گزارش پروژه همگام سازی - راه‌اندازی ارتباط اترنت بین دو میکروکنترلر STM32

مرداد ماه ۱۳۹۹

شماره سند: [Comments]

ویرایش: اول

آهار
شرکت خدمات نیروگاهی

فهرست مطالب :

۱- تعریف مساله و هدف پروژه ۴
۱-۱- مشخصات سیستم ۴
۱-۲- نحوه ارزیابی سیستم ۴
۲- پیاده سازی ۵
2-1- پیاده سازی client ۵
۲-۱-۱- تنظیمات و پیکربندی اولیه ۵
۲-۱-۲- برنامه client ۱۲
2-2- پیاده سازی server ۱۶
۲-۲-۱- تنظیمات و پیکربندی اولیه ۱۶
2-2-2- برنامه server ۱۶
۳- نکات تکمیلی ۲۰
۳-۱- ارسال پکت توسط client پس از فشار دادن کلید ۲۰
۳-۲- راهنمای کار با نرم‌افزار stmstudio ۲۱
۴- مراجع ۲۵
۵- سوابق سند ۲۶

۱- تعریف مساله و هدف پروژه

برای پیاده‌سازی استاندارد IEEE 1588 با استفاده از بستر ارتباطی اترنت، لازم است در ابتدا ارتباط اترنت بین دو میکروکنترلر را برقرار کنیم، تا بتوانند در بستر اترنت به تبادل بسته‌های اطلاعاتی بپردازند. چون در user manual مربوط به میکروکنترلرهای stm32 ذکر شده بود که پیام های مربوط به همگام سازی توسط پروتکل UDP[1] ارسال می شود، بنابراین در این گزارش نحوه راه‌اندازی ارتباط بین دو میکروکنترلر از طریق پروتکل UDP آورده شده است.

۱-۱- مشخصات سیستم

سیستم هدف برای راه اندازی ارتباط اترنت بین دومیکروکنترلر از طریق پروتکل UDP، شامل دو بورد NUCLEO_F767ZI شرکت ST می باشد که دارای میکروکنترلر STM32F767_ZIT6U بر روی برد هستند.

۱-۲- نحوه ارزیابی سیستم

قرار هست ارتباط اترنت بین دو برد STM32 برقرار شود، به نحوی که بتوانند از طریق بستر ارتباطی اترنت و پروتکل UDP با هم به تبادل داده بپردازند.

1- User Datagram Protocol

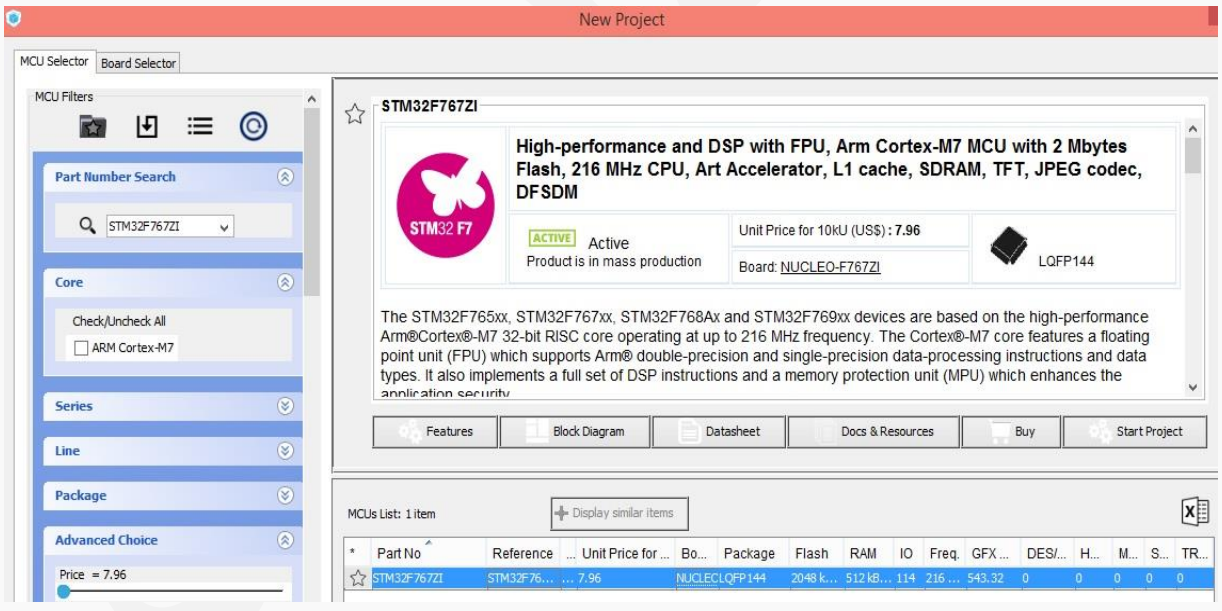
۲- پیاده سازی

برای پیاده سازی ارتباط اترنت بین دو میکروکنترلر، نیاز به دو سیستم می باشد. یکی از این دو سیستم server است و دیگری client می باشد. برای پیاده‌سازی تنظیمات اولیه میکروکنترلرها از نرم افزار STM32CUBEMX و برای برنامه نویسی از نرم‌افزار KEIL استفاده شده است. در ادامه پیاده سازی بخش client و server توضیح داده می‌شود.

۲-۱- پیاده سازی client

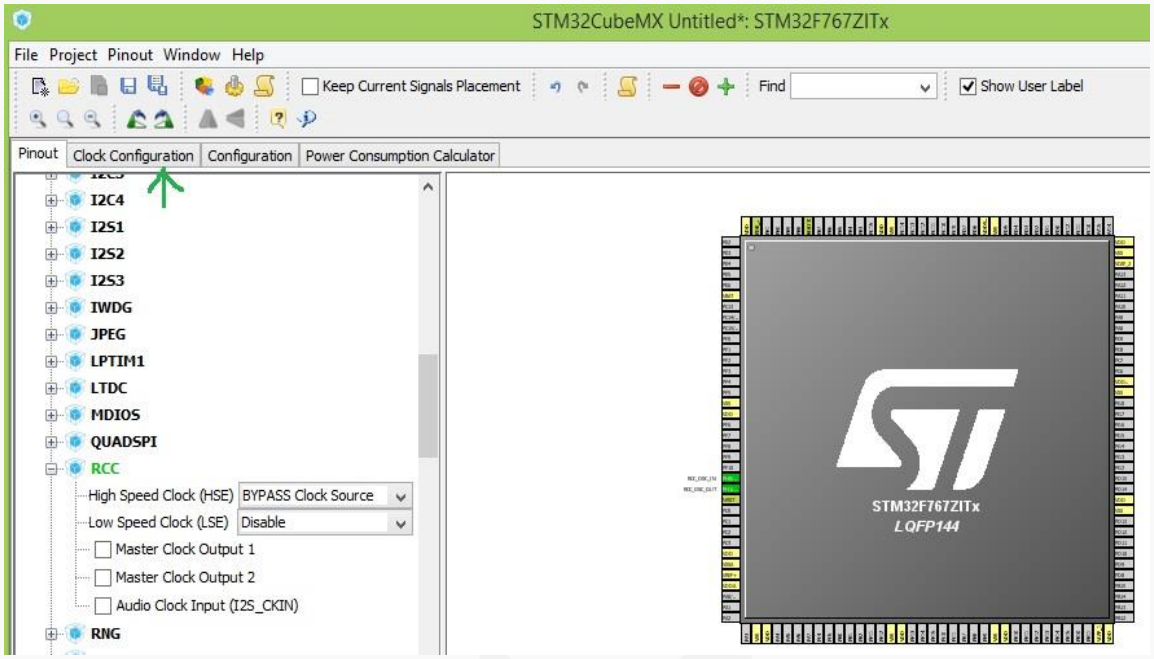
۲-۱-۱- تنظیمات و پیکربندی اولیه

ابتدا در نرم‌افزار STM32CUBEMX میکروکنترلر مورد نظر یعنی STM32F767_ZIT6U را انتخاب می‌کنیم (شکل ۱). توجه : با این که ما از برد NUCLEOF767 استفاده می‌کنیم، اما برد را انتخاب نمی‌کنیم و خود میکروکنترلر را انتخاب می‌کنیم. چون در صورت انتخاب برد مقداری تنظیمات اضافی و غیرضروری وجود دارد که حجم کد را زیاد می‌کند.

