3, corrected])
        hand.write(arr)
        corrected = mapping(4, list_dat[5])
        arr = bytearray([4, corrected])
        hand.write(arr)
    if len(list_dat) > 11:
```

```
        for i in range(0, 3):
            old_y = line[i].get_ydata()
            new_y = np.r_[old_y[1:], list_dat[(3+i)*2+1]]  # grab current data
            line[i].set_ydata(new_y)  # set the new ydata
    print("FPS "+str(int((1/(time.clock()-t)))))
    t = time.clock()
    return line
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=100,
interval=0,
                               blit=True)  # the fastest evaluation reached with
interval=0
plt.show()
```

Задание 3.5 (*2,5 балла*)

Напишите программу на языке Python, визуализирующую зависимость амлитуды сигнала ЭМГ от времени.
Амплитуду сигнала можно вычислять так: найти в последовательности из N считанных значений сигнала максимальное и минимальное значение, найти их разность - данную величину здесь и назовем "амплитудой сигнала". После этого повторяем данную процедуру для последующих N точек. N - определяется самостоятельно, исходя из того принципа, что время реакции системы должно быть сопоставимо со скоростью реакции человека.

Решение задачи 3.5

Для Arduino Nano в данной задаче используется скетч из задания 3.2. Пример программы на языке Python, реализующая условия задачи, представлен ниже.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import math
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
# hand = serial.Serial("COM7", 115200, timeout = None)
max_points = 100   #number of points in frame
# create a figure with two subplots
fig, (ax1, ax2, ax3) = plt.subplots(3,1)
# intialize three line objects (one in each axes)
line1, = ax1.plot(np.arange(max_points),
                  np.ones(max_points, dtype=np.float)*np.nan,
                  lw=2, animated=True)
line2, = ax2.plot(np.arange(max_points),
                  np.ones(max_points, dtype=np.float)*np.nan,
                  lw=2, animated=True)
line3, = ax3.plot(np.arange(max_points),
                  np.ones(max_points, dtype=np.float)*np.nan,
                  lw=2, animated=True)
line = [line1, line2, line3]
# the same axes initalizations as before (just now we do it for both of them)
ax1.set_ylim(0, 256)
ax1.set_xlim(0, 100)
ax1.grid()
ax2.set_ylim(0, 256)
ax2.set_xlim(0, 100)
ax2.grid()
ax3.set_ylim(0, 100000)
```

```
ax3.set_xlim(0, 100)
ax3.grid()
ax1.set_title("Amp_1")
ax2.set_title("Amp_2")
ax3.set_title("Alpha")
def init():
    return line
i = 0
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-30)/(145-30)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 53) / (224 - 53)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 66) / (223 - 66)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state - 48) / (214 - 48)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state - 34) / (204 - 34)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
n = 0
t = time.clock()
glove.flushInput()
new_y1 = [np.nan]*30
new_y2 = [np.nan]*30
new_y3 = [np.nan]*512
avg = [np.nan]*10
def animate(data):
    buff = 12 #number of bytes in series
    global t
    global n
    global new_y1
    global new_y3
    global new_y2
    global avg
    n += 1
    global i
    if n > 30:    #wait until initialization ends
        print("GLOVE: "+str(glove.in_waiting))
        delta_t = time.clock() - t
```

```
        print("Delta T " + str((delta_t)))
        t = time.clock()
        in_queue = glove.in_waiting
        dat = glove.read(in_queue)
        dat = struct.unpack('c'*in_queue, dat)
        list_dat = []
        for i in dat:
            list_dat.append(int.from_bytes(i, byteorder='big'))
        print(list_dat)
        emg1 = []
        i = 0
        while i+buff-1 < len(list_dat):
            emg1.append(list_dat[i+buff-1])
            i += buff
        eeg = []
        i = 0
        while i+buff-2-1 < len(list_dat):
            eeg.append(list_dat[i+buff-2-1])
            i += buff
        emg = []
        i = 0
        while i+buff-4-1 < len(list_dat):
            emg.append(list_dat[i+buff-4-1])
            i += buff
        new_y1 = np.r_[new_y1[len(emg1):], emg1]  # grab current data
        new_y2 = np.r_[new_y2[len(eeg):], eeg]  # grab current data
        areas = line[0].get_ydata()
        areas = np.r_[areas[-99:], max(new_y1)-min(new_y1)]
        line[0].set_ydata(areas)  # set the new ydata
        areas = line[1].get_ydata()
        areas = np.r_[areas[-99:], max(new_y2)-min(new_y2)]
        line[1].set_ydata(areas)
    return line
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,
interval=0,
                               blit=True)  # the fastest evaluation reached with
interval=0
plt.show()
```

Задание 3.6 (*3 балла*)

Напишите программу на языке Python, визуализирующую зависимость альфа-ритма сигнала ЭЭГ от времени.

Решение задачи 3.6

Для Arduino Nano в данной задаче используется скетч из задания 3.2. Пример программы на языке Python, реализующая условия задачи, представлен ниже.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import math
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
# hand = serial.Serial("COM7", 115200, timeout = None)
max_points = 100   #number of points in frame
# create a figure with two subplots
```

```
fig, (ax1) = plt.subplots(1,1)
# intialize three line objects (one in each axes)
line1, = ax1.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line = [line1]
# the same axes initalizations as before (just now we do it for both of them)
ax1.set_ylim(0, 256)
ax1.set_xlim(0, 100)
ax1.grid()
ax1.set_title("Alpha")
def init():
    return line
i = 0
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-30)/(145-30)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 53) / (224 - 53)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 66) / (223 - 66)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state - 48) / (214 - 48)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state - 34) / (204 - 34)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
n = 0
t = time.clock()
glove.flushInput()
new_y1 = [np.nan]*30
new_y2 = [np.nan]*30
new_y3 = [np.nan]*512
avg = [np.nan]*10
def animate(data):
    buff = 12 #number of bytes in series
    global t
    global n
    global new_y1
    global new_y3
    global new_y2
```

```
global avg
n += 1
global i
if n > 30:    #wait until initialization ends
    print("GLOVE: "+str(glove.in_waiting))
    delta_t = time.clock() - t
    print("Delta T " + str((delta_t)))
    t = time.clock()
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c'*in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    print(list_dat)
    emg1 = []
    i = 0
    while i+buff-1 < len(list_dat):
        emg1.append(list_dat[i+buff-1])
        i += buff
    print(emg1)
    eeg = []
    i = 0
    while i+buff-2-1 < len(list_dat):
        eeg.append(list_dat[i+buff-2-1])
        i += buff
    print(eeg)
    emg = []
    i = 0
    while i+buff-4-1 < len(list_dat):
        emg.append(list_dat[i+buff-4-1])
        i += buff
    print(emg)
    new_y1 = np.r_[new_y1[len(emg1):], emg1]  # grab current data
    new_y2 = np.r_[new_y2[len(eeg):], eeg]  # grab current data
    new_y3 = np.r_[new_y3[len(emg):], emg]  # grab current data
    # line[0].set_ydata(new_y1)  # set the new ydata
    print("before fourier")
    signal = np.array(new_y3, dtype=int)
    fourier = np.fft.fft(signal)
    print("FOURE")
    print(fourier)
    num = signal.size
    print("NUM")
    print(num)
    timestep = delta_t/(in_queue//buff)
    print(timestep)
    avg = np.r_[avg[1:], timestep]
    delta = sum(avg)/10
    print(delta)
    freq = np.fft.fftfreq(num, d=delta)
    fourier = fourier[:256]
    freq = freq[:256]
    amp = []
    for i in fourier:
        amp.append(pow(i.real*i.real+i.imag*i.imag, 1/2))
    print(freq)
    print(amp)
    interval = freq[1]-freq[0]
    area = 0
    for i in range(0, len(amp)):
        if (freq[i] >= 8)and(freq[i]<=14):
            area += amp[i]*interval
    areas = line[2].get_ydata()
```

