

Ethernet (ETH): Media Access Control (MAC) With DMA Controller

Bruno Guimarães Bitencourt, Ian Fernandes Miranda

Introdução

Trabalho apresentado à disciplina de Programação de Sistemas Embarcados, cursada no primeiro semestre DE 2021. O objetivo é a implementação de uma aplicação com o tema Ethernet (ETH): Media Access Control (MAC) With DMA Controller. Para tal, foi utilizada a placa NUCLEO-F7617ZI, que utiliza o microcontrolador STM32F7.

A aplicação proposta consiste na criação de um Web-Server na qual o objetivo foi integração do sistema com o microcontrolador. A partir do projeto desenvolvido, é possível acionar os LED's da própria placa e um Buzzer externo, através dos botões da página HTML implementada.

Sumário

Introdução	1
Ethernet (ETH): Media Access Control (MAC) With DMA Controller	2
Aplicação usando Ethernet com DMA	2
Configuração de Hardware	2
Configuração Clock	3
Configuração PWM	3
Configuração Ethernet	3
Pinagem	4
Software	4
Configuração do IP	4
Comunicação com WebServer	5
Página HTML	6
Conclusão	8

Ethernet (ETH): Media Access Control (MAC) With DMA Controller

A Ethernet (ETH): Media Access Control (MAC) é o periférico para suporte da camada de enlace do protocolo Ethernet. Implementa sua própria DMA para um controle automático do fluxo de dados. Dentre seus benefícios incluem uma conformidade completa no padrão IEEE 802 MAC e permite um desenvolvimento mais eficiente de aplicações baseadas no modelo TCP/IP. A Figura 1, mostra como é visto as camadas:

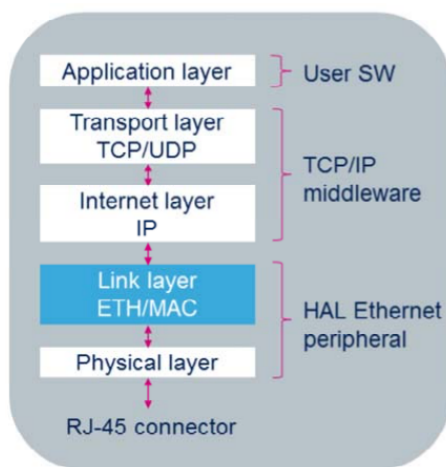


Figura 1: Modelo TCP/IP

As camadas superiores mostradas pela Figura 1 são implementadas via software, como por exemplo a camada de transporte e internet, que podem ser desenvolvidas a partir do LwIp.

Aplicação usando Ethernet com DMA

Configuração de Hardware

A aplicação faz a utilização de um *buzzer* externo e da comunicação Ethernet, sendo assim, é necessário configurar as conexões da placa de acordo com a Figura 2. O cabo de alimentação é necessário para programar a placa e, uma vez programada, pode ser substituído por uma fonte de alimentação externa.

O pino positivo do buzzer, deve ser conectado ao pino PC7 da NUCLEO-F767ZI, e o GND, ao GND da placa (existem diversos pinos GND nos conectores da NUCLEO).

