Артем Ларионенко

ACCESSCODE PROJECT

Весна 2021

Содержание

1 Введение

ACCESSCODE PROJECT это инженерный проект по информатике, который представляет из себя дополнительный модуль расширения, которым можно укомплектовать практически любой полноразмерный ПК с целью повышения его физической защиты от проникновения. Модуль представляет из себя плату Arduino MEGA с подключенными модулями и датчиками и позволяет блокировать доступ к ПК с помощью обесточивания блока питания ПК и блокировки замков от корпусных дверей, а также имеет систему сигнализации, при срабатывании которой может запускаться функция физического самоуничтожения твердотельного накопителя (SSD) ПК. Контроль над блокировкой и разблокировкой доступа к ПК, а также над дополнительными функциями происходит с помощью кейпада, RFID-модуля, сканера отпечатков пальцев, а также удаленно с помощью GSM-shield'а.

Система сигнализации включает в себя акселерометр, который фиксирует передвижения ПК и ультразвуковой датчик расстояния, который фиксирует открытия дверей корпуса.

Система самоуничтожения заключается в принудительном физическом уничтожении твердотельного накопителя (SSD) внутри компьютера с помощью термитной смеси (подробнее см. §2.5).

Разблокировка компьютера с этим модулем состоит из трех этапов: на первом этапе владелец ПК прикладывает к ридеру, встроенному в модуль, свою RFID-карту, на втором - к датчику отпечатков пальцев свой палей, а на третьем - вводит на кейпаде пин-код, который был случайно сгенерирован и отправлен ему через СМС. (подробнее см. §4.3)

Модуль устанавливается внутрь корпуса ПК и подключается к блоку питания на уровне реле, тем самым давая возможность напрямую контролировать питание ПК, а также для работы модуля необходимо наличие возможности вывода из корпуса кейпада, датчика отпечатков пальцев и RFID-модуля.

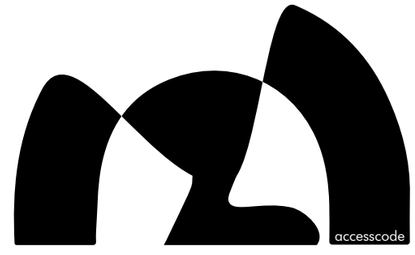
Помимо необходимости при каждом включении прикладывать RFID-карту, отпечаток пальцев и вводить случайно сгенерированный пин-код присылаемый через СМС, компьютер остается полностью функционирующим и с программной точки зрения совершенно не отличается от обычного ПК.

Модуль имеет свой аккумулятор и при подключении ПК к сети находится в состоянии постоянной подзарядки, благодаря этому при отсутствии питания ПК может автономно работать без GSM-Shield'а на протяжении некоторого времени.

Благодаря акселерометру, ультразвуковому датчику расстояния и замкам на корпусе, любая попытка вытащить SSD будет бессмысленна, так как вызовет его уничтожение.

Этот модуль можно использовать также без функции самоуничтожения, а в качестве сигнализации и средства защиты компьютера от физического проникновения. В таком случае при попытке проникновения корпус ПК будет заблокирован, блок питания выключен, а владельцу ПК будет отправлено уведомление, что его компьютер подвергся атаке.

Объективно говоря, режим самоуничтожения в ПК является очень специфической функцией, практически не имеющей практическое применение, однако никто не говорил, что так нельзя сделать, верно?



Логотип проекта

1.1 Принцип работы - краткое описание

При включении модуля пользователю дается 30 секунд на то, чтобы закрыть крышку своего компьютера и установить его на место, после этого включается режим блокировки. После того, как инициализация завершилась, звучит звуковой сигнал и включается режим защиты.

В режиме защиты любое передвижение ПК, или открытие корпуса приведут к срабатыванию сигнализации, следовательно к самоуничтожению.

В режиме защиты боковые двери корпуса заблокированы, а блок питания ПК отключен.

Чтобы отключить режим защиты, необходимо приложить RFID-карту к RFID-модулю, затем, палец к отпечатку пальцев. После этого в течении нескольких секунд приходит СМС с случайно сгенерированным пин-кодом, который нужно ввести в кейпад. После этого устройство выходит из режима защиты.

После двух неудачных попыток ввода пин-кода происходит выход из режима разблокировки (включается после ввода RFID). Также из режима разблокировки можно выйти нажатием клавиши на кейпаде.

После двух неправильных попыток ввода пин-кода устройство выходит из режима ввода пин-кода (необходимо заново вводить карту и отпечаток пальца). Также выйти из режима ввода можно с помощью кейпада.

После отключения защиты пользователь может включить ПК, а также разблокировать двери корпуса от клавиши на кейпаде.

Датчики движения/расстояния не активны при выключенной защите.

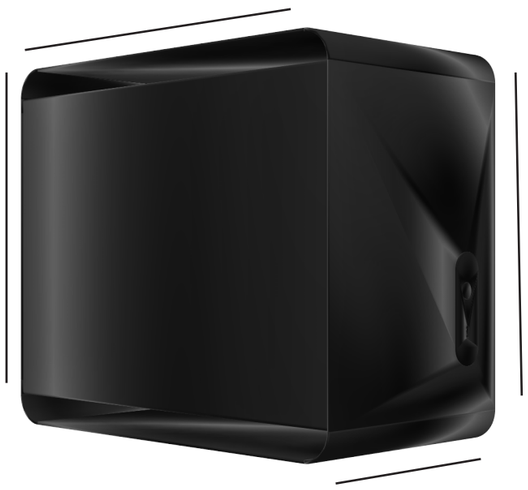
Блокировка устройства происходит от нажатия клавиши на кейпаде.

Функция самоуничтожения может быть вызвана через кейпад на заблокированном или разблокированном устройстве, при передвижении или вскрытии корпуса, а также от СМС владельца.