```
        areas = np.r_[areas[-99:], area]
        line[0].set_ydata(areas)
    return line
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,
interval=0,
                               blit=True)  # the fastest evaluation reached with
interval=0
plt.show()
```

4.4 Часть 4 - Объединение систем

Задание 4.1 (*6 баллов*)

Напишите программу на языке Python, с помощью которой макет руки будет повторять движения перчатки, надетой на кисть руки человека. В связи с тем, что на перчатке только 3 датчика изгиба, а на макете кисти руки 3 датчика давления необходимо правильно осуществлять формирование управляющих команд для всех пяти пальцев руки.

Решение задачи 4.1

В блоке заданий №4 происходит объединение систем. Поэтому будем использовать уже ранее написанные программы: для Arduino Nano (перчатка) - программа из задания 1.2, для Arduino Uno (макет протеза руки) - программа из задания 2.5.

Пример программы на языке Python, реализующая условия задания, представлена ниже.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
hand = serial.Serial("COM16", 115200, timeout = None)
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-25)/(200-25)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 18) / (200 - 18)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 18) / (200 - 18)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
```

```
        res = int(abs((abs(state) / (160)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state) / (160)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
while 1:
    glove.flushInput()
    hand.flushInput()
    time.sleep(0.1)  #could be changed
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    if len(list_dat) > 5:
        corrected = mapping(0, list_dat[1])
        arr = bytearray([0, corrected])
        hand.write(arr)
        corrected = mapping(1, list_dat[3])
        arr = bytearray([1, corrected])
        hand.write(arr)
        corrected = mapping(2, list_dat[3])
        arr = bytearray([2, corrected])
        hand.write(arr)
        corrected = mapping(3, list_dat[5])
        arr = bytearray([3, corrected])
        hand.write(arr)
        corrected = mapping(4, list_dat[5])
        arr = bytearray([4, corrected])
        hand.write(arr)
```

Задание 4.2 (*6 баллов*)

Реализуйте обратную связь: вибромоторы должны включаться на перчатке в тот момент, когда механическая рука упирается в препятствие, причем интенсивность вибрации повышается с увеличением воздействия на датчики давления.

Комментарий: Не допускайте длительную работу вибромоторов на полную мощность (больше 20 с) во избежание перегрева вибромоторов!

Решение задачи 4.2

В данном задании для Arduino Nano используется программа из задания 1.4, для Arduino Uno - программа из задания 2.5.

Ниже представлено возможное решение данной задачи на языке Python.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
```

```
from matplotlib import animation
import numpy as np
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
hand = serial.Serial("COM16", 115200, timeout = None)
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-25)/(200-25)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 18) / (200 - 18)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 18) / (200 - 18)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state) / (160)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state) / (160)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
while 1:
    glove.flushInput()
    hand.flushInput()
    time.sleep(0.1)  #could be changed
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    if len(list_dat) > 5:
        corrected = mapping(0, list_dat[1])
        arr = bytearray([0, corrected])
        hand.write(arr)
        corrected = mapping(1, list_dat[3])
        arr = bytearray([1, corrected])
        hand.write(arr)
        corrected = mapping(2, list_dat[3])
        arr = bytearray([2, corrected])
        hand.write(arr)
        corrected = mapping(3, list_dat[5])
```

```
        arr = bytearray([3, corrected])
        hand.write(arr)
        corrected = mapping(4, list_dat[5])
        arr = bytearray([4, corrected])
        hand.write(arr)
    in_queue = hand.in_waiting
    dat = hand.read(in_queue)
    dat = struct.unpack('c' * in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    print(list_dat)
    if (len(list_dat) > 5):
        arr = bytearray([0, list_dat[1]])
        glove.write(arr)
        arr = bytearray([1, list_dat[3]])
        glove.write(arr)
        arr = bytearray([2, list_dat[5]])
        glove.write(arr)
```

Задание 4.3 (*8 баллов*)

Реализуйте систему управления с помощью модулей ЭМГ и ЭЭГ следующим образом: с помощью первого модуля ЭМГ механическая рука должна сжиматься, с помощью второго модуля ЭМГ механическая рука должна разжиматься, с помощью модуля ЭЭГ (альфа-ритм) должна осуществляться активация вибромоторов

Решение задачи 4.3

В данном задании для Arduino Nano используется программа из задания 3.2, для Arduino Uno - программа из задания 2.5.

Ниже представлено возможное решение данной задачи на языке Python.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import math
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
hand = serial.Serial("COM16", 115200, timeout = None)
max_points = 100   #number of points in frame
# create a figure with two subplots
fig, (ax1, ax2, ax3) = plt.subplots(3,1)
# intialize three line objects (one in each axes)
line1, = ax1.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line2, = ax2.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line3, = ax3.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line = [line1, line2, line3]
# the same axes initalizations as before (just now we do it for both of them)
ax1.set_ylim(0, 256)
ax1.set_xlim(0, 100)
```

```
ax1.grid()
ax2.set_ylim(0, 256)
ax2.set_xlim(0, 100)
ax2.grid()
ax3.set_ylim(0, 30000)
ax3.set_xlim(0, 100)
ax3.grid()
ax1.set_title("Amp_1")
ax2.set_title("Amp_2")
ax3.set_title("Alpha")
def init():
    return line
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-30)/(145-30)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 53) / (224 - 53)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 66) / (223 - 66)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state - 48) / (214 - 48)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state - 34) / (204 - 34)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
n = 0
t = time.clock()
glove.flushInput()
new_y1 = [np.nan]*30
new_y2 = [np.nan]*30
new_y3 = [np.nan]*512
avg = [np.nan]*10
def animate(data):
    trigger_eeg = 10000
    trigger_emg1 = 230
    trigger_emg2 = 230
    buff = 12 #number of bytes in series
    global t
    global n
    global new_y1
    global new_y3
```

```
global new_y2
global avg
n += 1
global i
if n > 30:    #wait until initialization ends
    delta_t = time.clock() - t
    t = time.clock()
    in_queue = glove.in_waiting
    dat = glove.read(in_queue)
    dat = struct.unpack('c'*in_queue, dat)
    list_dat = []
    for i in dat:
        list_dat.append(int.from_bytes(i, byteorder='big'))
    emg1 = []
    i = 0
    while i+buff-1 < len(list_dat):
        emg1.append(list_dat[i+buff-1])
        i += buff
    eeg = []
    i = 0
    while i+buff-2-1 < len(list_dat):
        eeg.append(list_dat[i+buff-2-1])
        i += buff
    emg = []
    i = 0
    while i+buff-4-1 < len(list_dat):
        emg.append(list_dat[i+buff-4-1])
        i += buff
    new_y1 = np.r_[new_y1[len(emg1):], emg1]  # grab current data
    new_y2 = np.r_[new_y2[len(eeg):], eeg]  # grab current data
    new_y3 = np.r_[new_y3[len(emg):], emg]  # grab current data
    signal = np.array(new_y3, dtype=int)
    fourier = np.fft.fft(signal)
    num = signal.size
    timestep = delta_t/(in_queue//buff)
    avg = np.r_[avg[1:], timestep]
    delta = sum(avg)/10
    freq = np.fft.fftfreq(num, d=delta)
    fourier = fourier[:256]
    freq = freq[:256]
    amp = []
    for i in fourier:
        amp.append(pow(i.real*i.real+i.imag*i.imag, 1/2))
    interval = freq[1]-freq[0]
    area = 0
    for i in range(0, len(amp)):
        if (freq[i] >= 8)and(freq[i]<=14):
            area += amp[i]*interval
    areas = line[2].get_ydata()          #alpha
    areas = np.r_[areas[-99:], area]
    if area > trigger_eeg:
        print("EEG TRIGGERD")
        for i in range(0,3):
            arr = bytearray([i, 255])
            glove.write(arr)
    else:
        print("EEG UNTRIGGERD")
        for i in range(0, 3):
            arr = bytearray([i, 0])
            glove.write(arr)
    line[2].set_ydata(areas)
    areas = line[0].get_ydata()         #emg1
    amp_emg1 = max(new_y1)-min(new_y1)
    areas = np.r_[areas[-99:], amp_emg1]
```

```
        if amp_emg1 > trigger_emg1:
            print("EMG1 TRIGGERD")
            for i in range(1, 4):
                arr = bytearray([i, 140])
                print(arr)
                hand.write(arr)
                print('here')
            arr = bytearray([0, 11])
            print(arr)
            hand.write(arr)
            print('here')
            arr = bytearray([4, 11])
            print(arr)
            hand.write(arr)
            print('here')
        line[0].set_ydata(areas)  # set the new ydata
        areas = line[1].get_ydata()         #emg2
        amp_emg2 = max(new_y2)-min(new_y2)
        areas = np.r_[areas[-99:], amp_emg2]
        if amp_emg2 > trigger_emg2:
            print("EMG2 TRIGGERD")
            for i in range(1, 4):
                arr = bytearray([i, 21])
                print(arr)
                hand.write(arr)
            line[1].set_ydata(areas)
            arr = bytearray([0, 140])
            print(arr)
            hand.write(arr)
            arr = bytearray([4, 140])
            print(arr)
            hand.write(arr)
        line[1].set_ydata(areas)
    return line
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,
interval=0,
                               blit=True)  # the fastest evaluation reached with
interval=0
plt.show()
```