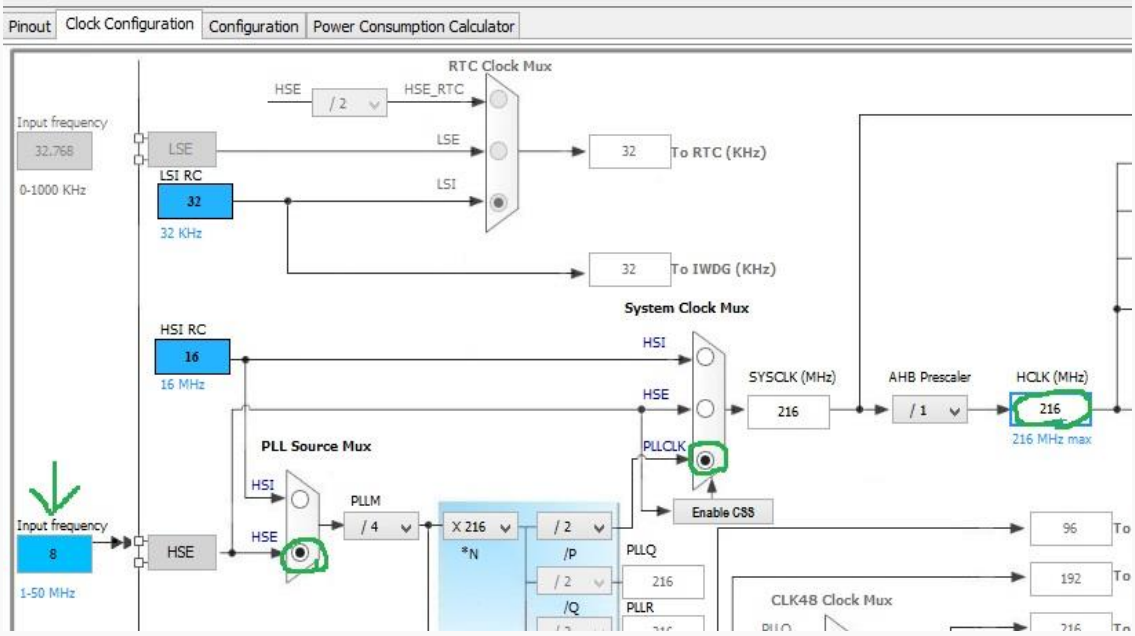


شکل ۱- انتخاب میکروکنترلر موردنظر

در قسمت RCC ، مقدار HSE را برابر با BYPASS Clock Source قرار می‌دهیم (شکل ۲). سپس به قسمت clock configuration رفته و تنظیمات را مطابق (شکل ۳) انجام می‌دهیم. (تنظیماتی که باید تغییر کند با رنگ سبز مشخص شده است).

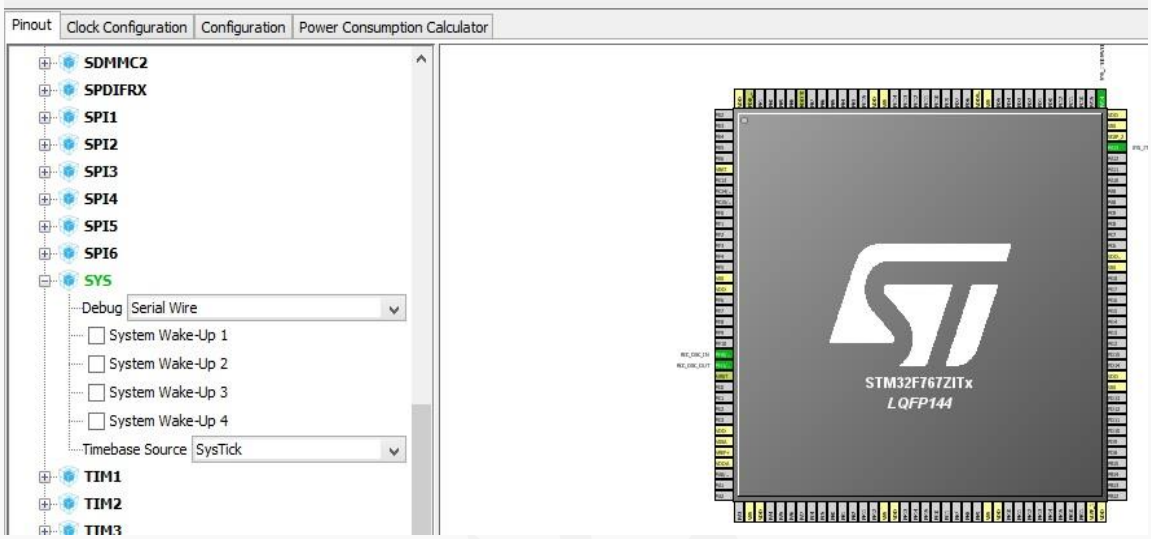


شکل ۲ – تنظیمات clk (۱)



شکل ۳ – تنظیمات clk (۲)

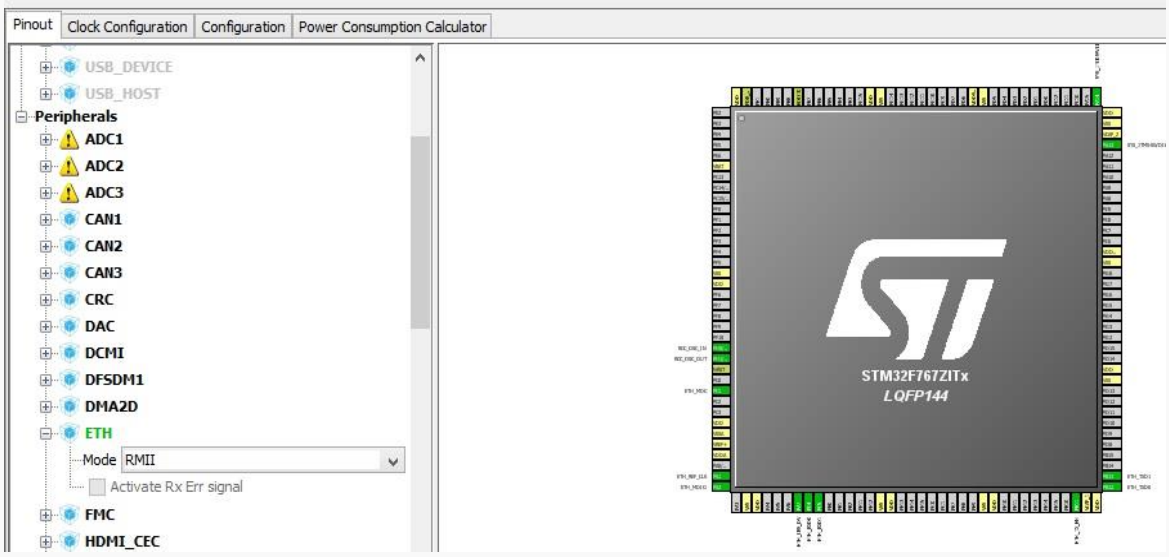
در قسمت SYS ، مقدار debug را برابر با serial wire قرار می‌دهیم (شکل ۴).



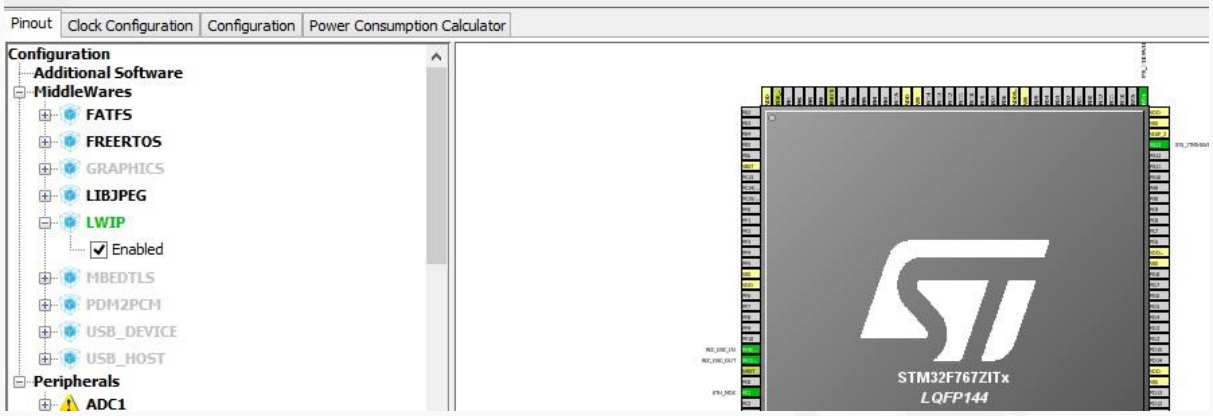
شکل ۴

Mode اترنت را برابر با RMII قرار می‌دهیم (شکل ۵). LWIP را فعال می‌کنیم (شکل ۶).

