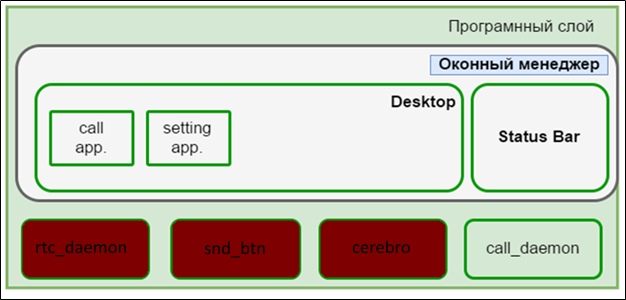
# 1. Описание ПО смартфона XXon

****

**Рисунок 1.** Суммарная схема разработанного ПО

Из разработанного программное обеспечение смартфона можно выделить следующие компоненты (рис.1):

* **демоны для контроля периферии (rtc\_daemon, snd\_btn, cerebro);**
* демон для контроля поступивших звонков;
* оконный менеджер;
* рабочий стол, созданный в собственном оконном менеджере;
* статус бар, созданный в собственном оконном менеджере;

**Курсовая работа описывает проектирование и разработку демонов контроля периферии.**

# 2. Описание RTC -демона

Алгоритм работы демона следующий: когда система запускается, при старте демона сначала выполняется подготовительная команда регистрации RTC-модуля в системе, и сразу же после этого происходит синхронизация часов внутреннего времени RaspBerry Pi с часами RTC-модуля. После этого демон завершает свою работу, однако из-за используемого флага RemainAfterExit=yes видится в системе как активный.

Когда система выключается, демон выполняет обратную операцию: синхронизирует часы RTC-модуля с часами RPi (т.к. в ходе работы может произойти синхронизация времени через NTP, или же пользователь самостоятельно изменит системное время или часовой пояс). После останова, демон выполняет команду отключения RTC-модуля из системы, и модуль вновь становится не зарегестрированным.

# 3. Описание работы демона Cerebro и МК Digispark

## 3.1. Описание демона Cerebro

Приложение состоит из двух зависимых потоков: поток «обслуживания запросов» (сторонних приложений) и поток «опроса состояния MCU».

Поток «**опроса состояния MCU**» опрашивает состояние микроконтроллера (MCU) командой 0xCC (получить статус). Поток «**обслуживание запросов**» принимает подключения сторонних приложений по управлению и получению информации с МК.

Зависимость потоков заключается в том, что оба потока используют общий I2C-интерфейс для взаимодействия с МК. Решение коллизий и их синхронизация производится посредством мьютексов.

***Примечание***: программа **xxblock** еще не разработана, и возможно алгоритм будет изменен.

**Поток «обслуживание запросов»** сторонних приложений представляет собой сервер, использующий UDS для подключения клиентов. Исходя из требований, которые были выдвинуты другими разработчика, установлено, что обращение к демону производится 1-2 приложение одновременное (не более). Так как каждая из 4 команд обслуживается довольно быстро, принято решение сделать сервер однопоточным, что позволяет не усложнять ПО зря.

**Поток «опроса состояния MCU»** опрашивает состояние микроконтроллера (MCU) командой 0xCC (получить статус), на что МК возвращает 1 байт**, нулевой бит** в котором показывает флаг о выключении RPI (если кнопка блокировки долго нажата, то МК устанавливает флаг выключения), а **первый бит** – флаг блокировки экрана.

Если установлен **флаг выключения** (0 – значит установлен) – демон инициализирует приложение **xxoff**, которое выводим диалоговое окно «выключения». Если пользователь ответит «выключить», то приложение xxoff посылает сигнал 0x88 рассматриваемому демону, а тот инициализирует процесс выключения.

Если установлен флаг блокировки экрана (0 – значит установлен) – демон инициализирует приложение **xxblock**, которое блокирует любые действия пользователя, при использовании сенсора. Когда флаг сброшен (то есть равен 1) – приложению посылается сигнал об отмене блокировки.

Для удовлетворения всех требований необходимо обеспечить поддержку следующего набора команд:

* получить текущий уровень заряд батареи;
* установить яркость экрана;
* выключить/включить миникомпьютер RPi;
* узнать текущее состояние экрана (заблокирован/разблокирован);

По той причине, что взаимодействие происходит с относительно медленным (на фоне RPi) микроконтроллером по достаточно медленной шине I2C, принято решение, что команды должны быть минималистичны: **представление команд в бинарном виде.**

## 3.2. Описание ПО МК DigiSpark (ATtiny85)

МК поддерживает, аналогичные демону, **четыре I2C-команды:**

1. задать яркость дисплея;
2. выдать показания с батареи;
3. выдать текущий статус;
4. считать RPI выключенным.

Наиболее важной командой из всех считает команда установить яркость, поэтому она имеет максимальный приоритет и при поступлении команды I2C\_CMD\_BRIGHT – все остальные команды откладывают, и МК переходит в режим ожидания следующей команды.

Второй по приоритетности является команда I2C\_CMD\_BATTERY , т.к. команда является самой дорогостоящей из всех: время срабатывания АЦП на хороших МК может варьироваться от 12мс до 29мс. МК выполняет блокирование работы, до завершения оцифровки, и затем по 1 байту передает данные. Это может вызвать задержку на I2C-шине, поэтому часть ответственная за анализ нажатия кнопки – опускается.

***Примечание***: МК выполняет **все задачи в одном потоке.**

# 4. Демон контроля звука (btn\_sound)

При запуске демона **производится** **резервирование** аппартной части – GPIO, которое служит для подключения кнопок.

После успешной инициализации GPIO (за это отвечает разработанны класс GPIOСontrol), демон переходит **в режим ожидания нажатия кнопки или истечения таймаута** этого события. Очень важно отметить, что за ожидание собтытия так же отвечает метод класса GPIOControl - **getValueOnEvent**, который позволяет детектировать события наGPIO, а не опрашивать состояние GPIO с определенной частотой. То есть событие носит асинхронный характер.

**Если событие наступило**, в дело вступает класс OrgOfonoCallVolumeInterface, который предоставляет методы взаимодействия с демоном Ofono (который отвечает за совершение любых действий с чипом связи SIM800L) по системной DBus-шине (не путать с сеансной dbus-шиной). После передачи командый демона некоторое время (250мс) не реагирует на нажатия кнопок, а затем вновь переходит в режим ожидания события/таймаута.

**Если событие не наступило**, а произошел таймаут – демон повторно переходит в режим ожидания события.

**Если поступило прерывание** (по сигналу SIGINT) – производится освобождение аппаратных ресурсов (GPIO) и завершение работы.

Приложение состоит из двух независимых потоков: **потоки «увеличения»/ «уменьшения» громкости** звука. Данный демон является самым обширным, хотя и не самым богатым с функциональной точки зрения.