#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <termios.h>

#include <errno.h>

#include <string.h>

int UART\_Open(int fd,char \*port)

{

fd = open(port,O\_RDWR|O\_NOCTTY|O\_NDELAY);

if(fd == -1)

{

perror("Can't Open Serial Port\n");

return -1;

}

if(fcntl(fd, F\_SETFL, 0) < 0)

{

perror("fcntl failed!\n");

return -1;

}

return fd;

}

void UART\_Close(int fd)

{

close(fd);

}

int UART\_Set(int fd,int speed,int flow\_ctrl,int databits,int stopbits,char parity)

{

int i;

int status;

int speed\_arr[] = {B115200,B19200,B9600,B4800,B2400,B1200,B300};

int name\_arr[] = {115200,19200,9600,4800,2400,1200,300};

struct termios opt;

if(tcgetattr(fd,&opt) != 0)

{

perror("Get SerialAttr Error!\n");

return -1;

}

for(i=0;i<sizeof(speed\_arr)/sizeof(int); i++)

{

if(speed == name\_arr[i])

{

cfsetispeed(&opt,speed\_arr[i]);

cfsetospeed(&opt,speed\_arr[i]);

}

}

opt.c\_iflag &= ~(BRKINT | ICRNL | INPCK | ISTRIP | IXON);

opt.c\_cflag |= CLOCAL | CREAD;

switch(flow\_ctrl)

{

case 0 :

opt.c\_cflag &= ~CRTSCTS;

break;

case 1 :

opt.c\_cflag |= CRTSCTS;

break;

case 2 :

opt.c\_cflag |= IXON | IXOFF | IXANY;

break;

default:

return -1;

}

opt.c\_cflag &= ~CSIZE;

switch (databits)

{

case 5:

opt.c\_cflag |= CS5;

break;

case 6:

opt.c\_cflag |= CS6;

break;

case 7:

opt.c\_cflag |= CS7;

break;

case 8:

opt.c\_cflag |= CS8;

break;

default:

return -1;

}

switch (parity)

{

case 'n':

case 'N':

opt.c\_cflag &= ~PARENB;

opt.c\_iflag &= ~INPCK;

break;

case 'o':

case 'O':

opt.c\_cflag |= (PARODD | PARENB);

opt.c\_iflag |= INPCK;

break;

case 'e':

case 'E':

opt.c\_cflag |= PARENB;

opt.c\_cflag &= ~PARODD;

opt.c\_iflag |= INPCK;

break;

case 's':

case 'S':

opt.c\_cflag &= ~PARENB;

opt.c\_cflag &= ~CSTOPB;

break;

default:

return -1;

}

switch (stopbits)

{

case 1:

opt.c\_cflag &= ~CSTOPB;

break;

case 2:

opt.c\_cflag |= CSTOPB;

break;

default:

return -1;

}

opt.c\_oflag &= ~OPOST;

opt.c\_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);

opt.c\_cc[VTIME] = 0;

opt.c\_cc[VMIN] = 0;

tcflush(fd,TCIFLUSH);

if (tcsetattr(fd,TCSANOW,&opt) != 0)

{

perror("Set Serial Error!\n");

return -1;

}

return 0;

}

int UART\_Recv(int fd, char \*rcv\_buf,int data\_len)

{

int len,fs\_sel;

fd\_set fs\_read;

FD\_ZERO(&fs\_read);

FD\_SET(fd,&fs\_read);

len = read(fd,rcv\_buf,data\_len);

if(len > 0)

{

// printf("Data Received, Length = %d\n",len);

return len;

}

else

{

return 0;

}

}

int UART\_Send(int fd, char \*send\_buf,int data\_len)

{

int len = 0;

len = write(fd,send\_buf,data\_len);

if (len == data\_len )

{

return len;

}

else

{

tcflush(fd,TCOFLUSH);

return -1;

}

}

int main(int argc,char \*\*argv)

{

int len;

int fd1;

int fd2;

float Temp = 0.0;

float Humi = 0.0;

char Uart2\_buf[50];

char Uart2\_tep[50];

char TempString[5] = "Test";

char HumiString[5] = "Test";

fd2 = UART\_Open(fd2,"/dev/ttymxc2");

UART\_Set(fd2,115200,0,8,1,'N');

printf("Start Process...\n");

while(1)

{

len = UART\_Recv(fd2,Uart2\_buf,50);

if (len > 0)

{

Uart2\_buf[len] = '\0';

strcpy(Uart2\_tep,Uart2\_buf);

strncpy(TempString,&Uart2\_tep[13],5);

strncpy(HumiString,&Uart2\_tep[29],5);

TempString[4] = '\0';

HumiString[4] = '\0';

Temp = atof(TempString);

Humi = atof(HumiString);

printf("%f %f\n",Temp,Humi);

}

sleep(1);

}

UART\_Close(fd2);

}