Sprawozdanie Niduc 2				
Numer grupy	7	Termin zajęć	Czw. parz. 11:15	
Skład grupy:		**Prowadzący**		Ocena
Aleksander Dziągwa	281055	Prof. hab. Stanisław Piestrak		
Filip Kwiek	280947			
Kacper Mikosiński	280985			

Spis treści

1 Cel projektu

Zadanie polega na rozważeniu systemu tolerującego uszkodzenia składającego się z pewnej liczby czujników lub kanałów obliczeniowych, zwielokrotnionych (min. 3) celem podwyższenia niezawodności, znajdujących zastosowanie w tzw. systemach krytycznych, których uszkodzenie może zagrażać zdrowiu lub życiu ludzkiemu (np. system wspomagania hamulców ABS, kolejowe lub lotnicze systemy sterowania , etc.). Wyniki dostarczane przez całkowicie sprawne czujniki lub kanały obliczeniowe w zasadzie powinny być identyczne lub przynajmniej zbliżone do siebie. Wynik uznany za poprawny jest wybierany z użyciem algorytmu głosującego (istnieje szereg modeli do wyboru). Celem projektu jest implementacja i porównanie różnych algorytmów głosowania dla róznych modeli błędów i uszkodzeń.

2 Implementacja projektu

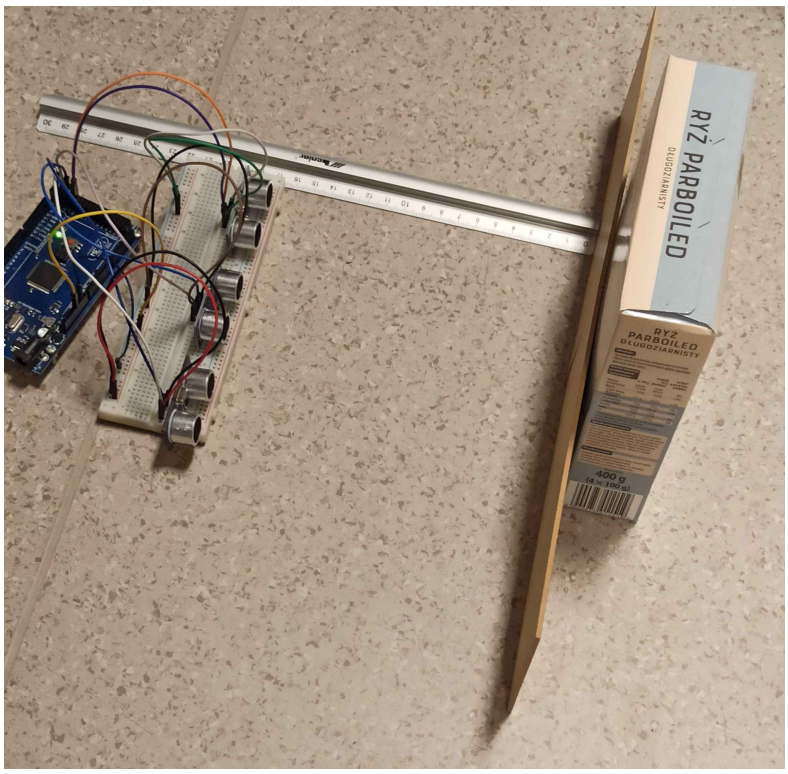
W naszej implementacji, projekt został wykonany za pomocą platformy Arduino Mega oraz algorytmów napisanych w języku C/C++. Jako algorytmów głosowania użyliśmy:

