תיעוד לפרויקט שליחת ping בין שני boardים

קבצים חיצוניים/Includes

נתחיל להסביר על הספריות/קבצים חיצוניים בהם השתמשנו בפרויקט, ועל התפקיד של כל אחת מהם:

שם הקובץ	תפקיד/שימוש עיקרי
lwip.h	מכיל פונקציות עזר לשליחת פקטת ip
lwip/icmp.h	מכיל פונקציות עזר עבור יצירה, קבלה ושליחה של פקטת icmp
lwip/inet_chksum.h	חישוב checksum לפקטות ip, icmp
stdlib.h	הקצאות והעתקת זיכרון
lwip/etharp.h	שליחה וקבלה של פקטות arp

השתמשנו ב middleware בשם lwip(הגענו אליה דרך התיקייה repositories שמותקנת עם stm32cubeide). זוהי ספרייה שמכילה הרבה קבצים שמטרתם לעזור לנו לבסס תקשורת ethernet על ידי פונקציות מוכנות שתפקידן הוא: להגדיר את כתובות הip והmac, הקצאות והעתקות זיכרון לבאפרים שיכילו את הפקטות אותן נרצה לשלוח ולקבל, להגדיר את הheaderים לפקטות arp,icmp,ip, עם אפשרות לקנפג את הפרמטרים של כל פקטה, פונקציות לחישוב checksum, פונקציות שליחה וקבלה של פקטות ועוד. בהמשך ניכנס לפונקציות בהן השתמשנו ונרחיב עליהם.

מטרת הפרוייקט וצעדים לביצוע

מטרת הפרוייקט היא לשלוח פקטת icmp בין שתי boardים מסוג stm32h745 nucleo. על מנת לעשות את זה, נידרש לבצע כמה צעדים בדרך:

- להגדיר לboardים שלנו כתובות ip, mac.
- לאפשר שימוש בממשק הפיזי lan8742(מובנה בboard), בממשק של השכבה השנייה eth, ובממשק lwip לשכבת ip.
- לקנפג את הlwip עבור שליחה וקבלה של פקטת icmp.

- ליצור פקטות של ip header, icmp header, לקנפג אותן כך שיתאימו לדרישות שלנו, לשרשר אותן, ולשלוח אותן על גבי ממשק הeth.

הסבר וניתוח של חלקים מרכזיים בקוד

הפונקציה send_icmp_ping:

קלט: `struct netif *netif` – מצביע לטיפוס מסוג ממשק רשת

פלט: אין.

מטרת הפונקציה: בנייה ושליחה של פקטת icmp על גבי ממשק eth.

```
struct pbuf *p;
struct pbuf *q;
```

הגדרה של packet buffers, p הפקטה לתוכה נכניס את פקטת הip, q הפקטה לתוכה נכניס את פקטת הicmp.

```
struct icmp_echo_hdr *icmphdr;
struct ip_hdr *iphdr;
```

הגדרה של טיפוסים ששדותיהם הם פרמטרים לפקטות הip, icmp. נקנפג אותם בהמשך.

```
ip_addr_t src_ip, dest_ip;

  // Set the source and destination IP addresses
IP4_ADDR(&src_ip, 192, 168, 0, 123);
IP4_ADDR(&dest_ip, 192, 168, 0, 156);
```

הגדרה של כתובות ה ip והזנת ערכים.

```
q = pbuf_alloc(PBUF_IP, sizeof(struct icmp_echo_hdr), PBUF_RAM);
p = pbuf_alloc(PBUF_IP, sizeof(struct icmp_echo_hdr) + IP_HLEN,
                    PBUF_RAM);
  //r = pbuf_alloc(PBUF_RAW_TX, SIZEOF_ETHARP_HDR, PBUF_RAM);
  iphdr = (struct ip_hdr *)p->payload;
 icmphdr = (struct icmp_echo_hdr *)q->payload;
```

הקצאת זיכרון לבאפרים שהגדרנו על ידי `pbuf_alloc`, הפונקציה מוגדרת בקובץ pbuf.h שנמצא בlwip middleware. לאחר מכן מגדירים שהבאפרים יצביעו על הheaderים שנכין בהמשך.

```
icmphdr->type = 8;
icmphdr->code = 0;
icmphdr->id = 0x5555;
icmphdr->seqno = 1;
icmphdr->chksum = 0;
icmphdr->chksum = inet_chksum(icmphdr, sizeof(struct icmp_echo_hdr));
```

קונפיגורציה של שדות פקטת הicmp:

- type = 8 – מציין שסוג פקטת icmp הוא echo(ping)
- Code = 0 – מספק מידע נוסף על סוג ההודעה, במקרה שלנו לא נזקק.
- Id = 0x5555 – מספר מזהה של ההודעה, שרירותי

