

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH**

Nhóm 7:

CHU TIẾN TRỌNG

**THỰC TẬP FPT-UIT
WEEK 1**

**GIAO TIẾP GPIO TRÊN BOARD STM32F429I-DICS1
VÀ MAKEFILE**

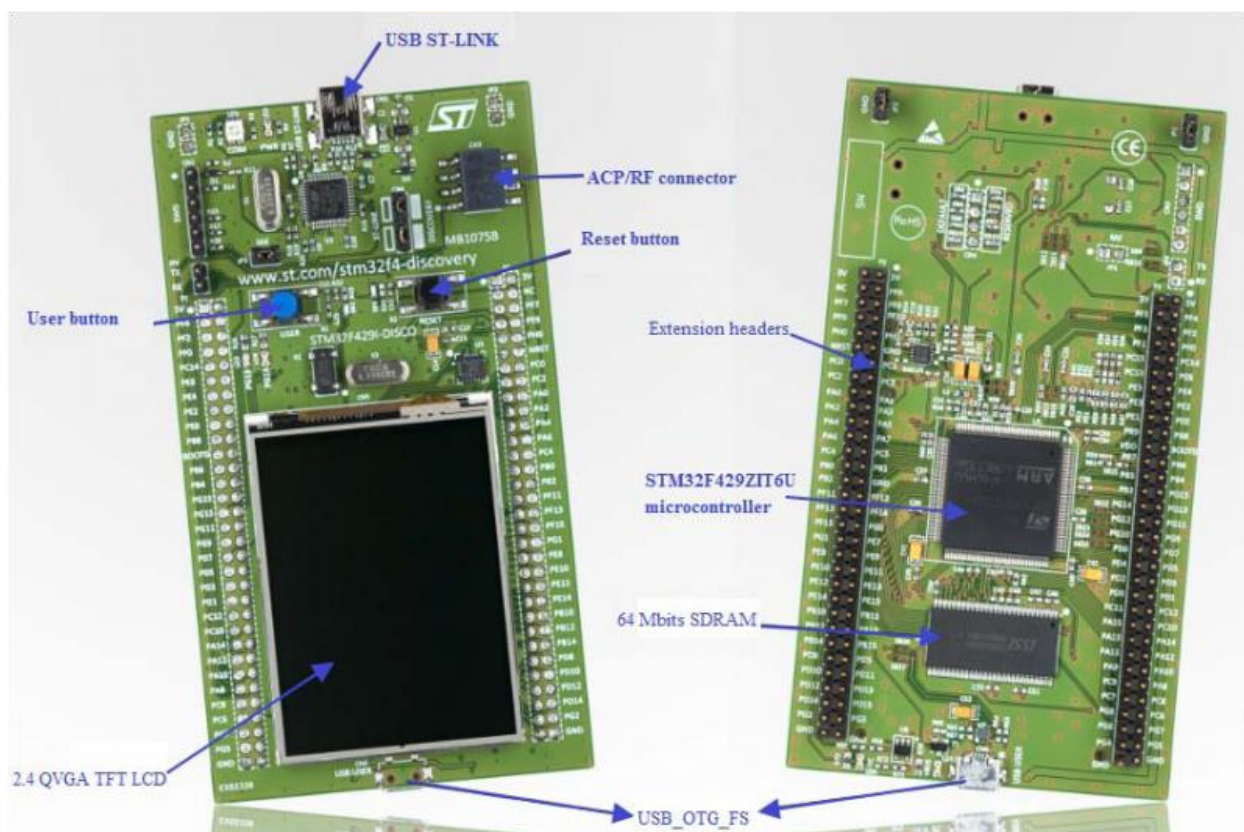
KỸ SƯ NGÀNH KỸ THUẬT MÁY TÍNH

TP. HỒ CHÍ MINH, 2021

MỤC LỤC

I.	TỔNG QUAN VỀ KIT STM32F429I-DISC1.....	3
II.	TÌM HIỂU VÀ CÀI ĐẶT CHƯƠNG TRÌNH ĐỂ BIÊN DỊCH CODE CHO DÒNG VI ĐIỀU KHIỂN STM32F4 BẰNG MAKEFILE.....	4
A.	Tìm hiểu về STM32Cube IDE	4
1.	Tổng quát về STM32Cube IDE	4
2.	Tất cả chức năng.....	5
B.	Cài đặt STM32CubeIDE.....	6
1.	STM32CubeIDE là 1 tools miễn phí của ST, vào link sau để download:	6
2.	Click và GetSoftware, đăng kí 1 tài khoản miễn phí bằng gmail và tải về.....	6
3.	Thực hiện cài đặt theo hướng dẫn của phần mềm.....	6
III.	SOẠN CHƯƠNG TRÌNH BẬT TẮT GPIO CHO VI ĐIỀU KHIỂN STM32F429.....	11
1.	Thư mục chương trình	11
2.	Main	11
3.	MakeFlie	16
IV.	TÌM HIỂU VỀ CÁCH VIẾT MAKEFIE	21
A.	MAKEFILE LÀ GÌ?.....	21
B.	CẤU TRÚC MỘT PROJECT	21
1.	Cấu trúc và sự phụ thuộc của project có thể được biểu diễn bằng một DAG (Directed Acyclic Graph).....	21
2.	Cấu trúc makefile.....	23
3.	Thứ tự thực hiện:	23
V.	THAM KHẢO:.....	23

