Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЕТ

по лабораторной работе №2

на тему

**ПАКЕТНАЯ ПЕРЕДАЧА ДАННЫХ**

|  |  |
| --- | --- |
| Выполнил студент группы № 050503  Преподаватель | Казак И. А  Марцинкевич В. А. |

Минск 2022

**1 КОД ПРОГРАММЫ**

**1.1 Пакет serial**

package serial

import (

"time"

"toks/serialDriver/netPackage"

"github.com/tarm/serial"

)

type Port struct {

Name string

Baund int

SerialPort \*serial.Port

}

func ReadByte(p Port) (int, byte, error) {

buf := make([]byte, 1)

var n int

var err error

go func() { n, err = p.SerialPort.Read(buf) }()

time.Sleep(100 \* time.Microsecond)

if err != nil {

return 0, 0, err

}

return n, buf[0], nil

}

func ReadPackage(p Port) (string, error) {

var newPackFlag, escFlag bool

var data []byte

for {

n, b, err := ReadByte(p)

if err != nil {

return "", err

}

if n == 0 {

break

}

// fmt.Println(string(b), newPackFlag, escFlag)

if b == byte(netPackage.Flag) && !escFlag && !newPackFlag {

newPackFlag = true

continue

}

if b == byte(netPackage.Esc) && !escFlag {

escFlag = true

continue

}

if b == byte(netPackage.Flag) && escFlag {

escFlag = false

}

data = append(data, b)

}

return string(data), nil

}

func WriteByte(p Port, text []byte) (int, error) {

n, err := p.SerialPort.Write(text)

if err != nil {

return 0, err

}

return n, nil

}

func WritePackage(p Port, address, text string) (int, error) {

n, err := p.SerialPort.Write(netPackage.CreatePackage(address, text))

if err != nil {

return 0, err

}

return n, nil

}

func InitPort(p \*Port) error {

c := &serial.Config{Name: p.Name, Baud: p.Baund}

temp, err := serial.OpenPort(c)

p.SerialPort = temp

return err

}

func Close(p Port) error {

if err := p.SerialPort.Close(); err != nil {

return err

}

return nil

}

func ChangeSpeed(p \*Port) error {

err := InitPort(p)

if err != nil {

return err

}

return nil

}

**1.2 Пакет main**

package main

import (

"bufio"

"fmt"

"log"

"os"

"toks/serialDriver/serial"

)

func main() {

portWrite := &serial.Port{Name: "/dev/ttys000", Baund: 9600}

portRead := &serial.Port{Name: "/dev/ttys023", Baund: 9600}

err := serial.InitPort(portRead)

if err != nil {

log.Fatal(err)

}

defer serial.Close(\*portRead)

err = serial.InitPort(portWrite)

if err != nil {

log.Fatal(err)

}

defer serial.Close(\*portWrite)

var data string

fmt.Print("Input data: ")

sc := bufio.NewScanner(os.Stdin)

sc.Scan()

data = sc.Text()

fmt.Println(serial.WritePackage(\*portWrite, "", data))

buf, err := serial.ReadPackage(\*portRead)

if err != nil {

log.Fatal(err)

}

fmt.Println("Read: ", buf)

}

**1.3 Пакет netPackage**

package netPackage

const (

Flag = 0x7E

Esc = 0x1B

)

type byteNetPackage struct {

flag byte

distAddr []byte

sourceAddr []byte

data []byte

}

type netPackage struct {

flag byte

distAddr string

sourceAddr string

data string

}

func CreatePackage(adress, text string) []byte {

pack := netPackage{Flag, adress, "", text}

return pack.convert().toByteOreder()

}

func (n \*netPackage) convert() byteNetPackage {

var pack byteNetPackage

pack.flag = n.flag

pack.distAddr = []byte(n.distAddr)

pack.sourceAddr = []byte(n.sourceAddr)

pack.data = []byte(n.data)

return pack

}

func (b byteNetPackage) toByteOreder() []byte {

var pack []byte

pack = append(pack, b.flag)

for \_, a := range b.distAddr {

pack = append(pack, a)

}

for \_, a := range b.sourceAddr {

pack = append(pack, a)

}

for \_, a := range byteStuffing(b.data) {

pack = append(pack, a)

}

return pack

}

func byteStuffing(data []byte) []byte {

var stuffedData []byte

for \_, b := range data {

if b == Flag || b == Esc {

stuffedData = append(stuffedData, Esc)

}

stuffedData = append(stuffedData, b)

}

return stuffedData

}

**2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

1. Разработать программный модуль реализации процедуры передачи (приема) пакета информации через последовательный интерфейс.
2. При формировании пакета использовать байт/бит-стаффинг.