



# Learning Report – Embedded C



*L&T Technology Services*



## Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
	08/03/2021	Akshansh Mishra (Ps No- 99003753)			

# Contents

<b>ACTIVITY 1 – COMPILATION APPROACH.....</b>	<b>4</b>
1.1- MAKE FILE .....	4
1.2- STARTUP CODE.....	5
1.3- LINKER FILE.....	8
1.4- DEBUGGING TECHNIQUES.....	9
 <b>ACTIVITY 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE .....</b>	 <b>10</b>
2.1 GPIO.....	10
2.2 EXTI.....	12
2.3 ADC.....	14
2.4 SPI.....	16
2.5 UART.....	19
 <b>ACTIVITY 3 – PROJECT ON BCM MODULE .....</b>	 <b>21</b>

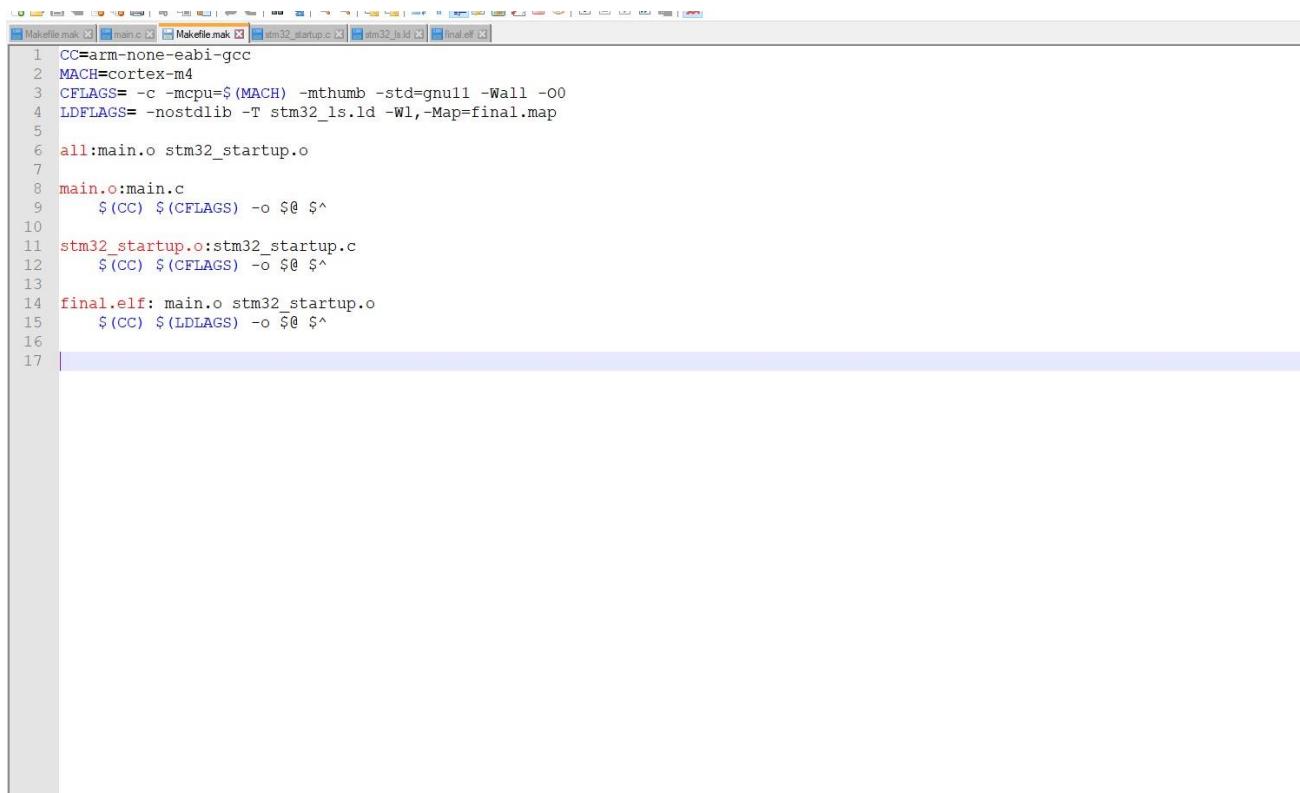
## Activity 1 – COMPILATION APPROACH

This is the complete compilation process of the sample program for ARM Cortex Mx processor based boards. Following are the compilation stages of a C program:

1. Preprocessor stage
2. Compilation stage
3. Assembly stage
4. Linking stage

### 1.1- MAKE FILE

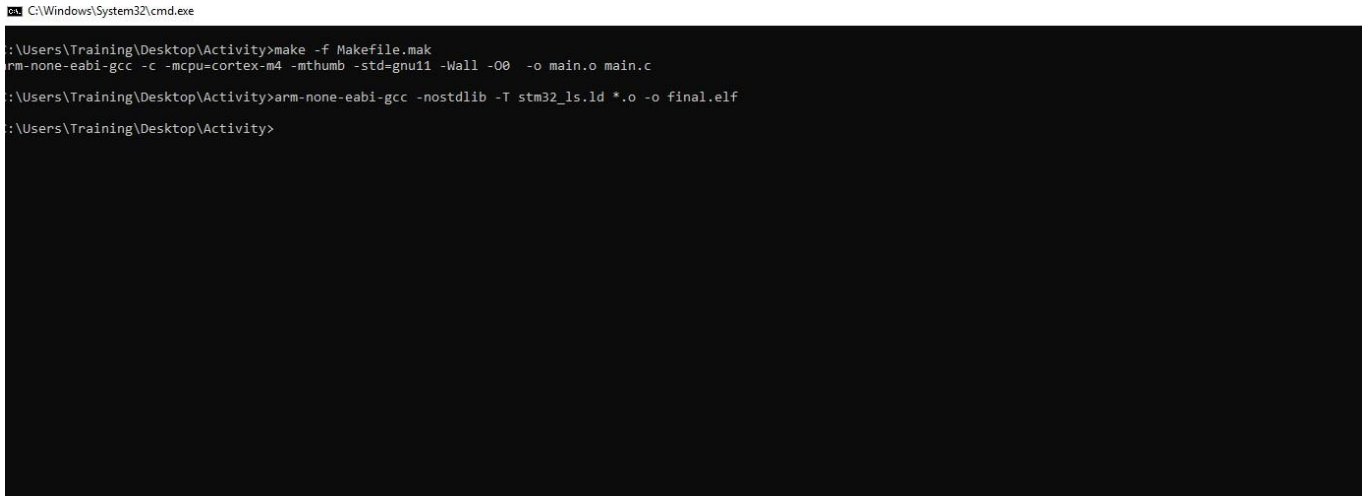
Below is the make file for the sample program:



```
1 CC=arm-none-eabi-gcc
2 MACH=cortex-m4
3 CFLAGS= -c -mcpu=$(MACH) -mthumb -std=gnu11 -Wall -O0
4 LDFLAGS= -nostdlib -T stm32_ls.ld -Wl,-Map=final.map
5
6 all:main.o stm32_startup.o
7
8 main.o:main.c
9     $(CC) $(CFLAGS) -o $@ $^
10
11 stm32_startup.o:stm32_startup.c
12     $(CC) $(CFLAGS) -o $@ $^
13
14 final.elf: main.o stm32_startup.o
15     $(CC) $(LDFLAGS) -o $@ $^
16
17
```

Fig 1.1.1 make file

The command to run this make file in the command prompt is:



```
C:\Windows\System32\cmd.exe
:\Users\Training\Desktop\Activity>make -f Makefile.mak
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -O0 -o main.o main.c
:\Users\Training\Desktop\Activity>arm-none-eabi-gcc -nostdlib -T stm32_ls.ld *.o -o final.elf
:\Users\Training\Desktop\Activity>
```

