„EDUSAT – Satelita Edukacyjny”

Maciej Siedlecki

Jakub Zmyślony

Dominik Kołodziej

Spis treści

Wstęp

Kosmos i technologia. Te dwie połączone pragnieniem poznawania rzeczy od zarania ludzkości wywierały na siebie wzajemnie ogromny wpływ. Gdyby nie fascynacja kosmosem, nie powstałby nigdy teleskop, „Mechanizm z Antykithiry", nie rozwinęłaby się nigdy fizyka. Gdyby nie technologia, wszechświat dalej byłby dla nas przestrzenią niedostępną, a zjawiska widoczne chociażby nad naszymi głowami moglibyśmy próbować wyjaśniać w najróżniejsze, jednakże najprawdopodobniej mijające się z prawdą i niemożliwe do dowiedzenia w obecnym stanie rzeczy tezy i spekulacje. To właśnie technologia jest kluczem do wrót kosmosu. To dzięki niej możemy dziś podziwiać piękno nie tylko naszej własnej planety, ale także miejsc w kosmosie tak odległych, że żaden punkt odniesienia znany nam z życia codziennego nie pozwoli pojąć tejże odległości. A jednak podziwiać te miejsca możemy nawet tu i teraz. To właśnie zasługa m.in. Kosmicznego Teleskopu Hubble'a – najbardziej znanego satelity astronomicznego. W dzisiejszych czasach jedną z możliwości poznania świata znajdującego się wysoko ponad nami są właśnie sztuczne satelity. Stąd nasza fascynacja i pomysł na projekt!

Satelity

Jak mówi Wikipedia – to „każde ciało o względnie małej masie, obiegające inne ciało o większej masie". Nas jednak interesuje prowadzenie badań w przestrzeni kosmicznej – a więc w naturalnym następstwie jesteśmy zafascynowani szeroko pojętymi satelitami naukowo-badawczymi. W tym właśnie momencie manifestują się nasze elektroniczno-informatyczne hobby. Chcieliśmy doświadczyć realnych problemów na które napotykać się muszą chociażby inżynierowie NASA. Jednakże pierwszy problem jaki przychodzi do głowy na myśl o budowie satelity to cena. Nie bez powodu przecież państwa i prywatni przedsiębiorcy inwestują w agencje kosmiczne tak ogromne kwoty. Nasz cel jednak nie legł w gruzach. Przeszukując dostępne nam źródła informacji natrafiliśmy na satelity typu CubeSat.

Satelity CubeSat

Jest to rodzaj sztucznych amatorskich nanosatelitów o standaryzowanych rozmiarach. Typowym zastosowaniem CubeSat'ów jest edukacja, stosunkowo tanie sprawdzenie nowych rozwiązań technicznych, obserwacja i badania kosmosu oraz inne badania naukowe. Zazwyczaj do budowy CubeSat'ów używa się dostępnych na rynku, gotowych elementów elektroniki. Charakterystyczną cechą tych satelitów są ich rozmiary. Są to z założenia sześciany, których rozmiar oznaczany jest jednostką U, gdzie 1U odpowiada rozmiarom 10cm x 10cm x 10cm. Specyfikacja CubeSat'ów opracowana została w 1999r. przez „California Polytechnic State University" i Uniwersytet Stanforda, aby promować i rozwijać umiejętności niezbędne do projektowania, produkcji i testowania bardzo małych satelitów przeznaczonych do wyniesienia na niską orbitę okołoziemską.

Nasza praca

Nasza praca zakłada stworzenie projektu programu na zajęcia dydaktyczne z zakresu informatyki, elektroniki i druku 3D, który ma na celu popularyzację astronomii wśród ambitnej młodzieży, rozwój umiejętności i nabywanie wiedzy w wyżej wymienionych kierunkach. Jako motyw przewodni wybraliśmy satelitę, ponieważ jego budowa wydała nam się ciekawa i satysfakcjonująca ze względu na konfrontację z prawdziwymi problemami związanymi z budową satelity oraz finalnie zbudowanie od podstaw przez uczniów amatorskiego satelity typu CubeSat.

Naszym celem jest rozwinięcie uczniów w dziedzinach druku 3D, aby zdobyć umiejętności związane z przygotowywaniem modeli do druku, kalibracji oraz obsługi drukarki. elektroniki, a dokładniej w zakresie interfejsów komunikacji między urządzeniami, a także w zakresie oprogramowywania i pracy z mikroprocesorami. W kontekście informatyki przedstawiamy materiał z zakresu teorii informacji opisując algorytmy wykrywające i korygujące przekłamania zaistniałe w wyniku transmisji danych. Chcemy także dać uczniom podwaliny do projektowania własnych niezawodnych protokołów transmisji danych. Uczniowie będą mogli także zdobyć wiedzę na temat budowy użytecznych interfejsów graficznych wykorzystując nowoczesne technologie. Podsumowaniem zdobytej wiedzy będzie zaś samodzielne wydrukowanie, zbudowanie i oprogramowanie satelity przez uczniów.

W dalszej części dokumentacji prezentujemy opracowany przez nas program wraz z przygotowanymi przez nas materiałami dydaktycznymi dotyczącymi poszczególnych tematów.

Druk 3D

Cele:

- Kalibracja drukarki 3D
    - Poziomowanie stołu
    - Sprawdzenie napięcia pasków
    - Tzw. pid-tuning
    - Kalibracja ekstrudera
- Przygotowanie modeli do druku
    - Zapoznanie się z dostępnym oprogramowaniem typu „slicer” (np. Cura, IdeaMaker, PrusaSlicer)
    - Obsługa programu tnącego
    - Zapoznanie się z parametrami wydruku

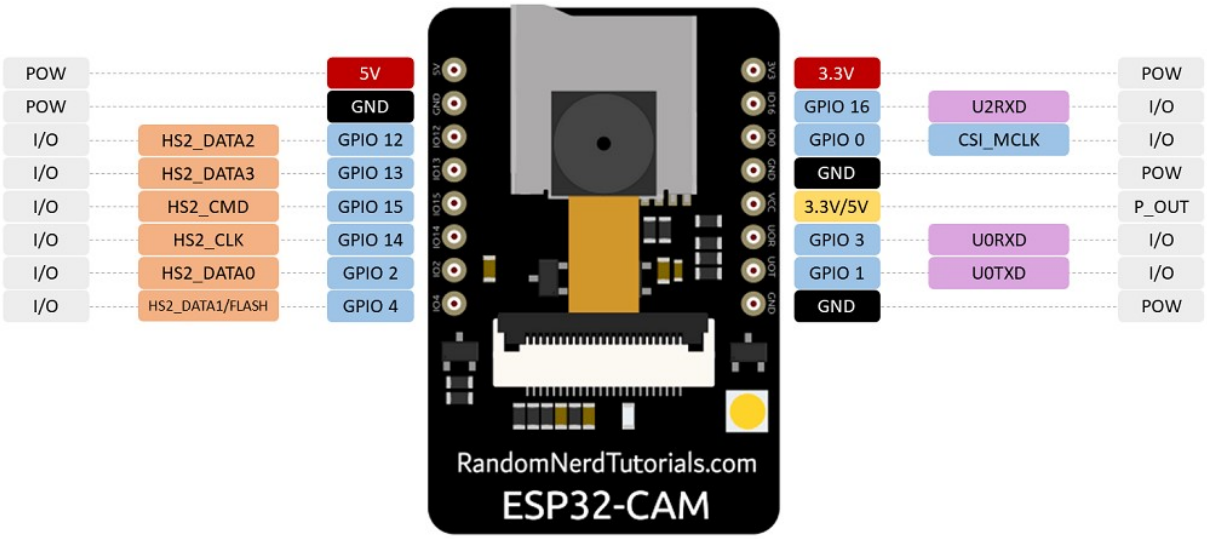
Praca z układami elektronicznymi

- Zapoznanie z zawartymi układami elektronicznymi w jakie wyposażony jest satelita
- Zapoznanie z wyprowadzeniami poszczególnych układów
- Zapoznanie ze sposobami programowania mikrokontrolerów

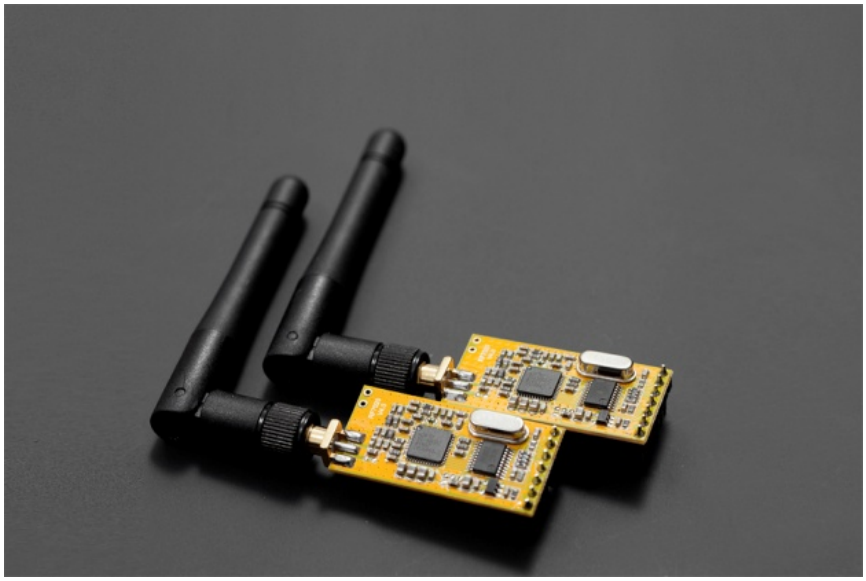
Układy elektroniczne

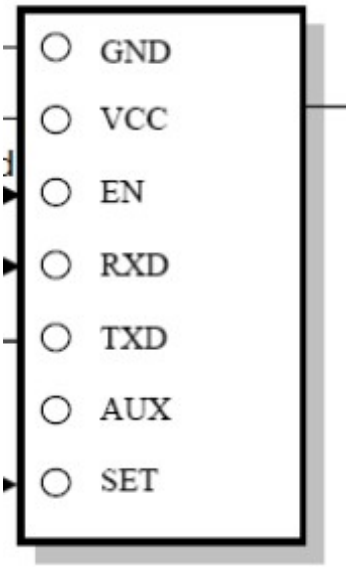
Mikrokontroler ESP32-CAM – układ wyposażony w kamerę. Zaopatrzony jest w interfejs UART, I2C oraz SPI.

Posiada także wiele cyfrowych pinów wejścia i wyjścia.



APC220 - radio do komunikacji zdalnej z satelitą. Do komunikacji wykorzystuje interfejs UART.





