Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт Информационных Технологий Математики и Механики

Отчёт по конкурсу студенческих программных проектов

**Концепт «умного» дома**

**на примере камина**

Выполнил:

студенты ИИТММ гр. 0826-2

Моренков А.С

Липатов И.Д

Проверил:

Баркалов К.А.

Шестакова Н.В

Нижний Новгород

2016 г.

Содержание

[Введение 3](#_Toc438497992)

[Постановка задачи и цели работы 4](#_Toc438497993)

[Методы решения задачи 4](#_Toc438497994)

[Программная реализация 5](#_Toc438497995)

[Результаты работы 5](#_Toc438497998)

[Руководство пользователя 7](#_Toc438497999)

[Заключение 8](#_Toc438498000)

[Список литературы 3](#_Toc438497992)

[Приложения 4](#_Toc438497993)

# Введение и постановка задачи

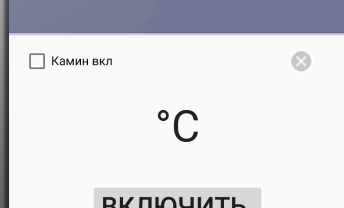
# Идея «умного» дома – довольно молодая. Когда она пришла на рынок, то сразу же получила популярность за своё удобство. Единственный минус заключается в том, что готовых решений сравнительно мало и стоят они дорого. Поэтому готовые продукты не подходили нам как решение главной нашей проблемы – использование летом на даче. В итоге мы решили написать свою программу ,т.к этот вариант подходит нам по всем необходимым параметрам:  1) низкий порог вхождения  2) относительная дешевизна (относительно готовых решений)  3) возможность опробовать на практике навыки программирования микроконтроллеров и написания прикладных программ для ОС Android.

# Методы решения задачи

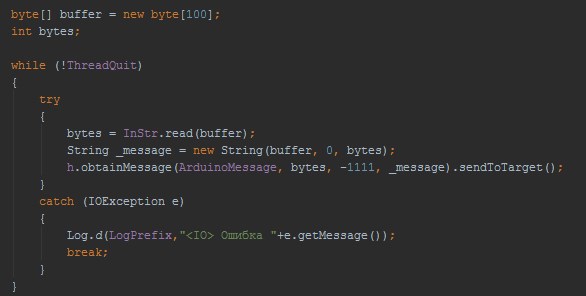
# Программная реализация

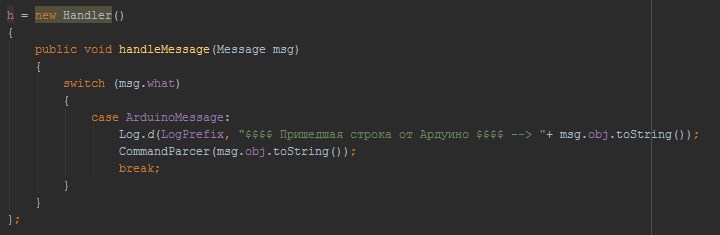
Android

1) Многопоточность

В приложении есть несколько потоков – первый Ch\_Thread – создаётся при переводе главного экрана (далее – активити) в положение OnResume после создания сокета (сокет – програмный интерфейс для обеспечения обмена данными между процессами). Это не тормозит активити, если доступа к Ардуино нет – в этом потоке раз в 3 секунды пытается соединиться с Ардуино по данному MAC и UUID. Если соединение прошло успешно, то поток создает необходимые для нормальной работы потоки и завершается. Если нет – засыпает на 3 секунды и пробует снова.  
Об успешности подключения к Ардуино можно судить по лампочке в правой верхней части экрана – если она серая с крестом, то ардуино оффлайн, если зелёная – онлайн.

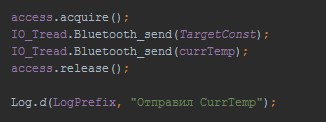
После удачного подключения к Ардуино запускается ещё два потока – R\_Thread и IO\_Thread. R\_Thread поток раз в 5 секунд спрашивает Ардуино о текущей температуре и засыпает. IO\_Thread – поток, к которому подключен буфер Bluetooth-адаптера а так же возможность отправки команд по Bluetooth.  
Через этот поток посылается множество запросов к контроллеру (кроме того, что отдельно функционирует в R), а так же «разгребает» всё, что прислала ему ардуино.

