

Министерство образования и науки Российской Федерации  
ФГАОУ ВПО «УрФУ имени первого Президента России Б. Н. Ельцина»  
Институт радиоэлектроники и информационных технологий - РтФ  
Департамент информационных технологий и автоматики

Wi-Fi микропроцессор CC3100 и инструменты  
разработки

ОТЧЕТ  
по научно-исследовательской работе

Преподаватель: Ермакова Галина Михайловна  
Студент: Сухполюев Илья Владимирович  
Группа: РИ-440001

Екатеринбург  
2017

# Содержание

Введение . . . . .	3
1 Обзор документации к устройствам . . . . .	5
1.1 CC3100 . . . . .	6
1.2 CC3100BOOST & CC31XXEMUBOOST . . . . .	12
2 CC3100SDK . . . . .	15
2.1 Получение SDK . . . . .	15
2.2 Внутреннее устройство SDK . . . . .	16
2.3 Настройка среды разработки . . . . .	17
3 Тестирование пропускной способности CC3100 . . . . .	18
Заключение . . . . .	21
Список использованных источников . . . . .	22
А Изображения устройств . . . . .	23
Б Исходный код программ . . . . .	25
Б.1 Тестирование пропускной способности . . . . .	25

## Введение

Целью данной работы является изучение возможностей и структуры wifi-чипа *CC3100*[1], разработанного и поставляемого компанией *Texas Instruments*, для использования в качестве передатчика информации в рамках IoT-решений (*IoT – Internet of Things – Интернет вещей*). Данное исследование является предварительным сбором сведений для возможной интеграции чипа с миниатюрными диктофонами ООО «Вторая лаборатория»[2], с целью осуществления удаленной настройки диктофона и получения уже записанных аудиоданных с диктофона. Поэтому одними из главных исследуемых характеристик будет пропускная способность и низкое энергопотребление, как при работе чипа, так и при выключенном (спящем) режиме.

Чип CC3100 создан для предоставления готового стека Интернет протоколов на устройства с отдельным модулем управления *MCU* (*Microcontroller Unit*), поэтому для проведения требуемых изменений помимо самого чипа требуется установка, позволяющая представить требуемое питание и задающая внешнее программное управление. Для этих целей компания Texas Instruments поставляет установки *CC3100BOOST*[3] и *CC31XXEMUBOOST*[4], а также прилагает к ним программные инструменты для разработки *SDK*(*Software Development Kit*) для запуска отладочных управляющих программ на стационарном компьютере, которые могут помочь в тестировании заявленных чипом CC3100 характеристик.

Таким образом, для достижения поставленной цели необходимо решить следующие задачи:

- Изучить документацию и рекомендации по разработке, прилагающиеся к CC3100;
- Ознакомиться с устройством и документацией схемы CC3100BOOST;
- Ознакомиться с устройством и документацией схемы CC31XXEMUBOOST;
- Собрать на основе схем CC3100BOOST и CC31XXEMUBOOST отладочную установку;
- Запустить на полученной установке примеры программ из CC3100SDK;
- Провести тестирование пропускной способности CC3100, используя рассмотренные примеры программ и руководства к разработке.

# 1 Обзор документации к устройствам

В этой главе будет проведен обзор документации, предоставляемой к исследуемому wifi-чипу CC3100, отладочной установке CC3100BOOST и ее связке с CC31XXEMUBOOST.

Важным предварительным моментом является то, где была получена документация. Конечно, хорошо иметь переведенный и адаптированный материал на родном, русском, языке, но обычно данные источники быстро устаревают, после обновления исходной документации на официальном сайте, а также очень часто могут содержать ошибки и неточности, возникшие при переводе, либо ошибки, унаследованные от первоисточников, без учета списка опечаток, предоставляемых Texas Instruments. Поэтому стоит использовать документы, распространяемые вместе странице продукта (как на рисунке 1.1), а также использовать Вики-ресурс посвященный линейке CC31XX[5].

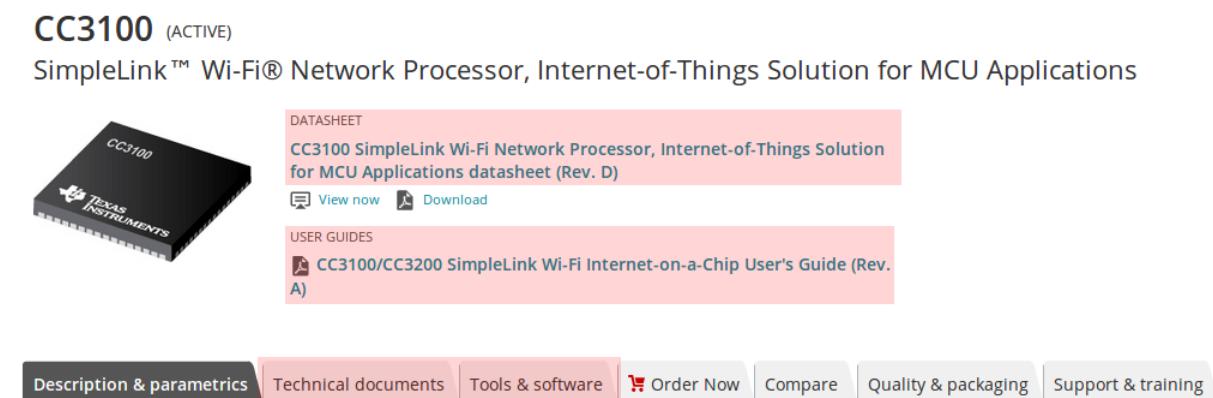


Рисунок 1.1 — Основные источники информации (Выделено красным)

## 1.1 CC3100

Рассмотрим основные источники информации к главному в нашем исследовании устройству – CC3100. В нашем случае, это два основных технических описания:

- a) Техническое описание процессора (*Datasheet*)[6];
- б) Руководство пользователя (*User Guide*)[7].

Основные характеристики *CC3100 SimpleLink Wi-Fi*:

— Состоит из Интернет процессора и подсистемы управления питанием;

— Wi-Fi Интернет подсистема:

- ★ ARM микроконтроллер для полной обработки Интернет-протоколов и связи с внешним микроконтроллером;
- ★ Поддержка 802.11 b/g/n;
- ★ Поддержка WPA2 Personal and Enterprise;
- ★ Работа в режиме станции, точки доступа и Wi-Fi ретранслятора;
- ★ стек протоколов TCP/IP:

До 8 одновременных TCP или UDP соединений;

До 2 одновременных TLS и SSL соединений.