4.5 Часть 5 - Испытания на стенде

Задание 5.1 (*9 баллов*)

Реализуйте следующую систему: механическая рука должна повторять жесты с перчатки: кулак, щепотка (захват большим пальцем и указательным), жест "ОК" (оттопыренный большой палец, остальные сжаты).

Комментарий: за каждый из жестов ставится по три балла. В течение одной попытки необходимо продемонстрировать все три жеста.

Решение задачи 5.1

В данном задании для Arduino Nano используется программа из задания 3.2, для Arduino Uno - программа из задания 2.5.

Ниже представлено возможное решение данной задачи на языке Python.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
```

```
from matplotlib import animation
import numpy as np
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
hand = serial.Serial("COM16", 115200, timeout = None)
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-25)/(200-25)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 18) / (200 - 18)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 18) / (200 - 18)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state) / (160)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state) / (160)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
corrected = mapping(0, 150)
arr = bytearray([0, corrected])
hand.write(arr)
corrected = mapping(1, 140)
arr = bytearray([1, corrected])
hand.write(arr)
corrected = mapping(2, 190)
arr = bytearray([2, corrected])
hand.write(arr)
corrected = mapping(3, 160)
arr = bytearray([3, corrected])
hand.write(arr)
corrected = mapping(4, 170)
arr = bytearray([4, corrected])
hand.write(arr)
time.sleep(1)  #could be changed
corrected = mapping(0, 150)
arr = bytearray([0, corrected])
hand.write(arr)
corrected = mapping(1, 140)
arr = bytearray([1, corrected])
hand.write(arr)
```

```
corrected = mapping(2, 10)
arr = bytearray([2, corrected])
hand.write(arr)
corrected = mapping(3, 5)
arr = bytearray([3, corrected])
hand.write(arr)
corrected = mapping(4, 10)
arr = bytearray([4, corrected])
hand.write(arr)
time.sleep(1)  #could be changed
corrected = mapping(0, 10)
arr = bytearray([0, corrected])
hand.write(arr)
corrected = mapping(1, 140)
arr = bytearray([1, corrected])
hand.write(arr)
corrected = mapping(2, 190)
arr = bytearray([2, corrected])
hand.write(arr)
corrected = mapping(3, 160)
arr = bytearray([3, corrected])
hand.write(arr)
corrected = mapping(4, 170)
arr = bytearray([4, corrected])
hand.write(arr)
```

Задание 5.2 (*21 баллов*)

Реализуйте следующую систему: механическая рука должна изображать жесты из предыдущего пункта. Управление механической рукой при этом осуществляется на основе данных с датчиков ЭМГ и ЭЭГ. За дополнительные жесты ставятся бонусные баллы.

Комментарий: за каждый из жестов ставится по три балла. В течение одной попытки необходимо продемонстрировать все три жеста. За каждый дополнительный жест ставится по 2 бонусных балла.

Решение задачи 5.2

В данном задании для Arduino Nano используется программа из задания 3.2, для Arduino Uno - программа из задания 2.5.

Ниже представлено возможное решение данной задачи на языке Python.

```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import math
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
hand = serial.Serial("COM16", 115200, timeout = None)
max_points = 100   #number of points in frame
# create a figure with two subplots
fig, (ax1, ax2, ax3) = plt.subplots(3,1)
# intialize three line objects (one in each axes)
line1, = ax1.plot(np.arange(max_points),
```

```
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line2, = ax2.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line3, = ax3.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line = [line1, line2, line3]
# the same axes initalizations as before (just now we do it for both of them)
ax1.set_ylim(0, 256)
ax1.set_xlim(0, 100)
ax1.grid()
ax2.set_ylim(0, 256)
ax2.set_xlim(0, 100)
ax2.grid()
ax3.set_ylim(0, 30000)
ax3.set_xlim(0, 100)
ax3.grid()
ax1.set_title("Amp_1")
ax2.set_title("Amp_2")
ax3.set_title("Alpha")
def init():
    return line
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-30)/(145-30)*(150-10)))
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 53) / (224 - 53)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 66) / (223 - 66)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state - 48) / (214 - 48)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state - 34) / (204 - 34)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
n = 0
t = time.clock()
glove.flushInput()
new_y1 = [np.nan]*30
```

```
new_y2 = [np.nan]*30
new_y3 = [np.nan]*512
avg = [np.nan]*10
def animate(data):
    trigger_eeg = 10000
    trigger_emg1 = 230
    trigger_emg2 = 230
    buff = 12 #number of bytes in series
    global t
    global n
    global new_y1
    global new_y3
    global new_y2
    global avg
    n += 1
    global i
    if n > 30:    #wait until initialization ends
        delta_t = time.clock() - t
        t = time.clock()
        in_queue = glove.in_waiting
        dat = glove.read(in_queue)
        dat = struct.unpack('c'*in_queue, dat)
        list_dat = []
        for i in dat:
            list_dat.append(int.from_bytes(i, byteorder='big'))
        emg1 = []
        i = 0
        while i+buff-1 < len(list_dat):
            emg1.append(list_dat[i+buff-1])
            i += buff
        eeg = []
        i = 0
        while i+buff-2-1 < len(list_dat):
            eeg.append(list_dat[i+buff-2-1])
            i += buff
        emg = []
        i = 0
        while i+buff-4-1 < len(list_dat):
            emg.append(list_dat[i+buff-4-1])
            i += buff
        new_y1 = np.r_[new_y1[len(emg1):], emg1]  # grab current data
        new_y2 = np.r_[new_y2[len(eeg):], eeg]  # grab current data
        new_y3 = np.r_[new_y3[len(emg):], emg]  # grab current data
        signal = np.array(new_y3, dtype=int)
        fourier = np.fft.fft(signal)
        num = signal.size
        timestep = delta_t/(in_queue//buff)
        avg = np.r_[avg[1:], timestep]
        delta = sum(avg)/10
        freq = np.fft.fftfreq(num, d=delta)
        fourier = fourier[:256]
        freq = freq[:256]
        amp = []
        for i in fourier:
            amp.append(pow(i.real*i.real+i.imag*i.imag, 1/2))
        interval = freq[1]-freq[0]
        area = 0
        for i in range(0, len(amp)):
            if (freq[i] >= 8)and(freq[i]<=14):
                area += amp[i]*interval
        areas = line[2].get_ydata()         #alpha
        areas = np.r_[areas[-99:], area]
        if area > trigger_eeg:
            print("EEG TRIGGERED")   #OK
```