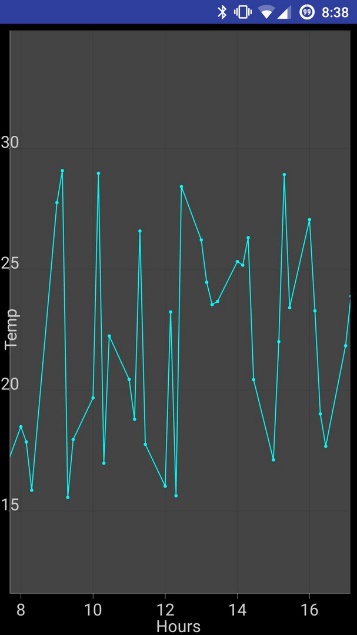
В этом и кроется главная проблема такого подхода – данные приходят в сторонний поток (IO\_Thread) а должны быть записаны в главное окно программы (имеется ввиду пришедшая температура, которая отображается на экране). Сторонние (по отношению к главному потоку программы) потоки не имеют доступа к элементам экрана, поэтому пришлось использовать handler. Handler - это механизм, который позволяет работать с очередью сообщений. Он привязан к конкретному потоку (в данном случае – к главному) и работает с его очередью. Handler умеет помещать сообщения в очередь. При этом он ставит самого себя в качестве получателя этого сообщения. И когда приходит время, система достает сообщение из очереди и отправляет его адресату (т.е. в Handler) на обработку. На рисунке показан фрагмент кода у IO\_Thread – он читает из Bluetooth буфера байты, делает из них строку и отправляет в Handler H – который привязан к главному потоку. Ниже мы видим код самого handlera – в него поступает сообщение, которое мы видели ранее (строка с прочитанным сообщением из буфера Bluetooth) , обнаруживается логгером и отправляется на дальнейшую обработку в CommandParcer ( функция, которая «разруливает» разные команды).

Использование многопоточности дает нам возможность гибко настраивать функциональность приложения под себя, а так же не тормозит главный поток, что важно в интерактивных приложениях.

2) Семафоры

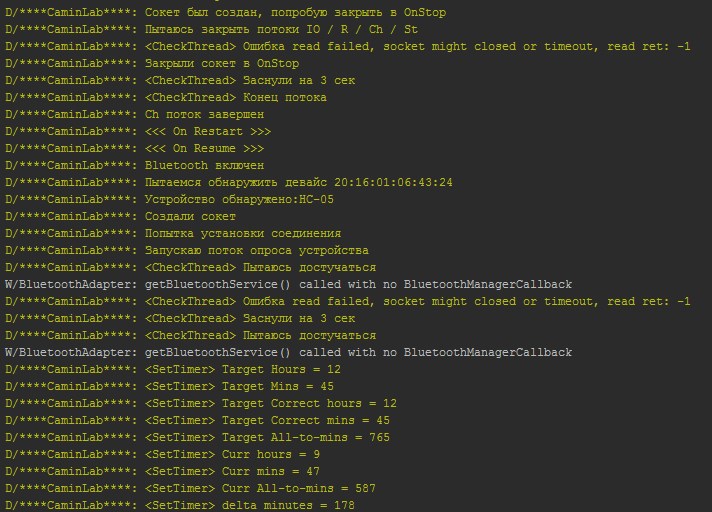
Семафоры используются для того, чтоб перед использованием ресурса проверить его доступность.  
Примером из жизни может служить тележка (общий ресурс) и два работника (потоки java). Один работник, к примеру, наполняет тележку песком. В это время второй работник, который перевозит груз и затем разгружает, не может взять тележку и отвезти ее.  
В то же время, если второй работник увез тележку, то первый работник не должен ничего наполнять.

