**Руководство программиста**

Клиент 1 является самостоятельным клиентом, имеющим свой набор топиков, которые он может отправлять и принимать. Клиент 1 состоит из одного класса и нескольких методов. Главный класс называется MainWindow. Далее будут описаны основные методы, отвечающие за передачу и принятие сообщений. Обращение к брокеру осуществляется следующим способом:

self.client = mqtt.Client()

        self.client.connect("test.mosquitto.org")

        self.start\_take()

В данном проекте используется брокер mosquitto.org. Брокер позволяет производить обмен сообщениями или реакциями между тремя клиентами. Ниже представлены подписки первого клиента:

        list\_for\_publish\_client\_1 = [

            'mqtt/example1',

            'device/ip',

            'mqtt/picture',

            'mqtt/get\_weather/temp',

            'mqtt/get\_weather/status',

            'android/vibro',

        ]

Для того, чтобы приложение выглядело более привлекательно, существуют методы, отключающие панель навигации от windows, так как у приложения есть собственная панель.

    def center(self):

        qr = self.frameGeometry()

        cp = QDesktopWidget().availableGeometry().center()

        qr.moveCenter(cp)

        self.move(qr.topLeft())

    def mousePressEvent(self, event):

        self.oldPos = event.globalPos()

    def mouseMoveEvent(self, event):

        try:

            delta = QPoint(event.globalPos() - self.oldPos)

            self.move(self.x() + delta.x(), self.y() + delta.y())

            self.oldPos = event.globalPos()

        except AttributeError:

            pass

После этого создаётся новый процесс, который отправляет сообщения. Метод также получает корректную дату и время для того, чтобы отобразить её в элементе textedit.

    def start\_send(self):

        threading.Thread(target=self.public(self.client),daemon=True).start()

    def public(self,client):

        now = datetime.now()

        cur\_time = now.strftime("%H:%M:%S")

        text = self.ui.lineEdit\_for\_writetext.text()

        pubtop = self.ui.comboBox\_for\_select\_topic.currentText()

Для того, чтобы отобразить отправляемое сообщение или реакцию на другом клиенте, необходимо выбрать её из выпадающего меню, а замет нажать кнопку отправить. Следующие условия отвечают за выбранные топики и отправку сообщений:

        if pubtop == 'android/vibro':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<вибрация> ' + '\n')

        if pubtop == 'mqtt/example1':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' +'<' + text + '> ' + '\n')

        if pubtop == 'device/ip':

            h\_name = socket.gethostname()

            IP\_addres = socket.gethostbyname(h\_name)

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<' + IP\_addres + '> '+ '\n')

        if pubtop == 'mqtt/picture':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<Picture1.jpg> ' + '\n')

        if pubtop == 'mqtt/get\_weather/temp':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<запрос на температуру> ' + '\n')

        if pubtop == 'mqtt/get\_weather/status':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<запрос на статус погоды> ' + '\n')

Кроме этого, на клиенте присутствуют методы, которые отвечают за принятие топиков с другого клиента. В данном случае это начало проигрывания трека и пауза:

    def start\_track(self):

        pygame.mixer.music.load('Track1.mp3')

        pygame.mixer.music.play()

        self.playing = True

    def stop\_track(self):

        if self.playing:

            pygame.mixer.music.pause()

Если на клиент приходит сообщение, то вызывается метод на клиенте. Условие, которое это реализует представлено ниже:

        if message.topic == 'device/memorystatus/harddrive/c':

            self.ui.textEdit\_for\_view.insertPlainText(message.topic + ' ' + str(self.hdd.free / (2\*\*30))+ '\n')

            self.ui.textEdit\_for\_view.setStyleSheet('background-color: red')

            time.sleep(3)

            self.ui.textEdit\_for\_view.setStyleSheet(

            'background-color: rgb(76, 79, 84);'

            'color: white;'

            'border-radius: 1px solid;')

Для корректной работы необходимо также создать обработку принятого сообщения, и вывод времени его прихода в textedit. Поэтому ниже представлено условие, обрабатывающее принятые сообщения:

        if message.topic == 'device/work/cpu':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + str(psutil.cpu\_percent(interval=1))+ '%' + '\n')

        if message.topic == 'mqtt/text/chat':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + message.payload.decode('utf-8') + '\n' )

        if message.topic == 'device/work/ram':

            values = psutil.virtual\_memory().percent

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + str(values) + '%' + '\n')

        if message.topic == 'music/track1/start':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + '  Track1 начал проигрывание' + '\n')

            self.start\_track()

        if message.topic == 'music/track1/stop':

            self.ui.textEdit\_for\_view.insertPlainText( '['+ cur\_time + '] '+ message.topic + '  Track1 закончил проигрывание' + '\n')

            self.stop\_track()

        if message.topic == 'mqtt/chat/client\_1/android':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + message.payload.decode('utf-8') + '\n' )

        if message.topic == 'mqtt/pc/client\_1/restart':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + '<Перезагрузка> ' + '\n' )

            self.restart()

        if message.topic == 'mqtt/file/client\_1/get\_text':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + '<Получить ascii> ' + '\n' + '\n' )

            self.get\_ascii()