- ★ Пропускная способность:

UDP: 16 Мбит/с;

TCP: **13 Мбит/с.**

— Подсистема управления питанием:

- ★  $V_{BAT}$  Wide-Voltage режим: от 2.1 до 3.6 Вольт;
- ★ Предустановленный 1.85-Вольт режим.

Более подробные характеристики указаны в тех. описании[6].

Главным для нашего исследования будет являться проверка заявленной пропускной способности (выделено жирным).

Хорошой иллюстрацией устройства внутренних подсистем и модулей изображено в техническом описании на рисунках 1.2 и 1.3.

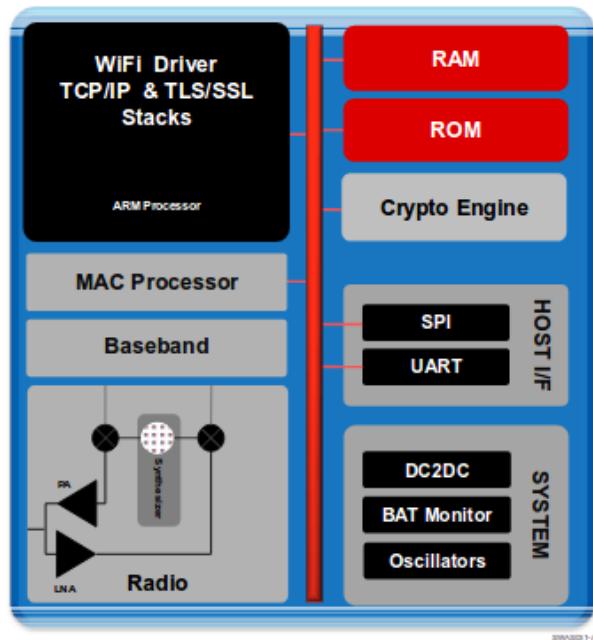


Рисунок 1.2 – Основные модули и подсистемы CC3100

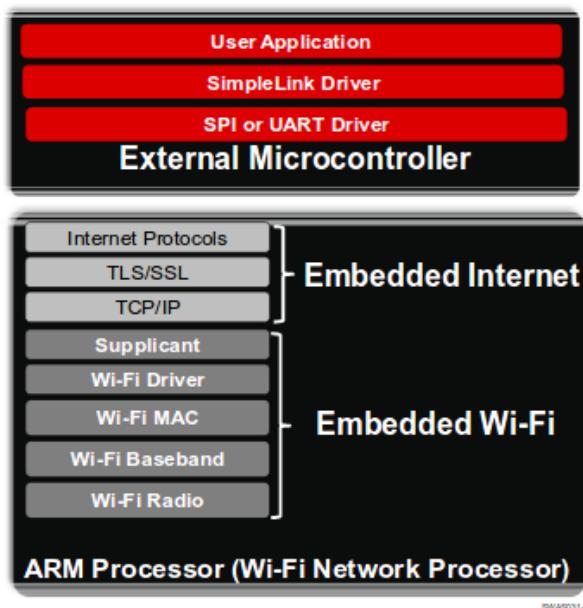


Рисунок 1.3 – Стек Интернет протоколов, реализованных в рамках CC3100. Также есть возможность реализации своих протоколов на внешнем микроконтроллере

Следующим важным моментом, который нужно рассмотреть в устройстве чипа – это как производится его управление с помощью внешнего микроконтроллера. CC3100 поддерживает иммет два типа портов – *UART* и *SPI*. Первый, *Universal Asynchronous Receiver-Transmitter*[8] – Универсальный асинхронный приёмопередатчик, используется для перепрощивки и первоначальной настройки чипа. В то время управление осуществляется через SPI (*Serial Peripheral Interface*)[9].

Шина SPI рассчитана на несимметричное соединение двух микроконтроллеров – один из них, ведущий (*master*), всегда инициирует обмен данными, на что второй, ведомый (*slave*), принимает данные и может в ответ передать аналогичное количество байт. В классическом виде SPI-шина состоит из четырех основных соединений (проводов):

- Chip Select или Slave Select /  $\overline{CS}$  или  $\overline{SS}$  – выбор устройства, на который будет происходить передача;
- Clock / CLK – тактовый сигнал;
- Master Output Slave Input / MOSI – выходной сигнал ведущего микроконтроллера;
- Master Input Slave Output / MISO – входной сигнал для ведущего микроконтроллера.

Передача происходит при установке мастером сигнала  $\overline{CS}$ , что сигнализирует выбор нужного устройства (Мастер может иметь несколько линий  $\overline{CS}$  для соединения с разными устройствами, при этом остальные провода могут быть подключены к одному и тому же порту). Далее, в зависимости от двух параметров: начало синхронизации с верхнего или нижнего уровня CLK и выборка данных происходит по заднему или переднему фронту CLK, SPI начинает считывать данные оцифрованные сигналы с шин MOSI и MISO.

На рисунке 1.4 изображены основные промежутки времени для работы по SPI.

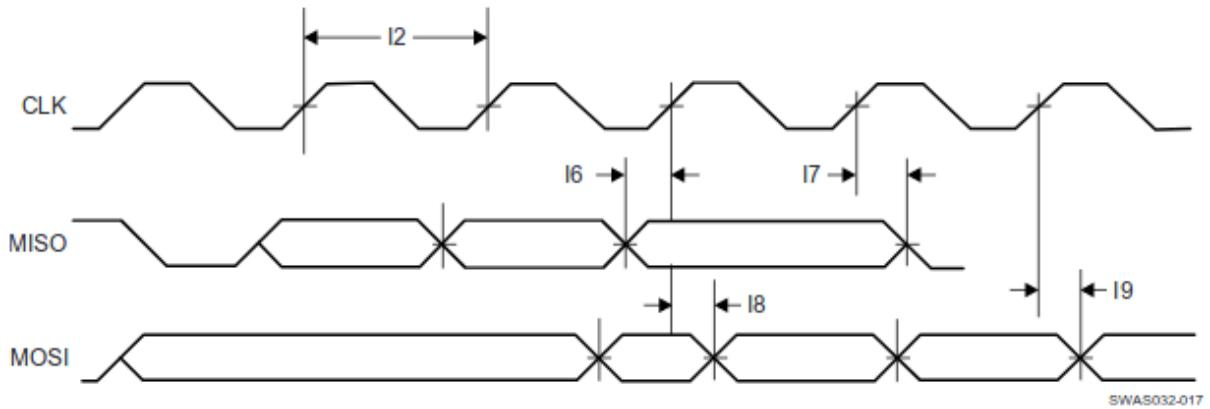


Рисунок 1.4 – Основные временные характеристики SPI CC3100

Таблица 1.1 – Основные характеристики SPI-соединения

Номер параметра	Название параметра
l2	тактовый период
l6	время на установку RX
l7	период удерживания RX
l8	задержка вывода TX
l9	период удерживания TX

Для полноценной работы всего стека Интернет протоколов чипу CC3100 требуется внешняя flash-память, соединенная по SPI, изображенная на схеме 1.5.

