Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и Системы Управления
КАФЕДРА Программное Обеспечение ЭВМ и Информационные Технологии

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

Загружаемый модуль ядра ОС семейства GNU/Linux для работы с гиростабилизатором GY80

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____
(Подпись, дата) (И.О.Фамилия)

Москва, 2016

Содержание

Реферат

Объект исследования и разработки

Объектом разработки является загружаемый модуль ядра ОС семейства GNU/Linux для работы с гиростабилизатором GY80.

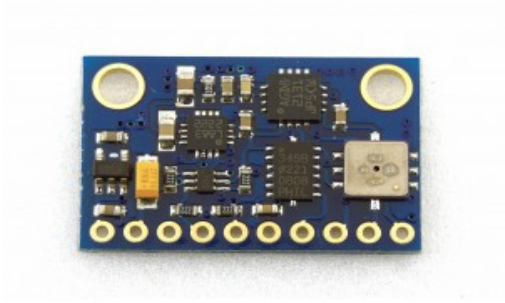

Рис. 1 — GY80

Модуль ядра взаимодействует с гиростабилизатором GY80 посредством обмена данными по шине данных I²C. GY80 включает в себя акселерометр (Analog Devices ADXL345), гироскоп (ST Microelectronics L3G4200D), магнитометр (Honeywell MC5883L) и барометр (Bosch BMP085). Модуль ядра создает виртуальное устройство /dev/gy80, при чтении с которого модуль передает показания сенсоров пользователю.

Цель и задачи работы

Цель работы — разработать является загружаемый модуль ядра ОС семейства GNU/Linux для работы с гиростабилизатором GY80.

Модуль ядра должен взаимодействовать с гиростабилизатором GY80 посредством обмена данными по шине данных I²C. GY80 включает в себя акселерометр (Analog Devices ADXL345), гироскоп (ST Microelectronics L3G4200D), магнитометр (Honeywell MC5883L) и барометр (Bosch BMP085). Модуль ядра должен создать виртуальное устройство /dev/gy80, при чтении с которого модуль должен будет передавать показания сенсоров пользователю.

В качестве примера работы с этим модулем необходимо также реализовать клиент-серверное приложение для отображения положения ПК в пространстве.

На серверной части необходимо периодически считывать показания сенсоров из /dev/gy80, с помощью алгоритмов AHRS преобразовывать эти показания к кватерниону положения и отправлять кватернион всем подключенным клиентам.

На клиентской части кватернион необходимо принять и отобразить трехмерную модель в соответствии с этим кватернионом.

Должны быть применены и по необходимости доработаны известные существующие алгоритмы, исследована математическая модель. Необходимо добиться от приложения максимальной производительности при максимальном качестве и функциональности, применив при этом наиболее современные технологии.

Результаты работы

В результате был разработан загружаемый модуль ядра ОС семейства GNU/Linux для работы с гиростабилизатором GY80.

Модуль ядра взаимодействует с гиростабилизатором GY80 посредством обмена данными по шине данных I²C, создает виртуальное устройство /dev/gy80, при чтении с которого передаются показания сенсоров пользователю.

В качестве примера работы с этим модулем также было реализовано клиент-серверное приложение для отображения положения ПК в пространстве.

На серверной части периодически считываются показания сенсоров из /dev/gy80, с помощью алгоритмов AHRS преобразуются к кватерниону положения, который отправляется всем подключенным клиентам.

На клиентской части кватернион необходимо принимается, и отображается трехмерная модель в соответствии с этим кватернионом.

Исследованы последние существующие алгоритмы, исследована математическая модель. Приложения имеет достаточно высокую производительность по сравнению с аналогами при максимально точном отображении положения в пространстве, были применены наиболее современные технологии.

Основные конструктивные, технологические и технико-эксплуатационные характеристики

Разработка велась в следующих условиях:

- ПК: Raspberry Pi (Model B) с процессором ARMv6 (BCM2835).
- ОС: GNU/Linux, Arch Linux ARM.
- Версия ядра: 4.4.35-1.