```
            for i in range(1, 4):
                arr = bytearray([i, 140])
                print(arr)
                hand.write(arr)
                print('here')
            arr = bytearray([0, 160])
            print(arr)
            hand.write(arr)
            print('here')
            arr = bytearray([4, 11])
            print(arr)
            hand.write(arr)
        line[2].set_ydata(areas)
        areas = line[0].get_ydata()        #emg1
        amp_emg1 = max(new_y1)-min(new_y1)
        areas = np.r_[areas[-99:], amp_emg1]
        if amp_emg1 > trigger_emg1:
            print("EMG1 TRIGGERED")  #fist
            for i in range(1, 4):
                arr = bytearray([i, 140])
                print(arr)
                hand.write(arr)
                print('here')
            arr = bytearray([0, 11])
            print(arr)
            hand.write(arr)
            print('here')
            arr = bytearray([4, 11])
            print(arr)
            hand.write(arr)
            print('here')
        line[0].set_ydata(areas)  # set the new ydata
        areas = line[1].get_ydata()         #emg2
        amp_emg2 = max(new_y2)-min(new_y2)
        areas = np.r_[areas[-99:], amp_emg2]
        if amp_emg2 > trigger_emg2:
            print("EMG2 TRIGGERED")     #hand
            for i in range(1, 4):
                arr = bytearray([i, 21])
                print(arr)
                hand.write(arr)
            line[1].set_ydata(areas)
            arr = bytearray([0, 140])
            print(arr)
            hand.write(arr)
            arr = bytearray([4, 140])
            print(arr)
            hand.write(arr)
        line[1].set_ydata(areas)
    return line
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,
interval=0,
                               blit=True)  # the fastest evaluation reached with
interval=0
plt.show()
```

Задание 5.3 (*20 баллов*)

На механическую руку планируется оказать воздействие, конкретно - на кончики пальцев руки. Вам необходимо реализовать обратную связь для того, чтобы распознать, на какой именно палец оказывается воздействие (прикосновение к пальцу) и произвести ответное действие соответствующим пальцем (при этом палец в согнутом положении должен остаться,

как минимум, в течение 3 секунд). При этом необходимо сжать только тот палец, на который оказано воздействие).

Комментарий 1: *Первые десять баллов начисляются за ответное действие, совершенное с помощью системы управления на основе датчиков изгиба.*

Остальные десять баллов начисляются за ответное действие, реализованное с помощью системы управления на основе модулей ЭМГ и ЭЭГ (при этом использование датчиков изгиба запрещено). Баллы начисляются по прогрессивной шкале.

Решение задачи 5.3

В данном задании для Arduino Nano используется программа из задания 3.2, для Arduino Uno - программа из задания 2.5.

Ниже представлено возможное решение данной задачи на языке Python.

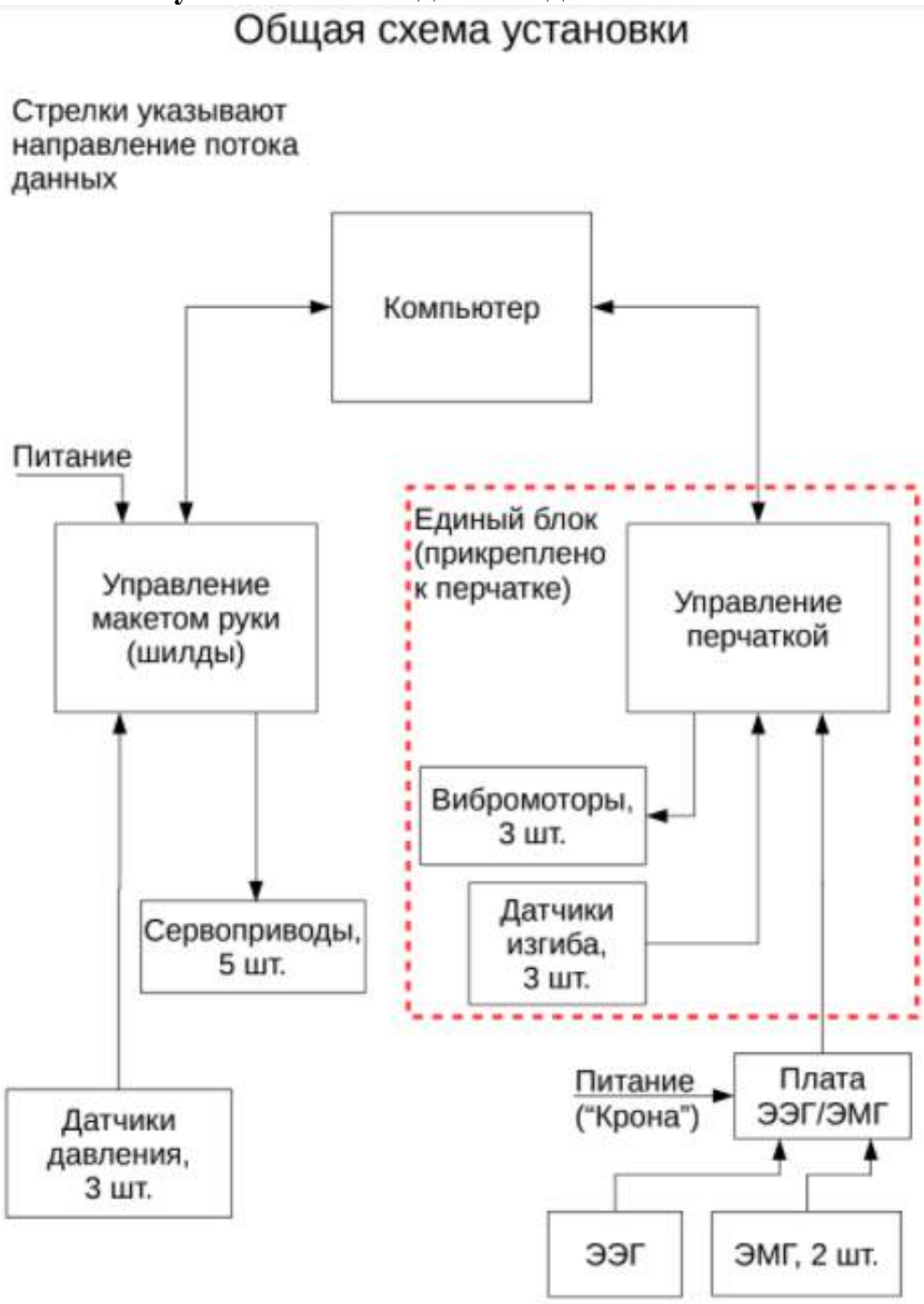
```
import serial
import time
import struct
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import math
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)  #change priority
glove = serial.Serial("COM3", 115200, timeout = None) #port number, baude rate
hand = serial.Serial("COM16", 115200, timeout = None)
max_points = 100   #number of points in frame
# create a figure with two subplots
fig, (ax1, ax2, ax3) = plt.subplots(3,1)
# intialize three line objects (one in each axes)
line1, = ax1.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line2, = ax2.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line3, = ax3.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line = [line1, line2, line3]
# the same axes initalizations as before (just now we do it for both of them)
ax1.set_ylim(0, 256)
ax1.set_xlim(0, 100)
ax1.grid()
ax2.set_ylim(0, 256)
ax2.set_xlim(0, 100)
ax2.grid()
ax3.set_ylim(0, 30000)
ax3.set_xlim(0, 100)
ax3.grid()
ax1.set_title("Amp_1")
ax2.set_title("Amp_2")
ax3.set_title("Alpha")
def init():
    return line
def mapping(number, state):
    if number == 0:
        res = int(abs(160 - abs(state-30)/(145-30)*(150-10)))
```