I. TỔNG QUAN VỀ KIT STM32F429I-DISC1

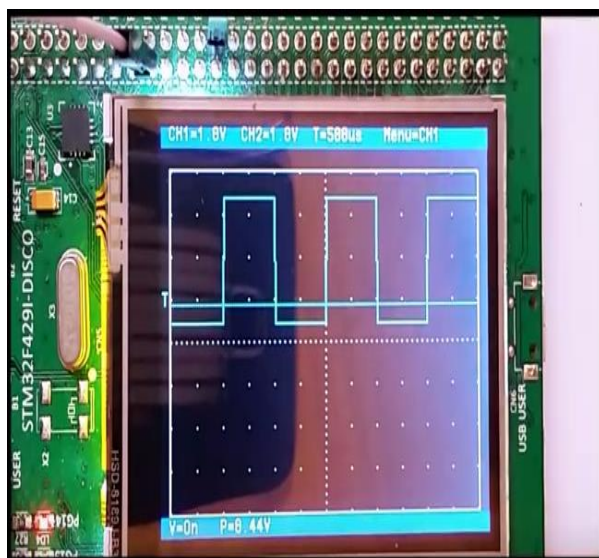


- Sau đây là các thông số cơ bản của board STM32F429I-DISC1

Vi xử lý STM32F429ZIT6: <ul style="list-style-type: none"> 2 Mbytes bộ nhớ Flash 256 Kbytes RAM 64 Mbit SDRAM 	<ul style="list-style-type: none"> Bộ nhớ Flash có dung lượng lớn, cho phép lưu trữ các chương trình có khối lượng lớn
ST-LINK/V2-B (mbed -enabled)	<ul style="list-style-type: none"> Cho phép nạp code và debug mà không cần phải mua thiết bị rời
Chức năng của cổng USB: <ul style="list-style-type: none"> Cổng debug Cổng COM ảo Interface giao tiếp trực tiếp với thiết bị lưu trữ Flash 	<ul style="list-style-type: none"> Vừa dùng để kết nối board vào máy tính (thực hiện nạp code và debug) mà còn dùng để cấp nguồn cho board
Bộ nguồn được cấp từ: <ul style="list-style-type: none"> Cổng USB Nguồn ngoài (nằm trong khoảng từ 3V – 5V) 	

Màn hình LCD: <ul style="list-style-type: none"> Kích thước 2.4’’ 262K màu RGB 240x320 dots Cảm ứng 	- Cho phép lập trình ứng dụng có giao diện đồ họa.
Con quay hồi chuyển	- Cho phép board cảm nhận về thế giới xung quanh theo trục tọa độ XYZ
Cổng kết nối USB-OTG	- Cho phép cắm các thiết bị lưu trữ ngoài như thẻ nhớ flash hoặc USB
Có nhiều ví dụ minh họa, có hỗ trợ thư viện	- Cung cấp các hàm để làm việc với thiết bị ngoại vi - Có nhiều ví dụ để tham khảo

- Một số dự án thực tế:



Máy oscilloscope



Chơi game

II. TÌM HIỂU VÀ CÀI ĐẶT CHƯƠNG TRÌNH ĐỂ BIÊN DỊCH CODE CHO DÒNG VI ĐIỀU KHIỂN STM32F4 BẰNG MAKEFILE

A. Tìm hiểu về STM32Cube IDE

1. Tổng quát về STM32Cube IDE

STM32CubeIDE

All-in-one STM32 development tool

TrueSTUDIO[®] for STM32



STM32
CubeIDE

- STM32CubeIDE là một bộ công cụ All-in-one hỗ trợ cho việc lập trình vi điều khiển STM32 bằng ngôn ngữ C/C++.
- Nó được tích hợp với công cụ STM32CubeMX cho phép cấu hình chân và khởi tạo code ban đầu cho dự án mới một cách nhanh chóng, với bộ biên dịch GCC cho Arm và GDB hỗ trợ việc gỡ lỗi (Debug).
- Được phát triển dựa trên framework ECLIPSE[™]/CDT , vì thế STM32CubeIDE còn hỗ trợ tích hợp hàng trăm các plugins có sẵn của Eclipse[™] IDE.

2. Tất cả chức năng

- STM32CubeIDE tích hợp cấu hình STM32 và các chức năng tạo dự án từ STM32CubeMX để cung cấp trải nghiệm công cụ tất cả trong một và tiết kiệm thời gian cài đặt và phát triển.
- Sau khi lựa chọn MCU hoặc MPU STM32 trống, hoặc bộ vi điều khiển hoặc bộ vi xử lý được cấu hình sẵn từ việc chọn bảng hoặc chọn một ví dụ, dự án sẽ được tạo và mã khởi tạo được tạo.
- Bất kỳ lúc nào trong quá trình phát triển, người dùng có thể quay lại quá trình khởi tạo và cấu hình thiết bị ngoại vi hoặc phần mềm trung gian và tạo lại mã khởi tạo mà không ảnh hưởng đến mã người dùng.
- STM32CubeIDE bao gồm các bộ phân tích xây dựng và ngăn xếp cung cấp cho người dùng thông tin hữu ích về tình trạng dự án và các yêu cầu về bộ nhớ.
- STM32CubeIDE cũng bao gồm các tính năng gỡ lỗi tiêu chuẩn và nâng cao bao gồm chế độ xem thanh ghi lỗi CPU, bộ nhớ và thanh ghi ngoại vi, cũng như đồng hồ biến trực tiếp, giao diện Serial Wire Viewer hoặc bộ phân tích lỗi.

