Командный практический тур

Задача III.2.1. Работа на стенде ОВКС. Восстановление закодированных и декодированных последовательностей (6 баллов)

Исправление закодированных кодом Хемминга последовательностей бит и нахождение исходных последовательностей. Прорези и стенки на шестернях представляют код Хемминга различной длины. По рисунку прорезей на каждой шестеренке необходимо определить код Хемминга: элемент с наименьшим угловым размером вида прорезь означает «1», стенка — «0». Контрольные биты находятся в позициях, соответствующих степеням двойки. Белая точка — начало закодированной последовательности. Последовательность кодировалась по часовой стрелке. Последовательность прорезей оптопарой может читаться как по часовой, так и против часовой стрелки.

Обратите внимание, что в двух из трех шестерней набора допущены однократные ошибки, которые вам необходимо найти и исправить.

Формат записи данных в файл

4 столбца данных, которые разделены пробелом. В первом столбце отсчет времени в Tick — номер временного тика (каждые 2 миллисекунды). Следующие три столбца содержат уровень сигнала в единицах АЦП с оптопары энкодера, который подключен в соответствующий номеру разъем: PD1, PD2, PD3.

Задание

Используя данные сигналов, для каждой шестерни стенда нужно определить исправленную закодированную кодом Хемминга последовательность бит и исходное кодовое слово (декодированную последовательность).

Результат работы

6 последовательностей, по 2 для каждой шестерни:

- исправленная закодированная последовательность;
- исходная декодированная последовательность.

Критерий проверки

Точное соответствие заданному коду.

Оценка

Максимальная сумма 6 баллов:

- 1 балл за определение исправленной закодированной последовательности каждой шестеренки,
- 1 балл за определение декодированной последовательности каждой шестеренки.

Подзадачи

1. Получить данные со стенда ОВКС (Приложение 1).
2. Решить поставленную задачу.
3. Ввести решение в разработанную систему управления стендом.

Алгоритм решения

1. Найти на шестернях начало кодовой последовательности, которое отмечено точкой.
2. Определить минимальный элемент с наименьшим угловым размером.
3. Разделить шестерню на минимальные элементы и поставить им в соответствие «1» и «0». Найти количество минимальных элементов — длину закодированного сообщения.

4. Выписать последовательность из «1» и «0» по часовой стрелки.
5. Научиться работать со стендом ОВКС: включение, снятие данных.
6. Построить графики сигнала по полученным данным, проверить соответствие полученной последовательности по пикам графика, с учетом направления вращения шестеренок (по часовой или против часовой стрелки).
7. Определить есть ли ошибка в коде, если есть, то исправить ее, получив исправленную закодированную последовательность каждой шестеренки.
8. Удалить контрольные биты, расположенные на позициях, соответствующих степеням 2, получив декодированную последовательность каждой шестеренки.
9. Найденные последовательности ввести в систему управления стендом.

Задача III.2.2. Работа на стенде ОВКС. Подбор передач (6 баллов)

В данной задаче Вам потребуется задействовать дополнительный блок передачи, а также наборы передаточных шестерней.

Для начала, выставьте все блоки энкодеров и передач на позиции, как показано на рисунке III.2.1.

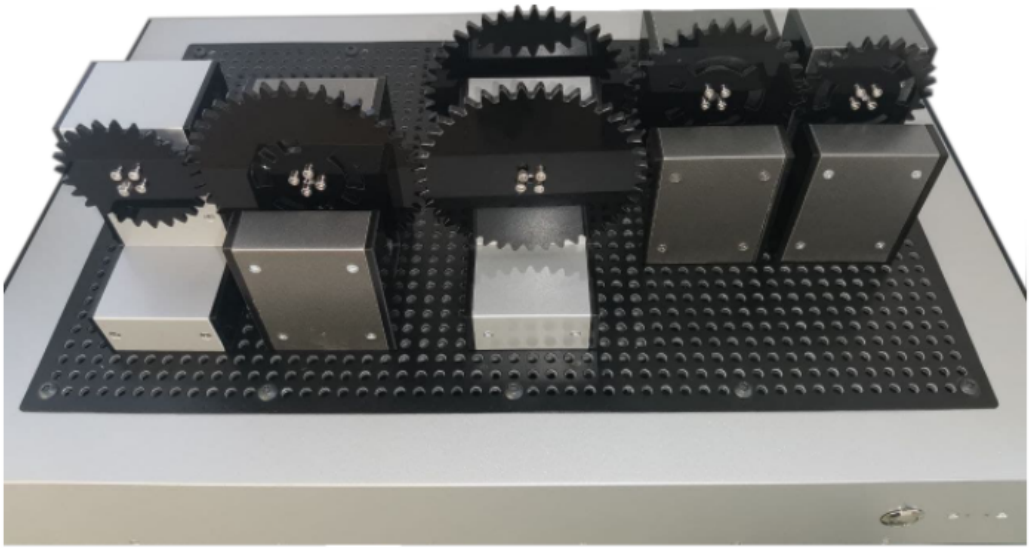

Рис. III.2.1: Стенд ОВКС

Задание

Установите на дополнительный блок передачи две шестерни (одна крепится спереди, другая сзади) так, чтобы период обращения малой шестерни в новой конфигурации совпал с периодом обращения средней шестерни в стандартной конфигурации, как в Задаче 1. После этого запустите систему и вычислите период обращения средней шестерни в новой конфигурации.

Результат работы

1 число — средний период вращения средней шестерни.

Критерий проверки

Точность определения периода вращения — 5 мс.

Оценка

6 баллов.

