GEBZE TECHNICAL UNIVERSITY

ELECTRONICS ENGINEERING

ELM335

Microprocessors Laboratory

LAB 5 Experiment Report

Prepared by
151024008 - Abdürrahim Deniz KUMBARACI
171024050 - Abdül Samet Karapınar
171024008 - Yasin Özbek

Problem 1

Code

main.c

```
#include "bsp.h"
#include "stm32g0xx.h"



int main(void) { //Empty main function it only calls init_xxx functions then rest at infinite loop
BSP_system_init();
BSP_UART_init(9600);

while(1){

	_print(0,"Yasin Samet",11);
	uart_tx(uart_rx());
	delay(1000);
}

}
```

bsp.h

```
#ifndef BSP_H_
#define BSP_H_

#define PoC  2
#define PoA  0
#define PoB  1
#define PoD  3
#define PoF  5

void BSP_system_init();

int _print(int fd,char *buf, int len);


unsigned char uart_rx(void);

void print(char *s);




void BSP_IWDG_init();
void BSP_UART_init(unsigned int baud);

void delay(volatile unsigned int);
void SysTick_Handler(void);

int BSP_button_read();
void uart_tx(unsigned char c);

void delay_ms(volatile unsigned int s);
```

```
void BSP_sled_set();
void BSP_sled_clear();
void BSP_sled_init();
void BSP_sled_toggle();
void BSP_sbutton_init();




#endif
```

bsp.c

```
#include "bsp.h"
#include "stm32g0xx.h"

static volatile unsigned int tick ;

void BSP_system_init(){
	__disable_irq();
	BSP_sled_init();
	BSP_sbutton_init();
	SysTick_Config(SystemCoreClock/1000);
	__enable_irq();
}

void SysTick_Handler(void){
	if(tick > 0){
		--tick;}
}

void delay(volatile unsigned int s){
	for(;s>0;s--);
}


int _print(int fd,char *buf, int len){
	(void)fd;
	for(int i=0; i<len;++i){
		uart_tx(buf[i]);
	}
	return len;
}

void print(char *s) {
	int c = 0;
	while( s[c] != '\0'){
		 c++;
	}
	 c = _print(0, s, c);
}

void uart_tx(unsigned char c){
		USART2->TDR = (uint16_t)c;
		while(! ( USART2->ISR & (1 << 6)));
}

unsigned char uart_rx(void){
	uint8_t data = (uint8_t)USART2->RDR;
	return data;
}


void BSP_UART_init(unsigned int baud){
	//Enable IOA and USART2 clocks
	RCC->IOPENR |= (1U << 0);
```

```
	RCC->APBENR1 |=(1U << 17);
	//Setting PA2 as alternative function mode "10"
	GPIOA->MODER  &= ~(3U << 2*2);
	GPIOA->MODER |= (2U << 2*2);

	GPIOA->AFR[0] &= ~(0xFU << 2*4);
	GPIOA->AFR[0] |= (1 << 2*4);
	//Setting PA3 as alternative function mode "10"
	GPIOA->MODER  &= ~(3U << 2*3);
	GPIOA->MODER |= (2U << 2*3);
	//Choosing alternative functions from MUX

	GPIOA->AFR[0] &= ~(0xFU << 4*3);
	GPIOA->AFR[0] |= (1 << 3*4);

	//Setup USart2
	USART2->CR1 = 0;
	USART2-> CR1 |= (1<<3);  //Transmitter enabled
	USART2-> CR1 |= (1<<2);   //Reciever enabled
	//USART2-> CR1      |= (1<<5);   //Reciever enabled



	USART2->BRR = (uint16_t)(SystemCoreClock/baud);

	USART2-> CR1 |= (1 << 0); //Lowpower usart enabled

	//NVIC_SetPriority(USART2_IRQn,1);
	//NVIC_EnableIRQ(USART2_IRQn);
}




void BSP_sbutton_init(){
	RCC->IOPENR |= (1U << 0);
	GPIOA->MODER &= ~(3U << 2*15);
	GPIOA->PUPDR |= (2U << 2*15);
}

void BSP_sled_init(){
	GPIOC->MODER &= ~(3U << 2*6);
	GPIOC->MODER |= (1U << 2*6);
}

void BSP_sled_toggle(){
		GPIOC->ODR ^= (1U << 6);
}
void BSP_sled_set(){
	  GPIOC->ODR |= (1U << 6);
}
void BSP_sled_clear(){
	GPIOC->ODR  &= ~(1U << 6);
}
```

Output of the Code

*main.c

```
1
2  #include "bsp.h"
3  #include "stm32g0xx.h"
4
5
6
7
8  int main(void) { //Empty main function it only calls init_xxx functions then rest at infinite loop
9  BSP_system_init();
10 BSP_UART_init(9600);
11
12 while(1){
13
14     _print(0,"Yasin Samet",11);
15     uart_tx(uart_rx());
16     delay(1000);
17 }
18
19 }
20
```

Console

Uart2 16 (CONNECTED)

```
 SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasi
n SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYas
in SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYa
sin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametY
asin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin Samet
Yasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin Same
tYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin Sam
etYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin SametYasin Sa
Yasin SametmetYasin SametYasin SametYasin SametYasin SametYasin SametYasin Samet
```

Comments and Questions

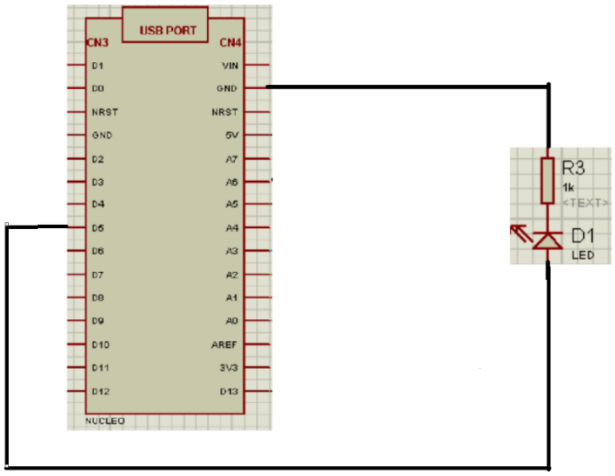
In this problem, the card was connected to the computer using the UART protocol. An initialization routine has been created for the UART and send-receive functions have been created.

Only external firmware was a connection with a PC via USB port.

