TASK1: Firmware Foundations & Environment Setup

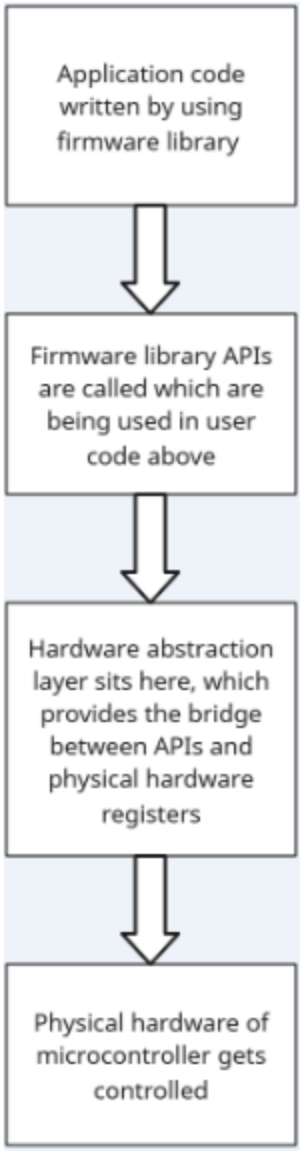
What is a firmware library?

Answer: Firmware library is the collection of pre-written code modules/functions which can be used while developing the firmware for a particular application.

These ready made functions are readable and re-usable. If the firmware library has not been there then one has needed to code the baremetal firmware which will be more difficult to code and understand. As baremetal firmware would have required the knowledge of the complete microcontroller internals.

The firmware library is used for developing our firmware code also called application code, which then interacts with firmware library APIs. Below firmware APIs there is a hardware abstraction layer (HAL). HAL plays the role of bridge between APIs and hardware. HAL controls the hardware according to the APIs called and used.

The block diagram below shows the flow from application code down to hardware interaction:


Application code written by using firmware library
Firmware library APIs are called which are being used in user code above
Hardware abstraction layer sits here, which provides the bridge between APIs and physical hardware registers
Physical hardware of microcontroller gets controlled

Why APIs are important in embedded systems?

Answer: APIs can be considered as a high level human readable code/functions. So basically APIs are hiding the hardware complexity below them.

Below snippet taken from the internet shows that without API if we would have to program a register then we need to directly manipulate the register.

While with pre-build functions (APIs) we can directly use them to do the same functionality in the readable and understandable form.

Without API

```c
PORTB |= (1 << 5);
```

With API

```c
GPIO_SetPin(LED_PIN, HIGH);
```

What was understood from the lab code?

Answer:

Main.c contains the application code made specific for GPIO. This code uses the already defined APIs to implement the functionality.