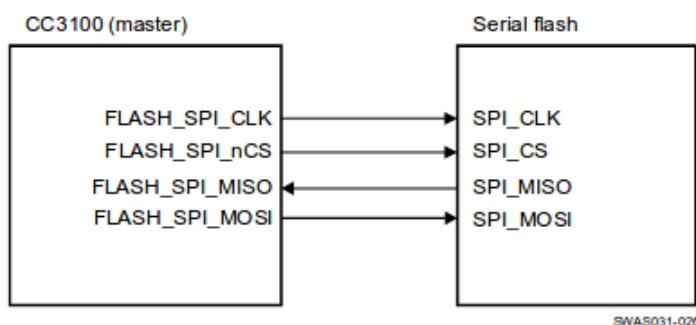


Рисунок 1.5 – Схема соединения CC3100 с внешней flash-памяти

Для управления внешним микроконтроллером используется более хитroе, 6-ти проводное, соединение (Рисунок 1.6). Помимо основных

SPI-соединений при управлении требуется еще *nHIB* сигнал для выключения wifi-чипа в режим сна (*Hibernate*), а сигнал INTR (*Interrupt*) потребуется для инициации прерывания со стороны Wi-Fi.

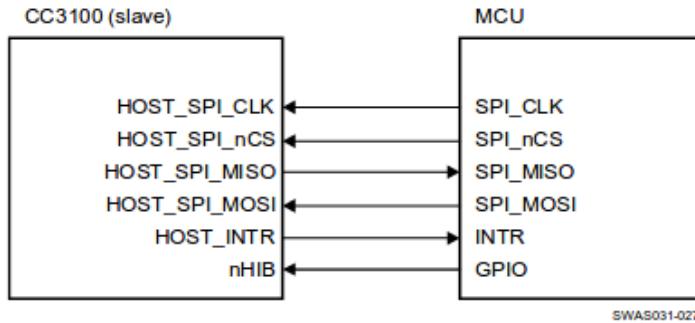


Рисунок 1.6 — Схема соединения CC3100 с ведущим микроконтроллером

Наконец, рассмотрим руководство для разработчика, содержащее множество примеров и пояснений, как программно работать с API драйвера, поставляемым вместе с CC3100, а так же кратко и на примерах объясняющее ту или иную поддерживаемую технологию, иллюстрируя это примером программы.

На данном этапе полезно ознакомится с описанием базового конечного, автомата, который необходимо реализовать для полноценного управления Wi-Fi подсистемой (Рисунок 1.7).

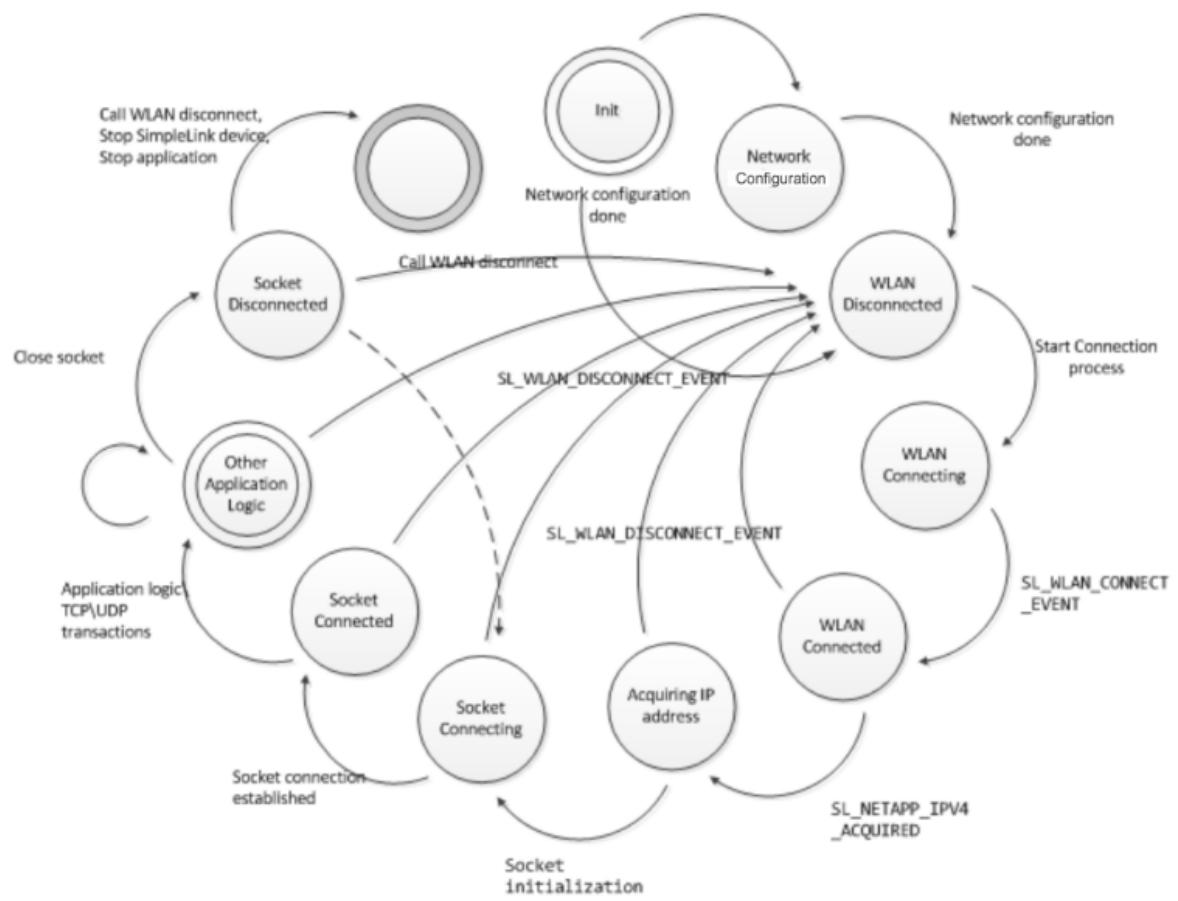


Рисунок 1.7 — Базовый конечный автомат для работы с Wi-Fi чипом

## 1.2 CC3100BOOST & CC31XXEMUBOOST

Рассмотрим набор руководств прилагающихся к CC3100BOOST и CC31XXEMUBOOST, обеспечивающих предварительной установкой для тестирования Wi-fi чипа. При желании, можно скачать с сайта Texas Instruments более подробное описание устройства их схем, но для быстрого старта будет достаточно следующих руководств:

- Руководство для быстрого старта (*Getting started guide*)[10];
- Руководство для пользователя (*User guide*)[11].