Wyprowadzenia czujnika:

VCC - czyli dodatnie napięcie zasilania, potocznie nazywany plusem (3.3 ~ 5.5V)

GND - czyli masa, potocznie nazywana minusem

EN – pin służący do wyłączenia układu

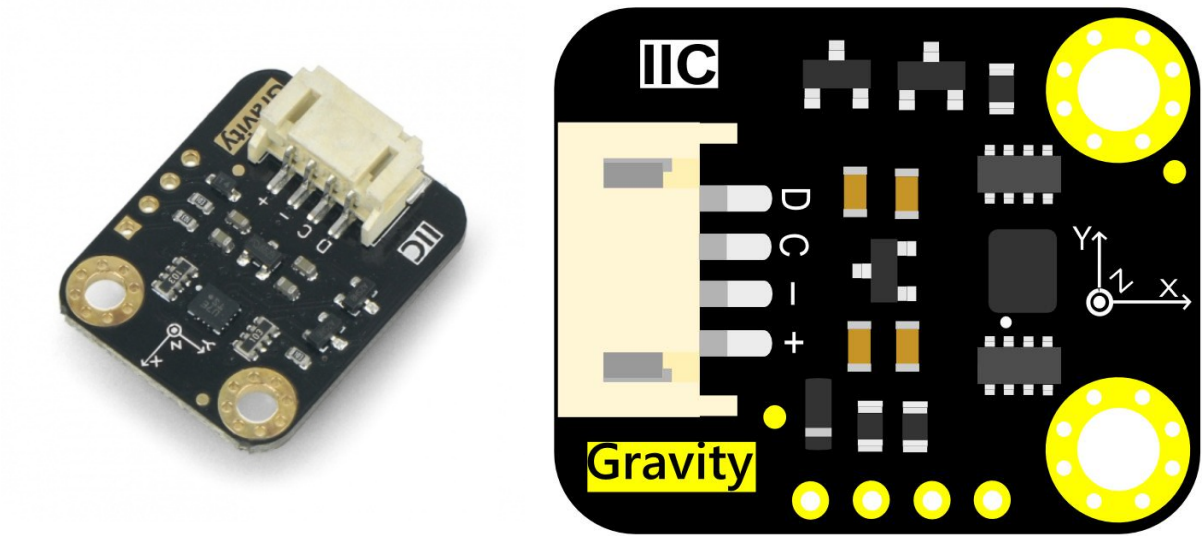
SET – przestawienie radia w tryb ustawiania parametrów

UART:

RXD – RX, odbiornik

TXD – TX, nadajnik

Czujnik ruchu wyposażony w 3-osiowy akcelerometr i 3-osiowy żyroskop od firmy DFRobot. Do komunikacji wykorzystuje magistralę I2C.



Wyprowadzenia czujnika:

+ - oznacza VCC, czyli dodatnie napięcie zasilania, potocznie nazywany plusem (3.2 ~ 6V)

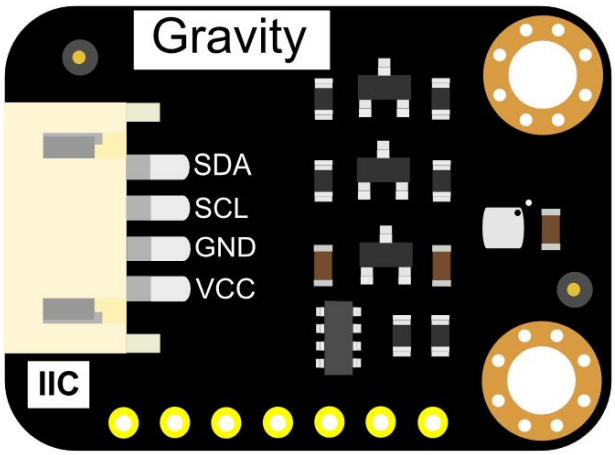
- - oznacza GND, czyli masę, potocznie nazywana minusem

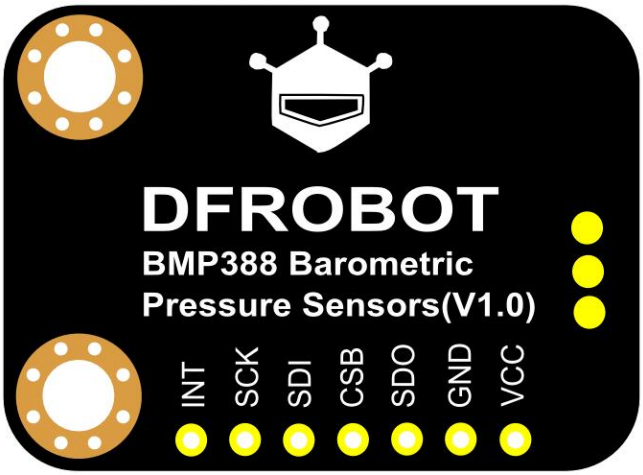
I2C:

D – oznacza SDA, jest to linia odpowiadająca za transmisję danych.

C – oznacza SCL, jest to linia zegarowa.

Cyfrowy barometr, czujnik ciśnienia, wysokości i temperatury od firmy DFRobot. Czujnik ten korzysta z magistrali I2C i SPI.





Wyprowadzenia czujnika:

VCC - czyli dodatnie napięcie zasilania, potocznie nazywany plusem (3.3 ~ 5.5V)

GND - czyli masa, potocznie nazywana minusem

I2C:

SDA – linia transmisji danych

SCL – linia zegarowa

SPI:

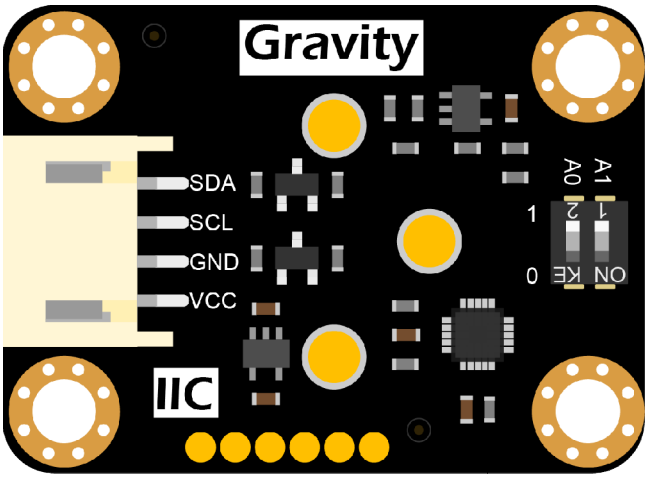
SCK – CLK, lina zegarowa.

SDI (Serial Data In) –MOSI, czyli linia danych wychodzących z mikrokontrolera a przychodzących do czujnika.

SDO (Serial Data Out) – MISO, czyli linia danych wychodzących z czujnika a przychodzących do mikrokontrolera.

CSB – CS/SS (Chip/Slave Select) – linia do aktywacji wybranego czujnika

Czujnik tlenu od firmy DFRobot. Czujnik ten korzysta z magistrali I2C.



Wyprowadzenia czujnika:

VCC - dodatnie napięcie zasilania, potocznie nazywany plusem (3.3 ~ 5V)

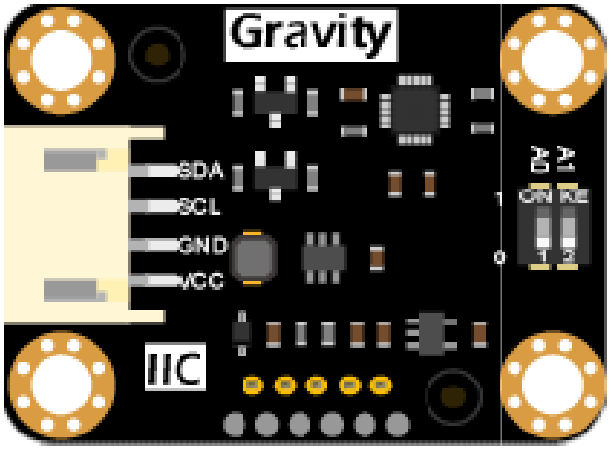
GND - masa, potocznie nazywana minusem

I2C:

SDA – jest to linia odpowiadająca za transmisję danych.

SCL – jest to linia zegarowa.

Elektrochemiczny czujnik ozonu od firmy DFRobot. Czujnik ten korzysta z magistrali I2C.



Wyprowadzenia czujnika:

VCC - dodatnie napięcie zasilania, potocznie nazywany plusem (3.3 ~ 5V)

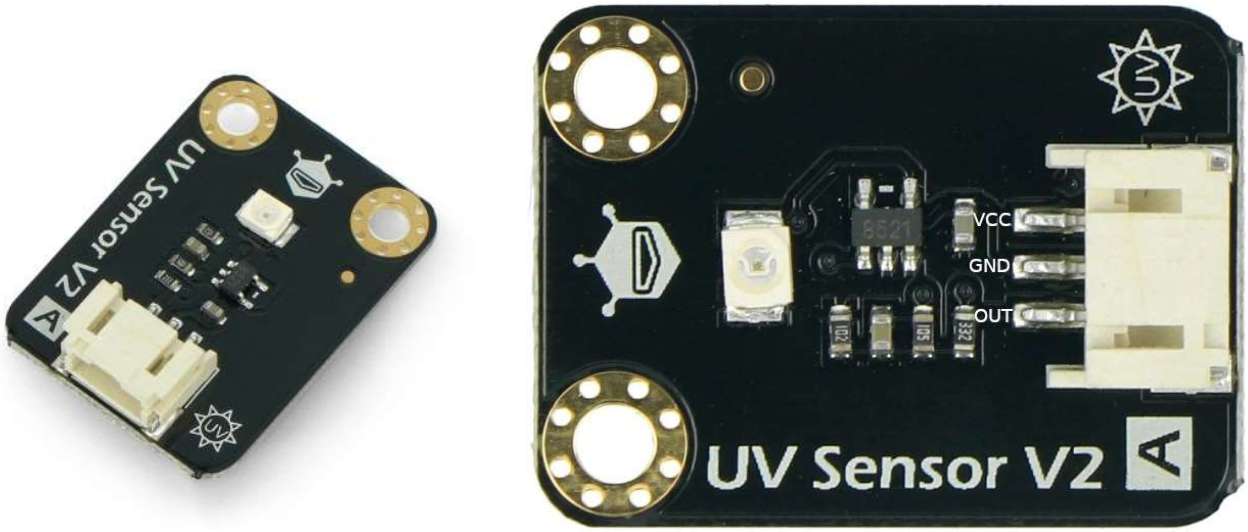
GND - masa, potocznie nazywana minusem

I2C:

SDA – jest to linia odpowiadająca za transmisję danych.

SCL – jest to linia zegarowa.

Czujnik światła ultrafioletowego od firmy DFRobot



Wyprowadzenia:

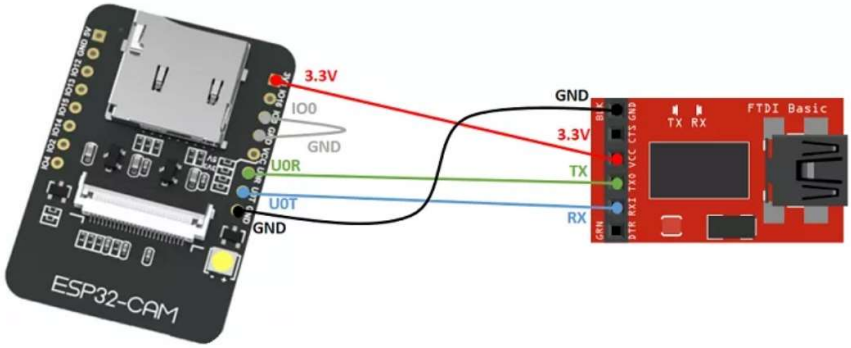
VCC - dodatnie napięcie zasilania, potocznie nazywany plusem (3.3 ~ 5V)

GND - masa, potocznie nazywana minusem

OUT – analogowe wyjście czujnika

Programowanie mikrokontrolerów

Jako przykład niech posłuży mikrokontroler ESP32-CAM i programator FTDI 232, są to urządzenia które sami wykorzystaliśmy przy budowanie naszego satelity. Programowanie odbywa się przez interfejs UART. Podłączenie tych dwóch urządzeń wygląda następująco. Pin IO0 zwarty do masy powoduje że EPS32 po zresetowaniu przejdzie w tryb programowania. Po przygotowywaniu urządzeń i podłączeniu naszego programatora do komputera, musimy skorzystać z odpowiedniego oprogramowania np. Arduino IDE, lub jak w naszym przypadku PlatformIO. Proces kompilowania i wgrywania programu wygląda trochę inaczej zależnie od wybranego programu, jednak w każdym z nich jest prosty i przejrzysty.