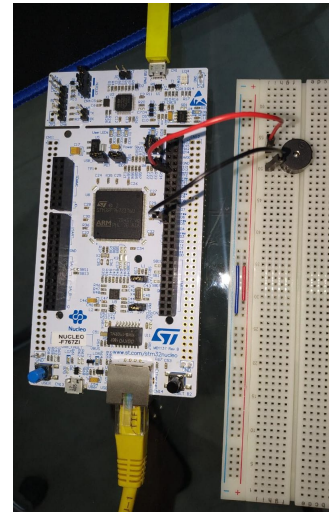
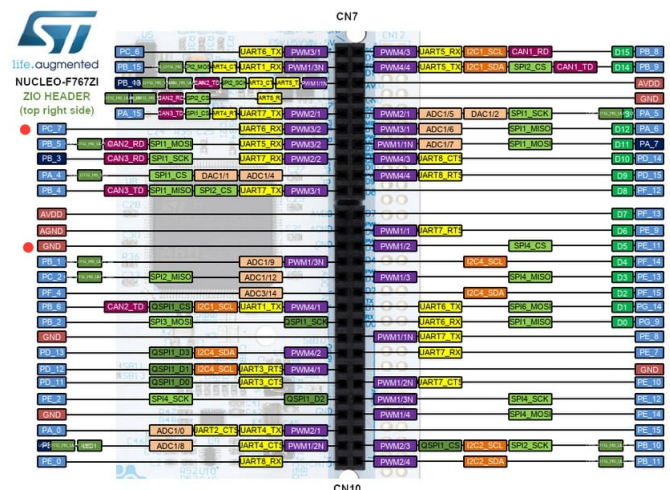


Figura 2: Configuração de Hardware

Os pinos utilizados na montagem podem ser vistos na Figura 3.



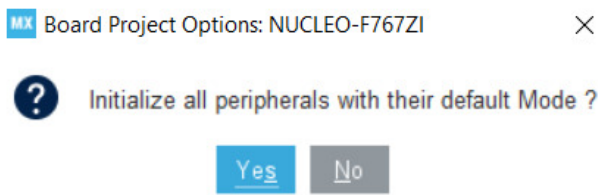


Figura 4: Inicialização dos periféricos

Configuração Clock

Para o projeto foi escolhido o *timer3 channel2* como timer para gerar a onda PWM necessária para acionar o *buzzer*. A primeira configuração a ser feita é a da árvore de clock, foi definido que a primeira nota a ser acionada pelo *buzzer* seria um lá, que tem frequência de 440Hz com 50% de *duty cycle*. Sendo assim, a configuração do clock como mostrada na Figura 5, nos dá uma frequência de 6MHz.

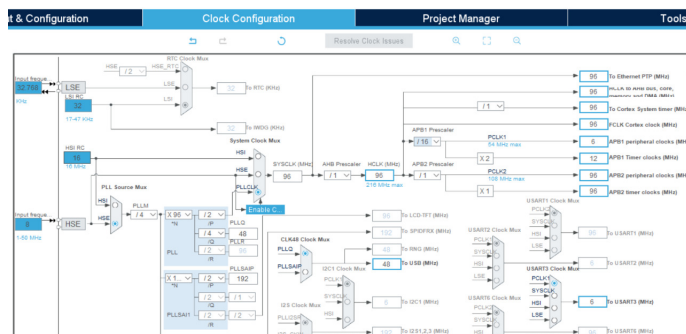


Figura 5: Configuração do clock

Configuração PWM

Para encontrar o valor do *Counter Period* (CP), basta dividir a frequência de 6MHz pela frequência desejada de 440Hz, nos dando assim, um CP de 13636. O prescaler também foi calculado fazendo a divisão da frequência do clock pelo produto entre o CP e a frequência desejada e subtraído de um. O valor encontrado foi igual a zero. Portanto a configuração do *timer* pode ser vista nas Figuras 6 e 7.

Configuração Ethernet

Após a configuração do *timer*, deve ser feita a configuração do Ethernet, que deve ser ligado no modo RMII.