Обратите внимание! Система является физической, и у шестеренок имеется трение. Ответ полученный математически может оказаться недостаточно точным с учетом трения.

Подзадачи

1. Получить последовательность сигналов с трех каналов стенда.
2. По полученной последовательности сигналов со стенда определить период вращения каждой шестеренки.
3. Исходя из передаточного соотношения, определить размер шестерней, которые необходимо посадить на блок дополнительной передачи, чтобы получить период обращения малой шестерни в новой конфигурации, равной периоду обращения средней шестерни в стандартной конфигурации первой задачи.
4. Получить последовательность сигналов со средней шестерни в новой конфигурации.
5. По полученной последовательности сигнала со стенда определить период вращения средней шестерни.
6. Результат занести в систему управления стендом.

Решение

Для определения типа шестерней необходимо использовать передаточное число. Передаточное число — это отношение количества зубцов на вращающей шестерне к количеству зубцов вращаемой шестерни.

Составляя систему шестерней с учётом дополнительной передачи, необходимо учитывать, что шестерни, находящиеся на одном блоке передач, передают свои вращательные характеристики напрямую.

Составив системы, исходя из передаточного соотношения, можно определить скорость/период вращения малой шестерни в новой конфигурации и средней шестерни в старой конфигурации.

Решение задачи основано на том факте, что в результате равномерного вращения (с постоянной угловой скоростью) сигнал, модулируемый шестернями, будет периодически повторяться.

Для поиска периодичности в функции можно воспользоваться автокорреляционным подходом. В данном подходе определяется энергия $f(i)$ произведения исходной

функции $p(j)$ и сдвинутой $p(i+j)$ на некоторый интервал i.

$$f(i) = \sum_{j=1}^{N} p(j) \times p(i+j)$$

При сдвиге i — кратном периоду функции, сдвинутая функция совпадает с исходной, и энергия при таких сдвигах даст максимальные значения. На рисунке III.2.2 представлена автокорреляционная функция $f(i)$, рассчитанная для каждой шестерни.

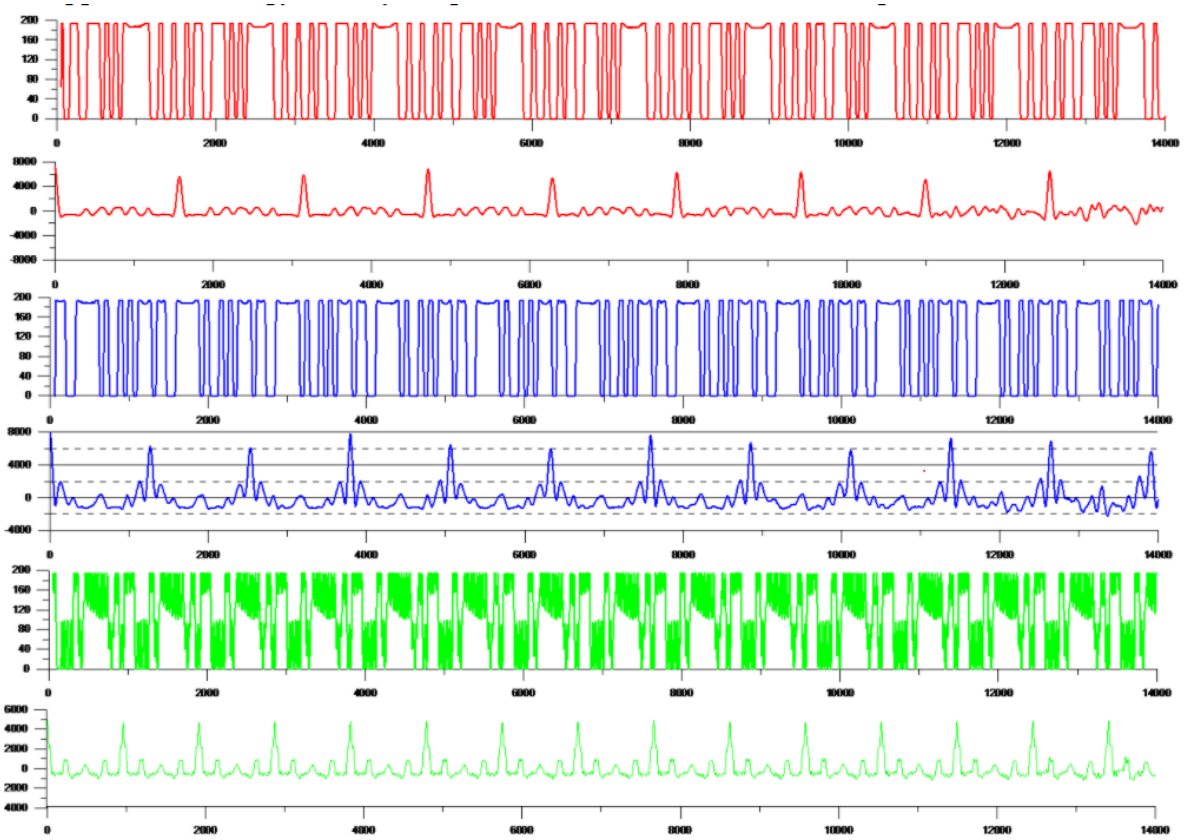

Рис. III.2.2: Вид автокорреляционной функции для трех шестерней

Далее, вычисляя расстояния между главными максимумами, полученных автокорреляционных функции $f(i)$, находим периоды вращения каждой шестерни.

После этого необходимо проделать аналогичную операцию с данными для новой конфигурации и вычислить период обращения средней шестерни.