Obsługa interfejsów komunikacji

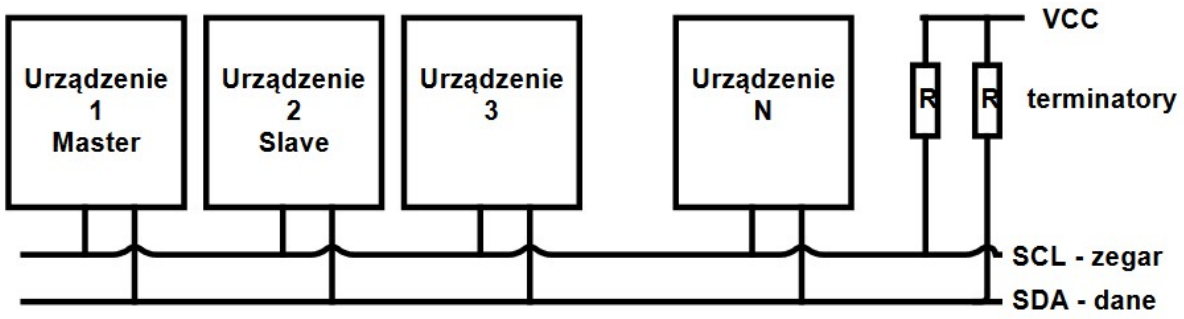
- Zapoznanie z interfejsami dostępnymi w mikrokontrolerze
- Podstawowa obsługa tych interfejsów
- Poznanie budowy i zasady działania

I2C (Inter Integrated Circuit)

Magistrala wprowadzona przez firmę Philips i przeznaczona do komunikacji wewnątrz urządzeń elektronicznych. Typowo stosowana do łączenia mikrokontrolera z takimi urządzeniami jak wyświetlacze, pamięci EEPROM, zegary RTC, przetworniki AD i DA, klawiatury. Jest to magistrala szeregowa.

Najważniejsze własności magistrali I2C:

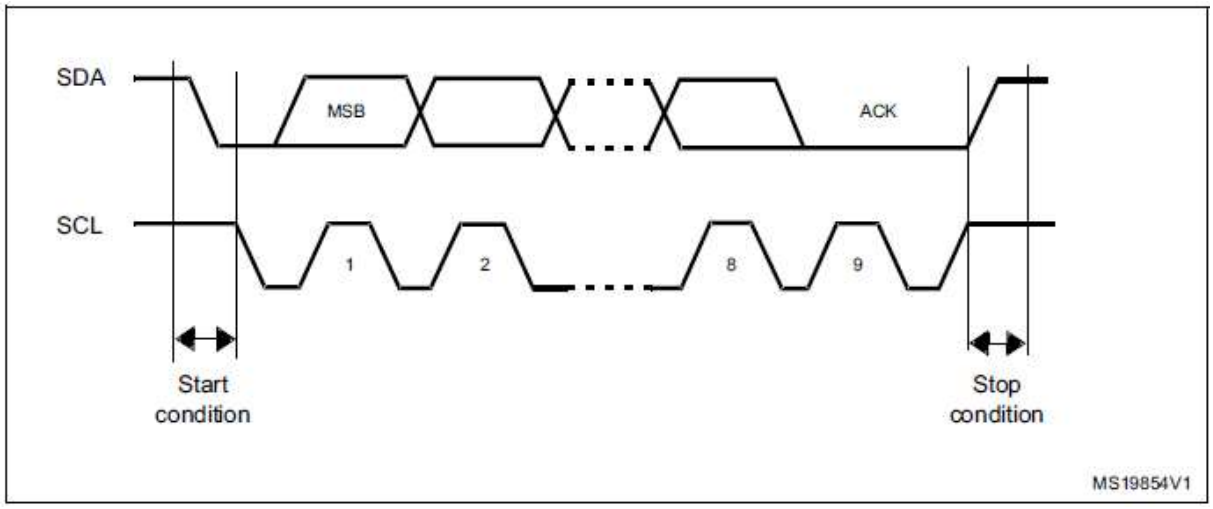
- Inicjować komunikację może tylko urządzenie nadrzędne (master), reszta urządzeń (slave) dostosowują się do tego co dyktuje urządzenie nadrzędne.
- Magistrala 2 przewodowa w skład której wchodzą sygnały: dane – SDA – (serial data line), zegar - SCL – (serial Clock)
- Każdy dołączony do magistrali układ ma unikalny adres. Adresowanie 7 bitowe (128 urządzeń) lub 10 bitowe (1024 urządzeń). Rodzaj transmisji master – slave
- Magistrala jest szeregowa, dwukierunkowa, zorientowana na transmisje 8 bitowe.
- Zasięg do kilku metrów



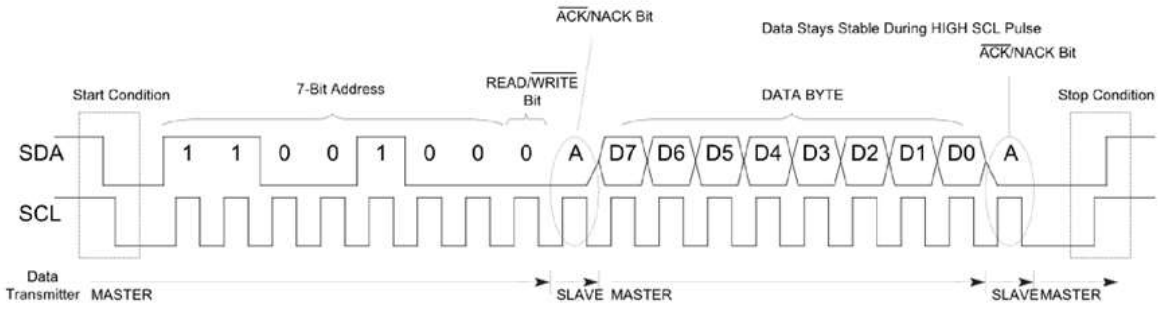
Użyty jest standard logiki dodatniej a więc wysokiemu poziomowi napięcia odpowiada logiczna 1 a niskiemu logiczne 0. Magistrala podłączona jest do napięcia +5V poprzez rezystor. Linie te są typu otwarty kolektor ze słabą 1 i mocnym 0. Znaczy to że domyślnie na linii danych zawsze znajduje się stan wysoki, zaś gdy jeden z układów wystawi jedynkę, a drugi 0 na magistrali będzie 0.

Transfer danych:

W magistrali I2C podczas transmisji danych panuje ważna zasada mówiąca że podczas gdy na linii SCL panuje stan wysoki to stan na linii SDA musi być stabilny, nie może się zmienić. Jest tak ponieważ gdyby taka zmiana zaszła zostałaby ona zinterpretowana jako specjalny sygnał sterujący.



Tymi sygnałami sterującymi są „Start" i „Stop", rozpoczynające i kończące transmisję. Sygnał startu nadawany jest wtedy kiedy na linii SCL panuje stan wysoki, a na linii SDA stan zmienia się z wysokiego na niski. Kiedy zaś kończy się transmisja sygnał stopu nadawany jest przez zmianę stanu na linii SDA z niskiego na wysoki, także w momencie kiedy panuje stan wysoki na linii SCL.



Zmiana stanu logicznego, a zarazem zmiana przesyłanego bitu, może zachodzić tylko wtedy kiedy na linii SCL panuje stan niski, zaś odczytanie tego bitu przez urządzenie docelowe, zajdzie w momencie zmiany z stanu niskiego na wysoki na linii zegarowej. Oznacza to że w każdym takcie zegara przesyłany jest jeden bit. Dane przesyłane są w ośmiu bitowych grupach (bajtach) od najbardziej znaczącego bitu.

Wiedząc to przejdźmy do omówienia całej transmisji. Zaczyna się ona od sygnału start. Następnie urządzenie master wysyła 7 lub 10 bitów (w zależności od wersji magistrali) adresu urządzenia docelowego, potem wysyłany jest kolejny bit oznaczający czy będzie przeprowadzany odczyt czy zapis. Po tym master zwalnia magistralę i nasłuchuje bitu od urządzenia docelowego potwierdzającego otrzymanie danych sygnałem ACK (wystawia 0 na linii danych). Teraz rozpoczyna się transmisja danych, w segmentach po osiem bitów (od najbardziej istotnego). Każdy segment jest potwierdzony sygnałem ACK przez odbiorcę. Transmisja kończy się sygnałem stop.

SPI (Serial Peripheral Interface)

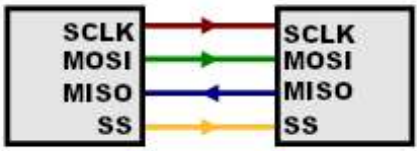
Szeregowy interfejs urządzeń peryferyjnych. Jeden z najczęściej używanych interfejsów komunikacyjnych pomiędzy systemami mikroprocesorowymi a układami peryferyjnymi.