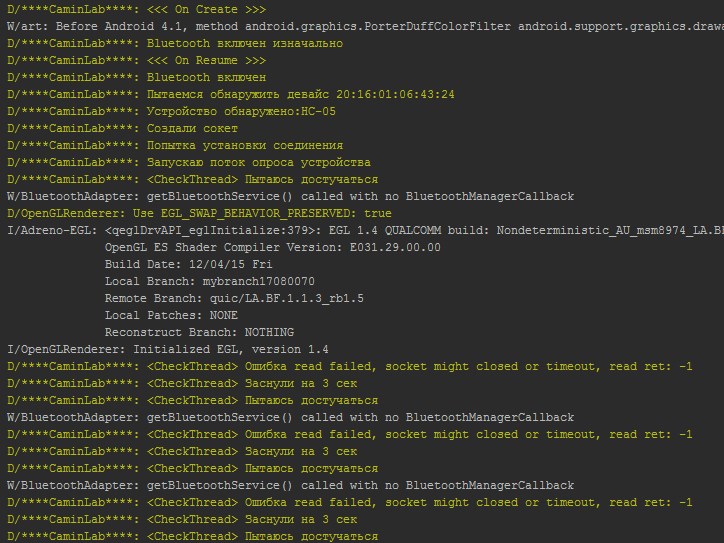
В нашем случае критическим ресурсом является функция отправки по Bluetooth. Так как у нас несколько потоков , которые, чисто теоретически, могут в одно и то же время попытаться писать в Bluetooth, то пришлось использовать семафоры. Особенно актуально это становится тогда, когда идёт синхронизация статистики с контроллером – ведь мы должны принять около 100 чисел ( запись ведётся на контроллере каждые 15 мин) , а это относительно долгий процесс, в течении которого очень велик шанс, что кто-то ещё попытается отправить сообщение. Так же семафоры используются в некоторых случаях, когда несколько команд передачи данных должны быть «одним целым» : пример этому может служить реализация таймера выключения – сначала отправляется управляющая команда, а потом время.  
(access – семафор. Метод «acquire» - занять семафор, «release» – освободить семафор)

  
3) Использование сторонних библиотек

На контроллере ведется сбор статистики по температуре каждые 15 минут, данные хранятся в «кольцевом буфере», состоящим из 96 записей (24 часа, в каждом по 4 записи). При включении приложения статистика выгружается в телефон и по ней строится график (ось Х – время, У – температура). Для этого пришлось воспользоваться сторонней библиотекой, найденной в интернете, которая строит графики – AchartEngine (библиотека распространяется под весьма мягкой лицензией Apache Version 2, так что мы вполне можем использовать AChartEngine даже в коммерческих проектах.). Подключение этой библиотеки существенно упростило создание графика.

**Результаты работы**

Реализованные функции были отлажены во встроенном в Android studio отладчике:



Используемый аналоговый термодатчик был откалиброван по цифровому термометру с точностью -+0.5 С.

# Руководство Пользователя

При успешном подключении к контроллеру вы увидите такой экран:

Если Ардуино в данный момент не онлайн, вы увидите такой экран (при возвращении ардуино в онлайн, сразу станет таким, как скриншот выше):

Главная кнопка отвечает за включение камина, если тот выключен, или выключение, если включен. О текущем состоянии камина нас информирует галочка ( или её отсутствие) в левом верхнем углу экрана:

Если установлено стабильное соединение с Ардуино, то можно использовать дополнительные функции:

* Установить желаемую температуру. Можно ввести температуру самому, а если в поле нет ничего, то кнопки « + » и « - » вернут в поле текущую температуру + или – 0.1 градус :
* Установить таймер отключения камина. При нажатии на поле ввода появляется клавиатура для времени, где и вводится желаемое время выключения камина:

И в заключении есть кнопка «график» - при нажатии на которую нас отправляет в новый экран, где будет нарисован график температуры за последние сутки с интервалом в 15 минут:

# Заключение

За время проекта мы добились своей цели, реализовав часть «умного» дома. И эта программа будет реально использоваться летом на даче.  
  
В ходе проекта мы реализовали:

- Общение смартфона с контроллером

- Собственная прошивка для контроллера, реализующая требуемый функционал:

- Включение / отключение реле, к которому могут быть подключены различные электроприборы

- Сбор данных с аналогового термодатчика

- Реагирование на различные команды

- Автономная работа

- Сбор статистики

- Планировщик

- Отдача команд смартфоном и обработка возвращаемых значений:

- Передача команды «вкл / выкл реле»

- Передача команды «запросить температуру» и приём ответа на неё

- Передача команды «установить таймер на» и целевое время

- Передача команды «установить целевую температуру на» и температуру

- Отображение графика статистики и принятие её с контроллера

# 

# Литература

1. Java руководство для начинающих. Гелберт Шилдт

2. Head First. Программирование для Android. Дон Гриффитс

3. [www.geektimes.ru](http://www.geektimes.ru)

4. www.stackoverflow.com

5. www.developer.android.com

6. www.arduino.cc