Fig 1.1.2 Make command

- **-mcpu=cortex-m4** is used to select our cortex-m4 processor which is used
- **-mthumb** is used to generate the code that executes in ARM state
- **main.o** is the target file
- **main.c** is the dependency

## 1.2- STARTUP CODE

- The startup file is responsible for setting up the right environments to run the code in main.c file.
- Some part of the startup code is target (processor) dependent.
- Role of startup file:
  1. Create a MCU specific vector table for microcontroller.
  2. To write a startup code which initializes .data and .bss section in SRAM.
  3. Call main()

```

1 #include<stdint.h>
2
3 #define SRAM_START 0x20000000U
4 #define SRAM_SIZE (128U * 1024U) //128KB
5 #define SRAM_END ((SRAM_START) + (SRAM_SIZE))
6
7 #define STACK_START SRAM_END
8
9 extern uint32_t _etext;
10 extern uint32_t _edata;
11 extern uint32_t _la_data;
12 extern uint32_t _la_data;
13
14 extern uint32_t _ebss;
15 extern uint32_t _ebss;
16
17 //prototype of main
18
19 int main(void);
20
21 void __libc_init_array(void);
22
23
24 /* function prototypes of STM32F407x system exception and IRQ handlers */
25 void Reset_Handler(void);
26
27 void NMI_Handler (void) __attribute__((weak, alias("Default_Handler")));
28 void HardFault_Handler (void) __attribute__((weak, alias("Default_Handler")));
29 void MemManage_Handler (void) __attribute__((weak, alias("Default_Handler")));
30 void BusFault_Handler (void) __attribute__((weak, alias("Default_Handler")));
31 void UsageFault_Handler (void) __attribute__((weak, alias("Default_Handler")));
32 void SVC_Handler (void) __attribute__((weak, alias("Default_Handler")));
33 void DebugMon_Handler (void) __attribute__((weak, alias("Default_Handler")));
34 void PendSV_Handler (void) __attribute__((weak, alias("Default_Handler")));
35 void SysTick_Handler (void) __attribute__((weak, alias("Default_Handler")));
36 void WWDG_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
37 void PVD_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
38 void TAMP_STAMP_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
39 void RTC_WKUP_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
40 void RCC_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
41 void EXTI0_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
42 void EXTI1_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
43 void EXTI2_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
44 void EXTI3_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
45 void EXTI4_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
46 void DMA1_Stream0_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
47 void DMA1_Stream1_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
48 void DMA1_Stream2_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
49 void DMA1_Stream3_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
50 void DMA1_Stream4_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
51 void DMA1_Stream5_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
52 void DMA1_Stream6_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
53 void DMA1_Stream7_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
54 void ADC_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
55 void CAN1_TX_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
56 void CAN1_RX0_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
57 void CAN1_RX1_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

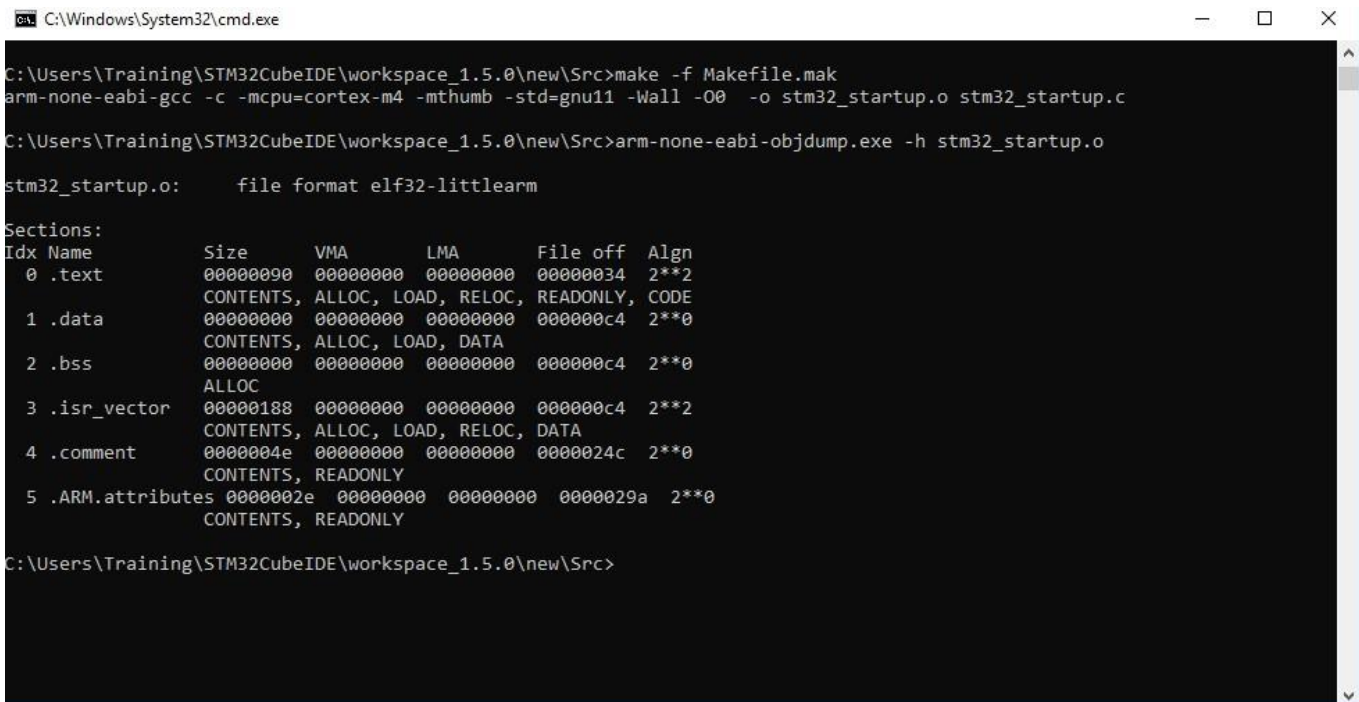
Fig 1.2.1 Startup code

In startup code we use variable attributes to store some variables in the user defined function.

Function attributes:

- Weak: Lets programmer override already defined weak function (dummy function) with the same function name.
- Alias: Lets programmer give any alias name for same function.

The startup.o file generated is of elf executable format, various sections of which are shown below:



```

C:\Windows\System32\cmd.exe

C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>make -f Makefile.mak
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -O0 -o stm32_startup.o stm32_startup.c

C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>arm-none-eabi-objdump.exe -h stm32_startup.o

stm32_startup.o:      file format elf32-littlearm

Sections:
Idx Name              Size      VMA           LMA           File off  Algn
 0 .text              00000090  00000000  00000000  00000034  2**2
    CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data              00000000  00000000  00000000  000000c4  2**0
    CONTENTS, ALLOC, LOAD, DATA
 2 .bss               00000000  00000000  00000000  000000c4  2**0
    ALLOC
 3 .isr_vector        00000188  00000000  00000000  000000c4  2**2
    CONTENTS, ALLOC, LOAD, RELOC, DATA
 4 .comment            0000004e  00000000  00000000  0000024c  2**0
    CONTENTS, READONLY
 5 .ARM.attributes    0000002e  00000000  00000000  0000029a  2**0
    CONTENTS, READONLY