```
        if res < 10:
            return 10
        if res > 150:
            return 150
        return res
    if number == 1:
        res = int(abs((abs(state - 53) / (224 - 53)) * (140 - 10) + 10))
        if res < 10:
            return 10
        if res > 140:
            return 140
        return res
    if number == 2:
        res = int(abs((abs(state - 66) / (223 - 66)) * (190 - 10) + 10))
        if res < 10:
            return 10
        if res > 190:
            return 190
        return res
    if number == 3:
        res = int(abs((abs(state - 48) / (214 - 48)) * (160 - 5) + 5))
        if res < 5:
            return 5
        if res > 160:
            return 160
        return res
    if number == 4:
        res = int(abs(170 - (abs(state - 34) / (204 - 34)) * (170 - 10)))
        if res > 170:
            return 170
        if res < 10:
            return 10
        return res
n = 0
t = time.clock()
glove.flushInput()
new_y1 = [np.nan]*30
new_y2 = [np.nan]*30
new_y3 = [np.nan]*512
avg = [np.nan]*10
def animate(data):
    trigger_eeg = 10000
    trigger_emg1 = 230
    trigger_emg2 = 230
    buff = 12 #number of bytes in series
    global t
    global n
    global new_y1
    global new_y3
    global new_y2
    global avg
    n += 1
    global i
    if n > 30:    #wait until initialization ends
        delta_t = time.clock() - t
        t = time.clock()
        in_queue = glove.in_waiting
        dat = glove.read(in_queue)
        dat = struct.unpack('c'*in_queue, dat)
        list_dat = []
        for i in dat:
            list_dat.append(int.from_bytes(i, byteorder='big'))
        emg1 = []
        i = 0
```