B. Cài đặt STM32CubeIDE

1. STM32CubeIDE là 1 tools miễn phí của ST, vào link sau để download:

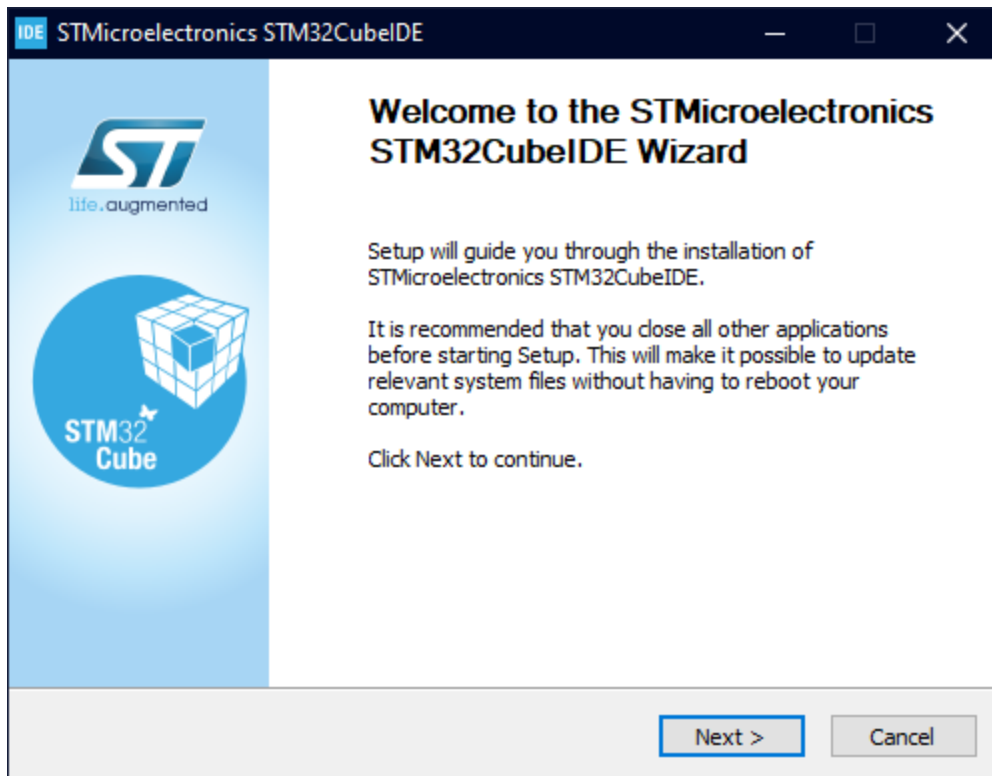
[STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics](#)

2. Click vào GetSoftware, đăng kí 1 tài khoản miễn phí bằng gmail và tải về
Get Software

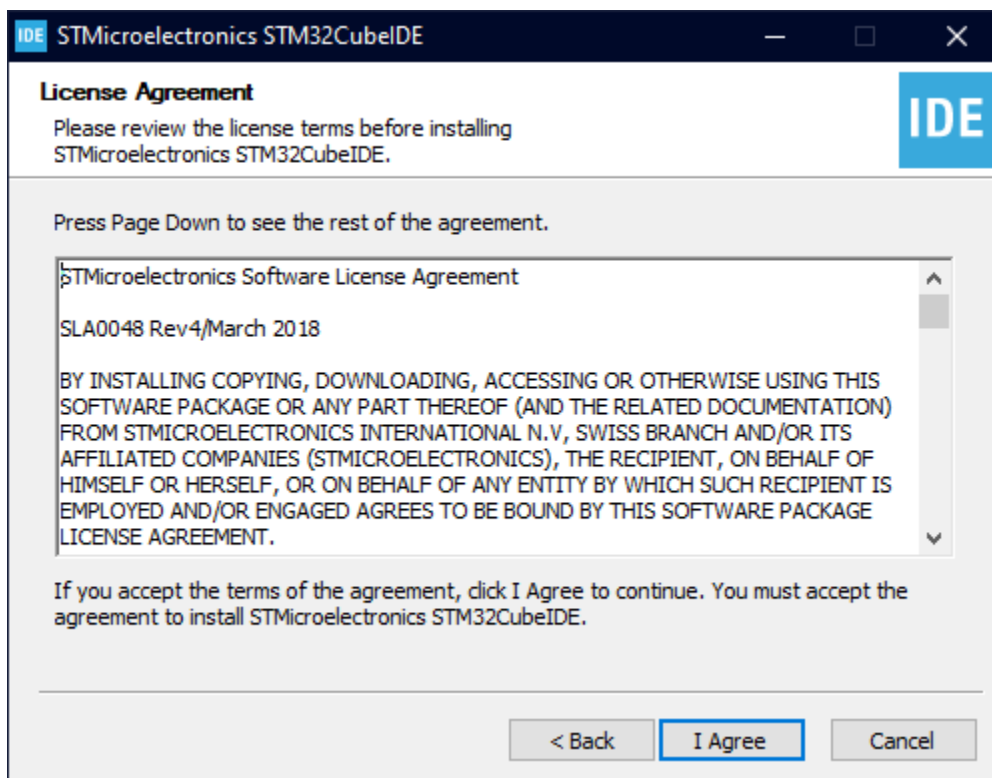
	Part Number ▲	General Description	Software Version	Download	Previous versions
+	STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.6.1	Get Software	Select version ▼
+	STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.6.1	Get Software	Select version ▼
+	STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.6.1	Get Software	Select version ▼
+	STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.6.1	Get Software	Select version ▼
+	STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.6.1	Get Software	Select version ▼

