CS 252 - Lab 03 Tele-Communication System Design

Report

Group Details

RollNo	Member
160050082	Avinash
160050095	Sai Kalyan
160050102	Chaitanya
160050104	Jagadeep

Title

Tele-communication System Design

The principle we used to transmit messages is transfferring the signal in form of high and low voltages between two raspberrypis.

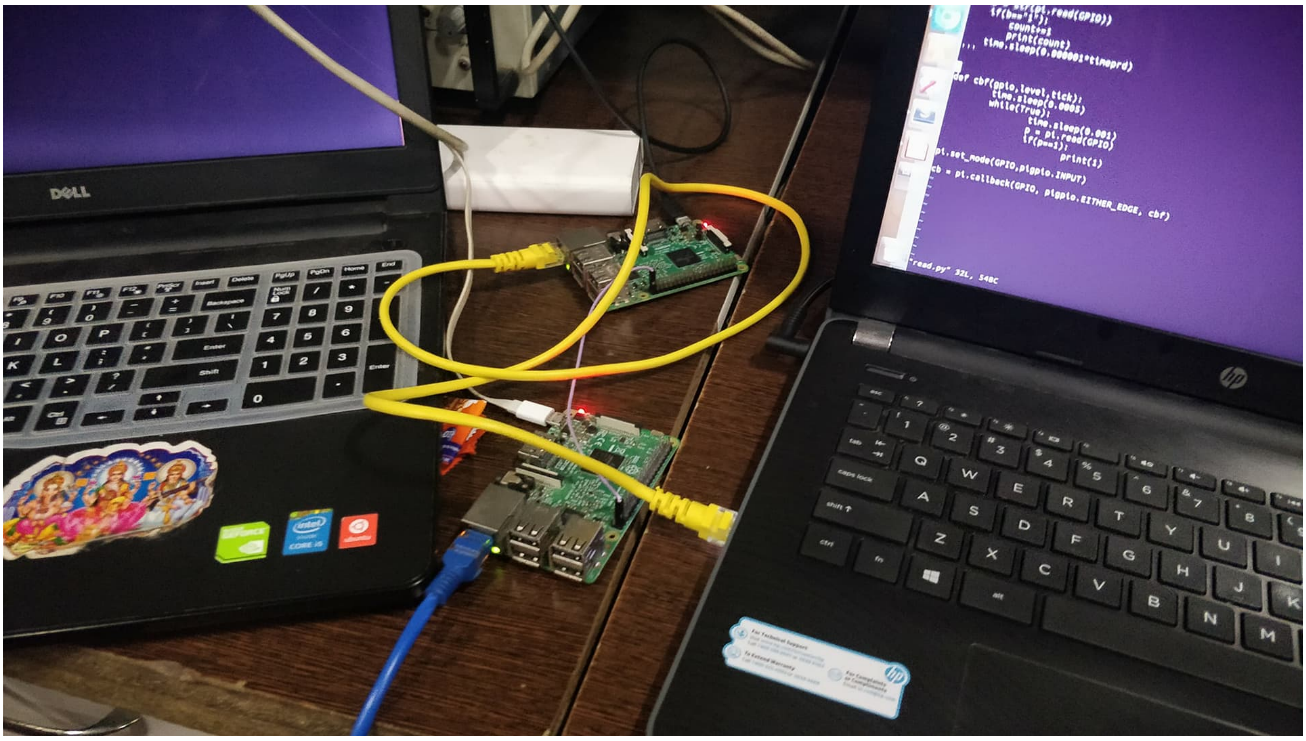
Design

The User gives input to the Rasp-berry pi through a computer connected to its command line through secure shell.To indicate which bits to be flipped the user adds letter 'e' just afer that bit.We used hamming encoding/decoding. The reciever checks the message with help of redundant bits and corrects the 1 bit error,and detects the 2bit error asking for retransmission(NACK).we also added start flag and end flag.