```
while i+buff-1 < len(list_dat):
    emg1.append(list_dat[i+buff-1])
    i += buff
eeg = []
i = 0
while i+buff-2-1 < len(list_dat):
    eeg.append(list_dat[i+buff-2-1])
    i += buff
emg = []
i = 0
while i+buff-4-1 < len(list_dat):
    emg.append(list_dat[i+buff-4-1])
    i += buff
new_y1 = np.r_[new_y1[len(emg1):], emg1]  # grab current data
new_y2 = np.r_[new_y2[len(eeg):], eeg]  # grab current data
new_y3 = np.r_[new_y3[len(emg):], emg]  # grab current data
print("LIST")
print(list_dat)
# if len(list_dat) > 5:
#     corrected = mapping(0, list_dat[1])
#     arr = bytearray([0, corrected])
#     hand.write(arr)
#     corrected = mapping(1, list_dat[3])
#     arr = bytearray([1, corrected])
#     hand.write(arr)
#     corrected = mapping(2, list_dat[3])
#     arr = bytearray([2, corrected])
#     hand.write(arr)
#     corrected = mapping(3, list_dat[5])
#     arr = bytearray([3, corrected])
#     hand.write(arr)
#     corrected = mapping(4, list_dat[5])
#     arr = bytearray([4, corrected])
#     hand.write(arr)
in_queue = hand.in_waiting
dat = hand.read(in_queue)
dat = struct.unpack('c' * in_queue, dat)
list_dat = []
for i in dat:
    list_dat.append(int.from_bytes(i, byteorder='big'))
print(list_dat)
if (len(list_dat) > 5):
    arr = bytearray([0, list_dat[1]])
    glove.write(arr)
    arr = bytearray([1, list_dat[3]])
    glove.write(arr)
    arr = bytearray([2, list_dat[5]])
    glove.write(arr)
signal = np.array(new_y3, dtype=int)
fourier = np.fft.fft(signal)
num = signal.size
timestep = delta_t/(in_queue//buff)
avg = np.r_[avg[1:], timestep]
delta = sum(avg)/10
freq = np.fft.fftfreq(num, d=delta)
fourier = fourier[:256]
freq = freq[:256]
amp = []
for i in fourier:
    amp.append(pow(i.real*i.real+i.imag*i.imag, 1/2))
interval = freq[1]-freq[0]
area = 0
for i in range(0, len(amp)):
    if (freq[i] >= 8)and(freq[i]<=14):
```

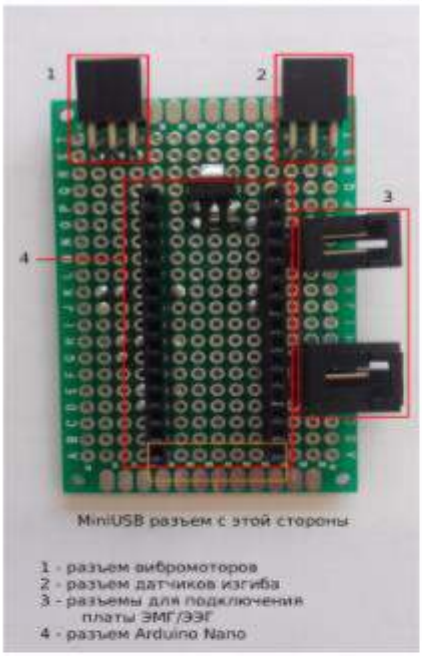
```
            area += amp[i]*interval
        areas = line[2].get_ydata()         #alpha
        areas = np.r_[areas[-99:], area]
        if area > trigger_eeg:
            print("EEG TRIGGERED")   #OK
            for i in range(1, 4):
                arr = bytearray([i, 140])
                hand.write(arr)
            arr = bytearray([0, 160])
            hand.write(arr)
            arr = bytearray([4, 11])
            hand.write(arr)
        line[2].set_ydata(areas)
        areas = line[0].get_ydata()        #emg1
        amp_emg1 = max(new_y1)-min(new_y1)
        areas = np.r_[areas[-99:], amp_emg1]
        if amp_emg1 > trigger_emg1:
            print("EMG1 TRIGGERED")  #fist
            arr = bytearray([0, 11])
            hand.write(arr)
            for i in range(1, 4):
                arr = bytearray([i, 140])
                hand.write(arr)
            arr = bytearray([4, 11])
            hand.write(arr)
        line[0].set_ydata(areas)  # set the new ydata
        areas = line[1].get_ydata()         #emg2
        amp_emg2 = max(new_y2)-min(new_y2)
        areas = np.r_[areas[-99:], amp_emg2]
        if amp_emg2 > trigger_emg2:
            print("EMG2 TRIGGERED")     #hand
            arr = bytearray([0, 140])
            hand.write(arr)
            for i in range(1, 4):
                arr = bytearray([i, 21])
                hand.write(arr)
            line[1].set_ydata(areas)
            arr = bytearray([4, 140])
            hand.write(arr)
        line[1].set_ydata(areas)
    return line
anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,
interval=0,
                               blit=True)  # the fastest evaluation reached with
interval=0
plt.show()
```

4.6 Приложение 1 к условию командного задания

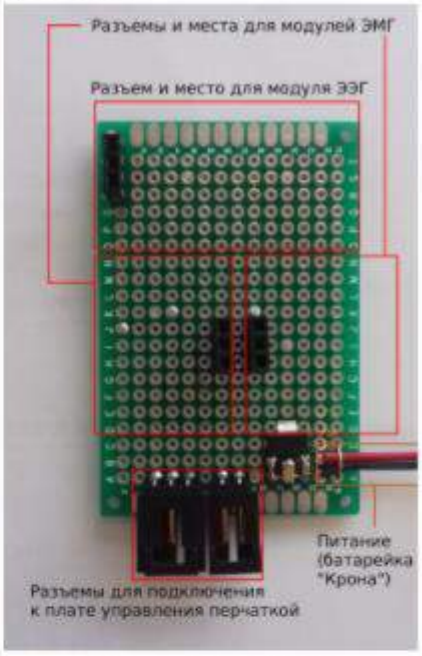


4.7 Приложение 2 к условию командного задания

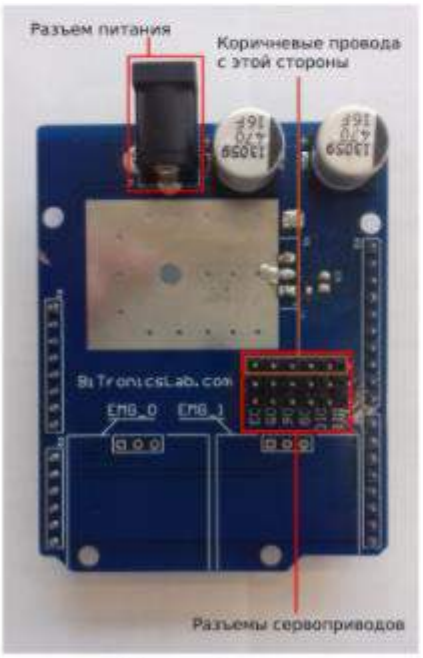
Схемы подключения и фото плат



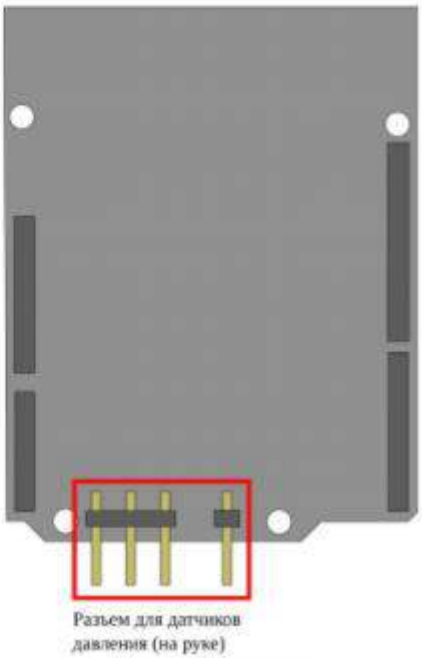
Плата управления перчаткой



Плата ЭМГ/ЭЭГ



Синий шилд



Черный шилд

4.8 Приложение 3 к условию командного задания

Описание протокола обмена данными

Подключение – USB. Скорость - 115200 бод.

Термины:

- Каждому устройству, принимающему или отдающему данные, соответствует ровно один **канал**.
- Канал, по которому устройство отдает данные, называется **восходящим**. Канал, по которому устройство принимает данные, называется **нисходящим**.
- Каждому каналу соответствует число от 0 до 7 – **номер канала**. Номера каналов должны начинаться с 0 и идти по возрастанию, без пропусков. Номера восходящих (нисходящих) каналов не должны повторяться, однако допустимо повторение номеров для каналов разного направления.
- Каждому восходящему каналу устройства соответствует нисходящий канал на компьютере и наоборот.

  Пример 1: Arduino, к которой подключены 3 датчика изгиба (3 восходящих канала) и 2 вибромотора (2 нисходящих канала). Восходящие каналы имеют номера 0, 1, 2, нисходящие – 0, 1. Соответственно, на компьютере 3 нисходящих канала с номерами 0, 1, 2 и 2 восходящих канала с номерами 0, 1.

Передача данных:

Способ передачи данных является одинаковым для всех каналов.

Данные передаются парами чисел (двумя байтами). Первое число – номер канала (от 0 до 7). Второе число – данные (от 0 до 255). Пары идут друг за другом по возрастанию номера канала. После пары от последнего канала идет пара от канала 0. Промежутки времени между передачами могут быть любыми.

Соединение дуплексное, т.е. данные могут одновременно передаваться в обе стороны – от компьютера к устройству и от устройства к компьютеру.

Пример 2: для установки из Примера 1 потоки данных могут выглядеть так:

От установки к компьютеру:

… 142 0 21 1 144 2 0 0 …

Подчеркнуты пары значений: канал 0 – 21, канал 1 – 144, канал 2 – 0; видны части предыдущей и последующей передач.

От компьютера к установке:

… 1 44 0 200 1 255 0 …

Аналогично, подчеркнуты пару значений: канал 1 – 44, канал 0 – 200, канал 1 – 255 (новое значение).

Примечание: нужно понимать, что передаются именно байты со значениями (например, 123), а не цифры ('123'). Во втором случае на самом деле передается 3 байта (со значениями соотв. 49, 50, 51; см. ASCII).

Прием данных:

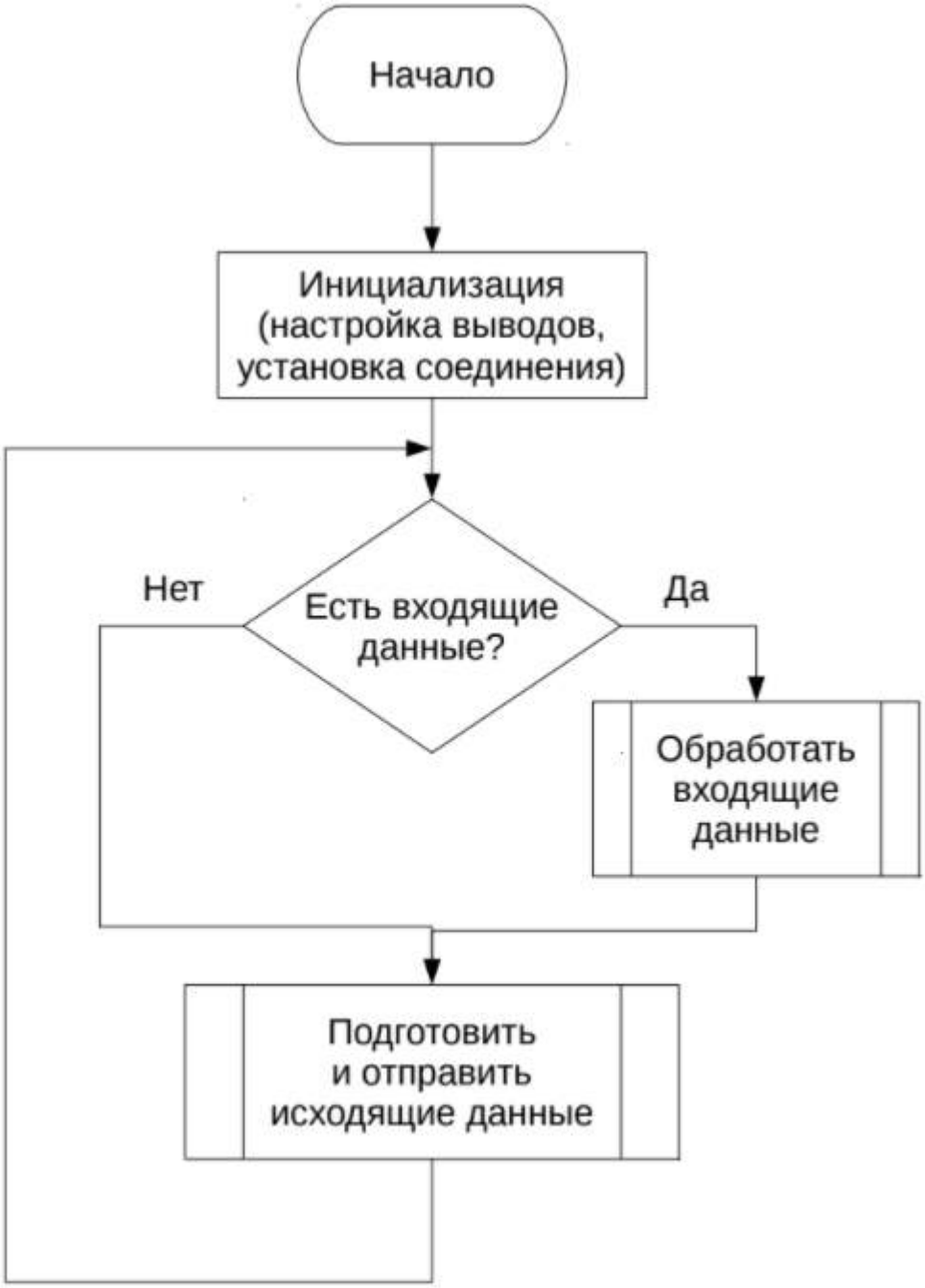
Для того, чтобы принимать данные, необходимо знать, какой байт обозначает номер канала, а какой – данные. Т. к. данные могут принимать любое значение от 0 до 255, то просто проверка того, что значение байта меньше, чем 8 (номера каналов – 0-7), не подойдет.

Предлагается способ синхронизации, основанный на следующей идее:

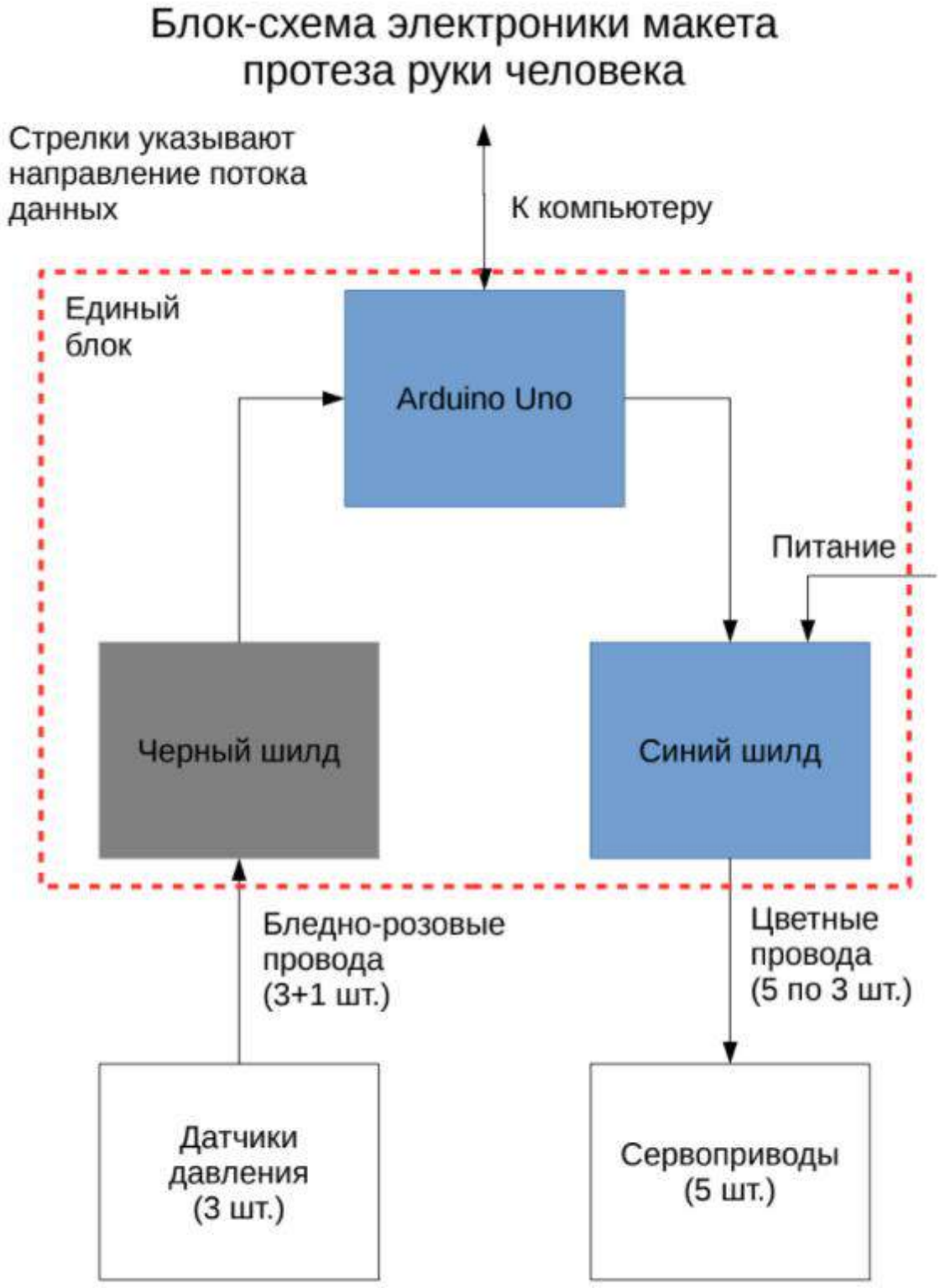
- читаются 3-5 идущих подряд байт
- проверяется, идут ли значения нечетных (или четных) байт по возрастанию (с учетом цикличности (за последним каналом – нулевой))
- если да, то нечетные (четные) байты – номера каналов, а четные (нечетные) – данные, если нет, то наоборот

Такой протокол требуется, как правило, только со стороны компьютера.

4.9 Приложение 4 к условию командного задания



4.10 Приложение 5 к условию командного задания



Замечание: сначала подключается блок питания, а затем USB. Отключение – в обратном порядке.

4.11 Приложение 6 к условию командного задания

Получение, обработка и отправка данных

Для работы приложений в реальном времени необходимы ресурсы, которые распределяются между всеми процессами, запущенными на компьютере. Так как они имеют различный приоритет выполнения, то распределение происходит неравномерно. Для того, чтобы наша программа могла получить достаточно ресурсов для работы, повысим ее приоритет при запуске, добавив в начале скрипта:

```
import psutil, os
p = psutil.Process(os.getpid())
p.nice(psutil.REALTIME_PRIORITY_CLASS)
```

Для взаимодействия с датчиками на перчатке и рукой-манипулятором воспользуемся интерфейсом COM-порта. Инициализируем устройства:

```
import serial
glove = serial.Serial("номер_порта", 115200, timeout = None)
hand = serial.Serial("номер_порта", 115200, timeout = None)
```

Номер порта может принимать значения **COM1,COM2,COM3** и далее. Необходимо самостоятельно контролировать соответствие подключаемого устройства и номера порта, например, с помощью Диспетчера Устройств.
При повторном подключении устройства в один и тот же разъем номер порта может измениться.
Для получения количества байт, находящихся в буфере, можно использовать:

```
in_queue = glove.in_waiting
```

Для считывания N байт из буфера, по умолчанию, N=1:

```
dat = glove.read(N)
```

В данном случае переменная dat будет иметь тип bytes. Для преобразования данных в список из целых десятичных чисел предлагаем воспользоваться:

```
dat = struct.unpack('c'*N, dat)
list_dat = []
for i in dat:
   list_dat.append(int.from_bytes(i, byteorder='big'))
```

Для управления устройствами необходимо передавать данные по COM-порту, за это отвечает функция записи:

```
arr = bytearray(список)
hand.write(arr)
```

Во время работы программы в буфер порта будет непрерывно поступать информация от устройств. Для того, чтобы очистить буферы устройств от текущего содержимого, можно использовать команды:

```
glove.flushInput()
hand.flushInput()
```

Также может оказаться полезным использование функции приостановления выполнения программы на заданное число секунд:

```
import time
time.sleep(сек)
```

Для определения времени можно воспользоваться функцией, которая возвращает время, прошедшее с момента первого вызова данной функции:

```
time.clock()
```

Визуализация данных

Для визуализации данных сразу с нескольких каналов рекомендуем воспользоваться модулем `animate` из библиотеки `matplotlib`:

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation

# количество точек в кадре
max_points = N

# создание figure с двумя графиками
fig, (ax1) = plt.subplots(2,1)

# инициализация двух line-объектов(по одному на каждый график)
line1, = ax1.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)
line2, = ax2.plot(np.arange(max_points),
                np.ones(max_points, dtype=np.float)*np.nan,
                lw=2, animated=True)

line = [line1, line2]


# инициализация осей графиков
for ax in [ax1, ax2]:
   ax.set_ylim(0, 260) #диапазон значений по оси y
   ax.set_xlim(0, 100) #диапазон значений по оси x
   ax.grid() #показывать сетку
ax1.set_title("первый_график")
ax2.set_title("второй_график")

#функция начальной инициализации figure
def init():
   return line

#функция, которая будет вызываться для обновления значений графиков
def animate(data):
      #место для вашего кода

      #обновление данных line-объектов
      old_y1 = line[0].get_ydata()
      #добавление нового значения и удаление первого
      new_y1 = np.r_[old_y1[1:], новое_значение]  line[0].set_ydata(new_y1)

      old_y2 = line[1].get_ydata()
      new_y2 = np.r_[old_y2[1:], новое_значение]
      line[1].set_ydata(new_y2)

      return line
```

```
#функция, вызывающая функцию animate для обновления графиков
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=200, interval=0, blit=True)
plt.show()
```

