SPRAWOZDANIE			PROSZĘ PODAĆ NR GRUPY:						
			ZIISS1	3	5	1	2	IO	
IMIĘ	**NAZWISKO**	**Temat ćwiczenia zgodny z wykazem tematów:**	**PONIŻEJ PROSZĘ PODAĆ TERMIN ZAJĘĆ:**					**ROK:**	
Amelia	Lis	**Podstawy programowania mikroprocesorów Dekodery**						**2023r.**	
			PN	**WT**	**SR**	**CZ**	**PT**	**SB**	**ND**
			GODZINA ROZPOCZĘCIA ZAJĘĆ:					**11 : 30**	

UWAGA !!! Wypełniamy tylko białe pola. W **punkcie 1**, proszę zakreślić odpowiednie pola i podać godzinę w której odbywają się zajęcia, zgodnie z planem zajęć.

Wprowadzenie teoretyczne:

Opisz rodzaje kodów spotykanych w elektronice cyfrowej Binarny, Gray'a Johnsona, Heksadecymalny, Aikena, (będzie Ci to potrzebne w dalszej części zadania).

Kod Binarny:

Kod binarny, będący fundamentem elektroniki cyfrowej, reprezentuje informację przy użyciu dwóch symboli: 0 i 1. Jest to najprostszy sposób reprezentacji danych, gdzie każdy bit ma wartość potęgi dwójki. Kod binarny jest niezbędny w cyfrowych obliczeniach i przechowywaniu danych, ale jego głównym ograniczeniem jest skłonność do generowania dużych ilości danych. Jednym z kluczowych elementów kodu binarnego jest umiejętność przejścia między stanami logicznymi. Przez odpowiednie ustawienie bitów możliwe jest generowanie różnych stanów, co jest kluczowe w układach cyfrowych i procesorach.

Kod Gray'a:

Kod Gray'a jest jednym z unikalnych kodów, w których sąsiednie liczby różnią się tylko jednym bitem. Jest wykorzystywany w celu eliminacji błędów, które mogą wystąpić w przypadku niezamierzonego zakłócenia sygnału. Dzięki swojej własności minimalnych zmian między kolejnymi liczbami, kod Gray'a jest popularny w zastosowaniach, gdzie istotna jest niezawodność transmisji, takich jak w systemach komunikacji radiowej. W przypadku długich linii przewodzących, takich jak przewody sygnałowe w przemyśle, kod Gray'a pomaga w redukcji interferencji elektromagnetycznej. Minimalna zmiana bitu pomiędzy kolejnymi wartościami zmniejsza ryzyko wprowadzenia błędów.

Kod Johnsona:

Kod Johnsona, znany również jako kod cykliczny, to rodzaj sekwencyjnego kodu binarnego, który ma unikalne właściwości cyklicznego przesuwania jedynki wzdłuż bitów. Jest często używany w różnych zastosowaniach, takich jak multipleksacja, demultipleksacja, układy sterowania oraz w zagadnieniach synchronizacji danych. Kod Johnsona charakteryzuje się tym, że sąsiednie stany różnią się od siebie tylko jednym bitem. Podstawową cechą kodu Johnsona jest to, że dla n-bitowego kodu jest 2^n różnych stanów. Każdy kolejny stan różni się od poprzedniego tylko jednym bitem.

Kod Heksadecymalny:

Kod heksadecymalny, bazujący na systemie szesnastkowym, jest używany do reprezentacji dużych ilości danych w sposób bardziej zwięzły niż kod binarny. Składa się z 16 symboli, od 0 do 9 i od A do F, co pozwala na reprezentację czterech bitów kodu binarnego jednym symbolem heksadecymalnym. Jest szeroko stosowany w programowaniu mikrokontrolerów, konfiguracji układów FPGA oraz w edycji i analizie kodu maszynowego.

