

فاطمه سادات سیفی – 98243035

پارسا محمدپور – 98243050

پروژه درس ریزپردازنده و زبان اسمبلی

## مقدمه:

این پروژه در اصل از ۳ بخش تشکیل شده است:

- ۱- Proteus
- ۲- Stm32CubeMX
- ۳- Keil

در ادامه ابتدا کلیت و نحوه انجام کار را توضیح می دهیم.

در این سوال از ما خواسته شده است که صدا ها را از طریق سیگنال ها، به اعداد و کاراکتر ها تبدیل کنیم!

برای این کار ابتدا یک ماژول speaker بر میداریم (در قسمت پرئوتو توضیح داده خواهد شد) که وظیفه تبدیل یک فایل عه audio ورودی به ولتاژ را دارد.

سپس در میکرو کنترلرمان، این ولتاژ های ورودی را دریافت می کنیم و به عنوان نمونه ذخیره می کنیم.

سپس الگوریتم گوئرتزل را بر روی آن ها و فرکانس های داده شده اجرا می کنیم.

سپس با توجه به خروجی الگوریتم گوئرتزل برای فرکانس های هر سطر و ستون، مشخص می کنیم که هر نمونه ورودی مربوط به کدام کاراکتر بوده است.

سپس سون سگمنت را برای آن کاراکتر ست می کنیم.

همچنین به منظور لاگ گرفتن و دیدن جزئیات (البته در ورژن نهایی خیلی از لاگ گرفتن ها پاک شده ان) از ترمینال استفاده کرده و

یک سری از موارد مانند سطر و ستون های هر کاراکتر (در صورت تشخیص دادن کاراکتر) را نمایش می دهیم.

## :Proteus

در این قسمت به توضیح المان ها و ریزه کاری ها و جزئیات مورد استفاده در نرم افزار شبیه ساز proteus می پردازیم.

برای شبیه سازی کردن این سوال، از المان های زیر استفاده کرده ایم:

- میکروکنترلر stm32f401re  
همان میکروکنترلرمان می باشد که فایل hex را دریافت می کند
- ماژول سون سگمنت عه کاتودی:  
active high می باشد، یعنی در صورت یک بودن ورودی هایش، هر کدام یک از 7 بخش را روشن می کند.
- ماژول Speaker :  
صرفاً برای پخش صدای مورد بررسی در نرم افزار پروتئوس است و کار خاصی را انجام نمی دهد.
- اسیلوسکوپ :  
وجود این ماژول هم اجباری نیست، صرفاً برای این است که شکل موج عه هر فایل صوتی (پس از تبدیل شدنش به ولتاژ) نشان داده شود
- ماژول AUDIO از قسمت GENERATOR ها:  
این ماژول، آدرس یک فایل عه صوتی را دریافت می کند، سپس با توجه به آن فایل، ولتاژ ورودی صدای موجود در آن را تولید می کند.  
همچنین در این ماژول قابلیت عهخ ست کردن عه مقدار offset و amplitude وجود دارد.
- Virtual terminal:  
از این ماژول، برای لاگ گرفتن از اطلاعات عه برنامه استفاده کرده ایم و نبود آن مشکلی ایجاد نمی کند.

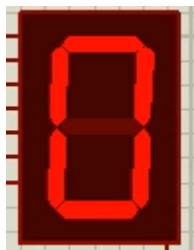
## نکاتی در رابطه با نرم افزار proteus:

- در این نرم افزار، برای virtual terminal، از USART استفاده کردیم و مقدار Baud rate عه آن را هم برابر با 115200 قرار دادیم.
- خروجی عه ماژول speaker (لیبل عه AUDIO) را به اسیلوسکوپ وصل کردیم تا شکل موج عه حاصل را نمایش دهد.
- در این پروژه، با ولتاژ عه رفرنس (VDD) عه 5v کار می کنیم.
- در ماژول عه speaker، برای اینکه ولتاژ حاصل از آن بین 0 ولت، تا 5 ولت باشد، مقدار عه offset آن را برابر با 2.5 v قرار دادیم و مقدار عه amplitude عه آن را برابر با 2.5 v قرار دادیم. تا ولتاژ حاصل و خروجی عه آن از فایل مورد نظر، بین صفر تا 5 ولت نوسان کند.
- آدرس عه فایل صوتی ورودی را در قسمت عه edit properties به ماژول عه audio generator می دهیم.
- در پین عه ورودی میکروکنترلرمان که مربوط به تبدیل آنالوگ به دیجیتال است، یک نسان دهنده ولتاژ قرار دادیم تا مقدار ولتاژ را در هر لحظه بتوانیم در پروتئوس ببینیم.
- برای اینکه در این پروژه از مبدل عه آنالوگ به دیجیتال استفاده شده است، ولتاژهای رفرنس عه میکروکنترلرمان را نیز به VDD و GND وصل می کنیم تا بتوانیم در میکروکنترلرمان عملیات عه تبدیل عه ورودی عه آنالوگ به دیجیتال را انجام بدهیم.

