Командный тур

6.1. Задачи

На этом этапе участникам необходимо было разработать систему оценки состояния водителя за рулем, определяющая опасные ситуации на дороге: сонливость и потеря внимания водителя. Для решения данной задачи участникам необходимо было использовать методы машинного обучения, компьютерного зрения и анализ биосигналов человека (кожно-гальваническая реакция, электрокардиограмма, электроэнцефалограмма).

Участникам было предложено решить 5 задач. Для решения задач необходимо было распределить между участниками команды следующие роли: физиолог, специалист по машинному обучению, специалист по компьютерному зрению, инженер-электронщик. Один человек может взять на себя несколько ролей.

Максимальное количество баллов — 100 баллов. В таблице ниже представлен график открытия задач и количество баллов, которые можно будет получить за задачи.

Таблица 1 - Баллы за задачи, график и роли

№	Кол-во баллов	Дата открытия	Срок сдачи (время местное)	Требуемые роли
1	35 баллов	19 марта	13:00, 22 марта	Физиолог, Специалист по машинному обучению
2	20 баллов	19 марта	13:00, 22 марта	Физиолог, Специалист по компьютерному зрению, инженер электронщик
3	30 баллов	20 марта	13:00, 22 марта	Все
4	10 баллов	20 марта	12:00, 22 марта	Физиолог
5	5 баллов	22 марта	13:00, 22 марта	Все

Задачи можно выполнять параллельно и сдавать в любом порядке. Внимательно ознакомьтесь с критериями оценивания задач (указаны после каждой задачи), количеством попыток и сроками для сдачи заданий.

Задача 6.1.1. (35 баллов)

Первое задание будет проходить в формате соревнования на платформе kaggle.com. Пригласительную ссылку на соревнование вы получите от судей на вашей площадке. Вам дана запись трёх сгенерированных сигналов, подобных тем, которые могут быть сняты с водителя большегрузного транспорта во время движения по дороге общего пользования. Ниже перечислены эти сигналы:

- одноканальная электроэнцефалограмма (ЭЭГ), зарегистрированная с затылочной части головы водителя;
- кожно-гальваническая реакция (КГР), электроды были закреплены на руке водителя;
- электрокардиограмма (ЭКГ), снятая в первом отведении.

Частота оцифровки данных сигналов составляет 250 Гц. Сигналы записаны в файлы train.csv и test.csv. Данные поделены на равные временные отрезки.

В первый столбец обоих файлов записаны идентификаторы временных отрезков данных. Во второй столбец train.csv записано число 1, если человек способен быстро реагировать на изменения обстановки во время движения, и число 0, если он находится в состоянии со сниженным уровнем внимания (или засыпает). Сигналы с датчиков ЭЭГ, КГР и ЭКГ записаны, соответственно, в третий, четвёртый и пятый столбцы файла train.csv и во второй, третий и четвёртый столбцы файла test.csv.

Постройте модель для определения состояния человека с точки зрения безопасности для остальных участников дорожного движения, проанализировав данные из файла train.csv. Воспользовавшись получившейся моделью, спрогнозируйте состояние водителя, основываясь на данных из файла test.csv. Свой ответ запишите в csv-файл, в первом столбце которого должны быть записаны идентификаторы отрезков данных из test.csv, а во втором столбце свои прогнозы состояния водителя соответствующие этим идентификаторам.

Разработчики данного задания при построении модели руководствовались статьями, перечисленными в Приложении 1.

Описание файлов:

Файлы можно скачать по ссылке: `https://drive.google.com/file/d/1ktSb7mZkAnekri6SVGfyuA3PUKYqy5oi/view?usp=sharing` (укороченная ссылка: `https://clck.ru/FVT9v`)

Данные разделены на две группы:

- Обучающая выборка (train.csv)
- Тестовая выборка (test.csv)

Обучающая выборка используется для построения ваших моделей. Для каждой строки из обучающей выборки мы даём вам информацию о том, способен ли водитель быстро реагировать на изменение окружающей обстановки во время движения или нет. Ваша модель может использовать сигналы ЭЭГ, КГР и ЭКГ для определения состояния водителя. Частота оцифровки сигналов 250 Гц.

Тестовая выборка применяется для оценки качества ваших моделей. Для примеров из тестовой выборки мы даём вам информацию о том, способен ли водитель быстро реагировать на изменение окружающей обстановки во время движения или нет. Примените свою модель для того чтобы определить состояние водителя на каж-

дом из временных отрезков тестовой выборки.

Кроме того, вам дан пример решения sample.csv, для которого значения столбца target были сгенерированны случайным образом.

Поля данных:

- ID — идентификатор временного отрезка данных;
- target — способен ли водитель быстро реагировать на изменение окружающей обстановки во время движения: 1 - способен, 0 - нет;
- eeg — одноканальная электроэнцефалограмма (ЭЭГ), зарегистрированная с затылочной части головы водителя;
- gsr — кожно-гальваническая реакция (КГР), электроды были закреплены на руке водителя;
- ecg — электрокардиограмма (ЭКГ), снятая в первом отведении.

Формат ответа: csv-файл, где в первом столбце находятся идентификаторы временных отрезков данных из test.csv, а во втором столбце - состояние человека (0 или 1).

Перед отправкой решения в систему Kaggle, необходимо передать код представителю жюри в аудитории. Решения без исходного кода засчитываться не будут.

Система оценки

Проверка точности решения осуществляется автоматически в системе Kaggle (подробнее см. Приложение 2). Оценка ставится пропорционально числу правильно определенных состояний, округляется до сотых по правилам математического округления. Например, если вы правильно определили состояние водителя на всех временных отрезках тестовых данных, вы получите 35 баллов. Если состояние водителя определено правильно только для половины отрезков, вы получите 17,5 баллов.

Число попыток: 3 попытки в сутки. Число попыток обновляется в 03:00 по московскому времени. Неиспользованные попытки сгорают.

Срок сдачи: 13:00, 22 марта 2019 г.

Пример программы-решения

Ниже представлено решение на языке Python3

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC

traindf = pd.read_csv('data\\train.csv')
testdf = pd.read_csv('data\\test.csv')
hz = 250  # частота оцифровки сигнала

# Функции для работы с ЭКГ
def getPeaks(ecg, thold=0.3):
        """Вычисление индексов точек, соответствующих вершинам R-зубцов
        thold - пороговое значение сигнала выше которого будут искаться
        пики"""
```

```
15          i = 0
16          peaks = []
17          while i < len(ecg) - 1:
18              if ecg[i] > 0.3 and ecg[i-1] < ecg[i] > ecg[i+1]:
19                  peaks.append(i)
20                  i += 100
21              else:
22                  i += 1
23          return np.array(peaks)
24
25
26  def getIntervals(ecg):
27          """Вычисление интервалов между R-зубцами"""
28          peaks = getPeaks(ecg)
29          intervals = np.zeros(len(peaks)-1)
30          for i in range(len(intervals)):
31              intervals[i] = peaks[i+1] - peaks[i]
32          return intervals
33
34
35  def getHR(ecg):
36          """Вычисление ЧСС по индексам пиков. Вычисляется по крайним
37          пикам в подаваемом в качестве аргумента сигнале"""
38          peaks = getPeaks(ecg)
39          hr = 60/((peaks[-1] - peaks[1])/hz/(len(peaks)-1))
40          return hr
41
42
43  # Функции для работы с КГР
44  def getDerivative(vector):
45          """Вычисляет производную сигнала"""
46          return vector[1:]-vector[:-1]
47
48
49  def stabilize(vector, begin, end):
50          """Убирает наклон графика КГР"""
51          vector[end+1:] -= vector[end]-vector[begin]
52          vector[begin:end+1] -= np.linspace(0, vector[end]-vector[begin],
53                                             end-begin+1)
54
55
56  def integrate(vector, const=0, thold=None):
57          """Интегрирует сигнал. vector - массив производных некого сигнала,
58          const - его начальное значение. Отсекает все производные, превышающие
59          по модулю пороговое значение thold."""
60          ans = np.zeros(len(vector)+1)
61          ans[0] = const
62          if thold is not None:
63              absVec = np.abs(vector)
64              if absVec[0] > thold:
65                  vector[0] = 0.0
66              for i in range(1, len(vector)):
67                  if absVec[i] > thold:
68                      vector[i] = vector[i-1]
69          for i, der in enumerate(vector):
70              ans[i+1] = ans[i] + der
71          return ans
72
73
74  def getLocalMaxsGSR(gsr):
```

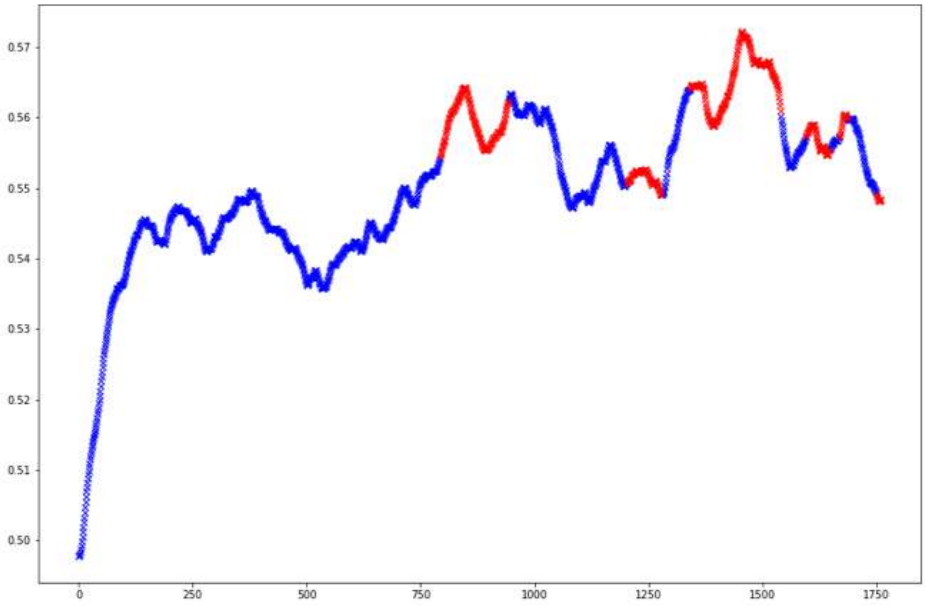
```
75          """Возвращает индексы локальных максимумов сигнала КГР"""
76          der = getDerivative(gsr)
77          i = 10
78          maxs = []
79          while i < len(der) - 10:
80              if der[i-10] < der[i] > der[i+10] and der[i]-der[i+10] > 1e-12\
81                                               and der[i]-der[i-10] > 1e-12:
82                  maxs.append(i)
83                  i += 249
84              i += 1
85          maxs = np.array(maxs)
86          return maxs
87
88
89  def getClosestInterval(index, intervalsDict):
90          """Возвращает ближайший к index временной интервал между
91          локальными максимумами производной сигнала КГР"""
92          closestIndex = min(intervalsDict.keys(), key=lambda x: abs(x-index))
93          return intervalsDict[closestIndex]
94
95
96  def processTestGSR(df):
97          """Обрабатывает GSR из тестовых данных: фильтрует производные,
98          сглаживает выборосы"""
99          der1 = getDerivative(df.gsr.values)
100         der2 = getDerivative(der1)
101         der1 = integrate(der2, thold=0.00000025)
102         testGSR = integrate(der1, df.gsr.values[0])
103         maxs = getLocalMaxsGSR(testGSR)
104         der = getDerivative(maxs)
105         for i in range(70, 200):
106             if der[i] < 11000:
107                 der[i] = der[i-1]
108         for i in range(250, len(der)):
109             if der[i] < 10000:
110                 der[i] = der[i-1]
111         intervalsDict = dict(zip(maxs[1:], der))
112         return intervalsDict
113
114
115 def processTrainGSR(df):
116         """Обрабатывает GSR из обучающих данных: фильтрует производные,
117         сглаживает выборосы"""
118         der1 = getDerivative(df.gsr.values)
119         der2 = getDerivative(der1)
120         der1 = integrate(der2, thold=0.00000025)
121         trainGSR = integrate(der1, df.gsr.values[0])
122         stabilize(trainGSR, 1820500, len(trainGSR)-1)
123         maxs = getLocalMaxsGSR(trainGSR)
124         der = getDerivative(maxs)
125         for i in range(120, 200):
126             if der[i] < 4000:
127                 der[i] = der[i-1]
128         for i in range(420, 450):
129             if der[i] < 4000:
130                 der[i] = der[i-1]
131         for i in range(470, 500):
132             if der[i] < 6000:
133                 der[i] = der[i-1]
134         for i in range(500, 700):
```