Задача III.2.3. Работа на стенде ОВКС. Определение углового ускорения (6 баллов)

Определение углового ускорения моделирующих шестеренок в системе заданной конфигурации. В данной задаче Вам потребуется использовать конфигурацию шестерней из задачи 2 и скачать данные динамики мощности принимаемых ИК-сигналов, модулируемых вращающимися шестернями.

Формат записи данных в файл

4 столбца данных, разделенных пробелом. В первом столбце отсчет времени в Tick — номер временного тика (каждые 2 миллисекунды). Следующие три столбца содержат уровень сигнала в единицах АЦП с оптопары энкодера, который подключен в соответствующий номеру разъем: PD1, PD2, PD3.

Задание

По измеренной последовательности сигналов определить угловое ускорение вращения каждой шестерни. Угловое ускорение измеряется в градусах/сек^2.

Результат работы

3 числа (угловое ускорение вращения [градусы/сек^2]), по одному для каждой шестерни.

Критерий проверки

Точность определения углового ускорения 0,75 градусов/сек^2.

Оценка:

6 баллов, по 2 балла за каждую шестерню.

Подзадачи:

1. Выставить блоки на стенде так, как при решении задачи 2, с учетом дополнительного блока передач и выставленных на нем шестерней.
2. Получить последовательность сигналов с трех каналов стенда.
3. По полученной последовательности сигнала со стенда определить ускорение вращения трех шестерней.
4. Результаты занести в систему управления стендом.

Решение

На рисунке III.2.3 показан пример динамики амплитуды ИК — сигнала, прошедшего через равноускоренно вращающиеся шестерни. По полученной последовательности сигналов команды определяют период вращения каждой шестеренки, оценивают стабильность вращения.

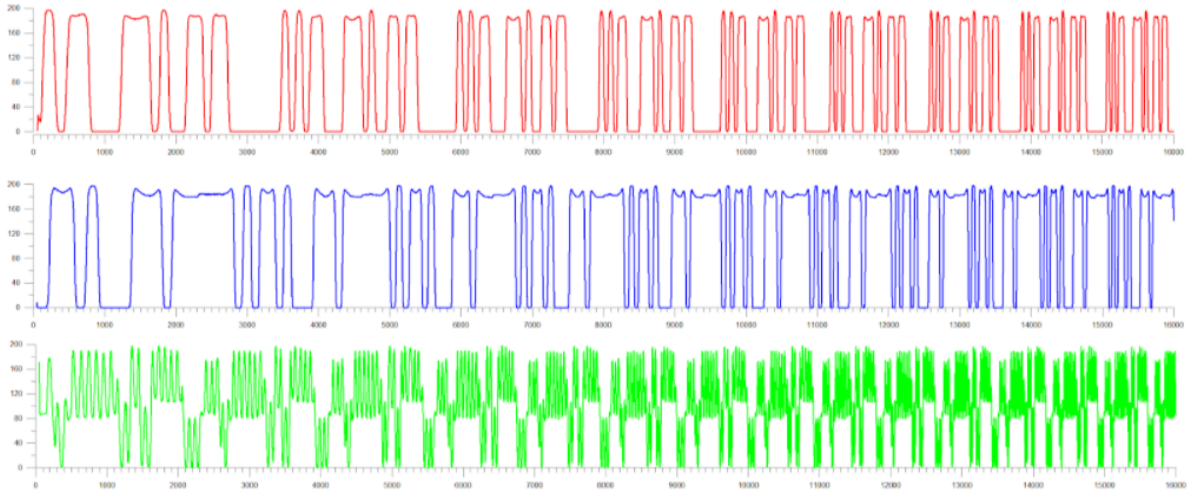

Рис. III.2.3: Динамика амплитуды ИК-сигнала, модулированного равноускорено вращающимися шестернями. Видно, что шестеренка наименьшего радиуса (зеленый график) модулирует сигнал, как прорезями/стенками, так и зубцами

Идея решения состоит в том, чтобы определить количество оборотов, совершенных каждой шестерней. Далее, оценив значение первого и последнего периодов, можно рассчитать величину углового ускорения.

Для этого используя формулу $a = dw/dt$ выразить скорость вращения через период по формуле $w = 360/T$.

Задача III.2.4. Работа на стенде УНКС. Передача данных по каналу связи (15 баллов)

«Спутник» движется по сложной траектории, проекцию которой вы видите при движении модели «Спутника» на стенде. По ИК — каналу связи нужно передать данные со «Спутника» на «Радар». В канале связи присутствуют шумы и помехи, вносящие ошибки в передаваемое сообщение.

Задание

Создать собственный протокол связи, позволяющий передавать данные с высокой точностью в условиях сильных помех.

Результат работы

3 программы: трекер, кодер, декодер.

Критерий проверки

Оценка оставшихся ошибок в принятом сообщении после его декодирования.

Оценка

Балл за задачу начисляется по формуле $y = 15 \cdot (x - k)/(100 - k)$, где k — значение точности передачи при запуске на демонстрационных программах, а x — процент, показывающий точность переданного файла при запуске на программах команды.

15 баллов — максимальное количество баллов при 100% точности передачи контрольного файла по каналу связи в условиях шумов.

0 баллов — точность передачи контрольного файла по каналу связи в условиях шумов, с использованием демонстрационных программ — трекера, кодера, декодера.

Ограничения

Время работы программ при передаче контрольного файла не должно превышать 5 минут.

Объем встроенного файла для тестирования на 20% меньше контрольного файла.

Результаты решения задачи 4 могут быть использованы при решении задач 5, 6.