Kod Aikena:

Kod Aikena, znany również jako kod "1 z N" lub kod 2421, to forma kodowania, w której tylko jedna pozycja posiada wartość 1 w danym momencie, a reszta pozycji ma wartość 0. Nazwa "2421" odnosi się do wzoru, który opisuje sposób reprezentacji kolejnych cyfr dziesiętnych. Kod ten jest szeroko stosowany w systemach, w których istotne jest minimalizowanie zużycia energii, ponieważ tylko jedno źródło generuje prąd w danym momencie. W praktyce jest stosowany w niektórych aplikacjach, takich jak wyświetlacze na urządzeniach niskonapięciowych, w których oszczędność energii jest kluczowa.

* rysunki proszę zamieszczać na drugiej stronie a w tekście podać odnośniki

Zadanie 1

W programie Simulide zaprojektuj system mikroprocesorowy zamieniający kod binarny na kod wyświetlacza siedmiosegmentowego

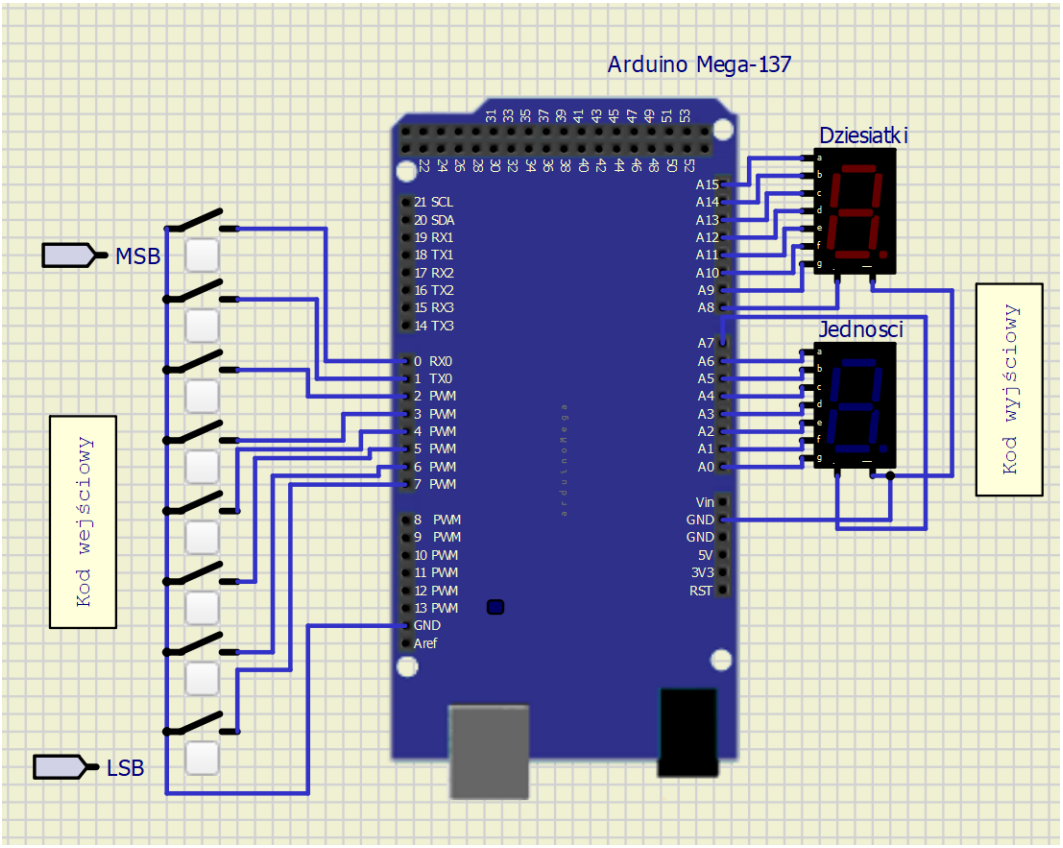
System mikroprocesorowy oparty jest na układzie Arduino Mega wykorzystujący procesor Atmega 2560

Zaprogramuj ten procesor w środowisku Simulide wykorzystując porty wyjściowe i wejściowe.