```
135             if der[i] < 6000:
136                 der[i] = der[i-1]
137         intervalsDict = dict(zip(maxs[1:], der))
138         return intervalsDict
139 
140 
141 # Обработка ЭЭГ
142 
143 
144 def compl(sig, coef=0.95):
145         """Комплементарный фильтр с коэффициентом coef.
146         Возвращает сигнал после фильтрации."""
147         for i in range(1, len(sig)):
148             sig[i] = coef * sig[i-1] + (1-coef) * sig[i]
149         return sig
150 
151 
152 def med(sig, coef=15):
153         """Возвращает массив длины len(sig)//coef, содержащий
154         медианный значения отрезков sig длиной coef."""
155         ret = np.zeros(len(sig)//coef)
156         for i in range(len(ret)):
157             ret[i] = np.average(sig[i*coef:(i+1)*coef])
158         return ret
159 
160 
161 def getAlBeta(eeg, chunkSize=15*hz):
162         """Вычисляет уровни альфа- и бета-ритмов в сигнале.
163         Вычисляется для chunkSize числа точек."""
164         nChunks = eeg.shape[0]//chunkSize
165         alphas = np.zeros(nChunks)
166         betas = np.zeros(nChunks)
167         coef = chunkSize // hz
168         for i in range(nChunks):
169             chunk = eeg[i*chunkSize:(i+1)*chunkSize]
170             spec = np.fft.fft(chunk)
171             spec = np.abs(spec)
172             alphas[i] = sum(spec[8*coef:13*coef+1])
173             betas[i] = sum(spec[15*coef:30*coef+1])
174         return pd.DataFrame({'alpha': alphas,
175                              'beta': betas
176                              })
177 
178 
179 def getAlphaToBetaSmooth(eeg, coef=15):
180         """Вычисляет отношение альфа-ритма к бета- в сигнале ЭЭГ.
181         После получение уровней альфа- и бета-ритмов сглаживает их.
182         Изначально уровни вычисляются во временном окне равном 1 с.
183         Параметр coef определяет число последовательных значений
184         уровней, по которым считается медианное значение."""
185         new = getAlBeta(eeg, chunkSize=hz)
186         alpha = new.alpha.values
187         beta = new.beta.values
188         alpha = compl(alpha)
189         beta = compl(beta)
190         alpha = med(alpha, coef)
191         beta = med(beta, coef)
192         alpha = compl(alpha)
193         beta = compl(beta)
194         alphaToBeta = compl(alpha/beta)
```

```
        return alphaToBeta


def getFeatures(df, chunkSize=15*hz, gsrProcess=processTrainGSR):
        """Генерируем вектор признаков для массива данных с датчиков."""
        # Вычисляем ЧСС
        hrs = np.zeros(df.shape[0] // chunkSize)
        # Вычисляем параметры КГР
        intervalsGSR = np.zeros(df.shape[0] // chunkSize)
        GSRDict = gsrProcess(df)
        for i in range(len(hrs)):
            hrs[i] = getHR(df.ecg[i*chunkSize:(i+1)*chunkSize].values)
            intervalsGSR[i] = getClosestInterval(i*chunkSize, GSRDict)
        # Вычисляем соотношение альфа- к бета-
        alphaToBeta = getAlphaToBetaSmooth(df.eeg.values)
        return pd.DataFrame({'HR': hrs,
                             'GSRInterval': intervalsGSR,
                             'albeta': alphaToBeta})

# Записываем вычисленные для обучающих данных признаки в X_train,
# верные для обучающих данных ответы в y_train, а вычисленные для
# тестовых данных признаки - в X_test
X_train = getFeatures(traindf, gsrProcess=processTrainGSR)
y_train = traindf.target[::15*hz].values
X_test = getFeatures(testdf, gsrProcess=processTestGSR)

# Обучаем модель на данных из train.csv и предсказываем значения
# из столбца target для test.csv. Полученные результаты записываем
# в myAnswer.csv, которые можно загрузить на kaggle.com в качестве
# ответа
svc = SVC(C=10, kernel='linear').fit(X_train, y_train)
answerDF = pd.DataFrame({'ID': testdf.ID.values[::15*hz],
                             'target': svc.predict(X_test)
                            })
answerDF.to_csv('myAnswer.csv', index=False)
```

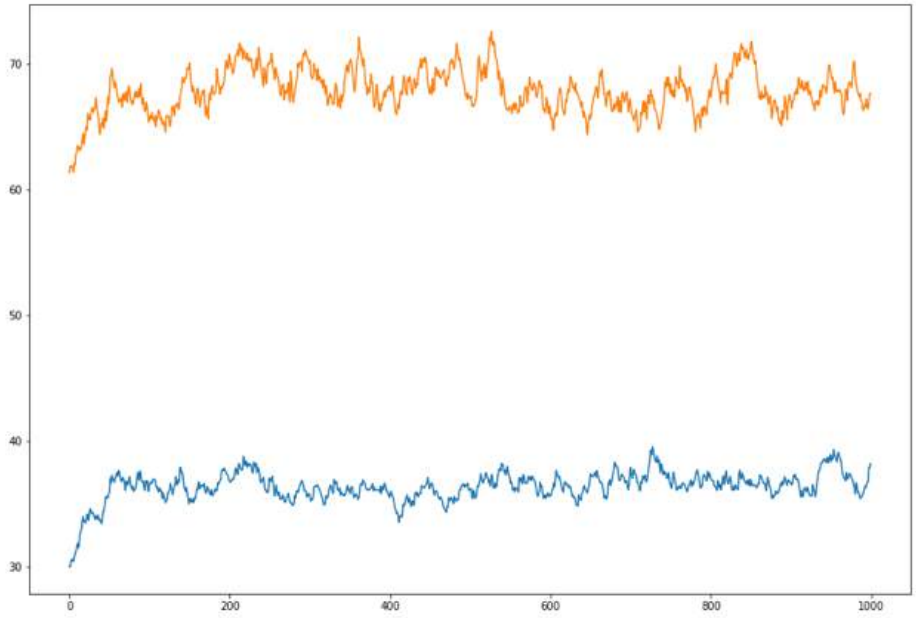
Анализ ЭЭГ

```
    # Построив график отношения альфа-ритма к бета-ритму в сигнале,
    # видим корреляцию между состоянием водителя и отношением альфа-
    # к бета-ритму.

    plt.figure(figsize=(15,10))
    alphaToBeta = getAlphaToBetaSmooth(traindf.eeg.values)
    df = pd.DataFrame({'aTob': alphaToBeta,
                       'target': traindf.target.values[::15*hz]
                       })
    plt.plot(df[df.target==1].aTob, 'bx')
    plt.plot(df[df.target==0].aTob, 'rx')
```



(На графике по оси X – номер временного отреза, по оси У – отношение уровня альфа- к бета- ритму в сигнале)

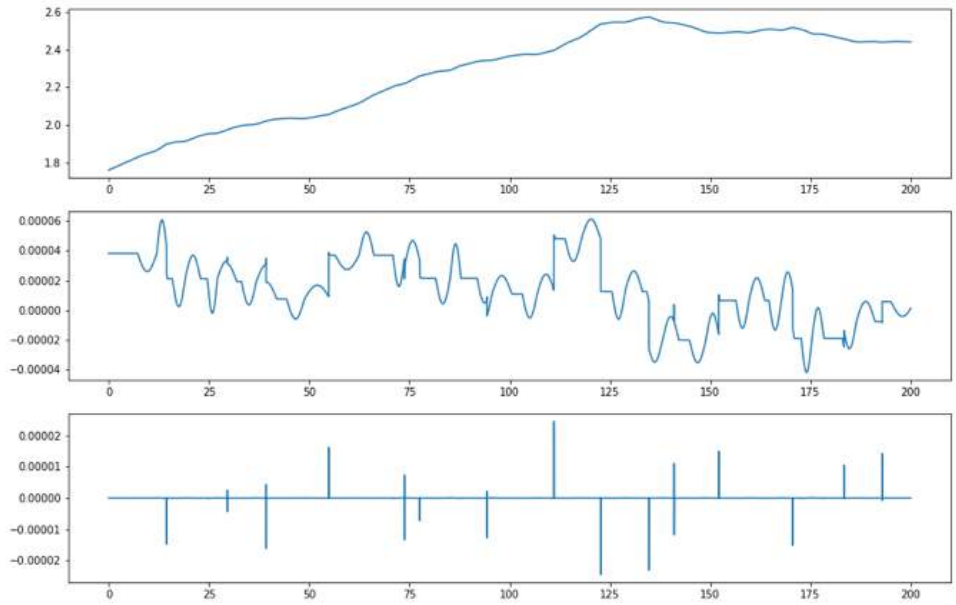
```
    # Вычисляем уровни альфа- и бета- ритмов в сигнале, сглаживаем их.
    # Мы видим что уровни альфа- и бета- изменяются с некоторой периодичностью

    albeta = getAlBeta(traindf.eeg.values, chunkSize=hz)
    plt.figure(figsize=(15,10))
    alpha = albeta.alpha.values
    beta = albeta.beta.values
    plt.plot(compl(alpha[:1000], 0.95))
    plt.plot(compl(beta[:1000], 0.95))
```



(На графике по оси X – время в секундах, по оси У – уровень ритма. Оранжевый график – уровень бета-ритма, Синий – уровень альфа-ритма).

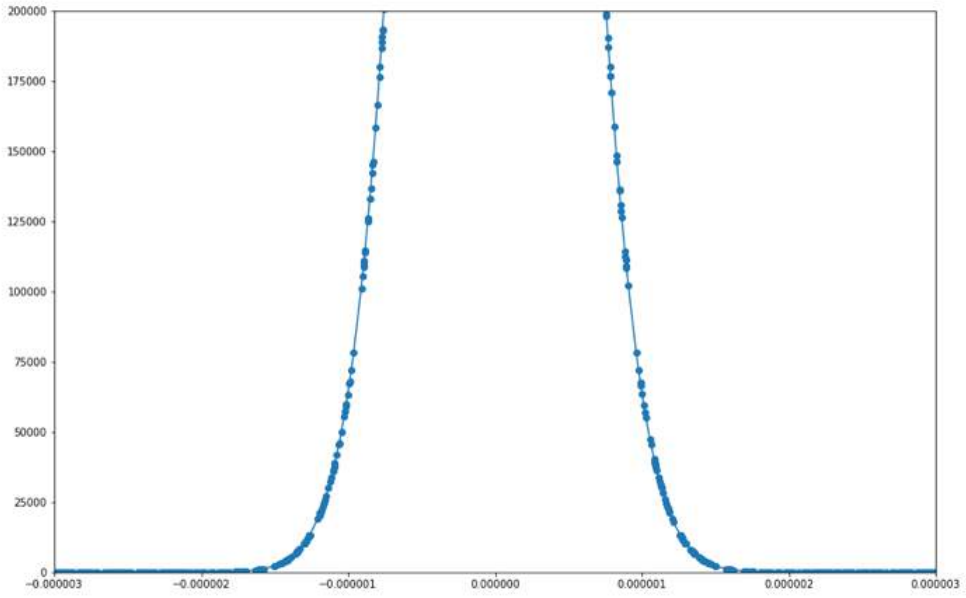
Анализ КГР

```
# Запишем КГР в отдельный массив
gsr = traindf.gsr.values

# Построим график сигнала, а также графики вторых и первых производных
# Мы видим аномальные скачки вторых производных, от которых следует избавиться
plt.figure(figsize=(15,10))
orig = gsr[:50000]
der1 = getDerivative(orig)
der2 = getDerivative(der1)
plt.subplot(3,1,1)
plt.plot(np.linspace(0,50000/250,50000),orig)
plt.subplot(3,1,2)
plt.plot(np.linspace(0,50000/250,49999),der1)
plt.subplot(3,1,3)
plt.plot(np.linspace(0,50000/250,49998),der2)
```



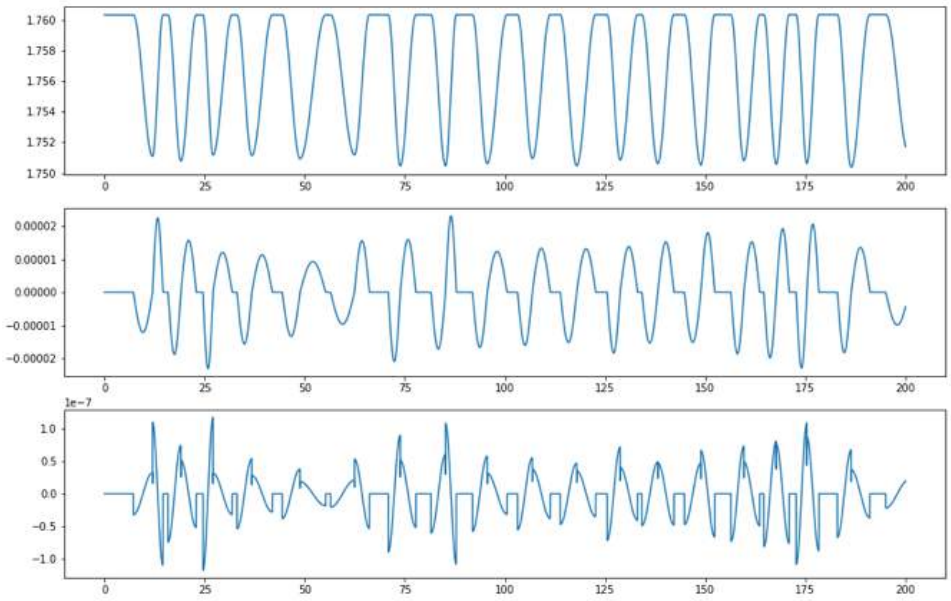
(на первом графике: по оси X — время в секундах, по оси У — значение КГР в Вольтах; на втором графике: по оси X — время в секундах, по оси У — первая производная КГР; на третьем графике: по оси X — время в секундах, по оси У — вторая производная КГР).