Описание модели

Модель канала представляет собой передатчик, расположенный на «Спутнике», который передает цифровые данные. Присутствующий в канале шум искажает сигнал, случайно меняя некоторые биты в принятом сигнале.

Для того, чтобы иметь возможность исправить возникающие ошибки необходимо на стороне «Спутника» (передатчик) разработать программу — кодер, кодирующую передаваемое сообщение помехоустойчивым кодом (вкладка **«Кодировщик»**), на стороне «Радара» (приемник) разработать программу — декодер, который должен определить, в каких битах принятого сигнала произошло искажение и, по возможности, исправить поврежденные биты, декодируя сообщение (вкладка **«Декодировщик»**).

Следует заметить, что для устойчивого приема сигнала необходимо сопровождать «Радаром» «Спутник», обеспечивая устойчивый прием и минимальное искажение данных (чем точнее идет сопровождение, тем меньше ошибок возникает в канале). Для этого вы должны написать программу сопровождения «Спутника» «Радаром» (вкладка **«Программа слежения»**).

В этой задаче требуется создать алгоритм устойчивого сопровождения радаром спутника.

Радар воспринимает команды поворота вправо и влево со скоростями от 0 до 100, и позволяет считать состояние своих регистров. Эти данные храняться в длинном целом числе.

- В первых 12 битах этого числа (0..11) находится смещение спутника относительно оси визирования радара.
- В битах 12..15 находится информация о состоянии радара — в какую сторону он движется.
- В битах 16..19 находится информация о положении радара (крайнем левом и

крайнем правом).

- В битах 20..43 записано внутреннее время радара (в миллисекундах)
- В битах 44..52 находится специальный ключ, использующийся при шифрации по второму (специальному) каналу

Загрузка файла проводится следующим образом:

- Кнопка [+] позволяет выбрать файл с исходным кодом программы, который вы написали (примеры программ расположены в директории UserExamples).
- После выбора файла у вас появится несколько возможностей:
  < > позволяет просмотреть содержимое выбранного файла,
  Стрелочка вверх — загрузить его в систему,
  Стрелочка кольцом — скомпилировать его.

При необходимости, вы можете сохранять данные при работе программы слежения в специальный файл для последующего анализа.

Для тестирования точности приема-передачи у вас есть возможность загрузить тестовые данные, которые будут передаваться «Спутником» (вкладка **«Тестовые данные»**).

Для запуска всей системы (сопровождение «Радаром» «Спутника» и передача данных) используйте кнопку Start tracking вкладки **«Начать слежение»**.

Для завершения сопровождения «Радаром» «Спутника» и приема данных используйте кнопку Stop tracking вкладки **«Остановить слежение»**.

У вас есть возможность просмотра файла, в который вы писали, когда выполнялось сопровождение «Радаром» «Спутника» (вкладка **Журнал слежения**).

Рассчитать точность передачи данных можно, воспользовавшись вкладкой **Точность передачи**. Но только в тот запуск, когда вы не загружаете свои тестовые данные, а используете встроенные.

Решение

В данной задаче имеется битовый шум, который с некоторой вероятностью инвертует один бит в передаваемых данных. Однако наличие двукратной ошибки исключено в пределах отдельного блока передачи, размер которого можно было узнать в ходе анализа переданного файла, который можно скачать и проанализировать во вкладке **«Скачать дешифрованные данные»**.

Для того, чтобы исключить такие ошибки, достаточно использовать кодирование данных кодом Хэмминга. При выборе параметров кода Хэмминга необходимо учитывать частоту ошибок, что также выясняется при анализе.

Базовые программы на языке C++

«Tracker»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "client2server.h"
#include "militime.h"
```

```
6  #include <unistd.h>
7  #include <time.h>
8
9  int tracker(FILE* tracklog)
10 {
11         static double hist_dx=0.;
12         client2server* c2s;
13         c2s=new client2server;
14         char str[500];
15
16         int dx=0;
17         int new_dx;
18         while(1)
19         {
20                 status_type new_status;
21                 int new_state,new_position;
22                 unsigned char payload[4];
23                 long new_ticks;
24
25                 new_status=(status_type)c2s->getStatus();
26                 new_dx=new_status&0x0000000FFF;
27                 dx=(new_dx<2048)?new_dx:(new_dx%2048)-2048;
28
29                 int i;
30
31                 fprintf(tracklog,"%d\n",
32                                 (int)dx
33                                 );
34                 fflush(tracklog);
35
36                 hist_dx=dx;
37                 if(fabs((double)hist_dx)<500.0)
38                 {
39                         if(fabs(hist_dx) < 10)
40                         {
41                                 c2s->moveStop();
42                         }
43                         else
44                         {
45                         int speed;
46                         speed=4;
47                         speed=(speed>100)?100:speed;
48                                 if(hist_dx>0)
49                                 {
50                                         c2s->moveLeft(speed);
51                                 }
52                                 else
53                                 {
54                                         c2s->moveRight(speed);
55                                 }
56                         }
57                 }
58         }
59         delete (c2s);
60         return 0;
61 }
```

«Encoder»