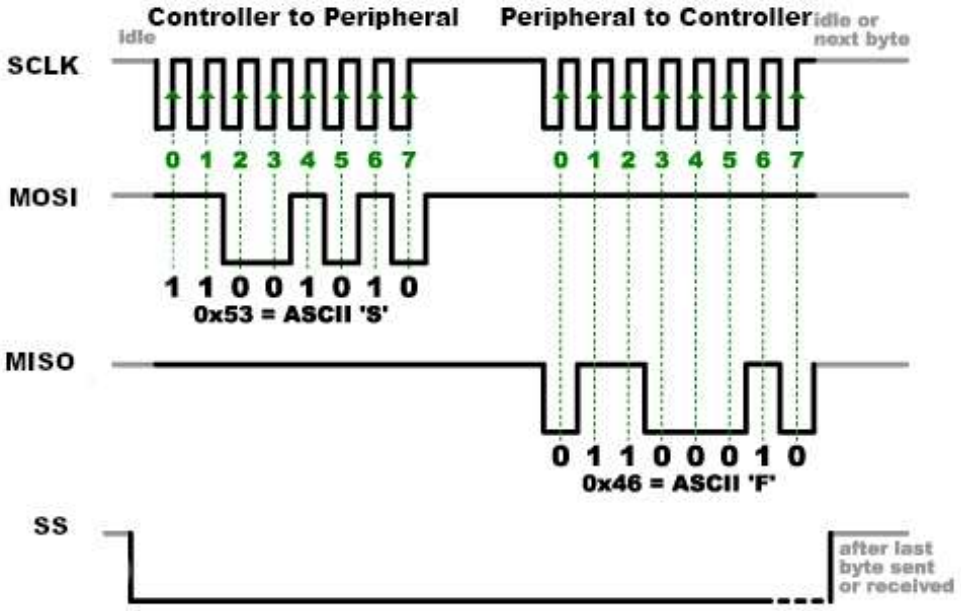
Komunikacja poprzez SPI odbywa się synchronicznie za pomocą 4 linii:

- MOSI (master output slave input) – dane dla układu peryferyjnego,
- MISO (master input slave output) – dane z układu peryferyjnego,
- SCLK (serial clock) – sygnał zegarowy (taktujący).
- SS (slave select) / CS (chip select) – linia służąca do aktywacji układu peryferyjnego (stan niski oznacza aktywację urządzenia)

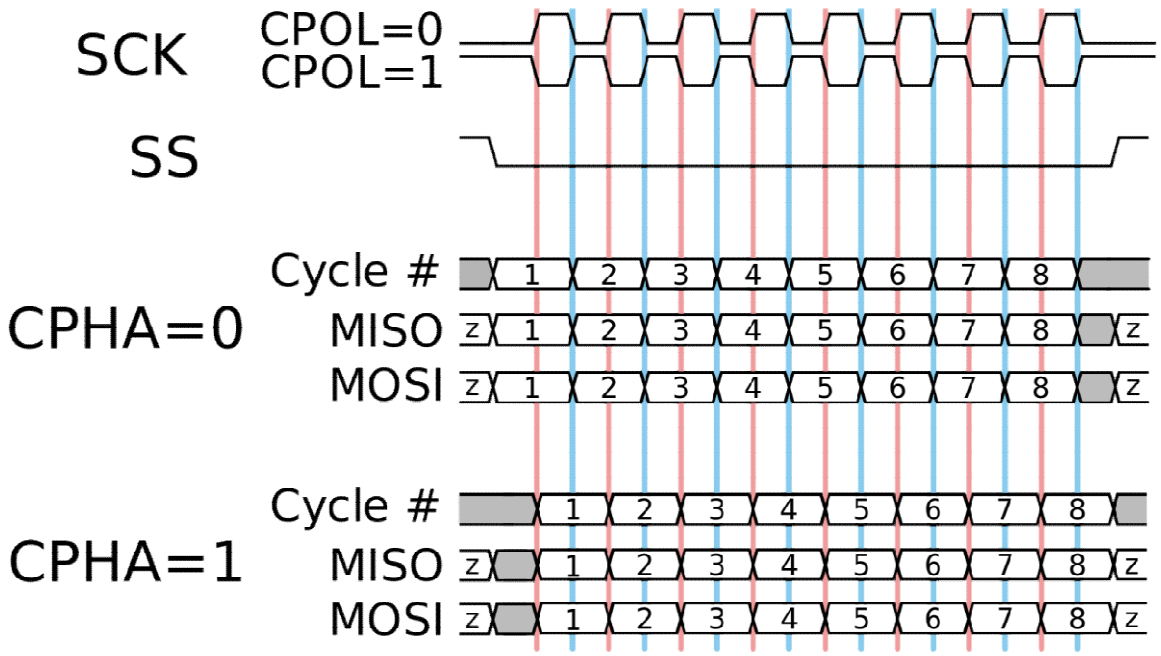
W magistrali SPI, podobnie jak w I2C, urządzenia nie są na tym samym poziome. W tym interfejsie także występuje podział na master i slave. Oznacza to że np. mikrokontroler użyty jako urządzenie nadrzędne dyktuje warunki komunikacji oraz nią zarządza.

W SPI oprócz częstotliwości zegara ważnym parametrem pracy jest moment odczytu i zmiany danych wystawionych na magistralę. Sposób działania zostanie opisany na przykładzie CPOL=1 CPHA=1.





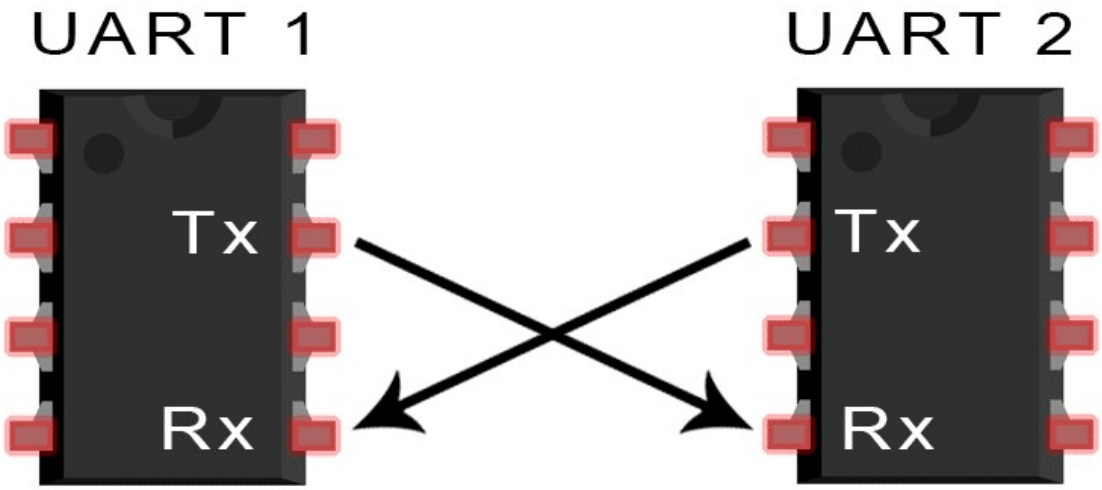
Master kontroluje linie SCLK, MOSI i SS. Pierwsza rzecz która rozpoczyna komunikację to zmiana stanu linii SS na niski. Od teraz na każde wzrastające zbocze na linii zegarowej będzie odczytywany bit po stronie urządzenia peryferyjnego jak i po stronie mikrokontrolera. Zaś na zbocze opadające nastąpi zmiana danych na kolejne. Komunikacja zostaje zatrzymana przez ponowne zmienienie stanu linii SS, tym razem na stan wysoki, dezaktywując urządzenie. Dzięki takiej konstrukcji na magistrali SPI, dane mogą być przekazywane jednocześnie w obu kierunkach.



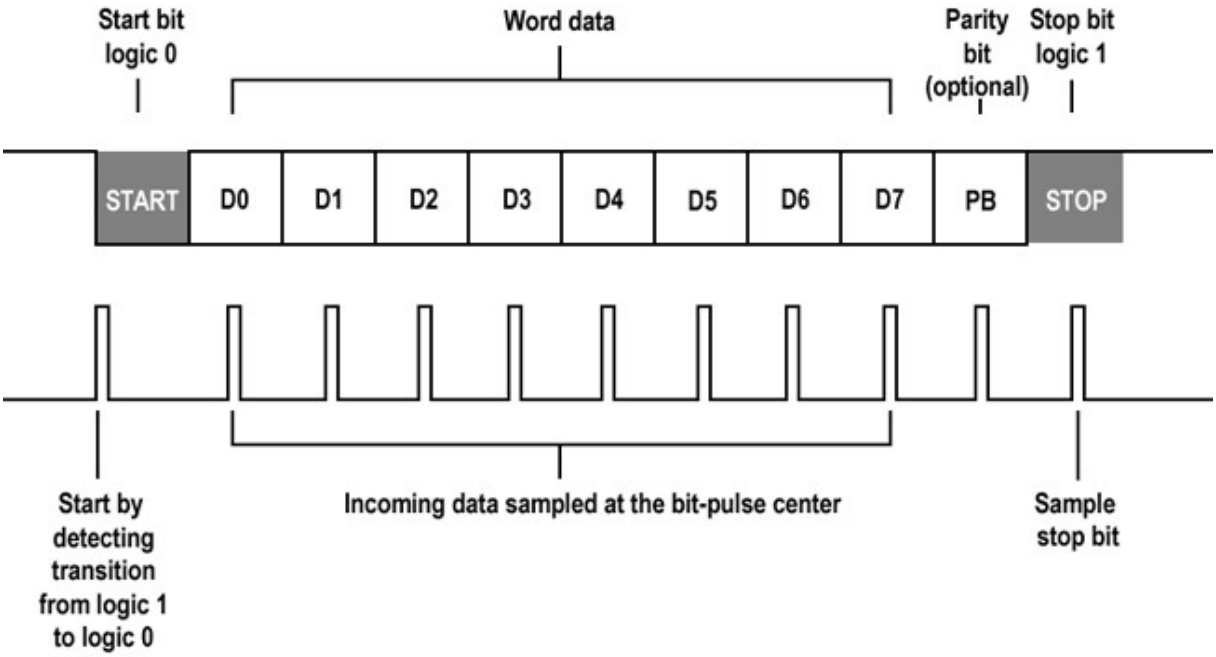
Wiedząc jak działa, możemy przejść do omówienia czym jest CPOL i CPHA. CPOL określa nam czy zegar w stanie spoczynku będzie w stanie wysokim czy niskim. W stanie niskim będzie kiedy CPOL=0, a w wysokim kiedy CPOL=1. CPHA określa nam czym dane odczytywane są na „lewym" zboczu, czy na „prawym", analogicznie zmieni się także moment zmiany danych na linii. Jeżeli CPHA=0 to dane zostaną odczytane na „lewym" zboczu, a zmienią się na „prawym".

UART (universal asynchronous receiver-transmitter)

Jest to asynchroniczny szeregowy protokół. Oznacza to, że w przeciwieństwie do powyższych protokołów ten nie wykorzystuje linii zegarowej do synchronizacji przesyłanych bitów. UART posiada dwie linie TX (nadajnik) i RX (odbiornik). Łączymy je ze sobą na krzyż to znaczy TX – RX, ponieważ jedno urządzenie nadaje TX – nadajnik, a drugie musi je odebrać RX – odbiornik.



Przed przystąpieniem do jakiejkolwiek transmisji oba urządzenia muszą posiadać takie same ustawienia, między innymi takie jak szybkość transmisji, czy długość słowa, ponieważ urządzenie nie ma odnośnika aby mogło samo na bieżąco dostosowywać parametry pracy.



W przypadku powyżej długość słowa to 8 bitów, bit parzystości (parity bit) jest opcjonalny, więc dla uproszczenia na razie zostanie pominięty. Transmisja rozpoczyna się bitem startu jest on logicznym 0, ponieważ linia transmisji danych w czasie spoczynku jest w stanie wysokim, powoduje to że odbiorcy jest łatwo określić start transmisji. Następnie przekazywane jest 8 bitów (zgodnie z ustawieniami długości słowa). Odbiorca wie jak odczytać bity, ponieważ ma z góry ustaloną prędkość przesyłu, czyli licząc czas od punktu startu jesteśmy w stanie określić kiedy otrzymamy kolejny bit do odczytania. Odczytywane one są pośrodku „okienka czasowego" w którym pojawił się aktualny bit aby zniwelować możliwość sczytania nieodpowiedniego stanu, np. kiedy bit był jeszcze w trakcie zmiany z jednego stanu na drugi. Na samym końcu mamy znak stopu oznaczony stanem wysokim, czyli powrót linii do stanu spoczynkowego.