LWIP(Light Weight IP) : مجموعه‌ای از کتابخانه‌های مرتبط با اترنت می‌باشد.

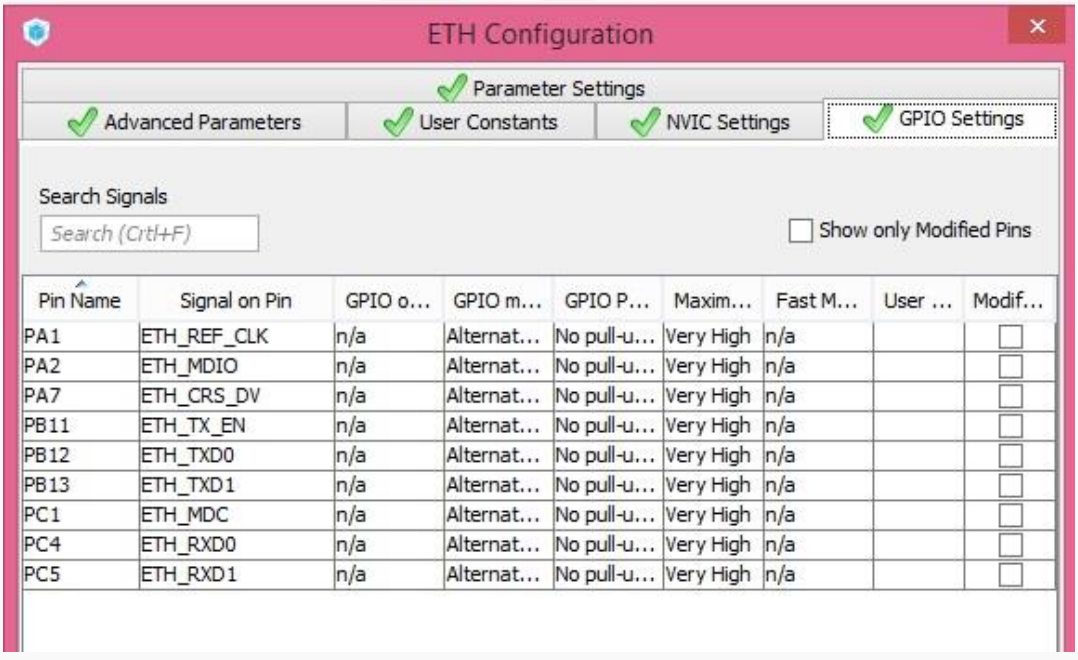


شکل ۵ – فعال‌سازی اترنت



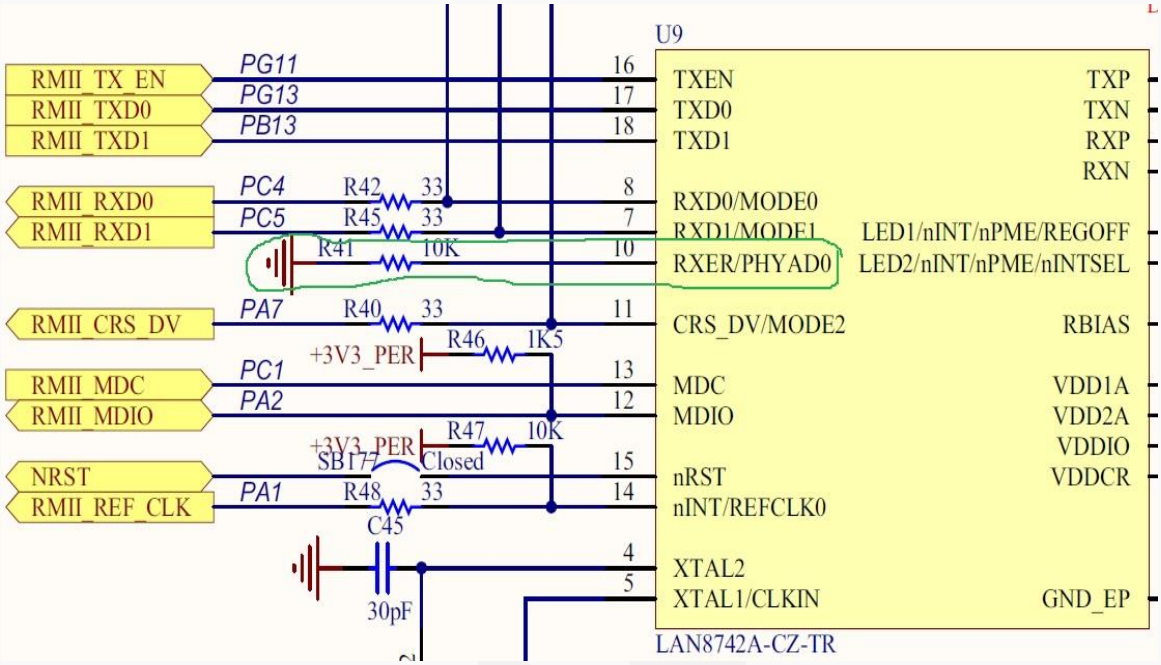
شکل ۶ – فعال سازی LWIP

سپس در قسمت تنظیمات اترنت، باید پایه‌ها را متناسب با شماتیک بردی (شکل ۸) که از آن استفاده می‌کنیم (NUCLEOF767)، تغییر دهیم (شکل ۷). هم‌چنین در قسمت parameter setting هم باید Auto Negotiation را غیرفعال کنیم و مقدار PHY Address را برابر صفر قراردهیم (شکل ۹). (مقدار PHY Address باتوجه به شماتیک تعیین می‌شود که در این برد به زمین متصل شده است. (در شکل ۸ علامت زده شده است.))