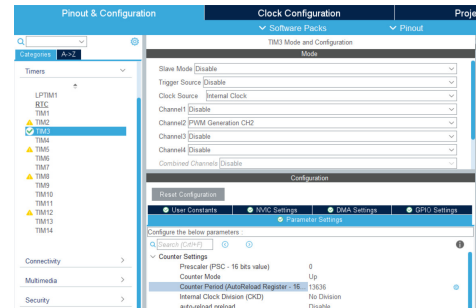


Figura 6: Configuração do timer

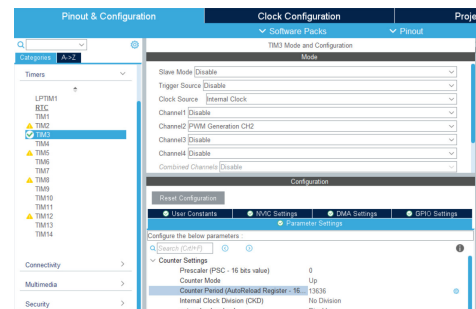


Figura 7: Configuração do timer

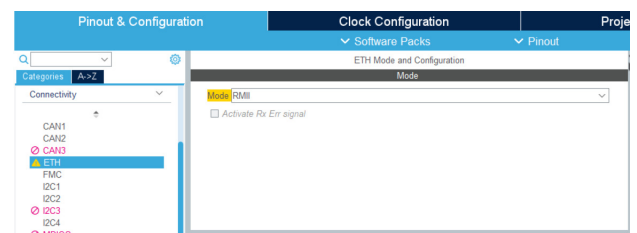


Figura 8: Configuração do Ethernet

Também será necessária a configuração do LWIP, de acordo com as Figuras 9 e 10.

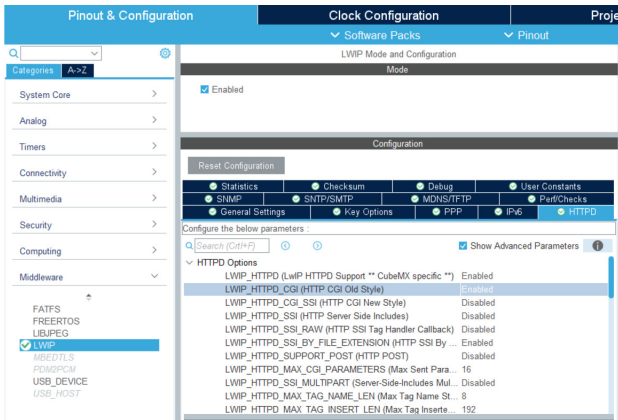


Figura 9: Configuração do LWIP

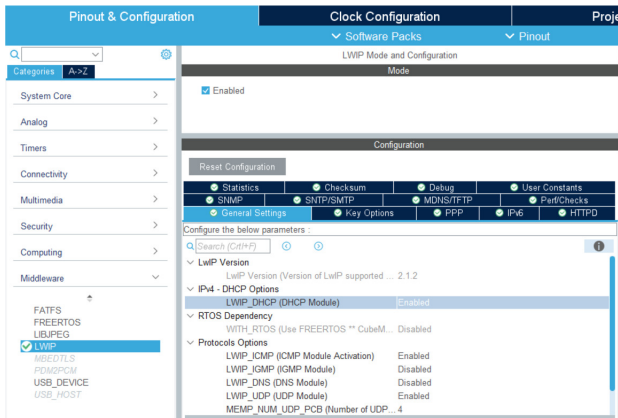


Figura 10: Configuração do LWIP

Pinagem

Após todas as configurações realizadas nas seções anteriores, a pinagem do projeto completa pode ser vista nas Figuras 11, 12, 13 e 13.