Преобразование Фурье

Для осуществления Фурье-преобразования воспользуемся функциями библиотеки `numpy`:

```
#если new_y1 имеет тип list, то переводим его к ndarray
signal = np.array(new_y1, dtype=int)

fourier = np.fft.fft(signal)
n = signal.size
timestep = время_между_соседними_измерениями_в_сек

#list, содержащий X-координату преобразования(частоты)
freq = np.fft.fftfreq(n, d=timestep)

#оставим только значения с положительными частотами
fourier = fourier[:n//2]
freq = freq[:n//2]

#list, содержащий Y-координату преобразования(амплитуды)
amp = []
for i in fourier:
   amp.append(pow(i.real*i.real+i.imag*i.imag, 1/2))

#визуализируем результаты преобразования(внутри функции animate)
line[0].set_data(freq, amp)
```

4.12 Критерии оценивания

Максимальное количество баллов за финальную часть - 100 баллов. В таблице ниже представлены критерии оценивания по каждой из задач финальной части:

№ задания	Критерий оценивания	Максимально возможный балл
За задания 1,2,3 блоков и за задания 4.1, 4.2 ставился либо полный балл, либо 0 баллов		
1.1	Полный балл за задачу ставится в том случае, если датчики изгиба и вибромоторы подключены согласно схеме, представленной в Приложении 2, а также при корректном креплении этих датчиков к перчатке (без повреждения функциональности датчиков). Только в случае правильного выполненного задания выдается провод для подключения Arduino Nano к компьютеру, необходимый для выполнения дальнейших заданий.	3
1.2	Полный балл за задачу ставится в том случае, если программа для Arduino Nano оцифровывает значения с датчиков изгиба и отправляет данные на компьютер согласно протоколу, представленному в Приложении 3.	3
1.3	Полный балл за задание ставится в том случае, если программа на языке Python принимает данные и выводит их на экран так, чтобы каждому каналу соответствовал свой столбец значений при выводе на экран.	2
1.4	Полный балл за задачу ставится в том случае, если программа (написанная на Python) включает и выключает вибромоторы в следующем порядке: сначала включается первый вибромотор, потом второй, затем третий. После этого выключался третий вибромотор, затем второй вибромотор и последним - первый вибромотор. Данный порядок был озвучен участникам в первый день Олимпиады.	2
2.1	Полный балл за задачу ставится, если датчики и сервоприводы подключены согласно схеме, представленной в приложении 5. Только в случае правильного выполненного задания выдается блок питания и провод miniUSB, необходимые для дальнейшего выполнения заданий.	2

2.2	Для сдачи данного задания участникам необходимо перевести пальцы макета руки в крайние положения. Следить за тем, чтобы в крайнем положении макета руки сервопривод обязательно останавливался. При этом параметры рабочих углов сервоприводов должны быть подобраны так, чтобы пальцы могли максимально возможно сжиматься/разжиматься (на сколько это механически допускает конструкция). Примеры неправильных ситуаций: палец максимально сжался, при этом соответствующий сервопривод работает; неправильно определено крайнее положение пальца. При правильной калибровке ставится полный балл за задачу.	3
2.3	Полный балл ставится, если скетч для Arduino Uno реализует сжатие кисти макета руки пока не достигается максимум сжатия кисти или пока пальцы макета не упрутся в препятствия (в этом случае сервоприводы должны обязательно остановится). При проверке в качестве препятствия использовался подставленный палец проверяющего. Палец человека в качестве препятствия мог подставляться под произвольные пальцы макета руки в случайном порядке.	2,5
2.4	Полный балл ставится, если программа корректно оцифровывает значения с датчиков давления и отправляет на компьютер согласно Протоколу, представленному в Приложении 3.	0,5
2.5	Задание оценивается аналогично пункту 2.3 (Отличие от пункта 2.3 в том, что в данном задание программа должна быть реализована на языке Python)	2
3.1	Полный балл ставится, если два модуля электромиограммы и модуль электроэнцефалограммы корректно (согласно Приложению 2) подключены к платам. Только при правильно выполненном задании команде выдается элемент питания, необходимый для дальнейшего выполнения заданий.	1
3.2	Полный балл ставится, если программа помимо данных из п.1.2 передает данные с модулей ЭМГ и ЭЭГ согласно протоколу из Приложения 3.	1
3.3	Полный балл ставится, если программа принимает данные от Arduino Nano и выводит их на экран таким образом, чтобы каждому каналу соответствовал свой столбец значений при выводе на экран.	0,5

3.4	Полный балл ставится, если полученный сигнал удовлетворяет критериям из условия задания.	2
3.5	Полный балл ставится, если визуализация зависимости амплитуды сигнала ЭМГ от времени корректно реализована: для проверки датчик подключался к одной из мышц на руке проверяющего, при напряжении мышцы проверяющего амплитуда сигнала на экране увеличивалась, при расслаблении - уменьшалась.	2,5
3.6	При проверке сначала один из членов команды демонстрирует визуализацию альфа-ритма от времени - испытуемый закрывает глаза, расслабляется - это соответствует состоянию альфа ритма (т. е. амплитуда сигнала на экране должна увеличиваться в данный момент). Затем то же самое повторяет проверяющий. При корректном отображении ставится полный балл.	3
4.1	Полный балл ставится за правильное осуществленное формирование управляющих команд для всех пяти пальцев руки (а не трех!), также при оценивании учитывается повторяемость движений (с учетом расположения датчиков).	6
4.2	Полный балл ставится за правильно реализованную обратную связь: вибромоторы должны включаться на перчатке в тот момент, когда механическая рука упирается в препятствие, причем интенсивность вибрации обязательно должна повышаться с увеличением воздействия на датчики давления.	6
4.3	В данном задании баллы начислялись следующим образом: Сжатие механической руки с помощью первого модуля электромиограммы (ЭМГ) - 2,5 балла; Разжатие механической руки с помощью второго модуля электромиограммы (ЭМГ) - 2,5 балла; Активация вибромоторов с помощью модуля электроэнцефалограммы (ЭЭГ) - 3 балла.	8
В блоке №5 дается три попытки для сдачи каждого задания из этого блока. На каждую попытку дается 5 минут. Очередность определяется случайной жеребьевкой.		
5.1	За каждый из жестов ставится по три балла. Чтобы набрать полный балл за задачу необходимо в течение одной попытки продемонстрировать все три жеста.	9

5.2	В данном задании баллы начисляются следующим образом: Реализация основных жестов (кулак, щепотка, жест "ОК") - по 3 балла за каждый жест. За каждые дополнительные жесты по 2 бонусных балла, не более 6 дополнительных жестов. Участники перед выполнением задания должны заранее сообщить, какие жесты они планируют показать. Проверяющий при необходимости мог попросить несколько раз повторить тот или иной жест (для проверки повторяемости работы системы управления).	21
5.3	Первые десять баллов начисляются за ответное действие, совершенное с помощью системы управления на основе датчиков изгиба (проверяющий в различных комбинациях оказывает 10 воздействий на пальцы макета протеза - в том числе одновременно на несколько пальцев макета). Остальные десять баллов начисляются за ответное действие (на 10 воздействий проверяющего), реализованное с помощью системы управления на основе модулей ЭМГ и ЭЭГ (без использования датчиков изгиба, использование датчика ЭЭГ обязательно). При проверке данного задания человек, реализовавший ответное действие закрывает глаза и не видит, на какой палец механической руки оказывается воздействие. Партнерам по команде запрещается общаться между собой во время сдачи данного задания.	20