```
1  #include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>

int encoder(FILE* r_fifo, FILE* w_fifo)
{
#define BLOCK_SIZE 1000
   char buf[BLOCK_SIZE];
   long s;
   s= fread( buf, sizeof(char), BLOCK_SIZE, r_fifo);
   for(;s>0;)
    {
     fwrite(buf, sizeof(char), s, w_fifo);
     s= fread(buf, sizeof(char), BLOCK_SIZE, r_fifo);
    }
 return 0;
}
```

«Decoder»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BLOCK_SIZE 1000

int decoder(FILE* r_fifo, FILE* w_fifo,FILE* tracklog)
{
   char buf[BLOCK_SIZE*2];
   long s;

   s= fread( buf, sizeof(char), BLOCK_SIZE*2, r_fifo);
   for(;s>0;)
    {
     if(s>1)
      fwrite(buf, sizeof(char), s, w_fifo);
     s= fread(buf, sizeof(char), BLOCK_SIZE*2, r_fifo);
    }
 return 0;
}
```

Задача III.2.5. Передача данных по каналу связи при краевых преградах между «Спутником» и «Радаром» (27 баллов)

«Спутник» движется по сложной траектории, проекцию которой вы видите при движении модели «Спутника» на стенде. По ИК — каналу связи нужно передать данные со «Спутника» на «Радар». В канале связи присутствуют шумы и помехи, вносящие ошибки в передаваемое сообщение. Между «Спутником» и «Радаром» существуют краевые преграды.

Задание

Создать собственный протокол связи, позволяющий передавать данные с высокой точностью в условиях сильных помех, с учетом краевых преград между «Спутником» и «Радаром».

Результат работы

3 программы: трекер, кодер, декодер.

Критерий проверки

Оценка оставшихся ошибок в принятом сообщении после его декодирования.

Оценка

Балл за задачу начисляется по формуле $y = 27 \cdot (x - k)/(100 - k)$, где k — значение точности передачи при запуске на демонстрационных программах, а x — процент, показывающий точность переданного файла при запуске на программах команды.

27 баллов 100% точность передачи контрольного файла по каналу связи в условиях шумов.

0 баллов — точность передачи контрольного файла по каналу связи в условиях шумов, с использованием демонстрационных программ — трекера, кодера, декодера.

Ограничения

Время работы программ при передаче контрольного файла не должно превышать 8 минут.

Объем встроенного файла для тестирования на 20% меньше контрольного файла.

Результаты решения задачи 5 могут быть использованы при решении задачи 6.

Описание модели

Модель канала представляет собой передатчик, расположенный на «Спутнике», который передает цифровые данные. Присутствующий в канале шум искажает сигнал, случайно меняя некоторые биты в принятом сигнале.

Для того, чтобы иметь возможность исправить возникающие ошибки необходимо на стороне «Спутника» (передатчик) разработать программу — кодер, кодирующую передаваемое сообщение помехоустойчивым кодом (вкладка **«Кодировщик»**), на стороне «Радара» (приемник) разработать программу — декодер, который должен определить, в каких битах принятого сигнала произошло искажение и, по возможности, исправить поврежденные биты, декодируя сообщение (вкладка **«Декодировщик»**).

Следует заметить, что для устойчивого приема сигнала необходимо сопровождать «Радаром» «Спутник», обеспечивая устойчивый прием и минимальное искажение данных (чем точнее идет сопровождение, тем меньше ошибок возникает в канале). Для этого вы должны написать программу сопровождения «Спутника» «Радаром» (вкладка **«Программа слежения»**)

В этой задаче требуется создать алгоритм устойчивого сопровождения радаром спутника.

Радар воспринимает команды поворота вправо и влево со скоростями от 0 до 100, и позволяет считать состояние своих регистров. Эти данные хранятся в длинном целом числе.

- В первых 12 битах этого числа (0..11) находится смещение спутника относительно оси визирования радара.
- В битах 12..15 находится информация о состоянии радара — в какую сторону он движется.
- В битах 16..19 находится информация о положении радара (крайнем левом и крайнем правом).
- В битах 20..43 записано внутреннее время радара (в миллисекундах)
- В битах 44..52 находится специальный ключ, использующийся при шифрации по второму (специальному) каналу.

Загрузка файла проводится следующим образом:

- Кнопка [+] позволяет выбрать файл с исходным кодом программы, который вы написали (примеры программ расположены в директории UserExamples).
- После выбора файла у вас появится несколько возможностей:
  < > позволяет просмотреть содержимое выбранного файла,
  Стрелочка вверх — загрузить его в систему,
  Стрелочка кольцом — скомпилировать его.

При необходимости, вы можете сохранять данные при работе программы слежения в специальный файл для последующего анализа.

Для тестирования точности приема-передачи у вас есть возможность загрузить тестовые данные, которые будут передаваться «Спутником» (вкладка **«Тестовые данные»**).

Для запуска всей системы (сопровождение «Радаром» «Спутника» и передача данных) используйте кнопку Start tracking вкладки **«Начать слежение»**.

Для завершения сопровождения «Радаром» «Спутника» и приема данных используйте кнопку Stop tracking вкладки **«Остановить слежение»**.

У вас есть возможность просмотра файла, в который вы писали, когда выполнялось сопровождение «Радаром» «Спутника» (вкладка **Журнал слежения**).

Рассчитать точность передачи данных можно, воспользовавшись вкладкой **Точность передачи**. Но только в тот запуск, когда вы не загружаете свои тестовые данные, а используете встроенные.

Решение

Для преодоления краевых преград со стороны слежения необходимо добавить программе «Tracker» функцию продолжения слежения после потери видимости «спутника» из-за преград.

В условиях блоковой потери данных из-за прегражденния канала передачи краевыми преградами, необходимо добавить программам «Encoder» и «Decoder» функцию блоковой передачи. То есть помимо кодирования данных кодом Хэмминга необходимо делить полученный код на отдельные пронумерованные блоки. После этого можно передать данные несколько раз, что позволит при декодировании добавить в ответную строку блоки, которые были потеряны в результате перекрытия канала

передачи данных краевыми преградами.