Эксплуатация возможна в следуюих условиях:

- Любой ПК, имеющий I²C шину для работы с устройствами.
- ОС: любая ОС из семейства GNU/Linux.
- Версия ядра: не ниже 2.6.

Для эксплуатации не важно какой тактовой частотой обладает процессор, не важно какой объем оперативной памяти имеет ПК, т. к. подразумевается, что если устройство способно функционировать на ОС семейства GNU/Linux, то оно обладает достаточными ресурсами для функционирования данного модуля ядра.

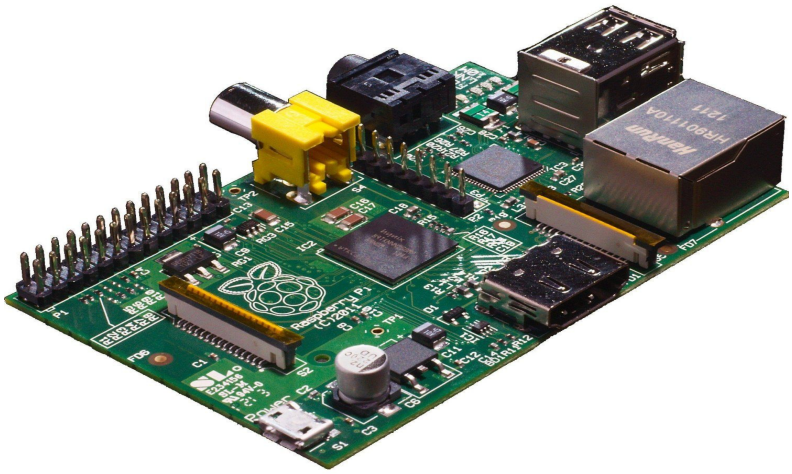

Рис. 2 — Raspberry Pi Model B.

Прогнозы и предположения о развития объекта исследования

Модуль ядра является законченным продуктом, но были задействованы не все возможности гиростабилизатора GY80, поэтому остаются не рассмотренными некоторые второстепенные детали.

1. Показания L3G4200D остаются неточными:
    1. возможно стоит исследовать работу встроенного High Pass фильтра сенсора L3G4200D;
    2. возможно стоит исследовать влияние электрического поля, создаваемого другими элементами Raspberry Pi или GY80 на зашумление данных.
2. В некорых местах, после отправления команды какому-либо сенсору, необходимо дождаться формирования готовых данных в определенном регистре сенсора — для этого стоит использовать прерывания на других регистрах сенсора, но в коде модуля пока используется ожидание на несколько микросекунд.

Введение

Целью ряботы является загружаемый модуль ядра ОС семейства GNU/Linux для работы с гиростабилизатором GY80. Для реализации цели необходимо решить следующие задачи:

- Аналитические:
  - анализ и выбор возможных существующих подходов:
    - для проектирования модуля ядра,
    - для осуществления коммуникации с устройством;
  - анализ и выбор алгоритма AHRS.
- Конструкторские:
  - разработка модуля ядра;
  - разработка демонстрационного приложения.
- Технологические:
  - выбор технологий для разработки модуля ядра;
  - выбор технологий для разработки демонстрационного приложения.
- Экспериментальные:
  - исследование максимально возможной скорости чтения с устройства GY80;

Аналитический Раздел

Проектирование модуля ядра

Загружаемый модуль ядра (loadable kernel module, LKM) — объектный файл, содержащий код, расширяющий возможности ядра операционной системы. Модули используются, чтобы добавить поддержку нового оборудования или файловых систем или для добавления новых системных вызовов. Когда функциональность, предоставляемая модулем больше не требуется, он может быть выгружен, чтобы освободить память и другие ресурсы.