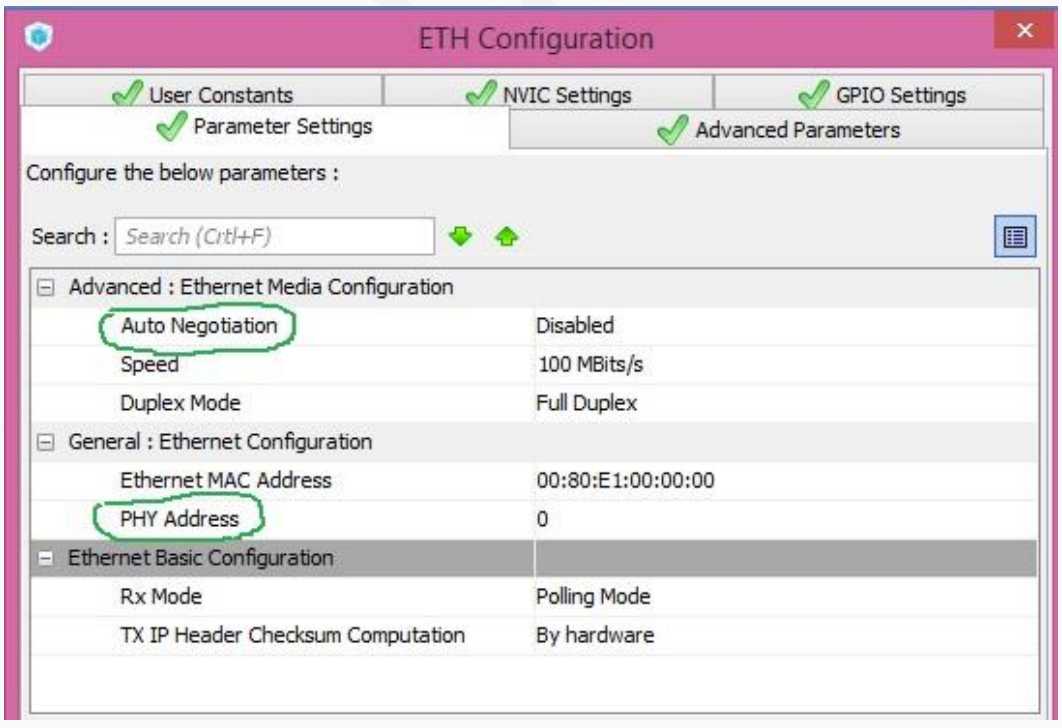


Pin Name	Signal on Pin	GPIO o...	GPIO m...	GPIO P...	Maxim...	Fast M...	User ...	Modif...
PA1	ETH_REF_CLK	n/a	Alternat...	No pull-u...	Very High	n/a		
PA2	ETH_MDIO	n/a	Alternat...	No pull-u...	Very High	n/a		
PA7	ETH_CRS_DV	n/a	Alternat...	No pull-u...	Very High	n/a		
PB11	ETH_TX_EN	n/a	Alternat...	No pull-u...	Very High	n/a		
PB12	ETH_TXD0	n/a	Alternat...	No pull-u...	Very High	n/a		
PB13	ETH_TXD1	n/a	Alternat...	No pull-u...	Very High	n/a		
PC1	ETH_MDC	n/a	Alternat...	No pull-u...	Very High	n/a		
PC4	ETH_RXD0	n/a	Alternat...	No pull-u...	Very High	n/a		
PC5	ETH_RXD1	n/a	Alternat...	No pull-u...	Very High	n/a		

شکل ۷ – تنظیمات اترنت (۱)



شکل ۸ – شماتیک اترنت و PHY برد NUCLEO F767

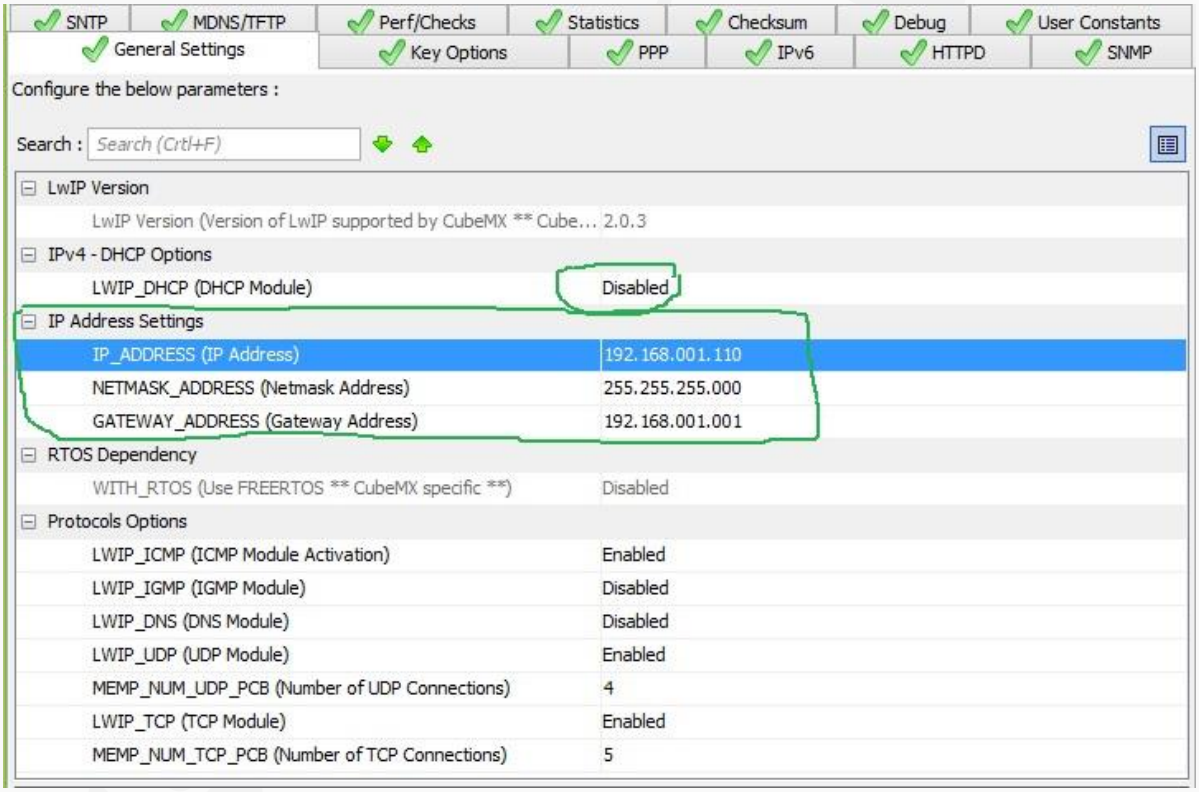


شکل ۹ – تنظیمات اترنت (۲)

سپس به قسمت تنظیمات LWIP رفته و اقدامات زیر را انجام می‌دهیم :

DHCP[1] را غیرفعال می‌کنیم و تنظیمات IP را همان طور که در شکل ۱۰ مشخص شده است انجام می‌دهیم. درقسمت IP ADDRESS، ۳ رقم آخر آن را که در اینجا به طور مثال برابر ۱۱۰ قرار داده‌ایم، می‌توانیم برابر هرمقدار دلخواه کوچکتر از ۲۵۵ قرار دهیم. اما باید به این نکته توجه داشت که در یک شبکه IP ADDRESS دو سیستم نباید برابر باشد.

DHCP : پروتکلی مرتبط با مفاهیم شبکه است که در آن DHCP Server به طور خودکار به client ها، IP اختصاص می‌دهد.



شکل ۱۰ – تنظیمات LWIP

[1] - Dynamic Host Configuration Protocol

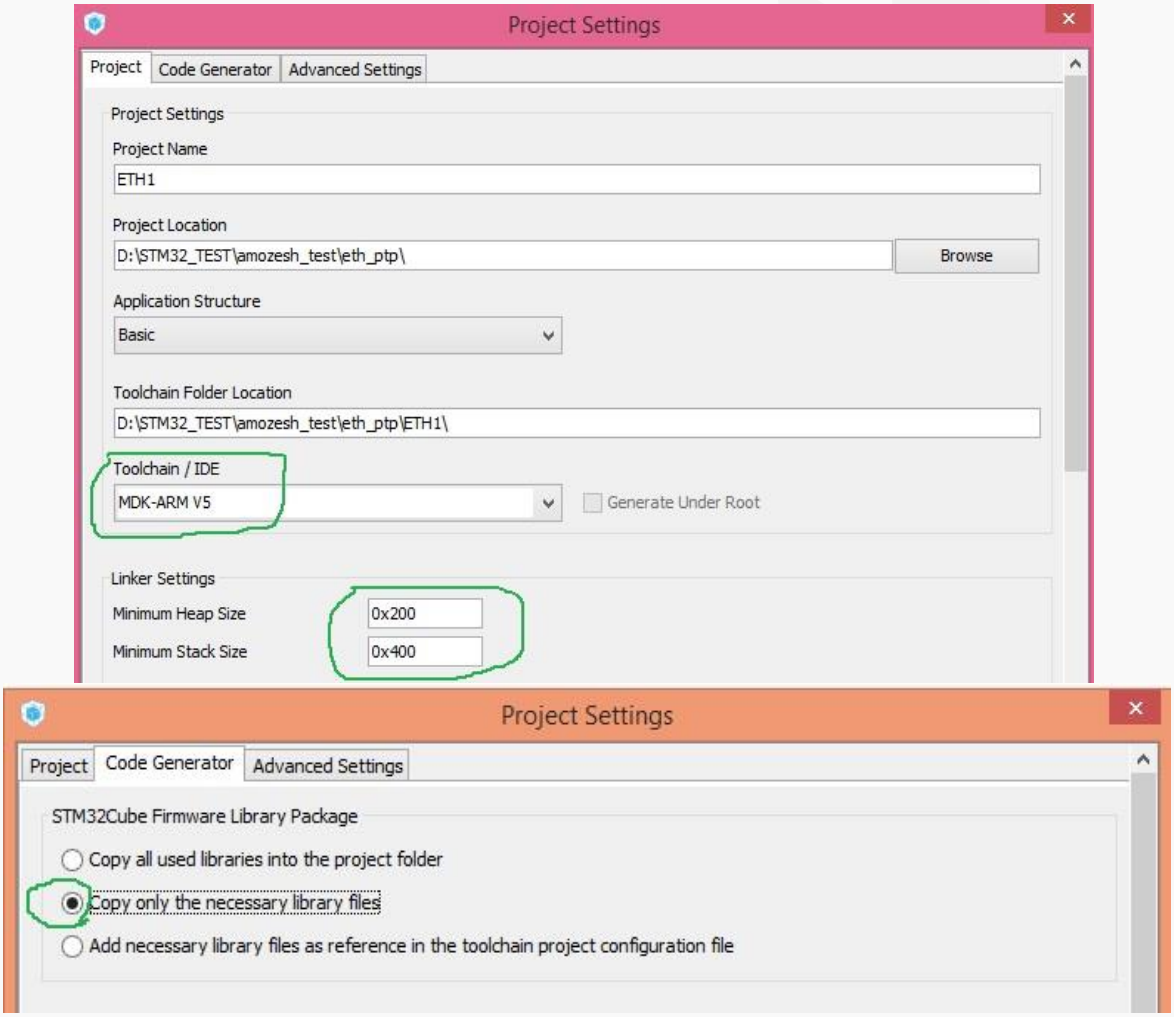
سپس یک تایمر مثلا تایمر ۴ را تنظیم می‌کنیم تا هر مثلا ۲ثانیه وقفه آن رخ دهد (این کار جزء ضروریات راه‌اندازی اترنت نیست، اما برای تست در این پروژه انجام می‌شود.).