نحوه نمایش کاراکترها بر روی سون سگمنت:

شکل عه سون سگمنت عه روشن شده هر کاراکتر را در زیر آن نمایش می دهیم:

کاراکتر '0':



کاراکتر '1':



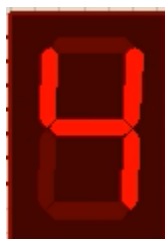
کاراکتر '2':



کاراکتر '3':



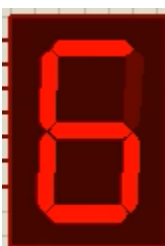
کاراکتر '4':



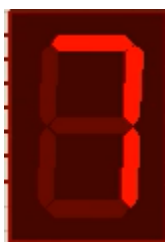
کاراکتر '5':



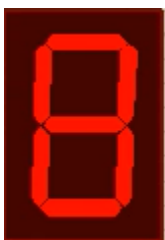
کاراکتر '6':



کاراکتر '7':



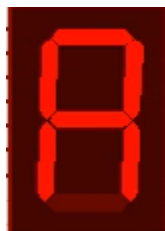
کاراکتر '8':



کاراکتر '9':



کاراکتر 'A':



کاراکتر 'B':



کاراکتر 'C':



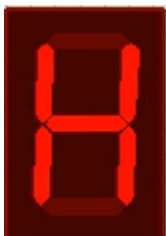
کاراکتر 'D':



کاراکتر '#':



کاراکتر '\*':





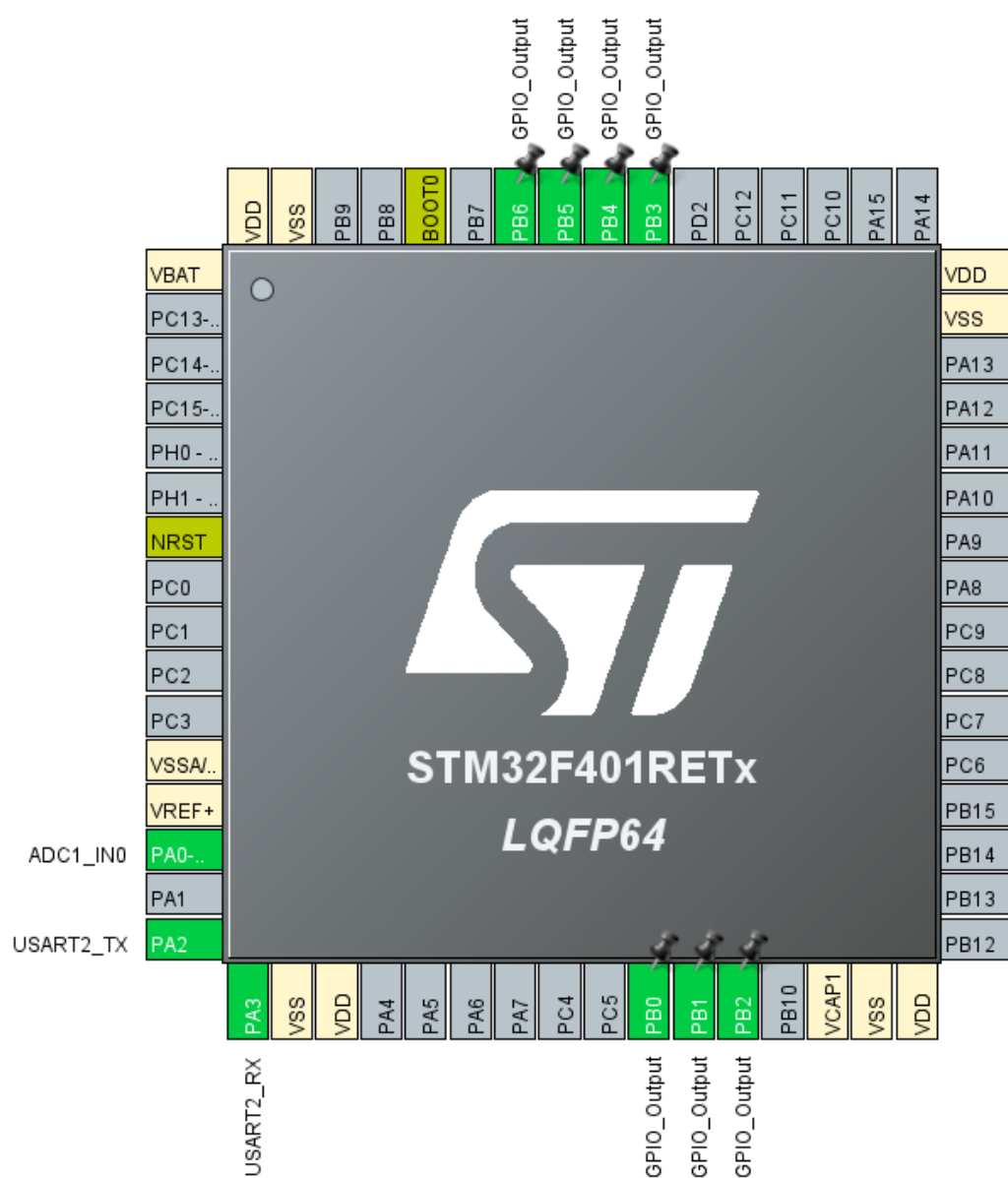
## STM32CubeMX:

این نرم افزار از بخش هایی تشکیل شده است که هر کدام را به توضیح می دهیم و عکس مربوط به هر قسمت از آن را نیز قرار می دهیم.

### • Pinout & configuration

در این قسمت، تنظیمات اصلی و اساسی مربوط به میکروکنترلرمان را انجام می دهیم. تنظیماتی مربوط به پین ها، تنظیمات عه مربوط به کانال های USART و ... .

عکس زیر، نمای میکروپی و شماتیک عه تنظیم پین ها را نشان می دهد:



خود این قسمت، همانطور که توضیح داده شد، از بخش های مختلفی تشکیل شده است که هر کدام را توضیح می دهیم:

## System Core ○

عکس های مربوط به این قسمت را قرار می دهیم:

Categories A-Z

System Core

DMA

**GPIO**

IWDG

NVIC

RCC

▲ SYS

WWDG

Analog

Timers

Connectivity

Multimedia

Computing

Middleware

Configuration

Group By Peripherals

**GPIO** **ADC** **USART**

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output I...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PB0	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB1	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB2	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB3	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB4	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB5	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>
PB6	n/a	Low	Output Push ...	No pull-up an...	Low		<input type="checkbox"/>

Configuration

Group By Peripherals

**GPIO** **ADC** **USART**

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output I...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	ADC1_IN0	n/a	Analog mode	No pull-up an...	n/a		<input type="checkbox"/>

Configuration

Group By Peripherals

**GPIO** **ADC** **USART**

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output I...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA2	USART2_TX	n/a	Alternate Fun...	No pull-up an...	Very High		<input type="checkbox"/>
PA3	USART2_RX	n/a	Alternate Fun...	No pull-up an...	Very High		<input type="checkbox"/>

## :Analog ○

عکس های مربوط به این قسمت را قرار می دهیم:

Q

Categories

A->Z

System Core >

Analog >

▲ ADC1

Timers >

Connectivity >

Multimedia >

Computing >

Middleware >

Q

Categories

A->Z

System Core >

Analog >

▲ ADC1

Timers >

Connectivity >

Multimedia >

Computing >

Middleware >

ADC1 Mode and Configuration

Mode

☒ IN0

☐ IN1

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

ADCs\_Common\_Settings

Mode

Independent mode

ADC\_Settings

Clock Prescaler

PCLK2 divided by 2

Resolution

12 bits (15 ADC Clock cycles)

Data Alignment

Right alignment

Scan Conversion Mode

Enabled

Continuous Conversion Mode

Enabled

Discontinuous Conversion Mode

Disabled

DMA Continuous Requests

Disabled

End Of Conversion Selection

EOC flag at the end of single channel conversion

ADC\_Regular\_ConversionMode

Number Of Conversion

1

External Trigger Conversion Source

Regular Conversion launched by software

External Trigger Conversion Edge

None

Rank

1

Channel

Channel 0

Sampling Time

480 Cycles

ADC\_Injected\_ConversionMode

Number Of Conversions

0

WatchDog

Enable Analog WatchDog Mode

☐

ADC1 Mode and Configuration

Mode

☒ IN0

☐ IN1

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

GPIO Settings

Search Constants

Search (Ctrl+F)