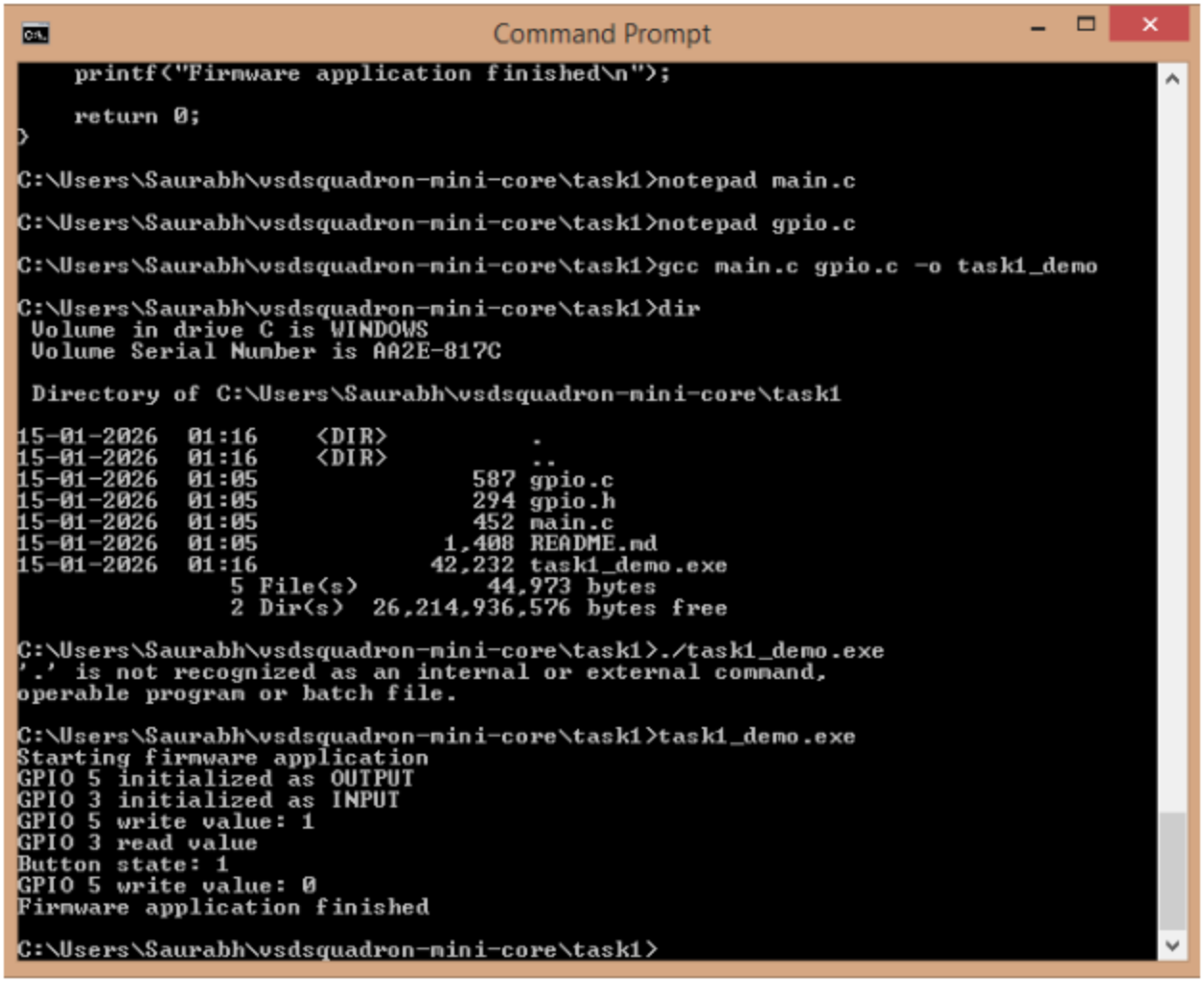
Gpio.h contains the definition of the GPIO specific functions which are called in main.c.

Gpio.c contains the implementation of the APIs defined in gpio.h file. One can get to know from this file that what exactly the API is going to do when called in main.c.

Finally we have compiled the code using GCC compiler and produced a **.exe** file out of it. Which in turn on running produces the output on console.

==========

Below screenshot shows the successful compilation and program output:


Command Prompt
printf("Firmware application finished\n");
return 0;
}
C:\Users\Saurabh\vsdsquadron-mini-core\task1>notepad main.c
C:\Users\Saurabh\vsdsquadron-mini-core\task1>notepad gpio.c
C:\Users\Saurabh\vsdsquadron-mini-core\task1>gcc main.c gpio.c -o task1_demo
C:\Users\Saurabh\vsdsquadron-mini-core\task1>dir
Volume in drive C is WINDOWS
Volume Serial Number is AA2E-817C
Directory of C:\Users\Saurabh\vsdsquadron-mini-core\task1
15-01-2026 01:16 <DIR> .
15-01-2026 01:16 <DIR> ..
15-01-2026 01:05 587 gpio.c
15-01-2026 01:05 294 gpio.h
15-01-2026 01:05 452 main.c
15-01-2026 01:05 1,408 README.md
15-01-2026 01:16 42,232 task1_demo.exe
5 File(s) 44,973 bytes
2 Dir(s) 26,214,936,576 bytes free
C:\Users\Saurabh\vsdsquadron-mini-core\task1>./task1_demo.exe
'.' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\Saurabh\vsdsquadron-mini-core\task1>task1_demo.exe
Starting firmware application
GPIO 5 initialized as OUTPUT
GPIO 3 initialized as INPUT
GPIO 5 write value: 1
GPIO 3 read value
Button state: 1
GPIO 5 write value: 0
Firmware application finished
C:\Users\Saurabh\vsdsquadron-mini-core\task1>