سپس در پنجره‌ای که پس از زدن generate code باز می‌شود، تنظیمات زیر را انجام می‌دهیم (شکل ۱۱) :

تغییر IDE به MDK – ARM V5

افزایش مقدار حداقل HEAP SIZE و STACK SIZE مثلا به ترتیب 0X800 و 0X1600 کردن آن.

انتخاب گزینه copy only the necessary library files



شکل ۱۱ – تنظیمات code generator

۲-۱-۲- برنامه client

ذکر این نکته ضروری است که در این قسمت به توضیح قسمت‌هایی از برنامه پرداخته می شود که توسط ما باید به برنامه اضافه شود. در زیر طرح کلی برنامه مشاهده می‌شود.

تنظیمات و پیکربندی‌های اولیه داخل main

- HAL_TIM_Base_Start_IT(&htim4)
- ip_asign()
- udp_client_init()

ارسال پکت

داخل روتین وقفه تایمر (با پریود دلخواه مثلا ۲ثانیه) تابع udp_send1() صدا زده می‌شود.

دریافت پکت

- MX_LWIP_Process() داخل while(1)
- udp_receive_callback ()

۲-۱-۲-۱- تنظیمات و پیکربندی‌های اولیه داخل main

- HAL_TIM_Base_Start_IT(&htim4) :

این تابع برای راه‌اندازی و فعال‌کردن وقفه تایمر۴ استفاده می‌شود.

- ip_asign() :

در خط اول این تابع ip_address ای را که به برد اختصاص داده‌ایم، درون local_ip می‌ریزیم تا در ادامه راحت‌تر از آن استفاده کنیم. در خط دوم تابع هم ip_address ای را که در آرگومان اول تابع ip4addr_aton به صورت رشته قراردهیم (در اینجا "192.168.1.100") تبدیل به عدد می‌کند و داخل آرگومان دوم (remote_ip) می‌ریزد.

remote_ip ، ip سروری است که می‌خواهیم به آن متصل شویم.

```
283  void ip_asign(void)
284  {
285    local_ip = gnetif.ip_addr;
286    ip4addr_aton("192.168.1.100", &remote_ip);
287  }
```

شکل ۱۲- تابع ip_asign

- udp_client_init() :

توجه : LWIP دارای سه نوع API[1] به نام‌های raw , NETCONN , socket هست. باتوجه به این که مراجع [3],[4] که استاندارد IEEE 1588 را برای میکروکنترلرهای STM32 پیاده‌سازی کرده‌اند، از raw API استفاده کرده‌اند و علاوه بر این raw API ، API مختص LWIP می‌باشد، بنابراین برای راه‌اندازی ارتباط اترنت بین دو میکروکنترلر که گام اول برای پیاده‌سازی استاندارد IEEE 1588 است، هم از raw API استفاده کرده‌ایم.
برای استفاده از API های مربوط به پروتکل udp ، باید #include "udp.h" را درابتدای برنامه اضافه کنیم.

در خط ۳۴۴(شکل ۱۳) توسط udp_new() یک UDP pcb[2] جدید می‌سازیم. سپس در خط ۳۴۹ به remote_ip (server) و پورت ۱۲۳۴ آن متصل می‌شویم. توجه شود که این اتصال ترافیکی بین client و server ایجاد نمی‌کند. سپس اگر اتصال خطایی نداشته باشد توسط udp_recv() (خط ۳۵۴) یک callback function به نام udp_recieve_callback مشخص می‌کنیم تا هنگام دریافت پکت‌های udp صدا زده شود. اگر هم خطایی در اتصال رخ داده باشد، بلوک کنترلی udp را حذف می‌کنیم (خط ۳۵۶).

```
339  void udp_client_init(void)
340  {
341    err_t err;
342
343     /* Create a new UDP control block  */
344    udpc = udp_new();
345
346    if (udpc!=NULL)
347    {
348      /* configure destination IP address and port */
349      err= udp_connect(udpc, &remote_ip, 1234);
350
351      if (err == ERR_OK)
352      {
353        /* Set a receive callback for the upcb */
354        udp_recv(udpc, udp_receive_callback, NULL);
355      }
356      else udp_remove(udpc);
357    }
358  }
```

شکل ۱۳- تابع udp_client_init

[1] - Application Programming Interface
[2] - protocol control block

۲-۱-۲-۲- ارسال پکت :

فرآیند ارسال به این نحو است که داخل روتین وقفه تایمر (با پریود دلخواه مثلا ۲ثانیه) تابع ()udp_send1 صدا زده می‌شود تا پکتی را ارسال کند. علاوه بر این برای یک led هم داخل روتین وقفه تایمر وارون می‌شود. (شکل ۱۴)

```
87  /* USER CODE BEGIN 0 */
88  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
89  {
90    if(htim->Instance == TIM4)
91      {
92       HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
93       udp_send1();
94      }
95
96  }
```

شکل ۱۴- روتین وقفه تایمر ۴

در خط ۳۲۵ (شکل ۱۵) توسط تابع sprintf متنی را که می‌خواهیم ارسال کنیم داخل بافری که تعریف کرده‌ایم میریزیم. این تابع طول رشته‌ای را برمی‌گرداند که آن را درون متغیر len می‌ریزیم. اگر نخواهیم رشته ارسال کنیم و فقط بخواهیم یک سری اعداد ارسال کنیم، کافی است آن اعداد را داخل buf بریزیم و نیازی به استفاده از تابع sprintf نیست.

سپس در خط ۳۲۶ در داخل RAM و به طول len ، pbuf ای اختصاص می‌دهیم. LWIP پکت بافرها را با استفاده از ساختار داده‌ای به نام pbuf مدیریت می‌کند. هم‌چنین API هایی برای کارکردن با pbuf ها مانند pbuf_alloc و... دارد.

سپس در خط بعد پیامی که می‌خواستیم ارسال کنیم و داخل buf ریخته بودیم، درون pbuf کپی می‌کنیم. اگر این عمل بدون خطا انجام شود توسط udp_send ، pbuf را ارسال می‌کنیم و بعد هم فضایی را که در حافظه به pbuf اختصاص داده بودیم، آزاد می‌کنیم.

```
319  void udp_send1(void)
320  {
321    struct pbuf *pb;
322    uint16_t len;
323    err_t err;
324
325    len = sprintf(buf, "salam che khabar?");
326    pb = pbuf_alloc(PBUF_TRANSPORT, len, PBUF_RAM);
327    err = pbuf_take(pb,buf,len);
328    if(err == ERR_OK )
329     {
330      udp_send(udpc, pb);
331      pbuf_free(pb);
332     }
333  }
```

شکل ۱۵ – تابع udp_send1

۲-۱-۲-۳- دریافت پکت :

MX_LWIP_Process() :

این تابع جزء توابع LWIP است که باید آن را درون while(1) داخل main قرار دهیم. وظیفه آن خواندن پکت دریافتی از بافرهای اترنت و فرستادن آن به lwip stack برای مدیریت آن است.