- Większościowego
- Medianowego
- Largest Difference Rejection
- Iteracyjnego

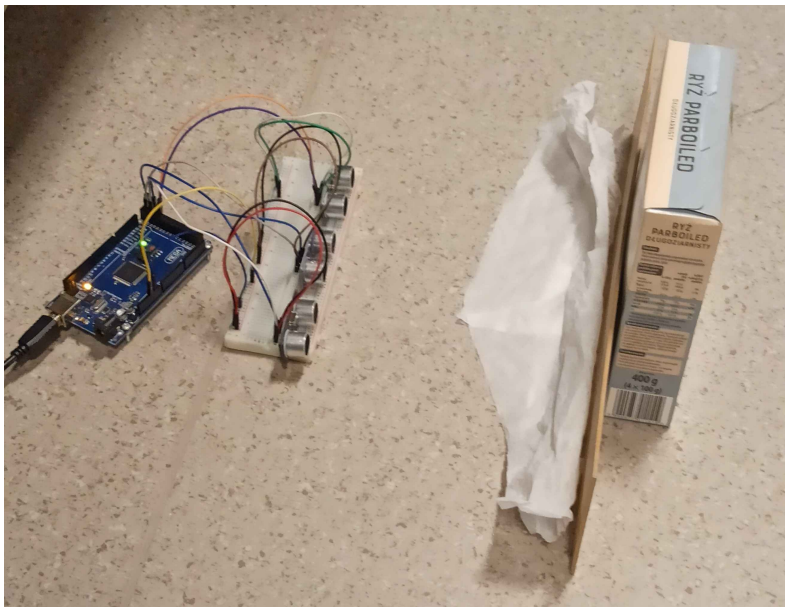
2.1 Opis systemu

Nasz system składał się z mikrokontrolera Arduino oraz trzech dalmierzy sonicznych. Mierzyły one dystans do najbliższej przeszkody, która była naprzeciwko ich. Następnie dane były zbierane przez nasz kod na płytce i następnie za pomocą wspomnianych wyżej algorytmów wykrywało, który dalmierz pokazywał wynik niezgodny z resztą, co może być wynikiem awarii lub jakiegoś bliżej nieznanego błędu.

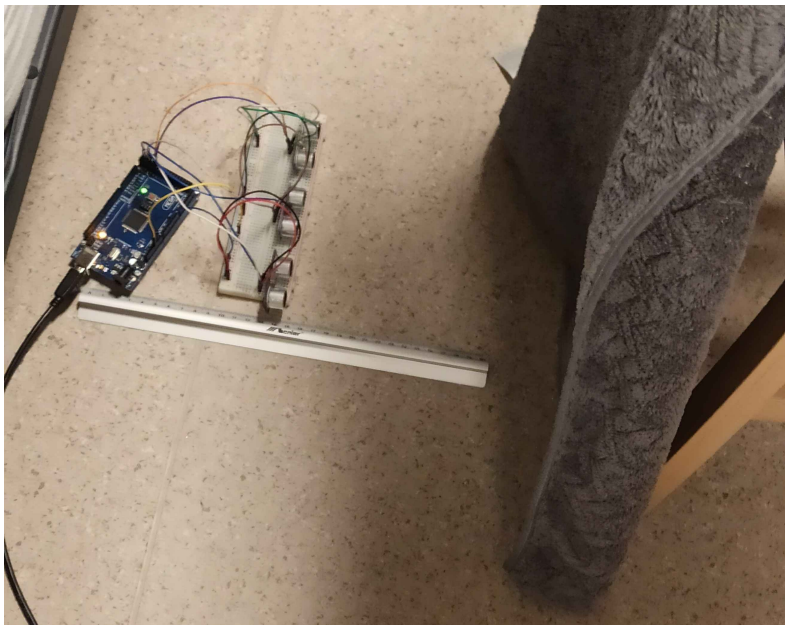
2.2 Implementacja na mikrokontrolerze Arduino



Obraz 1: Podstawowa implementacja systemu



Obraz 2: Implementacja z zakłóceniami w postaci kawałka chusteczki



Obraz 3: Implementacja systemu, który mierzy odległość od powierzchnii, która wytłumia dźwięk

