Serwer FTP z zabezpieczeniem “tamper”

Dokumentacja techniczna

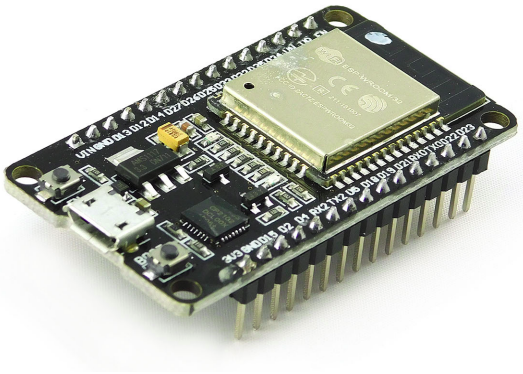
Autorzy: Szymon Stępień, Gabriel Cyganek

SPIS TREŚCI

WYKORZYSTANE KOMPONENTY

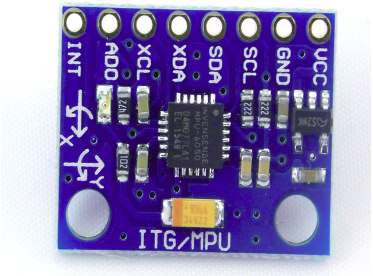
Do budowy urządzenia wykorzystano następujące komponenty:

➔ *Moduł rozwojowy DOIT ESP32 DEVKIT V1*



Źródło: nettigo.pl

➔ *Układ MPU 6050 - żyroskop i akcelerometr*

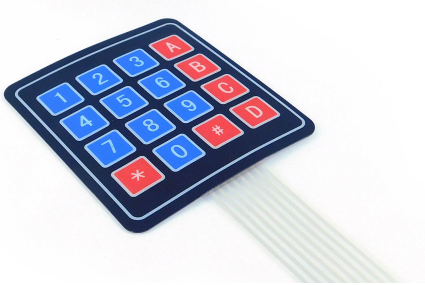


Źródło: nettigo.pl

➔ *Moduł z fotorezystorem i potencjometrem*

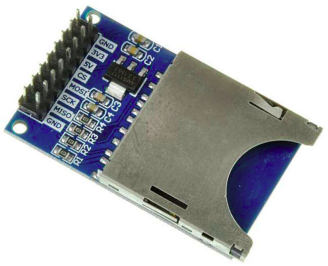
Źródło: botland.pl

➔ *Klawiatura membranowa - 16 klawiszy*



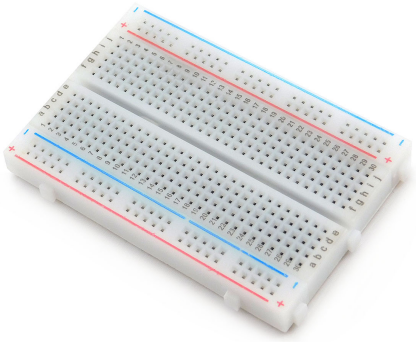
Źródło: nettigo.pl

➔ *Moduł czytnika kart SD - komunikacja z użyciem interfejsu SPI*



Źródło: nettigo.pl

➔ *2x płytka stykowa 400 otworów*

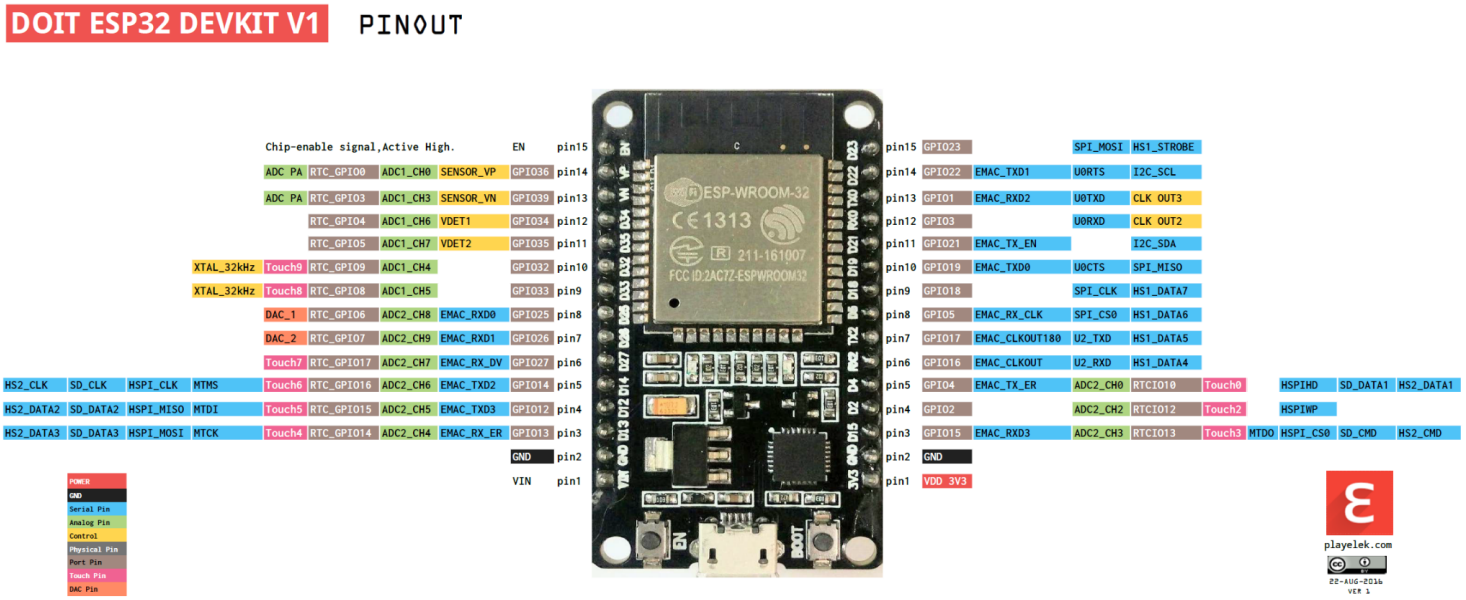


Źródło: nettigo.pl

➔ *Diody RGB*
➔ *Zworki do płytek stykowych*
➔ *Rezystory (do diod 220Ω, do czujnika światła 10kΩ)*

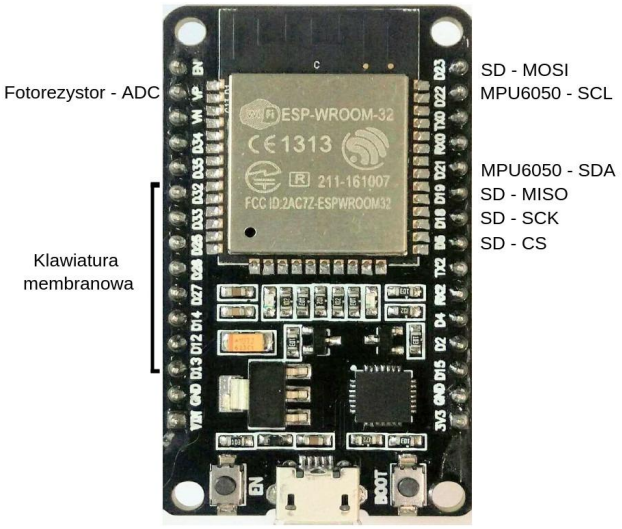
POŁĄCZENIE KOMPONENTÓW

Moduł rozwojowy DOIT ESP32 DEVKIT V1 udostępnia następujące wyprowadzenia:

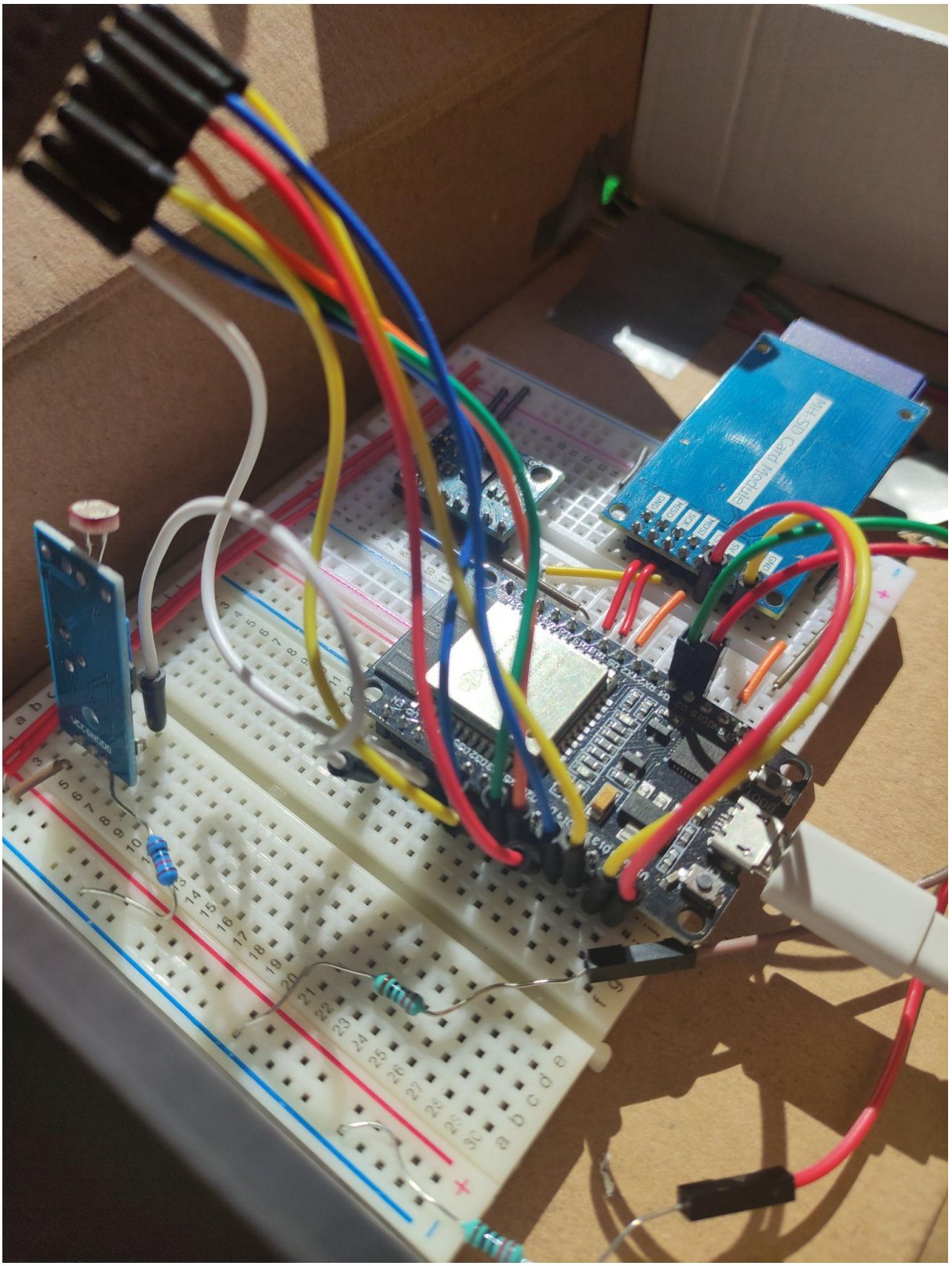


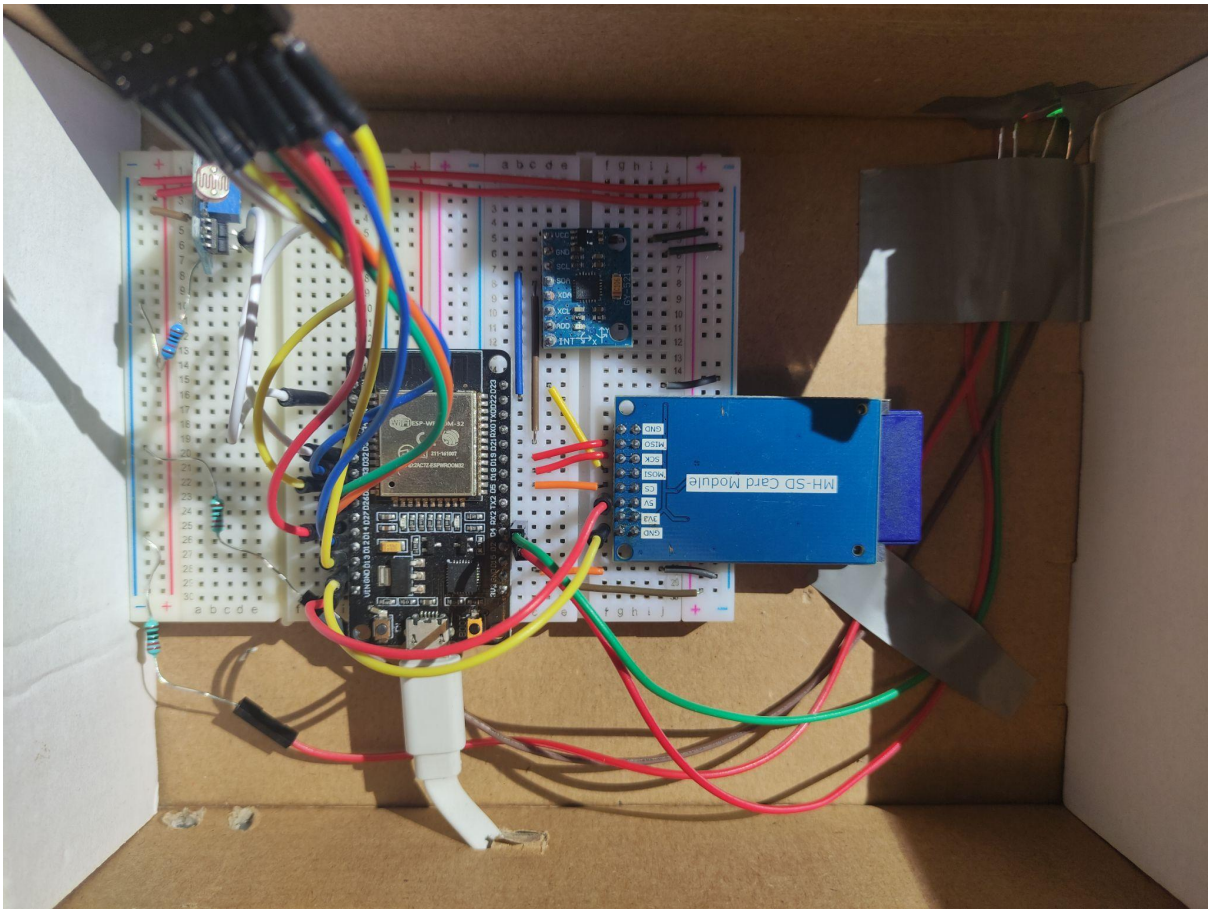
Źródło: github.com/playelek/pinout-doit-32devkitv1