Implementation

Sender end

The user gives input to the raspberry pi command line. The code computes redundant bits using **hamming encoding**.The final message consists of redundant bits and payload and start/end flags.These are sent in form of high and low voltages through GPIO pins.After it sends message it starts listening for ACK,If it recieves a NACK(0) it retransmits the message and process is repeated else sending one message is complete



```
    str(pi.read(GPIO))
    if(b=="1"):
        count+=1
        print(count)
''' time.sleep(0.000001*timeprd)

def cbf(gpio,level,tick):
        time.sleep(0.0005)
        while(True):
                time.sleep(0.001)
                p = pi.read(GPIO)
                if(p==1):
                        print(1)
pi.set_mode(GPIO,pigpio.INPUT)
cb = pi.callback(GPIO, pigpio.EITHER_EDGE, cbf)
"read.py" 32L, 548C
```

Figure 1: Connection of two raspberry pis

Reciever end

After the message is recieved,it reads the bits and error check occurs through hamming decoding .If there is an error of 2 bits NACK is sent otherwise errors of 1 bit are automatically corrected.All the signals of gpio pins are handled using pigpio library

Hamming code

Hamming codes are a class of binary codes. If the length of the code is n = 2^r - 1 then the length of the message will be 2^r - r - 1 and the number of parity bits will be r. The position of the parity bits are 2^0, 2^1, 2^3,...2^{r-1}. The method of finding the value of parity bit at the position 2^k is as follows :
1) Considers all the bits in the respective positions which contain '1' in their k^{th} least significant bit(the positions are in the binary form).
2) Add all the bits in those positions modulo 2(i.e. add as normal decimal numbers and divide by 2 at the end). 3) The remainder is the required value of the parity bit.
The method of recovering the bits from the parity bits is as follows:
1) Collect all the bits with 1 as their position's 1st least significant bit and add them modulo 2. Continue this for 2nd least significant,3rd,... 2) Now put all the bytes next to each other from left to right in the decreasing order of their significance. 3) Convert this value into decimal number and this is the required position of the error bit.
If the code length is between 2^r-1 and 2^{r-1}, then there will be r parity bits at 1,2,4,...,2^{r-1}. 1 bit can be detected and corrected using these many parity bits. But if an additional bit is added at the end, then 2 bits can be detected and 1 bit can be corrected. The additional parity bit is the even parity of the previously encoded string of bits.

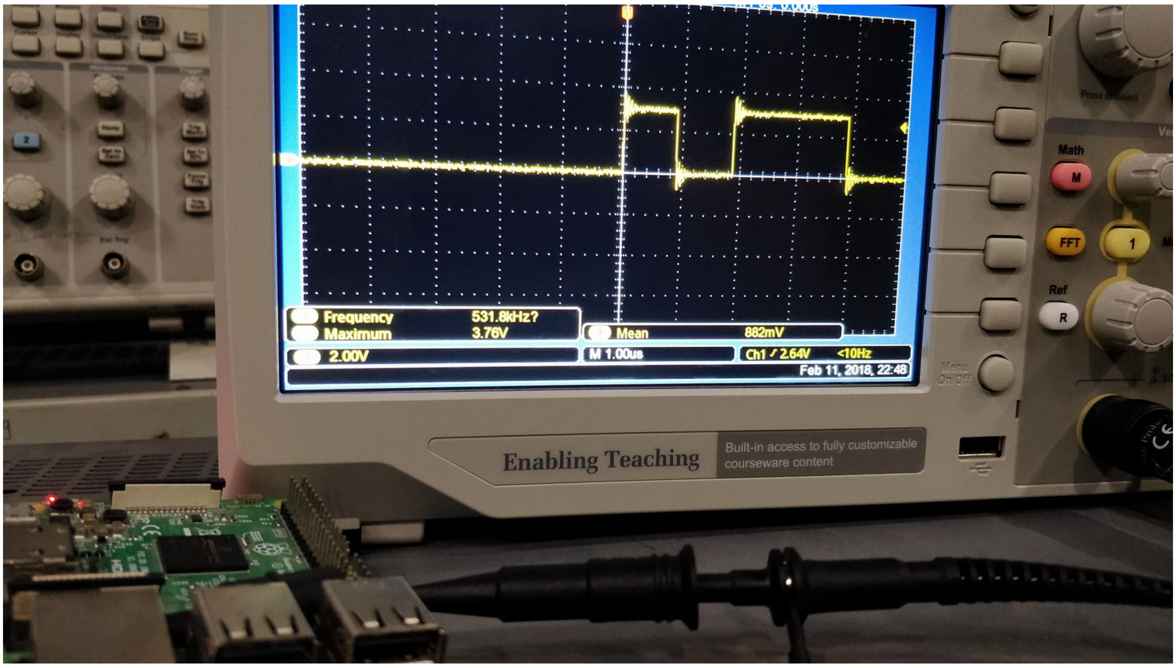


Figure 2: Checking input generation through oscilloscope

References

Hamming code - GeeksforGeeks
RaspberryPi functioning - RaspberryPi official site
pigpio package - Pigpio library