```
    import scipy.stats as stats

    # Строим график распределения значений вторых производных. Видим что пик
    # находится в диапазоне [-0.00000025, 0.00000025]

    h = sorted(getDerivative(getDerivative(gsr[:5000000])))

    fit = stats.norm.pdf(h, np.mean(h), np.std(h))
    plt.figure(figsize=(15,10))
    plt.xlim([-0.000003,0.000003])
    plt.ylim([0,200000])
    plt.plot(h,fit,'-o')
```



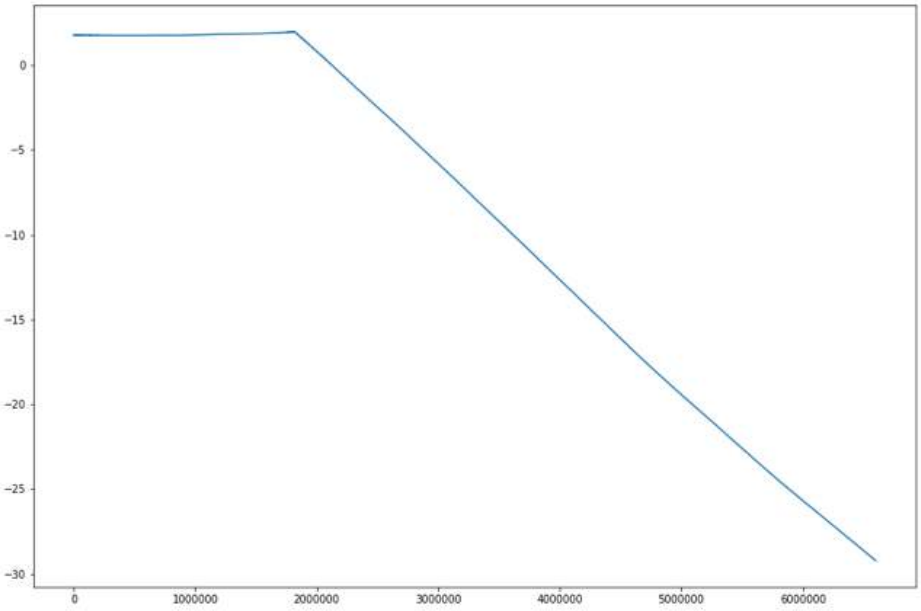
(На графике по оси X – значение второй производной , по оси У – сколько раз встретилось данное значение).

```
    # На участке сигнала находим вторые производные и восстанавливаем
    # по ним сигнал, отсекая производные превышающие по модулю 0.00000025.
    # Видим, что восстановленный сигнал имеет чёткие локальные минимумы.
    plt.figure(figsize=(15,10))
    orig = gsr[:50000]
    der1 = getDerivative(orig)
    der2 = getDerivative(der1)
    der1 = integrate(der2, thold=0.00000025)
    orig = integrate(der1, orig[0])
    plt.subplot(3,1,1)
    plt.plot(np.linspace(0,50000/250,50000),orig)
    plt.subplot(3,1,2)
    plt.plot(np.linspace(0,50000/250,49999),der1)
    plt.subplot(3,1,3)
    plt.plot(np.linspace(0,50000/250,49998),der2)
```



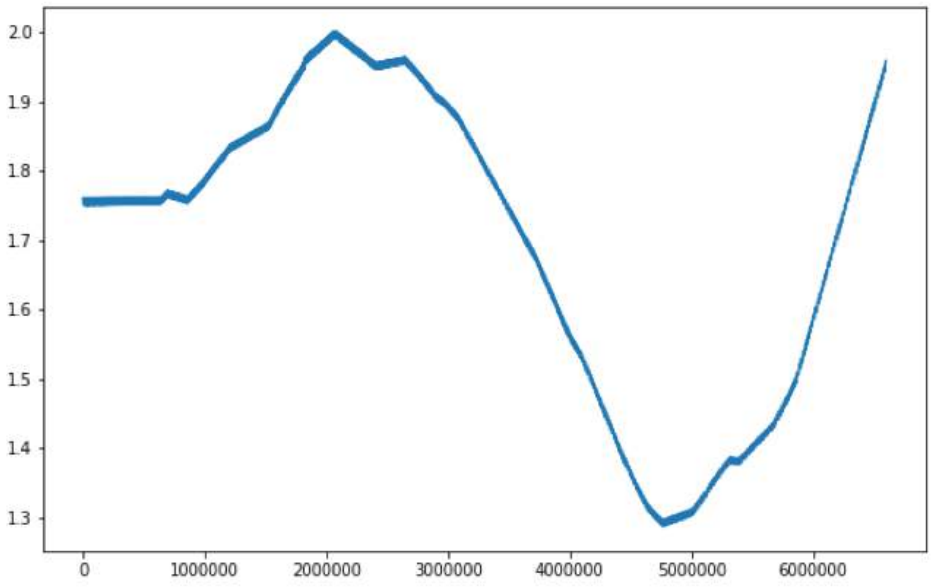
(на первом графике: по оси X – время в секундах, по оси У - значение КГР в Вольтах; на втором графике: по оси X – время в секундах, по оси У – первая производная КГР; на третьем графике: по оси X – время в секундах, по оси У – вторая производная КГР).

```
    # Очищаем весь сигнал так же, как очищали отрезок сигнала
    plt.figure(figsize=(15,10))
    der1 = getDerivative(gsr)
    der2 = getDerivative(der1)
    der1 = integrate(der2, thold=0.00000025)
    cleanGSR = integrate(der1, gsr[0])
    plt.plot(cleanGSR)
```



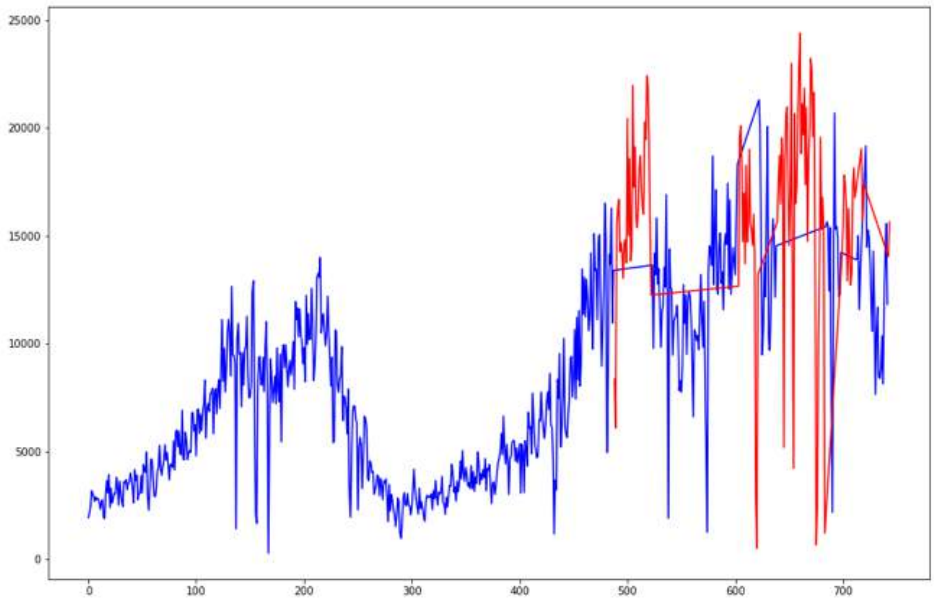
(На графике: зависимость сигнала КГР (восстановленного после очистки второй производной) от номера отсчета)

```
1       # Т.к. на очищенном сигнале имеется явный артефакт, выпрямляем сигнал.
2       plt.figure(figsize=(10,6))
3       stabilize(cleanGSR, 1820500, len(cleanGSR)-1)
4       plt.plot(cleanGSR)
```



(На графике: зависимость сигнала КГР от номера отсчета)

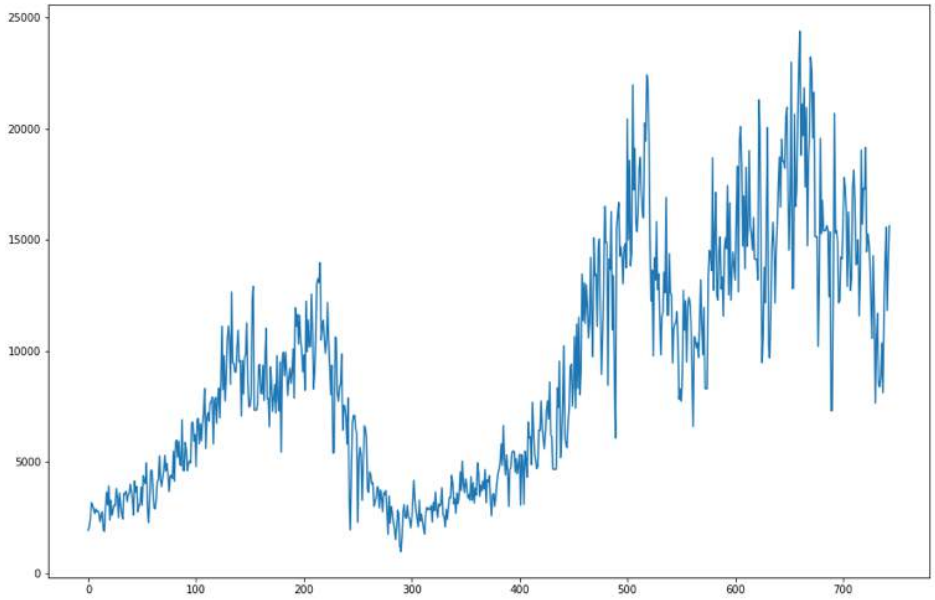
```
    # Находим индексы локальных максимумов КГР, считаем интервалы между ними
    # и записываем их в словарь.
    maxs = getLocalMaxsGSR(cleanGSR)
    intervalsDict = dict(zip(maxs[1:], getDerivative(maxs)))
    def getClosestInterval(index, intarvalsDict):
        """Возвращает ближайший к index временной интервал между
        локальными максимумами производной сигнала КГР"""
        closestIndex = min(intarvalsDict.keys(), key=lambda x: abs(x-index))
        return intervalsDict[closestIndex]

    # Построим график зависимости длительности временного интервала от
    # его порядкового номера. Красным цветом отметим точки, соответствующие
    # target==0. Мы видим что интервалы увеличиваются со снижением концентрации.
    # На графике видны выборосы, которые для большей точности следует убрать вручную.
    df = pd.DataFrame({'intervals': list(intervalsDict.values()),
                       'target': [traindf.target.values[i] for i in intervalsDict.keys()]})
    plt.figure(figsize=(15,10))
    plt.plot(df.intervals[df.target==1], 'b')
    plt.plot(df.intervals[df.target==0], 'r')
```



(На графике: зависимость интервала между локальными минимумами КГР (в точках, т.е. чтобы получить время необходимо поделить ее на частоту оцифровки) от порядкового номера интервала).

```
    # Тестируем код для очистки КГР от выбросов
    der1 = getDerivative(traindf.gsr.values)
    der2 = getDerivative(der1)
    der1 = integrate(der2, thold=0.00000025)
    trainGSR = integrate(der1, traindf.gsr.values[0])
    stabilize(trainGSR, 1820500, len(trainGSR)-1)
    maxs = getLocalMaxsGSR(trainGSR)
    der = getDerivative(maxs)
    for i in range(120,200):
```

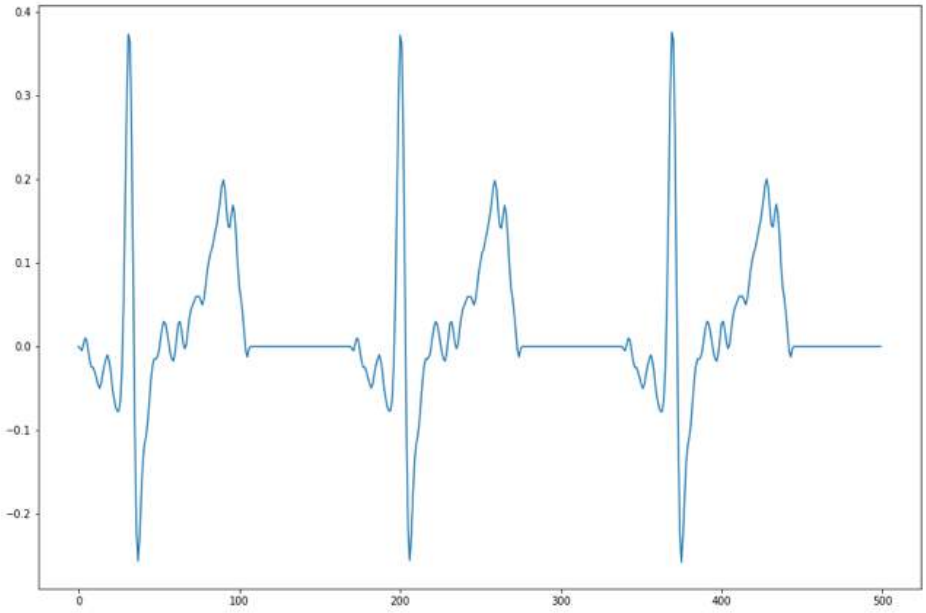
```
        if der[i] < 4000:
            der[i] = der[i-1]
    for i in range(420,450):
        if der[i] < 4000:
            der[i] = der[i-1]
    for i in range(470,500):
        if der[i] < 6000:
            der[i] = der[i-1]
    for i in range(500, 700):
        if der[i] < 6000:
            der[i] = der[i-1]
    intervalsDict = dict(zip(maxs[1:], der))
    plt.plot(intervalsDict.values())
```



(На графике: зависимость интервала между локальными минимумами КГР (в точках, т.е. чтобы получить время необходимо поделить ее на частоту оцифровки) от порядкового номера интервала).

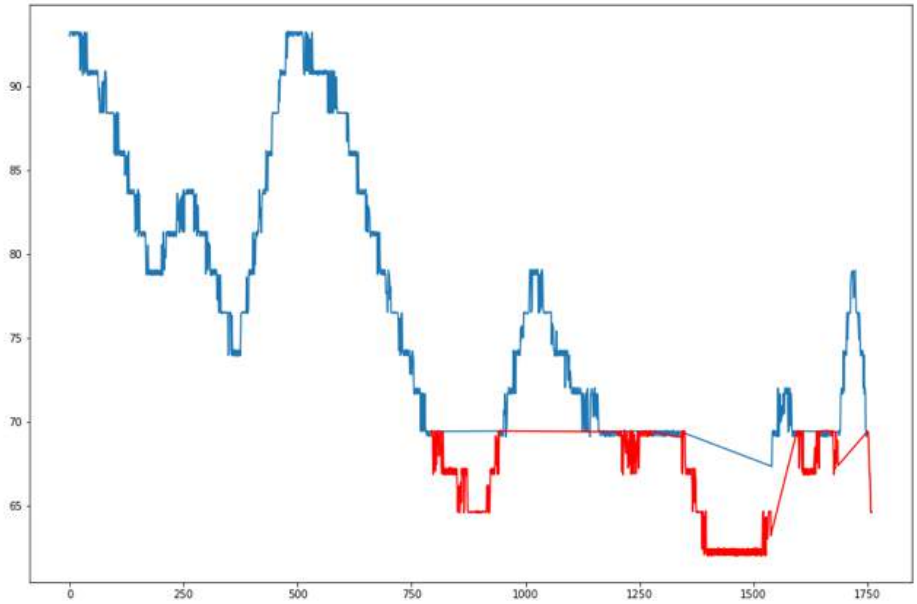
Анализ ЭКГ

```
    # Сохраним сигнал в отдельный массив
    ecg = traindf['ecg'].values
    # визуализируем участок сигнала
    plt.figure(figsize=(15,10))
    plt.plot(ecg[:500])
```



(На графике: по оси X – номер отсчета, по оси У – значение сигнала ЭКГ).