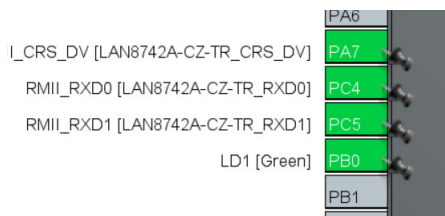


Figura 11: Pinagem do projeto



Figura 12: Pinagem do projeto

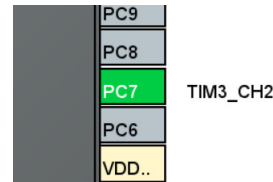


Figura 13: Pinagem do projeto

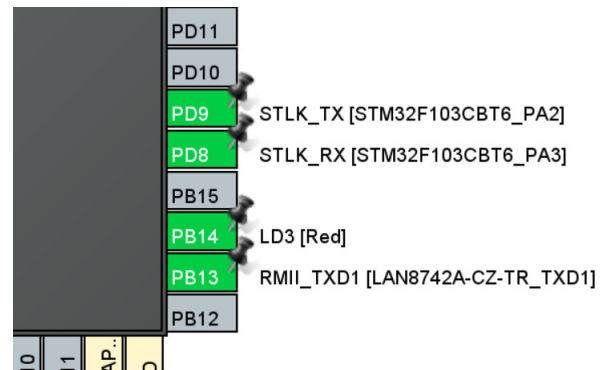


Figura 14: Pinagem do projeto

Software

Para programar o projeto, será necessário realizar mudanças nos arquivos main.c e lwip.c, além de adicionar os arquivos relacionados à página HTML.

Configuração do IP

A primeira modificação no lwip.c é a configuração do IP do microcontrolador. Para tal, foi adicionado a linhas de código 1, 2, que definiram o endereço IP, Máscara e Gateway. Dessa forma, a página web desenvolvida se encontra no endereço 192.168.192.157 (endereço de IP definido). Esse endereço de IP estático será utilizado posteriormente para acessar o servidor web criado pela NUCLEO.

```
uint8_t IP_ADDRESS[4]={192,168,192,157};
uint8_t NETMASK_ADDRESS[4]={255,255,255,0};
```

```
uint8_t GATEWAY_ADDRESS[4]={192,168,192,1};
```

Código 1: Declaração do IP

```
/* IP addresses initialization with DHCP (
   IPv4) */
IP4_ADDR(&ipaddr, IP_ADDRESS[0], IP_ADDRESS
[1], IP_ADDRESS[2], IP_ADDRESS[3]);
IP4_ADDR(&netmask, NETMASK_ADDRESS[0],
NETMASK_ADDRESS[1], NETMASK_ADDRESS[2],
NETMASK_ADDRESS[3]);
IP4_ADDR(&gw, GATEWAY_ADDRESS[0],
GATEWAY_ADDRESS[1], GATEWAY_ADDRESS[2],
GATEWAY_ADDRESS[3]);

netif_add(&gnetif, &ipaddr, &netmask, &gw,
NULL, &ethernetif_init, &ethernet_input);
```

Código 2: Configuração do IP dentro da função MX_LWIP_Init

Comunicação com WebServer

CGI foi o método utilizado para atualização dinâmica de conteúdos de páginas Web. De acordo com [3], CGI (Common Gateway Interface) fornece uma interface entre o HTTP Server e programas, gerando conteúdo web, sendo tais programas conhecidos como scripts de CGI. O servidor HTTP processa script linha por linha e chama as funções CGI quando necessárias. A saída de uma função CGI é enviada para o cliente (browser) como parte de uma página.

Para aplicação é mostrado os códigos 3, 4 e 5, que tem como função receber os valores vindos da página HTML. De maneira geral se houver o caracter "led", seguido de um valor 1, 2, 3 (valores correspondentes ao LED da placa do microcontrolador) ele acende e valor 4 apaga todos os leds. Se o caracter lido for frequência existe a possibilidade de três opções:

1. Toca o buzzer numa frequência de 440Hz
2. Toca o buzzer numa frequência menor que 440Hz.
3. Desliga o buzzer

```
// just declaring the function for the
   compiler [= CGI #2 =]
const char* LedCGIhandler(int iIndex, int
iNumParams, char *pcParam[], char *pcValue
[]);

// in our HTML file <form method="get" action
="/leds.cgi"> [= CGI #3 =]
```

```
const tCGI LedCGI = {"/leds.cgi",
LedCGIhandler};
```

```
// [= CGI #4 =]
tCGI theCGItable[1];
```

Código 3: Declaração de funções para o compilador

```
static void setPWM(TIM_HandleTypeDef,
uint32_t, uint16_t, uint16_t);
```