add

remove

Constant Name	Constant Value
---------------	----------------

Q

CategoriesA->Z

System Core

>

Analog

>

▲ ADC1

>

Timers

>

Connectivity

>

Multimedia

>

Computing

>

Middleware

>

CategoriesA->Z

System Core

>

Analog

>

▲ ADC1

>

Timers

>

Connectivity

>

Q

CategoriesA->Z

System Core

>

Analog

>

▲ ADC1

>

Timers

>

Connectivity

>

Multimedia

>

Computing

>

Middleware

>

ADC1 Mode and Configuration

Mode

☒ IN0

☐ IN1

Configuration

Reset Configuration

☒ Parameter Settings

☒ User Constants

☒ NVIC Settings

☒ DMA Settings

☒ GPIO Settings

NVIC Interrupt Table

Enabled

Preemption Priority

Sub Priority

ADC1 global interrupt

☐

0

0

ADC1 Mode and Configuration

Mode

☒ IN0

☐ IN1

Configuration

Reset Configuration

☒ Parameter Settings

☒ User Constants

☒ NVIC Settings

☒ DMA Settings

☒ GPIO Settings

DMA Request

Stream

Direction

Priority

ADC1 Mode and Configuration

Mode

☒ IN0

☐ IN1

Configuration

Reset Configuration

☒ Parameter Settings

☒ User Constants

☒ NVIC Settings

☒ DMA Settings

☒ GPIO Settings

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name

Signal on Pin

GPIO output I...

GPIO mode

GPIO Pull-up...

Maximum ou...

User Label

Modified

PA0-WKUP

ADC1\_IN0

n/a

Analog mode

No pull-up an...

n/a

☐

## :Connectivity ○

عکس های مربوط به این قسمت را قرار می دهیم:

The image displays two screenshots of the STM32CubeMX software interface, illustrating the configuration of the USART2 peripheral.

**Top Screenshot:** The left sidebar shows the 'Connectivity' category selected, with 'USART2' highlighted. The main panel shows the 'Mode' configuration for USART2, where 'Mode' is set to 'Asynchronous' and 'Hardware Flow Control (RS232)' is set to 'Disable'. Below this, the 'Configuration' tab is active, showing 'Parameter Settings' for USART2. The parameters are:

- Basic Parameters:**
  - Baud Rate: 115200 Bits/s
  - Word Length: 8 Bits (including Parity)
  - Parity: None
  - Stop Bits: 1
- Advanced Parameters:**
  - Data Direction: Transmit Only
  - Over Sampling: 16 Samples

**Bottom Screenshot:** The left sidebar is identical to the top screenshot. The main panel shows the 'Configuration' tab for USART2, but the 'Parameter Settings' are not visible. Instead, the 'Search Constants' section is active, showing a search bar and a table with columns 'Constant Name' and 'Constant Value'.

CategoriesA-Z

System Core >

Analog >

Timers >

Connectivity >

I2C1

I2C2

I2C3

SDIO

SPI1

SPI2

SPI3

USART1

USART2

USART6

USB\_OTG\_FS

Multimedia >

Computing >

Middleware >

CategoriesA-Z

System Core >

Analog >

Timers >

Connectivity >

I2C1

I2C2

I2C3

SDIO

SPI1

SPI2

SPI3

USART1

USART2

USART6

USB\_OTG\_FS

Multimedia >

Computing >

Middleware >

USART2 Mode and Configuration

Mode

ModeAsynchronous

Hardware Flow Control (RS232)Disable

Configuration

Reset Configuration

Parameter SettingsUser ConstantsNVIC SettingsDMA SettingsGPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
USART2 global interrupt	<input type="checkbox"/>	0	0

USART2 Mode and Configuration

Mode

ModeAsynchronous

Hardware Flow Control (RS232)Disable

Configuration

Reset Configuration

Parameter SettingsUser ConstantsNVIC SettingsDMA SettingsGPIO Settings

DMA Request	Stream	Direction	Priority
-------------	--------	-----------	----------