```
# Построим график зависимости ЧСС от номера отрезка данных. Участки,
# соответствующие состоянию со сниженным уровнем внимания отметим
# красным. Как видно, в таки моменты мы наблюдаем наиболее низкую ЧСС.

hr = pd.DataFrame({'hr': [getHR(ecg[i*15*hz:(i+1)*15*hz])
    for i in range(len(ecg)//(15*hz))],
        'target': [tar for tar in traindf.target.values[::15*hz]]})
plt.figure(figsize=(15,10))
plt.plot(hr.hr[hr.target==1])
plt.plot(hr.hr[hr.target==0], 'r')
```



(На графике: по оси X – номер временного отрезка, по оси У – значение частоты сердечных сокращений).

Задача 6.1.2. (20 баллов)

Напишите программу, которая будет выполнять следующие действия:

1. Считывать видео с веб-камеры в режиме реального времени и выводить его на экран. На изображении при этом лицо должно быть выделено прямоугольником. В случае, если человек, на которого направлена камера, дольше, чем на 1 секунду:
   - закрыл глаза;
   - смотрит на свой мобильный, который лежит у него на колене;
   - смотрит на автомагнитолу,

   необходимо активировать пьезоэлемент, подключенный к плате Arduino. При этом пьезоэлемент не должен работать дольше трех секунд за раз. Примеры вышеперечисленных ситуаций представлены на скриншотах в Приложении 3. Водитель во время движения может смотреть в зеркала заднего и бокового вида, что не должно вызывать активацию пьезоэлемента.
2. Одновременно с этим (вместе с п. 2.1 и 2.2) необходимо в режиме реального времени выводить на изображение значение напряжения на 3 аналоговом входе платы Arduino, к которому подключен потенциометр (см. схему в Приложении 4).

Система оценки

Решение этой задачи будет оцениваться путем запуска контрольного видео (см. отрывок видео `https://clck.ru/FVTCA`), на котором водитель определенное количе-

ство раз будет повторять вышеописанные ситуации. Отрывок видео имеет продолжительность 5 секунд, следует откалибровать скорость воспроизведения видео вашей программой по нему.

Баллы за решение данной задачи будут выставляться по следующей формуле:

$$Result = (\frac{K_{ok}}{K_{sit}}^{-}0.5 \cdot K_{wrong}) \cdot (1 - D),$$

где: K_ok — количество верных срабатываний (т.е. пьезоэлемент просигналил в те моменты, когда водитель находился в одной из ситуаций, описанных в п. 2.2 данной задачи); $_{sit}$ — количество ситуаций на видео, описанных в п. 2.2 данной задачи. K_{wrong} — количество неверных срабатываний пьезоэлемента после трех неверных срабатываний. Например, если было два неверных срабатывания, то $K_{wrong} = 0$. Если было 4 неверных срабатывания, то $K_{wrong} = 1$. D — коэффициент дисконтирования. Значение коэффициента дисконтирования в зависимости от даты сдачи представлен в таблице ниже:

Таблица 2 — Значение коэффициента дисконтирования

Дата сдачи задания	Коэффициент дисконтирования (D)
19 марта	0%
20 марта	20%
21 марта	30%
22 марта	50%

В случае, если $\frac{K_{wrong}}{K_{sit}} \cdot 20 \leqslant 0.5 \cdot K_{wrong}$, то за задачу ставится 0 баллов.

Например: Если вы сдали задачу 21 марта полностью правильно, то вы получите $20 \cdot 0.7 = 14$ баллов за данную задачу. Если вы сдали задачу 22 марта и вы набрали при сдаче 10 баллов, то с учетом дисконтирования вы получите: $10 \cdot 0.5 = 5$ баллов за задачу.

Число попыток: 3 попытки за всё время сдачи данного задания.

Срок сдачи: 13:00, 22 марта 2019 г.

Решение

В данной задаче командам необходимо было разобраться, как выводить оцифрованные значения с третьего аналогового входа платы Arduino поверх изображения с камеры, а также реализовать распознавание различных положений водителя за рулем.

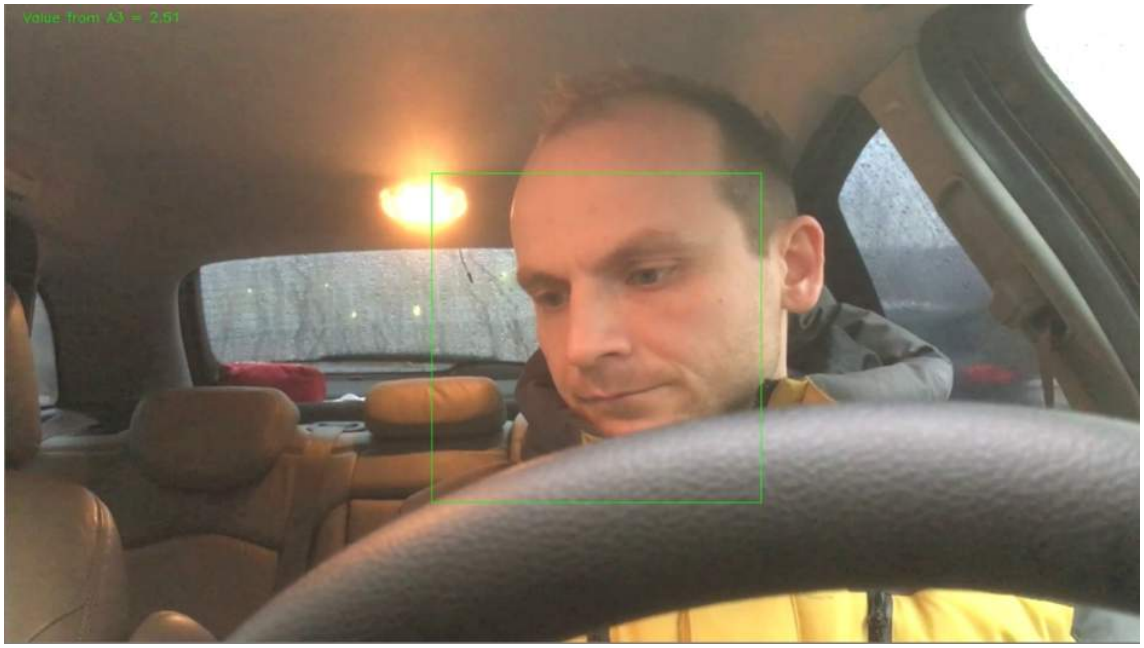
Для проверки данной задачи использовалось контрольное видео (участники в первый раз его видели при сдаче): `https://drive.google.com/file/d/17EHTJLTJGGk1VVMJ3AfoqtjRiMEvCD6U/view?usp=sharing` (сокращенная ссылка: `https://clck.ru/FVfGL`)

На данном видео были представлены следующие ситуации (1 – «не опасная» ситуация, 2,3,4 – «опасные» ситуации):

Момент времени на контрольном видео	Тип ситуации	Описание ситуации
00:00-00:06	1	Смотрит прямо/в зеркало
00:06-00:10	3	Смотрит на панель
00:10-00:17	1	Смотрит прямо/в зеркало
00:17-00:20	2	Закрыты глаза
00:20-00:27	1	Смотрит прямо/в зеркало
00:28-00:32	2	Закрыты глаза
00:32-00:37	1	Смотрит прямо/в зеркало
00:37-00:43	3	Смотрит на панель
00:43-00:48	1	Смотрит прямо/в зеркало
00:48-00:53	2	Закрыты глаза
00:53-00:58	1	Смотрит прямо/в зеркало
00:59-01:04	4	Смотрит на колено
01:04-01:13	1	Смотрит прямо/в зеркало
01:13-01:17	2	Закрыты глаза
01:17-01:25	1	Смотрит прямо/в зеркало
01:25-01:32	3, 2	Смотрит на панель, затем закрывает глаза
01:38-01:46	1	Смотрит прямо/в зеркало
01:46-01:53	3	Смотрит на панель
01:54-01:59	1	Смотрит прямо/в зеркало
02:00-02:03	2	Закрыты глаза

За каждую верно определенную «опасную» ситуацию можно было получить по 2 балла, в сумме не более 20 баллов.

Пример скриншота работающей программы для данной задачи:



Скетч для платы Arduino

Передает результаты оцифровки третьего аналогового входа по Serial и принимает команды на включение и выключение пьезоэлемента:

```
const int buzzerPin = 11,
          potPin = 3;

void setup() {
    Serial.begin(115200);
    pinMode(buzzerPin, OUTPUT);
}

void loop() {
    if (Serial.available()) {
        int cmd = Serial.read();
        if (cmd == 49)
            tone(buzzerPin, 500);
        if (cmd == 48)
            noTone(buzzerPin);
        int potVal = analogRead(potPin) / 4;
        Serial.write(potVal);
    }
    delay(1);
}
```

Программа для обработки видео

```
import cv2
import dlib
import numpy as np
from numpy.linalg import norm
from imutils import face_utils
from serial import Serial


class DriverControl:
    """Класс для определения направления поворота головы. Подробности по адресу
    www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/"""
    def __init__(self, size):
        """Аргументы:
            size - разрешение изображения
        Переменные экземпляра класса:
            maxFrames - Число последовательных кадров на видео, для которых
                        должны выполняться условия срабатывания
                        пьезоэлемента, прежде чем его активировать.
            earThold -  порог ширины открытия глаз выше ниже которого глаза
                        считаются закрытыми
        """
        self.size = size
        self.model_points = np.array([
                                (0.0, 0.0, 0.0),
                                (0.0, -330.0, -65.0),
                                (-225.0, 170.0, -135.0),
                                (225.0, 170.0, -135.0),
                                (-150.0, -150.0, -125.0),
                                (150.0, -150.0, -125.0)
                            ])
```

```
31              self.focal_length = size[1]
32              self.center = (size[1]/2, size[0]/2)
33              self.camera_matrix = np.array([
34                                      [self.focal_length, 0, self.center[0]],
35                                      [0, self.focal_length, self.center[1]],
36                                      [0, 0, 1]
37                                  ], dtype='double')
38              self.dist_coeffs = np.zeros((4, 1))
39              self.nFrames = 0
40              self.maxFrames = 16
41              self.earThold = 0.3
42
43          def getNosePoint(self, shape):
44              """Вычисляет направление поворота головы.
45              Аргументы:
46                  shape - массив ключевых точек лица
47              Возвращаемые значения:
48                  nosePoint - координаты конца вектора, указывающего направление
49                              головы. Начало вектора - ключевая точка лица
50                              с индексом 30.
51              """
52              if shape is None:
53                  return (0, 0)
54              image_points = np.array([
55                                          shape[30],          # Nose tip
56                                          shape[8],           # Chin
57                                          shape[45],          # Left eye left corner
58                                          shape[36],          # Right eye right corne
59                                          shape[54],          # Left Mouth corner
60                                          shape[48]           # Right mouth corner
61                                      ], dtype="double")
62
63              (success, rotation_vector,
64              translation_vector) = cv2.solvePnP(self.model_points,
65                                                 image_points,
66                                                 self.camera_matrix,
67                                                 self.dist_coeffs,
68                                                 flags=cv2.SOLVEPNP_ITERATIVE)
69              (nose_end_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0,
70                                                                          1000.0)]),
71                                                               rotation_vector,
72                                                               translation_vector,
73                                                               self.camera_matrix,
74                                                               self.dist_coeffs)
75              nosePoint = (int(nose_end_point2D[0][0][0]),
76                           int(nose_end_point2D[0][0][1]))
77              return nosePoint
78
79          def getEAR(self, shape):
80              """Вычисляет ширину открытия глаз.
81              Аргументы:
82                  shape - массив ключевых точек лица
83              """
84              leftH = (norm(shape[37]-shape[41]) + norm(shape[38]-shape[40])) / 2
85              leftW = norm(shape[36]-shape[39])
86              leftEAR = leftH / leftW
87              rightH = (norm(shape[43]-shape[47]) + norm(shape[44]-shape[46])) / 2
88              rightW = norm(shape[42]-shape[45])
89              rightEAR = rightH / rightW
90              EAR = (leftEAR + rightEAR) / 2
```

