УДК 681.32

А.Б. Бережной, С.В. Теплинский
Донецкий национальный технический университет
кафедра компьютерных наук и технологий

РАЗРАБОТКА ПРОШИВКИ БЛОКА РЕГИСТРАЦИИ ДВИЖЕНИЙ И УПРАВЛЯЮЩЕГО ПО

Аннотация

А.Б. Бережной, С.В. Теплинский. ***Разработка прошивки блока регистрации движений и управляющего ПО.*** *Выполнен обзор разработки прошивки и управляющего ПО блока регистрации движений, являющегося составной частью медицинского тренажера для восстановления координации движений.*

Ключевые слова: *функциональная схема, диспетчер, акселерометр, ДУС, фильтрация.*

Введение. В данной работе описывается процесс разработки прошивки для блока регистрации движений, входящего в состав тренажера для восстановления и диагностики функций опорно-двигательного аппарата. Концепция данного тренажера впервые была представлена на конференции 2013 года [1]. На данный момент блок регистрации движений практически полностью завершен и проводится его тестирование.

Структурная схема блока регистрации движений и принцип действия.

На рисунке 1 изображена функциональная схема блока регистрации движений и само устройство.

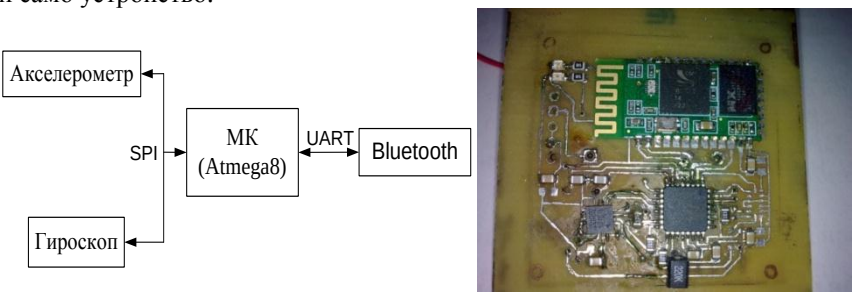

Рисунок 1 – Функциональная схема и готовое устройство

Блок построен на контроллере Atmega8L [2] и предназначен для замеров и фиксации параметров перемещения руки в пространстве. В качестве датчиков было выбрано два прибора: гироскоп и акселерометр. С помощью

сенсоров (гироскоп, акселерометр) выполняется контроль за перемещением руки в пространстве.

Основные функции устройства. Устройство должно выполнять следующие базовые функции:

1. Установка связи с ПК по интерфейсу Bluetooth-COM.
2. Предварительная настройка параметров датчиков.
3. Передача текущих настроек по радиоканалу к ПК для их конфигурирования.
4. Получение новых настроек от ПК и их установка.
5. Периодический опрос акселерометра и гироскопа.
6. Передача данных по радиоканалу от блока к основной станции.
7. Контроль заряда батареи.
8. Сохранение блока текущих настроек в энергонезависимой памяти МК.

Функции в данном списке расположены в порядке приоритета их реализации. Это объясняется спецификой устройства, поскольку например, без реализации функций 1, 2 и 3 проверить реализованную функцию 4 будет невозможно.

Разработка прошивки модуля. Разработка прошивки подобного устройства задача далеко не простая и требует тщательной проработки ее архитектуры. Иначе построение программы без четкой идеологии рано или поздно приведет к тому, что программу станет очень тяжело модифицировать и вносить изменения, и она обрушится под собственной сложностью. Из возможных вариантов реализации прошивки было решено остановиться на диспетчере. Данная организация программы требует небольшого количества управляющего кода, позволяет иметь динамическую очередь задач и набор программных таймеров для отложенных задач.

Основное достоинство – позволяет довольно быстро вносить изменения в код прошивки, добавлять и удалять задачи, а также существенно упрощает процесс отладки, так как каждая задача может быть отлажена отдельно.

Недостаток – если заносить задачи в очередь быстрее, чем они успевают отрабатывать, то может случиться так называемый срыв очереди и программа зависнет.