CategoriesA-Z

System Core >

Analog >

Timers >

Connectivity >

⚠ I2C1

⚠ I2C2

I2C3

SDIO

SPI1

SPI2

SPI3

USART1

⚠ USART2

USART6

USB\_OTG\_FS

Multimedia >

Computing >

Middleware >

Mode

ModeAsynchronous

Hardware Flow Control (RS232)Disable

Configuration

Reset Configuration

✔ Parameter Settings

✔ User Constants

✔ NVIC Settings

✔ DMA Settings

✔ GPIO Settings

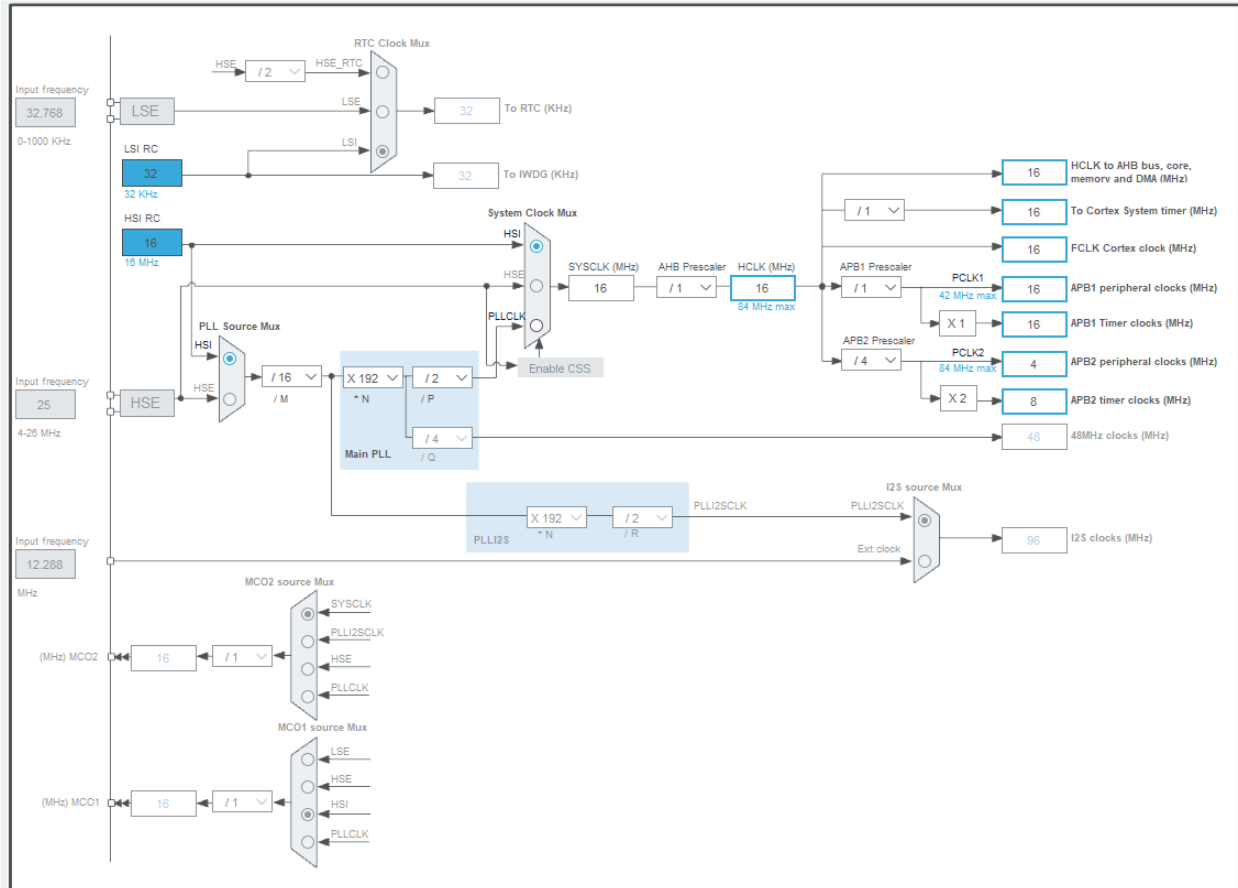
Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up	Maximum ou	User Label	Modified
PA2	USART2_TX	n/a	Alternate Fu...	No pull-up an...	Very High		<input type="checkbox"/>
PA3	USART2_RX	n/a	Alternate Fu...	No pull-up an...	Very High		<input type="checkbox"/>

## • Clock Configuration :



در این قسمت، با توجه به sampling rate و حساب کردن coefficient های مورد نظر برای الگوریتم عه گوئرتزل، باید در قسمت های APB2 Prescaler ضریب عه  $\frac{1}{4}$  را اعمال می کردیم و در قسمت عه بعد، برای ضریب عه 2 را اعمال می کردیم تا بتوانیم فرکانسی حدودا معادل 8000 (در اصل 8333.3333...) ایجاد کنیم.