В коде предусмотрена возможность экстренной блокировки/разблокировки через СМС от владельца, или через мастер-пароль, которые можно отключить. Также предусмотрена возможность удаленного запуска самоуничтожения через СМС.

При блокировке, разблокировке, открытии дверей корпуса и включении на устройство владельца приходят СМС-оповещения.

При выключении ПК от электросети GSM-Shield теряет питание, поэтому разблокировать не включенный в сеть ПК невозможно (только при включенном мастер-пароле).



2 Сборка схемы

2.1 Описание компонентов

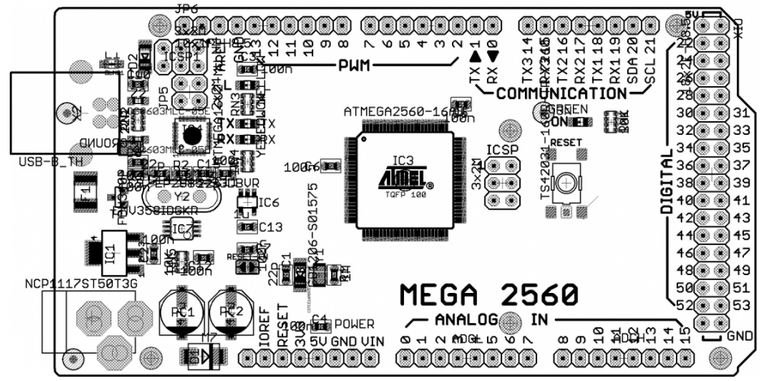
Основные компоненты проекта:

- Arduino Mega 2560 Rev3
- RFID-модуль MFRC522
- Датчик отпечатков пальцев FPM10A
- Кейпад
- Акселерометр MPU6050
- Ультразвуковой датчик расстояния HC-SR04
- Звуковой модуль с динамиком
- Реле 5V x2
- Электрозамки 12V x2
- GSM-Shield SIM900

Многие из этих комплектующих опциональны, также почти всё можно заменить аналогами. При доработке проект может быть изготовлен на печатной плате и поставлен на серийное производство.

2.1.1 Плата и микроконтроллер

За основу проекта была взята платформа Arduino, ввиду своей удобности для создания подобных прототипов, а именно плата с микроконтроллером Arduino Mega 2560. Выбор пал на эту плату из-за количества цифровых и аналоговых разъемов, которые были необходимы для подключения избыточного количество модулей, необходимых для функционирования системы.



2.1.2 RFID-модуль

В проекте используется RFID-модуль MFRC522, с помощью него совершается авторизация в устройство. В программном коде (см. Приложение 1) в переменной записан UID карты владельца ПК, с помощью которой производится разблокировка.

Подробнее об использованной для этого модуля библиотеке в §3.1.

2.1.3 Датчик отпечатков пальцев

В проекте был использован датчик отпечатков пальцев FPM10A. Датчики владельца ПК записываются в память датчика с помощью встроенного скрипта и хранятся на нем в зашифрованном виде. Подробнее об использованной для этого модуля библиотеке в §3.1.

Датчик отпечатков пальцев используется при авторизации в устройство.

2.1.4 Кейпад

В проекте был использован аналоговый кейпад 4x4 для ввода пин-кодов и управления устройством и авторизации. Кейпад работает по принципу матричной клавиатуры - несколько кнопок соединяются в прямоугольную матрицу, располагаясь в узлах рядов и столбцов из проводников.

2.1.5 Акселерометр

В проекте был использован акселерометр MPU6500, который реагирует на передвижения ПК и вызывает срабатывание сигнализации при перемещении ПК.

2.1.6 Ультразвуковой датчик расстояния

Использованный в проекте ультразвуковой датчик расстояния является частью системы сигнализации и устанавливается направленным в боковую дверь ПК с целью следить за её открытием.

2.1.7 Звуковой модуль с динамиком

Для работы звуковых уведомлений в проекте использовался звуковой модуль с динамиком. Он воспроизводит звуковые файлы с памяти SD-карты, установленной внутри него.

2.1.8 Реле

В проекте предполагается использование двух релейных модулей 5В: один устанавливается внутрь блока питания ПК и встает в разрыв между силовым проводом, чтобы контролировать питание БП; второй устанавливается в разрыв между 12В питанием электрозамка, который блокирует открытие корпуса при включенной защите.

2.1.9 Электрозамки

С целью защиты от физического отключения модуля защиты, в корпусе установлен электрозамок, который блокирует его открытие изнутри.

2.1.10 GSM-Shield

GSM-Shield служит средством коммуникации устройства с внешним миром: с помощью него отправляются уведомления о состоянии устройства, случайно сгенерированные пин-коды для авторизации, а также с помощью него работает функция удаленного самоуничтожения.

2.2 Построение схемы

В этом разделе будет описано подключениие всех модулей к плате и питание платы и модулей при установке в ПК.