Código 4: Declaração de função para manipulação do PWM. Fonte: [2]

```
// the actual function for handling CGI [=
   CGI #5 =]
const char* LedCGIhandler(int iIndex, int
iNumParams, char *pcParam[],
char *pcValue[])
{

uint32_t i = 0;

for (i = 0; i < iNumParams; i++)
{
if (strcmp(pcParam[i], "led") == 0)
{

if (strcmp(pcValue[i], "1") == 0)
{
HAL_GPIO_WritePin(LD1_GPIO_Port,
LD1_Pin, GPIO_PIN_SET);
}

else if (strcmp(pcValue[i], "2") == 0)
{
HAL_GPIO_WritePin(LD2_GPIO_Port,
LD2_Pin, GPIO_PIN_SET);
}

else if (strcmp(pcValue[i], "3") == 0)
{
HAL_GPIO_WritePin(LD3_GPIO_Port,
LD3_Pin, GPIO_PIN_SET);
}

else if (strcmp(pcValue[i], "4") == 0)
{
//turning the LED lights off
HAL_GPIO_WritePin(LD1_GPIO_Port,
LD1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(LD2_GPIO_Port,
LD2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(LD3_GPIO_Port,
LD3_Pin, GPIO_PIN_RESET);
}

}

else if (strcmp(pcParam[i], "frequencia")
== 0)
```



```

{
    if (strcmp(pcValue[i], "1") == 0)
    {
        setPWM(htim3, TIM_CHANNEL_2, 13636,
6818); //440Hz
    }

    else if (strcmp(pcValue[i], "2") == 0)
    {

        setPWM(htim3, TIM_CHANNEL_2, 25000,
6818);

    }
    else if (strcmp(pcValue[i], "3") == 0)
    {

        setPWM(htim3, TIM_CHANNEL_2, 15341,
0); //0Hz

    }
}

// the extension .shtml for SSI to work
return "/index.html";
} // END [= CGI #5 =]

// function to initialize CGI [= CGI #6 =]
void myCGIinit(void) {
    //add LED control CGI to the table
    theCGItable[0] = LedCGI;
    //give the table to the HTTP server
    http_set_cgi_handlers(theCGItable, 1);
} // END [= CGI #6 =]

```

Código 5: Implementação da função responsável pelo acionamento do LED e do buzzer. Fonte: [1]

```

// initializing the HTTPd [-HTTPd #2-]
httpd_init();

// initializing CGI [= CGI #7 =]
myCGIinit();

```

Código 6: Inicialização das funções declaradas. Fonte: [1]

```

while (1)
{
    /* USER CODE END WHILE */
    MX_LWIP_Process();

    /* USER CODE BEGIN 3 */
}

```

Código 7: Loop interno da main. Fonte: [1]

Além disso, para o funcionamento do buzzer foi utilizada a função setPWM, desenvolvida por [2] e mostrada pelo código 8

```

void setPWM(TIM_HandleTypeDef timer, uint32_t
channel, uint16_t period, uint16_t pulse)
{
    HAL_TIM_PWM_Stop(&timer, channel);
    TIM_OC_InitTypeDef sConfigOC;
    timer.Init.Period = period;
    HAL_TIM_PWM_Init(&timer);
    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = pulse;
    sConfigOC.OCpolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
    HAL_TIM_PWM_ConfigChannel(&timer, &
sConfigOC, channel);
    HAL_TIM_PWM_Start(&timer, channel);
}

```

Código 8: Implementação da função para manipulação do PWM. Fonte: [2]

Página HTML

Para interface do sistema foi desenvolvida a página HTML mostrada pela Figura 15. Na página é possível acender os LEDs 1, 2 ou 3 a partir dos botões. Ainda é possível acionar o buzzer em duas frequências distintas ou desligá-lo. O trecho em HTML da página é mostrado pelo Código 9.