```
91              return EAR
92
93          def checkDriver(self, shape):
94              """Возвращает True если человек дольше чем на maxFrames кадров закрыл
95              глаза, смотрит на свой мобильный, который лежит у него на колене или
96              смотрит на автомагнитолу; иначе - False.
97              Аргументы:
98                  shape - массив ключевых точек лица
99              """
100             # Получаем координату конца вектора направления поворота головы
101             nosePoint = self.getNosePoint(shape)
102             # Вычисляем ширину открытия глаз
103             ear = self.getEAR(shape)
104             # Проверяем не ниже ли порога ширина открытия глаз и не опустился ли
105             # конец вектора направления поворота головы ниже верхней губы
106             if ear < self.earThold or nosePoint[1] > shape[62][1]:
107                 self.nFrames += 1
108                 if self.nFrames >= self.maxFrames:
109                     return False
110             else:
111                 self.nFrames = 0
112                 return True
113
114
115     class LandmarkDetector:
116         """Класс для поиска ключевых точек лица на изображении"""
117         def __init__(self, modelPath='shape_predictor_68_face_landmarks.dat',
118                      showLandmarks=False):
119             """Аргументы:
120                 modelPath - путь к файлу с предобученной моделью для поиска
121                             ключевых точек.
122                 showLandmarks - отмечать ли ключевые точки на лице.
123             """
124             self.detector = dlib.get_frontal_face_detector()
125             self.predictor = dlib.shape_predictor(modelPath)
126             self.showLandmarks = showLandmarks
127
128         def getLandmarks(self, image):
129             """Возвращает массив с координатами ключевых точек.
130             Аргументы:
131                 image - изображение для поиска ключевых точек
132             """
133             gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
134             rects = self.detector(gray, 0)
135             shape = None
136             for rect in rects:
137                 (bX, bY, bW, bH) = face_utils.rect_to_bb(rect)
138                 cv2.rectangle(image, (bX, bY), (bX + bW, bY + bH),
139                               (0, 255, 0), 1)
140                 shape = self.predictor(gray, rect)
141                 shape = face_utils.shape_to_np(shape)
142                 if self.showLandmarks is True:
143                     for (x, y) in shape:
144                         cv2.circle(image, (x, y), 1, (0, 0, 255), -1)
145             return shape
146
147
148     def launchVideo(videoSource, arduinoPort):
149         """Воспроизводит видео, подаёт команды на Arduino. Для завершения программы
150         нажать q.
```

```
151         Аргументы:
152             videoSource - источник видеопотока.
153             arduinoPort - COM-порт к которому подключается Arduino"""
154         landDetector = LandmarkDetector()
155         cap = cv2.VideoCapture(videoSource)
156         res = np.array((cap.get(4), cap.get(3)), dtype=int)
157         dc = DriverControl(res)
158         ser = Serial(arduinoPort, 115200, timeout=0)
159         voltageA3 = 0.0
160         while True:
161             # Считываем кадр
162             ret, frame = cap.read()
163             if ret:
164                 # Ищем ключевые точки лица
165                 shape = landDetector.getLandmarks(frame)
166                 if shape is not None:
167                     # Проверяем не выполняется ли одно из условий задачи.
168                     driverOK = dc.checkDriver(shape)
169                     if driverOK is False:
170                         # Если выполняется, шлём на Arduino символ "1"
171                         ser.write(b'1')
172                     else:
173                         ser.write(b'0')
174                 # Считываем с Arduino результат оцифровки сигнала с потенциометра
175                 pot_val = ser.read()
176                 if pot_val != b'':
177                     # Переводим значение АЦП в напряжение
178                     voltageA3 = int.from_bytes(pot_val, 'big') * 5.0 / 255
179                 # Выводим напряжение на изображение
180                 cv2.putText(frame, 'Value from A3 = {:.2f}'.format(voltageA3),
181                             (20, 20), cv2.FONT_HERSHEY_SIMPLEX,
182                             0.5, (0, 255, 0), 1)
183                 cv2.imshow("Output", frame)
184                 if cv2.waitKey(1) & 0xFF == ord('q'):
185                     break
186 
187         cap.release()
188         ser.close()
189         cv2.destroyAllWindows()
190 
191 
192     if __name__ == '__main__':
193         launchVideo('test.mp4', 'COM13')
```

Задача 6.1.3. (30 баллов)

Подзадача 1 (6 баллов)

Напишите программу (скетч) для платы Arduino, которая будет осуществлять запись данных от подключенных к плате Arduino сенсоров в последовательный порт согласно протоколу, описанному в Приложении 5, с частотой 250 Гц.

Система оценки

Ставится по 2 балла за каждый корректно выведенный сигнал в программе BiTronics Studio.

Число попыток: 3 попытки за всё время.

Срок сдачи: 22 марта (до конца рабочего дня, согласно расписанию).

Подзадача 2 (24 балла)

Напишите программу для ПК, которая в режиме реального времени будет выводить поверх изображения с веб-камеры следующие параметры сигналов с сенсоров:

- (10 баллов) отношение амплитуды альфа-ритма (8-13 Гц включительно) к амплитуде бета-ритма (15-30 Гц включительно) для сигнала ЭЭГ. Временное окно для вычисления амплитуд составляет 1 секунду;
- (10 баллов) число сердечных сокращений в минуту, вычисляемое по 10 последним ударам сердца;
- (4 балла) значение сигнала с модуля кожно-гальванической реакции (в вольтах).

Система оценки

Корректность расчета вышеперечисленных величин будет проверяться путём подключения к вашему компьютеру устройства, передающего сигналы с заранее известными организаторам параметрами. За каждый верно выведенный параметр ставятся баллы, указанные в скобках выше.

Число попыток: 3 попытки за всё время.

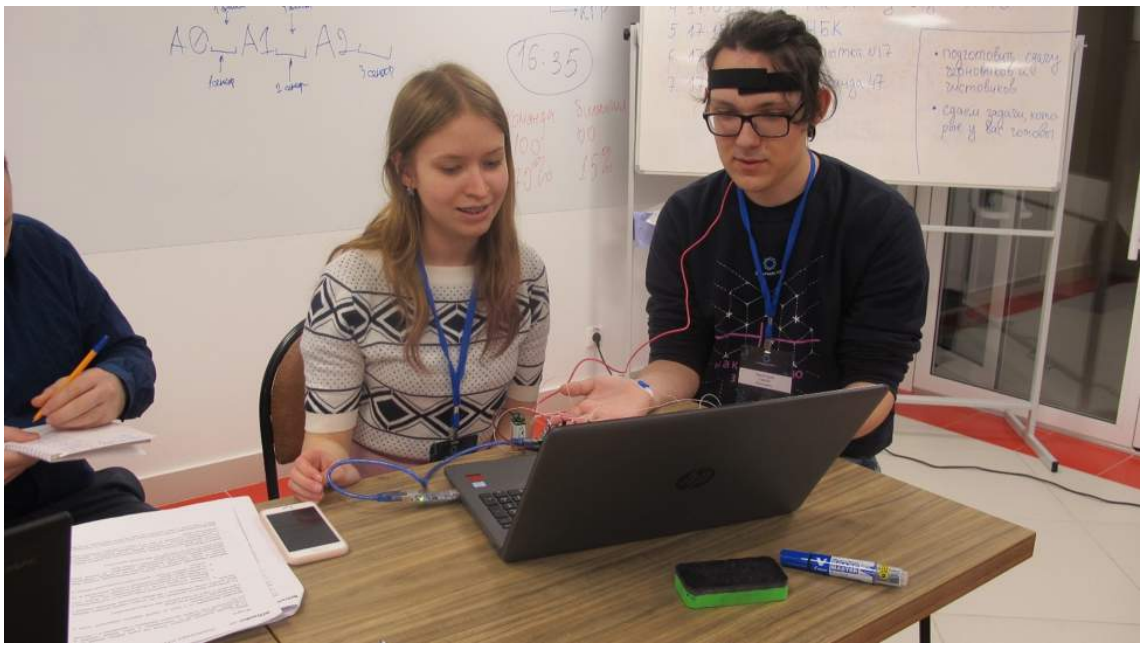
Срок сдачи: 22 марта (до конца рабочего дня, согласно расписанию).

Решение

Подзадача 1

В данной задаче участникам необходимо было отладить работу сенсоров биосигналов человека (в том числе правильно расположить электроды): электрокардиограмма, электроэнцефалограмма, кожно-гальваническая реакция, и получить не зашумленный сигнал. За каждый корректно выведенный сигнал в программе BiTronics Studio команды получали по 2 балла.

На фото ниже изображены сенсоры биосигналов человека на участниках одной из команд:



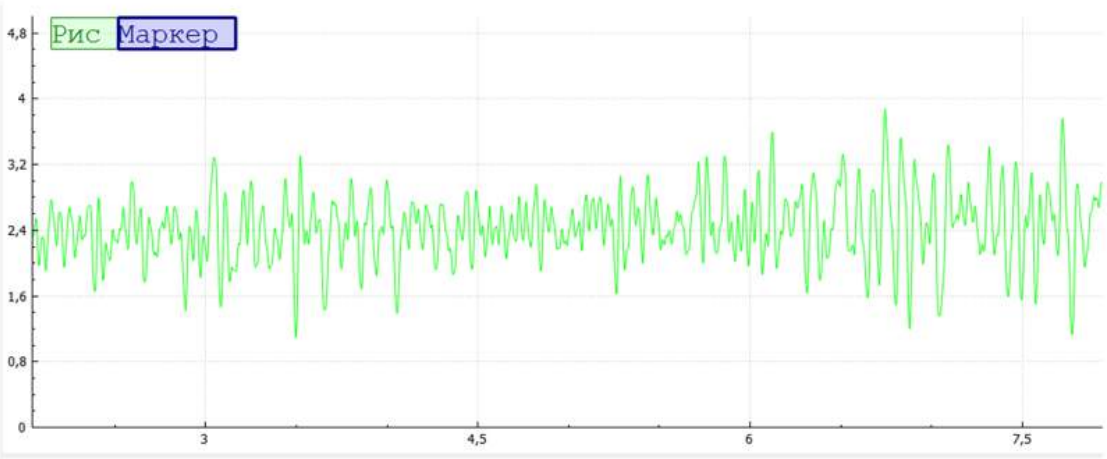
На голове закрепляется ободок с электродами, расположенные в затылочной части головы, который подключен к сенсору ЭЭГ через AUX-кабель. На запястьях закреплены электроды для регистрации сигнала ЭКГ (согласно первой схеме отведения). На пальцах закреплены электроды для считывания сигнала кожно-гальванической реакции.

Скетч для платы Arduino

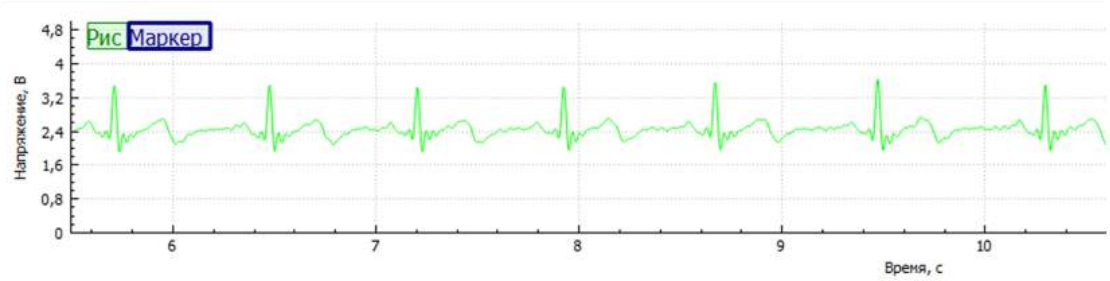
```
1     void setup() {
2         Serial.begin(115200);
3     }
4
5     void loop() {
6         Serial.write("A0");
7         Serial.write(analogRead(0)/4);
8         Serial.write("A1");
9         Serial.write(analogRead(1)/4);
10        Serial.write("A2");
11        Serial.write(analogRead(2)/4);
12        delay(4);
13    }
```

Пример получаемых сигналов:

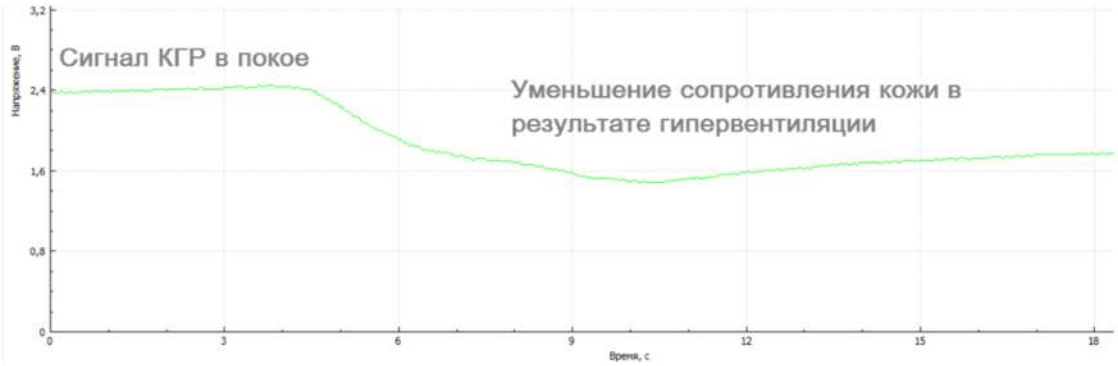
Сигнал, регистрируемый с одноканального сенсора электроэнцефалограммы (по оси X – временная шкала, секунды; по оси У – напряжение, мВ):



Сигнал, регистрируемый с сенсора электрокардиограммы (по схеме первого отведения):



Сигнал, регистрируемый с сенсора кожно-гальванической реакции:



Комментарий: при работе с данным сенсором КГР регистрируется сопротивление, а не проводимость.

Подзадача 2

В данной задаче необходимо было реализовать анализ биосигналов из задачи 3.1 и вывести поверх изображения с веб-камеры подсчитанные параметры.

Для проверки использовался генератор сигналов, с заранее известными параметрами для проверяющего. Для этого проверяющему необходимо было выполнить следующие действия:

1. Соединить два преобразователя USB-UART RX к TX, TX к RX, GND к GND с помощью проводов типа «мама-мама».
2. Запустить файл generator.py, например, в Spyder. Файлы Data и pack.bin должен быть в той же папке. Ссылка на файлы: https://clck.ru/FVfUv
3. Подключить ноутбук проверяющего к ноутбуку участника с помощью преобразователей USB-UART
4. При запуске генератор запрашивает номер COM-порта, на которые слать данные. Вводим номер порта на своём ноутбуке (т.е. если преобразователь на COM3, вводим просто цифру 3) и нажимаем «Enter».
5. Участник запускает свой код, указав в качестве порта для считывания данных тот, к которому подключён его преобразователь. В файле data.csv записаны данные ЭКГ, ЭЭГ и КГР (A0, A1, A2).

Программа участника должна вывести следующие значения параметров:

1. отношение амплитуды альфа-ритма (8-13 Гц включительно) к амплитуде бета-ритма (15-30 Гц включительно) для сигнала ЭЭГ. Временное окно для вычисления амплитуд составляет 1 секунду: 9.2 ($\pm$ 1). В скобках указана допустимая погрешность;
2. число сердечных сокращений в минуту, вычисляемое по 10 последним ударам сердца: - 60 ($\pm$ 1) ударов в минуту;
3. КГР - 2.5 ($\pm$ 0.05) Вольт.

Для платы Arduino используется тот же скетч, что и в подзадаче 3.1.

Для анализа биосигналов и подсчета параметров с дальнейшим выводом на изображение с веб-камеры использовалась следующая программа (одно из возможных решений):