C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>

```

Fig 1.2.2: Startup command

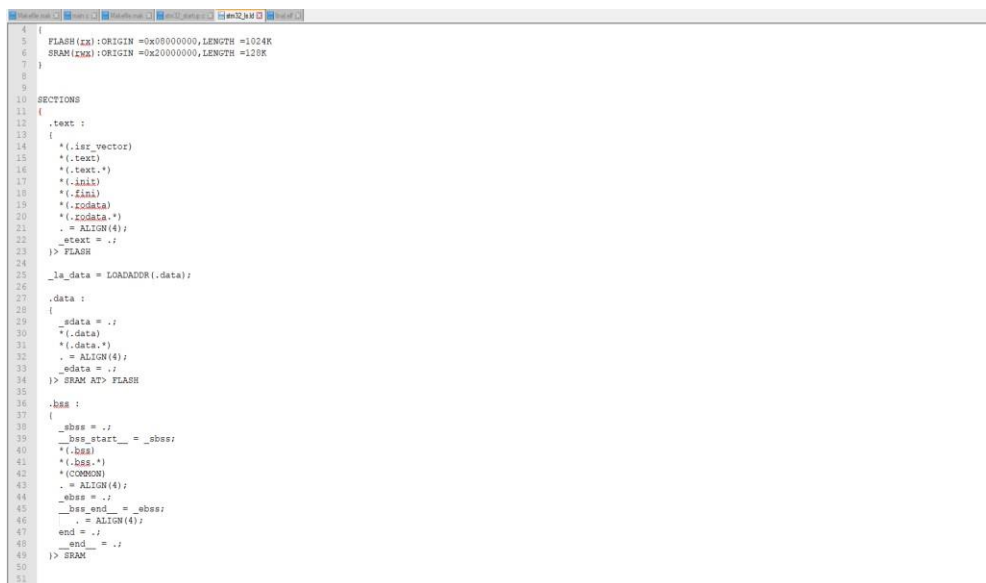


### 1.3- LINKER SCRIPT

- Linkers take one or more object files or libraries as input and combines them to create a single executable file as output.
- Linker scripts decide how different sections of object file should be merged to create an output file.
- Reset handler is the entry point to the application
- Entry command is used to set the “Entry point address” information in the header of final elf file generated.

Syntax: Entry(symbol\_name)

Entry(Reset\_Handler)



```
1  {
2      FLASH(1M):ORIGIN =0x08000000,LENGTH =1024K
3      SRAM(128K):ORIGIN =0x20000000,LENGTH =128K
4  }
5
6  SECTIONS
7  {
8      .text :
9      {
10         *(.isr_vector)
11         *(.text)
12         *(.text.*)
13         *(.init)
14         *(.fini)
15         *(.rodata)
16         *(.rodata.*)
17         . = ALIGN(4);
18     }> FLASH
19
20     _la_data = LOADADDR(.data);
21
22     .data :
23     {
24         .data = .;
25         *(.data)
26         *(.data.*)
27         . = ALIGN(4);
28     }> SRAM AT> FLASH
29
30     .bss :
31     {
32         .bss = .;
33         __bss_start__ = .bss;
34         *(.bss)
35         *(.bss.*)
36         *(COMMON)
37         . = ALIGN(4);
38         .bss = .;
39         __bss_end__ = .bss;
40         . = ALIGN(4);
41     }> SRAM
42
43     end = .;
44     _end = .;
45 }> SRAM
46
47
```



```
C:\Windows\System32\cmd.exe
C:\Users\Training\Documents\Embedded C\STM project\Activity1>arm-none-eabi-gcc -nostdlib -T stm32_ls.ld *.o -o final.elf
c:/program files (x86)/gnu arm embedded toolchain/10 2020-q4-major/bin/../lib/gcc/arm-none-eabi/10.2.1/../../../../arm-none-eabi/bin/ld.exe: stm3
2_startup.o: in function 'Reset_Handler':
stm32_startup.c:(.text+0x80): undefined reference to `_la_data'
collect2.exe: error: ld returned 1 exit status

C:\Users\Training\Documents\Embedded C\STM project\Activity1>arm-none-eabi-gcc -nostdlib -T stm32_ls.ld *.o -o final.elf

C:\Users\Training\Documents\Embedded C\STM project\Activity1>dir
Volume in drive C has no label.
Volume Serial Number is F8FC-05CA

Directory of C:\Users\Training\Documents\Embedded C\STM project\Activity1

23-02-2021  14:29    <DIR>          .
23-02-2021  14:29    <DIR>          ..
23-02-2021  14:29             135,752 final.elf
23-02-2021  14:24             2,034 main.c
23-02-2021  14:24             1,940 main.o
23-02-2021  11:35              220 Makefile.mak
23-02-2021  14:28              662 stm32_ls.ld
23-02-2021  14:27          12,847 stm32_startup.c
23-02-2021  14:24           5,688 stm32_startup.o
                7 File(s)          159,143 bytes
                2 Dir(s) 125,771,227,136 bytes free