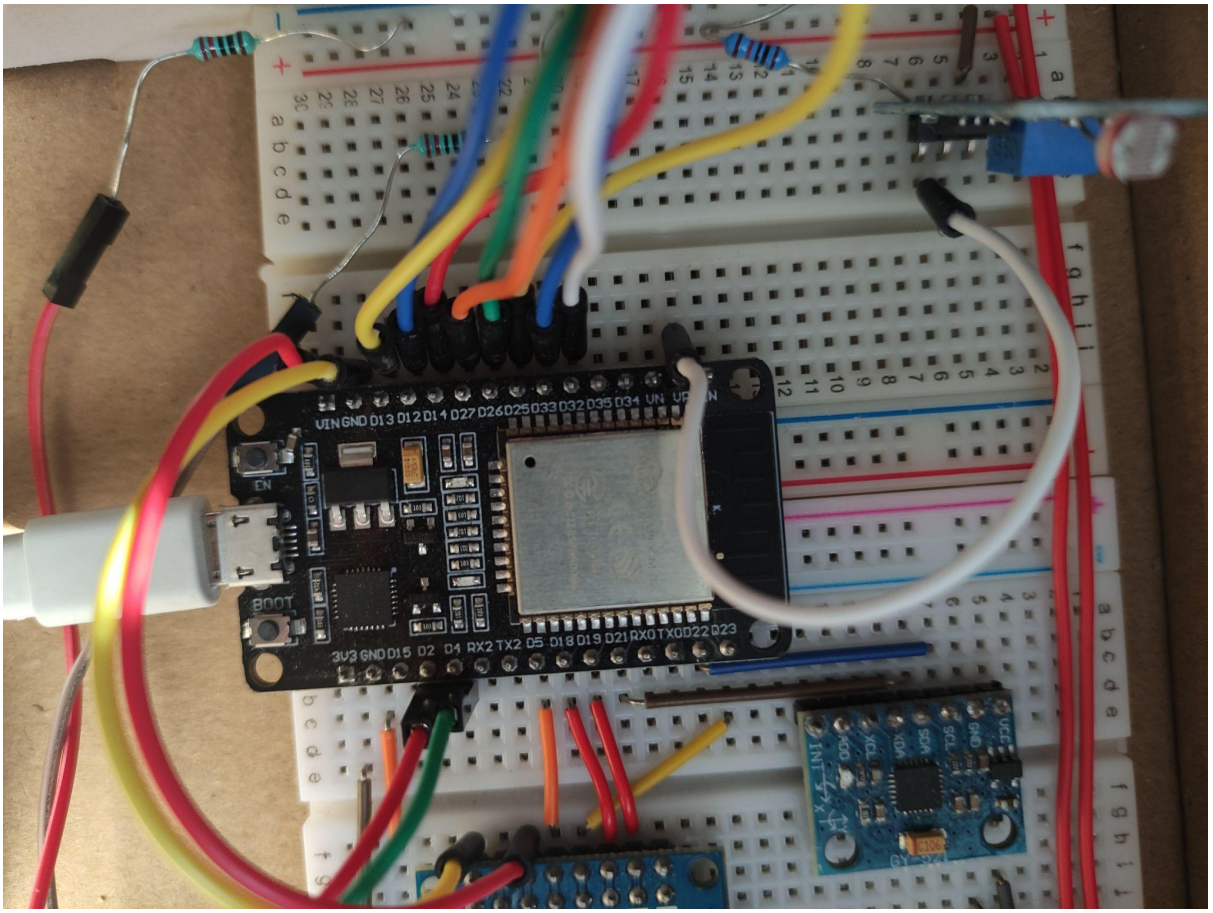
Wykorzystane przez nas wyprowadzenia (pominięto GND oraz zasilanie):