```
from collections import deque
from serial import Serial
from struct import unpack
import numpy as np
import threading
import cv2

albeta, gsr, hr = 0, 0, 0
runReading = True


def equalize(ser):
    """Считывает данные с последовательного порта таким образом, чтобы после
    завершения работы equalize следующее сообщение начиналось с "A0"
    """
    buf = (None, None)
    a0 = (b'A', b'0')
    while buf != a0:
        buf = (buf[1], ser.read(1))
    ser.read(7)


def getAlBeta(eeg):
    """Возвращает соотношение суммы амплитуд компонент альфа-ритма к сумме
    амплитуд компонент бета-ритма. eeg - отрезок сигнала ЭЭГ длительностью 1 с
    """
```

```
27          spectrum = np.abs(np.fft.fft(eeg))
28          albeta = sum(spectrum[8:14]) / sum(spectrum[15:31])
29          return albeta
30
31
32      class Pulse():
33          """Класс для вычисления ЧСС"""
34          def __init__(self):
35              """Длину дека data в котором хранятся данные берём из расчёта поместить
36              в него 10 сердечных сокращений при ЧСС 40 ударов в минуту."""
37              self.data = deque(maxlen=15*250)
38
39          def getHR(self, ecg):
40              """ Вычисляет ЧСС"""
41              # Записываем в конец дека последние поступившие данные
42              self.data += ecg
43              # Порог для поиска вершины R-зубца
44              thold = 0.9 * (max(self.data) - min(self.data)) + min(self.data)
45              peaks = 0
46              # В first записываем индекс вершины первого R-пика, в last - десятого
47              last, first = None, None
48              i = len(self.data) - 10
49              # Пики ищем с конца массива, так как нас интересуют последние биения
50              while i > 10:
51                  i -= 1
52                  if self.data[i] > self.data[i+10] and \
53                  self.data[i] > self.data[i-10] and self.data[i] > thold:
54                      peaks += 1
55                      i -= 80
56                  if peaks == 1:
57                      last = i
58                  if peaks == 10:
59                      first = i
60              if (last is not None) and (first is not None):
61                  dt = (last - first) / 250 / 9
62                  hr = 60 / dt
63                  return hr
64              return 0
65
66
67      def collectData(portNumber):
68          """Вычисляет отношение альфа к бета, значение КГР и ЧСС
69          Аргументы:
70              portNumer - название COM-порта к которому подключена Arduino"""
71          global albeta, gsr, hr
72          with Serial(portNumber, 115200) as ser:
73              pulse = Pulse()
74              equalize(ser)
75              while runReading:
76                  chunk = ser.read(250 * 9)
77                  albeta = getAlBeta(unpack(b'250B', chunk[5::9]))
78                  gsr = chunk[8] * 5 / 255
79                  hr = pulse.getHR(unpack(b'250B', chunk[2::9]))
80
81
82      if __name__ == '__main__':
83          # Запускаем collectData в отдельном потоке
84          collector = threading.Thread(target=collectData, args=('COM10',))
85          collector.start()
86          # Запускаем считывание с камеры
```

```
87          cap = cv2.VideoCapture(0)
88          while True:
89              # Считываем кадр
90              ret, frame = cap.read()
91              if ret:
92                  # Выводим на изображение нужные нам параметры
93                  cv2.putText(frame, 'Alpha to Beta = {:.2f}'.format(albeta),
94                              (20, 20), cv2.FONT_HERSHEY_SIMPLEX,
95                              0.5, (0, 255, 0), 1)
96                  cv2.putText(frame, 'Heart rate = {:.2f} bpm'.format(hr), (20, 40),
97                              cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
98                  cv2.putText(frame, 'GSR level = {:.2f} V'.format(gsr), (20, 60),
99                              cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
100                 cv2.imshow('ONTI2019', frame)
101                 # При нажатии клавиши q завершаем работу программы
102                 if cv2.waitKey(1) & 0xFF == ord('q'):
103                     runReading = False
104                     collector.join()
105                     break
106         cap.release()
107         cv2.destroyAllWindows()
```

Пример скриншота работающей программы для данного решения:

Задача 6.1.4. (10 баллов)

Опишите свою физиологическую модель, использовавшуюся для решения задачи 1. Для этого сформулируйте в текстовом виде (формат .txt) закономерности, выявленные в анализируемых сигналах, в соответствии с которыми выносилось решение о состоянии водителя (состояния “0” или “1” в задаче 1). Решение необходимо сдать представителю жюри.

Система оценки

На базе статей, которые были предоставлены для подготовки к финалу (Приложение 1), были выделены паттерны сигналов, характерные для разных состояний

водителя. На основании этих данных были сгенерированы данные для задачи 1. За каждый верно определенный паттерн выставляется по 2 балла. В сумме не более 10 баллов.

Число попыток: 1 попытка.

Срок сдачи: Данное задание можно сдавать 21 марта (весь день) и 22 марта (до 12-00), дается всего 1 попытка.

Решение

При генерации данных использовались следующие паттерны сигналов, характерные для разных состояний водителя (на основе статей из Приложения №1):

1. Временной интервал между локальными минимумами сигнала КГР изменяется пропорционально уровню внимания водителя. Чем ниже уровень внимания, тем больше временные промежутки. Сигнал был искажён симуляцией механических артефактов и изменения температуры в кабине автомобиля.
2. При снижении уровня внимания водителя частота сердечных сокращений падает от величин порядка 90 ударов в минуту до порядка 60.
3. Со снижением внимания амплитуда альфа-ритма относительно амплитуды бета-ритма сигнала ЭЭГ растёт.
4. Уровни альфа- и бета- ритмов в сигнале ЭЭГ периодически усиливаются и ослабляются на промежутках времени порядка 30-60 с. При этом длительность периодов усиления альфа-ритма со снижением уровня внимания растёт. Длительность периодов при этом колеблется.

Дополнительным паттерном, имеющим корреляцию с уровнем внимания водителя, является отношение спектральных компонент кардиоритмограммы (кардиоинтервалограммы): низкочастотной (LF) к высокочастотной (HF). При появлении сонливости наблюдается рост LF/ HF. Данный признак не был использован при генерации данных для задачи 1, но оценивается наравне с вышеупомянутыми признаками.

За каждый верно определенный паттерн ставилось 2 балла. В сумме – не более 10 баллов.

Комментарий: обратите внимание, что при регистрации КГР измеряется сопротивление, а не проводимость.

Задача 6.1.5. (5 баллов)

Реализуйте систему мониторинга состояния водителя, которая включит в себя решения задач 2 и 3, а именно, система должна одновременно выполнять следующие функции:

- (1 балл) На экран должно выводиться изображение с веб-камеры в режиме реального времени;
- (3 балла) В случае, если человек, на которого направлена камера дольше чем на 1 секунду, закрыл глаза, или смотрит на свой мобильный (который лежит на уровне коленей водителя), или смотрит на автомагнитолу, необходимо активировать пьезо-элемент, подключенный к плате Arduino.

- (1 балл) На экране в режиме реального времени поверх изображения с камеры должны выводится следующие параметры:
  - отношение амплитуды альфа-ритма (8-13 Гц включительно) к амплитуде бета-ритма (15-30 Гц включительно) для ЭЭГ. Временное окно для вычисления амплитуд составляет 1 секунду;
  - число сердечных сокращений в минуту, вычисляемое по 10 последним ударам сердца;
  - значение сигнала с модуля кожно-гальванической реакции (в вольтах).

Система оценки

Испытуемый по команде от члена жюри поочередно реализует вышеописанные ситуации. Баллы за первый пункт выставляются в случае вывода видео на экран. Баллы за второй пункт выставляются за корректную активацию пьезо-элемента: по 1 баллу за каждую из трёх ситуаций. Балл за третий пункт выставляется при корректном изменении параметров при совершении действий по команде члена жюри.

Число попыток: одна попытка 22 марта.

Решение

Данная задача является объединением всех задач 1, 2, 3, 4. Т.к. по отдельным частям данная система уже была проверена в предыдущих задачах, а проверка осуществлялась качественным путем, то за данную задачу ставилось максимум только 5 баллов.

На скриншоте ниже приведен скриншот программы для работающей системы:

На фото ниже представлена система для регистрации биосигналов и видеопотока (на примере одной из команд-участников).

Скетч для платы Arduino (на основе задачи 3)

```
#include <TimerOne.h>
const int buzzerPin = 11;

void setup() {
    Serial.begin(115200);
    Timer1.initialize(4000);
    Timer1.attachInterrupt(sendData);
    pinMode(buzzerPin, OUTPUT);
}

void loop() {
    if (Serial.available()) {
        int cmd = Serial.read();
        if (cmd == 49)
        tone(buzzerPin, 500);
        if (cmd == 48)
        noTone(buzzerPin);
    }
    delay(1);
}

void sendData() {
    int val1 = analogRead(0) / 4;
    int val2 = analogRead(1) / 4;
    int val3 = analogRead(2) / 4;
    Serial.write("A0");
    Serial.write(val1);
    Serial.write("A1");
    Serial.write(val2);
    Serial.write("A2");
    Serial.write(val3);
}
```

Программа для анализа видеопотока

```
import cv2
import dlib
import numpy as np
from numpy.linalg import norm
from imutils import face_utils
from serial import Serial
from collections import deque
from struct import unpack
import threading

albeta, gsr, hr = 0, 0, 0
runReading = True


class DriverControl:
    """Класс для определения направления поворота головы. Подробности по адресу
    www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/ """

    def __init__(self, size):
        """ Аргументы:
        size - разрешение изображения
        Переменные экземпляра класса:
        maxFrames - Число последовательных кадров на видео, для которых
        должны выполняться условия срабатывания
        пьезоэлемента, прежде чем его активировать.
        earThold -  порог ширины открытия глаз,
        выше ниже которого глаза считаются закрытыми """
        self.size = size
        self.model_points = np.array([
            (0.0, 0.0, 0.0),
            (0.0, -330.0, -65.0),
            (-225.0, 170.0, -135.0),
            (225.0, 170.0, -135.0),
            (-150.0, -150.0, -125.0),
            (150.0, -150.0, -125.0)
        ])
        self.focal_length = size[1]
        self.center = (size[1]/2, size[0]/2)
        self.camera_matrix = np.array([
            [self.focal_length, 0, self.center[0]],
            [0, self.focal_length, self.center[1]],
            [0, 0, 1]
        ], dtype='double')
        self.dist_coeffs = np.zeros((4, 1))
        self.nFrames = 0
        self.maxFrames = 16
        self.earThold = 0.3

    def getNosePoint(self, shape):
        """Вычисляет направление поворота головы.
        Аргументы:
                shape - массив ключевых точек лица
        Возвращаемые значения:
                nosePoint - координаты конца вектора, указывающего направление
                        головы. Начало вектора - ключевая точка лица
                        с индексом 30.
        """
        if shape is None:
```