Также необходимо учитывать ограничение по времени передачи, что приводит к необходимости обдуманно подбирать длину нумеруемых блоков.

Базовые программы на языке C++

«Tracker»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "client2server.h"
#include "militime.h"
#include <unistd.h>
#include <time.h>


int tracker(FILE* tracklog)
{
        static double hist_dx=0.;
        client2server* c2s;
        c2s=new client2server;
        char str[500];


        int dx=0;
        int new_dx;
        while(1)
        {
                    status_type new_status;
                    int new_state,new_position;
                    unsigned char payload[4];
                    long new_ticks;

                        new_status=(status_type)c2s->getStatus();
                        new_dx=new_status&0x0000000FFF;
                        dx=(new_dx<2048)?new_dx:(new_dx%2048)-2048;

                        int i;


                        fprintf(tracklog,"%d\n",
                                        (int)dx
                                        );
                        fflush(tracklog);

                        hist_dx=dx;
                if(fabs((double)hist_dx)<500.0)
                {
                        if(fabs(hist_dx) < 10)
                        {
                                c2s->moveStop();
                        }
                        else
                        {
                        int speed;
                        speed=4;
                        speed=(speed>100)?100:speed;
```

```
51                                  if(hist_dx>0)
52                                  {
53                                          c2s->moveLeft(speed);
54                                  }
55                                  else
56                                  {
57                                          c2s->moveRight(speed);
58                                  }
59                          }
60                  }
61          }
62          delete (c2s);
63          return 0;
64  }
```

«Encoder»

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4
5   int encoder(FILE* r_fifo, FILE* w_fifo)
6   {
7   #define BLOCK_SIZE 1000
8      char buf[BLOCK_SIZE];
9      long s;
10     s= fread( buf, sizeof(char), BLOCK_SIZE, r_fifo);
11     for(;s>0;)
12      {
13       fwrite(buf, sizeof(char), s, w_fifo);
14       s= fread(buf, sizeof(char), BLOCK_SIZE, r_fifo);
15      }
16   return 0;
17  }
```

«Decoder»

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #define BLOCK_SIZE 1000
5
6   int decoder(FILE* r_fifo, FILE* w_fifo,FILE* tracklog)
7   {
8      char buf[BLOCK_SIZE*2];
9      long s;
10
11     s= fread( buf, sizeof(char), BLOCK_SIZE*2, r_fifo);
12     for(;s>0;)
13      {
14       if(s>1)
15        fwrite(buf, sizeof(char), s, w_fifo);
16       s= fread(buf, sizeof(char), BLOCK_SIZE*2, r_fifo);
17      }
18   return 0;
19  }
```

Задача III.2.6. Передача данных по каналу связи при центральной преграде между «Спутником» и «Радаром» (40 баллов)

«Спутник» движется по сложной траектории, проекцию которой вы видите при движении модели «Спутника» на стенде. По ИК — каналу связи нужно передать данные со «Спутника» на «Радар». В канале связи присутствуют шумы и помехи, вносящие ошибки в передаваемое сообщение. Между «Спутником» и «Радаром» существует центральная преграда.

Задание

Создать собственный протокол связи, позволяющий передавать данные с высокой точностью в условиях сильных помех, с учетом центральной преграды между «Спутником» и «Радаром».

Результат работы

3 программы: трекер, кодер, декодер.

Критерий проверки

Оценка оставшихся ошибок в принятом сообщении после его декодирования.

Оценка

Балл за задачу начисляется по формуле $y = 40 \cdot (x - k)/(100 - k)$, где k — значение точности передачи при запуске на демонстрационных программах, а x — процент, показывающий точность переданного файла при запуске на программах команды.

40 баллов — 100% точность передачи контрольного файла по каналу связи в условиях шумов.

0 баллов — точность передачи контрольного файла по каналу связи в условиях шумов, с использованием демонстрационных программ — трекера, кодера, декодера.

Ограничения

Время работы программ при передаче контрольного файла не должно превышать 8 минут.

Объем встроенного файла для тестирования на 20% меньше контрольного файла.

Описание модели

Модель канала представляет собой передатчик, расположенный на «Спутнике», который передает цифровые данные. Присутствующий в канале шум искажает сигнал, случайно меняя некоторые биты в принятом сигнале.

Для того, чтобы иметь возможность исправить возникающие ошибки необходимо на стороне «Спутника» (передатчик) разработать программу — кодер, кодирующую передаваемое сообщение помехоустойчивым кодом (вкладка **«Кодировщик»**), на стороне «Радара» (приемник) разработать программу — декодер, который должен определить, в каких битах принятого сигнала произошло искажение и, по возможности, исправить поврежденные биты, декодируя сообщение (вкладка **«Декодировщик»**).

Следует заметить, что для устойчивого приема сигнала необходимо сопровождать «Радаром» «Спутник», обеспечивая устойчивый прием и минимальное искажение данных (чем точнее идет сопровождение, тем меньше ошибок возникает в канале). Для этого вы должны написать программу сопровождения «Спутника» «Радаром» (вкладка **«Программа слежения»**)

В этой задаче требуется создать алгоритм устойчивого сопровождения радаром спутника.

Радар воспринимает команды поворота вправо и влево со скоростями от 0 до 100, и позволяет считать состояние своих регистров. Эти данные хранятся в длинном целом числе.

- В первых 12 битах этого числа (0..11) находится смещение спутника относительно оси визирования радара.
- В битах 12..15 находится информация о состоянии радара — в какую сторону он движется.
- В битах 16..19 находится информация о положении радара (крайнем левом и крайнем правом).
- В битах 20..43 записано внутреннее время радара (в миллисекундах).
- В битах 44..52 находится специальный ключ, использующийся при шифрации по второму (специальному) каналу.

Загрузка файла проводится следующим образом:

- Кнопка [+] позволяет выбрать файл с исходным кодом программы, который вы написали (примеры программ расположены в директории UserExamples).
- После выбора файла у вас появится несколько возможностей:
  $< >$ позволяет просмотреть содержимое выбранного файла,
  Стрелочка вверх — загрузить его в систему,
  Стрелочка кольцом — скомпилировать его.

При необходимости, вы можете сохранять данные при работе программы слежения в специальный файл для последующего анализа.

Для тестирования точности приема-передачи у вас есть возможность загрузить тестовые данные, которые будут передаваться «Спутником» (вкладка **«Тестовые данные»**).

Для запуска всей системы (сопровождение «Радаром» «Спутника» и передача данных) используйте кнопку Start tracking вкладки **«Начать слежение»**.

Для завершения сопровождения «Радаром» «Спутника» и приема данных используйте кнопку Stop tracking вкладки **«Остановить слежение»**.

У вас есть возможность просмотра файла, в который вы писали, когда выполнялось сопровождение «Радаром» «Спутника» (вкладка **Журнал слежения**).

Рассчитать точность передачи данных можно, воспользовавшись вкладкой **Точность передачи**. Но только в тот запуск, когда вы не загружаете свои тестовые данные, а используете встроенные.

Решение

Расположение преграды в центре отличает конфигурацию шестой задачи от пятой тем, что перекрыта немного большая часть обзора радара. Соответственно, для получения большего процента точности передачи данных необходимо большее число раз дублировать отдельные блоки. Однако при ограниченном времени передачи перед отправкой исходный файл необходимо сжать.

Если посмотреть передаваемый файл (для этого достаточно убрать преграды, запустить базовые «Encoder» и «Decoder», после чего скачать переданный файл кнопкой **«Скачать дешифрованные данные»**), то можно увидеть, что он представляет собой текст, состоящий из символов 0, 1 и переносов строки. Такой тип файла отлично сжимается более чем в 10 раз, например, алгоритмом RLE. Это позволяет многократно повторять передачу в условии ограниченного времени.

Базовые программы на языке C++

«Tracker»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "client2server.h"
#include "militime.h"
#include <unistd.h>
#include <time.h>


int tracker(FILE* tracklog)
{
        static double hist_dx=0.;
        client2server* c2s;
        c2s=new client2server;
        char str[500];


        int dx=0;
        int new_dx;
        while(1)
        {
                    status_type new_status;
                    int new_state,new_position;
                    unsigned char payload[4];
                    long new_ticks;

                        new_status=(status_type)c2s->getStatus();
                        new_dx=new_status&0x0000000FFF;
                        dx=(new_dx<2048)?new_dx:(new_dx%2048)-2048;

                        int i;

