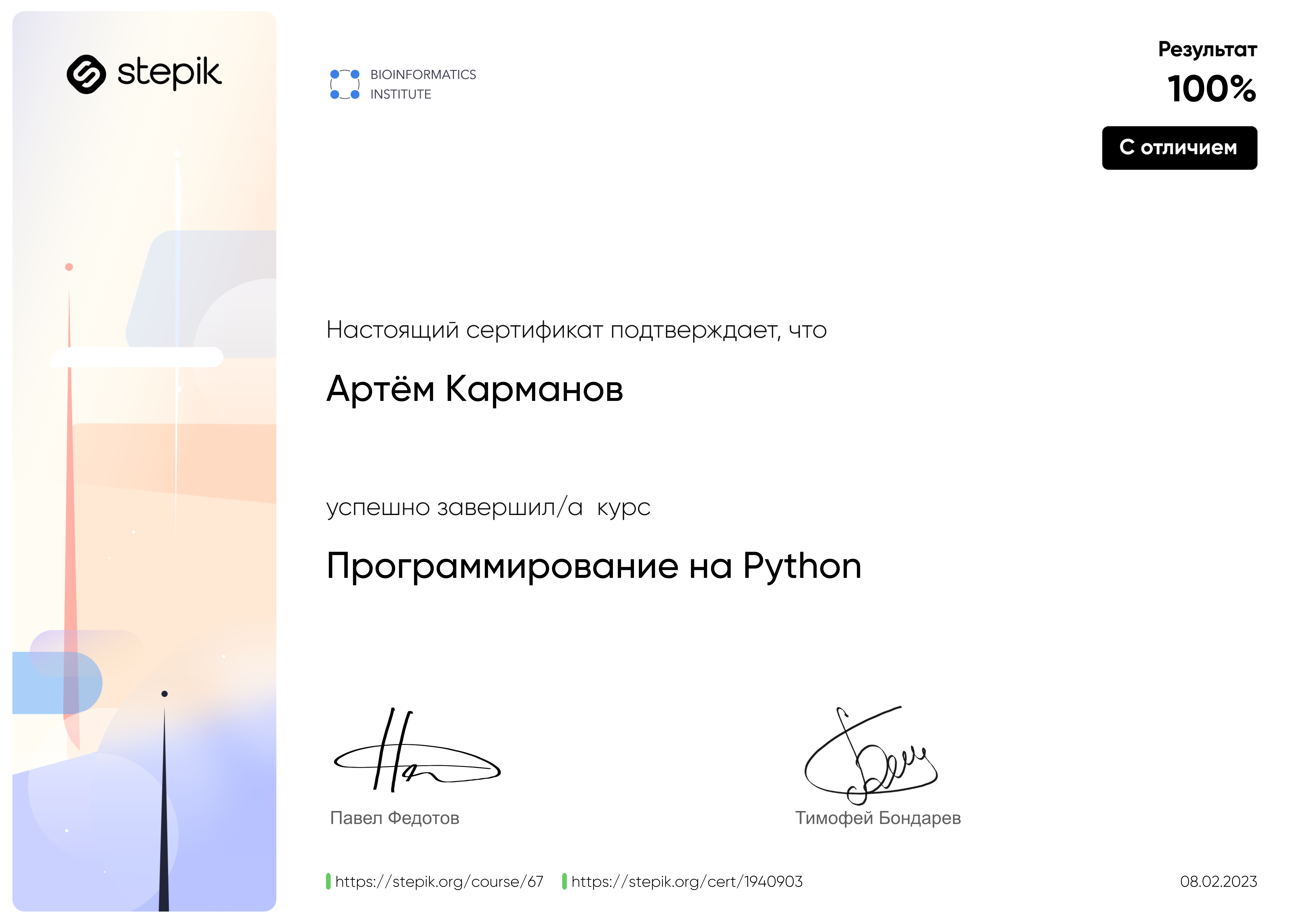
|  |  |
| --- | --- |
| **Задание** | **Срок выполнения** |
| 1. Изучение основ программирования на языке python. 3 недели. Форма отчета – сертификат о прохождении курсов. | 25.02.23 |
| 2. Изучение принципов построения приложения для мобильного телефона при помощи MIT App Inventor (<http://appinventor.mit.edu/>). 3 недели. Форма отчета – описание структуры будущего приложения. | 18.03.23 |
| 3. Разработка приложения для телефона, позволяющего собирать данные с какого-либо датчика в ОС Android и отправлять их по беспроводному каналу (Bluetooth) на ПК (ноутбук). 4 недели. Форма отчет – apk-файл с приложением. | 15.04.23 |
| 4. Разработка ПО на языке python для ПК, позволяющее получать данные с датчиков смартфона, сохранять их и выполнять отображение информации. 4 недели. Форма отчета – исходный код приложения. | 13.05.23 |
| 5. Представление проекта. 1 неделя. Форма отчета – готовый проект. | 20.05.23 |

**Решение задания №1:**



**Решение задания №2:**

**Для создания приложения, которое будет передавать данные с гироскопа смартфона по блютуз, необходимо выполнить следующие шаги:**

