УДК 616.12—073.7

РЕАЛИЗАЦИЯ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА СЧИТЫВАНИЯ ПОКАЗАНИЙ ЭКГ ЧЕЛОВЕКА В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

ЖУРАВЛЕВ Д. В., ПЕРЕТОКИН М. А.

Воронежский государственный технический университет
(г. Воронеж, Россия)

E-mail: ddom1@yandex.ru, peretokin00@mail.ru

Аннотация. Разработан программно-аппаратный комплекс для определения ЭКГ. Реализовано программное обеспечение для считывания данных с датчика, поступающих в виде необработанного потока байтов. Осуществлена визуализация данных в виде графика и их обработка с помощью фильтрации в режиме реального времени.

Abstract. A software and hardware complex for ECG determination has been developed. Implemented software to read sensor data as a raw byte stream. The visualization of data in the form of a graph and their processing using real-time filtering was carried out.

Введение

В ходе данной работы предполагается разработать техническое решение, для реализации считывания данных ЭКГ человека для дальнейшей их визуализации и обработки с помощью написания специального алгоритма извлечения данных, поступающих в виде необработанного массива байтов. Реализуемый программно-аппаратный комплекс будет состоять из программной и аппаратной части, причем основная роль программной компоненты – извлечение данных, поступающих с датчика для снятия кардиограммы человека, их обработка и дальнейшее представление в удобном виде для анализа полученной информации.

Основная часть

Для считывания данных кардиограммы будем использовать датчик BMD101, который имеет возможность беспроводной передачи информации с помощью технологии Bluetooth. Данное устройство, необходимо закрепить на теле человека в районе грудной клетки рядом с сердцем, для получения более точных показаний [1]. Далее, полученные данные необходимо отправить на персональный компьютер, для дальнейшей обработки информации и извлечения из нее полезных данных. Реализовывать данный механизм будем с помощью Bluetooth модуля HC-05 и адаптера USB-UART. Необходимо подключить выводы VCC и CND датчика HC-05 к соответствующим выводам адаптера, а выводы RX и TX Bluetooth модуля подключить к выводам TX и RX преобразователя соответственно. После этого необходимо вставить адаптер USB-UART в USB-порт компьютера и произвести настройку модуля HC-05. Сделать это можно с помощью программной среды разработки Arduino IDE. В ней необходимо открыть монитор порта для передачи необходимых команд на Bluetooth модуль, чтобы последний мог соединиться с датчиком для считывания кардиограммы человека и получать с него информацию, которую необходимо направить в ЭВМ [1].

Открыв монитор порта, необходимо ввести в терминал команду «AT», кратковременно нажать кнопку на Bluetooth модуле и отправить команду, при этом каждая команда (как и ответ) должна заканчиваться символами перевода строки «\r\n». Если связь установлена правильно, то в ответ модуль ответит OK, и данное сообщение отобразится в терминале. С помощью команды «AT+UART=57600,0,0» устанавливаем скорость приема и передачи данных 57600 бит в секунду с одним стоповым битом без проверки. С такой же скоростью передачи данных работает датчик BMD101, поэтому мы и установили такую скорость. Далее вводим в терминал команду «AT+ROLE=1», которая устанавливает режим подключения HC-05 к датчику ЭКГ, в данном

случае Bluetooth модуль находится в режиме ведущего. В нашем случае модуль HC-05 ведущее устройство, датчик для считывания показаний кардиограммы человека – ведомое. Введя в терминал команду «AT+INQ» находим адрес датчика BMD101, затем командой «AT+PAIR= 8CB8:7E:960144» создаем пару, где 8CB8:7E:960144 – адрес датчика BMD101 [2]. Далее командой «AT+PSWD=0000» устанавливаем код для подключения к ведомому устройству. Вводим в терминал команду «AT+RESET», тем самым, перезагружая модуль HC-05. После перезагрузки на модуле BMD101 постоянно горит синий светодиод, это свидетельствует о том, что данный модуль подключился к другому Bluetooth модулю HC-05 и передает ему данные в виде необработанных пакетов с байтами информации. Убедиться в этом можно, открыв монитор порта. В терминале будет отображаться искаженная информация, которую нужно обработать.

Для обработки информации необходимо изучить структуру передаваемого пакета и синтезировать алгоритм его обработки . В общем случае пакет данных состоит из трех основных элементов: заголовка (Header), полезной нагрузки (DataPayload) и контрольной суммы (CRC). Все эти элементы образуют полный пакет данных, который изображен на рис.1.