C:\Users\Training\Documents\Embedded C\STM project\Activity1>
```

Fig 1.3.1 command to generate final.elf file

## 1.4- DEBUGGING TECHNIQUES

- The STM32F407VG is embedded with on chip debugger for debugging the code.
- The OCD ON-Chip Debugger aims to provide debugging, in system programming and boundary scan testing for embedded target devices.
- OCD is a free and opensource host application allows you to program, debug, and analyze your applications using GDB.
- It supports various target boards based on different processor architecture.

## Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE

Implementation of protocols for STM32F407VG microcontroller featuring ARM32 bit ARM-cortex M4 with FPU core using HAL library.

### 2.1 GPIO:

Toggling LED at pin PD12 at GREEN\_LED\_GPIO\_PORT. Serial wire is enabled at pin PA13.

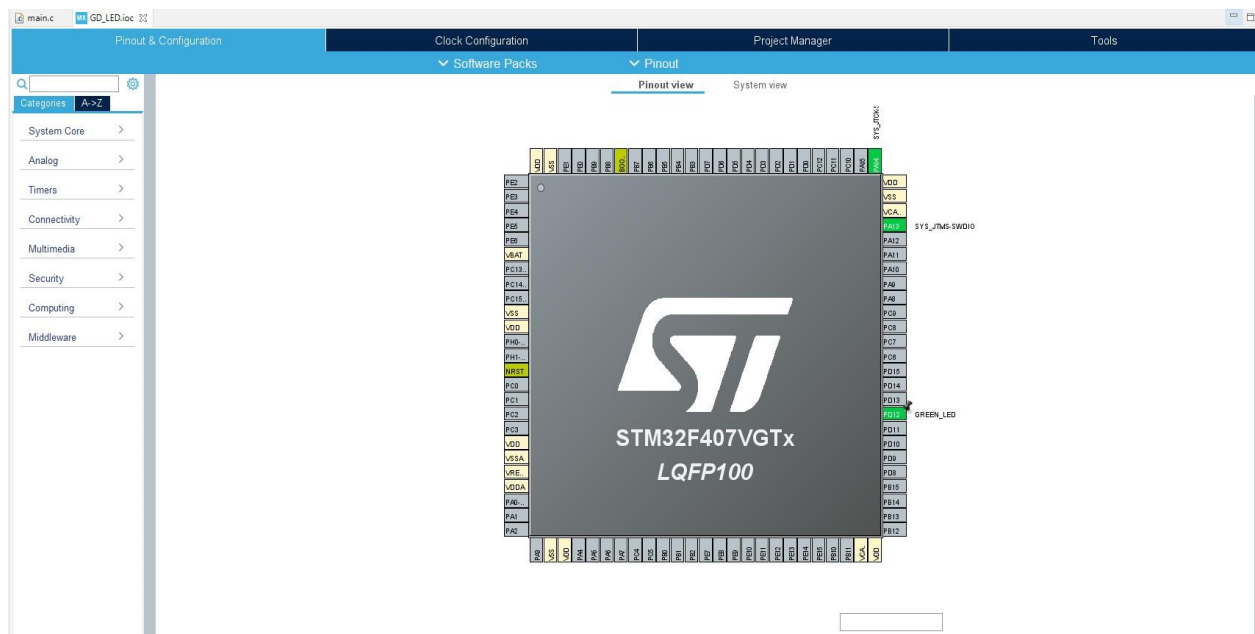


Fig: 2.1.1 GPIO pin configuration

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_TogglePin(GREEN_LED_GPIO_Port, GREEN_LED_Pin);
    HAL_Delay(500);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

Fig: 2.1.2 GPIO configuration code

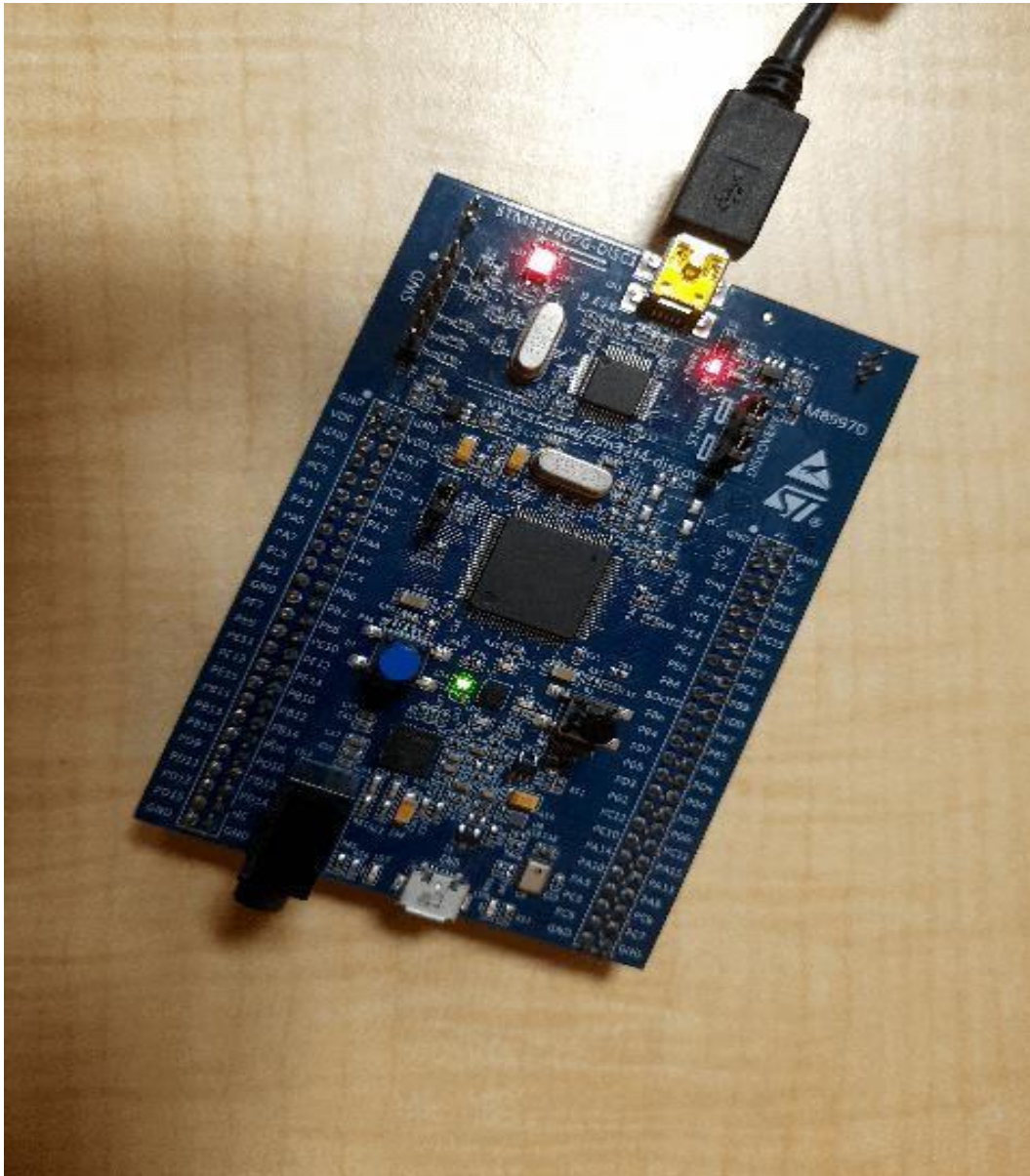


Fig: 2.1.2 LED toggling

## 2.2 EXTI :-

Blue button at PA0 works as an external interrupt.

When the blue button is pressed the Green LED at pin PD12 toggles.

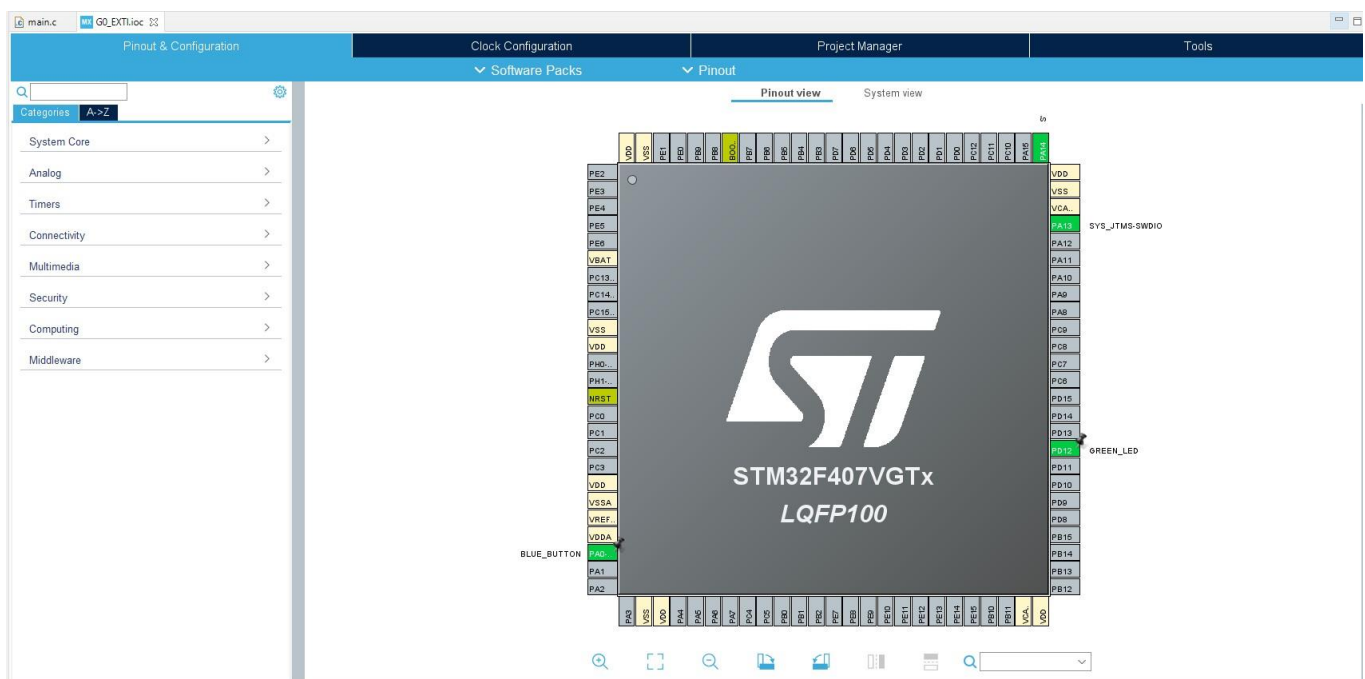


Fig: 2.2.1 EXTI pin configuration

In the main.c file a flag is initialized and if the flag == 1, the condition under the if loop executed to toggle the LED at PD12.

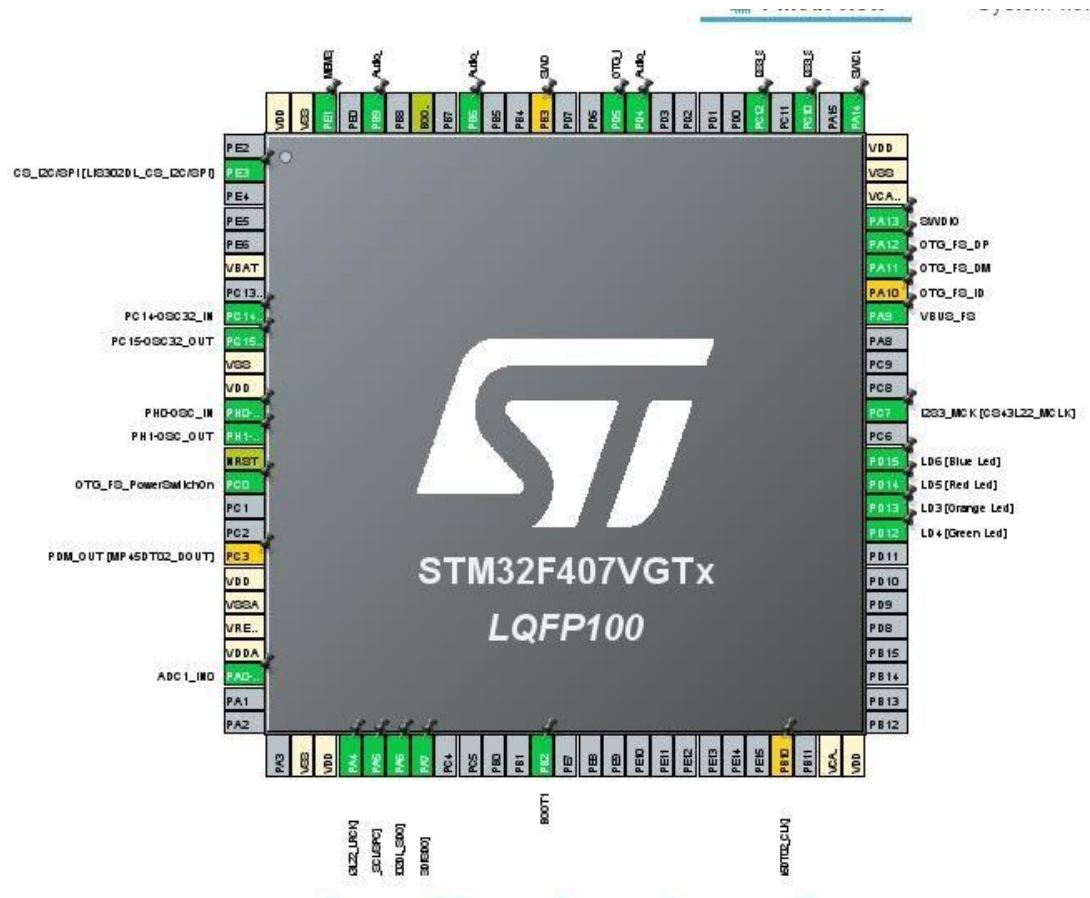
```