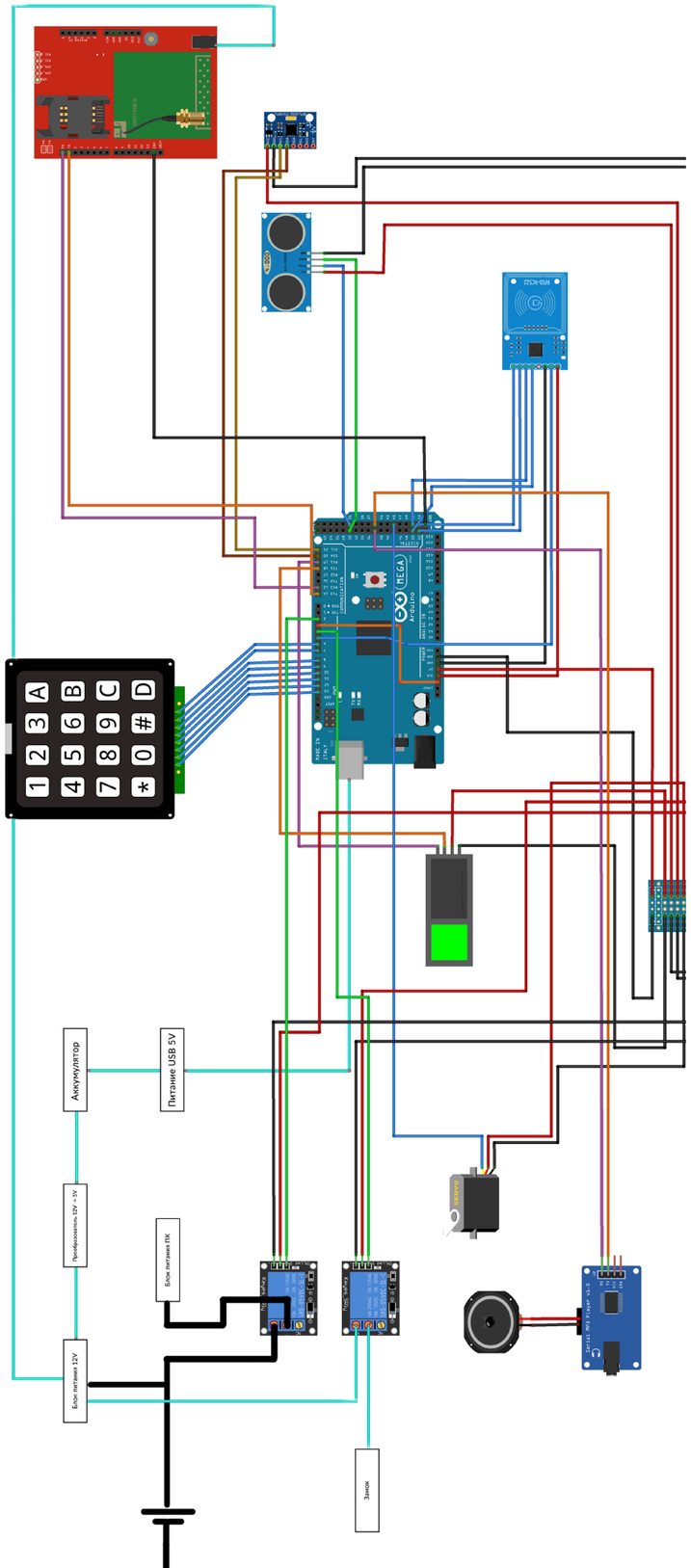
В таблице в пункте 2.2.1 записаны подключения каждого из модулей.

Более подробная схема подключения - в пункте 2.2.2.

2.2.1 Таблица подключений

RFID	SDA	D53
	SCK	D52
	MOSI	D51
	MISO	D50
	RST	D5
	GND	GND 3.3V
	VCC	VCC 3.3V
Кейпад	pin1	D13
	pin2	D12
	pin3	D11
	pin4	D10
	pin5	D9
	pin6	D8
	pin7	D7
	pin8	D6
Ультразвуковой сенсор	VCC	VCC 5V
	TRIG	D31
	ECHO	D30
	GND	GND 5V
Реле БП	VCC	VCC 5V
	GND	GND 5V
	IN	D4
Акселерометр	VCC	VCC 5V
	GND	GND 5V
	SCL	D21 (SCL)
	SDA	D20 (SDA)
Датчик отпечатков пальцев	VCC	VCC 3.3V
	GND	GND 3.3V
	TX	D18
	RX	D19
Реле Замки	VCC	VCC 5V
	GND	GND 5V
	IN	???

2.2.2 Схема подключения и питания



Для питания устройства в корпусе был установлен блок питания на 12В для GSM-Shield и электрозамков, а также понижающий модуль на 5В для питания платы.

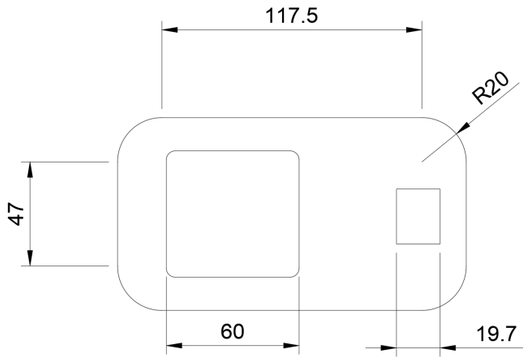
2.4 Моддинг корпуса ПК

Для вывода из корпуса кейпада и датчика отпечатка пальцев в нем были сделаны отверстия, через которые они были выведены и приклеены. Также было сделано отверстие для вывода антенны GSM.

Для этого проекта был специально выбран корпус, в котором блок питания не выводится полностью на задней стенке корпуса, а устанавливается внутри корпуса и проводится за корпус с помощью удлинителя. Таким образом от блока питания за корпус выводится лишь силовой разъем питания. Благодаря этой особенности появилась удобная возможность внутри корпуса подключить 12В блок питания к силовому кабелю, идущему к БП ПК.

Внутри самого блока питания было установлено реле, с помощью которого можно контролировать питание компьютера с помощью Arduino.

Для улучшения визуальной составляющей была изготовлена алюминиевая накладка, разработанная в среде 3D моделирования Autodesk Fusion 360, покрывающая верхнюю часть фронтальной части корпуса, в которой находятся кейпад и датчик отпечатков пальцев.



2.5 Модуль самоуничтожения

В качестве концепции модуля самоуничтожения предлагается использование сервопривода, который включает зажигательное устройство (пр. зажигалка), которая, в свою очередь, поджигает магниевую ленту, которая поджигает термитную смесь, расположенную на SSD-накопителе.

Термитная смесь — порошкообразная смесь алюминия с оксидами различных металлов (обычно железа). При воспламенении интенсивно сгорает с выделением большого количества тепла. Обычно имеет температуру горения 2300—2700 °C.

Небольшого количества термитной смеси будет достаточно для безвозвратного уничтожения памяти устройства.