Code is based on Lecture

Problem 3

Block Diagram



Code

main.c

```
#include "stm32g0xx.h"
#include "nucleo.h"
int i=0;
uint32_t
ar2[]={22500,22641,22783,22924,23065,23207,23348,23489,23630,23772,23913,24054,24195,24336,2447
7,24617,24758,24899,25039,25180,25320,25460,25600,25740,25880,26020,26159,26299,26438,26577,267
16,26855,26993,27132,27270,27408,27546,27684,27821,27958,28096,28232,28369,28505,28641,28777,28
913,29048,29183,29318,

      29453,29587,29721,29855,29988,30122,30254,30387,30519,30651,30783,30914,31045,31176,3130
6,31436,31565,31695,31823,31952,32080,32208,32335,32462,32589,32715,32841,32966,33091,33215,333
39,33463,33586,33709,33832,33953,34075,34196,34316,34437,34556,34675,34794,34912,35030,35147,35
264,35380,35495,35611,

      35725,35839,35953,36066,36178,36290,36402,36513,36623,36733,36842,36951,37059,37166,3727
3,37380,37485,37590,37695,37799,37902,38005,38107,38209,38310,38410,38510,38609,38707,38805,389
02,38998,39094,39189,39284,39377,39471,39563,39655,39746,39837,39926,40015,40104,40191,40278,40
365,40450,40535,40619,

      40703,40786,40868,40949,41029,41109,41188,41267,41344,41421,41497,41573,41647,41721,4179
4,41867,41938,42009,42079,42148,42217,42285,42352,42418,42483,42548,42611,42674,42737,42798,428
59,42918,42977,43036,43093,43149,43205,43260,43314,43368,43420,43472,43522,43572,43622,43670,43
717,43764,43810,43855,

      43899,43942,43984,44026,44067,44107,44146,44184,44221,44258,44293,44328,44362,44395,4442
7,44458,44489,44518,44547,44575,44601,44628,44653,44677,44700,44723,44745,44765,44785,44804,448
23,44840,44856,44872,44886,44900,44913,44925,44936,44946,44956,44964,44972,44978,44984,44989,44
993,44996,44998,45000,

      45000,45000,44998,44996,44993,44989,44984,44978,44972,44964,44956,44946,44936,44925,4491
3,44900,44886,44872,44856,44840,44823,44804,44785,44765,44745,44723,44700,44677,44653,44628,446
01,44575,44547,44518,44489,44458,44427,44395,44362,44328,44293,44258,44221,44184,44146,44107,44
067,44026,43984,43942,

      43899,43855,43810,43764,43717,43670,43622,43572,43522,43472,43420,43368,43314,43260,4320
5,43149,43093,43036,42977,42918,42859,42798,42737,42674,42611,42548,42483,42418,42352,42285,422
```

```
17,42148,42079,42009,41938,41867,41794,41721,41647,41573,41497,41421,41344,41267,41188,41109,41
029,40949,40868,40786,

      40703,40619,40535,40450,40365,40278,40191,40104,40015,39926,39837,39746,39655,39563,3947
1,39377,39284,39189,39094,38998,38902,38805,38707,38609,38510,38410,38310,38209,38107,38005,379
02,37799,37695,37590,37485,37380,37273,37166,37059,36951,36842,36733,36623,36513,36402,36290,36
178,36066,35953,35839,

      35725,35611,35495,35380,35264,35147,35030,34912,34794,34675,34556,34437,34316,34196,3407
5,33953,33832,33709,33586,33463,33339,33215,33091,32966,32841,32715,32589,32462,32335,32208,320
80,31952,31823,31695,31565,31436,31306,31176,31045,30914,30783,30651,30519,30387,30254,30122,29
988,29855,29721,29587,

      29453,29318,29183,29048,28913,28777,28641,28505,28369,28232,28096,27958,27821,27684,2754
6,27408,27270,27132,26993,26855,26716,26577,26438,26299,26159,26020,25880,25740,25600,25460,253
20,25180,25039,24899,24758,24617,24477,24336,24195,24054,23913,23772,23630,23489,23348,23207,23
065,22924,22783,22641,

      22500,22359,22217,22076,21935,21793,21652,21511,21370,21228,21087,20946,20805,20664,2052
3,20383,20242,20101,19961,19820,19680,19540,19400,19260,19120,18980,18841,18701,18562,18423,182
84,18145,18007,17868,17730,17592,17454,17316,17179,17042,16904,16768,16631,16495,16359,16223,16
087,15952,15817,15682,

      15547,15413,15279,15145,15012,14878,14746,14613,14481,14349,14217,14086,13955,13824,1369
4,13564,13435,13305,13177,13048,12920,12792,12665,12538,12411,12285,12159,12034,11909,11785,116
61,11537,11414,11291,11168,11047,10925,10804,10684,10563,10444,10325,10206,10088,9970,9853,9736
,9620,9505,9389,

      9275,9161,9047,8934,8822,8710,8598,8487,8377,8267,8158,8049,7941,7834,7727,7620,7515,741
0,7305,7201,7098,6995,6893,6791,6690,6590,6490,6391,6293,6195,6098,6002,5906,5811,5716,5623,552
9,5437,5345,5254,5163,5074,4985,4896,4809,4722,4635,4550,4465,4381,

      4297,4214,4132,4051,3971,3891,3812,3733,3656,3579,3503,3427,3353,3279,3206,3133,3062,299
1,2921,2852,2783,2715,2648,2582,2517,2452,2389,2326,2263,2202,2141,2082,2023,1964,1907,1851,179
5,1740,1686,1632,1580,1528,1478,1428,1378,1330,1283,1236,1190,1145,

      1101,1058,1016,974,933,893,854,816,779,742,707,672,638,605,573,542,511,482,453,425,399,3
72,347,323,300,277,255,235,215,196,177,160,144,128,114,100,87,75,64,54,44,36,28,22,16,11,7,4,2,
0,

      0,0,2,4,7,11,16,22,28,36,44,54,64,75,87,100,114,128,144,160,177,196,215,235,255,277,300,
323,347,372,399,425,453,482,511,542,573,605,638,672,707,742,779,816,854,893,933,974,1016,1058,

      1101,1145,1190,1236,1283,1330,1378,1428,1478,1528,1580,1632,1686,1740,1795,1851,1907,196
4,2023,2082,2141,2202,2263,2326,2389,2452,2517,2582,2648,2715,2783,2852,2921,2991,3062,3133,320
6,3279,3353,3427,3503,3579,3656,3733,3812,3891,3971,4051,4132,4214,

      4297,4381,4465,4550,4635,4722,4809,4896,4985,5074,5163,5254,5345,5437,5529,5623,5716,581
1,5906,6002,6098,6195,6293,6391,6490,6590,6690,6791,6893,6995,7098,7201,7305,7410,7515,7620,772
7,7834,7941,8049,8158,8267,8377,8487,8598,8710,8822,8934,9047,9161,

      9275,9389,9505,9620,9736,9853,9970,10088,10206,10325,10444,10563,10684,10804,10925,11047
,11168,11291,11414,11537,11661,11785,11909,12034,12159,12285,12411,12538,12665,12792,12920,1304
8,13177,13305,13435,13564,13694,13824,13955,14086,14217,14349,14481,14613,14746,14878,15012,151
45,15279,15413,

      15547,15682,15817,15952,16087,16223,16359,16495,16631,16768,16904,17042,17179,17316,1745
4,17592,17730,17868,18007,18145,18284,18423,18562,18701,18841,18980,19120,19260,19400,19540,196
80,19820,19961,20101,20242,20383,20523,20664,20805,20946,21087,21228,21370,21511,21652,21793,21
935,22076,22217,22359,
};



uint32_t ar[]={32768,32973,33179,33385,33591,33797,34003,34208,
```