```

```
33
34                          fprintf(tracklog,"%d\n",
35                                          (int)dx
36                                          );
37                          fflush(tracklog);
38
39                          hist_dx=dx;
40                  if(fabs((double)hist_dx)<500.0)
41                  {
42                          if(fabs(hist_dx) < 10)
43                          {
44                                  c2s->moveStop();
45                          }
46                          else
47                          {
48                          int speed;
49                          speed=4;
50                          speed=(speed>100)?100:speed;
51                                  if(hist_dx>0)
52                                  {
53                                          c2s->moveLeft(speed);
54                                  }
55                                  else
56                                  {
57                                          c2s->moveRight(speed);
58                                  }
59                          }
60                  }
61          }
62          delete (c2s);
63          return 0;
64  }
```

«Encoder»

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4
5   int encoder(FILE* r_fifo, FILE* w_fifo)
6   {
7   #define BLOCK_SIZE 1000
8      char buf[BLOCK_SIZE];
9      long s;
10     s= fread( buf, sizeof(char), BLOCK_SIZE, r_fifo);
11     for(;s>0;)
12      {
13       fwrite(buf, sizeof(char), s, w_fifo);
14       s= fread(buf, sizeof(char), BLOCK_SIZE, r_fifo);
15      }
16   return 0;
17  }
```

«Decoder»

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
```

```
4  #define BLOCK_SIZE 1000
5
6  int decoder(FILE* r_fifo, FILE* w_fifo,FILE* tracklog)
7  {
8     char buf[BLOCK_SIZE*2];
9     long s;
10
11    s= fread( buf, sizeof(char), BLOCK_SIZE*2, r_fifo);
12    for(;s>0;)
13     {
14      if(s>1)
15       fwrite(buf, sizeof(char), s, w_fifo);
16      s= fread(buf, sizeof(char), BLOCK_SIZE*2, r_fifo);
17     }
18   return 0;
19 }
```

Приложение 1. Описание стендов профиля «Технологии беспроводной связи 2020 г.»

Задачи профиля запроектированы и выполнялись на стендах образовательного комплекса «Беспроводные технологии связи», разработанного компанией ИнСити-Лаб при финансовой поддержке Фонда содействия инновациям.

Комплекс включает в себя стенды «Оптомеханическая визуализация кодирования сигналов» (ОВКС), «Узконаправленные низкоэнергетические каналы связи» (УНКС).