Модуль защиты уже имеет функцию запуска самоуничтожения, которая запускает сервопривод: единственное, что остается – просто установить зажигатель и термитную смесь в ПК.

3 Разработка

В данном разделе описывается более подробное рассмотрение прошивки проекта: принципы его создания, концепция работы системы и последующая реализация.
Прошивка проекта написана на модифицированных версиях двух языков: C и C++, а также она объединяется с библиотекой AVR Libc для более удобного создания скетчей (программ) на платформе Arduino.

3.1 Использованные библиотеки

В проекте было использованы различные библиотеки для работы с модулями, а также для воплощения в жизнь некоторых функций системы:

“Adafruit_Fingerprint.h” – библиотека для работы с датчиком отпечатков пальцев (см. п. 2.1.3)
- “SoftwareSerial.h“ – библиотека для реализации последовательного интерфейса на любых цифровых выводах Arduino с помощью программных средств
- “Keypad.h” – библиотека для работы с кейпадом при помощи использования аналоговых выходов Arduino (см. п. 2.1.4)
- “Servo.h” – библиотека для работы с сервоприводом, который участвует в работе модуля самоуничтожения (см. п. 2.5)
- “SPI.h” – библиотека для использования датчиков, требующих подключения по шине SPI (последовательному периферийному интерфейсу, работающем на высоких скоростях)
- “I2Cdev.h” – библиотека дли использования датчиков, требующих подключения по интерфейсной шине I2C (данные передаются всего по 2 проводам)
- “MPU6050.h” – библиотека для использования акселерометра (см. п. 2.1.5)
- “RedMP3.h” – библиотека для работы с звуковым модулем через последовательный интерфейс (см. п. 2.1.7)
- “MFRC522.h” – библиотека для удобного подключения и использования RFID-модуля (см. п. 2.1.2)

3.2 Методы и функции

Прошивка построена на отдельных методах и функциях, которые сливаются воедино в бесконечном цикле loop()

setup() – функция инициализации, подключения и начальной настройки всех модулей и систем проекта для дальнейшей работы

recieveFingerprint() – функция для получения данных о приложенном к датчику отпечатков пальце и последующей обработки и сравнения с базой данных отпечатков.

sendSms() – функция, позволяющая отправлять SMS-уведомления на мобильное устройство с помощь GSM-шилда (см. п. 2.1.10)

sendPin() – функция, генерирующая уникальный код-пароль для входа в систему

destruction() – функция самоуничтожения, запускающая процессы модуля самоуничтожения (см. п. 2.5)

protection() – функция, опрашивающая ультразвуковой датчик расстояния и акселерометр на предмет изменения положения корпуса в пространстве

blockSystem() и unlockSystem() – функции, отвечающие за блокировку и разблокировку системы

updateGsm() – функция для получения данных с мобильного устройства через SMS и дальнейшего анализа этих данных

passwordInput() и checkPassword() – функции для ввода кода-пароля с клавиатуры и последующего сравнения с истинным кодом разблокировки или мастер-паролем.

3.3 Описание алгоритма

Основной цикл loop() (см. Приложение 1, строчка 462) состоит из нескольких крупных условий, каждое из которых включает свои полусловия и проверки каждого этапа работы.

Первое условие отвечает за то, чтобы при постановке системы на блокировку запускалась непосредственно функция защиты (protection ())

Все следующие условия отвечают за вывод системы из режима защиты в обычный режим работы.

Здесь поочерёдно проверяются все стадии верификации пользователя. Для начала происходит опрос RFID-модуля на приложенную карту, и пока её идентификационный номер не будет таким же, что и тот, который указан в прошивке, система не перейдет на следующий этап.

После удачной проверки карты запускается опрос сканера отпечатка пальцев. Алгоритм работы аналогичен предыдущей стадии: датчик отпечатка пальцев ожидает поднесения пальца и после проверяет очередной скан отпечатка с сохранённым в прошивке. После чего система переходит на завершающую стадию верификации пользователя.

На данной стадии в работу вступает кейпад. После того, как была отсканирована правильная карта и приложен верный палец, система генерирует уникальный 6-значный код пароль, который после этого отправляется на мобильное устройство с помощью функции sendPin().

После отправки сообщения начинается опрос кейпада для того, чтобы пользователь мог ввести код. Если всё введено верно, то запускается функция unlockSystem(), которая выводит систему в рабочее состояние.

Также на этом этапе пользователь может ввести код для запуска системы самоуничтожения, нажав на клавишу «D», не дожидаясь кода в SMS.

После разблокировки на кейпаде может быть нажата клавиша «C», позволяющая открыть корпусные замки

На каждом этапе можно сбросить введённые данные, нажав на символ «*»

4 Использование

В данном разделе будет описан процесс использования модуля ACCESSCODE вместе с ПК, на базе которого установлен модуль, а также основные детали касательно начала работы и функций устройства.

4.1 Настройка и загрузка прошивки

Перед началом сборки устройства необходимо прошить плату, на базе которой будет собираться модуль защиты. В программном коде есть несколько переменных, которые владельцу устройства нужно отредактировать под себя:

```
#define PHONE_NUMBER «+79*********» // номер телефона владельца
String MASTER_PASSWORD = «******»; // мастер-пароль (рекомендуется отключить в прошив
  ке)
String TRUE_RFID_ID_1 = «**********»; // UID RFID-карты
String TRUE_PASSWORD_DESTRUCT = «******»; // пароль для самоуничтожения системы через
  кейпад
String SELF_DESTRUCTION_SMS = «79****************»; // первые цифры - номер телефона
  владельца, затем 6-и значный пароль для самоуничтожения через СМС
String UNLOCK_SMS = «79****************»; // первые цифры - номер телефона владельца,
  затем 6-и значный пароль для разблокировки через СМС (рекомендуется отключить в про
  шивке)
String LOCK_SMS = «79****************»; // первые цифры - номер телефона владельца,
  затем 6-и значный пароль для блокировки через СМС (рекомендуется отключить в прошив
  ке)
```

После редактирования кода необходимо его скомпилировать и прошить плату через USB COM-порт.