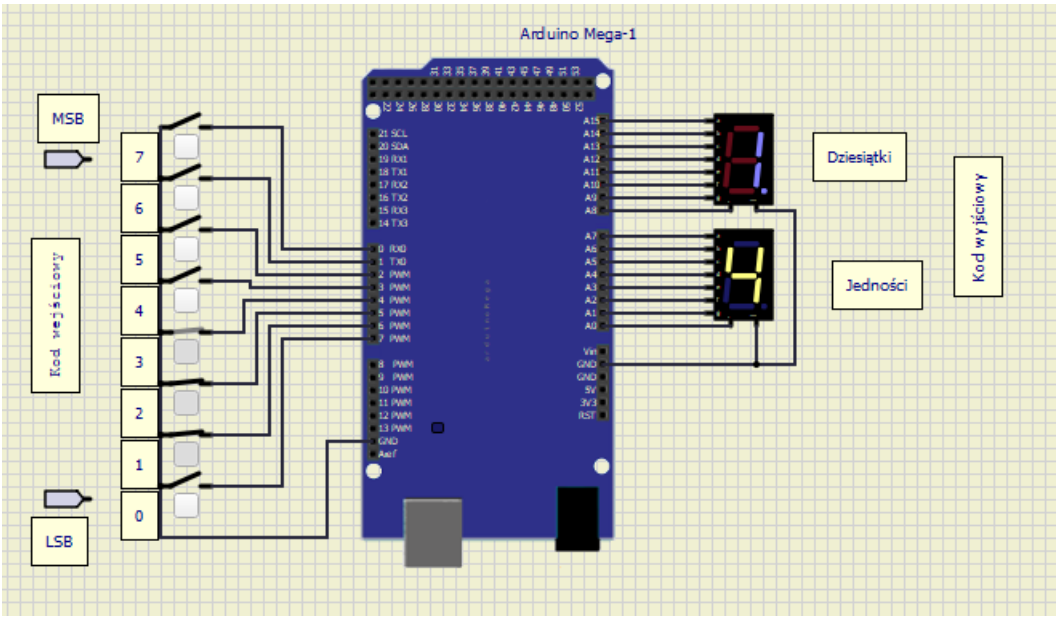
W przypadku gdy suma liczb występujących w Twoim numerze indeksu daje liczbę na wyświetlaczu „Dziesiątek" zapal kropkę a w przypadku liczb nieparzystych zapal kropkę na wyświetlaczu: „Jedności".

Komentarz: Suma cyfr w moim numerze indeksu wynosi 14. W moim przypadku, suma liczb daje liczbę na wyświetlaczu dziesiątek, więc jeśli jest to liczba większa od 9, to kropka wyświetla się. Nie jest to liczba nieparzysta, więc dlatego nie wyświetlam kropki na wyświetlaczu jedności.

Schemat układu:



Zrzut ekranu Schematu zaprojektowanego przez Ciebie



Tutaj proszę wkleić kod programu.

```
const int tensPins[] = {A15, A14, A13, A12, A11, A10, A9, A8};
const int onesPins[] = {A7, A6, A5, A4, A3, A2, A1, A0};


void setup() {
    for (int i = 0; i < 8; i++) {
        pinMode(tensPins[i], OUTPUT);
        pinMode(onesPins[i], OUTPUT);
    }

    for (int i = 0; i < 8; i++) {
        digitalWrite(i, HIGH);
    }
}

void loop() {
    checkNumber();
}

void checkNumber() {
    int numberDec = 0;

    for (int i = 0; i < 8; i++) {
        if (digitalRead(i) == LOW) {
            numberDec += (1 << (7 - i));
        }
    }

    int countOfTens = numberDec % 100;
    int numberOfOnes = countOfTens % 10;
    int numberOfTens = (countOfTens - numberOfOnes) / 10;

    displayNumberOnes(numberOfOnes);
    displayNumberTens(numberOfTens);

        if (numberOfTens > 0) {
           digitalWrite(A8, HIGH);
    } else {
                digitalWrite(A8, LOW);
        }
    if (numberDec % 2 == 1) {
           digitalWrite(A0, HIGH);
```

```
        } else {
                digitalWrite(A0, LOW);
        }
        clearDisplay();
}

void displayNumberOnes(int number) {
    // Implementacja za pomocą tablicy dwuwymiarowej
    const int displayPatterns[][7] = {
        {A7, A6, A5, A4, A3, A2},
        {A5, A6},
        {A7, A6, A1, A3, A4},
        {A7, A6, A5, A4, A1},
        {A6, A5, A1, A2},
        {A7, A5, A4, A1, A2},
        {A7, A5, A4, A3, A2, A1},
        {A7, A6, A5},
        {A7, A6, A5, A4, A3, A2, A1},
        {A7, A6, A5, A4, A2, A1}};

    for (int i = 0; i < 7; i++) {
        digitalWrite(displayPatterns[number][i], HIGH);
    }
}

void displayNumberTens(int number) {
    // Implementacja za pomocą tablicy dwuwymiarowej
    const int displayPatterns[][7] = {
        {A15, A14, A12, A13, A11, A10},
        {A14, A13},
        {A15, A14, A9, A11, A12},
        {A15, A14, A13, A12, A9},
        {A10, A9, A14, A13},
        {A15, A10, A9, A13, A12},
        {A15, A10, A11, A12, A13, A9},
        {A15, A14, A13},
        {A15, A14, A13, A12, A11, A10, A9},
        {A15, A14, A13, A12, A10, A9}};

    for (int i = 0; i < 7; i++) {
        digitalWrite(displayPatterns[number][i], HIGH);
    }
}

void clearDisplay() {
    delay(500);

    for (int i = 0; i < 7; i++) {
        digitalWrite(tensPins[i], LOW);
        digitalWrite(onesPins[i], LOW);
    }
}
```