Ostatecznie urządzenie wygląda następująco:


MH-SD Card Module


MH-SD Card Module


VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
EN
BOOT

PODSTAWOWE INFORMACJE O URZĄDZENIU

Urządzenie służy do przechowywania zaszyfrowanych danych na dołączonej karcie SD. Mikrokontroler wyposażony jest w czujniki pozwalające na wykrywanie anomalii - sensor ruchu oraz fotorezystor. Anomalie traktowane są jako próby włamania do urządzenia. Niepoprawne podanie pinu dostępu do urządzenia więcej niż dozwoloną liczbę razy jest obsługiwane jak anomalia.

STAN BEZPIECZNY I NIEBEZPIECZNY

Tryb niebezpieczny jest trybem domyślnym po uruchomieniu urządzenia. Zapewnia on wykrywanie i obsługę anomalii i nie pozwala na ingerencję w układ. Poprawne podanie pinu przy użyciu klawiatury membranowej pozwala przejść w stan bezpieczny. W stanie bezpiecznym anomalie są ignorowane, co pozwala na poruszanie urządzeniem, zmianę karty SD i dowolne modyfikacje. Ponowne poprawne wprowadzenie pinu w stanie bezpiecznym pozwala na przywrócenie trybu niebezpiecznego, jeśli urządzenie nie będzie wykrywać aktywności użytkownika przez 3 minuty to system automatycznie wróci w stan niebezpieczny. Tryb bezpieczny udostępnia również dodatkowe opcje korzystania z klawiatury takie jak reset czy wypisanie preferencji na porcie szeregowym.

Do sygnalizacji stanu urządzenia wykorzystano diody. Czerwona dioda jest zapalana przy uruchomieniu urządzenia i gaśnie po poprawnym podaniu PINu i inicjalizacji systemu, jest ponownie zapalana gdy została wykryta ingerencja. Zielona dioda jest zapalana gdy urządzenie pracuje w trybie bezpiecznym. W normalnych warunkach żadna dioda nie powinna się świecić.

WYKRYCIE I OBSŁUGA ANOMALII

Przy wykryciu ingerencji system wykonuje następujące kroki:

1. Usuwany oraz nadpisywany zerami zostaje klucz tajny użyty do szyfrowania danych na karcie SD (jeśli użytkownik wybrał opcję persystencji klucza w pamięci Flash).
2. Wysłany zostaje email powiadamiający o ingerencji na adres podany w pliku konfiguracyjnym informujący o naruszeniu bezpieczeństwa.
3. Naruszenie bezpieczeństwa zostaje sygnalizowane za pomocą diody.