43
44 /* USER CODE BEGIN PV */
45 uint8_t flag = 0;
46 /* USER CODE END PV */
47
48 /* Private function prototypes -----*/
49 void SystemClock_Config(void);
50 static void MX_GPIO_Init(void);
51 /* USER CODE BEGIN PFP */
52
53 /* USER CODE END PFP */
54
55 /* Private user code -----*/
56 /* USER CODE BEGIN 0 */
57
58 /* USER CODE END 0 */
59
60 /**
61  * @brief The application entry point.
62  * @retval int
63  */
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         if(flag){
99             HAL_GPIO_TogglePin(GREEN_LED_GPIO_Port, GREEN_LED_Pin);
100             HAL_Delay(500);
101             flag = 0;
102         }
103     }
104     /* USER CODE BEGIN 3 */
105
106     /* USER CODE END 3 */
107
108 }

```

Fig: 2.2.1 EXTI configuration code

## 2.3 ADC





## 2.4 SPI

```

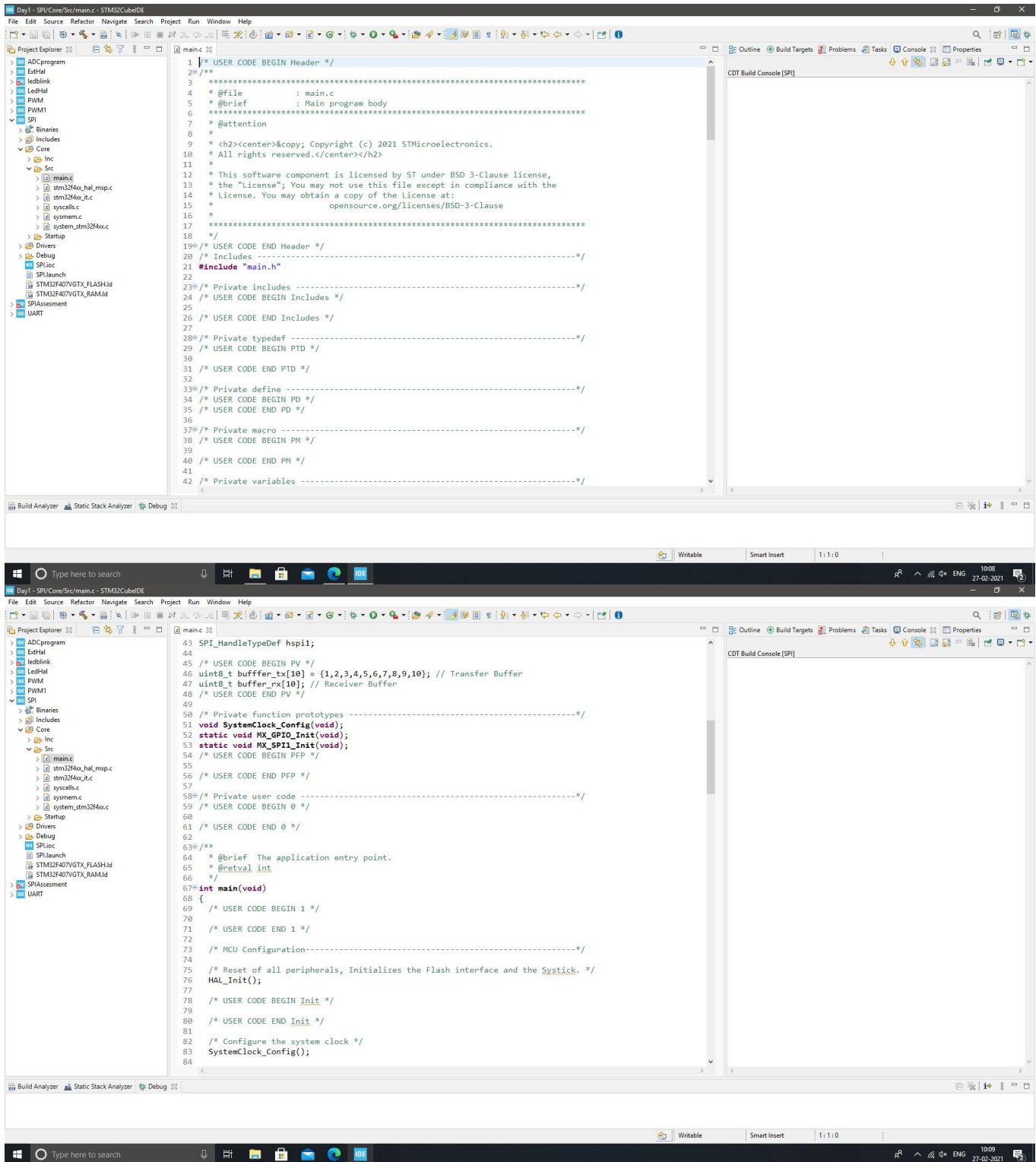
19  /* USER CODE END Header */
20  /* Includes -----*/
21  #include "main.h"
22  #include "usb_host.h"
23
24
25  ADC_HandleTypeDef hadc1;
26
27  I2C_HandleTypeDef hi2c1;
28
29  I2S_HandleTypeDef hi2s3;
30
31  SPI_HandleTypeDef hspi1;
32
33
34  /* Private function prototypes -----*/
35  void SystemClock_Config(void);
36  static void MX_GPIO_Init(void);
37  static void MX_I2C1_Init(void);
38  static void MX_I2S3_Init(void);
39  static void MX_SPI1_Init(void);
40  static void MX_ADC1_Init(void);
41  void MX_USB_HOST_Process(void);
42
43
44  uint32_t adc_value;
45
46  int main(void)
47  {
48
49      HAL_Init();
50
51      SystemClock_Config();
52
53      MX_GPIO_Init();
54      MX_I2C1_Init();
55      MX_I2S3_Init();
56
57
58
59
60      while (1)
61      {
62
63          MX_USB_HOST_Process();
64
65          HAL_ADC_Start(&hadc1);
66
67
68          if(HAL_ADC_PollForConversion(&hadc1, 5) == HAL_OK){
69              adc_value= HAL_ADC_GetValue(&hadc1);
70          }
71
72
73      }
74
75      HAL_ADC_Stop(&hadc1);
76      HAL_Delay(100);
77
78  }
79
80
81  }
82
83
84  void SystemClock_Config(void)
85  {
86      RCC_OscInitTypeDef RCC_OscInitStruct = {0};
87      RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
88      RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

```

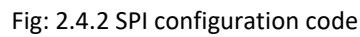
Fig: Configuration code