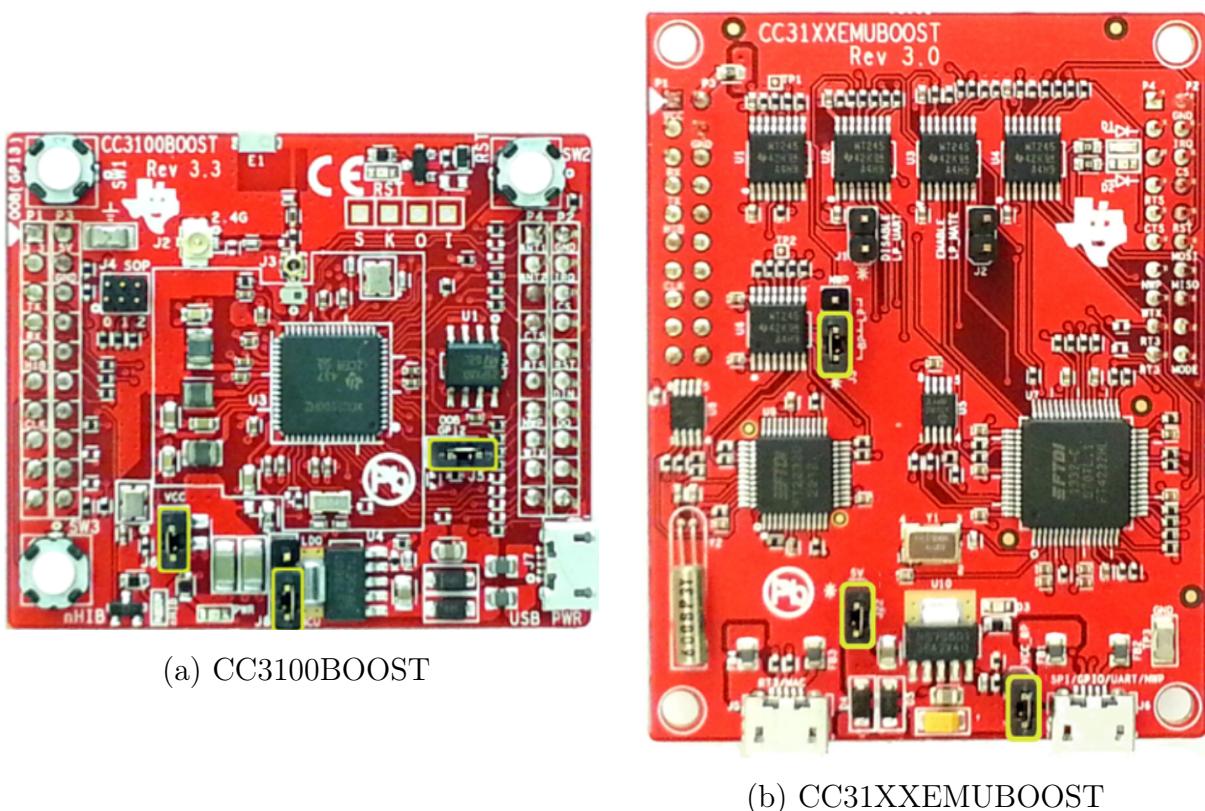


Рисунок 1.8 — Расположение джамперов на CC3100BOOST и CC31XXEMUBOOST для совместной работы

В руководстве для быстрого старта можно найти краткое описание устройств, а также процесс сборки, как самих схем, так и распаковки SDK и настройки сред разработки для запуска первых программ. На рисунке 1.8 выделено необходимое расположение джамперов (переключателей) для совместной работы. После их установки остается соединить обе платы между собой, чтобы совпали белые треугольники отмаркиро-

ванные на плате. Фото данных плат в другом ракурсе и получившуюся конструкцию можно рассмотреть в Приложении А.

В руководстве же пользователя, можно найти более детальное описание схемы CC3100BOOST, ка например расположение основных элементов на схеме (Рисунок 1.9) от расположения внешней антенны до элементов питания, а также способах измерения входного напряжения.