Zadanie 2

W programie Simulide zaprojektuj system mikroprocesorowy zamieniający kod Graya na kod wyświetlacza siedmiosegmentowego

System mikroprocesorowy oparty jest na układzie Arduino Mega wykorzystujący procesor Atmega 2560

Zaprogramuj ten procesor w środowisku Simulide wykorzystując porty wyjściowe i wejściowe.

W przypadku liczb parzystych występujących w Twoim numerze indeksu na wyświetlaczu „Dziesiątek" zapal kropkę a w przypadku liczb nieparzystych zapal kropkę na wyświetlaczu: „Jedności"

Komentarz: Nie do końca zrozumiałam treść zadania dotyczącego liczb w numerze indeksu. Obawiam się, że w poleceniu „W przypadku liczb parzystych…", zabrakło słowa „suma", czyli: „W przypadku liczb parzystych występujących w sumie liczb z indeksu zapal kropkę". Dlatego zaimplementowałam rozwiązanie, że jeśli liczba jest parzysta, wyświetla kropkę na wyświetlaczu dziesiątek, a w przypadku liczb nieparzystych, zapalić kropkę na wyświetlaczu jedynek.

Tutaj proszę wkleić kod programu (nie trzeba robić ponownie zrzutu ekranu projektu)

```
const int tensPins[] = {A15, A14, A13, A12, A11, A10, A9, A8};
const int onesPins[] = {A7, A6, A5, A4, A3, A2, A1, A0};
int pressedBits[] = {0,0,0,0,0,0,0,0};
int binaryArray[] = {0,0,0,0,0,0,0,0};

void setup() {
    for (int i = 0; i < 8; i++) {
        pinMode(tensPins[i], OUTPUT);
        pinMode(onesPins[i], OUTPUT);
    }

    for (int i = 0; i < 8; i++) {
        digitalWrite(i, HIGH);
    }
}

void loop() {

        for (int i=0; i<8; i++) {
                if (digitalRead(i) == LOW) {
                        pressedBits[i] = 1;
                } else {
                        pressedBits[i] = 0;
                }
        }

        changeGreyToBinary(); //najpierw zamieniam na binarny,
        changeBinaryToDecimal(); //by potem zamienić na decimal
}

void changeBinaryToDecimal() {
    int numberDec = 0;

    for (int i = 0; i < 8; i++) {
        if (binaryArray[i] == 1) {
            numberDec += (1 << (7 - i));
        }
    }

    int countOfTens = numberDec % 100;
    int numberOfOnes = countOfTens % 10;
    int numberOfTens = (countOfTens - numberOfOnes) / 10;

    displayNumberOnes(numberOfOnes);
    displayNumberTens(numberOfTens);

    if (numberOfOnes % 2 == 1) {
           digitalWrite(A0, HIGH);
        digitalWrite(A8, LOW);
        } else {
                digitalWrite(A8, HIGH);
                digitalWrite(A0, LOW);