34414,34619,34825,35030,35236,35441,35646,35851,
36056,36261,36466,36670,36874,37079,37283,37486,
37690,37893,38097,38300,38503,38705,38908,39110,
39311,39513,39714,39916,40116,40317,40517,40717,
40916,41116,41315,41513,41711,41909,42107,42304,
42501,42697,42893,43089,43284,43479,43673,43867,
44061,44254,44446,44638,44830,45021,45212,45402,
45592,45781,45970,46158,46346,46533,46719,46905,
47091,47276,47460,47644,47827,48009,48191,48373,
48553,48733,48913,49092,49270,49448,49624,49801,
49976,50151,50325,50499,50671,50844,51015,51186,
51355,51525,51693,51861,52028,52194,52359,52524,
52688,52851,53013,53175,53335,53495,53654,53813,
53970,54126,54282,54437,54591,54744,54896,55048,
55198,55348,55497,55645,55792,55938,56083,56227,
56370,56513,56654,56794,56934,57073,57210,57347,
57482,57617,57751,57884,58015,58146,58276,58404,
58532,58659,58785,58909,59033,59155,59277,59397,
59517,59635,59753,59869,59984,60098,60211,60323,
60434,60544,60652,60760,60867,60972,61076,61179,
61281,61382,61482,61580,61678,61774,61870,61964,
62056,62148,62239,62328,62416,62503,62589,62674,
62758,62840,62921,63001,63080,63158,63234,63309,
63383,63456,63527,63598,63667,63735,63802,63867,
63931,63994,64056,64117,64176,64234,64291,64346,
64401,64454,64506,64556,64605,64654,64700,64746,
64790,64833,64875,64915,64955,64993,65029,65065,
65099,65132,65163,65193,65222,65250,65277,65302,
65326,65348,65370,65390,65408,65426,65442,65457,
65470,65483,65494,65503,65512,65519,65525,65529,
65532,65534,65535,65534,65532,65529,65525,65519,
65512,65503,65494,65483,65470,65457,65442,65426,
65408,65390,65370,65348,65326,65302,65277,65250,
65222,65193,65163,65132,65099,65065,65029,64993,
64955,64915,64875,64833,64790,64746,64700,64654,
64605,64556,64506,64454,64401,64346,64291,64234,
64176,64117,64056,63994,63931,63867,63802,63735,
63667,63598,63527,63456,63383,63309,63234,63158,
63080,63001,62921,62840,62758,62674,62589,62503,
62416,62328,62239,62148,62056,61964,61870,61774,
61678,61580,61482,61382,61281,61179,61076,60972,
60867,60760,60652,60544,60434,60323,60211,60098,
59984,59869,59753,59635,59517,59397,59277,59155,
59033,58909,58785,58659,58532,58404,58276,58146,
58015,57884,57751,57617,57482,57347,57210,57073,
56934,56794,56654,56513,56370,56227,56083,55938,
55792,55645,55497,55348,55198,55048,54896,54744,
54591,54437,54282,54126,53970,53813,53654,53495,
53335,53175,53013,52851,52688,52524,52359,52194,
52028,51861,51693,51525,51355,51186,51015,50844,
50671,50499,50325,50151,49976,49801,49624,49448,
49270,49092,48913,48733,48553,48373,48191,48009,
47827,47644,47460,47276,47091,46905,46719,46533,
46346,46158,45970,45781,45592,45402,45212,45021,
44830,44638,44446,44254,44061,43867,43673,43479,
43284,43089,42893,42697,42501,42304,42107,41909,
41711,41513,41315,41116,40916,40717,40517,40317,
40116,39916,39714,39513,39311,39110,38908,38705,
38503,38300,38097,37893,37690,37486,37283,37079,
36874,36670,36466,36261,36056,35851,35646,35441,
35236,35030,34825,34619,34414,34208,34003,33797,
33591,33385,33179,32973,32768,32562,32356,32150,
31944,31738,31532,31327,31121,30916,30710,30505,
30299,30094,29889,29684,29479,29274,29069,28865,
28661,28456,28252,28049,27845,27642,27438,27235,

```
            27032,26830,26627,26425,26224,26022,25821,25619,
            25419,25218,25018,24818,24619,24419,24220,24022,
            23824,23626,23428,23231,23034,22838,22642,22446,
            22251,22056,21862,21668,21474,21281,21089,20897,
            20705,20514,20323,20133,19943,19754,19565,19377,
            19189,19002,18816,18630,18444,18259,18075,17891,
            17708,17526,17344,17162,16982,16802,16622,16443,
            16265,16087,15911,15734,15559,15384,15210,15036,
            14864,14691,14520,14349,14180,14010,13842,13674,
            13507,13341,13176,13011,12847,12684,12522,12360,
            12200,12040,11881,11722,11565,11409,11253,11098,
            10944,10791,10639,10487,10337,10187,10038,9890,
            9743,9597,9452,9308,9165,9022,8881,8741,
            8601,8462,8325,8188,8053,7918,7784,7651,
            7520,7389,7259,7131,7003,6876,6750,6626,
            6502,6380,6258,6138,6018,5900,5782,5666,
            5551,5437,5324,5212,5101,4991,4883,4775,
            4668,4563,4459,4356,4254,4153,4053,3955,
            3857,3761,3665,3571,3479,3387,3296,3207,
            3119,3032,2946,2861,2777,2695,2614,2534,
            2455,2377,2301,2226,2152,2079,2008,1937,
            1868,1800,1733,1668,1604,1541,1479,1418,
            1359,1301,1244,1189,1134,1081,1029,979,
            930,881,835,789,745,702,660,620,
            580,542,506,470,436,403,372,342,
            313,285,258,233,209,187,165,145,
            127,109,93,78,65,52,41,32,
            23,16,10,6,3,1,0,1,
            3,6,10,16,23,32,41,52,
            65,78,93,109,127,145,165,187,
            209,233,258,285,313,342,372,403,
            436,470,506,542,580,620,660,702,
            745,789,835,881,930,979,1029,1081,
            1134,1189,1244,1301,1359,1418,1479,1541,
            1604,1668,1733,1800,1868,1937,2008,2079,
            2152,2226,2301,2377,2455,2534,2614,2695,
            2777,2861,2946,3032,3119,3207,3296,3387,
            3479,3571,3665,3761,3857,3955,4053,4153,
            4254,4356,4459,4563,4668,4775,4883,4991,
            5101,5212,5324,5437,5551,5666,5782,5900,
            6018,6138,6258,6380,6502,6626,6750,6876,
            7003,7131,7259,7389,7520,7651,7784,7918,
            8053,8188,8325,8462,8601,8741,8881,9022,
            9165,9308,9452,9597,9743,9890,10038,10187,
            10337,10487,10639,10791,10944,11098,11253,11409,
            11565,11722,11881,12040,12200,12360,12522,12684,
            12847,13011,13176,13341,13507,13674,13842,14010,
            14180,14349,14520,14691,14864,15036,15210,15384,
            15559,15734,15911,16087,16265,16443,16622,16802,
            16982,17162,17344,17526,17708,17891,18075,18259,
            18444,18630,18816,19002,19189,19377,19565,19754,
            19943,20133,20323,20514,20705,20897,21089,21281,
            21474,21668,21862,22056,22251,22446,22642,22838,
            23034,23231,23428,23626,23824,24022,24220,24419,
            24619,24818,25018,25218,25419,25619,25821,26022,
            26224,26425,26627,26830,27032,27235,27438,27642,
            27845,28049,28252,28456,28661,28865,29069,29274,
            29479,29684,29889,30094,30299,30505,30710,30916,
            31121,31327,31532,31738,31944,32150,32356,32562,};
void TIM1_BRK_UP_TRG_COM_IRQHandler(void){
      for(i=0;i<=1000;++i){
            TIM1->CCR3 = ar[i];
            if(i>=1000){
                  i=0;
```

```
            }
            TIM1 -> SR &= ~ (1U << 0);
      }

}
int main(void) {
      RCC->APBENR2 |= RCC_APBENR2_TIM1EN;
      RCC->IOPENR |= RCC_IOPENR_GPIOAEN;
      GPIOA->MODER &= ~ GPIO_MODER_MODE10_0;
      GPIOA->MODER |= GPIO_MODER_MODE10_1;

      GPIOA->AFR[1] = (2U<<8);
      timer1_init();
      TIM1 ->PSC = 0;
      TIM1->ARR = 45000;
      TIM1->CCR3 = 1000;
      TIM1->CCMR2 |= TIM_CCMR2_OC3M_1 | TIM_CCMR2_OC3M_2 |TIM_CCMR2_OC3PE;
      TIM1->CCER |= TIM_CCER_CC3E;
      TIM1->BDTR |= TIM_BDTR_MOE;
      TIM1->CR1 |= TIM_CR1_CEN;
      TIM1 -> EGR |= TIM_EGR_UG;
    while(1) {

    }

    return 0;
}
```