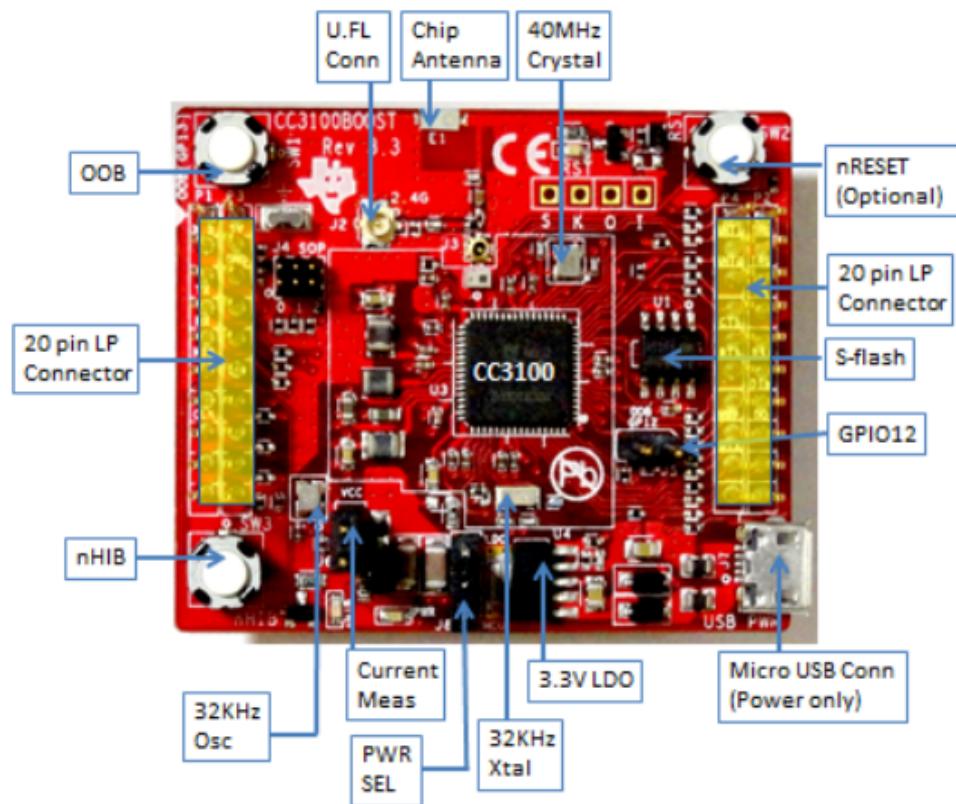


Рисунок 1.9 — Расположение элементов на CC3100BOOST

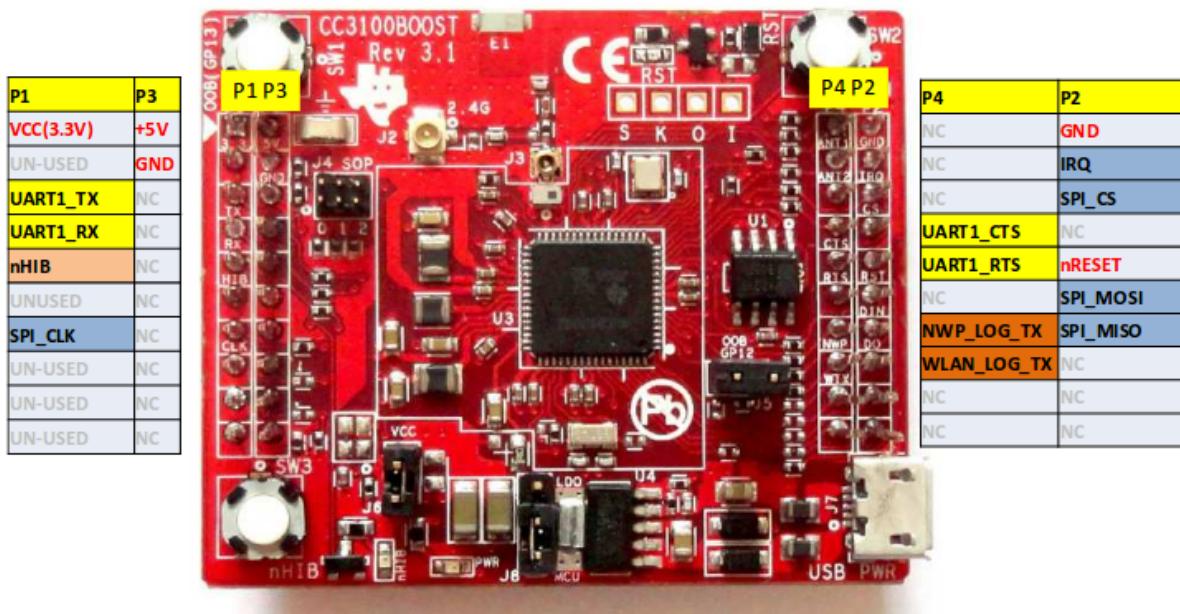


Рисунок 1.10 — Вывод ножек CC3100 на плате BOOST

Также, одним из самых интересных описаний в этой схеме, является вывод ножек на внешние выводы (Рисунок 1.10). Так, рассмотрев в предыдущем разделе выводы самого CC3100 мы видим необходимые нам SPI\_CS, SPI\_CLK, SPI\_MOSI, SPI\_MISO для соединения с внешним микроконтроллером по SPI, а также сигналы nHIB и IRQ для лучшего управления чипом. В добавок к этому, для полноценной проверки прототипа будущего устройства мы можем подвести внутреннее питание от целевого устройства на ножки VCC или +5V, и заземлить чип на ножках GND.

## 2 CC3100SDK

В этом разделе рассмотрены программных средств и настройки среды разработки для работы с комплектом CC3100BOOST и CC31XXEMUBOOST, рассмотренных в прошлой главе. Конечной целью данной главы будет запуск нескольких примеров, содержащихся в комплекте с SDK.

### 2.1 Получение SDK

Чтобы получить программные инструменты для разработки (SDK – *Software Development Kit*), достаточно найти ссылку на скачивание SDK на одной из страниц продуктов, после чего мы попадем на страницу выбора требуемого нам установочника для Windows-системы [12] (Рисунок 2.1). На момент написания работы, последней версией SDK была 1.3.0. После выбора SDK понадобиться заполнить форму от частного лица и/или лица компании и принять соответствующие условия использования, предоставляемого продукта. Как все условия будут выполнены вам придет ссылка для получения актуальной версии SDK.

« Back to software product page

#### CC3100 Software Development Kit (SDK), Service Pack, and Add-Ons

CC3100SDK\_1.3.0

Release Date: 14 Mar 2018

[View release notes](#) [Supported Platforms](#) [Release Information](#)

This page contains specific information about CC3100 Software Development Kit (SDK), Service Pack, and Add-Ons release package. Refer to the table below for download links and related content.

**Product downloads**

Download requires export approval (1 minute)

Title	Version	Description	Size
SDK Installers			
Windows Installer for CC3100SDK	1.3.0	Link to Windows Installer for CC3100SDK	25025K
Service Pack Installer for CC3100SDK	1.0.1.11-2.10.0.0	Link to Service Pack Installer for CC3100SDK	6456K
Host Programming Add-On for CC3100SDK	1.1.0	Link to Service Pack Installer for CC3100SDK	5147K

**Supported Platforms**

Platform	Supported Devices	Supported Kits and Boards
CC3100	• CC3100 • CC3100MOD	<ul style="list-style-type: none"><li>• CC31XXEMUBOOST: Advanced Emulation Kit for SimpleLink™ Wi-Fi® CC31xx BoosterPack™ plug-in module</li><li>• CC3100MODBOOST: SimpleLink Wi-Fi CC3100 module BoosterPack</li><li>• DISCOVERY-ADAPT: Discovery Adaptor Board for SimpleLink Wi-Fi CC31XX BoosterPack Plug-In Module</li><li>• CC3100BOOST: SimpleLink™ Wi-Fi® CC3100 wireless network processor BoosterPack™ plug-in module</li></ul>

Рисунок 2.1 — Страница получения SDK для CC3100

## 2.2 Внутреннее устройство SDK

Выполнив установку SDK в нужную папку, мы получаем платформонезависимый драйвер *SimpleLink*, документацию к нему, его портирование для использования с различными отладочными установками (В частности, к нашей CC3100BOOST+CC31XXEMUBOOST), а примеры программ и проектов IDE для изучения возможностей чипа.