4.3 Первое включение

Первое включение происходит при открытом корпусе. Благодаря проверке датчика расстояния система защиты не включается, когда корпус открыт, а вместо ее включения модуль уходит в «режим настройки» и срабатывает короткий звуковой сигнал. После того, как вся работа внутри корпуса будет закончена, для включения системы защиты необходимо нажать на кнопку ‘C’ на кейпаде, после этого будет включен «режим защиты».

Стоит отметить, что в дальнейшем, при включении модуля при закрытом корпусе всегда будет сразу запускаться «режим защиты».

4.4 Блокировка и разблокировка

Для разблокировки устройства нужно пройти трехэтапный алгоритм аутентификации, который включает в себя RFID-карту, датчик отпечатков пальцев, а также ввод случайно сгенерированного пин-кода, отправленного через СМС.

Первый шаг: приложите RFID-карту к кейпаду (за ним в прототипе располагается RFID-сканер). После того, как вы услышите звуковой сигнал, переходите ко второму шагу.

Второй шаг: приложите палец к датчику отпечатков пальцев. При удачном срабатывании вы услышите звуковой сигнал.

Третий шаг: в течении нескольких секунд после второго шага, на ваш телефон придет СМС с уникальным пин-кодом для входа в систему. Нажмите ‘A‘ и введите 6-и значный пин-код. Вы авторизовались в системе и теперь блок питания ПК включен, а датчики сигнализации (ультразвуковой датчик открытия корпуса и акселерометр) отключены.

Для блокировки вам необходимо нажать на кнопку ‘B‘ на кейпаде.

4.5 Функции устройства

Устройство в разблокированном режиме имеет возможность разблокировки боковой двери корпуса нажатием на ‘C‘: после этого замок разблокируется на 15 секнуд.

Нажатие на кнопку ‘D‘ с последующим введением пароля для самоуничтожения в любом режиме работы устройства вызывает функцию самоуничтожения.

Одна из важнейших функций устройства - возможность управления устройством через СМС сообщения: существует три пароля, отправя на номер устройства которые вы можете разблокировать, заблокировать, или вызвать функцию самоуничтожения.

Также стоит отметить то, что устройство практически полностью автономно (кроме GSM) и может работать в течении недели от встроенного аккумулятора. Таким образом, при отключении питания из сети, защита продолжает работать.

Стоит также упомянуть о функции выхода из режима авторизации нажатием на «*».

5 Описание прототипа и заметки

Прототипом является ПК с установленным модулем физической защиты ACCESSCODE.

ПК собран на базе процессора AMD A8-9600 на сокете AM4 и материнской платы ASRock на чипсете A320M форм-фактора Mini-ITX с 16 Гб DDR4 RAM.

Сборка в корпусе STREACOM DA2, в котором были вырезанны отверстия под датчик отпечатков пальцев и кейпад.

Также на корпусе была размещена декоративная накладка на передней панели.

Этот корпус был выбран из-за удобного модульного расположения комплектующих.

На ПК установлен дистрибутив ОС Kali Linux 2021.1, дизайн которого был кастомизирован под проект.

* В версии прототипа v.1.0 отсутствует электрозамок, а также функция защиты от DDos атаки на RFID и GSM модули и функция автоматического включения модуля GSM, которые будут добавлены в следующих версиях.

6 Ссылки

1. Adafruit Fingerprint Library - https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library

2. i2cdevlib - https://github.com/jrowberg/i2cdevlib

3. mpu6050 - https://github.com/ElectronicCats/mpu6050

4. RedMP3 - https://github.com/artronshop/OPEN-SMART-RedMP3

5. Arduino, official site - https://www.arduino.cc/

6. MCG 5184, Mechatronics Laboratory Manual, Theja Ram Pingali

7. Seytonic, Self Destructing SSD - https://www.youtube.com/watch?v=1ecB7xaHkeE

Inspired by Mr. Robot.