4. Zatrzymywana zostaje usługa FTP, pomijając wcześniej nawiązane połączenie, oraz zadania odpowiedzialne za obsługę klawiatury i czujników.
5. System wchodzi w tryb zawieszenia.

DZIAŁANIE MODUŁÓW URZĄDZENIA

Najważniejszymi modułami urządzenia są:

➔ *serwer FTP*

Podstawowe informacje

Główną funkcjonalnością urządzenia jest udostępnienie możliwości przechowywania danych na karcie SD dołączonej do mikrokontrolera poprzez czytnik kart SD. Do zapisu i odczytu danych służy serwer FTP. Liczba równoległych połączeń do serwera jest ograniczona do jednego. Serwer posiada dwa połączenia - połączenie służące do odbierania i odpowiadania na komendy zalogowanego użytkownika oraz połączenie służące do transferu danych, które domyślnie działa w trybie aktywnym.

Obsługiwany typ danych oraz tryb przesyłania danych

Obsługiwane są typy danych Ascii Non-print oraz Image. Obsługiwany jest tryb strumieniowy przesyłania danych.

Status sesji klienta

Po ustaleniu połączenia klienta z serwerem tworzona jest sesja klienta, która może przyjąć jeden z pięciu stanów:

1. **AWAIT_USERNAME** - oczekiwanie na podanie nazwy użytkownika przez klienta
2. **AWAIT_PASSWORD** - oczekiwanie na podanie hasła przez klienta
3. **AWAIT_COMMAND** - oczekiwanie na podanie przez klienta komendy do przetworzenia przez serwer
4. **LOGOUT** - stan spowodowany przez otrzymanie od klienta komendy QUIT
5. **REINITIALIZATION** - stan spowodowany przez otrzymanie od klienta komendy REIN

Status transferu danych

1. **NO_TRANSFER** - kiedy nie przeprowadzany jest tranfser danych pomiędzy klientem a serwerem oraz klient nie przesłał żądania odczytu lub zapisu danych, stan transferu danych dla sesji klienta jest równy NO_TRANSFER.
2. **READ_IN_PROGRESS** - jeśli klient zażądał poprzez przesłanie odpowiedniej komendy do serwera odczytu danych oraz zostało nawiązane połączenie potrzebne do przesłania danych z serweru do klienta, stan transferu danych dla sesji klienta zmienia się na READ_IN_PROGRESS.
3. **WRITE_IN_PROGRESS** - jeśli klient zażądał poprzez przesłanie odpowiedniej komendy do serwera zapisu danych oraz zostało nawiązane połączenie potrzebne do przesłania danych od klienta do serwera, stan transferu danych dla sesji klienta zmienia się na WRITE_IN_PROGRESS.
4. **FINISHED** - stan tymczasowy po zakończeniu transmisji danych pomiędzy serwerem a klientem. Serwer po detekcji tego stanu jak najszybciej zamyka połączenie na sockecie, który służył do transmisji danych.

Tryby transferu danych

1. **Pasywny** - serwer nasłuchuje na inicjalizację połączenia od strony klienta, aby rozpocząć transfer danych
2. **Aktywny** - serwer inicjalizuje połączenie z klientem na wskazanym adresie, aby rozpocząć transfer danych

Do zamiany trybów transferów danych oraz modyfikacji ich działania klient powinien wykorzystać komendy PASV oraz PORT

Szyfrowanie danych

Do szyfrowania danych używany jest algorytm ChaCha-20 używający 128-bitowego klucza tajnego podanego przez użytkownika w pliku z konfiguracją urządzenia. Algorytm do poprawnego szyfrowania wymaga wykorzystywania wektora inicjalizacyjnego (IV), jest to 64 bitowa liczba losowo generowana przez urządzenie i zapisywana w postaci jawnej na początku każdego pliku, przy odszyfrowaniu jest ona z niego usuwana. Każdy plik szyfrowany jest z wykorzystaniem innego wektora inicjalizacyjnego. Z uwagi na brak możliwości wykonania operacji dodawania danych do pliku bez ponownego szyfrowania całego

pliku komenda ta została wycofana z implementacji serwera FTP na tym urządzeniu.

Obsługiwane komendy FTP

Serwer FTP obsługuje następujące komendy:

- USER,
- PASS,
- AUTH,
- CWD,
- CDUP,
- REIN,
- QUIT,
- FEAT,
- RETR,
- STOR,
- STOU,
- ALLO,
- REST,
- RNFR,
- RNTO,
- MDTM,
- ABOR,
- DELE,
- RMD,
- PORT,
- PASV,
- TYPE,
- STRU
- MODE.