Os botões encontram-se num formulário, que ao ser submetido, tem uma ação “LED.cgi”, que é utilizada no trecho de código em C

Para a construção da página o grupo utilizou como referência o site W3Schools. Inicialmente foi utilizado CSS, entretanto devido a complicações ao gerar o HTML, tal uso foi retirado.

```

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device
-width, initial-scale=1">
    <style>

    </style>
</head>

<body>

    <div class="header">
        <h2><center>Programacao de Sistemas
Embarcados</center></h2>

```

Programacao de Sistemas Embarcados

Grupo: Bruno Guimaraes Bitencourt e Ian Fernandes
Professor: Ricardo Oliveira Duarte

[Aplicacao Ethernet](#)

LED 01:

Funcao: Acender LED verde da Placa

[1]

LED 02:

Funcao: Acender LED azul da Placa

[2]

LED 03:

Funcao: Acender LED vermelho da Placa

[3]

Apagar LEDs:

Funcao: Apagar todos os LEDs da Placa

[4]

Buzzer:

Frequencia A:

[1]

Frequencia B:

[2]

Desligar:

[3]

Figura 15: Página HTML desenvolvida

```
<p><b>Grupo:</b> Bruno Guimaraes
Bitencourt e Ian Fernandes<br>
<b>Professor:</b> Ricardo Oliveira
Duarte</p>
</div>

<div class="topnav" id="myTopnav">
  <a href="#home" class="active">
Aplicacao Ethernet</a>
</div>

<form method="get" action="/leds.cgi">
<div class="row">
  <div class="column">
    <h2>LED 01: </h2>
    <p>Funcao: Acender LED verde da Placa
</p>
    <label class="switch">
      <input type="submit" name="led"
value="1">
    </label>
  </div>

  <div class="column">
    <h2>LED 02: </h2>
    <p>Funcao: Acender LED azul da Placa<
/p>
    <label class="switch">
      <input type="submit" name="led"
value="2">
    </label>
  </div>

  <div class="column">
    <h2>LED 03: </h2>
    <p>Funcao: Acender LED vermelho da
```

```
Placa</p>
  <label class="switch">
    <input type="submit" name="led"
value="3">
  </label>
</div>

<div class="column">
  <h2>Apagar LEDs: </h2>
  <p>Funcao: Apagar todos os LEDs da
Placa</p>
  <label class="switch">
    <input type="submit" name="led"
value="4">
  </label>
</div>

<div class="column">
  <h2>Buzzer: </h2>

  <form>
    <div class="col-75">
      <p>Frequencia A: </p>
      <input type="submit" name="
frequencia" value="1">
    </div>

    <div class="col-75">
      <p>Frequencia B: </p>
      <input type="submit" name="
frequencia" value="2">
    </div>

    <div class="col-75">
      <p>Desligar: </p>
      <input type="submit" name="
frequencia" value="3">
    </div>

  </form>

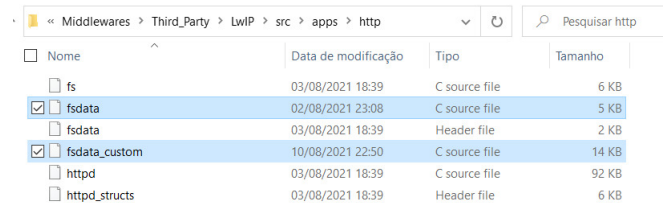
</div>
</form>

</body>
</html>
```

Código 9: Página de Interface com STM32

A NUCLEO-F767ZI não permite a importação direta de páginas no formato HTML puro, portanto foi necessário gerar dois arquivos .c a partir da página HTML criada. Essa conversão de formatos foi feita utilizando o *makefsdata*, disponibilizado por [1]. O arquivo *fsdata.custom.c* é criado a partir do seguinte comando: *perl makefsdata.pl* e deve ser incluído na pasta

do projeto dentro da pasta Middlewares/ Third_Party/ LwIP/ src/ apps/ http/, como pode ser visto na Figura 16