nucleo.c

```
/*
 * nucleo.c
 *
 *  Created on: Nov 29, 2021
 *      Author: Deniz
 */
#include "nucleo.h"
#include "stm32g0xx.h"

#define KILO    1000
#define MEGA    1000000
void nucleo_PA0_button_init(){
      RCC -> IOPENR |= (1U <<0 );
      GPIOA -> MODER &= ~ (3U << 0);
      GPIOA -> PUPDR &= ~ (3U << 0);
      GPIOA -> PUPDR |=  (2U << 0);
}
int nucleo_PA0_button_read(void) {

      int a = ((GPIOA -> IDR >> 2 ) & 0x01);
            if (a) return 0;
            else return 1;

}
void nucleo_PA0_button_INT(){
      EXTI -> RTSR1 |= (1U <<0 );
      EXTI -> EXTICR[0] |= (0U <<0 );

      EXTI -> IMR1 |= (1U <<0 );
      NVIC_SetPriority(EXTI0_1_IRQn , 1);
      NVIC_EnableIRQ(EXTI0_1_IRQn);
}
void nucleo_PA0_button_statclear(){
      EXTI -> RPR1 &= ~ (1U << 0);
}
```

```c
void nucleo_led_init(void){
	RCC -> IOPENR |= (1U <<2 );
	GPIOC -> MODER &= ~ (3U << 2*6);
	GPIOC -> MODER |= (1U << 2*6);
	GPIOC -> BRR |= (1U << 6);

}
void nucleo_led_set(void){
	GPIOC -> ODR |= (1U << 6) ;
}
void nucleo_led_clear(void){
	GPIOC -> BRR |= (1U << 6);
}
void nucleo_led_toggle(void){
	GPIOC -> ODR ^= (1U << 6) ;
}
void nucleo_button_init(void){
	RCC -> IOPENR |= (1U << 5 );
	GPIOF -> MODER &= ~ (3U << 2*2);

}
int nucleo_button_read(void) {

	int a = ((GPIOF -> IDR >> 2 ) & 0x01);
		if (a) return 0;
		else return 1;

}
void nucleo_ext_led_init(void){

	RCC -> IOPENR |= (1U <<0 );
	GPIOB -> MODER &= ~ (3U << 2*4);
	GPIOB -> MODER |= (1U << 2*4);
	GPIOB -> BRR |= (1U << 4);

}
void nucleo_ext_led_set(void){
	GPIOB -> ODR |= (1U <<4 );
}
void nucleo_ext_led_clear(void){
	GPIOB -> BRR |= (1U <<4 );
}
void nucleo_ext_led_toggle(void){
	GPIOB -> ODR ^= (1U <<4 );
}
void timer1_init(void) {

	RCC -> APBENR2 |= (1U << 11 );
	TIM1 -> CR1 = 0;
	TIM1 -> CR1 |= (1 << 7 );
	TIM1 -> CNT = 0;
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16000;
	TIM1 -> DIER |= (1 << 0 );
	TIM1 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn , 1);
	NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
}
void timer1_s(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16000;
}
void timer2_s(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 16000;
```

```
}

void timer1_s2(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 8000;
}
void timer2_s2(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 8000;
}
void timer1_s3(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 1600;
}
void timer2_s3(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 1600;
}
void timer1_s4(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 160;
}
void timer2_s4(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 160;
}

void timer1_s5(void){
	TIM1 -> PSC = 999;
	TIM1 -> ARR = 16;
}
void timer2_s5(void){
	TIM2 -> PSC = 999;
	TIM2 -> ARR = 16;
}
void timer2_init(void) {
	SystemCoreClockUpdate();
	RCC -> APBENR1 |= (1U << 0 );
	TIM2 -> CR1 = 0;
	TIM2 -> CR1 |= (1 << 7 );
	TIM2 -> CNT = 0;
	TIM2 -> DIER |= (1 << 0 );
	TIM2 -> CR1 |= (1 << 0 );
	NVIC_SetPriority(TIM2_IRQn  , 0);
	NVIC_EnableIRQ(TIM2_IRQn );
}
void systic_init(void){
	SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;
	SysTick->VAL=0;
	SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;
	NVIC_EnableIRQ(SysTick_IRQn);
	NVIC_SetPriority (SysTick_IRQn,0);
}
void timer1_statclear(void){
	TIM1 -> SR &= ~ (1U << 0);
}
void timer2_statclear(void){
	TIM2 -> SR &= ~ (1U << 0);
}

void systick_delay_ms() {
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / KILO));
}
void systick_delay_s(){
```

```
	SystemCoreClockUpdate();
	SysTick_Config((SystemCoreClock / MEGA));
}
// Project functions
// GPIO Functions
// Input Init
void Init_PA0_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*0);
}
void Init_PA1_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*1);
}
void Init_PA2_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*2);
}
void Init_PA3_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*3);
}
void Init_PA4_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*4);
}
void Init_PA5_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*5);
}
void Init_PA6_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*6);
}
void Init_PA7_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*7);
}
void Init_PA8_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*8);

}
void Init_PA9_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*9);

}
void Init_PA10_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*10);

}
void Init_PA11_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*11);

}
void Init_PA12_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*12);

}
void Init_PA13_Input(){
    RCC->IOPENR |= (1U << 0);
```

```
    GPIOA->MODER &= ~(3U << 2*13);

}
void Init_PA14_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*14);

}
