Telekomunikacja - laboratorium	Studia stacjonarne - inzynierskie

Nazwa zadania	Protokół Xmodem				
Dzień	Wtorek	Godzina	12.15	Rok akademicki	2020/2021

Imię i Nazwisko	**Danel Malicki**
Imię i Nazwisko	**Maciej Włodarczyk**
Imię i Nazwisko	

Opis programu, rozwiązania problemu.

Program korzysta z protokołu Xmodem. Z pomocą programu można odbierać oraz wysyłać pliki. Program posiada algorytm CRC. Możliwe jest również działanie na sumie kontrolnej. Do kontroli portów i testowania przesyłania postanowiliśmy skorzystać z programów "Virtual Serial Port Tools" oraz "Tera Term". W przypadku wysyłania program czyta dane z pliku i rozdziela je na pakiety, które następnie przesyła w zależności od wybranego trybu (CRC/Parzystość bitów). Należy pamiętać o tym, że plik, który jest wysyłany jest "dopełniany" za pomocą znaku 32 z kodu ASCII do liczby pełnego pakietu. W przypadku odbioru odbieramy i sprawdzamy poprawność pakietów. Po zakończeniu transmisji odbiornik na sygnał EOT od nadajnika wysyła ACK kończąc połączenie.

Najważniejsze elementy kodu programu z opisem.

Funkcja, która służy do wysyłania danych. Początkowo dane czytane są z pliku i dzielone na pakiety. Następnie sprawdzamy, czy transmisja ustawiona jest na tryb CRC czy parzystości bitów. Po przesłaniu wszystkich pakietów program wysyła sygnał EOT (End Of Transmission), co kończy transmisję.

```
def send_data(readpath):
    databytes = read_file(readpath)
       returnetpackets = split_data(databytes)
     packet_number = 0
       initial_answer = ser.read()
    while initial_answer != C and initial_answer != NAK:
              initial_answer = ser.read()
              print(initial_answer)
              continue
     mode = initial_answer
       for bitpack in returnetpackets:
              ser.flush()
            send_packet(bitpack, packet_number, mode)
              packet_number += 1
       # Po zakończeniu transmisji wysyłamy sygnał End Of Transmission
       ser.write(EOT)
```

Funkcja odpowiadająca za wysłanie pojedynczego pakietu. Korzystamy z niej w funkcji, w której wysyłamy dane. Każdy przesyłany blok zaopatrywany jest w nagłówek SOH, numeru bloku i dopełnienia tego bloku do 255.

```
def send_packet(packet_to_send, numberp, mode):
        while True:
                header = bytearray()
                   header.append(int.from_bytes(SOH, 'big'))
                   header.append(numberp + 1)
                header.append(254 - numberp)
                   full = header + packet_to_send
                if mode == C:
                            full = full + crc16_mine(packet_to_send)
                   if mode == NAK:
                            full.append(checksuma(packet_to_send))
                   ser.write(full)
                print(full)
                   print(len(full))
                   ser.flush()
                answer = ser.read()
                   print(answer)
                   if answer == ACK:
                            break
                   if answer == CAN:
                            break
```

Funkcja, która służy do odbierania danych. Początkowo oznajmiamy, że jesteśmy gotowi na odbiór pliku. Następnie odbieramy dane do momentu końca transmisji, który sygnalizowany jest EOT.

```
def recive_data(savepath, mode=NAK):
   initial_recive = start_recive(mode)
         file_bytes = bytearray()
         while 1:
                     data_pack = recive_packet(mode, initial_recive)
                     if data_pack:
                                 file_bytes = file_bytes + data_pack
                     initial_recive = ser.read()
               if initial_recive == EOT:
                                 break
         print(file_bytes)
         f = open(savepath, "wb")
     f.write(file_bytes)
       f.close()
```

Funkcja odpowiadająca odebraniu pojedynczego pakietu. Korzystamy z niej w funkcji, w której odbieramy dane.

```
def recive_packet(mode, initial_recive):
       recived_header = bytearray()
       recived_header += initial_recive
       recived_header += ser.read()
       recived_header += ser.read()
       packet = recive_data_packet()
       if check_packet(mode, packet):
              return packet
       return False
```

Funkcja odpowiadająca za algorytm CRC.

```
def crc16(self, data: bytearray, poly=0x1021):
        crc = 0x0000
        for b in data:
            cur_byte = 0xFF & b
            for _ in range(0, 8):
                if (crc & 0x0001) ^ (cur_byte & 0x0001):
                    crc = (crc >> 1) ^ poly
                else:
                    crc >>= 1
                cur_byte >>= 1
        crc = (~crc & 0xFFFF)
        crc = (crc << 8) | ((crc >> 8) & 0xFF)
        return crc & 0xFFFF
```

Funkcja odpowiadająca za operację przy użyciu sumy kontrolnej.

```
def checksuma(self, data: bytearray):
    temp_sum = 0
    for byte in data:
        temp_sum += int(byte)
    return temp_sum % 256
```

Podsumowanie wnioski.

Napisany przez nas program komunikuje się z programem zewnętrznym Tera Term. Zarówno operacja wysłania jak i odebrania pliku zachodzi pomyślnie.
Program ten był powszechnie używany w przeszłości i spełnia swoje zadanie w przesyłaniu prostych plików.