Любой модуль ядра проектируется классическим образом: он состоит из функций, которые вызываются на соответствующем событии (инициализация, выход и т. д.). Наш модуль также должен будет создавать виртуальное устройство — /dev/gy80 — поэтому также будет содержать функцию, которая будет вызываться на попытке чтения /dev/gy80 пользователем. Помимо этого выделем отдельно набор утилитарных функций для работы с подустройствами gy80.

Осуществление коммуникации с устройством

Единственный способ коммуникации, поддерживаемый GY80 — коммуникация через шину данных I²C.

I²C (IIC, англ. Inter-Integrated Circuit) — последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов. Использует две двунаправленные линии связи (SDA и SCL), применяется для соединения низкоскоростных периферийных компонентов с процессорами и микроконтроллерами (например, на материнских платах, во встраиваемых системах, в мобильных телефонах).

Анализ и выбор AHRS

AHRS (Attitude and Heading Reference System) — система определения курса и пространственного положения. По своим возможностям, такие системы приближаются к инерциальным навигационным системам, но отличаются от них большей простотой и меньшей стоимостью.

В основе таких систем лежит фильтрующий алгоритм. На данный момент широко известны открытый алгоритм Калмана, алгоритм Мажвика, алгоритм Махони.

Алгоритм Калмана является одним из самых ранних алгоритмов в этой области. Алгоритм Мажвика представляет собой несколько улучшенную версию алгоритма Калмана, относительно недавно был опубликован в одной статье в двух вариациях — с учетом показаний магнитометра и без. Он относительно хорошо справляется с зашумленными показаниями сенсоров, но является недостаточно производительным для его применения в промышленности или военной индустрии. Он отлично подходит для нашей задачи, т. к. нам не требуется сверхвысокая производительность и мы используем одни из самых простых и дешевых сенсоров. Ниже приводятся графики из оригинальной статьи, которые показывают различие результатов работы между этим алгоритмом и алгоритмом Калмана:

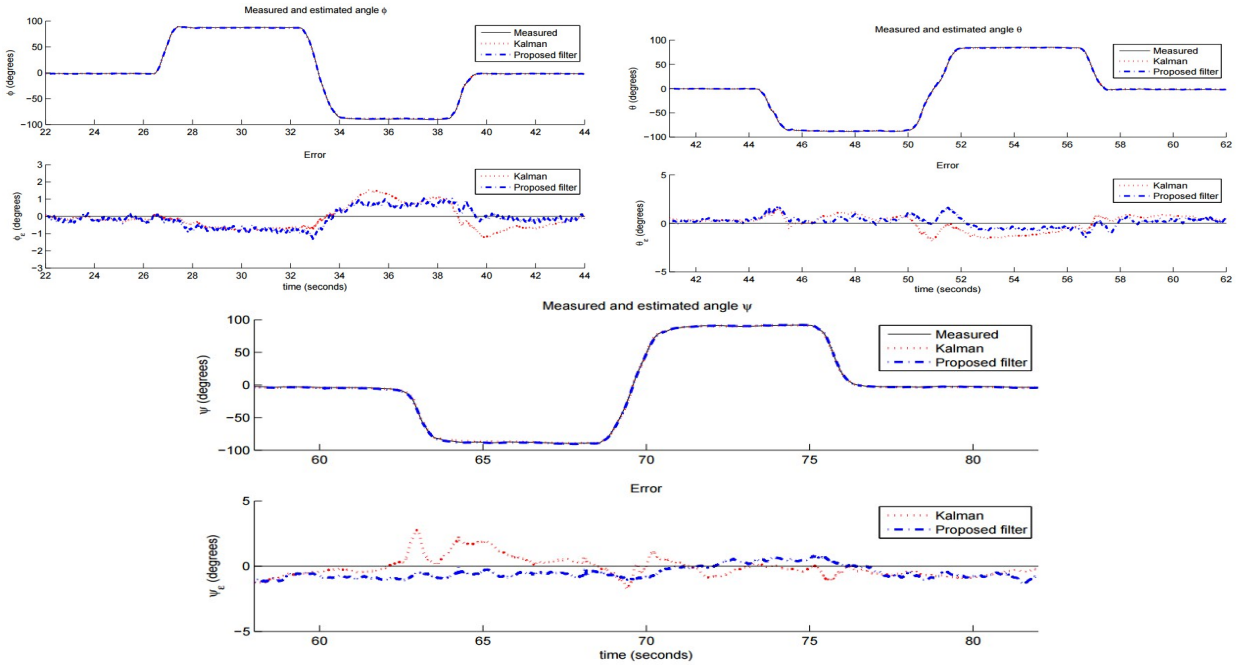

Рис. 3 — графики ошибки для алгоритмов Мажвика и Калмана

Конструкторский раздел

Разработка модуля ядра

Разработка модуля ядра велась классическим типовым подходом для разработки модулей ядра. Единственной конструктивной особенностью являлось то, что из модуля ядра необходимо было обращаться к I²C шине. Это вызвало некоторые сложности.

Как известно, в пространстве ядра запрещено использовать символы из пространства пользователя. Разрешено использовать только символы ядра, причем только те, которые экспортированны. Символы экспортируются с помощью специального макроса *EXPORT_SYMBOL* или *EXPORT_SYMBOL_GPL*. Во втором случае символы будут доступны только если разрабатываемый модуль соответствует лицензии GPL.

В большинстве современных версий ядра символы *sys_open, sys_write, sys_read, sys_ioctl, sys_close* больше не экспортируются в связи с соображениями безопасности. Единственный способ разрешить их использование — исправить исходный код ядра и пересобрать его. Но это разрешит их использование лишь на нашем ядре, то есть на других системах модуль все равно не будет работать. Поэтому нужно было другое решение.

Однако в современных версиях ядра экспортируется символ:

```
unsigned long kallsyms_lookup_name(const char *name);
```

Эта функция позволяет получить абсолютный адрес символа, если ядро о нем знает. Таким образом можно получить адреса всех необходимых нам функций, написать им соответствующий прототип и использовать в пространстве ядра, предварительно переключив область адресов на пользовательскую. Таким образом можно использовать любую функцию в пространстве ядра:

```
asmlinkage long sys_ioctl(unsigned int fd, unsigned
        int cmd, unsigned long arg);

asmlinkage long sys_write(unsigned int fd, const
      char __user *buf, size_t count);

asmlinkage long sys_open(const char __user
     *filename, int flags, umode_t mode);
```

```
asmlinkage long sys_read(unsigned int fd, char
__user *buf, size_t count);

asmlinkage long sys_close(unsigned int fd);
```

Теперь коммуникация с устройством GY80 осуществляется через шину I²C с помощью функций *sys_open, sys_write, sys_read, sys_ioctl, sys_close*. Принцип работы с устройством. Сначала каждое подустройство настраивается — в определенные регистры записываются значения в соответствии со спецификацией (например, включение устройства, установка масштаба для гироскопа и т.д.). Затем происходит периодическое считывание значений с подустройств — результирующие данные находятся в известном регистре в известном формате.

Разработка демонстрационного клиент-серверного приложения

Полученные данные от сенсоров сами по себе не несут никакого прикладного смысла. Чтобы их соотнести с реальностью, достаточно передать их в AHRS, чтобы получить информацию о положении ПК в пространстве.

Но наш ПК, как многие встроенные системы для решения утилитарных задач, не имеет дисплея, и подключать его — не самое рациональное решение. Гораздо интереснее разработать клиент-серверное приложение, в котором информация о положении в пространстве передавалась бы подключенным клиентам, которые уже своими средставми могли бы визуализировать полученные данные.

Для необходимой скорости передачи данных между сервером и клиентами будет открываться дополнительное соединение через WebSocket. Требуется передача данных на скорости порядка 25 раз в секунду. Этого сложно добиться с помощью классических AJAX-запросов. К тому же, с каждым AJAX-запросом пересылается слишком много сервисных данных.