Работа на командах. Интерпретатор команд. Принимая в расчет, что блок регистрации движений не является самостоятельным устройством, а выполняет команды от основного блока, была разработана система команд, позволяющая управлять его работой. Также был разработан программный интерпретатор команд. Было решено реализовать каждую команду, которая может прийти в устройство, как отдельную процедуру со строго фиксированным номером, а для получения интерпретатора реализовать прием кода команды через прерывание RS-232 интерфейса и процедуру обработчик кода команды.

Также был учтен тот факт, что помимо команд устройство должно принимать байты настроек для датчиков, поэтому прерывание по приему данных разветвляется на две ветви: прием команды и прием данных. Алгоритм работы прерывания изображен на рисунке 2.

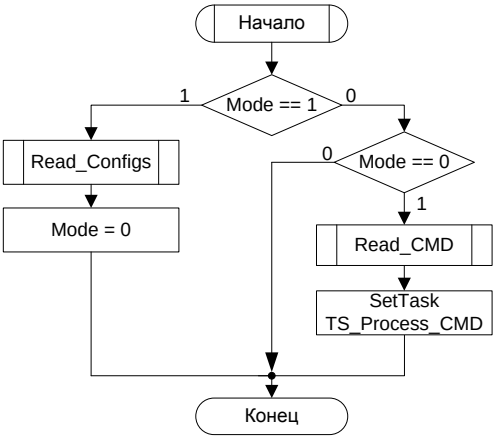

Рисунок 2 – Блок схема работы прерывания приема байта по RS232

После сохранения команды в оперативной памяти микроконтроллера, в очередь задач с помощью макроса SetTask заносится процедура – обработчик команды TS_Process_CMD. При вызове на следующем такте она проанализирует код поступившей команды, и либо сразу поставит в очередь соответствующую команде процедуру обработки, либо сначала подготовится к приему данных для этой команды от основного блока через прерывание. Во втором случае процедура обработки будет поставлена в очередь после приема всего блока данных.

Взаимодействие устройства с ПК. Блок регистрации движений не является самостоятельным устройством, он может лишь выполнять команды основного блока. Поэтому для проверки его работы необходимо разрабатывать основной управляющий блок, либо специализированное ПО для компьютера. Выбор был сделан в пользу второго варианта, т.к. разработка ПО процесс более быстрый, чем проектировка основного блока.

Разработка управляющего ПО. Управляющая программа должна в первую очередь предоставлять пользователю удобный интерфейс для взаимодействия с устройством и в то же самое время максимально скрывать от него любые внутренние подробности работы. Такой подход позволяет максимально упростить работу пользователя с ПО и самим устройством.

Основные функции управляющей программы:

1. Соединение с блоком регистрации движений по команде пользователя.
2. Вывод текущих настроек датчиков в удобочитаемом формате.
3. Редактирование текущих настроек.
4. Установка новых настроек в устройство.
5. Получение данных с датчиков через определенные промежутки времени.
6. Сбор данных для последующего анализа.

Для написания программы было принято решение использовать язык C++ и библиотеку MFC, которая представляет собой реализацию базовых классов WinAPI. Основным критерием выбора была возможность относительно легкой работы с периферией, в частности с COM – портом, т.к. со стороны ПК устройство идентифицируется как COM – порт. Также MFC позволяет легко и быстро создавать и настраивать удобный графический интерфейс пользователя, что является несомненным плюсом.

Саму программу решено было разрабатывать с применением основных принципов ООП, то есть разбить ее на несколько функционально законченных блоков и каждый блок реализовать как отдельный класс.

В первую очередь было разработано основное окно программы, изображенное на рисунке 3. Окно состоит из меню, через которое выполняется управление и двух областей для вывода данных от гироскопа и акселерометра соответственно.