```
59                  return (0, 0)
60              image_points = np.array([
61                  shape[30],          # Nose tip
62                  shape[8],           # Chin
63                  shape[45],          # Left eye left corner
64                  shape[36],          # Right eye right corne
65                  shape[54],          # Left Mouth corner
66                  shape[48]           # Right mouth corner
67              ], dtype="double")
68
69              (success, rotation_vector,
70              translation_vector) = cv2.solvePnP(self.model_points,
71                                                 image_points,
72                                                 self.camera_matrix,
73                                                 self.dist_coeffs,
74                                                 flags=cv2.SOLVEPNP_ITERATIVE)
75              (nose_end_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0,
76                                                                          1000.0)]),
77                                                               rotation_vector,
78                                                               translation_vector,
79                                                               self.camera_matrix,
80                                                               self.dist_coeffs)
81              nosePoint = (int(nose_end_point2D[0][0][0]),
82                           int(nose_end_point2D[0][0][1]))
83              return nosePoint
84
85          def getEAR(self, shape):
86              """Вычисляет ширину открытия глаз.
87              Аргументы:
88                      shape - массив ключевых точек лица
89              """
90              leftH = (norm(shape[37]-shape[41]) + norm(shape[38]-shape[40])) / 2
91              leftW = norm(shape[36]-shape[39])
92              leftEAR = leftH / leftW
93              rightH = (norm(shape[43]-shape[47]) + norm(shape[44]-shape[46])) / 2
94              rightW = norm(shape[42]-shape[45])
95              rightEAR = rightH / rightW
96              EAR = (leftEAR + rightEAR) / 2
97              return EAR
98
99          def checkDriver(self, shape):
100             """Возвращает True если человек дольше чем на maxFrames кадров закрыл
101             глаза, смотрит на свой мобильный, который лежит у него на колене или
102             смотрит на автомагнитолу; иначе - False.
103             Аргументы:
104                 shape - массив ключевых точек лица
105             """
106             # Получаем координату конца вектора направления поворота головы
107             nosePoint = self.getNosePoint(shape)
108             # Вычисляем ширину открытия глаз
109             ear = self.getEAR(shape)
110             # Проверяем не ниже ли порога ширина открытия глаз и не опустился ли
111             # конец вектора направления поворота головы ниже верхней губы
112             if ear < self.earThold or nosePoint[1] > shape[62][1]:
113                 self.nFrames += 1
114                 if self.nFrames >= self.maxFrames:
115                     return False
116             else:
117                 self.nFrames = 0
118                 return True
```

```
119
120
121     class LandmarkDetector:
122         """Класс для поиска ключевых точек лица на изображении"""
123
124         def __init__(self, modelPath='shape_predictor_68_face_landmarks.dat',
125                      showLandmarks=False):
126             """Аргументы:
127             modelPath - путь к файлу с предобученной моделью для поиска
128                         ключевых точек.
129             showLandmarks - отмечать ли ключевые точки на лице."""
130             self.detector = dlib.get_frontal_face_detector()
131             self.predictor = dlib.shape_predictor(modelPath)
132             self.showLandmarks = showLandmarks
133
134         def getLandmarks(self, image):
135             """Возвращает массив с координатами ключевых точек.
136             Аргументы:
137                     image - изображение для поиска ключевых точек"""
138             gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
139             rects = self.detector(gray, 0)
140             shape = None
141
142             for rect in rects:
143                 (bX, bY, bW, bH) = face_utils.rect_to_bb(rect)
144                 cv2.rectangle(image, (bX, bY), (bX + bW, bY + bH),
145                               (0, 255, 0), 1)
146                 shape = self.predictor(gray, rect)
147                 shape = face_utils.shape_to_np(shape)
148                 if self.showLandmarks is True:
149                     for (x, y) in shape:
150                         cv2.circle(image, (x, y), 1, (0, 0, 255), -1)
151
152             return shape
153
154
155     def launchVideo(videoSource, arduinoPort):
156         """Воспроизводит видео, подаёт команды на Arduino. Для завершения программы
157         нажать q.
158         Аргументы:
159         videoSource - источник видеопотока.
160         arduinoPort - COM-порт к которому подключается Arduino"""
161         landDetector = LandmarkDetector()
162         cap = cv2.VideoCapture(videoSource)
163         res = np.array((cap.get(4), cap.get(3)), dtype=int)
164         dc = DriverControl(res)
165         ser = Serial(arduinoPort, 115200)
166         collector = threading.Thread(target=collectData, args=(ser,))
167         collector.start()
168         while True:
169         # Считываем кадр
170             ret, frame = cap.read()
171             if ret:
172                 # Ищем ключевые точки лица
173                 shape = landDetector.getLandmarks(frame)
174                 if shape is not None:
175                     # Проверяем не выполняется ли одно из условий задачи.
176                     driverOK = dc.checkDriver(shape)
177                 if driverOK is False:
178                             # Если выполняется, шлём на Arduino символ "1"
```

```
179                     ser.write(b'1')
180                 else:
181                     ser.write(b'0')
182                 cv2.putText(frame, 'Alpha to Beta = {:.2f}'.format(albeta),
183                             (20, 20), cv2.FONT_HERSHEY_SIMPLEX,
184                             0.5, (0, 255, 0), 1)
185                 cv2.putText(frame, 'Heart rate = {:.2f} bpm'.format(hr),
186                             (20, 40), cv2.FONT_HERSHEY_SIMPLEX,
187                             0.5, (0, 255, 0), 1)
188                 cv2.putText(frame, 'GSR level = {:.2f} V'.format(gsr),
189                             (20, 60), cv2.FONT_HERSHEY_SIMPLEX,
190                             0.5, (0, 255, 0), 1)
191                 cv2.imshow("Output", frame)
192                 if cv2.waitKey(1) & 0xFF == ord('q'):
193                     break
194             global runReading
195         runReading = False
196         collector.join()
197         cap.release()
198         ser.close()
199         cv2.destroyAllWindows()
200
201
202     def equalize(ser):
203         """Считывает данные с последовательного порта таким образом, чтобы после
204         завершения работы equalize следующее сообщение начиналось с "A0"
205         """
206         buf = (None, None)
207         a0 = (b'A', b'0')
208         while buf != a0:
209             buf = (buf[1], ser.read(1))
210             ser.read(7)
211
212
213     def getAlBeta(eeg):
214         """Возвращает соотношение суммы амплитуд компонент альфа-ритма к сумме
215         амплитуд компонент бета-ритма. eeg - отрезок сигнала ЭЭГ длительностью 1 с
216         """
217         spectrum = np.abs(np.fft.fft(eeg))
218         albeta = sum(spectrum[8:14]) / sum(spectrum[15:31])
219         return albeta
220
221
222     class Pulse():
223         """Класс для вычисления ЧСС"""
224
225         def __init__(self):
226             """Длину дека data в котором хранятся данные берём из расчёта поместить
227             в него 10 сердечных сокращений при ЧСС 40 ударов в минуту."""
228             self.data = deque(maxlen=15*250)
229
230         def getHR(self, ecg):
231             """ Вычисляет ЧСС"""
232             # Записываем в конец дека последние поступившие данные
233             self.data += ecg
234             # Порог для поиска вершины R-зубца
235             thold = 0.9 * (max(self.data) - min(self.data)) + min(self.data)
236             peaks = 0
237             # В first записываем индекс вершины первого R-пика, в last - десятого
238             last, first = None, None
```