Odczytywanie pomiarów z czujników

Przykładowy kod obsługujący jeden z czujników:

Funkcja inicjalizująca czujnik ciśnienia i funkcja odczytująca dane z niego przy pomocy biblioteki wykorzystującej interfejs I2C do komunikacji z czujnikiem, przy użyciu mikrokontrolera ESP32-CAM.

```cpp
#include "Adafruit_BMP3XX.h"

Adafruit_BMP3XX bmp;


void initBarometer() {
  if (!bmp.begin_I2C()) {
    Serial.println("Could not find a valid BMP3 sensor, check wiring!");
    return;
  }

  //Inicjalizowanie nadpróbkowania i filtrowania
  bmp.setTemperatureOversampling(BMP3_OVERSAMPLING_8X);
  bmp.setPressureOversampling(BMP3_OVERSAMPLING_4X);
  bmp.setIIRFilterCoeff(BMP3_IIR_FILTER_COEFF_3);
  bmp.setOutputDataRate(BMP3_ODR_50_HZ);

}

void readBarometerData() {
  //Próba odczytu danych z czujnika
  if(!bmp.performReading()) {
    Serial.println("Failed to perform pressure read :(");
    return;
  }

  //Obliczenie ciśnienia
  float pressure = bmp.pressure / 100;

  //Obliczenie wysokości względem ciśnienia na poziomie morza
  float altitude = bmp.readAltitude(1013.25);

  //Pobranie temperatury
  float temperature = bmp.temperature;

  //Wysłanie danych przez UART
  Serial.println(pressure);
  Serial.println(altitude);
  Serial.println(temperature);
}
```

Zad 1. Wybierz czujnik i odnajdź do niego bibliotekę oraz zaimplementuj prosty odczyt danych korzystając z przykładów w bibliotece.

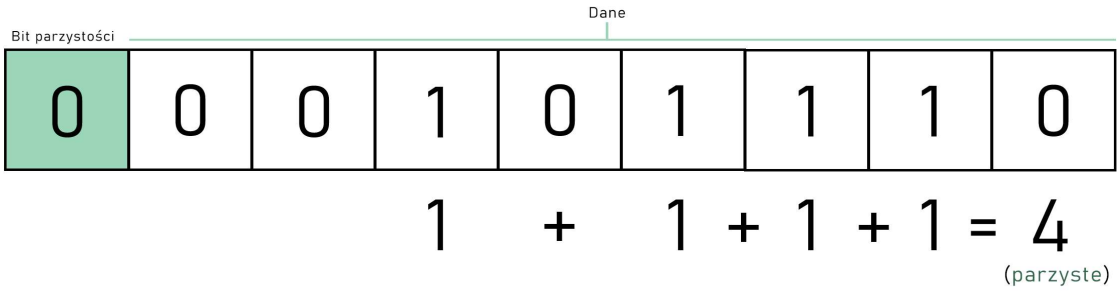
Weryfikacja integralności przesyłanych danych:

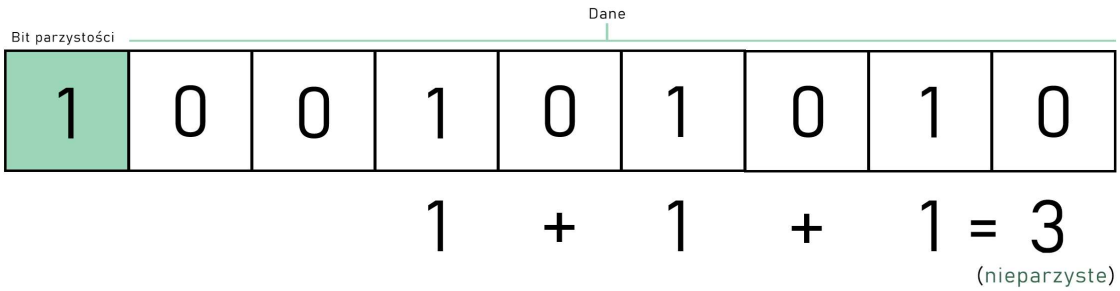
Kontrola parzystości

Kontrola parzystości to jedna z najprostszych metod wykrywania przekłamań w transmitowanych wiadomościach. Polega na dodawaniu do wysyłanej wiadomości bitu kontrolnego.

W przypadku, gdy bit kontrolny jest bitem parzystości, przyjmuje on wartość 1, gdy całkowita liczba jedynek w binarnej reprezentacji danych do wysłania jest nieparzysta (jedynka jest dokładana, aby całkowita ilość jedynek włącznie z bitem kontrolnym była parzysta) i 0 kiedy jest parzysta.

W przypadku bitu nieparzystości, wartości mają odwrotne znaczenie: 1 – ilość jedynek jest parzysta, 0 – jeśli jest nieparzysta.





Przykładowe zadania (opcjonalne):

KontrolaParzystości.1. Oblicz wartość bitu parzystości dla ramki danych 101100110 (reprezentacja binarna).

KontrolaParzystości.2. Sprawdź, czy w ramce danych 101100110110110 wystąpiło przekłamanie bitu, jeśli pierwszy bit to bit parzystości.

KontrolaParzystości.3. Zaimplementuj kontrolę parzystości (detekcję przekłamania danych) w wybranym języku programowania.

Suma kontrolna

Suma kontrolna (ang. checksum) to liczba uzyskana według specjalnego algorytmu służąca do weryfikacji tego, czy wszystkie dane zostały prawidłowo dostarczone do odbiorcy.

Zasada działania:

Komputer wysyłający dane oblicza ich sumę kontrolną i dołącza ją do pakietu danych.

Komputer odbierający dane oblicza w taki sam sposób ich sumę kontrolną i porównuje z tą dołączoną do pakietu danych.

Obie sumy kontrolne muszą być identyczne.

Przykładowe algorytmy obliczania sumy kontrolnej: CRC, MD5, SHA

Algorytm CRC

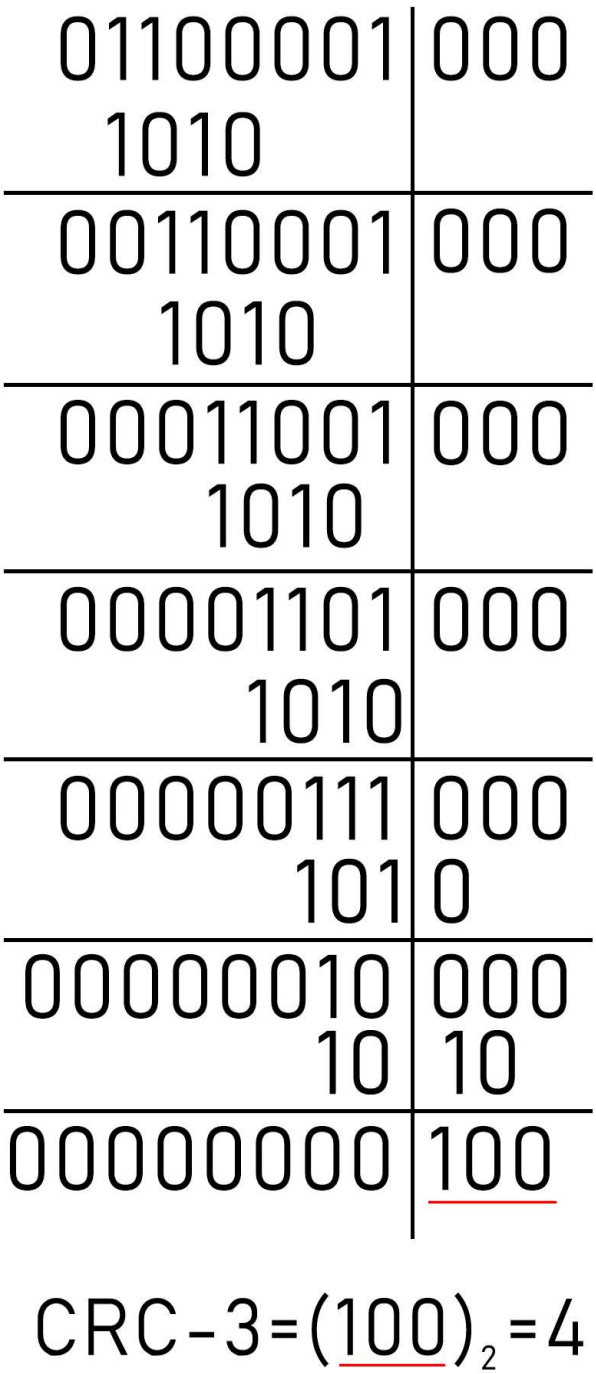
CRC (ang. Cyclic Redundancy Code/Check – Cykliczny Kod Nadmiarowy)

n-bitowy cykliczny kod nadmiarowy (n-bitowy CRC) definiuje się jako resztę z dzielenia ciągu danych przez n+1 bitowy dzielnik (liczba) CRC (zwany wielomianem CRC)

Algorytm postępowania w celu obliczenia n-bitowego CRC jest następujący:

1. do ciągu danych dodaje się n wyzerowanych bitów,
2. w linii poniżej wpisuje się (n+1)-bitowy dzielnik CRC,
3. jeżeli nad najstarszą pozycją dzielnika jest wartość 0, to przesuwa się dzielnik w prawo o jedną pozycję, aż do napotkania 1,
4. wykonuje się operację XOR pomiędzy bitami dzielnika i odpowiednimi bitami ciągu danych, uwzględniając dopisane n bitów
5. wynik zapisuje się w nowej linii – poniżej,
6. jeżeli liczba bitów danych jest większa lub równa n+1, przechodzi się do kroku 2,
7. n najmłodszych bitów stanowi szukane CRC, czyli cykliczny kod nadmiarowy:

Przykład 3-bitowego CRC dla dzielnika postaci 1010 (binarnie) i dla danych postaci 01100001 (binarnie):



Przykładowe zadania (opcjonalne):

CRC.1. Oblicz CRC-3 dla dzielnika 1011 (binarnie) i dla danych wejściowych 10110110.

CRC.2. Oblicz CRC-5 dla dzielnika 101101 (binarnie) i dla danych wejściowych 10110110.

CRC.3. Zaimplementuj algorytm CRC-n w wybranym języku programowania.