void Init_PA15_Input(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*15);

}
void Init_PB0_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*0);

}
void Init_PB1_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*1);

}
void Init_PB2_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*2);

}
void Init_PB3_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*3);

}
void Init_PB4_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*4);
}
void Init_PB5_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*5);
}
void Init_PB6_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*6);
}
void Init_PB7_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*7);
}
void Init_PB8_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*8);
}
void Init_PB9_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*9);
}
void Init_PB10_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*10);
}
void Init_PB11_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*11);
}
```

```
void Init_PB12_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*12);
}
void Init_PB13_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*13);
}
void Init_PB14_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*14);
}
void Init_PB15_Input(){
    RCC->IOPENR |= (1U << 1);
    GPIOA->MODER &= ~(3U << 2*15);
}
//Input Functions
//Output Init
void Init_PB0_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*0);
    GPIOB->MODER |= (1U << 2*0);
}
void Init_PB1_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*1);
    GPIOB->MODER |= (1U << 2*1);
}
void Init_PB2_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*2);
    GPIOB->MODER |= (1U << 2*2);
}
void Init_PB3_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*3);
    GPIOB->MODER |= (1U << 2*3);
}
void Init_PB4_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*4);
    GPIOB->MODER |= (1U << 2*4);
}
void Init_PB5_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*5);
    GPIOB->MODER |= (1U << 2*5);
}
void Init_PB6_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*6);
    GPIOB->MODER |= (1U << 2*6);
}
void Init_PB7_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*7);
    GPIOB->MODER |= (1U << 2*7);
}
void Init_PB8_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*8);
    GPIOB->MODER |= (1U << 2*8);
}
void Init_PB9_Output(){
    RCC->IOPENR |= (1U << 1);
```

```
    GPIOB->MODER &= ~(3U << 2*9);
    GPIOB->MODER |= (1U << 2*9);
}
void Init_PB10_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*10);
    GPIOB->MODER |= (1U << 2*10);
}
void Init_PB11_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*11);
    GPIOB->MODER |= (1U << 2*11);
}
void Init_PB12_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*12);
    GPIOB->MODER |= (1U << 2*12);
}
void Init_PB13_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*13);
    GPIOB->MODER |= (1U << 2*13);
}
void Init_PB14_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*14);
    GPIOB->MODER |= (1U << 2*14);
}
void Init_PB15_Output(){
    RCC->IOPENR |= (1U << 1);
    GPIOB->MODER &= ~(3U << 2*15);
    GPIOB->MODER |= (1U << 2*15);
}
void Init_PA0_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*0);
    GPIOB->MODER |= (1U << 2*0);
}
void Init_PA1_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*1);
    GPIOB->MODER |= (1U << 2*1);
}
void Init_PA2_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*2);
    GPIOB->MODER |= (1U << 2*2);
}
void Init_PA3_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*3);
    GPIOB->MODER |= (1U << 2*3);
}
void Init_PA4_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*4);
    GPIOB->MODER |= (1U << 2*4);
}
void Init_PA5_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*5);
    GPIOB->MODER |= (1U << 2*5);
}
void Init_PA6_Output(){
    RCC->IOPENR |= (1U << 0);
```

```
    GPIOB->MODER &= ~(3U << 2*6);
    GPIOB->MODER |= (1U << 2*6);
}
void Init_PA7_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*7);
    GPIOB->MODER |= (1U << 2*7);
}
void Init_PA8_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*8);
    GPIOB->MODER |= (1U << 2*8);
}
void Init_PA9_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*9);
    GPIOB->MODER |= (1U << 2*9);
}
void Init_PA10_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOA->MODER &= ~(3U << 2*10);
    GPIOA->MODER |= (1U << 2*10);
}
void Init_PA11_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*11);
    GPIOB->MODER |= (1U << 2*11);
}
void Init_PA12_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*12);
    GPIOB->MODER |= (1U << 2*12);
}
void Init_PA13_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*13);
    GPIOB->MODER |= (1U << 2*13);
}
void Init_PA14_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*14);
    GPIOB->MODER |= (1U << 2*14);
}
void Init_PA15_Output(){