- udp_receive_callback () :

همان طور که در ادامه (شکل ۲۰) توضیح داده می‌شود، server پس از دریافت پکت از client ، یک عدد یک بایتی که پس از هر دریافت یکی زیاد می‌شود را برای client می‌فرستد.
برنامه client پس از دریافت این عدد یک بایتی، وارد udp_recieve_callback (شکل ۱۶) می‌شود. در خط ۳۶۴ ، led متصل به PB14 وارون می‌کنیم تا پس از دریافت هرپکت وارون شدن این LED را بر روی برد ببینیم. سپس در خط ۳۶۶ داده دریافتی را که در pbuf ای به نام p قراردارد به بافری به نام rcv_buf که خودمان تعریف کرده‌ایم، منتقل می‌کنیم. برای مشاهده صحت عملکرد برنامه و دیدن داده دریافتی می‌توانیم از تابع printf استفاده کنیم و یا با استفاده از نرم افزار stmstudio ، rvc_buf را مستقیما مشاهده کنیم. درانتها هم فضایی را که در حافظه به pbuf اختصاص داده بودیم، آزاد می‌کنیم.

```
360  void udp_receive_callback
361    (void *arg, struct udp_pcb *upcb, struct pbuf *p, const ip_addr_t *addr, u16_t port)
362  {
363
364    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
365
366    pbuf_copy_partial( p, rcv_buf, p->len, 0);
367    //printf("rcv_buf len is:%d\r\n",p->len);
368    //printf("rcv data is: %d, %d",rcv_buf[0], rcv_buf[1]);
369
370    /* Free receive pbuf */
371    pbuf_free(p);
372  }
```

شکل ۱۶- udp_recieve_callback

۲-۱-۲-۴- متغیرهای global برنامه client :

در شکل ۱۷ متغیرهای global برنامه client که بیرون از main تعریف شده اند، آورده شده است.

```
65  ip_addr_t local_ip;
66  ip_addr_t remote_ip;
67  extern struct netif gnetif;
68  struct udp_pcb *udpc;
69  char buf[100];
70  uint8_t rcv_buf[75];
```

شکل ۱۷- متغیرهای global برنامه client

۲-۲- پیاده سازی server

۲-۲-۱- تنظیمات و پیکربندی اولیه

مشابه قسمت ۲-۱-۱- است، فقط با این تفاوت که باید Ethernet MAC Address متفاوتی نسبت به برد client به آن اختصاص دهیم. مثلا آخرین رقم سمت راست آن را یک کنیم (شکل ۹).

هم چنین باید IP address متفاوتی از برد client ، به آن اختصاص دهیم. مثلا IP address را برابر 192.168.1.100 قرار دهیم (شکل ۱۰).

۲-۲-۲- برنامه server

تنظیمات و پیکربندی‌های اولیه داخل main

- ip_asign()
- udp_server_init()

ارسال پکت

پس از دریافت پکت و داخل udp_receive_callback می‌توان پکت ارسال کرد. هم‌چنین داخل روتین وقفه تایمر (با پریود دلخواه) که این امر بستگی به کاربرد دارد.

دریافت پکت

- MX_LWIP_Process() داخل while(1)
- udp_receive_callback ()

۲-۲-۲-۱- تنظیمات و پیکربندی‌های اولیه داخل main

- ip_asign() :

این تابع (شکل ۱۸) مشابه تابع ip_asign در بخش client است. فقط با این تفاوت که ip ، remote_ip کلاینتی است که می‌خواهیم به آن متصل شده و پکت ارسال کنیم. البته در بعضی کاربردها نیازی به مقداردهی remote_ip نیست، چون server می‌تواند client ip را که به server پکت ارسال کرده به دست آورد و به همان ip پاسخ دهد.

```
279  void ip_asign(void)
280  {
281    local_ip = gnetif.ip_addr;
282    ip4addr_aton("192.168.1.50", &remote_ip);
283  }
```

شکل ۱۸ – تابع ip_asign

- Udp_server_init() :

در خط ۲۹۳(شکل ۱۹) توسط udp_new() یک UDP pcb جدید می‌سازیم. سپس در خط ۲۹۹ UDP pcb را به server ip address و پورت ۱۲۳۴ آن bind می‌کنیم. سپس اگر اتصال خطایی نداشته باشد توسط udp_recv() (خط ۳۰۴) یک callback function به نام udp_recieve_callback مشخص می‌کنیم تا هنگام دریافت پکت‌های udp صدا زده شود.

اگر هم خطایی در اتصال رخ داده باشد، بلوک کنترلی udp را حذف می‌کنیم (خط ۳۰۸).

```
287  void udp_server_init(void)
288  {
289    struct udp_pcb *udpc;
290    err_t err;
291
292    /* Create a new UDP control block  */
293    udpc = udp_new();
294
295    if (udpc)
296    {
297      /* Bind the upcb to the UDP_PORT port */
298      /* Using IP_ADDR_ANY allow the upcb to be used by any local interface */
299       err = udp_bind(udpc, &local_ip, 1234);
300
301       if(err == ERR_OK)
302       {
303         /* Set a receive callback for the upcb */
304         udp_recv(udpc, udp_echoserver_receive_callback, NULL);
305       }
306       else
307       {
308         udp_remove(udpc);
309       }
310    }
311  }
```

شکل ۱۹ – تابع udp_server_init

۲-۲-۲-۲- دریافت و ارسال پکت

MX_LWIP_Process() :

این تابع جزء توابع LWIP است که باید آن را درون while(1) داخل main قرار دهیم. وظیفه آن خواندن پکت دریافتی از بافرهای اترنت و فرستادن آن به lwip stack برای مدیریت آن است.

- udp_receive_callback () :

همان طور که در شکل ۲۰ مشاهده می‌شود، پس از دریافت پکت، برنامه این callback را اجرا می‌کند. در خط ۳۲۲ ، led متصل به پورت B0 را وارون می‌کنیم تا پس از دریافت هر پکت این علامت را روی برد ببینیم. سپس در خط ۳۲۴ ، داده دریافتی را که در p قراردارد به بافری به نام rcv_buf که خودمان تعریف کرده‌ایم، منتقل می‌کنیم. سپس در دو خط بعدی توسط تابع printf طول و محتوای بافر دریافتی را چاپ می‌کنیم. برای مشاهده داده‌های چاپ شده توسط printf می‌توان برنامه را در حالت debug اجرا کرد.

سپس می‌خواهیم، سرور پس از هر بار دریافت پکت یک عدد ۱بایتی که یکی یکی زیاد می‌شود را برای client به عنوان تایید بفرستد. در خط ۳۲۸ و ۳۲۹ ، pbuf را برای این منظور با آن یک بایت مدنظر پرمی‌کنیم. سپس در خط ۳۳۴ ، server به همان ip_address و شماره پورتی که client برایش پکت ارسال کرده بود، متصل می‌شود. در خط ۳۳۵ داده را به client ارسال می‌کنیم.

در خط ۳۳۹ ارتباط را قطع می‌کنیم و در خط ۳۴۲ فضایی را که در حافظه به pbuf اختصاص داده بودیم، آزاد می‌کنیم.

```
317  void udp_echoserver_receive_callback
318    (void *arg, struct udp_pcb *udpc, struct pbuf *p, const ip_addr_t *addr, u16_t port)
319  {
320    err_t err1;
321
322    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
323
324    pbuf_copy_partial( p, rcv_buf, p->len, 0);
325    printf("rcv_buf len is:%d\r\n",p->len);
326    printf("rcv data is: %s",rcv_buf);
327
328    buf[0] = j++;
329    err1 = pbuf_take(p, buf, 1);
330
331    if(err1== ERR_OK)
332      {
333       //Connect to the remote client
334       udp_connect(udpc, addr, port);
335       udp_send(udpc, p);
336      }
337
338    /* free the UDP connection, so we can accept new clients */
339    udp_disconnect(udpc);
340
341    /* Free the p buffer */
342    pbuf_free(p);
343  }
```

شکل ۲۰ - udp_recieve_callback

۲-۲-۲-۳- متغیرهای global برنامه server :

در شکل ۲۱ متغیرهای global برنامه server که بیرون از main تعریف شده اند، آورده شده است.

```
66  ip_addr_t local_ip;
67  ip_addr_t remote_ip;
68  extern struct netif gnetif;
69  struct udp_pcb *udpc;
70  char buf[100];
71  char rcv_buf[50];
72  uint8_t j;
```