Kodowanie korekcyjne

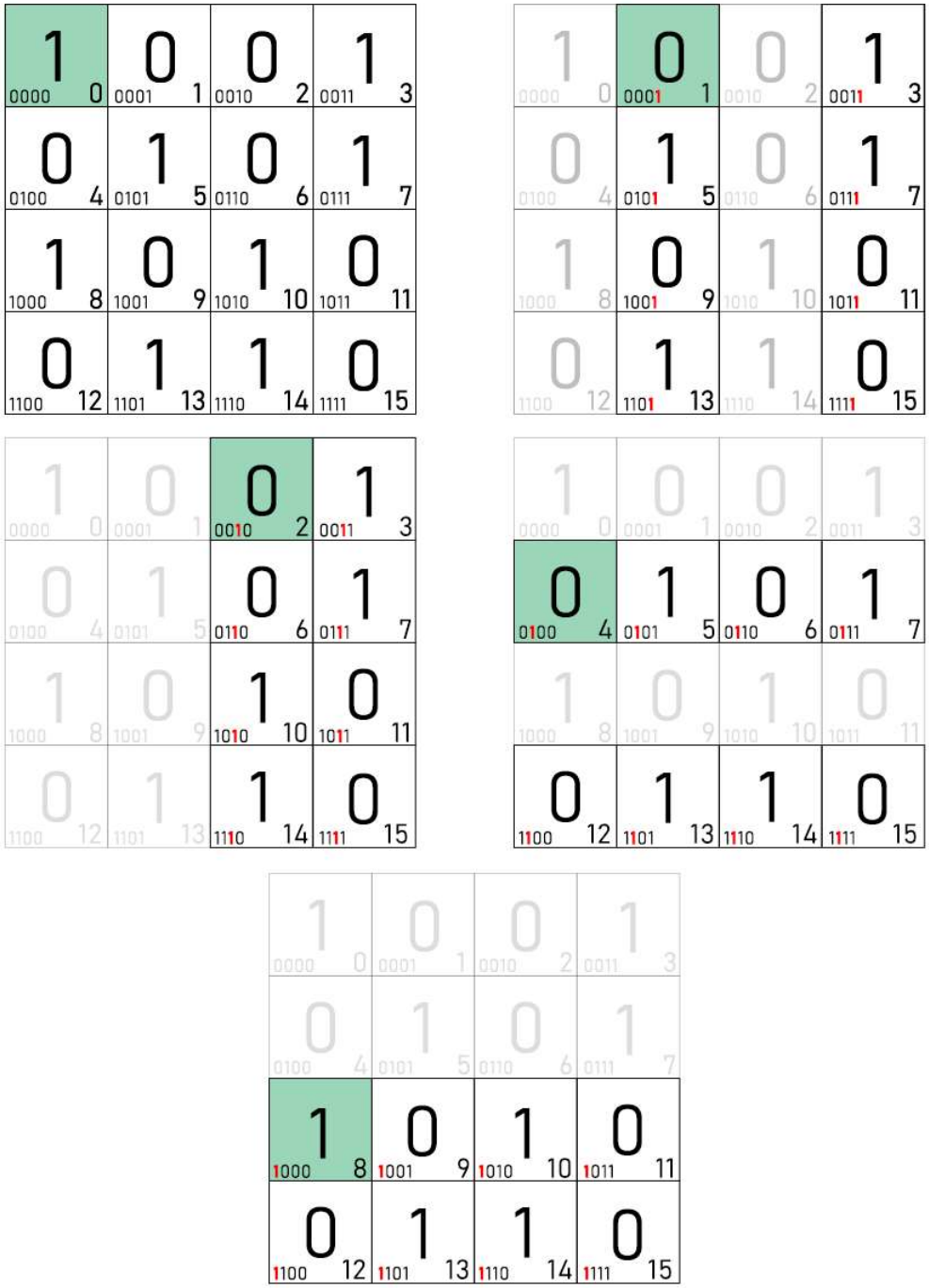
Kodowanie korekcyjne (ang. ECC – error correction coding, FEC – forward error correction) to technika dodawania nadmiarowych danych do transmitowanych informacji po to, aby wykryć i naprawić błędy powstałe w wyniku zakłóceń.

Kod Hamminga – wykrywa i koryguje błędy polegające na przekłamaniu jednego bitu.

Algorytm kodowania:

1. Na pozycjach będących kolejnymi potęgami 2 (pozycja 1, 2, 4, 8…) zapisujemy bit parzystości wartości danych na pozycjach które w swoim zapisie binarnym zawierają daną potęgę dwójki (patrz Rys.Hamming.1.).
2. Na pozycji 0 zapisujemy bit parzystości wszystkich danych.

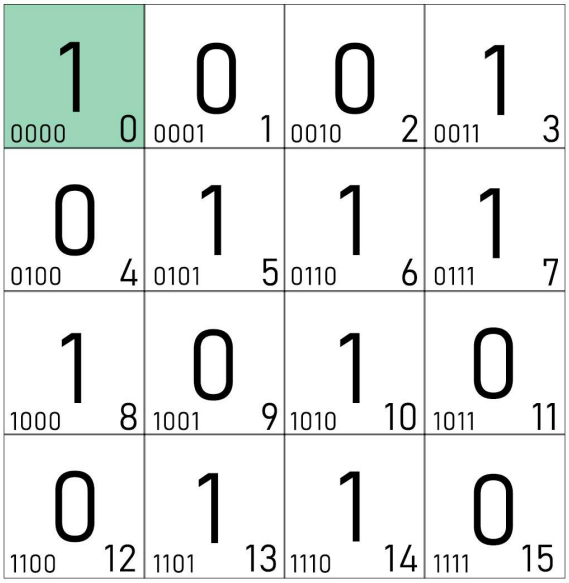
(Rys.Hamming.Legenda)



(Rys.Hamming.1.)

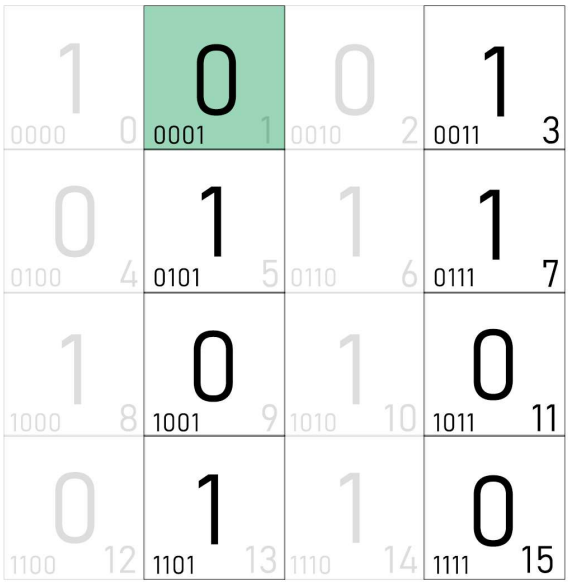
Wykrywanie i korekcja błędów:

1. Sprawdzamy poprawność bitu parzystości (bit na pozycji 0)
2. Tworzymy zmienną pomocniczą o długości (w bitach) równej ilości bitów parzystości nie uwzględniając bitu parzystości na pozycji 0
3. Sprawdzamy poprawność poszczególnych bitów parzystości na pozycjach będących potęgami liczby 2
4. Jeśli wykryjemy błąd w którymś z bitów parzystości, to wiemy, że w grupie którą on obejmuje musi występować błąd. Uzupełniamy więc bit któremu odpowiada pozycja bitu parzystości liczbą 1 w naszej zmiennej pomocniczej. Jeśli błąd nie występuje uzupełniamy ją liczbą 0.
5. Jeśli wartość naszej zmiennej pomocniczej nie jest równa 0, to właśnie bit na tej pozycji należy odwrócić, aby naprawić cały pakiet danych. Jeśli wynosi 0, to błąd nie wystąpił
   (lub to bit na pozycji 0 został odwrócony).



0+0+1+0+1+1+1+1+0+1+0+0+1+1+0=8
(parzysta)

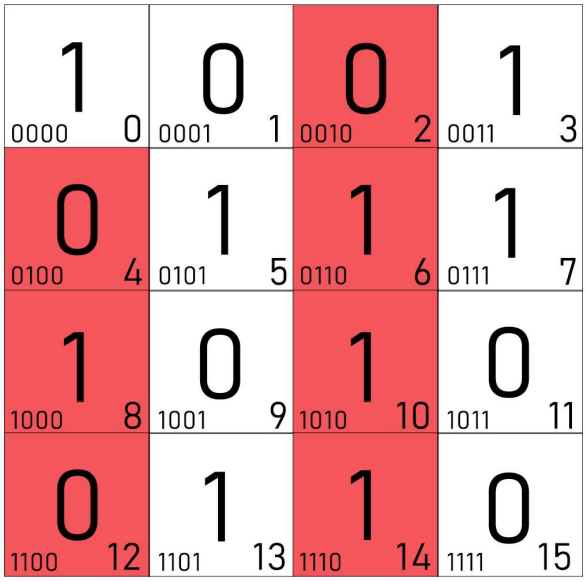
Bit parzystości = 1
Wniosek: Błąd w paczce danych



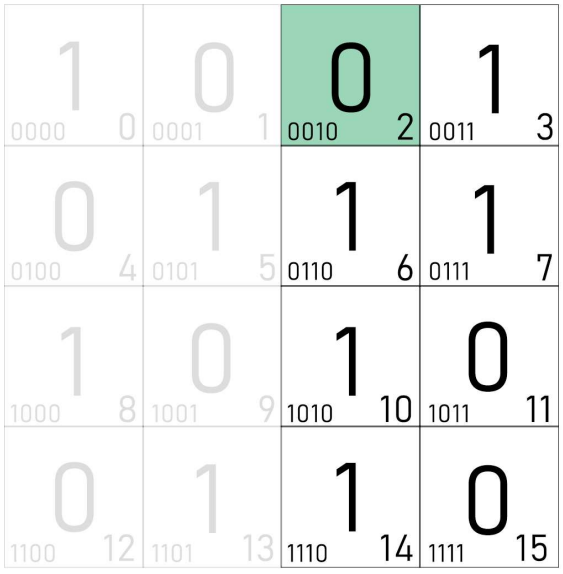
1+1+1+0+0+1+0=4
(parzysta)

Bit parzystości = 0
Wniosek: Grupa, którą obejmuje ten bit parzystości nie zawiera błędu

Bit zawierający błąd musi wyglądać następująco: xxx0 (wykluczając bit na pozycji 0), gdzie: x = 0 lub x = 1



Obecny stan wiedzy: błąd w jednym z zamalowanych na czerwono bitów

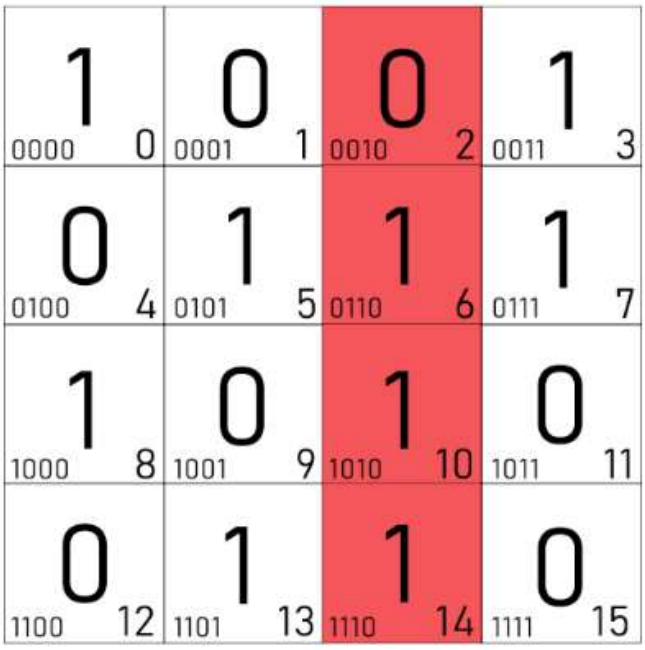


1+1+1+1+0+1+0=5
(nieparzysta)