3 Opis algorytmów oraz ich implementacja w języku C

3.1 Algorytm głosowania większościowego

Algorytm głosowania większościowego porównuje parami dwa pomiary, jeżeli ich różnica jest mniejsza niż granica tolerancji, zwraca jeden z pomiarów, natomiast gdy każda para wykracza poza tolerancję algorytmu, zwracany jest błąd czujników.

```c
#define EPSILON 2                       //definicja epsilonu (jednostka -> cm)

bool double_equal(double a, double b) {     //porównanie z uwzględnieniem epsilonu
    return fabs(a - b) < EPSILON;
}

double majority_vote(double module1, double module2, double module3) {      //funkcja głosująca algorytmem głosowania większościowego
                                                                            //w przypadku różnic pomiędzy odczytami większych od wartości
                                                                            //epsilon funkcja zwraca -1, sygnalizując brak wyboru

    if (double_equal(module1, module2)) {
        return module1;
    } else if (double_equal(module1, module3)) {
        return module1;
    } else if (double_equal(module2, module3)) {
        return module2;
    } else {
        return -1;
    }
}

double abs_diff(double a, double b) {       //funkcja obliczająca wartość bezwzględną z różnicy wyników dwóch pomiarów
    return fabs(a - b);
}
```

Obraz 1: algorytm większościowy

3.2 Algorytm Medianowy

Sortuje otrzymane pomiary rosnąco, na koniec zwraca wynik najbliższy prawdzie.

```c
void sort(double *a, double *b, double *c) {    //funckja sortująca rosnąco podane 3 wartości

    double temp;

    if(*a > *b){
        temp = *a;
        *a = *b;
        *b = temp;
    }

    if(*b > *c){
        temp = *b;
        *b = *c;
        *c = temp;
    }

    if(*a > *b){
        temp = *a;
        *a = *b;
        *b = temp;
    }
}

double find_median_sorted(double a, double b, double c) {       //funkcja implementująca algorytm medianowy
                                                                //wynikiem tej funkcji jest mediana z pomiarów
    sort(&a, &b, &c);

    return b;
}
```

Obraz 2: algorytm medianowy

3.3 Algorytm Largest Difference Rejection

Oblicza różnice między wszystkimi parami wartości wejściowych, po czym znajduje parę wartości o najmniejszej różnicy, a jako wynik zwraca średnią arytmetyczną wartości tej pary.

```c
double abs_diff(double a, double b) {       //funkcja obliczająca wartość bezwzględną z różnicy wyników dwóch pomiarów
    return fabs(a - b);
}

double large_difference_rejection(double s1, double s2, double s3) {        //funkcja implementująca algorytm odrzucenia największej różnicy
                                                                            //algorytm ten odrzuca wynik najbardziej odbiegający od reszty
                                                                            //jako wynik zwraca średnią z pozostałych wyników

    double d12 = abs_diff(s1, s2);
    double d23 = abs_diff(s2, s3);
    double d13 = abs_diff(s1, s3);

    if (d12 >= d23 && d12 >= d13)
        return (s1 + s2) / 2.0;

    if (d23 >= d12 && d23 >= d13)
        return (s2 + s3) / 2.0;


    return (s1 + s3) / 2.0;

}
```

Obraz 3: algorytm ldr

3.4 Algorytm Iteracyjny

Algorytm ma na celu znalezienie stabilnej średniej ze wszystkich (w naszym wypadku 3) modułów, aby w sposób iteracyjny odsiać wartości, które mocno

się różnią od aktualnej średniej arytmetycznej lub tolerancji.

```
#define EPSILON 2                                   //definicja epsilonu (jednostka -> cm)

double Average(double* values, bool* isValid)       //funkcja licząca średnią z
{
    double sum = 0.0;
    int count = 0;

    for (int i = 0; i < 3; ++i)
    {
        if (isValid[i])
        {
            sum += values[i];
            count++;
        }
    }

    return (count > 0) ? (sum / count) : 0.0;
}

double iterative_vote(double module1, double module2, double module3)   //funkcja głosująca algorytmem iteracyjnym
                                                                        //program iteracyjnie odrzuca wyniki odbiegające od średniej
                                                                        //w większym stopniu niż wartość epsilon
                                                                        //jako wynik zwraca średnią z nieodrzuconych wyników
{
    bool isValid[3] = {true, true, true};
    double values[3] = {module1, module2, module3};

    for (int i = 0; i < 3; ++i)
    {
        double mean = Average(values, isValid);                         //średnia z nieodrzuconych wyników

        bool stability = true;

        for (int i = 0; i < 3; ++i)
        {
            if (isValid[i] && fabs(values[i] - mean) > EPSILON)
            {
                isValid[i] = false;                                     //odrzuca wyniki odbiegające od średniej
                stability = false;
            }
        }

        if (stability) break;                                           //jeżeli żaden wynik nie został odrzucony, kończy działanie funkcji
    }

    return Average(values, isValid);
}
```