```
1  /* USER CODE BEGIN Header */
2  /**
3   * @file
4   * @brief : Main program body
5   * @attention
6   *
7   * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
8   * All rights reserved.</center></h2>
9   *
10   * This software component is licensed by ST under BSD 3-Clause license,
11   * the "License"; You may not use this file except in compliance with the
12   * License. You may obtain a copy of the License at:
13   * opensource.org/licenses/BSD-3-Clause
14   *
15   */
16  /*
17   */
18  /* USER CODE END Header */
19  /* Includes -----*/
20  #include "main.h"
21  /* Private includes -----*/
22  /* USER CODE BEGIN Includes */
23  /* USER CODE END Includes */
24  /* Private typedef -----*/
25  /* USER CODE BEGIN PTD */
26  /* USER CODE END PTD */
27  /* Private define -----*/
28  /* USER CODE BEGIN PD */
29  /* USER CODE END PD */
30  /* Private macro -----*/
31  /* USER CODE BEGIN PM */
32  /* USER CODE END PM */
33  /* Private variables -----*/
34  /* USER CODE BEGIN PV */
35  /* USER CODE END PV */
36  /* Private function prototypes -----*/
37  void SystemClock_Config(void);
38  static void MX_GPIO_Init(void);
39  static void MX_SPI1_Init(void);
40  /* USER CODE BEGIN PFP */