3. Thực hiện cài đặt theo hướng dẫn của phần mềm.

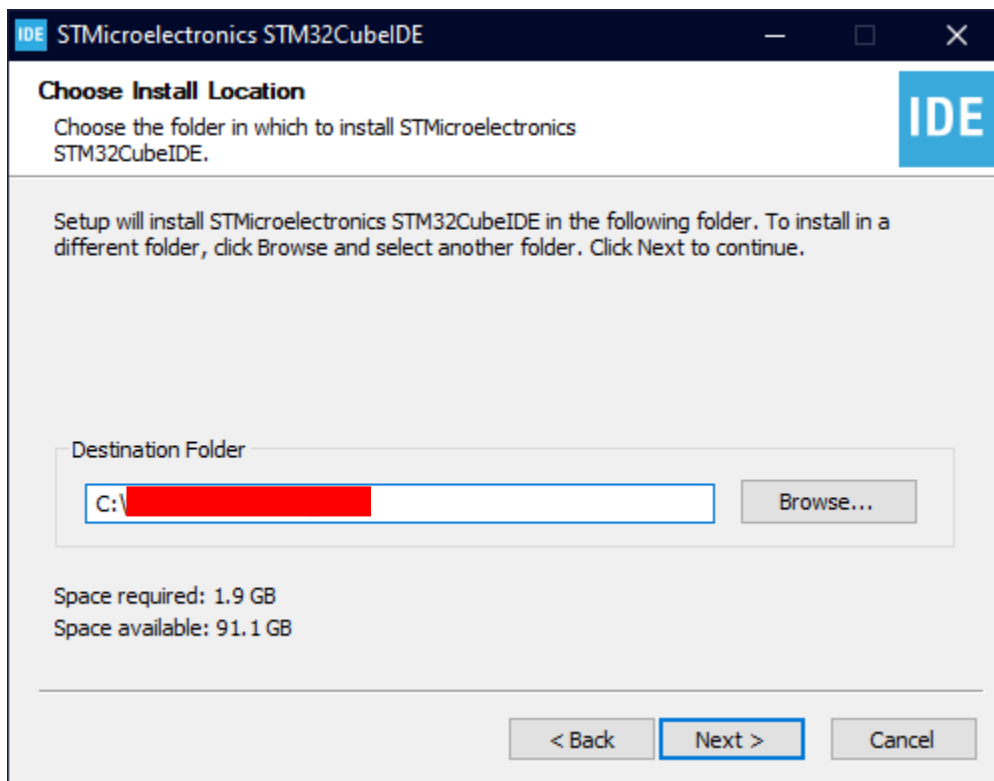
- Sau khi tải về bộ cài theo đúng hệ điều hành mà bạn đang sử dụng, bạn hãy giải nén và chạy file cài đặt:



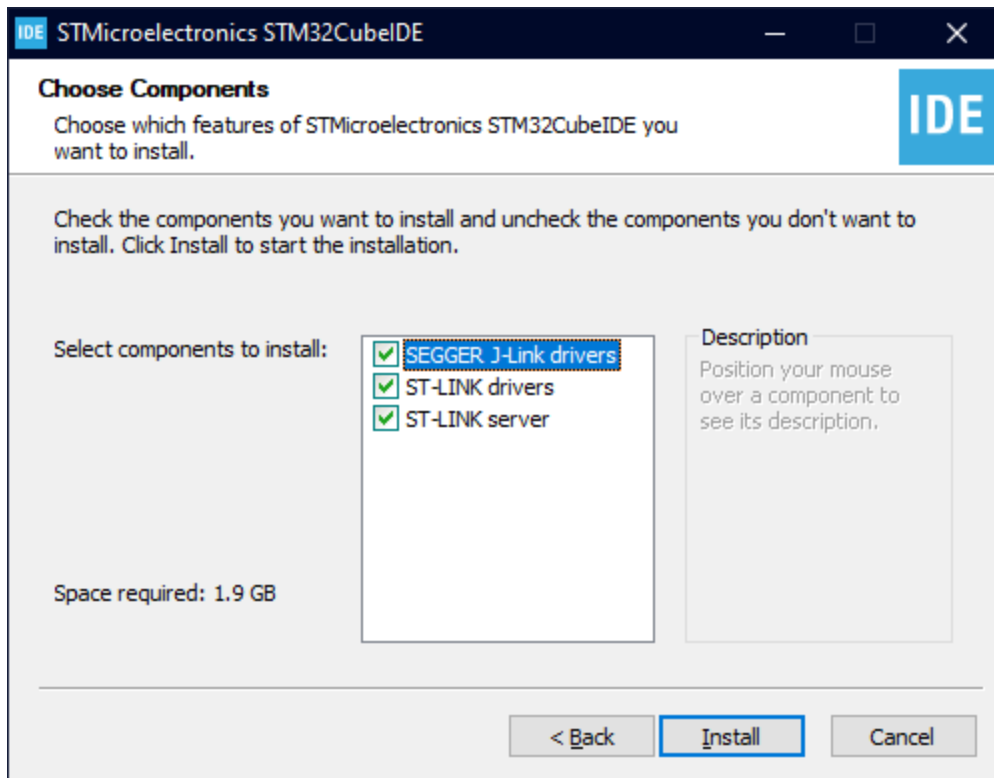
- Chọn **Next** để qua bước tiếp theo



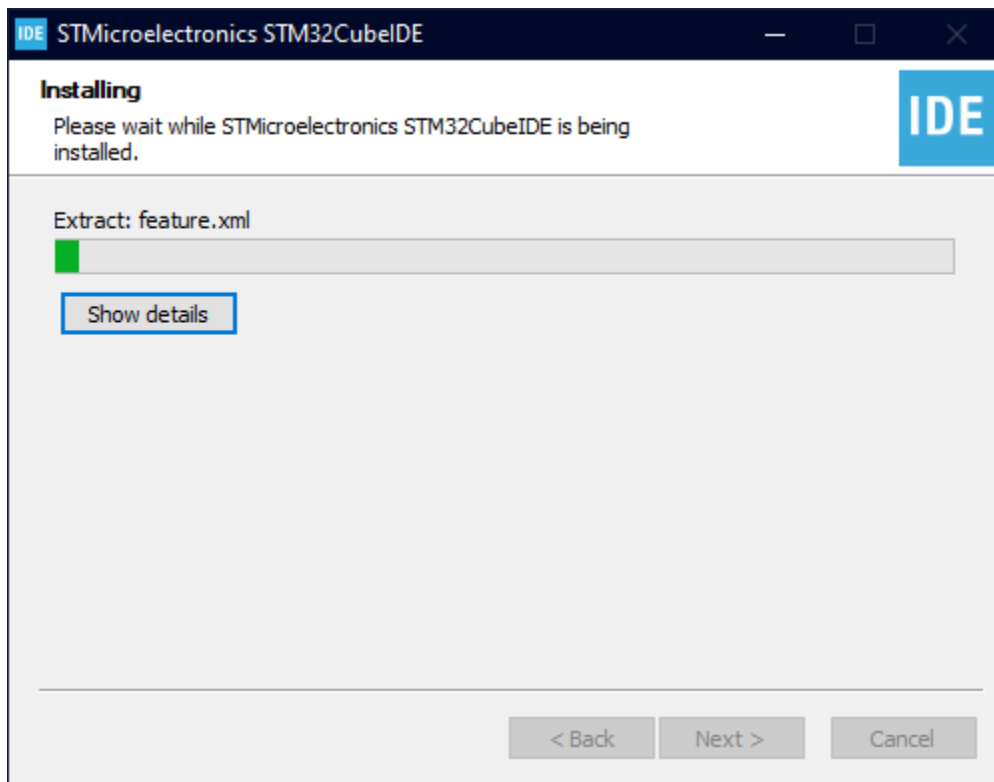
- Trang điều khoản sử dụng, bạn phải đồng ý với điều khoản trước khi tiếp tục. Bấm **I Agree**



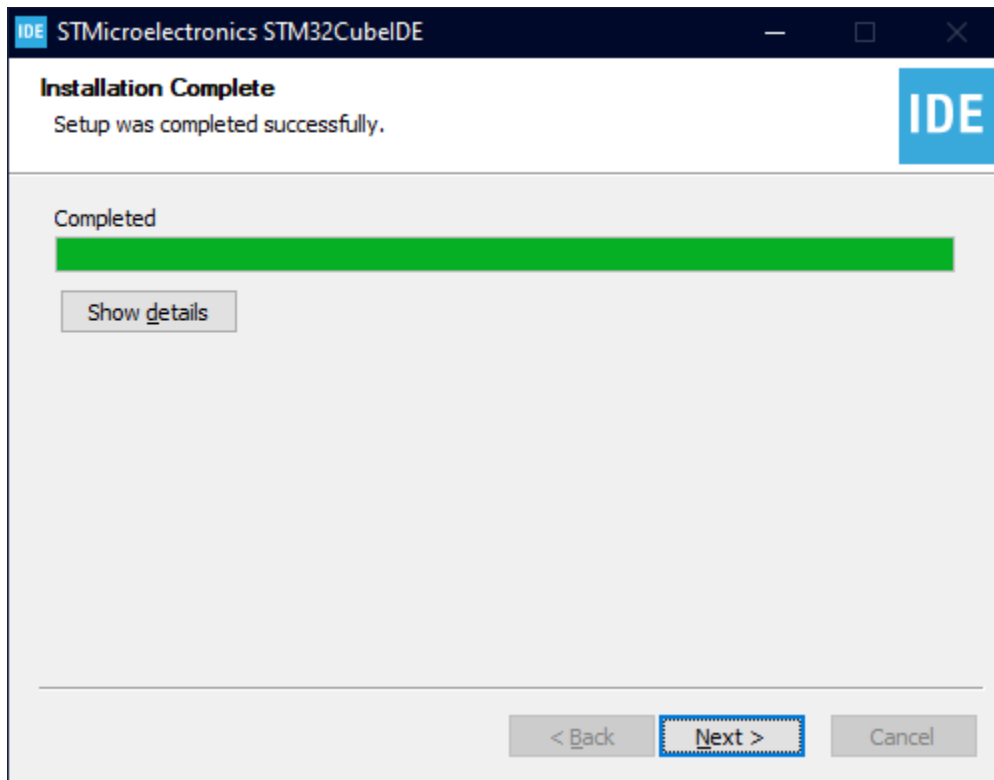
- Chọn nơi cài đặt. Mặc định sẽ nằm ở ổ đĩa cài hệ điều hành. Bấm **Next** để qua bước tiếp theo



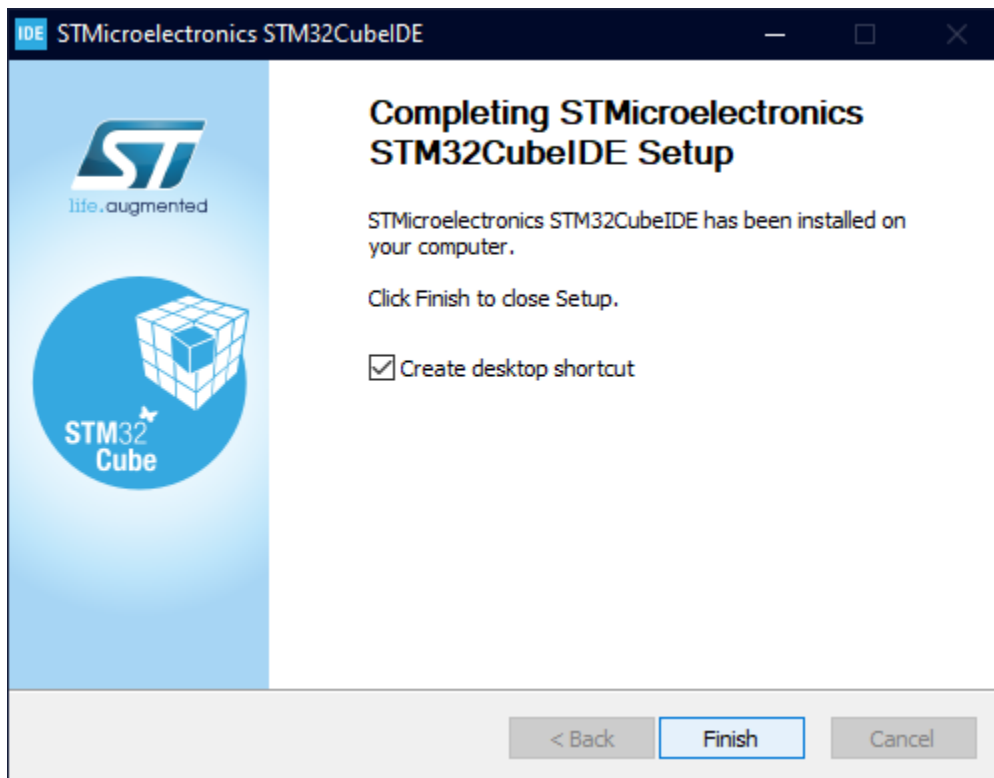
- Lựa chọn cài đặt thêm một số Drivers, sau đó bấm **Install**



