LAB-2 REPORT

Subject: **Embedded Hardware Design**

Subject Code: **EL203**

Members: 1) Shantanu Tyagi **(201801015)**
2) Nikhil Mehta **(201801030)**
3) Sudhanshu Mishra **(201801114)**
4) Bhavana Kolli **(201701082)**

Installation of the Software

Download Keil Microcontroller Development Kit from the link as provided
https://www2.keil.com/mdk5

Environment Setup

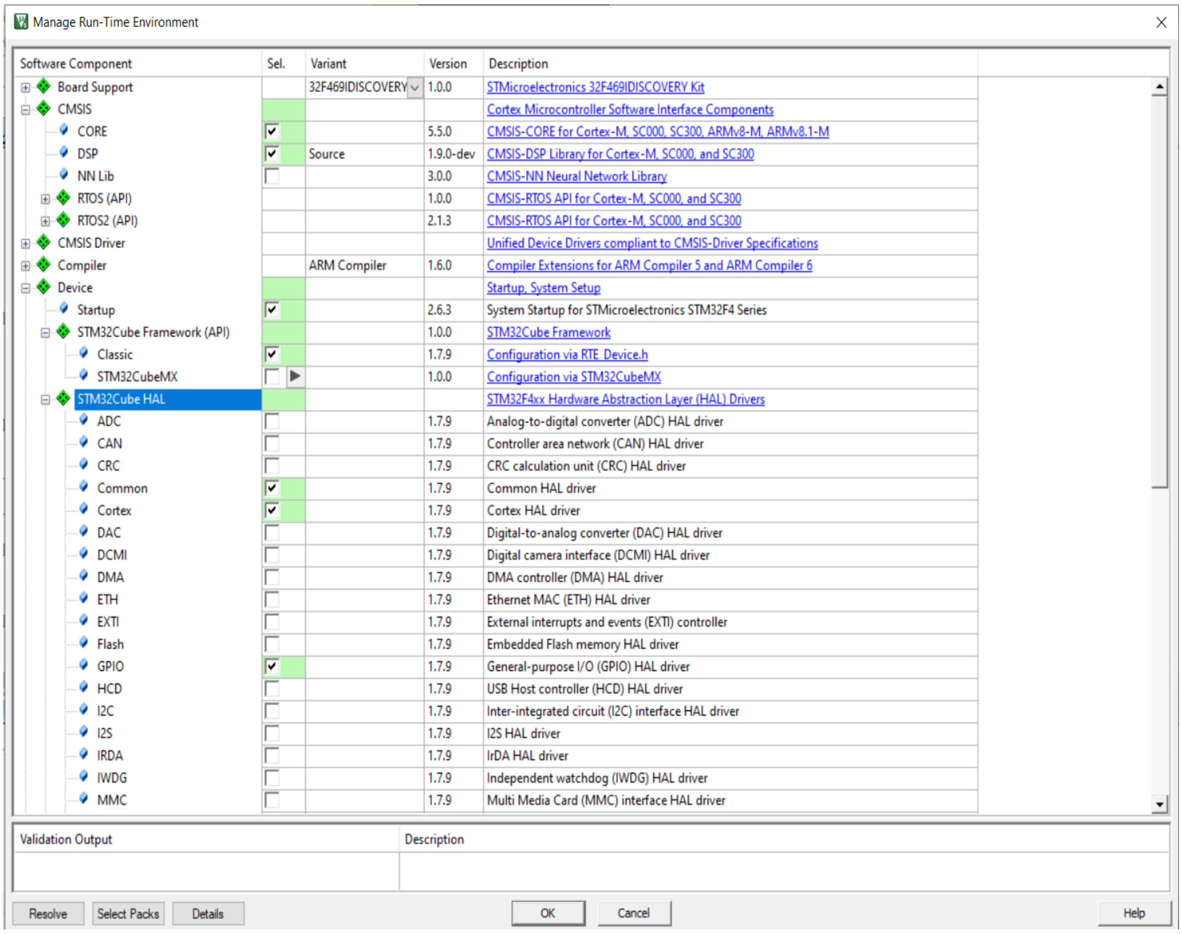
In the IDE, install the Keil::STM32F4xx DFP Package from Pack Manager.

1) Create a new Project and rename it using the appropriate nomenclature.

   1.1.1.1. Set STM32F407VGTx as the target device under the device header

   1.1.1.2. Modify the options for the target device:

2.



Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Board Support		32F469IDISCOVERY	1.0.0	STMicroelectronics 32F469IDISCOVERY Kit
CMSIS				Cortex Microcontroller Software Interface Components
CORE	✓		5.5.0	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M
DSP	✓	Source	1.9.0-dev	CMSIS-DSP Library for Cortex-M, SC000, and SC300
NN Lib	☐		3.0.0	CMSIS-NN Neural Network Library
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.3	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Startup	✓		2.6.3	System Startup for STMicroelectronics STM32F4 Series
STM32Cube Framework (API)			1.0.0	STM32Cube Framework
Classic	✓		1.7.9	Configuration via RTE_Device.h
STM32CubeMX	☐		1.0.0	Configuration via STM32CubeMX
STM32Cube HAL				STM32F4xx Hardware Abstraction Layer (HAL) Drivers
ADC	☐		1.7.9	Analog-to-digital converter (ADC) HAL driver
CAN	☐		1.7.9	Controller area network (CAN) HAL driver
CRC	☐		1.7.9	CRC calculation unit (CRC) HAL driver
Common	✓		1.7.9	Common HAL driver
Cortex	✓		1.7.9	Cortex HAL driver
DAC	☐		1.7.9	Digital-to-analog converter (DAC) HAL driver
DCMI	☐		1.7.9	Digital camera interface (DCMI) HAL driver
DMA	☐		1.7.9	DMA controller (DMA) HAL driver
ETH	☐		1.7.9	Ethernet MAC (ETH) HAL driver
EXTI	☐		1.7.9	External interrupts and events (EXTI) controller
Flash	☐		1.7.9	Embedded Flash memory HAL driver
GPIO	✓		1.7.9	General-purpose I/O (GPIO) HAL driver
HCD	☐		1.7.9	USB Host controller (HCD) HAL driver
I2C	☐		1.7.9	Inter-integrated circuit (I2C) interface HAL driver
I2S	☐		1.7.9	I2S HAL driver
IRDA	☐		1.7.9	IrDA HAL driver
IWDG	☐		1.7.9	Independent watchdog (IWDG) HAL driver
MMC	☐		1.7.9	Multi Media Card (MMC) interface HAL driver

2.1. Create a new group under the project

2.2. Make modifications for the following additional options for the target device:

   2.2.1. Set ARM compiler version 6 under the Target -> Code Generation header

2.2.2. Select the Simulator radio button and load the KEIL_STM.ini file as an initialization file under the Debug header

2.3. Add the main.c and sinewave.c file under the created group in the project with the relevant header file (#include "stm32f4xx_hal.h",#include "arm_math.h")

Running the Simulation

1 To run the code, first we have to compile the main.c file. This is done by clicking on the build button in Keil IDE.
2 Once the build is complete, we are prompted with errors and we need to fix them.
3 After fixing the bugs we move to the debug section and run the code to get the results.
4 To see the results, we can either see the result value or check out the graph for the simulation.
5 We can see the values of the variables directly by clicking on the variable and then on 'add to watch'.
6 In order to get the graph, we select the variable and click on 'analyze'.
7 After checking the output, we will have to close the debug session to make further changes in the code. Then follow the steps mentioned above again to see the new output.

CODE:

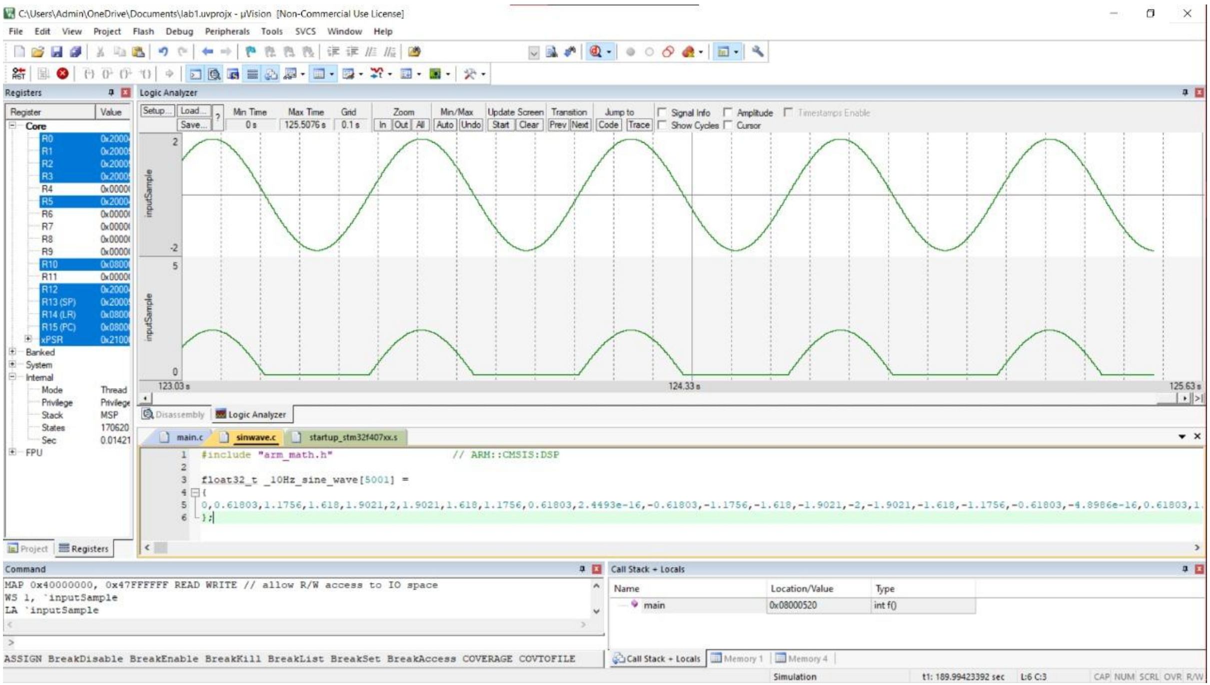
```
% MATLAB
clear all;

freq = 5;
Amp = 2;
ts = 0.01;
T = 5;

t=0:ts:T;
y = Amp*sin(2*pi*freq*t);
plot(t,y)
csvwrite("_5Hz_sine_wave.txt",y)
```

```
// main.c
# include "stm32f4xx_hal.h"                // Keil::Device:STM32Cube
HAL:Common
# include "arm_math_f16.h"                 // ARM::CMSIS:DSP

# define SIG_LENGTH 5001
```