Поэтому было разработано небольшое клиент-серверное приложение.

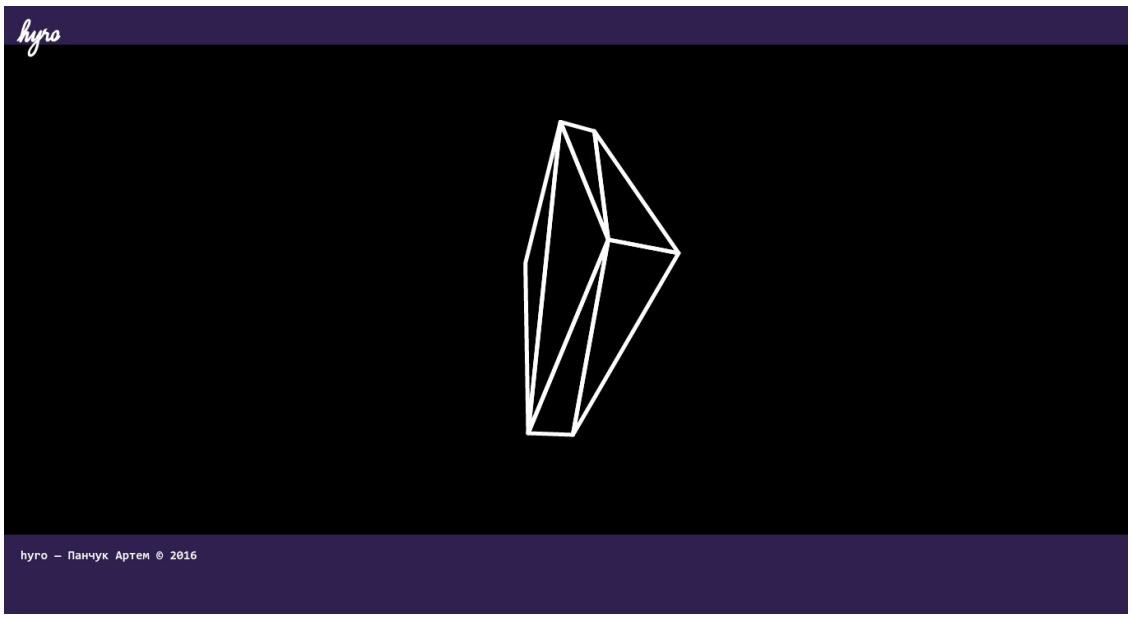

Рис. 4 — снимок экрана работы демонстрационного приложения

Технологический раздел

Выбор технологий для разработки модуля ядра

Модуль ядра разрабатывается на языке C с использованием стандарта c99. Этот язык является минималистичным и идеально подходит для решения задачи.

Помимо средств, предложенных операционной системой, никакие библиотеки не используются, т. к. функциональности встроенных средств достаточно.

Коммуникация с устройством GY80 осуществляется через шину I²C с помощью функций из *linux/syscalls.h: sys_open, sys_write, sys_read, sys_ioctl, sys_close.*

В пространстве ядра запрещены операции с вещественными числами.

Выбор технологий для разработки демонстрационного приложения

Демонстрационное приложение имеет клиент-серверную архитектуру.

Серверная часть запускается на Raspberry Pi. Язык реализации — JavaScript (ES 5). Используется фреймворк Node.js. При подключении каждого клиента, открывается отдельный сокет — WebSocket. Поэтому на серверной части формально запущено два сервера.

Клиентская часть работает в браузере, поэтому реализована также на JavaScript. Для визуализации используются средства WebGL с поддержкой аппаратного ускорения и библиотека Three.js.

Экспериментальный раздел

После реализации модуля ядра и демонстрационного приложения возникла задача оценить скорость считывания данных модулем ядра с устройства GY80. Как очевидно, нельзя добиться бесконечной скорости считывания с устройства. Поэтому проведем эксперимент — начнем считывать данные с частотой 1 раз в секунду, постепенно увеличивая частоту. Затем установим максимально возможную частоту считывания, быстрее которой читать данные уже не получится.