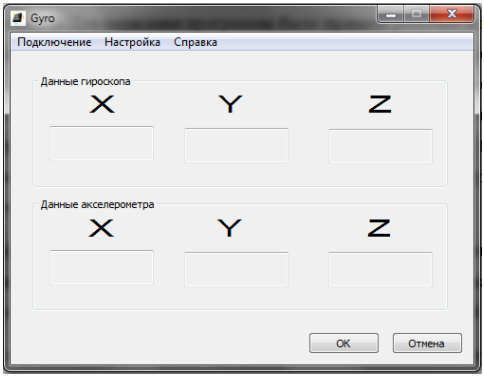

Рисунок 3 – Внешний вид основного окна управляющей программы

Затем была реализована функция установки и разрыва соединения с устройством. С точки зрения программы подключение к устройству – это просто открытие COM порта функциями WinAPI. С целью упростить работу с COM портом был разработан специальный класс, который позволяет открыть и настроить порт с указанным именем и параметрами. Также данный класс позволяет работать с портом через асинхронные запросы. Для этого создается дополнительный поток, который запускается по событию приема байта от

устройства. Таким образом, можно избавиться от пустых циклов ожидания приема байтов, а выполнять прием только когда в буфере есть данные.После установки подключения программа выдаст соответствующее сообщение, а на устройстве загорится зеленый светодиод.

Редактирование настроек устройства. Следующим шагом была реализация возможности просмотра и редактирования настроек датчиков устройства. Настройки представляют собой ячейки в памяти датчика, куда заносятся байты, где значение битов определяют текущую конфигурацию. С полным перечнем настроек можно ознакомиться в справочной документации на соответствующий датчик. Были разработаны диалоговые окна, где в двоичной системе отображается текущее значение настроек.

Чтение данных с устройства. Данные получаемые от устройства представляют собой 3 двухбайтных значения угловой скорости с гироскопа и 3 значения проекций силы тяжести с акселерометра. Данные считываются с датчиков с частотой 20Hz, то есть каждые 50 мс. Чтение реализуется целиком внутри устройства, диспетчер сам каждые 50 мс вызывает процедуру, которая считывает данные с гироскопа и акселерометра и через прерывание по передаче байта через RS-232 интерфейс отправляет эти данные в основной блок.

Принципы получения данных о положении устройства. Чтобы понять, как получить информацию о текущем положении устройства, необходимо вначале выяснить, что же показывают на выходе датчики и их принцип действия.

MEMS гироскопы, отличаются от традиционных гироскопов тем, что:

- они не содержат вращающихся частей;
- они являются датчиками угловой скорости (ДУСами);
- они обладают сравнительно невысокой точностью.

При вращении ДУСа вокруг какой-либо оси возникает сила Кориолиса, которая отклоняет обкладку конденсатора, соответственно меняется емкость конденсатора, которая измеряется внутренним АЦП. Сила Кориолиса зависит от угловой скорости вращения, соответственно на выходе датчика получаем угловую скорость. Зная угловую скорость, можно получить угол, для этого необходимо ее интегрировать, а т.к. данные поступают дискретно (через промежуток dt = 50мс) то достаточно просто суммировать показания угловой скорости, умноженные на 50 мс, что является приращением угла за dt.

Однако существуют два важных фактора, которые необходимо учитывать:

- Уход ДУСа - это отклонение точки нуля гироскопа.
- Шум ДУСа - это некоторое быстро изменяющееся случайное слагаемое, которое присутствует в выходном сигнале.

В результате получается, что при интегрировании показаний с гироскопа даже при неподвижном устройстве будет накапливаться ошибка за счет шума и

отклонения от точки нуля, в результате чего определяемый угол очень быстро «уходит» от реального значения. Поэтому использовать один лишь гироскоп для определения положения устройства в пространстве нельзя.

Акселерометр. Измеряет проекцию кажущегося ускорения на ось чувствительности. То есть по каждой оси чувствительности акселерометр выдает проекцию силы тяжести и абсолютного ускорения.
Если система находится в статическом положении или движется с постоянной скоростью (собственное ускорение равно нулю), то углы наклона рассчитывается по следующим формулам (для трехосевого случая):

$$\alpha = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right) \quad \beta = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right) \quad \gamma = \arctan\left(\frac{A_z}{\sqrt{A_y^2 + A_x^2}}\right)$$

Также необходимо учитывать, что акселерометр выдает относительно точные значения только в состоянии покоя, т.к. в акселерометре тоже присутствуют шумы. В движении к показаниям прибавляется проекция вектора собственного ускорения и значение угла «уплывает».
В результате можно сделать вывод, что совместная работа двух датчиков способна дать весьма неплохой результат, то есть с помощью гироскопа определяются неточные значения углов, а затем корректируются с помощью акселерометра. Но даже в этом случае скорректировать можно только два угла, т.к. угол поворота вокруг оси Z через акселерометр определить невозможно.