Obraz 4: algorytm iteracyjny

4 Testy systemu

Przeprowadziliśmy 3 serie testów, w których nasz system sprawdzał odległości bez zakłóceń i z zakłóceniami. Jako zakłócenia uznaliśmy wcześniej pokazany kawałek chusteczki oraz koc, który pochłaniał dźwięk soniczny, oraz sami produkowaliśmy zakłócenia dźwiękowe, które będą widoczne w tabelach, jako wartości od razu widoczne jako nieprawidłowe. Odległość mierzona była w centymetrach.

4.1 Tabele wyników testów

Lp	Czujnik 1	Czujnik 2	Czujnik 3	Większościowy	Mediana	ONR	Iteracyjny
1	21,56	21,13	20,97	21,56	21,13	21,27	21,22
2	20,56	21,03	20,96	20,56	20,96	20,8	20,85
3	21,09	21,13	20,96	21,09	21,09	21,05	21,06
4	21,56	20,81	20,55	20,81	20,81	21,05	20,68
5	20,67	21,13	20,96	20,67	20,96	20,9	20,92
6	21,09	21,13	20,96	21,09	21,09	21,05	21,06
7	21,45	21,03	20,85	21,45	21,03	21,15	21,11
8	21,56	21,13	20,96	21,56	21,13	21,26	21,22
9	21,56	21,12	20,97	21,56	21,12	21,27	21,22
10	21,45	21,13	20,96	21,45	21,13	21,21	21,18
11	21,09	21,15	20,96	21,09	21,09	21,05	21,07
12	21,09	21,13	20,85	21,09	21,09	20,99	21,03
13	21,15	20,81	20,97	21,15	20,97	20,98	20,98
14	21,09	21,15	20,96	21,09	21,09	21,05	21,07
15	20,97	20,71	20,44	20,97	20,71	20,71	20,71
16	20,67	21,13	20,96	20,67	20,96	20,9	20,92
17	21,56	21,13	20,55	21,56	21,13	21,05	21,35
18	20,58	21,13	20,96	20,58	20,96	20,86	20,89
19	20,67	21,13	20,55	20,67	20,67	20,84	20,78
20	20,68	21,13	20,97	20,68	20,97	20,91	20,93
21	21,56	21,15	20,55	21,56	21,15	21,05	21,35
22	21,09	21,13	20,96	21,09	21,09	21,05	21,06
23	20,97	21,03	20,85	20,97	20,97	20,94	20,95
24	20,68	21,13	20,97	20,68	20,97	20,91	20,93
25	21,08	21,13	20,96	21,08	21,08	21,05	21,06
26	20,58	21,03	20,55	20,58	20,58	20,79	20,72
27	20,68	21,13	20,55	20,68	20,68	20,84	20,79
28	21,09	21,13	20,56	21,09	21,09	20,85	20,93
29	21,45	20,83	20,55	20,83	20,83	21	20,69
30	21,09	21,13	20,55	21,09	21,09	20,84	20,92
31	20,68	21,13	20,97	20,68	20,97	20,91	20,93
32	20,68	21,13	20,55	20,68	20,68	20,84	20,79
33	20,68	21,13	20,55	20,68	20,68	20,84	20,79
34	20,67	21,13	20,87	20,67	20,87	20,9	20,89
35	21,09	21,13	20,55	21,09	21,09	20,84	20,92
36	20,68	21,13	20,97	20,68	20,97	20,91	20,93
37	21,09	20,81	20,44	21,09	20,81	20,77	20,78
38	20,68	21,13	20,55	20,68	20,68	20,84	20,79