Выяснилось, что максимальная частота находится примерно около 50 Гц. Был получен график:

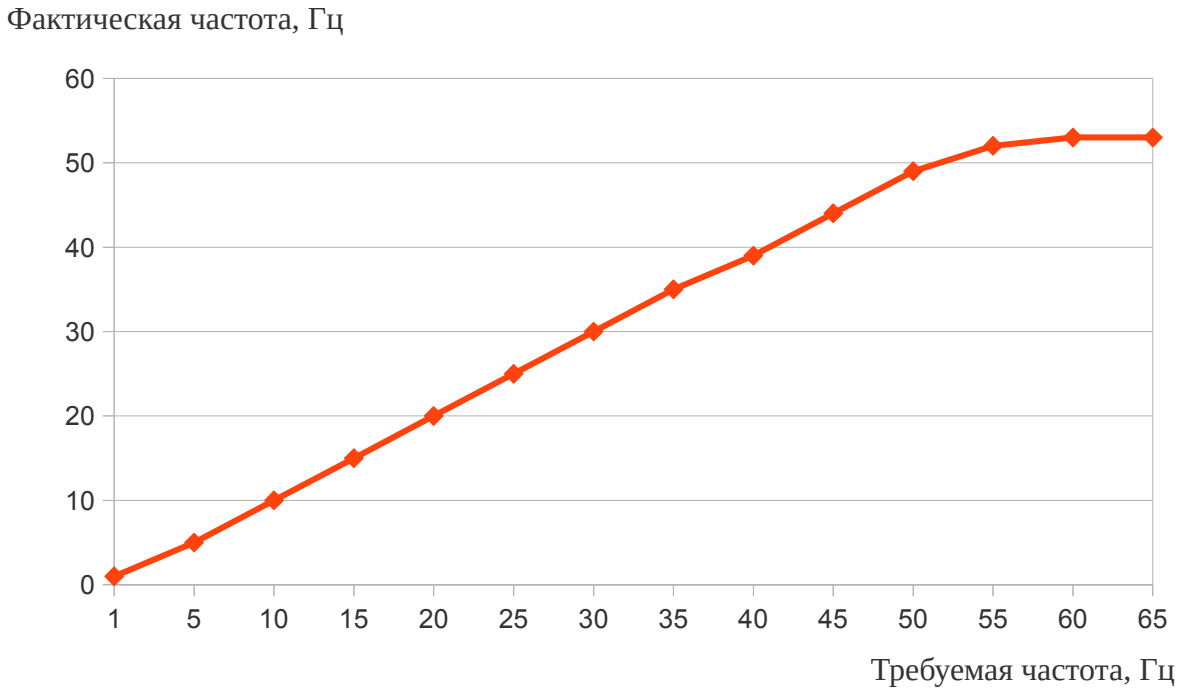

Рис. 5 — зависимость фактической частоты от требуемой

Заключение

Реализован загружаемый модуль ядра ОС семейства GNU/Linux для работы с гиростабилизатором GY80 и клиент-серверное демонстрационное приложение.

Код модуля и приложения находится в открытом доступе на личном GitHub репозитории.

Были реализованны все пункты из технического задания.

Список литературы

1. L3G4200D Datasheet (http://www.st.com/content/ccc/resource/technical/document/datasheet/04/46/d6/00/be/d9/46/ae/CD00265057.pdf/files/CD00265057.pdf/jcr:content/translations/en.CD00265057.pdf);
2. ADXL345 Datasheet (http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf);
3. HMC5883L Datasheet (https://aerocontent.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf);
4. BMP085 Datasheet (https://www.sparkfun.com/datasheets/Components/General/BST-BMP085-DS000-05.pdf);
5. Перевод оригинальной статьи Мажвика (https://habrahabr.ru/post/255661/);
6. Лекции по курсу «Операционные Системы» МГТУ им. Баумана.