Стенд «Оптомеханическая визуализация кодирования сигналов» (ОВКС) представляет собой имитатор некоторого механизма, расположенного на удаленном объекте. Стенд посвящен основным задачам, связанным с кодами, низкоуровневым представлением сигналов, корреляционным функциям, визуализации сигналов. Включает в себя основание, набор сменных блоков оптических энкодеров. Позволяет работать с задачами по информатике, теории кодирования, решению обратных задач.

Стенд «Узконаправленные низкоэнергетические каналы связи» (УНКС) — имитатор канала связи с удаленным объектом (макет системы «спутник — радар»). Стенд посвящен задачам, связанным с организацией спутниковых каналов связи, написанию алгоритмов слежения за спутником и управления радарным комплексом, работу с PID-регуляторами и библиотеками OpenCV, написанию программ кодеров и декодеров для работы в зашумленных каналах связи. Включает в себя основание, систему управления подвижным объектом — «спутник», систему слежения — «радар». Позволяет работать с задачами по информатике, теории кодирования, адаптивному управлению.

Принцип работы стенда ОВКС. В качестве механизма, расположенного на некотором удаленном объекте, был создан макет с системой шестеренок, приводимых в движение двигателем и оборудованный тремя оптопарами. Этот макет позволяет, при включении, регистрировать оптический сигнал, проходящий через прорези в шестеренках и записывать его в файл. Оптический сигнал считывается системой регистрации макета (на базе RaspberryPi и STM32L432) через равные промежутки времени. Сформированный таким образом файл представляет собой последовательность из «0» и «1» (двоичный код) с каждой шестерни макета и в дальнейшем передается на компьютер управления, для последующего анализа участниками трека.

Принцип работы стенда УНКС. Имитатор канала связи по легенде трека назван системой «радар-спутник» и состоит из подвижного элемента «спутник» и вращающегося вокруг вертикальной оси «радара». Элемент «спутник» может передвигаться вдоль рельсы (ось X) и имеет инфракрасный (ИК) передатчик, а также графический маркер. Управление осуществляется с помощью компьютера RaspberryPi, который получает необходимые для передачи данные от компьютера управления и передает их во время движения. Блок «Радар» оборудован компьютером RaspberryPi и цифровой видеокамерой, с помощью библиотеки OpenCV он распознает положение графического маркера «спутника» в поле зрения камеры и передает положение маркера на управляющий компьютер. Одновременно с этим ИК-приемник «радара» принимает поток данных, передаваемый «спутником».

На рисунке III.2.4 1 представлены основные части стенда:

1 — подвижная передающая каретка («спутник») стенда УНКС, с помощью

двигателя и приводного ремня может перемещаться вдоль оси X влево-вправо по программно-задаваемому закону движения. Имеет на борту ИК-передатчик, передающий, во время движения, данные телеметрии бортового механизма.

2 — подвижная приемная каретка («радар») стенда УНКС имеет ИК-приемник, для приема данных со «спутника», видеокамеру для отслеживания «спутника» и двигатель для вращения каретки вокруг вертикальной оси.

3 — стенд ОВКС, позволяет регистрировать в цифровом виде сигналы с оптических пар, установленных на каждой шестеренке и записывать их в файлы, для управления данной системой используется компьютер RaspberryPi и вспомогательный микропроцессор STM32L432.

4 — компьютер управления стендом, предоставляет участникам трека web-интерфейс к задачам, управляет всеми основными блоками стенда, а также позволяет создавать и загружать программное обеспечение для решения задач на языках C, C++, Python, Java.

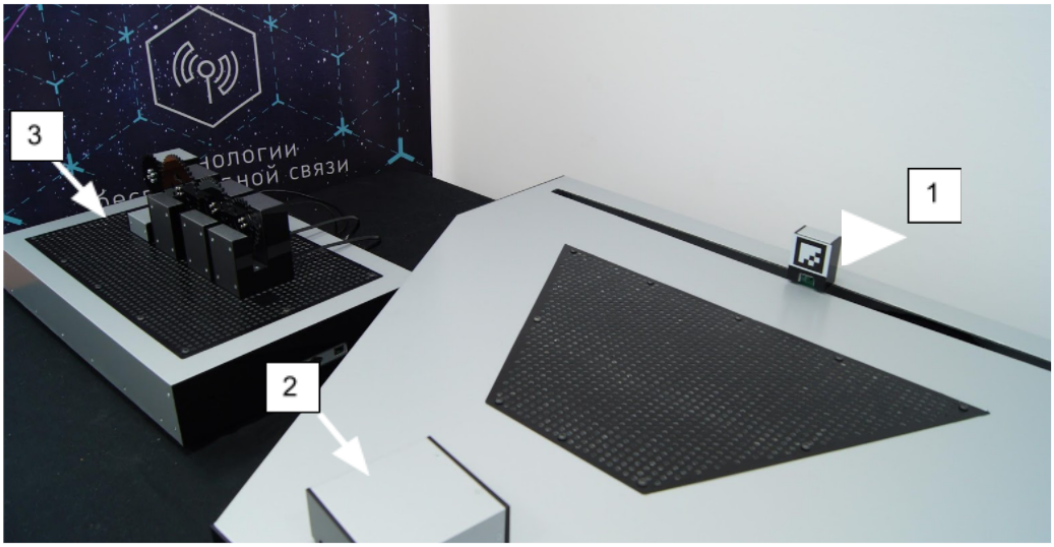

Рис. III.2.4: Общий вид стенда: 1 — подвижная передающая каретка «спутник» с графическим маркером стенда УВКС, 2— подвижная приемная каретка «радар» с видеокамерой стенда УВКС, 3 — стенд ОВКС