    RCC->IOPENR |= (1U << 0);
    GPIOB->MODER &= ~(3U << 2*15);
    GPIOB->MODER |= (1U << 2*15);
}
//output Functions
void Set_PB0(){
    GPIOB->ODR |= (1U<<0) ;
}
void Set_PB1(){
    GPIOB->ODR |= (1U<<1) ;
}
void Set_PB2(){
    GPIOB->ODR |= (1U<<2) ;
}
void Set_PB3(){
    GPIOB->ODR |= (1U<<3) ;
}
void Set_PB4(){
    GPIOB->ODR |= (1U<<4) ;
}
void Set_PB5(){
```

```
    GPIOB->ODR |= (1U<<5) ;
}
void Set_PB6(){
    GPIOB->ODR |= (1U<<6) ;
}
void Set_PB7(){
    GPIOB->ODR |= (1U<<7) ;
}
void Set_PB8(){
    GPIOB->ODR |= (1U<<8) ;
}
void Set_PB9(){
    GPIOB->ODR |= (1U<<9) ;
}
void Set_PB10(){
    GPIOB->ODR |= (1U<<10) ;
}
void Set_PB11(){
    GPIOB->ODR |= (1U<<11) ;
}
void Set_PB12(){
    GPIOB->ODR |= (1U<<12) ;
}
void Set_PB13(){
    GPIOB->ODR |= (1U<<13) ;
}
void Set_PB14(){
    GPIOB->ODR |= (1U<<14) ;
}
void Set_PB15(){
    GPIOB->ODR |= (1U<<15) ;
}
void Set_PA0(){
    GPIOA->ODR |= (1U<<0) ;
}
void Set_PA1(){
    GPIOA->ODR |= (1U<<1) ;
}
void Set_PA2(){
    GPIOA->ODR |= (1U<<2) ;
}
void Set_PA3(){
    GPIOA->ODR |= (1U<<3) ;
}
void Set_PA4(){
    GPIOA->ODR |= (1U<<4) ;
}
void Set_PA5(){
    GPIOA->ODR |= (1U<<5) ;
}
void Set_PA6(){
    GPIOA->ODR |= (1U<<6) ;
}
void Set_PA7(){
    GPIOA->ODR |= (1U<<7) ;
}
void Set_PA8(){
    GPIOA->ODR |= (1U<<8) ;
}
void Set_PA9(){
    GPIOA->ODR |= (1U<<9) ;
}
void Set_PA10(){
    GPIOA->ODR |= (1U<<10) ;
}
```

```
void Set_PA11(){
    GPIOA->ODR |= (1U<<11) ;
}
void Set_PA12(){
    GPIOA->ODR |= (1U<<12) ;
}
void Set_PA13(){
    GPIOA->ODR |= (1U<<13) ;
}
void Set_PA14(){
    GPIOA->ODR |= (1U<<14) ;
}
void Set_PA15(){
    GPIOA->ODR |= (1U<<15) ;
}
void Clear_PB0(){
    GPIOB->ODR &= ~(1U<<0) ;
}
void Clear_PB1(){
    GPIOB->ODR &= ~(1U<<1) ;
}
void Clear_PB2(){
    GPIOB->ODR &= ~(1U<<2) ;
}
void Clear_PB3(){
    GPIOB->ODR &= ~(1U<<3) ;
}
void Clear_PB4(){
    GPIOB->ODR &= ~(1U<<4) ;
}
void Clear_PB5(){
    GPIOB->ODR &= ~(1U<<5) ;
}
void Clear_PB6(){
    GPIOB->ODR &= ~(1U<<6) ;
}
void Clear_PB7(){
    GPIOB->ODR &= ~(1U<<7) ;
}
void Clear_PB8(){
    GPIOB->ODR &= ~(1U<<8) ;
}
void Clear_PB9(){
    GPIOB->ODR &= ~(1U<<9) ;
}
void Clear_PB10(){
    GPIOB->ODR &= ~(1U<<10) ;
}
void Clear_PB11(){
    GPIOB->ODR &= ~(1U<<11) ;
}
void Clear_PB12(){
    GPIOB->ODR &= ~(1U<<12) ;
}
void Clear_PB13(){
    GPIOB->ODR &= ~(1U<<13) ;
}
void Clear_PB14(){
    GPIOB->ODR &= ~(1U<<14) ;
}
void Clear_PB15(){
    GPIOB->ODR &= ~(1U<<15) ;
}
void Clear_PA0(){
    GPIOA->ODR &= ~(1U<<0) ;
```

```
}
void Clear_PA1(){
    GPIOA->ODR &= ~(1U<<1) ;
}
void Clear_PA2(){
    GPIOA->ODR &= ~(1U<<2) ;
}
void Clear_PA3(){
    GPIOA->ODR &= ~(1U<<3) ;
}
void Clear_PA4(){
    GPIOA->ODR &= ~(1U<<4) ;
}
void Clear_PA5(){
    GPIOA->ODR &= ~(1U<<5) ;
}
void Clear_PA6(){
    GPIOA->ODR &= ~(1U<<6) ;
}
void Clear_PA7(){
    GPIOA->ODR &= ~(1U<<7) ;
}
void Clear_PA8(){
    GPIOA->ODR &= ~(1U<<8) ;
}
void Clear_PA9(){
    GPIOA->ODR &= ~(1U<<9) ;
}
void Clear_PA10(){
    GPIOA->ODR &= ~(1U<<10) ;
}
void Clear_PA11(){
    GPIOA->ODR &= ~(1U<<11) ;
}
void Clear_PA12(){
    GPIOA->ODR &= ~(1U<<12) ;
}
void Clear_PA13(){
    GPIOA->ODR &= ~(1U<<13) ;
}
void Clear_PA14(){
    GPIOA->ODR &= ~(1U<<14) ;
}
void Clear_PA15(){
    GPIOA->ODR &= ~(1U<<15) ;
}

void Toggle_PB0(){

      GPIOB->ODR ^= (1U<<0) ;
}
void Toggle_PB1(){

      GPIOB->ODR ^= (1U<<1) ;
}
void Toggle_PB2(){

      GPIOB->ODR ^= (1U<<2) ;
}
void Toggle_PB3(){

      GPIOB->ODR ^= (1U<<3) ;
}
void Toggle_PB4(){
```

```
	GPIOB->ODR ^= (1U<<4) ;
}
void Toggle_PB5(){

	GPIOB->ODR ^= (1U<<5) ;
}
void Toggle_PB6(){

	GPIOB->ODR ^= (1U<<6) ;
}
void Toggle_PB7(){

	GPIOB->ODR ^= (1U<<7) ;
}
void Toggle_PB8(){

	GPIOB->ODR ^= (1U<<8) ;
}
void Toggle_PB9(){

	GPIOB->ODR ^= (1U<<9) ;
}
void Toggle_PB10(){

	GPIOB->ODR ^= (1U<<10) ;
}
void Toggle_PB11(){

	GPIOB->ODR ^= (1U<<11) ;
}
void Toggle_PB12(){

	GPIOB->ODR ^= (1U<<12) ;
}
void Toggle_PB13(){

	GPIOB->ODR ^= (1U<<13) ;
}
void Toggle_PB14(){

	GPIOB->ODR ^= (1U<<14) ;
}
void Toggle_PB15(){

	GPIOB->ODR ^= (1U<<15) ;
}
void Toggle_PA0(){

	GPIOA->ODR ^= (1U<<0) ;
}
void Toggle_PA1(){

	GPIOA->ODR ^= (1U<<1) ;
}
void Toggle_PA2(){

	GPIOA->ODR ^= (1U<<2) ;
}
void Toggle_PA3(){

	GPIOA->ODR ^= (1U<<3) ;
}
void Toggle_PA4(){

	GPIOA->ODR ^= (1U<<4) ;
```

```
}
void Toggle_PA5(){

	GPIOA->ODR ^= (1U<<5) ;
}
void Toggle_PA6(){

	GPIOA->ODR ^= (1U<<6) ;
}
void Toggle_PA7(){

	GPIOA->ODR ^= (1U<<7) ;
}
void Toggle_PA8(){

	GPIOA->ODR ^= (1U<<8) ;
}
void Toggle_PA9(){

	GPIOA->ODR ^= (1U<<9) ;
}
void Toggle_PA10(){

	GPIOA->ODR ^= (1U<<10) ;
}
void Toggle_PA11(){

	GPIOA->ODR ^= (1U<<11) ;
}
void Toggle_PA12(){

	GPIOA->ODR ^= (1U<<12) ;
}
void Toggle_PA13(){

	GPIOA->ODR ^= (1U<<13) ;
}
void Toggle_PA14(){

	GPIOA->ODR ^= (1U<<14) ;
}
void Toggle_PA15(){

	GPIOA->ODR ^= (1U<<15) ;
}
```