        }
        clearDisplay();
```

```
}

void displayNumberOnes(int number) {
    // Implementacja za pomocą tablicy dwuwymiarowej
    const int displayPatterns[][7] = {
        {A7, A6, A5, A4, A3, A2},
        {A5, A6},
        {A7, A6, A1, A3, A4},
        {A7, A6, A5, A4, A1},
        {A6, A5, A1, A2},
        {A7, A5, A4, A1, A2},
        {A7, A5, A4, A3, A2, A1},
        {A7, A6, A5},
        {A7, A6, A5, A4, A3, A2, A1},
        {A7, A6, A5, A4, A2, A1}};

    for (int i = 0; i < 7; i++) {
        digitalWrite(displayPatterns[number][i], HIGH);
    }
}

void displayNumberTens(int number) {
    // Implementacja za pomocą tablicy dwuwymiarowej
    const int displayPatterns[][7] = {
        {A15, A14, A12, A13, A11, A10},
        {A14, A13},
        {A15, A14, A9, A11, A12},
        {A15, A14, A13, A12, A9},
        {A10, A9, A14, A13},
        {A15, A10, A9, A13, A12},
           {A15, A10, A11, A12, A13, A9},
        {A15, A14, A13},
        {A15, A14, A13, A12, A11, A10, A9},
        {A15, A14, A13, A12, A10, A9}};

    for (int i = 0; i < 7; i++) {
        digitalWrite(displayPatterns[number][i], HIGH);
    }
}

void clearDisplay() {
    delay(500);

    for (int i = 0; i < 7; i++) {
        digitalWrite(tensPins[i], LOW);
        digitalWrite(onesPins[i], LOW);
    }
}


void changeGreyToBinary() {
        int numberDec = 0;
        binaryArray[0] = pressedBits[0];
        for (int i=1; i<8; i++) {
                if (pressedBits[i] == binaryArray[i-1]) {
                        binaryArray[i] = 0;
                } else {
                        binaryArray[i] = 1;
                }
        }


}
```

Zadanie 3

Rozbuduj układ tak aby przyciski podłączone do pinów od 31 do 35, które mogą przyjmować w kodzie binarnym wartości wartości 0, 1, 2, 3, 4, 5 zaprogramuj tak aby wartość ustawiona na górnym przycisku transformowała kody:

0 – transformował kod binarny na kod dziesiętny wyświetlając go na wyświetlaczu siedmiosegmentowym (ten program już masz gotowy)

1 – transformował kod Graya wyświetlał kod dziesiętny na wyświetlaczu siedmiosegmentowym (ten program masz gotowy)

2 – transformował kod binarny i wyświetlał na wyświetlaczu kod szesnastkowy (jedności pierwszy półbajt, dziesiątki drugi półbajt)

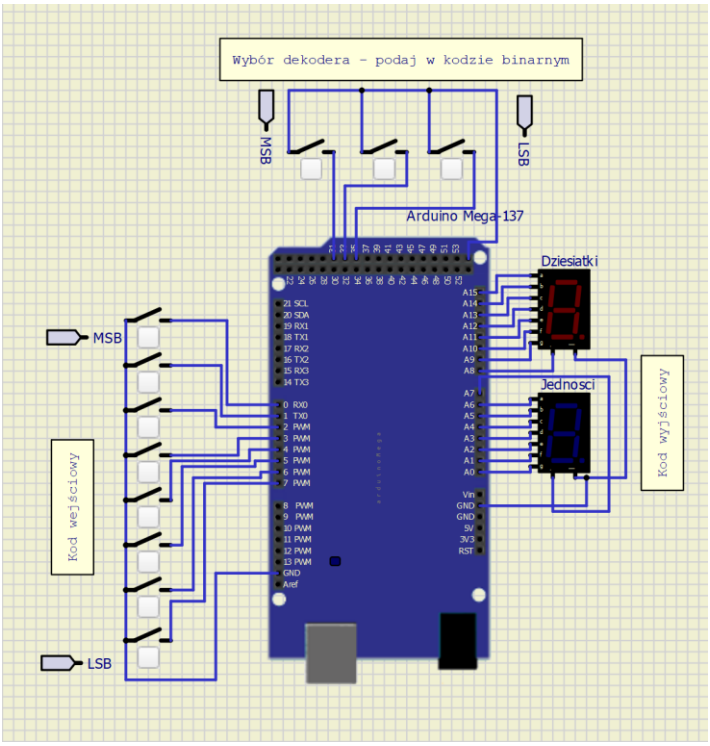
3 – transformował kod Graya i wyświetlał go na wyświetlaczu w postaci kodu szesnastkowego.