Заголовок пакета состоит из 3 байтов: двух байтов синхронизации (SYNC) 0xAA и 0xAA, за которыми следует длина полезной нагрузки (pLength), составляющая 1 байт. Два байта синхронизации используются для обозначения нового поступающего пакета. Байт длины полезной нагрузки указывает длину в байтах полезной нагрузки данных пакета [3].

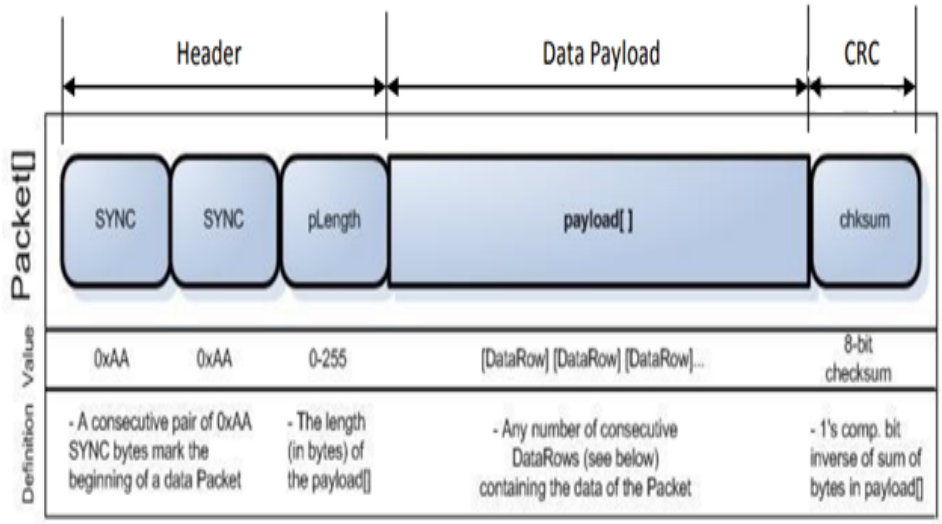

Рис. 1. Структура передаваемого пакета с датчика для снятия показаний кардиограммы человека

Полезная нагрузка – это просто последовательность байтов. Количество байтов полезной нагрузки данных в пакете задается байтом pLenth из заголовка пакета. Извлечение данных из байтов полезной нагрузки подробно описано ниже. Обратите внимание, что полезная нагрузка не должна анализироваться до того, как не будет произведена проверка контрольной суммы, так как если вычесленная контрольная сумма не будет совпадать с той, что находится в конце пакета, то данные будут являться недостоверными и их обработка не будет иметь никакого смысла.

Контрольная сумма пакета должна использоваться для проверки целостности полезной нагрузки данных пакета и вычиляется следующим образом – сначала суммируются все байты полезной нагрузки, затем следует операция взятия младших 8 бит суммы и после этого выполняется инверсия для младших 8 бит. Программное обеспечение, должно вычислить контрольную сумму для полученных ей полезных данных и затем сравнить её с байтом контрольной суммы CRC, полученным вместе с пакетом. Если рассчитанное и полученное значения не совпадают, весь пакет следует отбросить как недействительный. Если они совпадают, то из данных полезной нагрузки можно извлекать информацию.

Сама полезная нагрузка данных состоит из непрерывной серии строк данных (DataRow). Анализ полезной нагрузки данных включает в себя анализ каждой строки данных до тех пор, пока не будут проанализированы все байты полезной нагрузки данных. Структуру пакета полезной нагрузки данных можно увидеть на рис. 2.