nucleo.h

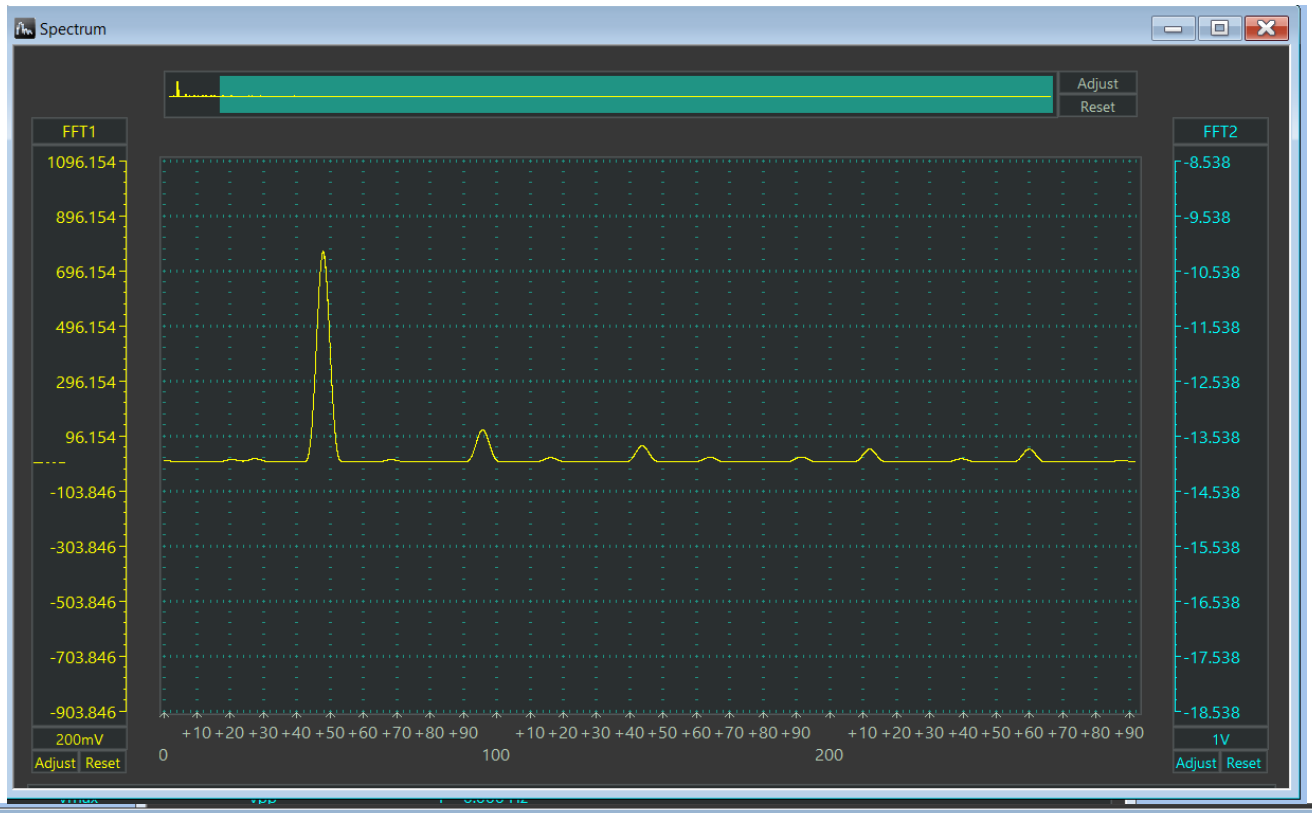
```
/*
 * nucleo.h
 *
 *  Created on: Nov 29, 2021
 *      Author: Deniz
 */
#ifndef NUCLEO_H_
#define NUCLEO_H_
// On-Board LED //
void nucleo_led_init();
void nucleo_led_set();
void nucleo_led_clear();
void nucleo_led_toggle();
void nucleo_ext_led_init();
void nucleo_ext_led_set();
void nucleo_ext_led_clear();
void nucleo_ext_led_toggle();
// Button Functions//
```

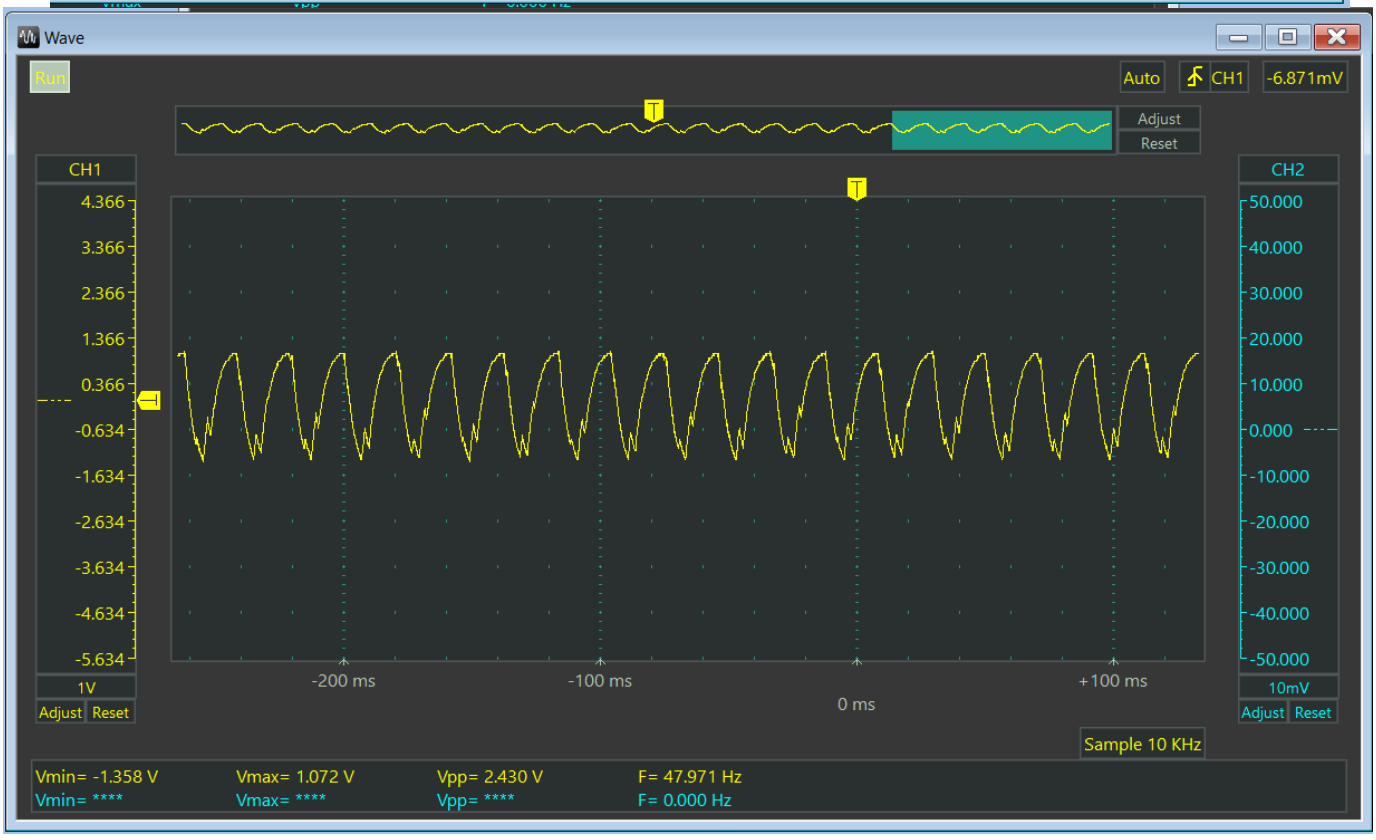
```
void nucleo_button_init();
int nucleo_button_read();
void nucleo_PA0_button_init();
int nucleo_PA0_button_read();
void nucleo_PA0_button_INT();
void nucleo_PA0_button_statclear();
//
// Timer interrupts
void timer1_init();
void timer2_init();
void timer2_s();
void systic_init();
void systick_delay_ms();
void systick_delay_s();
void timer1_interrupt();
void timer1_statclear();
void timer2_statclear(void);
void timer1_s();
void timer1_s2();
void timer1_s3();
void timer1_s4();
void timer1_s5();
// Project functions
// GPIO Functions
// Input Initialise
void Init_PA0_Input();
void Init_PA1_Input();
void Init_PA2_Input();
void Init_PA3_Input();
void Init_PA4_Input();
void Init_PA5_Input();
void Init_PA6_Input();
void Init_PA7_Input();
void Init_PA8_Input();
void Init_PA9_Input();
void Init_PA10_Input();
void Init_PA11_Input();
void Init_PA12_Input();
void Init_PA13_Input();
void Init_PA14_Input();
void Init_PA15_Input();
void Init_PB0_Input();
void Init_PB1_Input();
void Init_PB2_Input();
void Init_PB3_Input();
void Init_PB4_Input();
void Init_PB5_Input();
void Init_PB6_Input();
void Init_PB7_Input();
void Init_PB8_Input();
void Init_PB9_Input();
void Init_PB10_Input();
void Init_PB11_Input();
void Init_PB12_Input();
void Init_PB13_Input();
void Init_PB14_Input();
void Init_PB15_Input();
// Output Initialise
void Init_PB0_Output();
void Init_PB1_Output();
void Init_PB2_Output();
void Init_PB3_Output();
void Init_PB4_Output();
void Init_PB5_Output();
void Init_PB6_Output();
```