41  /* USER CODE END PFP */
42  /* Private user code -----*/
43  /* USER CODE BEGIN 0 */
44  /* USER CODE END 0 */
45  /**
46   * @brief The application entry point.
47   * @retval int
48   */
49  int main(void)
50  {
51  /* USER CODE BEGIN 1 */
52  /* USER CODE END 1 */
53  /* MCU Configuration -----*/
54  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
55  HAL_Init();
56  /* USER CODE BEGIN Init */
57  /* USER CODE END Init */
58  /* Configure the system clock */
59  SystemClock_Config();
60  }
```



## 2.5 UART

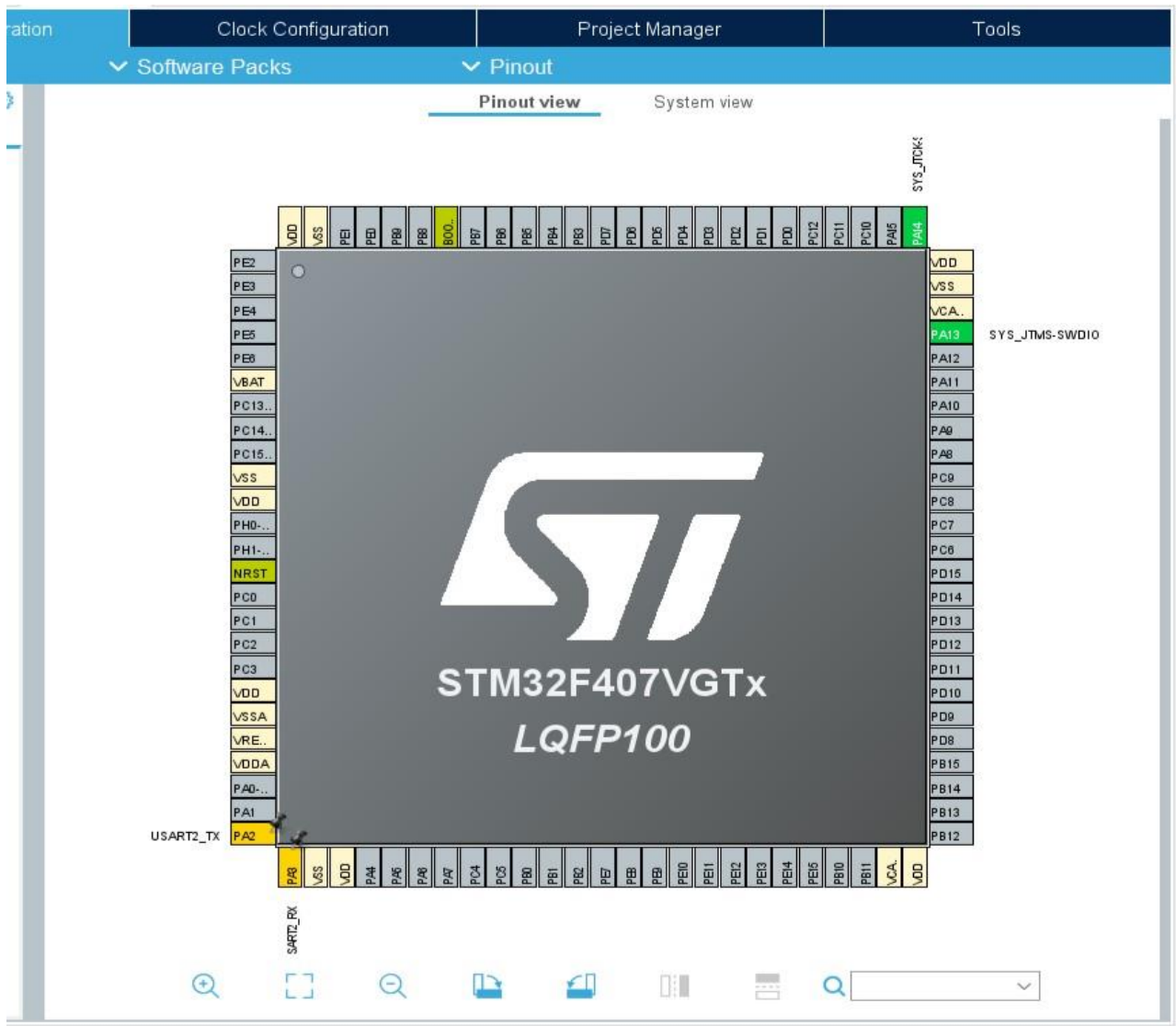


Fig: 2.5.1 UART Pin configuration

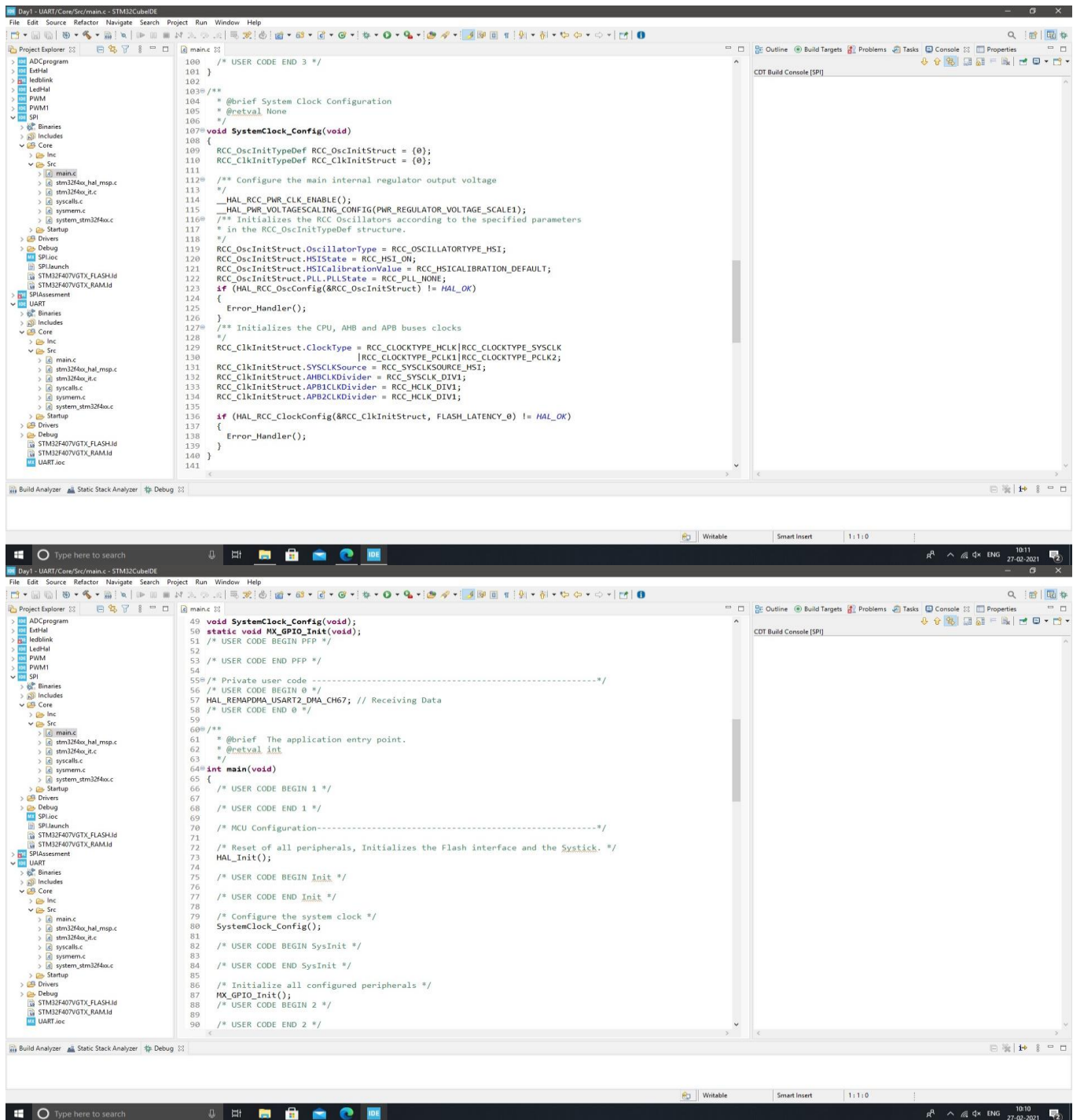


Fig: 2.5.2 UART configuration code

### Activity 3 – PROJECT ON BCM MODULE

BCM module was implemented using STM32f407VG microcontroller featuring 32 bit ARM-cortex M4 with FPU core.

This BCM module have following features:

1. Seat belt System
2. Sunroof Control System

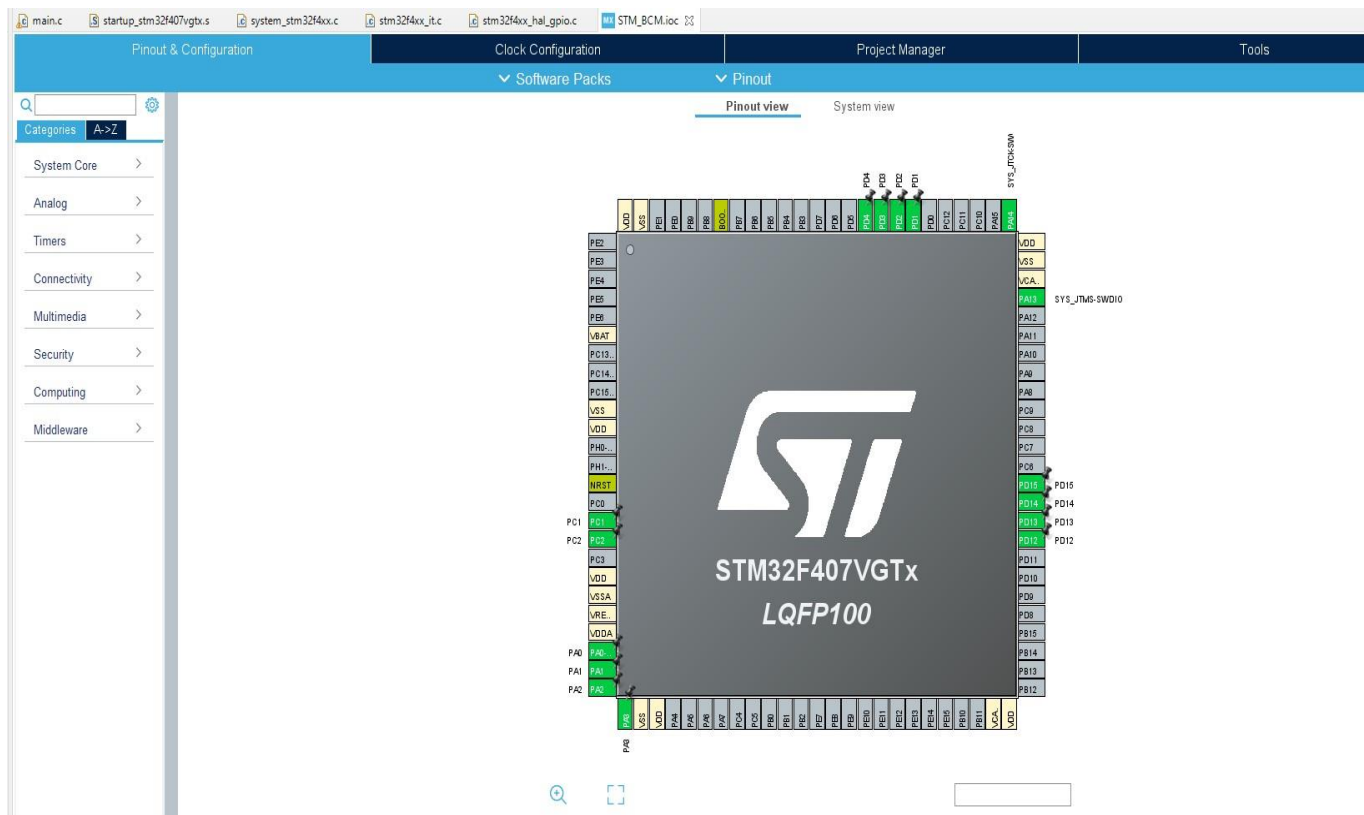


Fig 3.2 Pin configuration

## Implementation

### 1. Seat belt System

Buzzer status denotes status of Seat Belt.

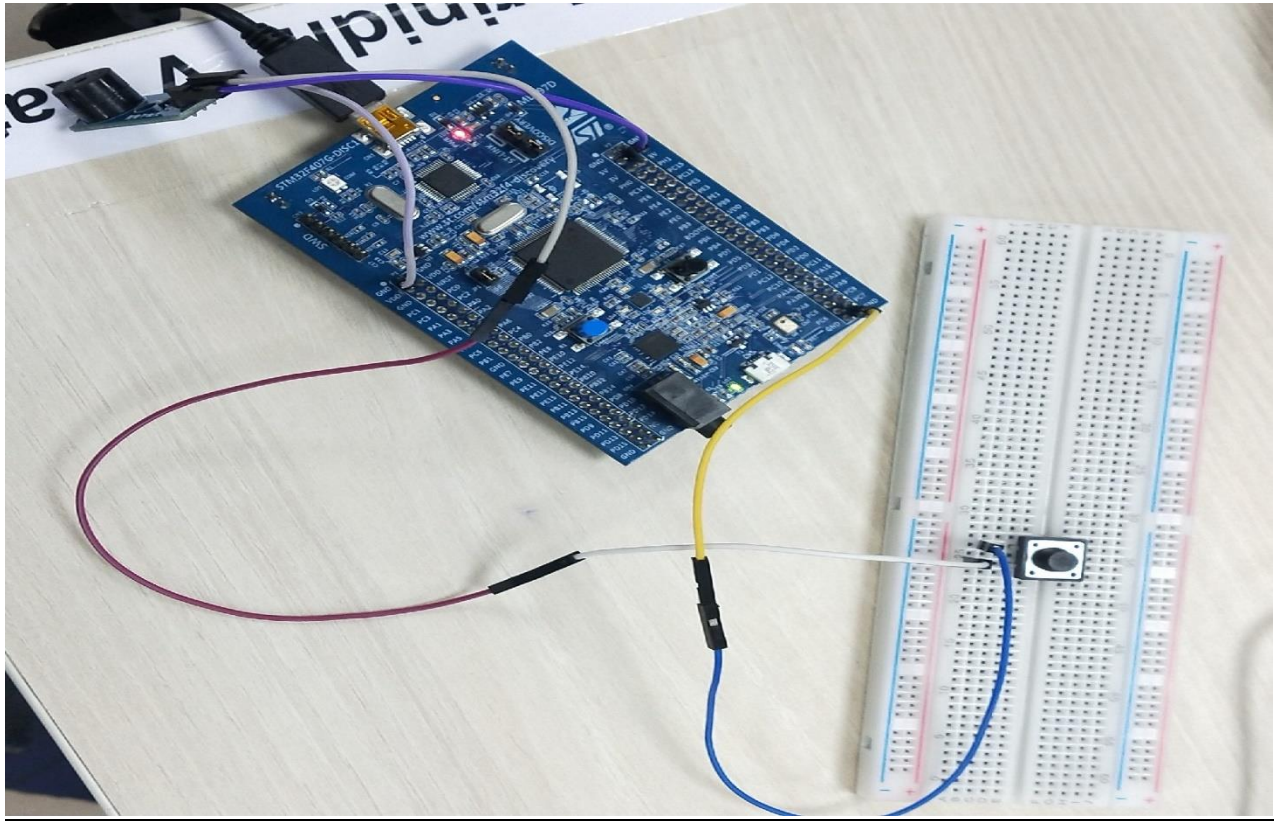
Input: Push Button switch(PB14);

Output: Buzzer makes sound;