- Đợi cho quá trình cài đặt hoàn tất













- Bấm **Next** để tới bước cuối



- Cài đặt hoàn tất, nếu bạn muốn tạo đường dẫn ngoài màn hình thì tích vào ô **Create desktop shortcut** sau đó bấm **Finish**.

III. SOẠN CHƯƠNG TRÌNH BẬT TẮT GPIO CHO VI ĐIỀU KHIỂN STM32F429

1. Thư mục chương trình

	.metadata	22/05/2021 8:14 SA	File folder	
	.vscode	22/05/2021 8:05 SA	File folder	
	build	22/05/2021 9:18 SA	File folder	
	Core	22/05/2021 8:04 SA	File folder	
	Drivers	22/05/2021 8:04 SA	File folder	
	Week1	22/05/2021 8:54 SA	File folder	
	.mxproject	22/05/2021 8:04 SA	MXPROJECT File	7 KB
	Makefile	22/05/2021 8:04 SA	File	6 KB
	startup_stm32f429xx.s	19/05/2021 4:47 CH	Assembler Source	25 KB
	STM32F429ZITx_FLASH.ld	22/05/2021 8:04 SA	LD File	7 KB
	Week1.ioc	22/05/2021 8:04 SA	STM32CubeMX	5 KB

2. Main

a Code:

```

/* USER CODE BEGIN Header */
/**
 * *****
 * @file      : main.c
 * @brief     : Main program body
 * *****
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *
 *         opensource.org/licenses/BSD-3-Clause
 *
 * *****
 */

```

```

    */
/* USER CODE END Header */
/* Includes -----*/
#include "main.h"

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */
/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize all configured peripherals */
    MX_GPIO_Init();

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
        HAL_Delay(250);

        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
        {
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
        }
        else
        {
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
        }
    }
}
/**

```

```

* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 8;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }
    /** Initializes the CPU, AHB and APB buses clocks
    */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                  |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
    {
        Error_Handler();
    }
}

/**
* @brief GPIO Initialization Function
* @param None

```

```

    * @retval None
    */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOG_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13|GPIO_PIN_14, GPIO_PIN_RESET);

    /*Configure GPIO pin : PA0 */
    GPIO_InitStructure.Pin = GPIO_PIN_0;
    GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*Configure GPIO pins : PG13 PG14 */
    GPIO_InitStructure.Pin = GPIO_PIN_13|GPIO_PIN_14;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOG, &GPIO_InitStructure);

    /* EXTI interrupt init*/
    HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI0_IRQn);
}

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */
    __disable_irq();
    while (1)
    {
    }
}

```

```

/* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 *        where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/***** (C) COPYRIGHT STMicroelectronics *****END OF FILE*****/

```

b Giải thích code

Code:

```

while (1)
{
    /* USER CODE END WHILE */
    HAL_GPIO_TogglePin(GPIOG,GPIO_PIN_13);
    #Thay đổi trạng thái HIGH->LOW hoặc LOW->HIGH tại chân PG13
    ##Theo thiết kế của STM32F429I-DISC1, 2 chân LED được gán tại 2 chân
    PG13|PG14

    HAL_Delay(250);
    #Dừng chương trình trong 250ms

    if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
    #Đọc giá trị tại chân PA0 -> Chân User Button
    {
        HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);
        #Ghi giá trị 1 tại chân PG14 (Bật sáng đèn)
    }
}

```



```

else
{
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET);
    #Ghi giá trị 0 tại chân PG14 (Tắt đèn)
}
}
#####

```

Sau khi nạp code cho Kit cần bấm nút Reset để bắt đầu

-> Đèn Led Xanh (PG13) chớp tắt liên tục (250ms)

-> Khi bấm nút User Button sẽ kích hoạt Ngắt (Interupt), và đèn Led Đỏ (PG14) sẽ sáng lên.

-> Khi thả nút User Button thì sẽ kích hoạt ngắt và đèn Led Đỏ sẽ tắt.