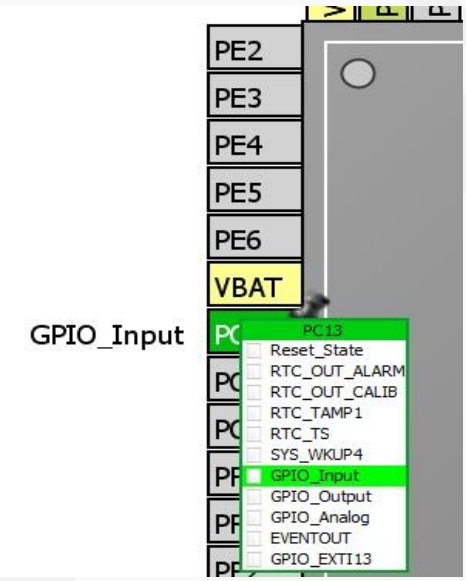
شکل ۲۱- متغیرهای global برنامه server

۳- نکات تکمیلی

۳-۱- ارسال پکت توسط client پس از فشار دادن کلید

در قسمت ۲-۱-۲-۲- گفته شد که داخل روتین اینتراپت تایمر ۴ (مثلا هر ۲ثانیه) پکت ارسال می‌شود. می‌توان برنامه را طوری نوشت که پس از هربار زدن کلید روی برد ارسال پکت رخ دهد. برای این منظور اقدامات زیر را باید انجام دهیم :

- در نرم‌افزار cubemx ، پین PC13 را که کلید (کلید آبی رنگ در برد NUCLEOF767) به آن متصل است، ورودی کنیم (شکل ۲۲). در برد NUCLEOF767 این پین به زمین متصل شده است.
- مقدار period و prescaler تایمر ۴ را به گونه‌ای تنظیم می‌کنیم که اینتراپت آن هر ۲۰ میلی ثانیه رخ دهد. سپس همان طور که در خط ۹۷ شکل ۲۳ مشاهده می شود، داخل اینتراپت تایمر چک می‌کنیم که اگر کلید در حال فشرده شدن است، مقدار متغیر touch_cnt را یکی اضافه کنیم. پس از حدود ۱۰۰ میلی ثانیه اینتراپت تایمر ۵ بار رخ داده است و اگر در این مدت کلید در حال فشردن بوده باشد، متغیر touch_cnt برابر با ۵ می‌شود و پکت ارسال می‌شود.



شکل ۲۲

```
92  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
93  {
94    if(htim->Instance == TIM4)
95      {
96       //i++;
97       if( HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) )
98         {
99         touch_cnt++;
100        if(touch_cnt==5)
101           {
102           HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_0);
103           udp_send1();
104           touch_cnt = 0;
105           }
106        }
107     }
108
109 }
```

شکل ۲۳ – اینتراپت تایمر برای ارسال پکت با فشردن کلید

۳-۲- راهنمای کار با نرم‌افزار stmstudio

همان طور که در قسمت ۲-۱-۲-۳- ذکر شده است، rcv_buf که حاوی عددی است که server برای client ارسال کرده است را توسط برنامه stmstudio مقدارش را مشاهده می‌کنیم.

به طور کلی برای مشاهده مقادیر متغیرها در حین اجرای برنامه توسط نرم‌افزار stmstudio به شرح زیر عمل می‌کنیم:

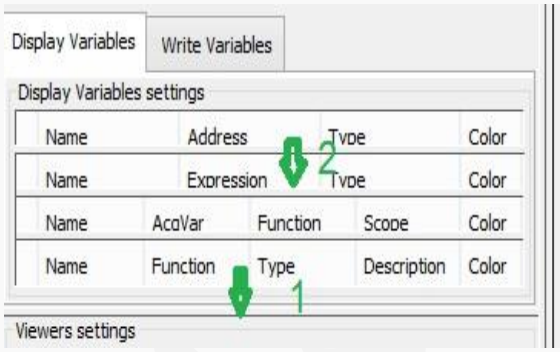
- ابتدا همان طور که در شکل ۲۴ مشخص شده است به ترتیب به نواحی مشخص شده می‌رویم تا نشانگر ماوس به فلش دو جهته تبدیل شود. سپس با کشیدن به پایین فضای خالی در آن قسمت صفحه ایجاد می شود. در این فضای خالی کلیک راست کرده و import را می‌زنیم (شکل ۲۵) تا کادر شکل ۲۶ باز شود. در قسمت ۱ این شکل مسیری که فایل axf. پروژه در آن قرار دارد را وارد می‌کنیم. برای مثال اگر نام پروژه‌ای که ساخته‌ایم را eth_stm32 گذاشته‌ایم، این فایل در مسیر زیر قرار دارد :

  MDK-ARM\ eth_stm32\ eth_stm32.axf

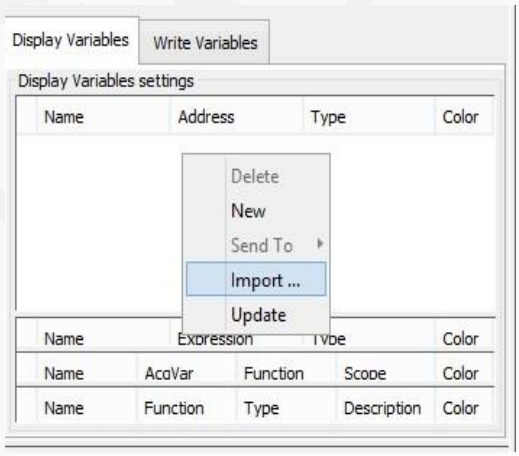
  سپس در کادر وسط صفحه (قسمت ۲ شکل ۲۶) نام متغیری که می‌خواهیم مانیتور شود را وارد می‌کنیم و سپس دکمه import را می‌زنیم (قسمت ۳ شکل ۲۶). برای درج متغیرهای دیگر نیز این ۲ مرحله را تکرار می‌کنیم. سپس مطابق شکل ۲۷ متغیرها را انتخاب کرده و کلیک راست کرده و sendto-> varviewer1 را انتخاب می‌کنیم.

سپس همان گونه که در قسمت ۱ شکل ۲۸ مشخص شده است، نوع نمایش متغیرها را انتخاب می‌کنیم. نمایش به صورت نمودار (curve)، به صورت نمودار میله‌ای (Bar Graph) و یا به صورت جدول (table).

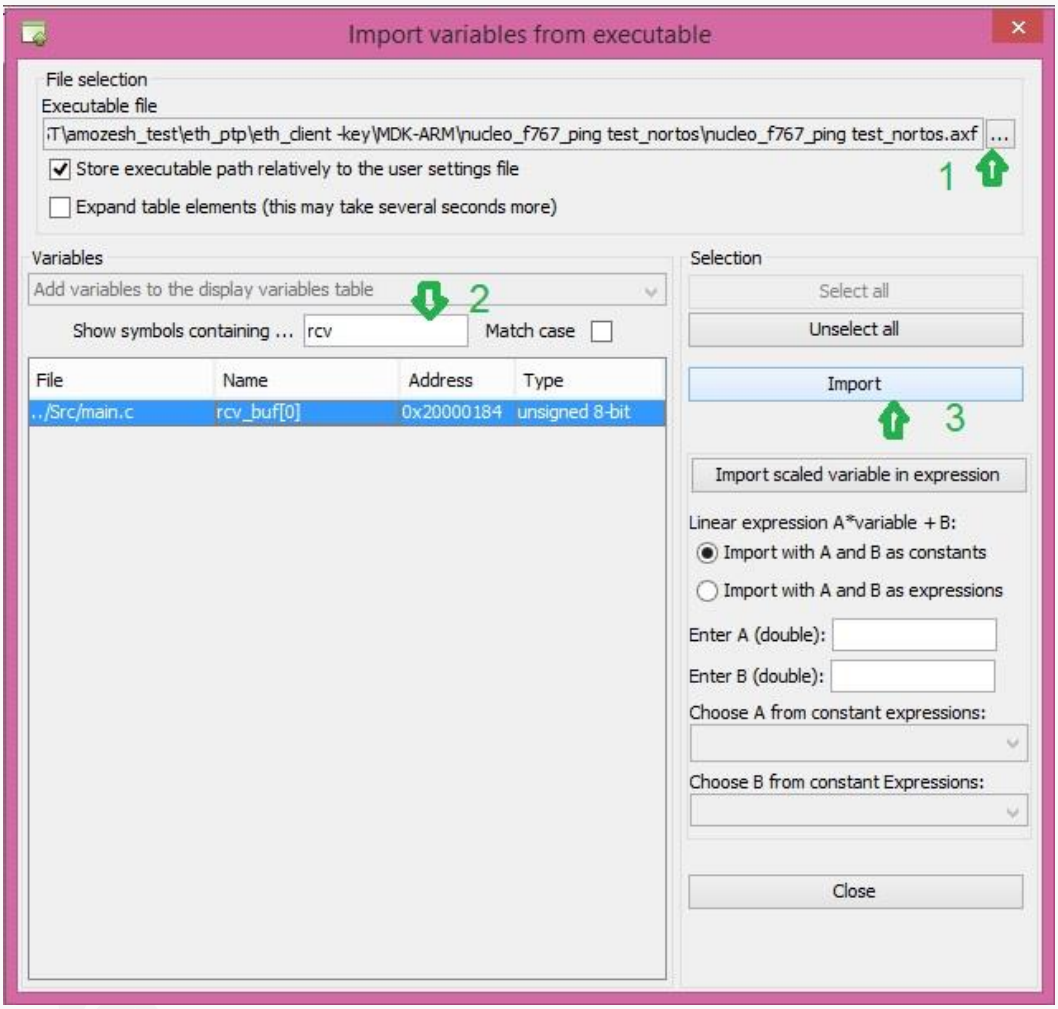
سپس کلید run در بالای صفحه را می‌زنیم تا برنامه اجرا شود (قسمت ۲ شکل ۲۸).