## :Project Manager •

Code Generator	STM32Cube MCU packages and embedded software packs <input type="radio"/> Copy all used libraries into the project folder <input checked="" type="radio"/> Copy only the necessary library files <input type="radio"/> Add necessary library files as reference in the toolchain project configuration file
Advanced Settings	Generated files <input type="checkbox"/> Generate peripheral initialization as a pair of '.c/.h' files per peripheral <input type="checkbox"/> Backup previously generated files when re-generating <input checked="" type="checkbox"/> Keep User Code when re-generating <input checked="" type="checkbox"/> Delete previously generated files when not re-generated
	HAL Settings <input type="checkbox"/> Set all free pins as analog (to optimize the power consumption) <input type="checkbox"/> Enable Full Assert
	Template Settings Select a template to generate customized code <a href="#">Settings...</a>

Project	Project Settings Project Name <input type="text" value="Micro-Project"/> Project Location <input type="text" value="C:\Users\parsa\Desktop\Riz\Project\CubeMX"/> <a href="#">Browse</a> Application Structure <input type="text" value="Advanced"/> <input type="checkbox"/> Do not generate the main() Toolchain Folder Location <input type="text" value="C:\Users\parsa\Desktop\Riz\Project\CubeMX\Micro-Project\"/> Toolchain / IDE <input type="text" value="MDK-ARM"/> Min Version <input type="text" value="V5.32"/> <input type="checkbox"/> Generate Under Root
Code Generator	
Advanced Settings	Linker Settings Minimum Heap Size <input type="text" value="0x200"/> Minimum Stack Size <input type="text" value="0x400"/>
	Thread-safe Settings Cortex-M/MNS <input type="checkbox"/> Enable multi-threaded support Thread-safe Locking Strategy <input type="text" value="Default - Mapping suitable strategy depending on RTOS selection"/>
	Mcu and Firmware Package Mcu Reference <input type="text" value="STM32F401RETx"/> Firmware Package Name and Version <input type="text" value="STM32Cube_FW_F4_V1.27.0"/> <input checked="" type="checkbox"/> Use Default Firmware Location Firmware Relative Path <input type="text" value="C:\Users\parsa\STM32Cube\Repository\STM32Cube_FW_F4_V1.27.0"/> <a href="#">Browse</a>

Project

Code Generator

Advanced Settings

Univer Selector

Search (Ctrl+F)

RCC

GPIO

ADC

ADC1

USART

USART2

HAL

HAL

HAL

HAL

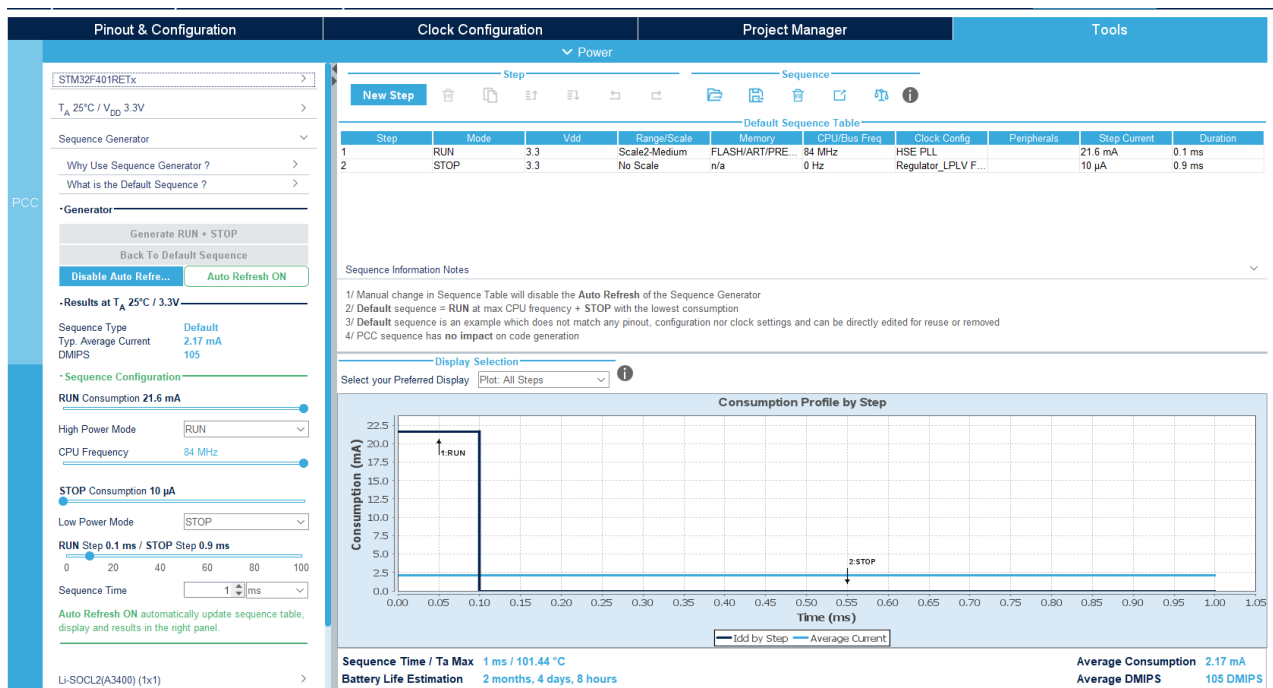
HAL

HAL

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Name	Do Not Generate Function Call	Visibility (Static)
<input checked="" type="checkbox"/>	1	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	2	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	3	MX_ADC1_Init	ADC1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	4	MX_USART2_UART_Init	USART2	<input type="checkbox"/>	<input checked="" type="checkbox"/>

:Tools •



در اینجا توضیحات مربوط به نرم افزار عه STM32CubeMX به پایان می رسد.