```
///-----seat belt warning system.-----  
while (1)  
{  
    /* USER CODE END WHILE */  
    MX_USB_HOST_Process();  
    /* USER CODE END WHILE */  
    if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_14)==1)  
    {  
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_6,GPIO_PIN_SET);  
    }  
    else  
    {  
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_6,GPIO_PIN_RESET);  
    }  
}
```

Figure :- Coding Part of Seat Belt System





**Figure :- Implementation of Seat Belt Control system**

## 2. Sunroof control system.

LED status denotes OPEN/CLOSE status of sunroof.

Input: Push Button switch(PB14);

Output: GREEN LED BLINK;

[illegible]

Figure :- Coding Part of Sunroof Control System

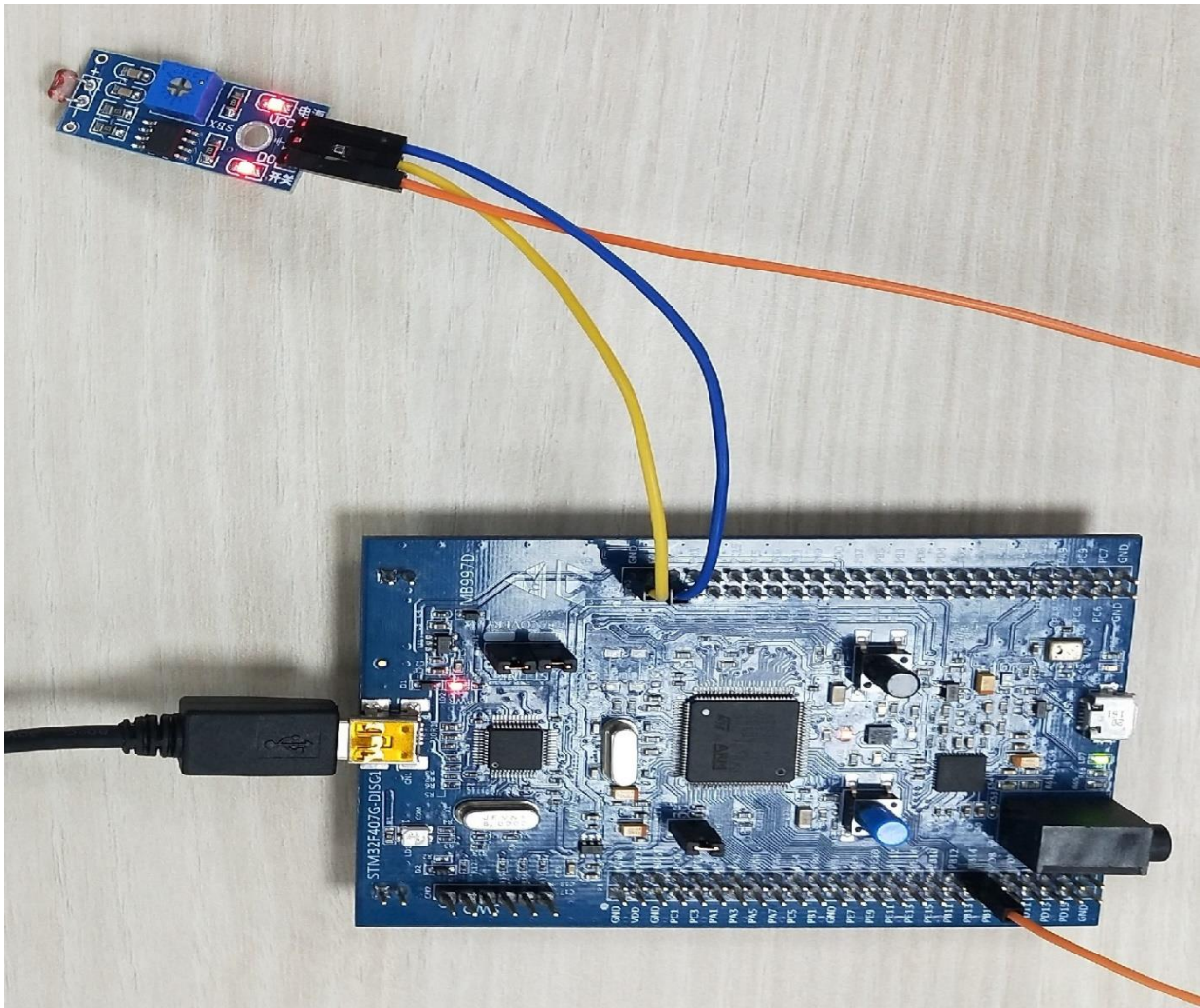


Figure :- Implementation Part of Sunroof Control System