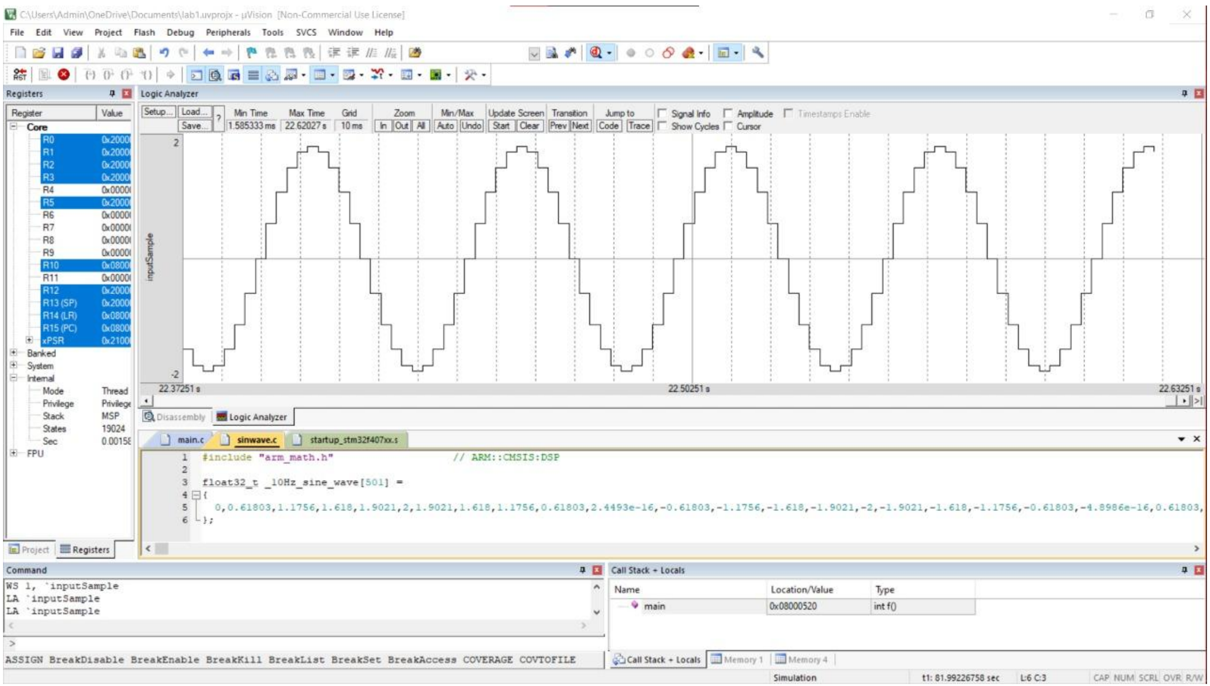
```c
uint32_t freq;
extern float32_t _10Hz_sine_wave[SIG_LENGTH];
//void SysTick_Handler(void);
void plot_sine_wave(void);

float32_t inputSample;

int main()
{
	HAL_Init();
	freq = HAL_RCC_GetHCLKFreq();
	plot_sine_wave();
	while(1)
	{
		plot_sine_wave();
	}
}
void plot_sine_wave(void)
{
	int i,j;
	for(i = 0;i<SIG_LENGTH;i++)
		{
			inputSample = _10Hz_sine_wave[i];
			HAL_Delay(1);
			//for(j=0;j<3000;j++){}
		}
}
void SysTick_Handler(void)
{
	HAL_IncTick();
	HAL_SYSTICK_IRQHandler();
}
```

```c
// sinewave.c
#include "arm_math.h"                    // ARM::CMSIS:DSP

float32_t _10Hz_sine_wave[501] =
{
   // enter 501 values of sine wave generated from matlab.
}
```

SIG_LENGTH=5001



SIG_LENGTH=501



Conclusion:

1. As we increase the size of the array containing sine wave values, it tends closer to the actual sinewave and interpolation error reduces.
2. This happens when we reduce the time step value in MATLAB giving us higher accuracy.