3. MakeFlie

Code:

```

#####
# File automatically-generated by tool: [projectgenerator] version: [3.13.0-B3] date: [Sat Ma
#####

# -----
# Generic Makefile (based on gcc)
#
# ChangeLog :
#   2017-02-10 - Several enhancements + project update mode
#   2015-07-22 - first version
# -----

#####
# target
#####
TARGET = Week1

#####
# building variables
#####
# debug build?
DEBUG = 1
# optimization
OPT = -Og

```

```
#####
# paths
#####
# Build path
BUILD_DIR = build

#####
# source
#####
# C sources
C_SOURCES = \
Core/Src/main.c \
Core/Src/stm32f4xx_it.c \
Core/Src/stm32f4xx_hal_msp.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_tim.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_tim_ex.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_rcc.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_rcc_ex.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_flash.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_flash_ex.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_flash_ramfunc.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_gpio.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_dma_ex.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_dma.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_pwr.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_pwr_ex.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_cortex.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal.c \
Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal_exti.c \
Core/Src/system_stm32f4xx.c

# ASM sources
ASM_SOURCES = \
startup_stm32f429xx.s

#####
# binaries
#####
PREFIX = arm-none-eabi-
# The gcc compiler bin path can be either defined in make command via GCC_PATH variable (> ma
# either it can be added to the PATH environment variable.
ifdef GCC_PATH
```

```

CC = $(GCC_PATH)/$(PREFIX)gcc
AS = $(GCC_PATH)/$(PREFIX)gcc -x assembler-with-cpp
CP = $(GCC_PATH)/$(PREFIX)objcopy
SZ = $(GCC_PATH)/$(PREFIX)size
else
CC = $(PREFIX)gcc
AS = $(PREFIX)gcc -x assembler-with-cpp
CP = $(PREFIX)objcopy
SZ = $(PREFIX)size
endif
HEX = $(CP) -O ihex
BIN = $(CP) -O binary -S

#####
# CFLAGS
#####
# cpu
CPU = -mcpu=cortex-m4

# fpu
FPU = -mfpu=fpv4-sp-d16

# float-abi
FLOAT-ABI = -mfloat-abi=hard

# mcu
MCU = $(CPU) -mthumb $(FPU) $(FLOAT-ABI)

# macros for gcc
# AS defines
AS_DEFS =

# C defines
C_DEFS = \
-DUSE_HAL_DRIVER \
-DSTM32F429xx

# AS includes
AS_INCLUDES =

# C includes
C_INCLUDES = \
-ICore/Inc \
-IDrivers/STM32F4xx_HAL_Driver/Inc \

```

```

-IDrivers/STM32F4xx_HAL_Driver/Inc/Legacy \
-IDrivers/CMSIS/Device/ST/STM32F4xx/Include \
-IDrivers/CMSIS/Include

# compile gcc flags
ASFLAGS = $(MCU) $(AS_DEFS) $(AS_INCLUDES) $(OPT) -Wall -fdata-sections -ffunction-sections

CFLAGS = $(MCU) $(C_DEFS) $(C_INCLUDES) $(OPT) -Wall -fdata-sections -ffunction-sections

ifeq ($(DEBUG), 1)
CFLAGS += -g -gdwarf-2
endif

# Generate dependency information
CFLAGS += -MMD -MP -MF"$(@:%.o=%.d)"

#####
# LDFLAGS
#####
# link script
LDSCRIPT = STM32F429ZITx_FLASH.ld

# libraries
LIBS = -lc -lm -lnosys
LIBDIR =
LDFLAGS = $(MCU) -specs=nano.specs -T$(LDSCRIPT) $(LIBDIR) $(LIBS) -Wl,-Map=$(BUILD_DIR)/$(TARGET).map -fdata-sections
sections

# default action: build all
all: $(BUILD_DIR)/$(TARGET).elf $(BUILD_DIR)/$(TARGET).hex $(BUILD_DIR)/$(TARGET).bin

#####
# build the application
#####
# list of objects
OBJECTS = $(addprefix $(BUILD_DIR)/,$(notdir $(C_SOURCES:.c=.o)))
vpath %.c $(sort $(dir $(C_SOURCES)))
# list of ASM program objects
OBJECTS += $(addprefix $(BUILD_DIR)/,$(notdir $(ASM_SOURCES:.s=.o)))
vpath %.s $(sort $(dir $(ASM_SOURCES)))

```

```

$(BUILD_DIR)/%.o: %.c Makefile | $(BUILD_DIR)
    $(CC) -c $(CFLAGS) -Wa,-a,-ad,-alms=$(BUILD_DIR)/$(notdir $(<:.c=.lst)) $< -o $@

$(BUILD_DIR)/%.o: %.s Makefile | $(BUILD_DIR)
    $(AS) -c $(CFLAGS) $< -o $@

$(BUILD_DIR)/$(TARGET).elf: $(OBJECTS) Makefile
    $(CC) $(OBJECTS) $(LDFLAGS) -o $@
    $(SZ) $@

$(BUILD_DIR)/%.hex: $(BUILD_DIR)/%.elf | $(BUILD_DIR)
    $(HEX) $< $@

$(BUILD_DIR)/%.bin: $(BUILD_DIR)/%.elf | $(BUILD_DIR)
    $(BIN) $< $@

$(BUILD_DIR):
    mkdir $@

#####
# clean up
#####
clean:
    -rm -fR $(BUILD_DIR)

#####
# dependencies
#####
-include $(wildcard $(BUILD_DIR)/*.d)

# *** EOF ***

```

Kết quả sau khi Build Makefile:

```

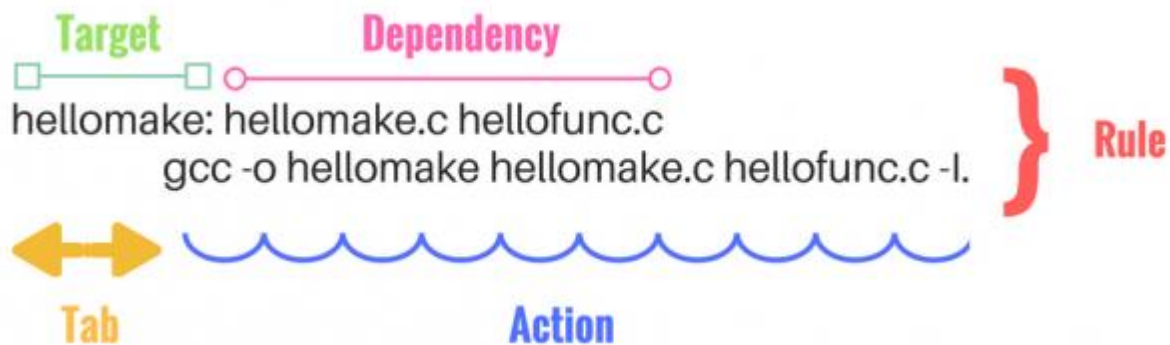
n stm32f4xx_hal_gpio.exh
h stm32f4xx_hal_gpio.h
h stm32f4xx_hal_pwm.exh
h stm32f4xx_hal_pwm.h
h stm32f4xx_hal_rcc.exh
h stm32f4xx_hal_rcc.h
h stm32f4xx_hal_tim.exh
h stm32f4xx_hal_tim.h
h stm32f4xx_hal.h
> Src
> Week1
> .mxproject
> Makefile
> startup_stm32f429xx.s
> STM32F429ZITX_FLASH.ld
> Week1.joc
> OUTLINE
STM32CubeMX - Configure 25 0
Ln 190, Col 14 (5328 selected) Tab Size: 4 UTF-8 CRLF makefile