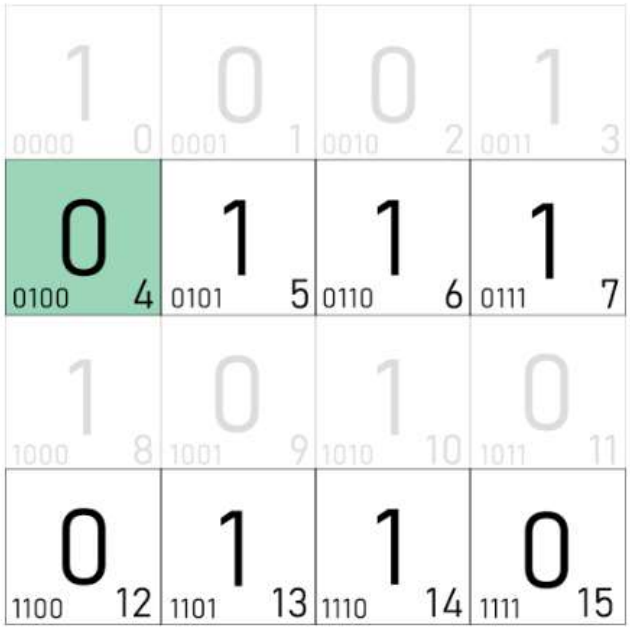
Bit parzystości = 0

Wniosek: Grupa, którą obejmuje ten bit parzystości zawiera błąd

Bit zawierający błąd musi wyglądać następująco: xx10,
gdzie: x = 0 lub x = 1



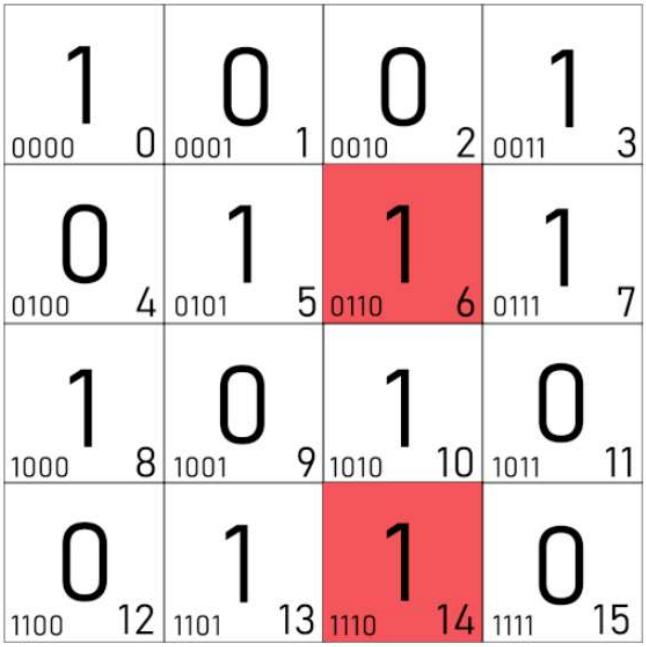
Obecny stan wiedzy: błąd w jednym z zamalowanych na czerwono bitów



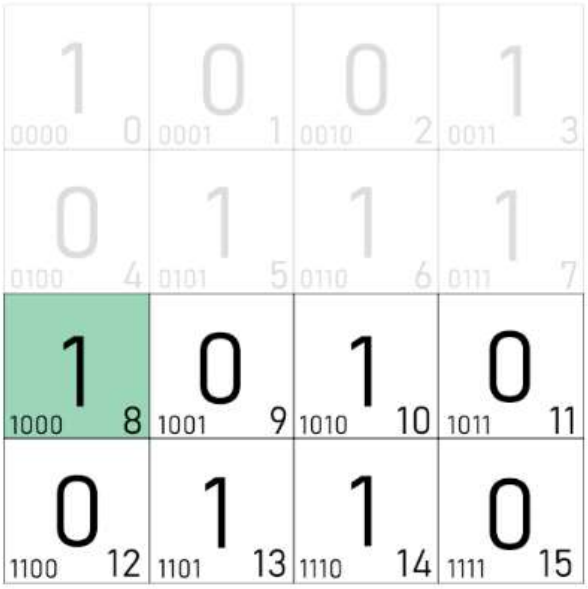
1+1+1+0+1+1+0=5
(nieparzysta)

Bit parzystości = 0
Wniosek: Grupa, którą obejmuje ten bit parzystości zawiera błąd

Bit zawierający błąd musi wyglądać następująco: x110,
gdzie: x = 0 lub x = 1



Obecny stan wiedzy: błąd w jednym z zamalowanych na czerwono bitów



0+1+0+0+1+1+0=3
(nieparzysta)

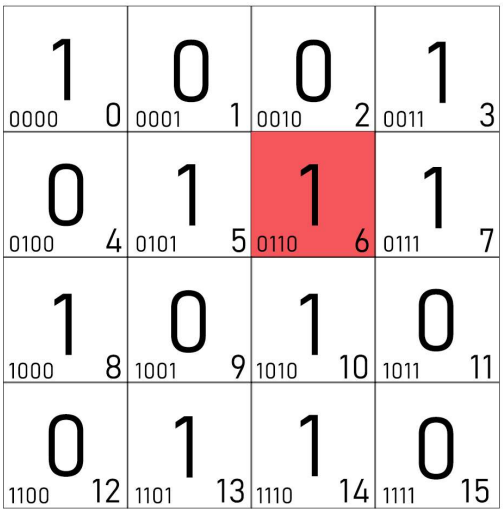
Bit parzystości = 1

Wniosek: Grupa, którą obejmuje ten bit parzystości nie zawiera błędu

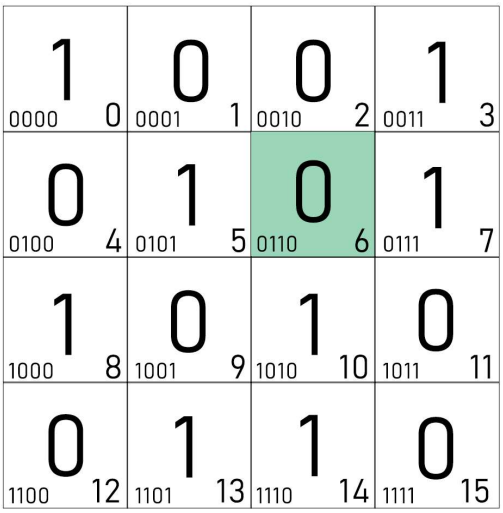
Bit zawierający błąd musi wyglądać następująco: 0110