Semantyka komend i opis zwracanych odpowiedzi jest zgodny ze standardem FTP z rfc 959.

W celu połączenia z serwerem należy użyć dowolnego klienta FTP, na przykład Filezilla. Jako parametry połączenia należy wpisać adres IP urządzenia (jest wypisywany na porcie szeregowym po wpisaniu PINu), port TCP 21 oraz nazwę użytkownika i hasło podane w pliku konfiguracyjnym na karcie SD.

➔ *sensor ruchu*

Czujnik ruchu został oparty na akcelerometrze oraz żyroskopie operujących na współrzędnych XYZ. Odpowiednio skonfigurowane stałe czułości oraz automatyczna kalibracja pozwala uśredniając pomiary uzyskać bardzo dobre możliwości wykrywania anomalii. Do implementacji wykorzystano bibliotekę *Adafruit_MPU6050*, dostarczająca wygodniejszego API do obsługi układu MPU6050.

➔ *fotorezystor*

Czujnik światła po uruchomieniu urządzenia dokonuje automatycznej kalibracji na podstawie pierwszych odczytanych wartości natężenia światła. Kolejne odczytywane wartości są porównywane do tej wyliczonej w czasie kalibracji. Jeśli występują między nimi zbyt duże różnice, następuje zgłoszenie anomalii.

➔ *klawiatura membranowa*

Klawiatura membranowa służy do wprowadzania pinu podanego w konfiguracji urządzenia przed jego uruchomieniem. Pin może składać się tylko z cyfr. Klawisz *#* służy do zatwierdzania wprowadzonego pinu. Klawisz *** pozwala ponownie rozpocząć wprowadzanie pinu czyszcząc poprzednio wprowadzony ciąg cyfr. W trybie bezpiecznym dostępne są dodatkowe opcje:

- **A** - wypisuje preferencje (pomijając klucz tajny) na porcie szeregowym
- **DDDD** - usuwa z pamięci Flash zapisane preferencje i resetuje urządzenie, pozwala to na zmianę preferencji, jeśli na karcie SD jest zapisany plik /.conf, opisany w sekcji konfiguracja
- **ABCD** - resetuje urządzenie

KOD ŹRÓDŁOWY

Kod źródłowy dostępny jest w repozytorium na platformie GitHub - repozytorium. Działanie poszczególnych modułów urządzenia opisane jest dokładniej w poprzednim rozdziale. Najważniejszymi sekcjami kodu są:

➔ *src/main.cpp*

Miejsce startowe oprogramowania urządzenia. W kodzie wykorzystywane są funkcje systemu FreeRTOS do wykonywania zdefiniowanych zadań - obsługa czujnika światła, czujnika ruchu, klawiatury membranowej, zapobiegania włamaniom oraz serwera FTP. Każde z tych zadań jest obsługiwane na osobnym wątku, z czego serwer FTP uruchamiany jest na pierwszym rdzeniu ESP32, a pozostałe zadania na drugim. Ponadto, inicjalizowane jest działanie urządzenia, zostaje załadowana konfigurację z karty SD i następuje połączenie modułu z siecią poprzez Wi-Fi.

➔ *lib/FTPServer*

Folder zawierający implementację serwera FTP. Plik *FTPServer.cpp* posiada metodę uruchamiającą serwer i akceptującą połączenia od klientów oraz korzysta z modułów zdefiniowanych w osobnych podfolderach *CommandProcessor* oraz *DataProcessor* do obsługi ustanowionych sesji klienta. Pierwszy podfolder stanowi moduł obsługi komend FTP odebranych od klienta, drugi zawiera obsługę szyfrowanego za pomocą szyfru ChaCha transferu danych pomiędzy klientem a serwerem.

➔ *lib/KeypadModule.cpp*

Implementacja obsługi klawiatury membranowej służącej do wpisywania zdefiniowanego w konfiguracji urządzenia pinu oraz podawania opcji w trybie bezpiecznym.

➔ *lib/LightSensor.cpp*

Plik zawiera kod implementujący kalibrację fotorezystora, odczytywanie i przetwarzanie wartości pobieranych z fotorezystora oraz wykrywanie anomalii.