```

IV. TÌM HIỂU VỀ CÁCH VIẾT MAKEFIE

A. MAKEFILE LÀ GÌ?

Makefile là một tập script chứa các thông tin : Cấu trúc project (file, dependence), và các lệnh để tạo ra file. Các file được tạo ra gọi là Target, các file phụ thuộc được gọi là Dependence , lệnh được dùng để compile source code được gọi là Action. 1 cú pháp bao gồm Target,dependence , Action được gọi là 1 Rule.



B. CẤU TRÚC MỘT PROJECT

1. Cấu trúc và sự phụ thuộc của project có thể được biểu diễn bằng một DAG (Directed Acyclic Graph).

- Thí dụ:
 - Chương trình chứa 3 file: **main.c**, **sum.c**, **sum.h**
 - File **sum.h** được dùng bởi cả 2 file **main.c** và **sum.c**
 - File thực thi là **sum**
- Source code như sau :

Sum.h:

```

#ifndef SUM_H_
#define SUM_H_
#include <stdio.h>
int sum(int a, int b);
#endif /* SUM_H_ */

```

Sum.c:

```

#include "sum.h"

int sum(int a, int b){
    return (a+b);
}

```

Main.c:

```

#include <stdio.h>
#include "sum.h"

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

    int x;
    x= sum(1, 2);
    printf("x = %d \n", x);

    return 1;
}

```

Makefile:

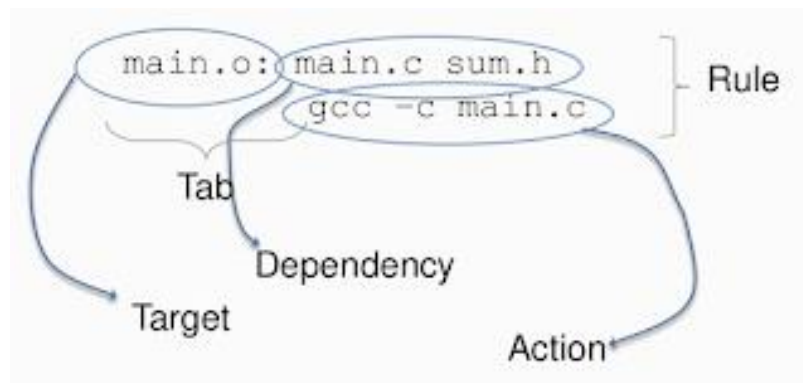
```

sum: main.o sum.o
    gcc -o sum main.o sum.o
main.o: main.c sum.h
    gcc -c main.c
sum.o: sum.c sum.h
    gcc -c sum.c

```

- Lưu ý : Mọi khoảng trắng trong makefile đều được thực hiện bởi nút Tab (nút phía trên nút Caps Lock) . Không được dùng phím Kháng cách (Space).

2. Cấu trúc makefile



3. Thứ tự thực hiện:

Khi bạn thực hiện lệnh make, chương trình make sẽ nhảy đến target đầu tiên là sum với mục đích để tạo ra nó, để làm được điều đó make đi kiểm tra lần lượt (từ trái qua phải: main.o -> sum.o) xem các dependency của sum đã tồn tại chưa. Dependency đầu tiên là main.o chưa có, cần phải tìm rule nào đó mà ở đó main.o đóng vai trò là target, make tìm ra rule thứ 2 và nó nhảy đến thực hiện rule thứ 2 để tạo ra main.o (lưu ý khi nó chạy rule 2 thì cũng giống y như khi chạy rule đầu tiên, có thể coi như là đệ quy). Sau khi tạo ra main.o, make trở về rule 1 để tiến hành kiểm tra tiếp xem dependency thứ hai là sum.o đã tồn tại chưa, sum.o chưa có vì thế make tiến hành các bước tương tự như đối với main.o. Sau khi tất cả các dependency được tạo ra, make mới có thể tạo ra file chạy cuối cùng là sum.

- Make thực hiện theo nguyên tắc / thứ tự như sau:

+ Tạo ra các file object trước (main.o, sum.o)

+ Tạo ra chương trình nhị phân cuối cùng từ các file object đã được tạo ra trước đó (sum)

- Như vậy, thứ tự làm việc của make như sau :

Rule1 (check **main.o**) => Rule2 (sinh **main.o**) => Rule1 (check **sum.o**) => Rule3 (sinh **sum.o**) => complete Rule1 (tạo ra file **main**).

V. THAM KHẢO:

[32F429IDISCOVERY - Discovery kit with STM32F429ZI MCU * New order code STM32F429I-DISC1 \(replaces STM32F429I-DISCO\) - STMicroelectronics](#)

[STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics](#)

[Makefile \(Part 1\) \(eslinuxprogramming.blogspot.com\)](#)