Nome	Data de modificação	Tipo	Tamanho
fs	03/08/2021 18:39	C source file	6 KB
fsdata	02/08/2021 23:08	C source file	5 KB
fsdata	03/08/2021 18:39	Header file	2 KB
fsdata_custom	10/08/2021 22:50	C source file	14 KB
httpd	03/08/2021 18:39	C source file	92 KB
httpd_structs	03/08/2021 18:39	Header file	6 KB

Figura 16: Configuração página HTML dentro do projeto

Após adicionar os dois arquivos .c, eles devem ser excluídos da build dentro do projeto, como mostrado nas Figuras 17 e 18.

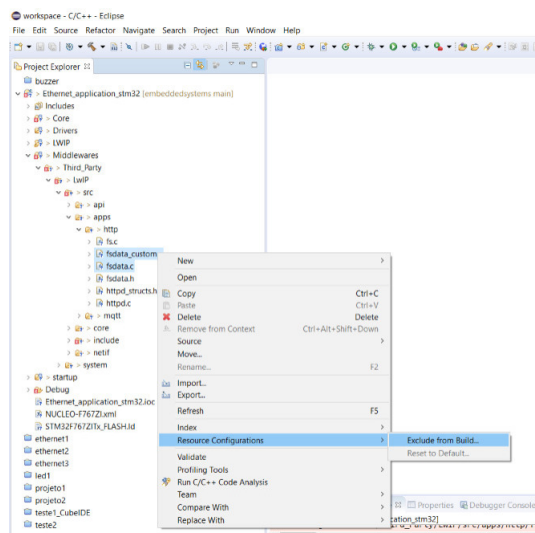


Figura 17: Exclusão da build

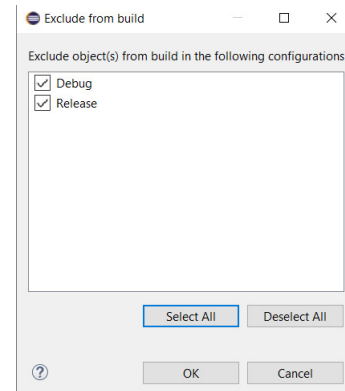


Figura 18: Exclusão da build

Após todas essas configurações, o projeto está apto para ser compilado e passado à NUCLEO-F767ZI. Para a verificação de seu correto funcionamento, basta acessar o IP definido durante a configuração (192.168.192.157) utilizando um browser de internet. Uma página igual à mostrada na Figura 15 irá se abrir e ao acionar os botões, pode-se perceber o acionamento dos LEDs e do buzzer.

Conclusão

O trabalho possibilitou um maior entendimento sobre a programação de microcontroladores ARM e principalmente sobre aplicação utilizando Ethernet. Foi necessário definir um IP estático para acesso do servidor web, diferentemente do mostrado por [1]. Além disso a aplicação permitiu o acionamento de um buzzer através de um sinal PWM gerado pelo Timer3.

A partir dos cálculos demonstrados para a configuração da nota lá (440Hz), é possível estender a aplicação para o acionamento do buzzer em diferentes frequências de forma a funcionar como um piano, sendo necessária a inclusão de novos botões na página HTML.

Tabela 1: Histórico de Revisão.

Revisão	Autor	Mudança

Referências

- [1] HTTPd web-server on STM32 NUCLEO-F767ZI microcontroller. Programmed in STM32CubeIDE <<http://ausleuchtung.ch/stm32-nucleo-f767zi-web-server/>>.
- [2] Aula 13 - PWM (Pulse Width Modulation) - Modulação por Largura de Pulso - Notas de Aula - Ricardo de Oliveira Duarte.
- [3] HTTP Server - Dynamic Web Content - CGI Programming - keil.com/pack/doc/mw/Network/html/group_ws_script_language.html