Tabela 1: Tabela wyników z czujników bez zakłóceń

Lp	Czujnik 1	Czujnik 2	Czujnik 3	Większościowy	Mediana	ONR	Iteracyjny
1	3403.30	18.52	19.41	18.52	19.41	1710.91	0.00
2	16.46	16.28	1197.24	16.46	16.46	606.76	0.00
3	19.74	16.28	20.43	19.74	19.74	18.35	20.08
4	3455.64	16.28	18.04	16.28	18.04	1735.96	0.00
5	20.53	16.29	18.81	20.53	18.81	18.41	19.67
6	3450.61	16.28	18.81	-1.00	18.81	1733.44	0.00
7	19.11	16.28	18.38	19.11	18.38	17.69	17.92
8	172.87	16.28	18.40	-1.00	18.40	94.57	0.00
9	3399.71	16.28	19.55	-1.00	19.55	1707.99	0.00
10	19.19	16.28	20.85	19.19	19.19	18.56	19.19
11	3383.35	16.28	20.96	-1.00	20.96	1699.81	0.00
12	19.64	16.38	20.44	19.64	19.64	18.41	20.04
13	3404.89	16.28	149.00	-1.00	149.00	1710.58	0.00
14	18.81	16.28	1197.04	-1.00	18.81	606.66	0.00
15	3412.30	16.28	147.39	-1.00	147.39	1714.29	0.00
16	18.81	16.28	1196.95	-1.00	18.81	606.61	0.00
17	3410.31	16.28	1197.29	-1.00	1197.29	1713.29	0.00
18	18.81	16.28	1197.17	-1.00	18.81	606.72	0.00
19	3646.11	16.28	151.21	-1.00	151.21	1831.19	0.00
20	18.71	16.28	1197.04	-1.00	18.71	606.66	0.00
21	192.99	16.28	147.85	-1.00	147.85	104.63	0.00
22	191.98	16.29	148.67	-1.00	148.67	104.13	0.00
23	192.44	16.60	151.49	-1.00	151.49	104.52	0.00
24	3467.95	16.28	151.97	-1.00	151.97	1742.11	0.00
25	19.67	16.28	1196.83	-1.00	19.67	606.55	0.00
26	3460.68	16.28	1196.92	-1.00	1196.92	1738.48	0.00
27	19.23	16.60	1197.05	-1.00	19.23	606.83	0.00
28	3390.04	16.60	150.73	-1.00	150.73	1703.32	0.00
29	18.80	16.64	1196.97	-1.00	18.80	606.80	0.00
30	3363.54	16.38	1197.22	-1.00	1197.22	1689.96	0.00
31	18.80	16.28	1197.02	-1.00	18.80	606.65	0.00
32	3358.04	16.28	1197.33	-1.00	1197.33	1687.16	0.00
33	20.08	16.38	1197.12	-1.00	20.08	606.75	0.00
34	127.03	16.28	148.21	-1.00	127.03	82.24	0.00
35	3342.65	16.29	1196.86	-1.00	1196.86	1679.47	0.00
36	18.80	16.28	1197.04	-1.00	18.80	606.66	0.00
37	126.22	16.28	1197.05	-1.00	126.22	606.66	0.00
38	3356.31	16.28	1197.17	-1.00	1197.17	1686.29	0.00
39	19.21	16.28	148.09	-1.00	19.21	82.18	0.00
40	124.08	16.60	149.39	-1.00	124.08	83.00	0.00
41	3350.17	16.28	1197.50	-1.00	1197.50	1683.22	0.00
42	19.57	16.28	1197.29	-1.00	19.57	606.78	0.00
43	3354.01	16.60	20.97	-1.00	20.97	1685.30	0.00

Tabela 2: Tabela wyników testów z zakłóceniami w postaci chusteczki higienicznej oraz zakłóceń dźwiękowych