Документация, содержащаяся в SDK, при определенном опыте может частично или полностью заменить руководство пользователя CC3100[7] и являться более точным и подробным справочным материалом по использованию тех или иных возможностей драйвера. К руководству стоит обращаться для получения первоначального представления о тех или иных подсистемах и протоколах, с которыми требуется работать.

Структура SDK:

```
./uninstall.exe          # deinсталлятор SDK
./cc3100_sdk_1_2_0_license.pdf # лицензионное соглашение
./cc3100_sdk_1_2_0_manifest.html # Описание включенных компонент
                                # для соблюдения лицензирования

./cc3100-sdk
| ./readme.txt
| ./docs          # Документация к SDK
| ./examples      # основные примеры программ
| ./netapps       # Примеры использования app-layer
| ./oslib          # файлы для разработки на freeRTOS
| ./platform      # примеры portированные в IDE
| ./simplelink     # код simplelink-драйвера
| ./simplelink_extlib # код сторонних библиотек
| ./third_party/freertos # исходный код базовой ОС
| ./tools          # собранные бинарные библиотеки
```

## 2.3 Настройка среды разработки

Чтобы начать работу с примерами из SDK, достаточно проследовать руководству по быстрому старту CC3100BOOST[10]. В ходе которого необходимо убедится, что устройство определилось на компьютере правильно и воспринимается как серия COM портов, работающих поверх USB (Рисунок 2.2).

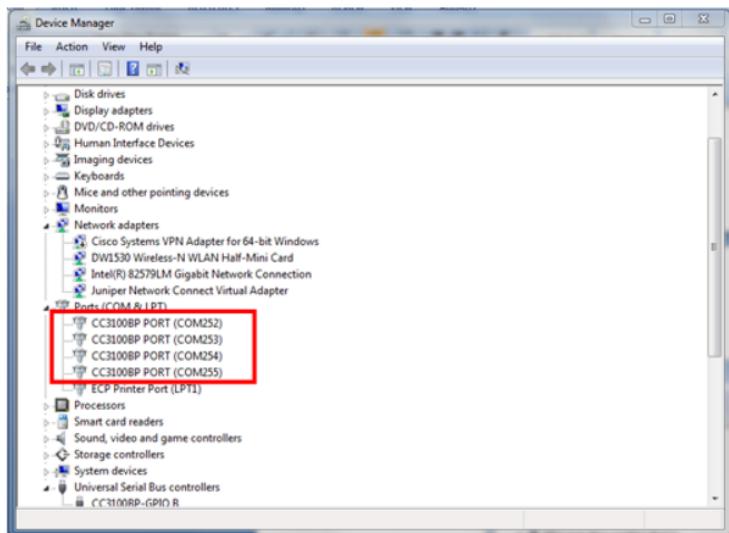


Рисунок 2.2 — cc3100sdk-env

Как только устройство начало определяться верно, можно приступить к скачиванию и установке:

- MinGW – открытого набора компиляторов;
- Eclipse – среда разработки.

Далее остается только открыть интересующий нас Eclipse-проект из папки *cc3100-sdk/platforms*, после чего потребуется его предварительная настройка, такая как указание необходимых путей для работы с MinGW, а также работы Eclipse с внутренним Makefile-ом, который описывает порядок сборки проекта. При его детальном рассмотрении можно провести преобразование проекта в более удобный для разработчика вид.

### 3 Тестирование пропускной способности CC3100

Последним этапом этой работы, является тестирование заявленной пропускной способности, которая судя по разделу 1 должна быть около 13 Мбит в секунду.

Для этого возьмем пример из SDK, который демонстрирует работу CC3100 в качестве пользовательской станции и производит проверку наличия соединения с сетью Интернет с помощью ICMP[13].

Подкорректировав предварительную настройку и разместив ее в функцию `setUp_and_check()`, нам остается только дописать работу с *berkeley-like* сокетом для соединения с тестирующим скорость TCP-сервером и передачи файла, название которого задается через командную строку.

Сам TCP-сервер написан на интерпретируемом языке *Python* версии 3.5. Его основная задача заключается в циклическом ожидании подключения устройства, получения тестовых данных и их сверку с эталонным файлом, чтобы убедится в том, что данные передались правильно и целостно. Подробнее исходный код приложения можно изучить в приложении Б.

Основная логика приложения, работающая с CC3100:

```
int main(int argc, char *argv[])
{
    setUP_and_check();

    // считывание тестового файла
    uint64_t len = (uint64_t)atoi(argv[1]);
    const char * filename = argv[2];
    FILE* f = fopen(filename, "rb");
    void* data = malloc(len);
    fread(data, 1, len, f);
    close(f);

    // создание соединения
    int32_t sockId = sl_Socket(SL_AF_INET, SL SOCK_STREAM,
                                SL IPPROTO_TCP);

    SlSockAddrIn_t addr = {
        .sin_family = SL_AF_INET,
        .sin_port = sl_Htons(8080),
        .sin_addr.s_addr = sl_Htonl(
            0xC0A80067 // 192.168.0.103
        ),
    };
    int16_t status = sl_Connect(sockId, (SlSockAddr_t *) &addr,
                                sizeof(SlSockAddrIn_t));

    // передача данных на TCP сервер
    for(int32_t i = 0; i < len / 1024; i++) {
        status = sl_Send(sockId, data + i * 1024, 1024, 0);
    }
    sl_Close(sockId);
}
```