Калибровка датчиков. Как уже было сказано, оба датчика подвержены шумам и т.н. дрейфу нуля, а это может вносить в показания существенную ошибку даже при использовании сразу двух датчиков. И если шумы практически невозможно подавить, то проблему дрейфа нуля хорошо решает калибровка. Алгоритм калибровки выглядит следующим образом:

- Расположение устройства в строго горизонтальном положении
- Произведение 500 замеров с датчиков
- Расчет среднего значения нуля по каждой из осей датчика.

Полученные после калибровки значения нуля необходимо просто вычитать из показаний соответствующих осей. Калибровка может выполняться как единожды за весь цикл работы, так и перед каждым измерением.

Использование фильтров. Получив значения углов с гироскопа и акселерометра, требуется их «смешать», чтобы получить истинный угол. Для этого применяют фильтры, самыми распространенными являются комплиментарный фильтр и фильтр Калмана [3]. Фильтры могут быть реализованы различными способами, например через матрицы, или кватернионы.

Результаты работы устройства. В ходе работы были реализованы оба варианта фильтров и оба показали весьма неплохие результаты, в ходе анализа полученных было выявлено, что фильтр Калмана несколько точнее альфа-бета фильтра. Это объясняется более сложным алгоритмом его работы. Результат работы фильтра Калмана представлен на рисунке 4.

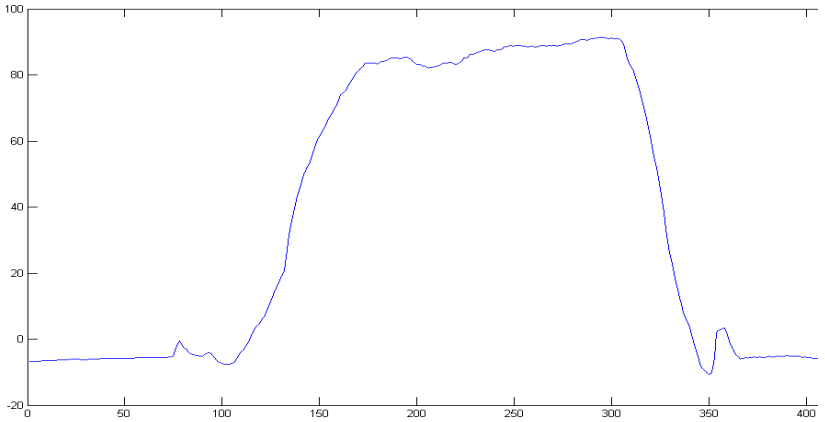

Рисунок 4 – Результат работы устройства для оси X

Ось Y на графике представляет собой угол отклонения по оси в глобальной системе координат, а ось X – количество замеров.

Вывод. Разработка цифрового устройства на микроконтроллере процесс весьма трудоемкий, требующий знаний и умений не только в области электроники. В ходе работы было реализовано несколько весьма полезных классов, которые могут быть использованы в дальнейших разработках, а также прошивка устройства, отдельные блоки которой также могут быть использованы в других проектах.

Список литературы

1. Разработка устройства-тренажера для восстановления координации движений//Інформаційні управляючі системи та комп'ютерний моніторинг (ІУС КМ - 2013) : IV Всеукраїнська науково-технічна конференція студентів, спірантів та молодих вчених, 24-25 квітня 2013 р., м.Донецьк зб. доп. / Донец. національ. техн. ун-т; редкол. В.А. Світлична. – Донецьк: ДонНТУ, 2013. – В 2 тт. - Т.1. 765 с. 118-123.

2. Евстифеев, А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. — М.:Издательский дом «Додэка-XXI», 2007. — 592 с: ил.

3. Описание фильтра Калмана: http://www.ocf.berkeley.edu/~tmtong/howto/kalman/writeup.pdf