Кроме всего перечисленного, для того чтобы в целом видеть топики другого клиента, нужно быть подписанным на них. Метод take\_message позволяет это сделать. Он подписывается как на топики второго клиента, так и на топики android-клиента:

    def take\_message(self,client):

        self.client.subscribe('device/memorystatus/harddrive/c')

        self.client.subscribe('device/work/cpu')

        self.client.subscribe('mqtt/text/chat')

        self.client.subscribe('device/work/ram')

        self.client.subscribe('music/track1/start')

        self.client.subscribe('music/track1/stop')

        self.client.subscribe('mqtt/chat/client\_1/android')

        self.client.subscribe('mqtt/pc/client\_1/restart')

        self.client.subscribe('mqtt/file/client\_1/get\_text')

        self.client.loop\_start()

        client.on\_message = self.on\_message

Клиент 2 идентичен первому. Он отличается разве что набором отправляемых и принимаемых топиков. Например, список подписок клиента 2 выглядит следующим образом:

        list\_for\_publish = [

            'mqtt/text/chat',

            'device/memorystatus/harddrive/c',

            'device/work/cpu',

            'device/work/ram',

            'music/track1/start',

            'music/track1/stop',

            'android/get\_ascii',

            'android/get\_ip',

            'android/tts'

        ]

Соответственно у второго клиента другие условия, отвечающие за отображение отправляемой информации с других клиентов:

        if subtop == 'device/memorystatus/harddrive/c':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<'+ str(self.hdd.free / (2\*\*30)) + '> ' + '\n')

        if subtop == 'device/work/cpu':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<'+ str(psutil.cpu\_percent(interval=1)) + '> ' + '%' + '\n')

        if subtop == 'mqtt/text/chat':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<mqtt/text/chat>  '+ text + '\n')

        if subtop == 'device/work/ram':

            values = psutil.virtual\_memory().percent

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<'+str(values) + '> '+ '%' + '\n')

        if subtop == 'music/track1/start':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<команда старт для Track1> ' + '\n')

        if subtop == 'music/track1/stop':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<команда стоп для Track1> ' + '\n')

        if subtop == 'android/get\_ascii':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<запрос на ascii> ' + '\n')

        if subtop == 'android/get\_ip':

            self.ui.textEdit\_for\_view2.insertPlainText('['+ cur\_time + '] ' + '<запрос на ip> ' + '\n')

Android-клиент похож на pc-клиенты. Он также подключается к брокеру mosquito.org. Для того, чтобы отправить реакции на другие клиенты, существует метод public, который состоит из условий. Выбор отправляемых топиков происходит с помощью выпадающего меню. Метод представлен ниже:

def public(self,client):

text = self.ids.text\_input.text

pubtop = self.ids.spin\_top.text

if pubtop == 'mqtt/chat/client\_1/android':

self.ids.label\_out.text = 'Вы отправили -> ' + text + '\n'

self.client.publish(pubtop,text)

if pubtop == 'mqtt/pc/client\_1/restart':

self.ids.label\_out.text = 'Вы отправили -> <Перезагрузка>'

self.client.publish(pubtop)

if pubtop == 'mqtt/file/client\_1/get\_text':

self.ids.label\_out.text = 'Вы отправили -> <ASCII> '

self.client.publish(pubtop)

if pubtop == 'mqtt/browser/client\_2/open':

self.ids.label\_out.text = 'Вы отправили -> <Открыть браузер> '

self.client.publish(pubtop)

if pubtop == 'mqtt/pc/client\_2/get\_screen':

self.ids.label\_out.text = 'Вы отправили -> <Сделать скриншот> '

self.client.publish(pubtop)

if pubtop == 'android/get\_ip/return':

self.ids.label\_out.text = 'Вы отправили -> <Возврат ip> ' + '\n'

В данном руководстве были представлены основные методы, которые служат для взаимодействия трёх клиентов между собой с помощью брокера.