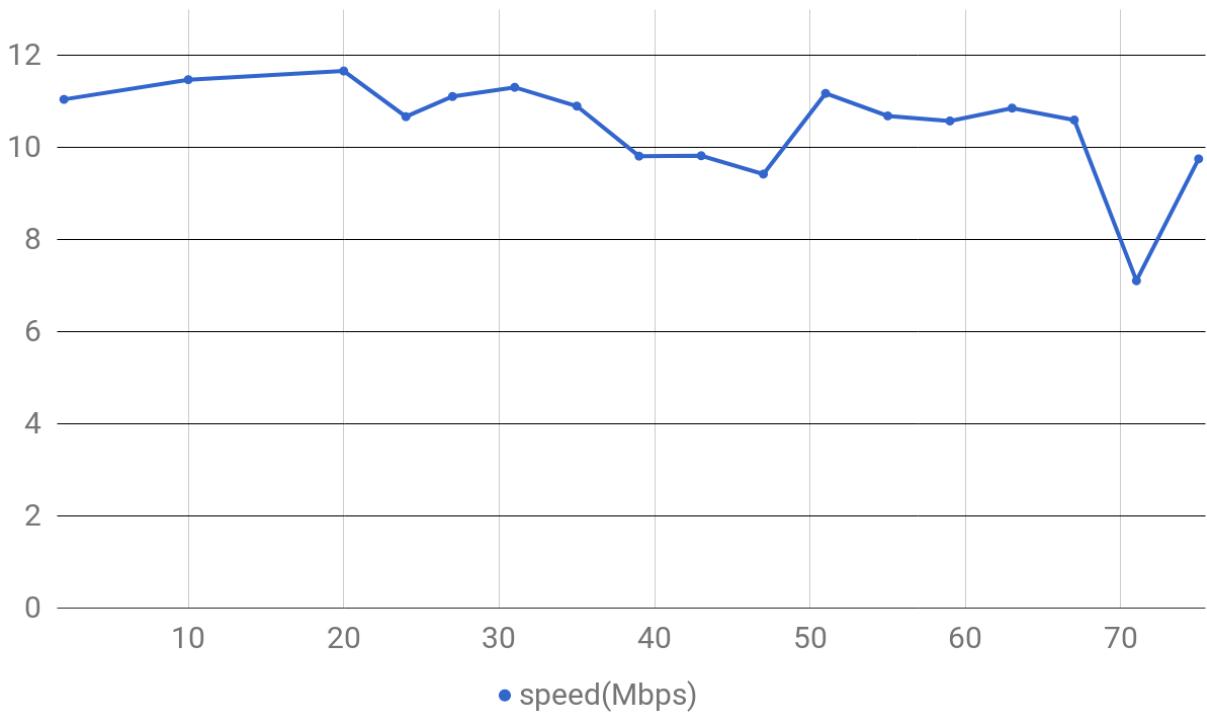


Рисунок 3.1 — Результаты тестирования пропускной способности в зависимости от дальности передачи данных

Модернизировав полученный пример выводом дополнительных сведений о соединении, а также включив код измерения пропускной способности в среднем, был проведен ряд измерений для наблюдения пропускной способности в зависимости от расстояния. Полученные результаты можно увидеть на графике 3.1, на котором можно наблюдать, что пропускная способность держится в диапазоне от 10 да 12 Мбит в секунду, что близко к заявленной.

## **Заключение**

В результате проделанной работы стало ясно, что ничего не ясно...

## **Список использованных источников**

1. CC3100 SimpleLink™ Wi-Fi® Network Processor, Internet-of-Things Solution for MCU Applications | TI.com. — <http://www.ti.com/product/CC3100>.
2. Миниатюрные диктофоны - ООО "Вторая Лаборатория". — <http://www.labi2.ru/produkciya/miniatyurnye-diktofony/>.
3. CC3100BOOST-RD SimpleLink Wi-Fi CC3100 BoosterPack Reference Design | TI.com. — <http://www.ti.com/tool/CC3100BOOST-RD>.
4. CC31XXEMUBOOST Advanced Emulation Kit for SimpleLink™ Wi-Fi® CC31xx BoosterPack™ plug-in module | TI.com. — <http://www.ti.com/tool/cc31xxemubost>.
5. CC3100 & CC3200 – Texas Instruments Wiki. — [http://processors.wiki.ti.com/index.php/CC3100\\_%26\\_CC3200](http://processors.wiki.ti.com/index.php/CC3100_%26_CC3200).
6. CC3100 SimpleLink Wi-Fi Network Processor, Internet-of-Things Solution for MCU Applications datasheet (Rev. D). — <http://www.ti.com/lit/ds/symlink/cc3100.pdf>.
7. CC3100/CC3200 SimpleLink Wi-Fi Internet-on-a-Chip User's Guide (Rev. A). — <http://www.ti.com/lit/ug/swru368a/swru368a.pdf>.
8. Universal asynchronous receiver-transmitter — Wikipedia. — [https://en.wikipedia.org/wiki/Universal\\_asynchronous\\_receiver-transmitter](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter).
9. Serial Peripheral Interface — Википедия. — [https://ru.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface](https://ru.wikipedia.org/wiki/Serial_Peripheral_Interface).
10. CC3100 SimpleLink™ Wi-Fi® and IoT Solution Getting Started Guide. — <http://www.ti.com/lit/ug/swru375c/swru375c.pdf>.
11. CC3100 SimpleLink™ Wi-Fi® and IoT Solution BoosterPack Hardware. — <http://www.ti.com/lit/ug/swru371b/swru371b.pdf>.
12. CC3100SDK\_1.3.0 | TI.com. — <http://www.ti.com/tool/download/CC3100SDK>.
13. RFC 792 - Internet Control Message Protocol. — <https://tools.ietf.org/html/rfc792>.