Lp	Czujnik 1	Czujnik 2	Czujnik 3	Większościowy	Mediana	ONR	Iteracyjny
1	18,25	15,95	18,16	18,25	18,16	17,1	18,2
2	18,25	15,95	18,62	18,25	18,25	17,29	18,25
3	18,25	15,97	17,75	18,25	17,75	17,11	18
4	18,66	15,97	17,73	18,66	17,73	17,31	17,73
5	18,66	15,97	17,73	18,66	17,73	17,31	17,73
6	17,78	15,95	17,75	17,78	17,75	16,87	17,77
7	17,78	15,97	17,73	17,78	17,73	16,88	17,76
8	17,78	15,97	17,75	17,78	17,75	16,88	17,77
9	18,66	15,95	17,73	18,66	17,73	17,3	17,73
10	19,59	12,66	16,84	-1	16,84	16,12	16,84
11	20,36	17,25	17,73	17,25	17,73	18,8	17,73
12	270,23	17,24	17,73	17,24	17,73	143,73	0
13	20,89	18,52	147,97	-1	20,89	83,25	0
14	3393,59	17,24	148,3	-1	148,3	1705,41	0
15	18,76	18,52	147,52	18,76	18,76	83,02	0
16	20,01	15,97	11,28	-1	15,97	15,65	15,97
17	35,81	16,28	17,2	16,28	17,2	26,04	0
18	18,99	15,97	17,27	-1	17,27	17,48	17,27
19	19,55	15,95	16,84	15,95	16,84	17,75	16,84
20	19,55	15,95	16,84	15,95	16,84	17,75	16,84
21	17,8	16,28	16,77	16,28	16,77	17,04	16,95
22	70,45	59,29	109,52	-1	70,45	84,4	0
23	77,26	64,98	78,63	-1	77,26	71,81	0
24	76,52	75,99	75,32	76,52	75,99	75,92	75,95
25	74,62	36,79	107,98	-1	74,62	72,38	0
26	3391,82	34,69	1197,46	-1	1197,46	1713,26	0
27	3388,89	24,99	1197,09	-1	1197,09	1706,94	0
28	3410,5	73,57	1196,86	-1	1196,86	1742,04	0
29	80,52	28,97	1197,09	-1	80,52	613,03	0
30	3407,17	19,17	1197,19	-1	1197,19	1713,17	0
31	76,32	20,77	1197,19	-1	76,32	608,98	0
32	3404,27	21,3	1196,81	-1	1196,81	1712,79	0
33	77,96	75,99	1197,09	-1	77,96	636,54	0
34	25,09	22,05	1197,02	-1	25,09	609,54	0
35	24,37	66,39	1196,92	-1	66,39	610,64	0
36	3450,05	22,66	1197,38	-1	1197,38	1736,35	0
37	25,97	23,24	111,51	-1	25,97	67,37	0
38	3402,59	23,14	1196,95	-1	1196,95	1712,86	0
39	80,85	22,05	1197,28	-1	80,85	609,67	0
40	3400,95	37,73	110,77	-1	110,77	1719,34	0
41	24,03	22,55	1196,97	-1	24,03	609,76	0
42	3389,39	62,44	1196,86	-1	1196,86	1725,92	0
43	25,02	22,05	1196,71	-1	25,02	609,38	0
44	3350,42	22,57	1197,05	-1	1197,05	1686,5	0
45	24,3	65,12	1196,78	-1	65,12	610,54	0
46	3377,37	65,56	1197,1	-1	1197,1	1721,47	0
47	24,82	65,87	1196,93	-1	65,87	610,87	0
48	3360,46	65,12	1196,74	-1	1196,74	1712,79	0
49	78,82	24,61	1197,31	-1	78,82	610,96	0
50	3443,46	24,71	1196,86	-1	1196,86	1734,09	0

Tabela 3: Tabela wyników czujników z zakłóceniami dźwiękowymi oraz powierzchnią pochłaniającą dźwięk