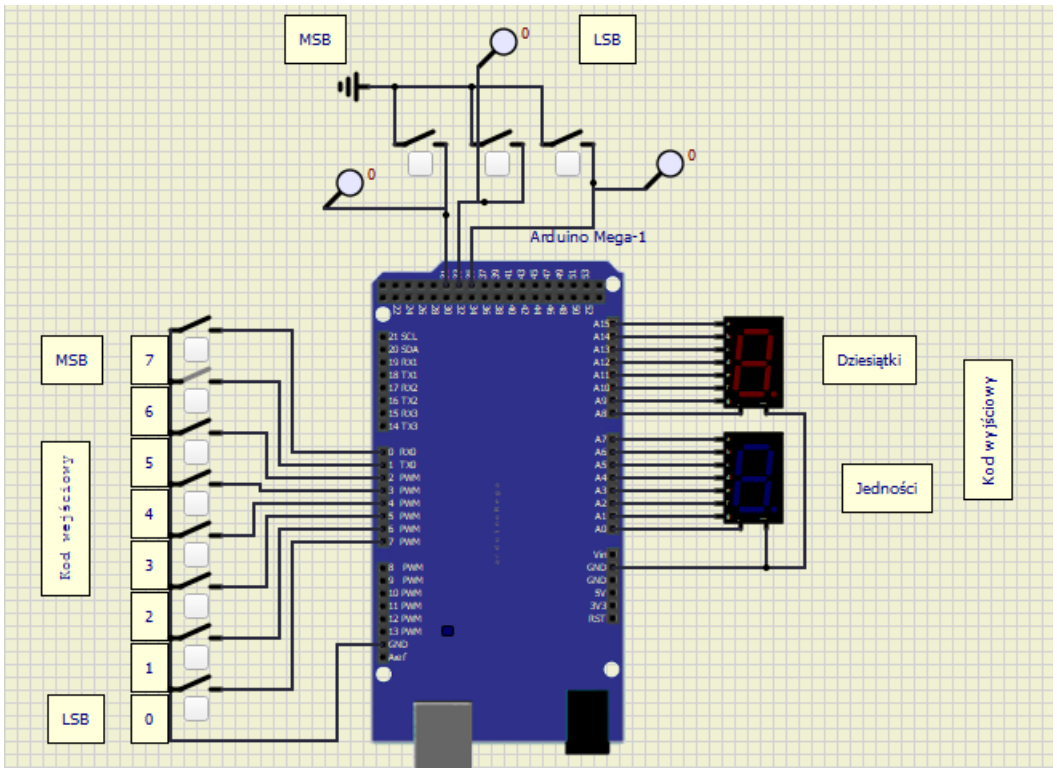
4 - transformował kod AIKENA wyświetlał go na wyświetlaczu w postaci kodu dziesiętnego

5 – transformował kod AIKENA i wyświetlał go na wyświetlaczu w postaci kodu szesnastkowego.