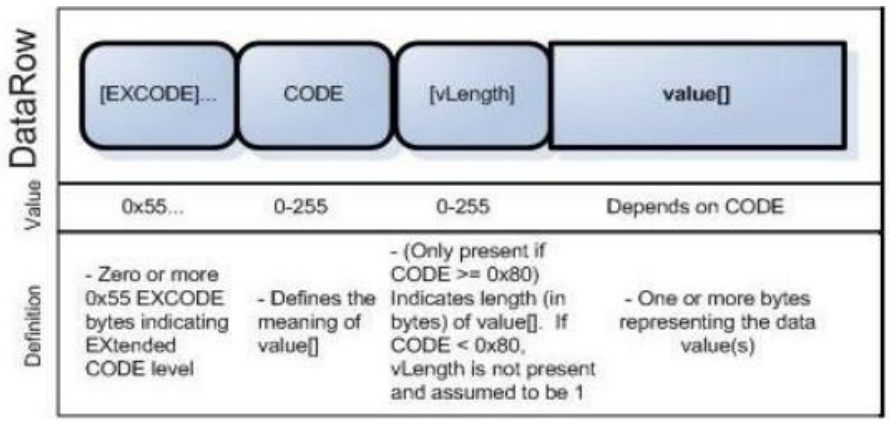

Рис. 2. Структура пакета полезной нагрузки данных

Строка данных может начинаться с нуля или более байт расширеного кода (EXCODE), с значением байта 0x55. Количество байтов расширенного кода указывает на уровень расширенного кода. Расширенный уровень кода, в свою очередь, используется вместе с байтом CODE для определения типа данных, которые содержит эта строка [3].

Если байт CODE находится между значениями 0x00 и 0x7F, то байт vLength отсутствует, а байт с полезными данными value следует сразу после байта CODE и составляет 1-байтовое значение и является концом данной строки данных. Однако, если байт CODE находится между значениями 0x80 и 0xFF, то за ним следует байт vLenght, указывающий количество байтов полезных данных value. Эти, более высокие значения CODE используются для возврата массивов значений, которые не могут поместиться в один байт, или значений, для представления которых требуется различное количество байтов. Следует отметить, что данные для построения графика ЭКГ содержатся только в байте CODE равном 0x80 имеющим длину два байта.

На основании вышесказанного, можно составить пошаговый алгоритм извлечения полезных данных из пакетов байтов, приходящих от датчика BMD101 и поступающих на Bluetooth модуль HC-05. Данный алгоритм состоит из следующих пунктов:

1. Необходимо считывать данные с датчика до тех пор, пока не встретится байт синхронихации 0xAA.
2. Если встретился байт синхронизации 0xAA необходимо прочитать следующий за ним байт и также убедиться, что он является байтом синхронизации. Если этого не произошло, необходимо вернуться к пункту 1.
3. Если программа обнаружила два следующих друг за другом байта сихнронизации, то нужно прочитать байт длины всего пакета pLength.
4. Затем следует прочитать все байты из полезной нагрузки (DataPayload), просуммировать их по мере чтения и сохранить результат в отдельную переменную.
5. После этого следует взять младшие 8 бит, получившейся суммы байтов и инвертировать их.
6. Далее нужно сравнить получившуюся контрольную сумма с той, что находится в конце пакета (CRC). Если контрольные суммы совпадают, то следует продолжить выполнение алгоритма, иначе программа возвращается к шагу 1.
7. Выполняется следующий цикл – пока все байты (и, следовательно, строки данных) не будут проанализированы из массива полезной нашрузки будут выполняться следующие действия:

   А) Программа получает данные и подсчитывает количество байтов дополнительного кода EXCODE 0x55, которые могут быть в начале текущей строки данных.

   Б) Программа извлекает байт CODE для текущей строки данных.

   В) Если возможно, программа анализирует байт vLength для текущей строки данных.

   Г) Программа получает и обработывает байт или байты value текущей строки данных на основе уровня расширенного кода EXCODE, типа данных CODE и длины этого типа данных vLength этой строки.

Д) Если не все байты были проанализированы из массива полезной нагрузки, программа вернитесь к шагу А, чтобы получить данные из следующей строки.

После того, как алгоритм разработан, следует перейти к его программной реализации. Её мы будем осуществлять с помощью языка Python в программной среде разработки Visual Studio Code. Листинг данного алгоритма показан на рис. 3

```
def parse_data(self):
    if  self.my_serial.read(1) == b'\xaa' and self.my_serial.read(1) == b'\xaa':
        judgeDataType = self.my_serial.read(1)
        if judgeDataType == b'\x04':
            self.RawDataA = bytearray(self.my_serial.read(5))
            check_sum_received = self.RawDataA.pop()
            data_sum = 0
            for val in self.RawDataA:
                data_sum += val
            checksum_calc = ~(data_sum & 0xff) & 0xff
            if check_sum_received == checksum_calc:
                self.isDataValid = True
                data_a = ((self.RawDataA[2] << 8) + self.RawDataA[3])
                self.parsedDataA = data_a & 0xffff
                if data_a >= 32768:
                    self.parsedDataA = data_a - 65536
                else:
                    self.parsedDataA = data_a
            else:
                self.isDataValid = False
            check_sum_received = 0
            checksum_calc = 0
```