```
239             i = len(self.data) - 10
240             # Пики ищем с конца массива, так как нас интересуют последние биения
241             while i > 10:
242                 i -= 1
243                 if self.data[i] > self.data[i+10] and \
244                         self.data[i] > self.data[i-10] and self.data[i] > thold:
245                         peaks += 1
246                         i -= 80
247                 if peaks == 1:
248                     last = i
249                 if peaks == 10:
250                     first = i
251             if (last is not None) and (first is not None):
252                 dt = (last - first) / 250 / 9
253                 hr = 60 / dt
254                 return hr
255             return 0
256 
257 
258     def collectData(ser):
259         """Вычисляет отношение альфа к бета, значение КГР и ЧСС
260         Аргументы:
261         ser - объект класса Serial"""
262         global albeta, gsr, hr
263         pulse = Pulse()
264         equalize(ser)
265         while runReading:
266             chunk = ser.read(250 * 9)
267             albeta = getAlBeta(unpack(b'250B', chunk[5::9]))
268             gsr = chunk[8] * 5 / 255
269             hr = pulse.getHR(unpack(b'250B', chunk[2::9]))
270 
271 
272     if __name__ == '__main__':
273         launchVideo(0, 'COM10')
```

6.2. Приложения к задачам

Приложение 1 — Список литературы к задаче №1 и №4

1. Регуляция цикла бодрствование-сон, Ковальзон В.М., Долгих В.В. Неврологический журнал, №6, 2016 стр.316-322. http://www.sleep.ru/lib/Nevrol_6-2016.pdf
2. Нейрофизиология и нейрохимия сна, Ковальзон В.М. Сомнология и медицина сна. Национальное руководство памяти А.М.Вейна и ЯЯ.И.Левина/ Ред. М.Г. Полуэктов. М.: «Медфорум». 2016. С.11 – 55. http://www.sleep.ru/lib/Medforum_2016_1.pdf
3. Прогнозирование моментов критического снижения уровня бодрствования по показателям зрительно-моторной координации, Г.Н. Арсеньев, О.Н.Ткаченко, Ю.В. Украинцева, В.Б.Дорохов Журнал высшей нервной деятельности, 2014, том 64, №1, с. 64-76. http://www.sleep.ru/lib/arsenev2014_ru.pdf

4. Сомнология и безопасность профессиональной деятельности, В.Б.Дорохов. Журнал высшей нервной деятельности, 2013, том 63, №1, с. 33 – 47 http://www.sleep.ru/lib/Dorokhov_ed_2013JourVND.pdf

5. Психомоторный тест для исследования зрительно-моторной координации при выполнении монотонной деятельности по прослеживанию цели, В.Б.Дорохов, Г.Н.Арсеньев, О.Н.Ткаченко, Д.В.Захарченко, Т.П.Лаврова, В.В.Дементиенко. Журнал высшей нервной деятельности, 2011, том 61 №4, с. 476 – 484 http://www.sleep.ru/lib/VND0476.pdf

6. Биоматематическая модель процесса засыпания человека-оператора, В.В.Дементиенко, В.Б.Дорохов, С.В.Герус, Л.Г.Коренева, А.Г.Марков, В.М.Шахнарович. Физиология человека, 2008, том 34, №4, с. 1 – 10 http://www.sleep.ru/lib/Dementienko_Dor_Rus.pdf

7. Альфа-активность ЭЭГ при дремоте, как необходимое условие эффекторного взаимодействия с внешним миром, Дорохов В.Б. Электронный журнал «Исследовано в России», 2003. http://www.sleep.ru/download/192.pdf

8. Альфа-веретена и К-комплекс – фазические активационные паттерны при спонтанном восстановлении нарушений психомоторной деятельности на разных стадиях дремоты, В.Б.Дорохов Журнал высшей нервной деятельности, 2003, том 53, №4, с.502 – 511 http://www.sleep.ru/lib/Dorokhov-K-compl.pdf

9. Электродермальные показатели субъективного восприятия ошибок в деятельности при дремотных изменениях сознания, В.Б.Дорохов, В.В.Дементиенко, Л.Г.Коренева, А.Г.Марков, В.М.Шахнарович Журнал высшей нервной деятельности, 2000., том 50, №2, с. 206 – 218 http://www.sleep.ru/lib/Dorokhov2000.pdf

10. Анализ психофизиологических механизмов нарушения деятельности при дремотных изменениях сознания, В.Б.Дорохов. Вестник РГНФ, 2003, с. 137 – 144 http://www.sleep.ru/download/Dorohov_04.pdf

11. Обзор систем бдительности водителя, перевод и публикация с разрешения Совета по Безопасности и Стандартам на Железных Дорогах Великобритании (RSSB), 2002 http://www.neurocom.ru/pdf/press/report_rssb_russian.pdf

12. Hypothesis about the nature of electrodermal reactions, Dementienko V.V., Koreneva L.G., Tarasov A.V., Shakhnarovitch V.M. Journal of Psychophysiology, 1998.V.30/1-2, p.267 http://www.neurocom.ru/pdf/press/edrhype.pdf

13. Удаленный контроль состояния водителя как средство повышения безопасности перевозок, Тагиров М.К., Юров А.П., Иванов И.И., Макаев Д.В. Материалы II Международной научно-практической конференции «Научно-технические аспекты комплексного развития транспортной отрасли» в рамках Международного форума Донецкой Народной Республики 25-26 мая 2016 года. Секция «Информационные процессы, технологии и системы на транспорте». http://www.diat.edu.ua/files/2016.pdf#page=52

14. Программно-аппаратная реализация бортовых оперативно-советующих экспертных систем на транспорте, Н.В.Корнеев, А.В. Гребенников. Известия Самарского научного центра Российской академии наук, т.16, №4, 2014, с. 116-122

https://cyberleninka.ru/article/n/programmno-apparatnaya-realizatsiya-bortovyh-operativno-sovetuyuschih-ekspertnyh-sistem-na-transporte

15. Вариабельность сердечного ритма во время сна у здоровых людей, И.М.Воронин, Е.В.Бирюкова Вестник аритмологии, №30, 2002, с. 68 – 71 http://www.vestar.ru/atts/2965/2965voronin.pdf
16. Эффективность систем мониторинга водителя, В.В.Дементиенко, В.Б.Дорохов, С.В.Герус, А.Г.Марков, В.М.Шахнарович. Журнал технической физики, 2007, том 77, вып. 6, с. 103 – 108. https://journals.ioffe.ru/articles/viewPDF/9153
17. Оценка эффективности систем контроля уровня бодрствования человека-оператора с учетом вероятностной природы возникновения ошибок при засыпании, В.В.Дементиенко, В.Б.Дорохов Журнал высшей нервной деятельности, 2013, том 63, №1, с. 24 – 32 http://www.sleep.ru/lib/JourVND_63_01.pdf
18. Система анализа физиологического состояния водителей транспортных средств, Щербакова Т.Ф., Култынов Ю.И., Осипова О.С. Актуальные проблемы и достижения в медицине/ Сборник научных трудов по итогам международной научно-практической конференции. №2. Самара, 2015.Секция №21. Медицина труда. http://izron.ru/articles/aktualnye-problemy-i-dostizheniya-v-meditsine-sbornik-nauchnykh-trudov-po-itogam-mezhdunarodnoy-nauch/sektsiya-21-meditsina-truda-spetsialnost-14-02-04/sistema-analiza-fiziologicheskogo-sostoyaniya-voditeley-transportnykh-sredstv/
19. Бортовая система мониторинга функционального состояния оператора транспортного средства, В.В.Савченко. Механика машин, механизмов и материалов, 2012, №1, с. 20 – 25 http://mmmm.by/pdf/ru/2012/1_2012/3.pdf
20. Электроэнцефалографические показатели дремотного состояния при выполнении монотонной операторской деятельности, О.Н.Ткаченко, А.А.Фролов. Труды МФТИ, 2010, том 2, №2, с. 41 – 45 https://mipt.ru/science/trudy/2_6/41-45-arphcxl1tgs.pdf
21. Основы сомнологии: физиология и нейрохимия цикла «Бодрствование - сон», В.М.Ковальзон – М.: БИНОМ. Лаборатория знаний, 2012. – 239 с. https://docplayer.ru/26078038-V-m-kovalzon-osnovy-somnologii.html

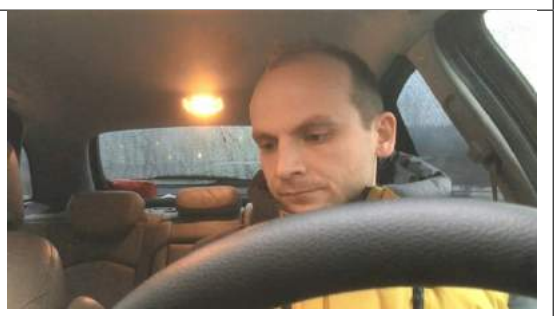
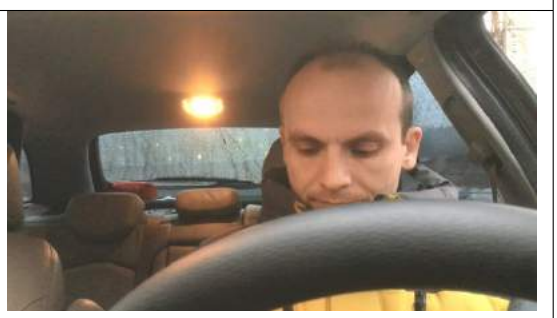
Приложение 2 — Система оценивания на платформе Kaggle

Первое задание будет проходить в формате соревнования на платформе Kaggle. Точность решения будет рассчитываться как доля правильно классифицированных временных отрезков тестовых данных от общего числа временных отрезков.

Тестовые данные разделены на две равные части - публичные и приватные. До окончания соревнования на вкладке Public Leaderboard на странице соревнования вам будет доступна точность вашего решения для публичных данных. Вкладка Private Leaderboard, где указана точность решения на приватных данных, станет вам доступна лишь после завершения соревнования. Это сделано для того, чтобы участники не “подгоняли” свои решения под ответ. Баллы за задачу выставляются с использованием результатов из Private Leaderboard.

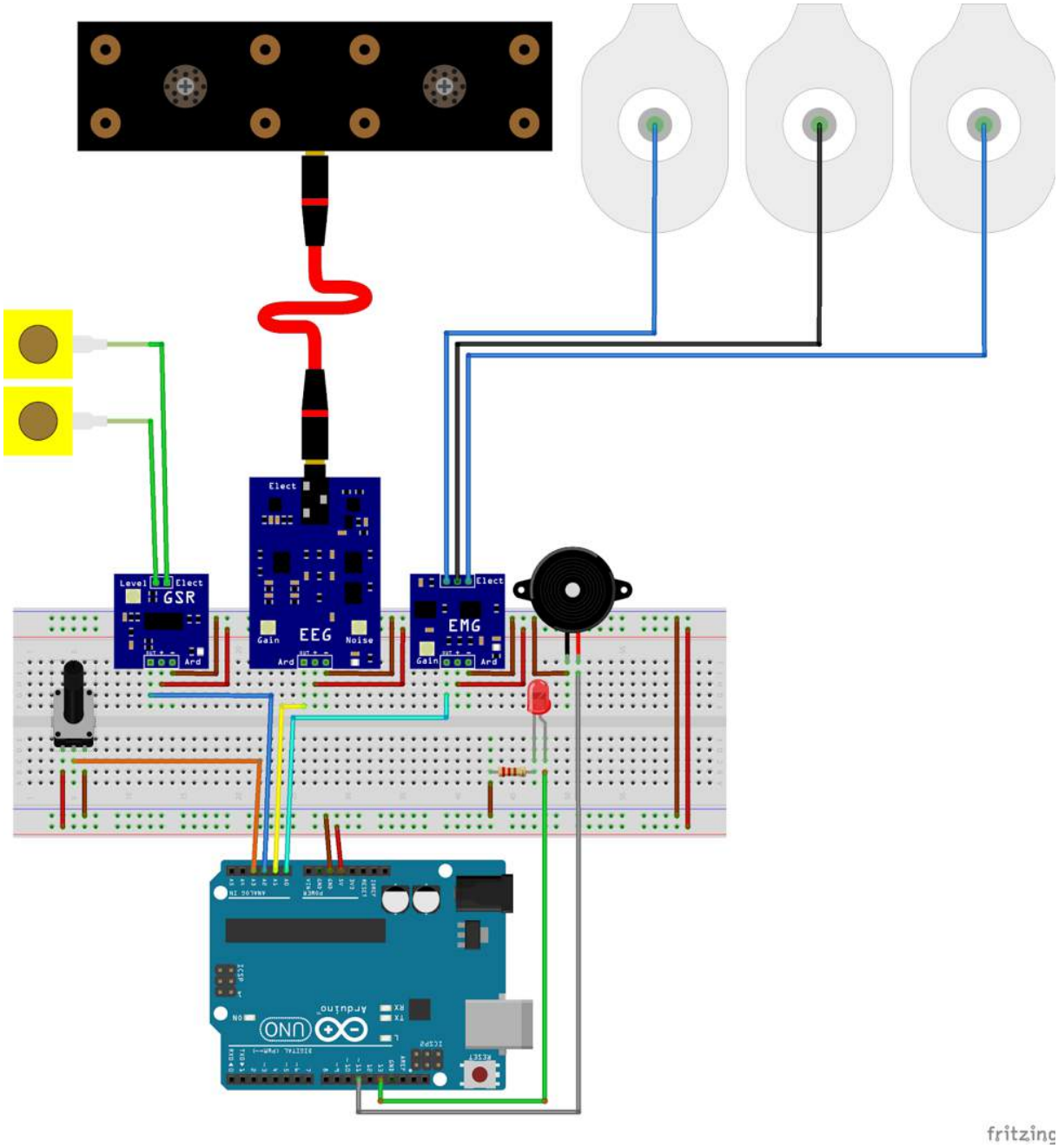
В течение соревнования вы успеете отправить несколько решений. При выставлении баллов в Private Leaderboard по умолчанию используется решение с наивысшим баллом на Public Leaderboard. Если вас это не устраивает, вы можете на вкладке My Submissions самостоятельно выбрать до двух решений, которые будут использоваться для выставления оценки за задания, отметив их галочкой в столбце Use for Final Score.

Приложение 3 — Скриншоты к задаче №2

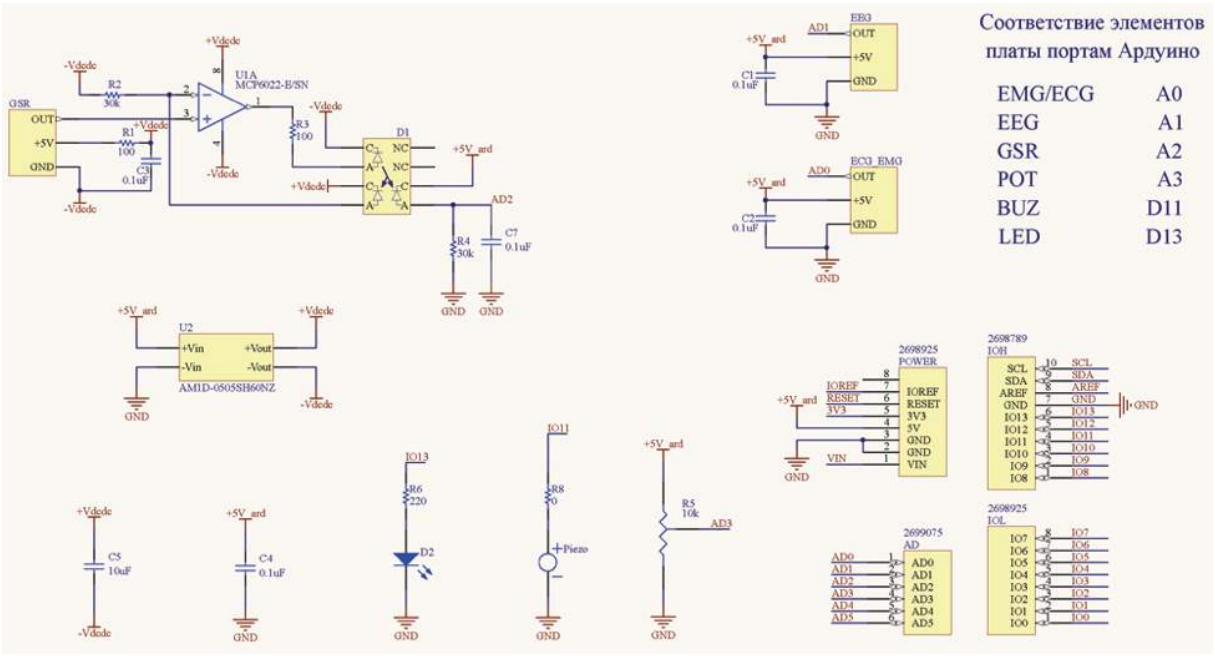
Обычное состояние	
Закрытые глаза	
Взгляд на магнитолу	
Использование мобильного телефона	
Взгляд в зеркало заднего вида	
Взгляд в зеркало бокового вида	

Приложение 4 — Схема подключения модулей и радиодеталей

Схема подключения модулей и радиодеталей, эквивалентная реализованной на плате расширения.



Электрическая принципиальная схема платы расширения


Соответствие элементов платы портам Ардуино
EMG/ECG A0
EEG A1
GSR A2
POT A3
BUZ D11
LED D13
GSR
U1A MCP6022-E/SN
D1
U2 AM1D-0505SH60NZ
EEG
ECG_EMG
2698925 POWER
2698789 IOH
2698925 IOL
2699075 AD
R1 100
R2 30k
R3 100
R4 30k
R5 10k
R6 220
R8 0
C1 0.1uF
C2 0.1uF
C3 0.1uF
C4 0.1uF
C5 10uF
C7 0.1uF
D2
Piezo
AD1
AD0
AD2
AD3
IO13
IO11
+5V_ard
-Vdcdc
+Vdcdc
GND

Приложение 5 — Протокол передачи данных

Загруженный на плату Arduino скетч должен опрашивать три аналоговых входа платы (A0, A1, A2) и отправлять результаты оцифровки с данных входов на последовательный (Serial) порт компьютера. Для того чтобы программа на компьютере смогла различать данные с разных входов, необходимо их разделить. Для разделения используются строки с названиями портов: “A0”, “A1”, “A2”. Таким образом, на последовательный порт в одном сообщении мы отправляем 9 байт информации: “A0XA1YA2Z”.

1. A0, A1, A2 — три строки, каждая из которых состоит из двух символов. Символ занимает 1 байт.
2. X, Y, Z — результаты оцифровки сигналов с аналоговых входов A0, A1, A2. Эти данные предварительно нужно привести к размеру 8 бит каждое. Плата Arduino имеет 10 битный аналогово-цифровой преобразователь, поэтому функция analogRead возвращает значение от 0 до 1023 ($1023 = 2^{10} - 1$). Для приведения считанного значения к 8-битному диапазону можно использовать функцию map, деление на 4 или побитовый сдвиг вправо на две позиции.

Частота обмена данными с COM-портом должна составлять 115200 бит/с. На компьютер должно передаваться 250 описанных выше сообщений в секунду. Равные временные промежутки между циклами оцифровки сигнала можно формировать при помощи функции delay или средствами библиотеки TimerOne.