7 Приложениия

7.1 accessCode.ino // Программный код

```
1.   // Main executable file of the ACCCESCODE project — v.1.0
2.
3.  //---------------------------------------LIBRARIES---------------------------------------------------
4.   #include <Adafruit_Fingerprint.h>
5.   #include <SoftwareSerial.h>
6.   #include <Keypad.h>
7.   #include <Servo.h>
8.   #include <SPI.h>
9.   #include "I2Cdev.h"
10.  #include "MPU6050.h"
11.  #include "RedMP3.h"
12.  #include <MFRC522.h>
13.
14. //---------------------------------------DEFINES-----------------------------------------------------
15.  #define mySerial Serial1
16.  #define TIME_OUT 20
17.  #define MP3_RX 38 // pin RX player
18.  #define MP3_TX 39 // pin TX player
19.  #define RST_PIN 5 // pin RST RFID scanner
20.  #define SS_PIN 53 // pin SS RFID scanner
21.  #define PIN_TRIG 31 // TRIG pin of the distance sensor
22.  #define PIN_ECHO 30 // ECHO pin of the distance sensor
23.  #define PIN_POWER_SUPPLY 4 // logic pin of the power supply relay
24.  #define PIN_LOCKS 2 //logic pin of the locks relay
25.  #define RESET_PIN 3
26.
27. //---------------------------------------SOUNDS------------------------------------------------------
28.
29. //sounds
30.  #define BLOCK_SOUND 0x03 // lock sound
31.  #define UNLOCK_SOUND 0x03 // unlock sound
32.  #define TRUE_INPUT_SOUND 0x01 // sound of correct login (for RFID, Fingerpint)
33.  #define FALSE_INPUT_SOUND 0x05 // sound of incorrect login (for RFID, Fingerpint)
34.  #define KEYBOARD_SOUND 0x01 // sound of pressing the keyboard
35.  #define DESTRUCTION_SOUND 0x02 // self-destruction sound
36.  #define TRUE_INPUT_SOUND_2 0x06 // sound of correct login
37.  #define DOOR_SOUND 0x07 // very special sound
38.
39. //volume
40.  #define BLOCK_VOLUME 0x1c // lock sound volume
41.  #define UNLOCK_VOLUME 0x1a // unlock sound volume
42.  #define TRUE_INPUT_VOLUME 0x1a // sound volume of correct login
43.  #define FALSE_INPUT_VOLUME 0x16 // sound volume of incorrect login
44.  #define KEYBOARD_VOLUME 0x16 // volume of the sound of pressing the keyboard
45.  #define DESTRUCTION_VOLUME 0x16 // self destruct sound volume
46.
47. //time
48.  #define BLOCK_TIME 1500 // duration of the lock sound (мс)
49.  #define UNLOCK_TIME 1500 // duration of the unlock sound (мс)
50.  #define TRUE_INPUT_TIME 1500 // sound duration of correct login (мс)
51.  #define FALSE_INPUT_TIME 1000 // sound duration of incorrect login (мс)
52.  #define KEYBOARD_TIME 50 // duration of sound of pressing the keyboard (мс)
53.  #define DESTRUCTION_TIME 19000 // self-destruction sound duration (мс)
54.
55. //---------------------------------------NOTIFICATIONS-----------------------------------------------
56.
57.  #define PHONE_NUMBER "+79000000000"
58.  #define UNLOCK_NOTIFY "Your system was unlocked\nACCESSCODE" // message text for unlock process
59.  #define DOORS_NOTIFY "Side doors on your system were unlocked\nACCESSCODE" // message text for opening doors
60.  #define DESTRUCTION_NOTIFY " !!! SELF-DESTRUCTION PROCESS WAS STARTED !!!\nACCESSCODE" // message text for destruction process
61.  #define WELCOME_NOTIFY "Your protection system is up\nACCESSCODE"
62.  #define SETUP_NOTIFY "Your protection system is turned on in setup mode: the side doors are open\nACCESSCODE"
63.
64. //---------------------------------------CONSTANTS---------------------------------------------------
65.
66.  String MASTER_PASSWORD = "000000"; // will remove
67.
68.  String RECIEVED_PASSWORD;
69.  String TRUE_RFID_ID_1 = "1111111111"; //ID of RFID card
70.  String TRUE_RFID_ID_2 = "0000000000";
71.  String TRUE_PASSWORD_UNLOCK = "000000"; //sms login password
72.  String TRUE_PASSWORD_DESTRUCT = "333333"; //self-destruction password
73.  int COUNTER_LIMIT = 5; // delay in seconds for resetting the event counter
74.  int COUNTER_MAX_EVENTS = 5; // number of gyro position changes before starting self-destruction
75.  int MAX_LENGTH_SONAR = 10; // distance between distance sensor and side cover
76.
77.  String SELF_DESTRUCTION_SMS = "79000000000111111"; // +79000000000, 111111
78.  String UNLOCK_SMS = "79000000000222222"; // +79000000000, 222222
79.  String LOCK_SMS = "79000000000333333"; // +79000000000, 333333
80.
81. //---------------------------------------OTHER_INITIALISATIONS----------------------------------------
82.
83.  MPU6050 accgyro;
84.  Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
85.  Servo myservo;
```

```
  MFRC522 mfrc522(SS_PIN, RST_PIN);
  MP3 mp3(MP3_RX, MP3_TX);
  unsigned long int t1;
  const byte ROWS = 4; // number of lines in the keyboard
  const byte COLS = 4; // the number of columns in the keyboard
  char keys[ROWS][COLS] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
    };
  byte rowPins[ROWS] = {9,8,7,6}; // rows connection pins
  byte colPins[COLS] = {13,12,11,10}; // collumns connection pins
  Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

  int pos = 0; // home position of the servo in degrees

  long DURATION_SONAR, LENGTH_SONAR;
  int low_x, low_y, low_z;
  int high_x, high_y, high_z;
  int COUNTER_EVENTS = 0;
  int COUNTER = 0;
  int fingerIDS = 0;

  char key;
  String notify;

  String messageCheck;

//-------------------------------------BOOLS-------------------------------------------------------

  bool isProtectionOn = true; // current position of the protection function
  bool isRFIDChecked = false; // RFID tag check position
  bool isFingerChecked = false; // scanned print check position
  bool isPinSent = false; // verification of sending a message with an unlock code

//-------------------------------------------------------------------------------------------------

void rfidInitialization(){ // RFID module initialization
  while (!Serial);
  SPI.begin();
  mfrc522.PCD_Init();
  delay(4);
  mfrc522.PCD_DumpVersionToSerial();
}

void playSound(int8_t index, int8_t volume, int delayTime){ // music playback (sound index, volume, duration)