Рис. 3. Листинг алгоритма извлечения полезных данных из пакета с помощью языка программирования Python

Как мы можем видеть, алгоритм сначала проверяет условие двух идущих друг за другом байтов 0xAA и если оно выполняется, то читается следующий байт, который сигнализирует о величине пакета с данными. Так как за показания типа данных ЭКГ отвечает код CODE 0x80, имеющий диапазон значений от -32768 до 32767 и имеющий длину два байта, то длина пакета с полезной нагрузкой должна быть равна четырем – первый байт отвечает за тип кода, второй байт отвечает за длину полезных данных и следующие два байта содержат в себе значение показаний ЭКГ в диапазоне от -32768 до 32767. Далее прочитывается весь пакет полезной нагрузки и контрольная сумма и эти данные помещаются в отдельный массив с байтами. Затем из этого массива байтов извлекается последний элемент, который является контрольной суммой и этот элемент записывается в отдельную переменную. После этого идет суммирование всех элементов массива полезной нагрузки взятие младших 8 бит от суммы и их инвертирование и сравнение с ранее полученной контрольной суммой. Если переменные совпадают происходит операция бинарного сдвига влево первого байта value полезной нагрузки и затем к нему добавляется второй байт value полезной нагрузки. Далее происходит операция побитового умножения суммы двух байтов на число 0xffff в шестнадцатиричной системе счисления. В двоижной же форме оно будет равно шестнадцати единицам, в десятичной оно будет равно 65535. Если получившееся число больше или равно чем 32768, то оно уменьшается на 65536 так как значение типа данных ЭКГ не может быть больше 32767.

Реализовав алгоритм работы извлечения данных, следует перейти к их графическому отображению. Наиболее полдходящей для реализации этой задачи является библиотека PyQtGraph, которая обеспечивает масштабирование графика и его перемещение во времени. Данные добавляются на график с помощью функции update, вызов которой происходит каждую миллисекунду по истеченнии таймера. Листинг данной функции изображен на рис. 4.

```
def update():
    global curve, data, ptr
    my_ecg.parse_data()
    data.append(float(my_ecg.parsedDataA))
    xdata = np.array(data, dtype='float64')
    curve.setData(xdata)
    ptr += 1
    app.processEvents()
```

Рис. 4. Листинг функции update

Переменная curve отвечает за сам график, переменная ptr служит для определения значений по оси абсцисс. После извлечения данных об уровне ЭКГ сигнала, они добавлются в массив data, затем эти данные преобразуются число с плавающей запятой и с помощью метода setData() фиксируются на графике. Затем к значению по оси абсцисс добавляется единица и вызывается метод приложения processEvents(), чтобы обновить изображение графика на экране. Также необходимо задать тот COM-порт, к которому будет подключен модуль Bluetooth HC-05, и скорость чтения данных.

Запустив данное программное обеспечение и подождав несколько секунд, на экране монитора отобразился следующий график ЭКГ человека, который можно увидеть на рис. 5.