## :Keil

در اینجا به توضیحات کد های زده شده و برخی از فرض های در نظر گرفته شده می پردازیم.

### • Includes:

علاوه بر کد های تولید شده به وسیله نرم افزار عه STM32CubeMX، یک سری include هم ما در کد قرار دادیم.

```
#include <stdio.h>
```

```
#include <string.h>
```

این دو خط برای این می باشند که ما در کدمان قادر به استفاده از توابع عه string و نوشتن عه یک عدد بر روی یک string باشیم.  
(عموما برای تابع هایی که برای لاگ گرفتن یا همان نمایش بر روی virtual terminal با استفاده از usart استفاده می شوند.)

### • Defines:

#### ○ Coefficient:

این مقدار، در اصل برای هر فرکانس، محاسبه شده فرمولی می باشند که به صورت کامنت در بالای همین قسمت اشاره شده است.  
این فرمول را در قسمت زیر آورده ایم:

```
coefficient = 2 * (cos(2 * PI * (Freq / 8000.0)))
```

حاصل این فرمول را برای فرکانس های سطر و ستون و دوبرابر عه فرکانس های سطر و ستون (که این دوبرابر در هارمونی عه دوم استفاده می شود) حساب کرده ایم و مقدارشان را به صورت defines قرار داده ایم تا از محاسبه چند باره آنها بپرهیزیم.

#### ○ N یا sampling Number:

در این قسمت، N بزرگ را تعریف کرده ایم.

این مقدار N، در اصل برابر با تعداد دفعاتی است که ما ولتاژ عه ورودی را در یک آرایه ذخیره می کنیم تا در الگوریتم عه گوئرتزل، از آن استفاده کنیم.

همانطور که در کد مشخص است، مقدار این N را برابر با 114 (این عدد با آزمون و خطا و محاسبات بسیار بدست آمد) قرار داده ایم.

#### ○ سون سگمنت:

در این قسمت هایی را که مربوط به قسمت عه سون سگمنت می باشد، قرار داده ایم.

برای مثال، شماره بین a و ... و g را در این قسمت قرار داده ایم تا در ادامه در صورت نیاز از آن ها استفاده کنیم. (البته در ادامه از آن ها استفاده ای نشد)

### ○ Mask ها :

در این قسمت، Mask ها را تعریف کرده ایم تا در صورت نیاز، اعداد موردنیاز را برایمان ایجاد کنند. (البته در ادامه از این قسمت نیز استفاده نشد)

### ○ End Line :

در این قسمت، string ای را تعریف کرده ایم که با آن می توان دستور عه رفتن به خط بعد را به virtual terminal داد.

### • Variables ها :

#### ○ Samples :

در این قسمت، ولتاژ های ورودی عه نمونه برداری شده را ذخیره و نگهداری می کنیم. پس این متغیر در اصل یک آرایه ای به طول n (که مقدار N را برابر با 114 قرار داده ایم) می باشد و در ادامه داخل آن ولتاژ های ورودی را ذخیره می کنیم.

#### ○ seven\_segment :

در این قسمت، به ازای هر کاراکتر، هر کدام از پین های متصل به سون سگمنت باید مقدار خاصی را دریافت کنند، در این آرایه عه دو بعدی، آن مقدار ها را (اگر مثلاً کاراکتر عه تشخیص داده شده، در سطر اول و ستون دوم باشد، مقدار متناظر آن در این آرایه برای سون سگمنت، در ایندکس عه [1][0] قرار دارد) به output data register عه GPIOB، assign می کنیم.

پس این متغیر یک آرایه عه دو بعدی عه 4 در 4 می باشد.