* Создание нового проекта: Необходимо создать новый проект в MIT App Inventor и назначить ему уникальное имя.
* Добавление компонентов: Для приложения необходимо добавить два основных компонента - компонент гироскопа и компонент блютуз.
* Создание блоков: Для создания логики приложения необходимо создать блоки, которые будут обрабатывать данные с гироскопа и передавать их по блютуз.
* Подключение к гироскопу: Для подключения к гироскопу необходимо использовать компонент гироскопа и его блоки. С помощью блока "When Gyroscope1.SensorChanged" можно получать данные о текущем положении устройства.
* Подключение к блютузу: Для подключения к блютузу необходимо использовать компонент блютуза и его блоки. С помощью блока "BluetoothClient1.Connect" можно установить соединение с другим устройством.
* Отправка данных по блютузу: Для отправки данных по блютузу необходимо использовать блок "BluetoothClient1.SendText". В качестве параметра нужно передать данные, которые необходимо отправить.
* Обработка ошибок: Необходимо добавить обработку ошибок, которые могут возникнуть при подключении к гироскопу или блютузу.
* Тестирование приложения: После создания приложения необходимо провести тестирование для проверки его работоспособности. Для этого можно использовать эмулятор Android или установить приложение на реальное устройство.

**Структура приложения MIT App Inventor, которое будет передавать данные с гироскопа по блютуз, будет следующей:**

* **Главный экран:** Это будет основной экран приложения, на котором будет расположен визуальный компонент гироскопа и интерфейс управления приложением.
* **Кнопка для выбора блютуз сервера:** Нажатие на эту кнопку будет инициировать появление выпадающего списка доступных блютуз устройств.
* **Кнопка начать отправку:** нажатие на эту кнопку инициирует процедуру отправки показаний гироскопа на выбранный блютуз сервер
* **Графические прогресс бары:** 3 элемента графического интерфейса, визуализирующие показания гироскопа по осям X, Y, Z.

**Решение задания №3 (исходный код приложения для смартфона):**

| **Исходный код** | **Комментарии** |
| --- | --- |
|  | Инициализация глобальных переменных, используемых для символьного обращения к цветам пользовательского интерфейса |
|  | Обработчик ошибок, связанных с блютуз подключением |
|  | Получение списка доступных блютуз устройств |
|  | Действия, выполнимые после выбора блютуз устройства из списка доступных: подключение, вывод имени подключенного устройства в статус индикатор, изменение цвета элементов интерфейса на соответствующий подключенному состоянию |
|  | Действия, выполняемы после нажатия на кнопку *send* (Начать отправку) в зависимости от  текущего статуса.  Если *connect\_status* (подключено к блютуз устройству), то по после нажатия присваивается *activ\_status,* который в свою очередь выполняет роль флага, разрешающего отправку данных по срабатыванию таймера (см. подпрограмму таймера).  В случае повторного нажатия (ветвление по условию *если activ\_status*) выполняется отключение от блютуз сервера с соответствующим обновлением надписей и цветов интерфейса. |
|  | Действия, выполняемые по срабатыванию таймера (каждые 20 мс, настраивается в окне графического дизайнера App Inventor).  Получение данных от гироскопа, их вывод в графические индикаторы (X, Y, Z val), графические бары (X, Y, Z bar) и отправка по блютуз с минимальным форматированием. |

**Решение задания №4 (исходный код приложения для ПК):**

Проект с исходным кодом приложения находится во вложении к настоящему отчёту.

Код ревью листинга main файла приведён ниже:

1. Импорты Код содержит следующие импорты:
   * PyQt5.QtWidgets
   * PyQt5.QtSerialPort
   * PyQt5.QtCore
   * PyQt5.QtGui

Импорты PyQt5 классов, требуемых для создания приложения с графическим интерфейсом пользователя (GUI), а также классов, связанных с последовательным портом.

1. Настройка приложения
   * Создание экземпляра класса QApplication.
   * Загрузка иконки для приложения из файла.
   * Получение пути к каталогу исполняемого файла или абсолютного пути к файлу приложения.
   * Загрузка файла интерфейса пользователя (UI) и установка заголовка окна приложения.
2. Подготовка списка последовательных портов
   * Создание экземпляра класса QSerialPort.
   * Получение списка доступных последовательных портов и добавление их в список portList.
   * Добавление списка скоростей передачи данных (baud\_rates) в выпадающий список (comboBox\_baud\_rates) и установка скорости передачи данных по умолчанию (default\_baud\_rate).
3. Функция is\_valid\_data()
   * Проверка данных, принятых из последовательного порта, на наличие шести значений.
   * Проверка, что первое, третье и пятое значение в каждом наборе данных соответствуют символам 'X', 'Y' и 'Z'.
   * Проверка, что значения 'X', 'Y' и 'Z' лежат в диапазоне от -100 до 100.
   * Если данные прошли все проверки, функция возвращает True, в противном случае - False.
4. Функция onRead()
   * Объединение полученных данных в буфер data\_buffer.
   * Если данные в буфере завершаются символом ';', то вызывается функция is\_valid\_data() для проверки данных.
   * Если данные проходят проверку, то значения 'X', 'Y' и 'Z' отображаются на метках и на ползунках, и буфер data\_buffer очищается.
5. Функция onOpen()
   * Получение выбранной скорости передачи данных и установка ее для последовательного порта.
   * Установка выбранного порта для последовательного порта.
   * Открытие последовательного порта для чтения и записи.
6. Функция Close()
   * Закрытие последовательного порта.
7. Инициализация GUI и запуск приложения
   * Добавление списка доступных последовательных портов в выпадающий список (comboBox).
   * Подключение сигнала readyRead() к слоту onRead().
   * Подключение кнопок

**Решение задания №5 (презентация готового проекта):**

Скомпилированные файлы приложений для смартфона и ПК, а также видеопрезентация, демонстрирующая их работоспособность, находятся во вложении к настоящему отчёту.