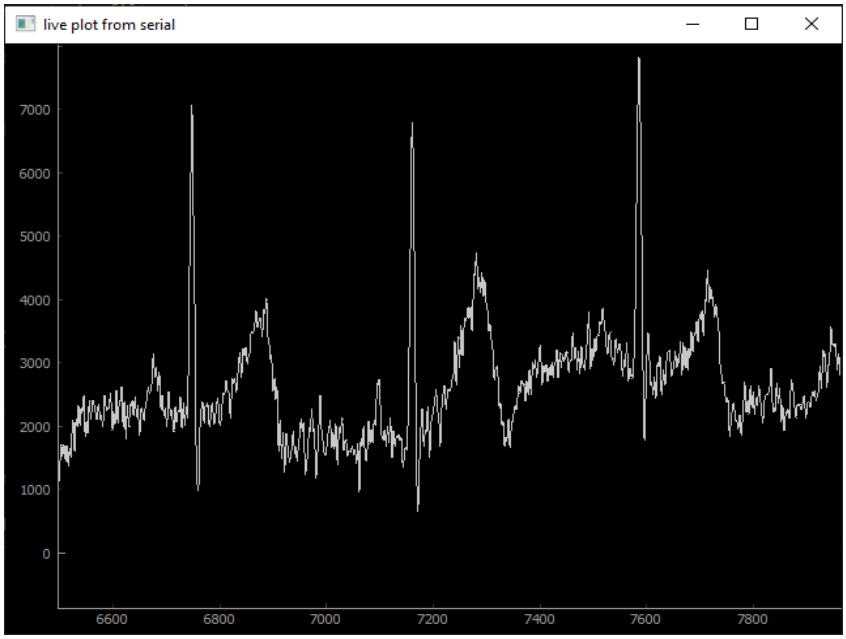

Рис. 5. График ЭКГ человека, построенный в режиме реального времени, с помощью разработанного программного обеспечения на языке программирования Python

Видим, что график получился достаточно с большиш количеством пульсаций и шумов. Для устранения данных недостатков, следует применить фильтр нижних частот, так как это наиболее оптимальный вариант. В качестве наиболее оптимального решения использовался цифровой фильтр Баттерворта с бесконечной импульсной характеристикой, максимальной пульсацией в полосе пропускания 0.01 дБ и частотой выборки равной 512 Гц. При этом частота среза фильтра составила 25 Гц, а его проядок равен 5 с частотой выборки 512 Гц. Результат фильтрации сигнала в режиме реального времени представлен на рисунке 6.

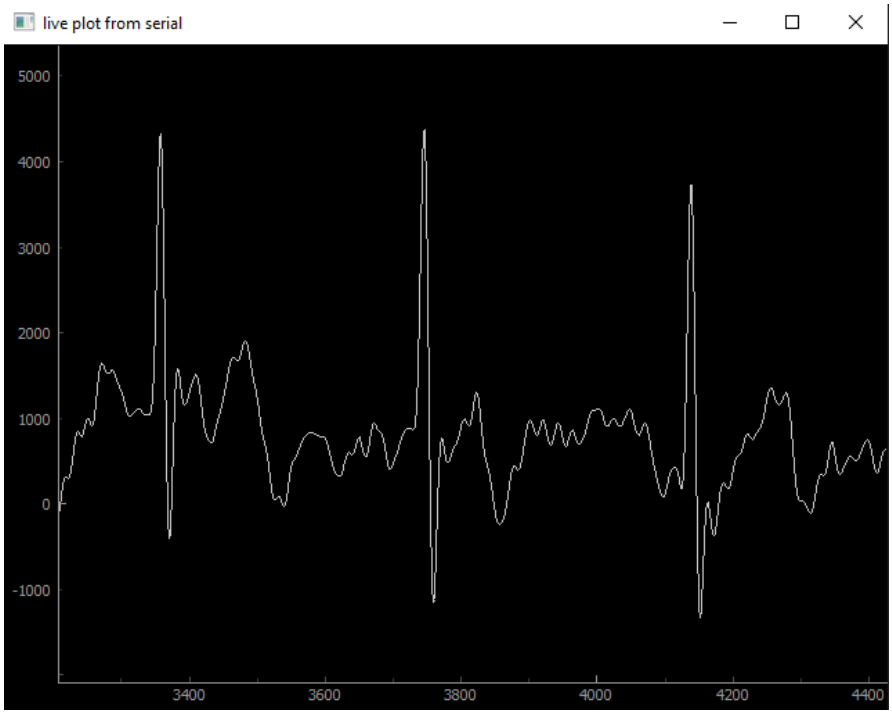

Рис. 6. Результат фильтрации сигнала ЭКГ человека с помощью применения цифрового фильтра нижних частот

Заключение

В результате проведенного исследования реализован программно-аппаратный комплекс считывания данных кардиограммы человека в режиме реального времени. Разработан алгоритм извлечения данных а также осуществлена его программная реализация с помощью одного из языков программирования.

Список использованных источников

1. Журавлев Д.В. Аппаратура для электроэнцефалографических исследований: монография / Д.В. Журавлёв; ФГБОУ ВО «Воронежский государственный технический универси-тет». – Воронеж: Изд-во ВГТУ, 2021. – 258 с.

2. Журавлев, Д. В. Система комплексной экспресс-оценки функциональной готовности человека / Д. В. Журавлев, А. А. Проводников // Вестник Воронежского государственного технического университета. – 2021. – Т. 17. – № 3. – С. 121-126.

3. URL: https://datasheetspdf.com/pdf-file/836731/NeuroSky/BMD101/1 (Дата обращения: 07.11.2022).