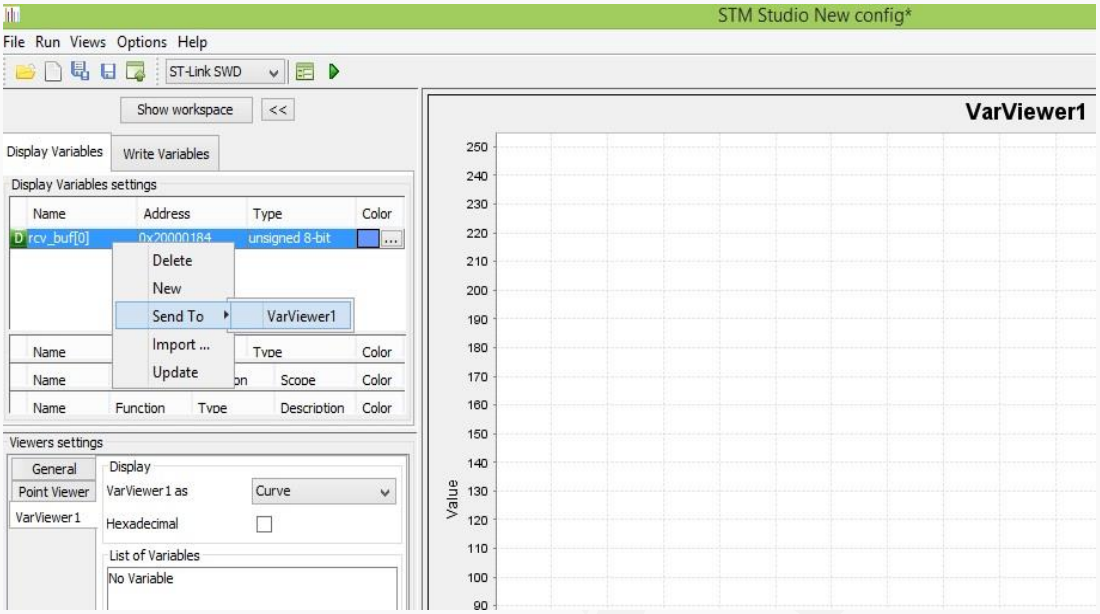
شکل ۲۴ – نرم افزار stmstudio (۱)



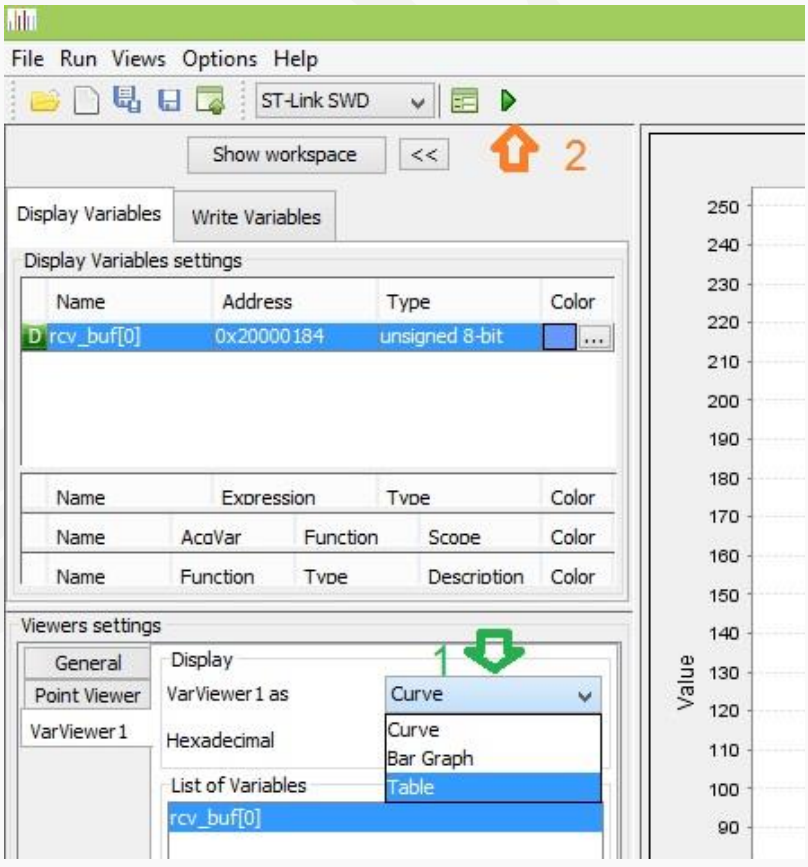
شکل ۲۵- نرم افزار stmstudio (۲)



شکل ۲۶ – نرم‌افزار stmstudio (۳)



شکل ۲۷ – نرم‌افزار stmstudio (۴)



شکل ۲۸ – انتخاب نوع نمایش متغیرها و اجرای برنامه

۴- مراجع

[1] RM0410 (STM32F76XXX and STM32F77XXX Reference Manual)

[2] www.st.com / AN3966 (LwIP TCP/IP stack demonstration for STM32F4x7 microcontrollers)

[3] Guang You Yang, Yi Zheng, Zhi Yan Ma, and Xin Yu Hu. 2012. “The Implementation of IEEE 1588-2008 Precision Time Protocol on the STM32F107”. Key Engineering Materials 522 (2012), 868–873.

[4] - www.st.com / AN3411 (IEEE 1588 precision time protocol demonstration for STM32F107 connectivity line microcontroller)

۵- سوابق سند

نام نویسنده	تاریخ	پروژه مرتبط	امضا دیجیتال	تایید کننده	امضا تایید کننده
عماد عرفانیان و حمید نوری	۱۵/۰۹/۲۰۲۰	1234Q456q1230		[Manager]	
شماره سند	[Comments]				
کلمات کلیدی	[Keywords]				
اسناد مرتبط	تمپلیت گزارش فنی فارسی				
توضیحات مختصری در ارتباط با محتوی سند: [Abstract]					

ویرایشگر اول	تاریخ	پروژه مرتبط	امضا دیجیتال	تایید کننده	امضا تایید کننده
عماد عرفانیان	۲۰۲۰/۱۰/۱۲	1234Q456q1230			
توضیح دقیق تغییرات ایجاد شده: اضافه کردن عنوان شماره ۳ (نکات تکمیلی)					

ویرایشگر دوم	تاریخ	پروژه مرتبط	امضا دیجیتال	تایید کننده	امضا تایید کننده
		1234Q456q1230			
توضیح دقیق تغییرات ایجاد شده:					

ویرایشگر سوم	تاریخ	پروژه مرتبط	امضا دیجیتال	تایید کننده	امضا تایید کننده
توضیح دقیق تغییرات ایجاد شده :					

10/12/2020

Address: Ahar Building, Science & Tech. Park, 12th km Asian Road, Mashhad, Iran.

Tel:+98(0)51 35 42 41 00 Fax: +98 (0)51 35 42 41 61

info@aharco.com

Email: info@aharco.com