6 – transformował kod JOHNSONA i wyświetlał go na wyświetlaczu w postaci kodu dziesiętnego

7- transformował kod JOHNSONA i wyświetlał go na wyświetlaczu w postaci kodu szesnastkowego





Tutaj wklej listing Twojego programu

```
const int tensPins[] = {A15, A14, A13, A12, A11, A10, A9, A8};
const int onesPins[] = {A7, A6, A5, A4, A3, A2, A1, A0};
int pressedBits[] = {0,0,0,0,0,0,0,0};
int binaryArray[] = {0,0,0,0,0,0,0,0};

void setup() {
    for (int i = 0; i < 8; i++) {
        pinMode(tensPins[i], OUTPUT);
        pinMode(onesPins[i], OUTPUT);
    }

    for (int i = 0; i < 8; i++) {
        digitalWrite(i, HIGH);
    }
        pinMode(31, INPUT);
        pinMode(33, INPUT);
        pinMode(35, INPUT);
        digitalWrite(31, HIGH);
        digitalWrite(33, HIGH);
        digitalWrite(35, HIGH);

}

void loop() {

        for (int i=0; i<8; i++) {
                if (digitalRead(i) == LOW) {
                        pressedBits[i] = 1;
                } else {
                        pressedBits[i] = 0;
                }
        }
        checkAlgorithm();
        //changeGreyToBinary(); //najpierw zamieniam na binarny,
    //changeBinaryToDecimal(); //by potem zamienić na decimal
}

void changeBinaryToDecimal() {
```

```
    int numberDec = 0;

    for (int i = 0; i < 8; i++) {
        if (binaryArray[i] == 1) {
            numberDec += (1 << (7 - i));
        }
    }

    int countOfTens = numberDec % 100;
    int numberOfOnes = countOfTens % 10;
    int numberOfTens = (countOfTens - numberOfOnes) / 10;

    displayNumberOnes(numberOfOnes);
    displayNumberTens(numberOfTens);

        clearDisplay();
}

void displayNumberOnes(int number) {
    // Implementacja za pomocą tablicy dwuwymiarowej
    const int displayPatterns[][15] = {
        {A7, A6, A5, A4, A3, A2},
        {A5, A6},
        {A7, A6, A1, A3, A4},
        {A7, A6, A5, A4, A1},
        {A6, A5, A1, A2},
        {A7, A5, A4, A1, A2},
        {A7, A5, A4, A3, A2, A1},
        {A7, A6, A5},
        {A7, A6, A5, A4, A3, A2, A1},
        {A7, A6, A5, A4, A2, A1},
           {A7, A6, A5, A3, A2, A1}, //A
           {A5, A4, A3, A2, A1}, //b
           {A7, A4, A3, A2}, //C
           {A6, A5, A4, A3, A1}, //d
           {A7, A4, A3, A2, A1}, //E
           {A7, A3, A2, A1} //F
};

    for (int i = 0; i < 15; i++) {
        digitalWrite(displayPatterns[number][i], HIGH);
    }
}

void displayNumberTens(int number) {
    // Implementacja za pomocą tablicy dwuwymiarowej
    const int displayPatterns[][15] = {
        {A15, A14, A12, A13, A11, A10},
        {A14, A13},
        {A15, A14, A9, A11, A12},
        {A15, A14, A13, A12, A9},
        {A10, A9, A14, A13},
        {A15, A10, A9, A13, A12},
           {A15, A10, A11, A12, A13, A9},
        {A15, A14, A13},
        {A15, A14, A13, A12, A11, A10, A9},
        {A15, A14, A13, A12, A10, A9},
           {A15, A14, A13, A11, A10, A9}, //A
           {A13, A12, A11, A10, A9}, //b
           {A15, A12, A11, A10}, //C
           {A14, A13, A12, A11, A9}, //d
           {A15, A12, A11, A10, A9}, //E
           {A15, A11, A10, A9} //F

};

    for (int i = 0; i < 15; i++) {
        digitalWrite(displayPatterns[number][i], HIGH);
    }
}

void clearDisplay() {
    delay(500);
```

```
    for (int i = 0; i <= 7; i++) {
        digitalWrite(tensPins[i], LOW);
        digitalWrite(onesPins[i], LOW);
    }
}


void changeGreyToBinary() {
        int numberDec = 0;
        binaryArray[0] = pressedBits[0];
        for (int i=1; i<8; i++) {
                if (pressedBits[i] == binaryArray[i-1]) {
                        binaryArray[i] = 0;
                } else {
                        binaryArray[i] = 1;
                }
        }
}

void changeAikenToDec() {
        int aiken[] = {2,4,2,1};

        int firstAiken = 0;
        for (int i = 0; i <=3; i++){
                if (pressedBits[i] == 1){
                        firstAiken += aiken[i];
                }
        }
        int secondAiken = 0;
        for (int i = 0; i <=3; i++){
                if (pressedBits[i + 4] == 1){
                        secondAiken += aiken[i];
                }
        }

        displayNumberTens(firstAiken);
        displayNumberOnes(secondAiken);

        clearDisplay();
}

void changeBinaryToHex() {
        int firstByte = 0;
        int secondByte = 0;

        for (int i=0; i<=3; i++) {
                if (binaryArray[i] == 1) {
                        firstByte += (1 << (3-i));
                }
        }

        for (int i=0; i<=3; i++) {
                if (binaryArray[4+i] == 1) {
                        secondByte += (1 << (3-i));
                }
        }
        displayNumberTens(firstByte);
        displayNumberOnes(secondByte);

        clearDisplay();
}


void checkAlgorithm() {
        int algorithmNumber = 0;

        if (digitalRead(31)==LOW) {
                algorithmNumber += 4;
        }
        if (digitalRead(33)==LOW) {
                algorithmNumber += 2;
        }
        if (digitalRead(35)==LOW) {
                algorithmNumber += 1;
```

```
        }
        //BIN TO DEC
        if (algorithmNumber == 0) {
                memcpy(binaryArray, pressedBits, sizeof(binaryArray));
                changeBinaryToDecimal();
        }
        //GRAY TO DEC
        if (algorithmNumber == 1) {
                changeGreyToBinary();
                changeBinaryToDecimal();
        }
        //BIN TO HEX
        if (algorithmNumber == 2) {
                memcpy(binaryArray, pressedBits, sizeof(binaryArray));
                changeBinaryToHex();
        }
        //GRAY TO HEX
        if (algorithmNumber == 3) {
                changeGreyToBinary();
                changeBinaryToHex();
        }
        //AIKEN TO DEC
        if (algorithmNumber==4) {
                changeAikenToDec();
        }
        //AIKEN TO HEX
        if (algorithmNumber==5) {
                changeAikenToDec();
        }
        if (algorithmNumber==6) {
                changeJohnsonToDecimal();
        }
        if (algorithmNumber==7) {
                changeJohnsonToHex();
        }
}


void changeJohnsonToDecimal() {
}
Void changeJohnsonToHex() {
}
```