اما مقدار عنصر های هر ایندکس آرایه و پر شدنشان در جدول زیر مشخص می باشد:

کاراکتر	a	b	c	d	e	f	g	عدد به باینری	عدد به هگز
0	1	1	1	1	1	1	0	011 1111	0x3F
1	0	1	1	0	0	0	0	000 0110	0x6
2	1	1	0	1	1	0	1	101 1011	0x5B
3	1	1	1	1	0	0	1	100 1111	0x4F
4	0	1	1	0	0	1	1	110 0110	0x66
5	1	0	1	1	0	1	1	110 1101	0x6D
6	1	0	1	1	1	1	1	111 1101	0x7D
7	1	1	1	0	0	0	0	000 0111	0x07
8	1	1	1	1	1	1	1	111 1111	0x7F
9	1	1	1	1	0	1	1	110 1111	0x6F
A	1	1	1	0	1	1	1	111 0111	0x77
B	0	0	1	1	1	1	1	111 1100	0x7C
C	0	0	0	1	1	0	1	101 1000	0x58
D	0	1	1	1	1	0	1	101 1110	0x5E
#	0	0	1	1	1	0	1	101 1100	0x5C
*	0	1	1	0	1	1	1	111 0110	0x76

- **:Functions**

- **:PrintString**

در این تابع، صرفاً یک String ورودی را با توجه به کانال USART و تابع هایی که STM32 در اختیارمان قرار داده، چاپ می کنیم.

- **:PrintNumber**

در این تابع، صرفاً یک عدد را دریافت می کنیم، سپس آن را با توجه به کتابخانه هایی که در ابتدای کد include کرده بودیم، به string تبدیل می کنیم و سپس آن را با توجه به تابعی که STM32 در اختیارمان قرار داده، چاپ می کنیم.

- **:Read\_ADC\_OSC**

از این تابع، برای خواندن که ولتاژ ورودی استفاده می کنیم.

- **:GetAudioVoltage**

از این تابع، برای گرفتن عه (خواندن) تعداد N تا از ورودی عه آنالوگ و تبدیل آن به ولتاژ عه ورودی و ذخیره کردنشان در آرایه samples استفاده می شود.

در حالت عادی، ما باید، صرفاً برای بدست آوردن عه ولتاژ ورودی، عددی که میکروکنترلر به ما می دهد را تقسیم بر 4095 و ضربدر 5 (ولتاژ منبع) می کردیم تا حاصل بدست بیاید، اما در اینجا، عددی که میکرو به ما می دهد، بین 1401 تا 1718 می باشد. برای همین، ابتدا عددی که میکرو به ما می دهد را منهای 1401 می کنیم و سپس آن را تقسیم بر بازه کل اعداد (  $1718 - 1401 = 317$  ) می کنیم.

- **:GortzelAlgorithm**

در این تابع، ابتدا coefficient عه محاسبه شده برای سیگنال را به عنوان عه ورودی دریافت می کنیم و سپس با توجه به فرمول های موجود برای الگوریتم عه گوئرتزل، آن را محاسبه می کنیم و سپس خروجی را برای ورودی متناظر باز می گردانیم.

- **:FindTone**

در این تابع، ابتدا مقدار خروجی عه الگوریتم عه گوئرتزل را برای هر یک از فرکانس ها (در اصل با توجه به توضیحات داده شده، برای coefficient عه محاسبه شده برای هر فرکانس) محاسبه می کنیم و سطر و ستونی که بیشترین مقدار را داشته باشد و پیدا می کنیم.

در همین هنگام این شرط را که مقدار این سطر و ستون از مینیمم مقدار ای که باید باشد تا اصلا خروجی برای سیگنال معتبر به حساب بیاید، بیشتر باشد، بیشتر است یا خیر، اگر بیشتر نباشد، اجرای تابع تمام می شود و در ترمینال عبارت `Under The Min` را چاپ می کنیم.

اما اگر از مینیمم مقدار بیشتر بود، سپس مقدار عه خروجی عه الگوریتم عه گوئرتزل را برای دوبرابر سیگنال ها (هارمونی دوم) محاسبه می کنیم و شرط هارمونی دوم را چک میکنیم.

سپس اگر جواب معتبر بود، مقدار عه سطر و ستون را چاپ می کنیم و همینطور مقدار عه `output data register` عه `GPIOB` را هم برابر با ایندکس متناظر عه آرایه دوبعدی عه `seven_segment` قرار می دهیم تا کاراکتر مورد نظر در سون سگمنت چاپ شود.

#### • تابع `main`:

در تابع `main` هم ابتدا قبل از `while(1)` ، با چاپ کاراکتر هایی بر روی ترمینال، یک دیلی ایجاد می کنیم (تایمینگ ها به شدت حساس می باشد و با ذره ای عوض کردن کد، احتمال دارد تایمینگ بهم بریزد) سپس هر سری ابتدا مقدار عه آرایه عه `samples` را پر می کنیم و سپس تابع `FindTone` را صدا می زنیم، سپس 220 (این عدد با آزمون و خطا های بسیار زیاد بدست آمد) میلی ثانیه، تاخیر ایجاد می کنیم.