**1. Краткое описание ПО**

Предоставляемое программное обеспечение включает в себя 3 демона:

**cerebro** – демон управления микроконтроллером. Демон предоставляет unixSocket для приема/передачи команд. Демон поддерживает 4 команды: установить яркость экрана, узнать заряд батареи, узнать статус МК и предупреждение о выключении.

**btn\_sound** – демон отвечает за увеличение/уменьшение громкости звука во время разговора; демона автономен.

**rtc** – демон синхронизации времени, демон так же автономен и взаимодействие с ним не требуется.

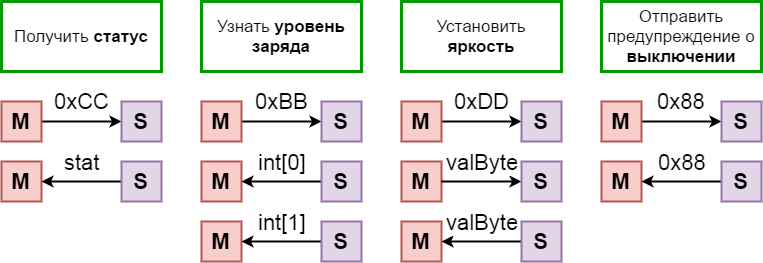
**2. Предоставляемое API**

Единственным демоном, который предоставляет API, с которым можно взаимодействовать, является демон Cerebro, который обеспечивает поддержку следующего набора команд:

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

Для повышения скорости работы взаимодействие происходит с помощью **команд в бинарном виде.** На рисунке 1 приведен формат протокола взаимодействия с демоном «Cerebro» (обозначено буквой **S**) и стороннего приложения (обозначено буквой **M**) через unixSocket. Инициатором всех команды является клиент (М), который устанавливает соединение демоном.

Клиентское приложение использует соответствующие 4 команды для получения статуса MCU (какие кнопки нажаты), получения текущего уровня заряда батареи, установки яркости экрана и предупреждения о своем выключении.



**Рисунок 8.** Протокол взаимодействия демона «Cerebro» и MCU через I2C-шину

**3. Возвращаемые значения**

* команда «получить статус» возвращает значение из кеша, который обновляется каждую секунду;
* команда «узнать уровень заряда» возвращает значение от 0 до 100%;
* команда «установить яркость» возвращает 0, если яркость установлена корректно;
* команда 0x88 имеет иное значение: пользователь отдал команду на выключение.

***Примечание:*** в остальном протоколы взаимодействия между демоном-приложениями и демоном-микроконтроллером полностью совпадают.

Приведем пример получения статуса МК(листинг 1).

**Листинг 1. DaemonCheck.cpp**

#include <sys/socket.h>

#include <sys/un.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define I2C\_CMD\_CHECK 0xCC // "Check general status" (frequency == 1 sec.)

//char \*socket\_path = "./socket4242";

char \*socket\_path = "\0hidden";

int main(int argc, char \*argv[]) {

struct sockaddr\_un addr;

char buf[100];

int fd,rc;

if (argc > 1) socket\_path=argv[1];

if ( (fd = socket(AF\_UNIX, SOCK\_STREAM, 0)) == -1) {

perror("socket error");

exit(-1);

}

memset(&addr, 0, sizeof(addr));

addr.sun\_family = AF\_UNIX;

if (\*socket\_path == '\0') {

\*addr.sun\_path = '\0';

strncpy(addr.sun\_path+1, socket\_path+1, sizeof(addr.sun\_path)-2);

} else {

strncpy(addr.sun\_path, socket\_path, sizeof(addr.sun\_path)-1);

}

if (connect(fd, (struct sockaddr\*)&addr, sizeof(addr)) == -1) {

perror("connect error");

exit(-1);

}

buf[0] = I2C\_CMD\_CHECK;

if (write(fd, buf, 1) != 1) {

fprintf(stderr,"write error");

exit(-1);

}

if (read(fd,buf,1) != 1) {

fprintf(stderr,"read error");

exit(-2);

}

printf("Display: %s\n", ((buf[0] & 0x01) == 0x01) ? "WORKING" : "BLOCKED");

printf("Raspberry Pi: %s\n", ((buf[0] & 0x02) == 0x02) ? "WORKING" : "OFF");

return 0;

}