Wnioski:
Implementacja algorytmu translacji kodu binarnego na decymalny i szesnastkowy sprawiły mi najmniej problemów, jednakże implementacja pozostałych rozwiązań wymagała większej ilości wysiłku.

Implementacja algorytmu kodowania Graya wymagała staranności w obszarze manipulacji bitami. Testowanie na różnych przypadkach wejściowych było kluczowe dla zapewnienia poprawności działania algorytmu, ze względu na moją małą znajomość kodu Graya. Algorytm translacji był łatwy do zrozumienia i implementacji.

Algorytm zamiany kodu Aikena (inaczej kod 2421) na system szesnastkowy, działa tak samo jak zamiana kodu Aikena na kod dziesiątkowy, gdyż kod ten ma 10 możliwości wyświetlania cyfr (0,1,2,3,4,5,6,7,8,9). Każde 4 bity traktuję jako osobną cyfrę, więc jest to niemożliwe, aby zamiana kodu Aikena na system szesnastkowy wyświetlała litery A, B, C, D, E lub F.

Jeśli chodzi o kod Johnsona, nie miałam pewności co do implementacji rozwiązania, ze względu na małą liczbę informacji jak działa algorytm translacji na kod dziesiątkowy i szesnastkowy.
Z dostępnych informacji, wywnioskowałam, że kod Johnsona może być zależny od tego ile bitów znajduje się w dekoderze.
Liczniki Johnsona mają 2N stanów (gdzie N to liczba przerzutników) w porównaniu do stanów 2^n w przypadku zwykłego licznika binarnego.

W czasie implementacji poszczególnych algorytmów, zauważyłam zależność, że lepiej jest najpierw konwertować dany kod/system na binarny, by dopiero potem skonwertować go na system dziesiętny/szesnastkowy. Sposób ten skutecznie przyspieszył czas stworzenia algorytmów translacji.