  delay(100);
  mp3.playWithVolume(index,volume);
  delay(delayTime);
}

//-------------------------------------------------------------------------------------------------

void setup(){
  digitalWrite(RESET_PIN, HIGH);
  pinMode(RESET_PIN, OUTPUT);

  Serial.begin(9600);
  Serial2.begin(9600);
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
  long int t = millis();
  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(5);
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);
  DURATION_SONAR = pulseIn(PIN_ECHO, HIGH);
  LENGTH_SONAR = (DURATION_SONAR / 2) / 29.1;


  if (LENGTH_SONAR > 3){
    playSound(TRUE_INPUT_SOUND_2, TRUE_INPUT_VOLUME, TRUE_INPUT_TIME);
    sendSms(SETUP_NOTIFY);
  while (key != 'C'){
     key = keypad.getKey();
     if (key != NO_KEY){
       playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
       Serial.println(key);
     }
  }
}

  finger.begin(57600);
  rfidInitialization();
  accgyro.initialize();
  pinMode(PIN_LOCKS, OUTPUT); //relay 1
  pinMode(PIN_POWER_SUPPLY, OUTPUT); //relay 2

  digitalWrite(PIN_POWER_SUPPLY, HIGH);
  //digitalWrite(PIN_LOCKS, HIGH);

  finger.getParameters();
  finger.getTemplateCount();

  int ax, ay, az, gx, gy, gz;
  accgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  low_x = ax - 1000;
  low_y = ay - 1000;
  low_z = az - 1000;
  high_x = ax + 1000;
  high_y = ay + 1000;
  high_z = az + 1000;
  delay(100);

  Serial.println("INITIALIZATION COMPLETED");
  delay(500);
  playSound(BLOCK_SOUND, BLOCK_VOLUME, BLOCK_TIME);
  sendSms(WELCOME_NOTIFY);
}

void resetFunc(){
  digitalWrite(RESET_PIN, LOW);
}

bool checkFingerprint(int current){ // checking the scanned fingerprint for presence in the database
  if (current != 0){
    playSound(TRUE_INPUT_SOUND, TRUE_INPUT_VOLUME, TRUE_INPUT_TIME);
    return true;
    }
  else{
    return false;
    }
}
bool recieveFingerprint() { // obtaining a print using a scanner
  protection();
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      break;
    case FINGERPRINT_NOFINGER:
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      return p;
    case FINGERPRINT_IMAGEFAIL:
      return p;
    default:
      return p;
    }
  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      break;
    case FINGERPRINT_IMAGEMESS:
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      return p;
    case FINGERPRINT_FEATUREFAIL:
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      return p;
    default:
    return p;
  }

  p = finger.fingerSearch();
  if (p == FINGERPRINT_OK){
    fingerIDS = finger.fingerID;
    }
  else if (p == FINGERPRINT_PACKETRECIEVEERR){
    return p;
    }
  else if (p == FINGERPRINT_NOTFOUND){
    return p;
    }
  else{
    return p;
    }
}

void restInPeace(){ // shutdown cycle after self-destruction
  while (true){
    playSound(DESTRUCTION_SOUND, DESTRUCTION_VOLUME, DESTRUCTION_TIME);
    delay(2400);
    }
}

void sendSms(String current_data) {
    delay(1000);
    Serial2.println("AT");
    delay(100);
    Serial2.println("AT+CMGF=1");
    delay(200);
    Serial2.print("AT+CMGS=\"");
    Serial2.print(PHONE_NUMBER); // mobile number with country code
    Serial2.println("\"");
    delay(500);
    Serial2.println(current_data); // Type in your text message here
    delay(500);
    Serial2.println((char)26); // This is to terminate the message
```

```
    delay(1000);
  }

  void sendPin(){
    randomSeed(micros());
    Serial.println("SENDING PIN CODE");
    TRUE_PASSWORD_UNLOCK = random(100000, 1000000);
    notify = "Your verification code is " + TRUE_PASSWORD_UNLOCK + " \nACCESSCODE";
    Serial.println(notify);
    sendSms(notify);
    isPinSent = true;
  }

  void destruction(){ // self-destruction function
    if (isProtectionOn == false){
      digitalWrite(PIN_POWER_SUPPLY, HIGH);
    }
    sendSms(DESTRUCTION_NOTIFY);
    myservo.attach(45);

    Serial.println("STARTING SELF-DESTRUCTION");
    for (pos = 0; pos <= 180; pos += 1) {
      myservo.write(pos);
      delay(10);
      }

    restInPeace();
  }

  void protection(){ // sensor polling protection function
    Serial.println("PROTECTION IS RUNNING");
    long int t = millis();
    int ax, ay, az, gx, gy, gz;
    accgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    digitalWrite(PIN_TRIG, LOW);
    delayMicroseconds(5);
    digitalWrite(PIN_TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(PIN_TRIG, LOW);
    DURATION_SONAR = pulseIn(PIN_ECHO, HIGH);
    LENGTH_SONAR = (DURATION_SONAR / 2) / 29.1;
    if (ax > high_x or ax < low_x or ay > high_y or ay < low_y or az > high_z or az < low_z or LENGTH_SONAR > MAX_LENGTH_SONAR){
      COUNTER_EVENTS++;
      Serial.println("TRACKING SENSORS");
      }
    if (COUNTER_EVENTS >= COUNTER_MAX_EVENTS){
      destruction();
      }
    COUNTER++;
    if (COUNTER >= COUNTER_LIMIT){
      COUNTER = 0;
      COUNTER_EVENTS = 0;
      }
    delay(50);

  }

  String passwordInput(){ // function of entering 6-digit codes from the keyboard
    playSound(TRUE_INPUT_SOUND_2, TRUE_INPUT_VOLUME, TRUE_INPUT_TIME);
    String passkeys;
    for (int i = 0; i < 6; i++){
      char key = keypad.getKey();
      while (key == NO_KEY){
        if (isProtectionOn){
          protection();
          }
        key = keypad.getKey();
      }
      playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
      passkeys.concat(key);
    }
    return passkeys;
  }

  int checkPassword(char input_type, String input_password){ // checking the correctness of the entered code and processing it
    if ((input_type == 'A' && input_password == TRUE_PASSWORD_UNLOCK) or (input_password == MASTER_PASSWORD)){
      return 1; // return 1 if the login password was checked and the password was entered correctly
      }
    else if (input_type == 'D' && input_password == TRUE_PASSWORD_DESTRUCT){
      return 2; // return 2 if the self-destruction password was checked and the password was entered correctly