- Seqno = 1 מציין שזה הping הראשון שנשלח.
- Checksum – מחושב על ידי הפונקציה inet_chksum על המידע בפקטת הicmp. הפונקציה inet_chksum לקוחה מתוך הקובץ inet_chksum.h שנמצאת בlwip middleware.

```
iphdr->_ttl = 255;
iphdr->src.addr = src_ip.addr;
iphdr->dest.addr = dest_ip.addr;
iphdr->_tos = 0;
iphdr->_offset = 0;
iphdr->_proto = 1;
iphdr->_v_hl = 0x45;
iphdr->_id = 0xABCD;
iphdr->_len = htons(46);
iphdr->_chksum = 0;
iphdr->_chksum = inet_chksum(iphdr, IP_HLEN);
```

<u>קונפיגורציה של שדות פקטת הip:</u>

- time to live – ttl = 255
- השמה של כתובות המקור והיעד של הפקטה.
- tos = 0 – אין שירות ספציפי שהפקטה נותנת.
- Proto = 1 – סוג הפרוטוקול של המידע שמופיע לאחר הheader, במקרה שלנו icmp.
- V_hl – שדה שמחולק לשני חלקים, חלק ראשון זה מספר גרסה של פקטת הip, במקרה שלנו משתמשים בipv4, ולכן הערך של הניבל העליון הוא 4, החלק השני מציין את מספר המילים(32 בתים) שיש בהודעה, במקרה שלנו מדובר בip header סטנדרטי ולכן הערך הוא 5.
- Id = 0xabcd – מספר זהות של ההודעה, נקבע שרירותית.
- Len = htons(46) – האורך הכולל של ההודעה, header + payload. במקרה שלנו מדובר בפקטת ping ולכן האורך הכולל הוא 46.
- Checksum – מחושב על ידי הפונקציה inet_chksum על המידע בפקטת הip. הפונקציה inet_chksum לקוחה מתוך הקובץ inet_chksum.h שנמצאת בlwip middleware.

```
SMEMCPY((u8_t *)p->payload + (IP_HLEN), (u8_t *)q->payload,
                                sizeof(struct icmp_echo_hdr))
```

השתמשנו בפונקציה SMEMCPY על מנת להעתיק את המידע של פקטת הicmp לסוף הheader של הip.

```
struct eth_addr mac_src;
struct eth_addr mac_dst;  // Destination MAC address (example)
uint8_t mac_dst_addr[6] = {0x06, 0x06, 0x06, 0x06, 0x06, 0x05};
memcpy(mac_src.addr, netif->hwaddr, 6);
memcpy(mac_dst.addr, mac_dst_addr, 6);
```

נגדיר טיפוסים שיכילו את הכתובות MAC זאת בשביל לציין את הכתובות שמהן ואליהן אנחנו שולחים את הפקטה. במקרה שלנו כתובת הMAC של הboard המקבל היא 06-06-06-06-06-05 שזאת הכתובת שקבענו ידנית.

```
if(ethernet_output(netif, p, &mac_src, &mac_dst, eth_type) !=
ERR_OK){
pbuf_free(q);
pbuf_free(p);
return;}
```

נשלח את הפקטה שבנינו באמצעות הפונקציה ethernet_output שמוגדרת בקובץ ethernet.h תחת lwip middleware. אפשר לראות שהפרמטרים לפונקציה הם ממשק הרשת שלנו שמכיל את כתובת הip של היעד, הבאפר שמכיל את המידע של הפקטה, כתובות הMAC של המקור והיעד, וסוג ההודעה, אצלנו מוגדר כETHTYPE_IP, שזה מציין שהמידע שמופיע בpayload של פקטת הethernet זה בפרוטוקול ipv4.

הפונקציה main:

```
ip_addr_t src_ip, dest_ip;
struct netif *netif;
IP4_ADDR(&src_ip, 192, 168, 0, 123);
IP4_ADDR(&dest_ip, 192, 168, 0, 156);
netif = ip4_route_src(&src_ip, &dest_ip);
```

הגדרה של כתובות ip וממשק רשת.

```
netif = ip4_route_src(&src_ip, &dest_ip);
etharp_request(netif, &dest_ip);
HAL_Delay(2000);
MX_LWIP_Process();
send_icmp_ping(netif);
HAL_Delay(2000);
MX_LWIP_Process();
```

זהו הקוד שנמצא בתוך הלולאה הראשית. נשלח בקשת arp ונחכה לתשובה שתתקבל דרך הפונקציה MX_LWIP_Process, לאחר מכן נקרא לפונקציה ששולחת פקטת icmp ונחכה לתשובה שתתקבל בפונקציה MX_LWIP_Process.