➔ *lib/MotionSensor/MotionSensor.h*

Kod zawierający implementację inicjalizacji i kalibracji wykrywacza ruchu. Ponadto, są też metody odczytujące i przetwarzające wartości ze wspomnianego wyżej modułu oraz wykrywające anomalie.

➔ *lib/TamperGuard/TamperGuard.cpp*

Kod zawierający implementację opisanej wcześniej w rozdziale **WYKRYCIE I OBSŁUGA ANOMALII** obsługi anomalii. Anomalie zgłaszane są przez zadania systemu odpowiedzialne za ich wykrywanie na podstawie przetwarzanych wartości pobieranych z sensora ruchu oraz fotorezystorą. Znajdują się tu także metody:

- ➢ sprawdzająca poprawność wprowadzonego przez użytkownika przy użyciu klawiatury membranowej pinu,
- ➢ wysyłająca maile o potencjalnym włamaniu po wykryciu anomalii do podanych w konfiguracji urządzenia adresatów,
- ➢ zmieniająca tryb działania urządzenia z bezpiecznego na niebezpieczny lub odwrotnie.

KONFIGURACJA - PLIK */.conf*

Użytkownik przed uruchomieniem urządzenia zapisuje na karcie SD plik */.conf* zgodnie z następującym formatem:

- **secret** - ciąg znaków długości 16, używany do szyfrowania danych na karcie SD, użytkownik powinien go również zapisać w bezpiecznym miejscu poza kartą SD, by w razie ingerencji (również przypadkowej) była możliwość odzyskania zaszyfrowanych danych
- **ssid -** identyfikator sieci WiFi, w której ma funkcjonować urządzenie
- **wifiPasswd** - hasło do powyższej sieci
- **pin** - ciąg cyfr długości minimum 4, służący do inicjalizacji urządzenia oraz do zmiany trybów pracy
- **ftpUsername** - nazwa użytkownika służąca do połączenia się z serwerem FTP
- **ftpPasswd** - hasło użytkownika służące do połączenia się z serwerem FTP
- **email** - email, z którego mają być wysyłane powiadomienia o ingerencji
- **emailPasswd** - hasło do powyższego maila
- **emailToNotify** - email na który mają być wysyłane powiadomienia o ingerencji
- **persistSecret** - znak '**y**' gdy klucz tajny ma być zapisany na karcie SD, lub znak '**n**' gdy ma być umieszczony jedynie w pamięci ulotnej, dla zwiększonego bezpieczeństwa zalecana jest opcja '**n**'
- **obowiązkowa pusta linia**

Przykładowy plik /.conf ma postać:

```
secret=super-secret1234
ssid=myNet
wifiPasswd=qwerty1234
pin=1234
ftpUsername=username
ftpPasswd=password
email=ftpemail@gmail.com
emailPasswd=ftpemailpasswd
emailToNotify=owner@gmail.com
persistSecret=n
```

Jeśli użytkownik nie wybrał opcji persystencji klucza, to zostanie on umieszczony jedynie w pamięci RAM. Użytkownik powinien zapisać go w bezpiecznym miejscu i przy kolejnych uruchomieniach wpisać go do pliku */.conf* na karcie SD. Jeśli wybrana została opcja persystencji klucza, system zapisze go w pamięci Flash i odczyta po kolejnym uruchomieniu, w takim przypadku użytkownik nie musi ponownie zapisywać pliku /.conf.

Jeśli system wykryje, że w pamięci Flash zapisane są wszystkie potrzebne preferencje to przy uruchomieniu poprosi użytkownika o wpisanie kodu PIN. Błędne podanie go pięć razy skutkuje usunięciem aktualnych preferencji. Po poprawnym podaniu zostanie sprawdzone, czy istnieje nowy plik */.conf*. Jeśli tak, preferencje zostaną nadpisane, po czym plik */.conf* zostanie usunięty i nadpisany zerami.

W przypadku braku pewnych konfiguracji w pamięci Flash system sprawdzi czy istnieje na karcie SD plik */.conf*, jeśli nie to wyświetli za pomocą diody stan błędu i nie przejdzie do dalszego działania. Jeśli plik ten zostanie znaleziony, system wczyta preferencje, zapisze w pamięci Flash, analogicznie, wyczyści plik */.conf* po czym rozpocznie działanie.