A więc błąd jest na pozycji 0110 = 6

Przed korekcją

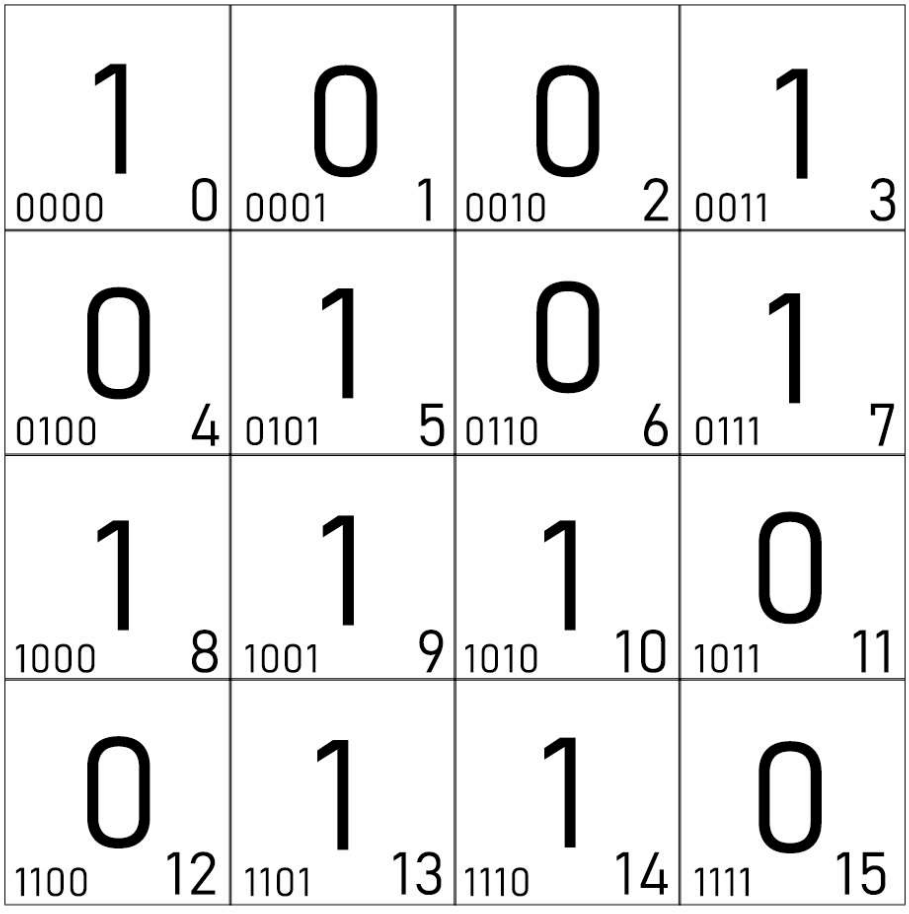


Po korekcji

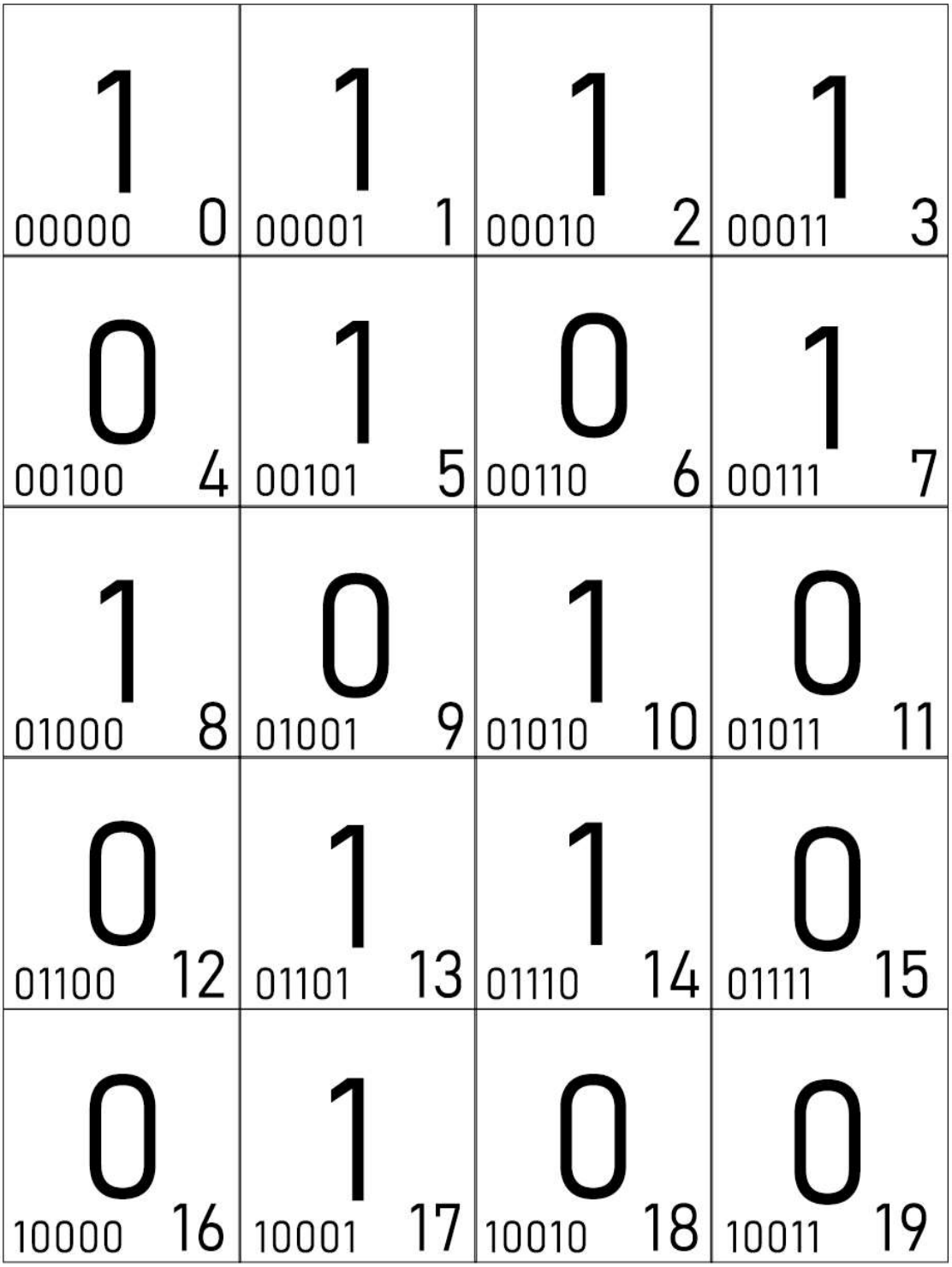


Przykładowe zadania (opcjonalne):

Hamming.1. Zakładając, że bity parzystości są w pozycjach będących potęgami liczby 2 oraz w pozycji 0 (parzystość całego pakietu) znajdź błąd w następującym pakiecie danych:



Hamming.2. Zakładając, że bity parzystości są w pozycjach będących potęgami liczby 2 oraz w pozycji 0 (parzystość całego pakietu) znajdź błąd w następującym pakiecie danych:



Hamming.3. Zaimplementuj algorytm dodający do pakietu danych bity nadmiarowe służące do detekcji i korekcji błędu na miejscach podanych w Hamming.1.

Hamming.4. Zaimplementuj algorytm detekcji i korekcji błędu w pakiecie zakodowanym kodem korekcyjnym Hamminga (zawierającym odpowiednie bity nadmiarowe na miejscach podanych
w Hamming.1.)

Zaprojektowanie protokołu komunikacji radiowej.

Protokół komunikacji radiowej powinien zapewniać jak największą odporność na szumy i błędy podczas komunikacji, dlatego też powinien wykorzystać wcześniej omówione narzędzia.

Protokół.1. Zaimplementuj następujący protokół komunikacji radiowej:

Gdzie:

Jeśli ID opcji to 1, długość wartości wynosi 2 bajty

Jeśli ID opcji to 2, długość wartości wynosi 3 bajty

Jeśli ID opcji to 3, długość wartości wynosi 4 bajty

Wartość może być losowymi bajtami o zadanej długości

Protokół.2. Zaimplementuj kodowanie Hamminga dla protokołu Protokół.1.

Protokół.3. Zaprojektuj własny protokół zapewniający jak największą odporność na szumy i błędy podczas komunikacji.

Protokół.4. Zaimplementuj zaprojektowany protokół z Protokół.3.

Protokół.5. Przeprowadź analizę protokołów Protokół.1., Protokół.2. i zaimplementowanego protokołu Protokół.4. Sprawdź jak często występują błędy w komunikacji i jak skutecznie dany protokół jest sobie w stanie z nimi poradzić.

Projektowanie i zbudowanie interfejsu graficznego do użytkowania satelity

Cele:

- Komunikacja z satelitą, wykorzystanie zaprojektowanego protokołu
- Wizualizacja, analiza i zapis danych z czujników
- Odbiór widoku z kamery

Proponowane technologie:

- Python
- node.js
- C#

Nasz projekt

Na podstawie opisanego wyżej programu stworzyliśmy własny projekt satelity CubeSat.

Obecna wersja jest zaprojektowana przez nas i w pełni wydrukowana na drukarce 3D.

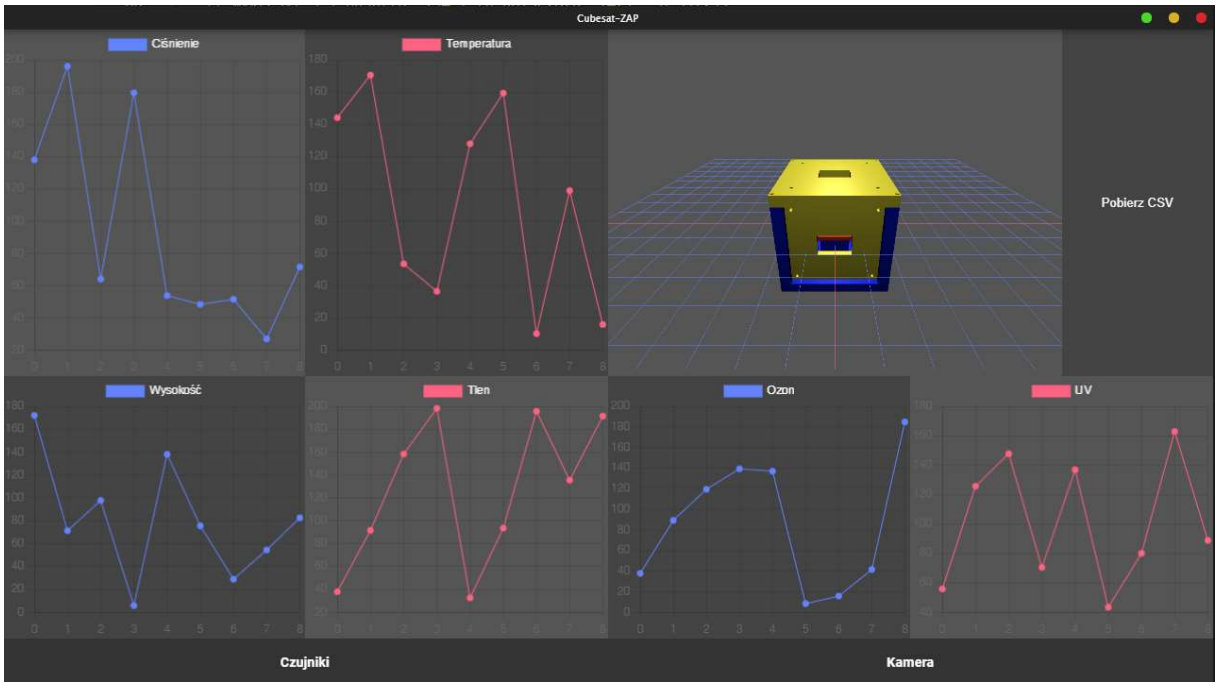
Satelita może być zasilany bezpośrednio przewodowo lub może pracować na ogniwach litowo-jonowych. Zamontowaliśmy także układ ładowania baterii zasilany przez panele słoneczne znajdujące się na obudowie.

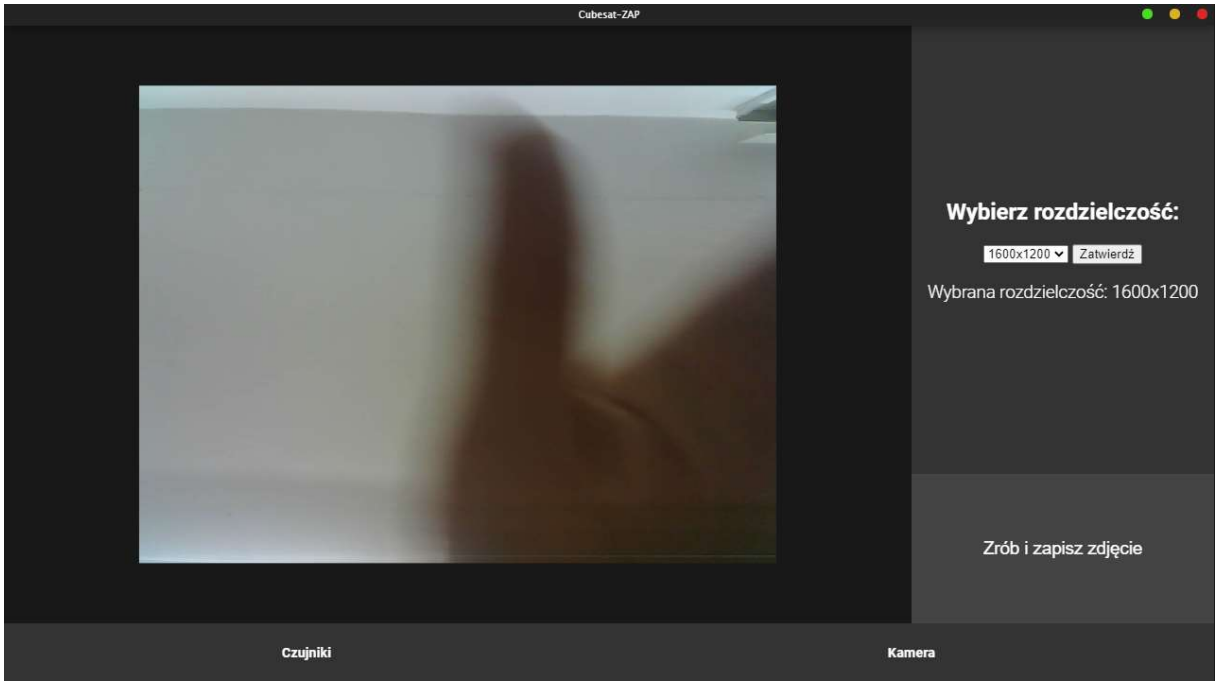
Wykorzystaliśmy i oprogramowaliśmy proponowany przez nas mikrokontroler i czujniki.

Stworzyliśmy własny protokół przesyłu danych drogą radiową.

Stworzyliśmy interfejs graficzny do użytkowania satelity:

- Wykorzystana technologia: node.js (electron)
- Możliwości:
    - pobieranie i wyświetlanie danych z czujników w postaci wykresów
    - możliwość zapisania danych z czujników na dysku w formacie csv
    - pobieranie zdjęcia z kamery mikrokontrolera w wybranym trybie kamery
    - wizualizacja 3D ułożenia satelity (akcelerometr, żyroskop)
- Zrzuty ekranu:





Pierwotnie projekt miał być przystosowany do lotu w kosmos i ostatecznie po zbadaniu prototypu możliwe było wysłanie naszego satelity statkiem transportowym „Progress” na stację kosmiczną ISS za pośrednictwem „SWSU” – („Południowo-zachodni Uniwersytet Stanowy w Rosji” („ЮгоЗападный государственный университет”)), jednakże ze względu na obecną sytuację polityczną taka możliwość już nie istnieje.

W obecnej sytuacji projekt docelowo zakończyć ma się lotem balonem stratosferycznym.