## Приложение А Изображения устройств



Рисунок А.1 — Wifi-чип CC3100



Рисунок А.2 — CC3100BOOST – отладочная установка

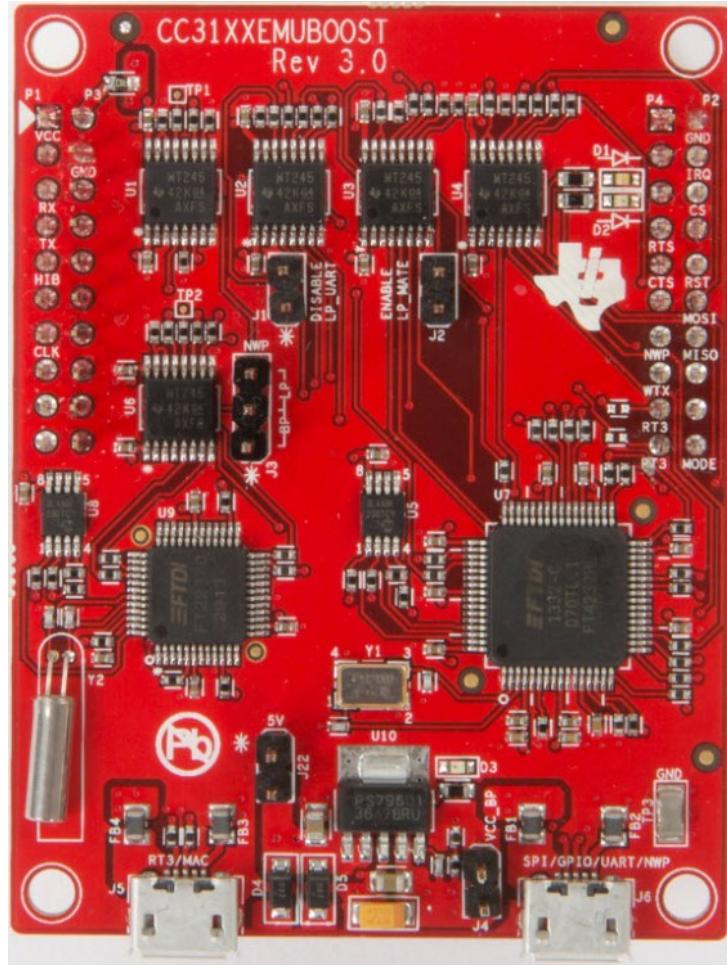


Рисунок А.3 – CC31XXEMUBOOST – установка для предоставления  
управления ПК через USB порт

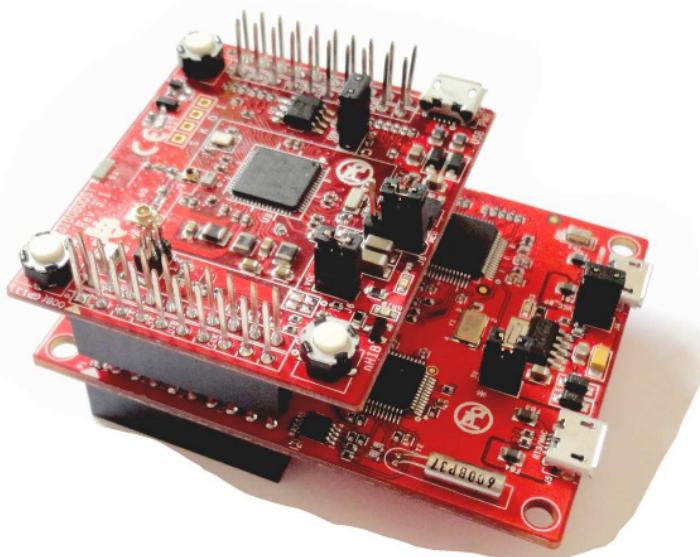


Рисунок А.4 – Соединенные CC3100BOOST и CC31XXEMUBOOST

## Приложение Б Исходный код программ

### Б.1 Тестирование пропускной способности

<https://bitbucket.org/cpractice/cc3100-test>

```
#!/usr/bin/env python
import argparse
import socket
import sys
import time

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('-f', '--filename', type=str, required=True)
    return parser.parse_args()

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('0.0.0.0', 8080))
    server.listen()
    print('[server] start listening', file=sys.stderr)
    (client, address) = server.accept() # : type client: socket.socket
    print('[server] [%s:%d] get connectinon.' % address, file=sys.stderr)
    start = time.time()

    data = []
    recived_bytes = 0
    while True:
        part = client.recv(4096)
        if part == b'':
            break
        recived_bytes += len(part)
```

```

        data.append(part)

    end = time.time()
    client.close()
    print('[server] [%s:%d] close connection. Recievd %f Mb; Time: %f' %
          address[0], address[1], received_bytes / (1024 * 1024), end - start,
          file=sys.stderr)

    return (data, received_bytes / (1024 * 1024), end - start)

def check_data(data, src_filename):
    with open(src_filename, mode='rb') as file:
        for part in data:
            origin_part = file.read(len(part))
            if part != origin_part:
                print('[server] [ERROR] Data check is failed.', file=sys.stderr)
                sys.exit(status=-1)
    print('[server] Data check is passed.', file=sys.stderr)

def main():
    args = parse_args()
    (data, size, recv_time) = start_server()
    check_data(data, args.filename)
    print(size, recv_time)

if __name__ == '__main__':
    main()

#include "header.h"

```

```

void setUP_and_check() {
    int32_t retVal = configureSimpleLinkToDefaultState(defaultDevName);
    if(retVal < 0) {
        DEBUG("Failed to configure in default state; Error: %ld");
        exit(-1);
    }
    DEBUG("Device is configured in default state");

    retVal = sl_Start(0, defaultDevName(), 0);
    if (retVal < 0 || ROLE_STA != retVal) {
        DEBUG("Failed to start the device; Error: %ld", retVal);
        exit(-1);
    }
    DEBUG("Device started as STATION");

    retVal = connectToAP("labi2", SL_SEC_TYPE_WPA_WPA2, "labi2_wpa");
    if(retVal < 0) {
        DEBUG("Failed to establish connection with AP; Error: %ld");
        exit(-1);
    }
    DEBUG("Connection established with AP and IP is acquired");

    DEBUG("Checking LAN connection - pinging Gateway...!");
    retVal = pingTargetHost(g_GatewayIP);
    if(retVal < 0) {
        DEBUG("Device couldn't connect to LAN; Error: %ld", retVal);
        exit(-1);
    }
    DEBUG("Device successfully connected");
}

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

```

```

{

    if(argscount < 3) {
        fprintf(stderr, 'not enough arguments\n');
        exit(-1);
    }

    setUP_and_check();

    uint64_t len = (uint64_t)atoi(args[1]);
    const char * filename = args[2];
    DEBUG("Read file: %s, filesize: %d", filename, len);
    FILE* f = fopen(filename, "rb");
    void* data = malloc(len);
    fread(data, 1, len, f);
    close(f);
    DEBUG("data has been read");

    int32_t sockId = sl_Socket(SL_AF_INET, SL SOCK_STREAM, SL_IPPR);
    assert(sockId >= 0);

    SlSockAddrIn_t addr = {
        .sin_family = SL_AF_INET,
        .sin_port = sl_Htons(8080),
        .sin_addr.s_addr = sl_Htonl(
            // 0xC0A80064 // 192.168.0.100
            0xC0A80067 // 192.168.0.103
            // 0xC0A82B48 // 192.168.43.72
        ),
    };
    int16_t status = sl_Connect(sockId, (SlSockAddr_t *) &addr, si
    assert(status >= 0);
    DEBUG("connected");
}

```

```
    for(int32_t i = 0; i < len / 1024; i++) {
        status = sl_Send(sockId, data + i * 1024, 1024, 0);
        assert(status > 0);
    }
    sl_Close(sockId);

    DEBUG("data sended");
    return 0;
}
```