      }
    else {
      return 3;
      }
  }

  bool checkRFID(){ // verification of the correctness of the presented RFID key
    if ( ! mfrc522.PICC_IsNewCardPresent()) {
      return false;
      }
    if ( ! mfrc522.PICC_ReadCardSerial()) {
      return false;
      }

    unsigned long UID_unsigned;
    UID_unsigned =  mfrc522.uid.uidByte[0] << 24;
    UID_unsigned += mfrc522.uid.uidByte[1] << 16;
    UID_unsigned += mfrc522.uid.uidByte[2] <<  8;
    UID_unsigned += mfrc522.uid.uidByte[3];
    String UID_string = (String)UID_unsigned;

    Serial.println(UID_string);
    if (UID_string == TRUE_RFID_ID_1 or UID_string == TRUE_RFID_ID_2){
      playSound(TRUE_INPUT_SOUND, TRUE_INPUT_VOLUME, TRUE_INPUT_TIME);
      return true;
      }
    else {
      playSound(FALSE_INPUT_SOUND, FALSE_INPUT_VOLUME, FALSE_INPUT_TIME);
      return false;
      }
  }

String updateGsm(){
  char data;
  String result;
  String info;
  String returnedPassword;
  while(Serial2.available())
  {
    delay(10);
    data = Serial2.read();
    result.concat(data);
  }

  if (result[0] = "+"){
    info = result;
  }

  if (info != ""){
    for (int i = 10; i < 21; i++){
    returnedPassword.concat(info[i]);
    }

    for (int i = 50; i < 56; i++){
    returnedPassword.concat(info[i]);
    }
  }

  return returnedPassword;

}

  void zeroBools(){
    fingerIDS = 0;
    isRFIDChecked = false;
    isFingerChecked = false;
    isPinSent = false;
  }

  void blockSystem(){ // lock function
    isProtectionOn = true;
    //digitalWrite(PIN_LOCKS, HIGH);
    digitalWrite(PIN_POWER_SUPPLY, HIGH);
    Serial.print("SYSTEM LOCKED");
    zeroBools();
    //playSound(BLOCK_SOUND, BLOCK_VOLUME, BLOCK_TIME);
    //sendSms(WELCOME_NOTIFY);
    resetFunc();
    delay(150);

  }

  void openDoors(){ // function of disabling door locks
    Serial.print("DOOR OPENED, LOCKS ARE DISABLED");
    sendSms(DOORS_NOTIFY);
    //digitalWrite(PIN_LOCKS, LOW);
    for (int i = 0; i < 12; i++){
      playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME); // Можно заменить на DOOR_SOUND
      delay(1000);
    }
    for (int i = 0; i < 3; i++){
      playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME); // Можно заменить на DOOR_SOUND
      delay(50);
    }
    //digitalWrite(PIN_LOCKS, HIGH);
  }

  void unlockSystem(){
    isProtectionOn = false;
    zeroBools();
    digitalWrite(PIN_POWER_SUPPLY, LOW);
    playSound(UNLOCK_SOUND, UNLOCK_VOLUME, UNLOCK_TIME);
    Serial.println("SYSTEM UNLOCKED");
    sendSms(UNLOCK_NOTIFY);
  }

  void loop(){
    if (isProtectionOn){
      protection();
  }
```

```
466.
467.  if (isRFIDChecked != true && isProtectionOn){
468.    isRFIDChecked = checkRFID();
469.    }
470.
471.  if (isRFIDChecked && isProtectionOn){
472.      if (isFingerChecked != true){
473.        recieveFingerprint();
474.        isFingerChecked = checkFingerprint(fingerIDS);
475.        Serial.print(isFingerChecked);
476.
477.      key = keypad.getKey();
478.      if (key != NO_KEY){
479.        playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
480.        Serial.println(key);
481.        if (key == '*'){
482.          delay(100);
483.          playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
484.          zeroBools();
485.        }
486.      }
487.        }
488.      else {
489.        if (isPinSent == false){
490.          sendPin(); // генерирует pin, отправляет pin, назначает pin
491.          }
492.        else {
493.          key = keypad.getKey();
494.          if (key != NO_KEY){
495.          playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
496.            Serial.println(key);
497.            if (key == 'A' or key == 'D'){
498.              Serial.println("PASS IS BEING ENTERED");
499.              String enteredPassword = passwordInput();
500.              Serial.println(enteredPassword);
501.              int checkedTypeOfPassword = checkPassword(key, enteredPassword);
502.                if (checkedTypeOfPassword == 1){
503.                  unlockSystem();
504.                  }
505.                else if (checkedTypeOfPassword == 2){
506.                  destruction();
507.                  }
508.                else if (checkedTypeOfPassword == 3){
509.                  playSound(FALSE_INPUT_SOUND, FALSE_INPUT_VOLUME, FALSE_INPUT_TIME);
510.                  }
511.          }
512.            else if (key == '*'){
513.              delay(100);
514.              playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
515.              zeroBools();
516.            }
517.        }
518.      }
519.    }
520.    }
521.
522.    else {
523.      key = keypad.getKey();
524.      if (key != NO_KEY){
525.        playSound(KEYBOARD_SOUND, KEYBOARD_VOLUME, KEYBOARD_TIME);
526.        Serial.println(key);
527.        if (key == 'D'){
528.          Serial.println("PASS IS ENTERING");
529.          String enteredPassword = passwordInput();
530.          Serial.println(enteredPassword);
531.          int checkedTypeOfPassword = checkPassword(key, enteredPassword);
532.            if (checkedTypeOfPassword == 2){
533.                destruction();
534.                }
535.            else {
536.                playSound(FALSE_INPUT_SOUND, FALSE_INPUT_VOLUME, FALSE_INPUT_TIME);
537.                }
538.           }
539.        if (key == 'B' && isProtectionOn == false){
540.          blockSystem();
541.          }
542.        if (key == 'C' && isProtectionOn == false){
543.          openDoors();
544.          }
545.       }
546.    }
547.
548.
549.     messageCheck = updateGsm();
550.
551.     if (messageCheck.equals(SELF_DESTRUCTION_SMS)){
552.      destruction();
553.     }
554.     if (messageCheck.equals(UNLOCK_SMS) && isProtectionOn){
555.      unlockSystem();
556.     }
557.     if (messageCheck.equals(LOCK_SMS) && isProtectionOn == false){
558.      blockSystem();
559.     }
560.   }
```