5 Porównanie algorytmów

5.1 Algorytm głosowania większościowego

Algorytm ten sprawdza się dobrze w sytuacjach, gdzie dane z czujników są generalnie zgodne, a liczba zakłóceń jest niewielka. Dzięki prostocie jego implementacji i niskim wymaganiom obliczeniowym, jest szczególnie użyteczny w systemach czasu rzeczywistego o ograniczonych zasobach obliczeniowych. Jednak w obecności dużych zakłóceń lub systematycznych błędów jednego z czujników może zwracać niepoprawne wyniki, gdyż nie potrafi efektywnie odrzucać wartości odstających.

5.2 Algorytm medianowy

Algorytm medianowy wyróżnia się odpornością na wartości odstające. Z tego powodu znajduje zastosowanie w systemach, gdzie zakłócenia są częste i mogą znacząco odbiegać od typowych wartości. Jego wadą jest większa złożoność obliczeniowa w porównaniu do algorytmu większościowego, co może być istotne w systemach o ograniczonej mocy obliczeniowej. Warto również zauważyć, że algorytm medianowy najlepiej działa, gdy liczba czujników jest nieparzysta, co zapewnia jednoznaczność wyniku.

5.3 Algorytm Largest Difference Rejection (LDR)

Algorytm LDR jest przydatny w systemach, gdzie dokładność wyników ma kluczowe znaczenie, a liczba czujników jest wystarczająca do wykrycia i odrzucenia najbardziej odstających wartości. Jego działanie jest szczególnie efektywne w sytuacjach, gdy zakłócenia są mniejsze, ale bardziej równomiernie rozłożone pomiędzy czujnikami. Wadą algorytmu jest wyższa złożoność obliczeniowa w porównaniu do większościowego i medianowego, co może być ograniczeniem w przypadku systemów czasu rzeczywistego.

5.4 Algorytm iteracyjny

Algorytm iteracyjny jest najbardziej zaawansowanym z analizowanych rozwiązań. Dzięki iteracyjnej eliminacji odstających wartości, charakteryzuje się wysoką skutecznością w sytuacjach, gdzie zakłócenia są znaczące i nieprzewidywalne. Algorytm ten wymaga jednak znacznie większej mocy obliczeniowej i jest bardziej skomplikowany do implementacji, co może ograniczać jego zastosowanie w systemach o niskiej wydajności. Algorytm iteracyjny znajduje zastosowanie w krytycznych systemach bezpieczeństwa, gdzie poprawność wyników jest kluczowa.

5.5 Porównanie zastosowań algorytmów

W tabeli 4 zestawiono przypadki użycia każdego algorytmu:

Algorytm	Przydatność	Ograniczenia
Większościowy	Proste systemy, niewielka liczba zakłóceń	Niska odporność na wartości odstające
Medianowy	Odporność na duże zakłócenia	Większa złożoność obliczeniowa
LDR	Dokładne systemy, równomierne zakłócenia	Wysoka złożoność, wymagania obliczeniowe
Iteracyjny	Krytyczne systemy, znaczne zakłócenia	Złożoność implementacji, wymagania sprzętowe

Tabela 4: Porównanie zastosowań algorytmów.

6 Wnioski

Podsumowując, wybór odpowiedniego algorytmu powinien być uzależniony od specyfiki systemu, dostępnych zasobów oraz rodzaju i częstotliwości występujących zakłóceń. Algorytm większościowy będzie dobrym wyborem w systemach o niskiej złożoności, podczas gdy algorytm iteracyjny najlepiej sprawdzi się w złożonych systemach krytycznych.

7 Literatura (Artykuły naukowe)

- "Experimental Comparison of Voting Algorithms in Cases of Disagreement" (EUROMICRO 1997)
- "A Taxonomy for Software Voting Algorithms Used in Safety-Critical Systems" (IEEE Transactions on Reliability, 2004)
- "A Taxonomy of Voting Schemes for Data Fusion and Dependable Computation" (Reliability Engineering and System Safety, 1996)