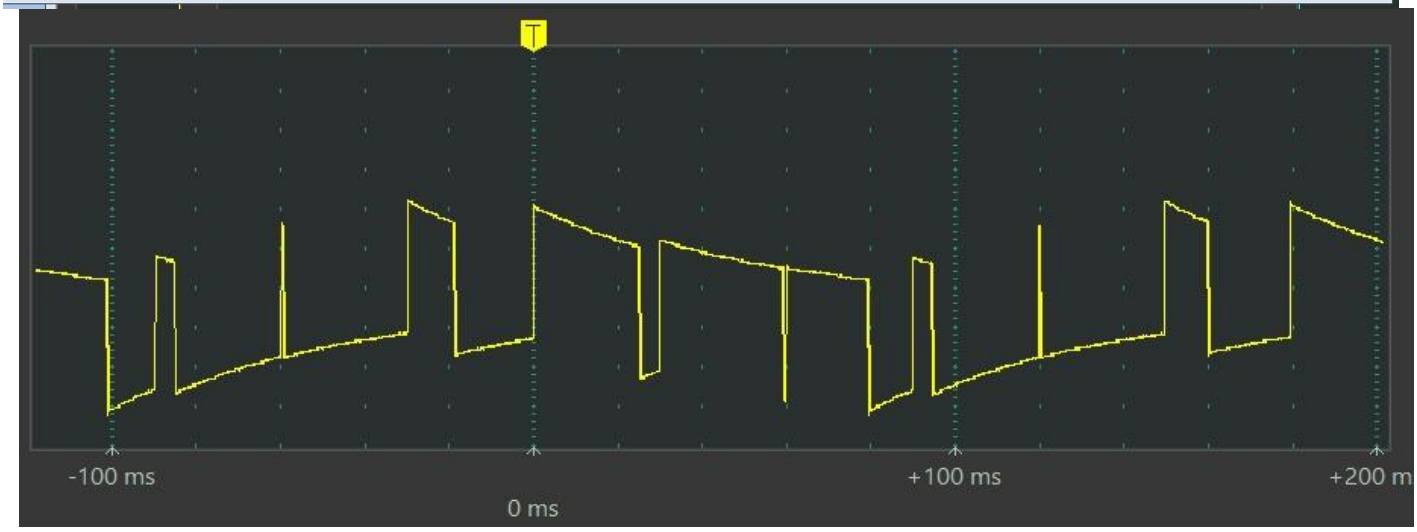
```
void Init_PB7_Output();
void Init_PB8_Output();
void Init_PB9_Output();
void Init_PB10_Output();
void Init_PB11_Output();
void Init_PB12_Output();
void Init_PB13_Output();
void Init_PB14_Output();
void Init_PB15_Output();
void Init_PA0_Output();
void Init_PA1_Output();
void Init_PA2_Output();
void Init_PA3_Output();
void Init_PA4_Output();
void Init_PA5_Output();
void Init_PA6_Output();
void Init_PA7_Output();
void Init_PA8_Output();
void Init_PA9_Output();
void Init_PA10_Output();
void Init_PA11_Output();
void Init_PA12_Output();
void Init_PA13_Output();
void Init_PA14_Output();
void Init_PA15_Output();
// Set Output
void Set_PB0();
void Set_PB1();
void Set_PB2();
void Set_PB3();
void Set_PB4();
void Set_PB5();
void Set_PB6();
void Set_PB7();
void Set_PB8();
void Set_PB9();
void Set_PB10();
void Set_PB11();
void Set_PB12();
void Set_PB13();
void Set_PB14();
void Set_PB15();
void Set_PA0();
void Set_PA1();
void Set_PA2();
void Set_PA3();
void Set_PA4();
void Set_PA5();
void Set_PA6();
void Set_PA7();
void Set_PA8();
void Set_PA9();
void Set_PA10();
void Set_PA11();
void Set_PA12();
void Set_PA13();
void Set_PA14();
void Set_PA15();
void Clear_PA0();
void Clear_PA1();
void Clear_PA2();
void Clear_PA3();
void Clear_PA4();
void Clear_PA5();
void Clear_PA6();
```

```
void Clear_PA7();
void Clear_PA8();
void Clear_PA9();
void Clear_PA10();
void Clear_PA11();
void Clear_PA12();
void Clear_PA13();
void Clear_PA14();
void Clear_PA15();
void Clear_PB4();
void Clear_PB0();
void Clear_PB1();
void Clear_PB2();
void Clear_PB3();
void Clear_PB4();
void Clear_PB5();
void Clear_PB6();
void Clear_PB7();
void Clear_PB8();
void Clear_PB9();
void Clear_PB10();
void Clear_PB11();
void Clear_PB12();
void Clear_PB13();
void Clear_PB14();
void Clear_PB15();
void Toggle_PB0();
void Toggle_PB1();
void Toggle_PB2();
void Toggle_PB3();
void Toggle_PB4();
void Toggle_PB5();
void Toggle_PB6();
void Toggle_PB7();
void Toggle_PB8();
void Toggle_PB9();
void Toggle_PB10();
void Toggle_PB11();
void Toggle_PB12();
void Toggle_PB13();
void Toggle_PB14();
void Toggle_PB15();
void Toggle_PA0();
void Toggle_PA1();
void Toggle_PA2();
void Toggle_PA3();
void Toggle_PA4();
void Toggle_PA5();
void Toggle_PA6();
void Toggle_PA7();
void Toggle_PA8();
void Toggle_PA9();
void Toggle_PA10();
void Toggle_PA11();
void Toggle_PA12();
void Toggle_PA13();
void Toggle_PA14();
void Toggle_PA15();
#endif /* NUCLEO_H_ */
```

Oscilloscope Images







Comments and Questions

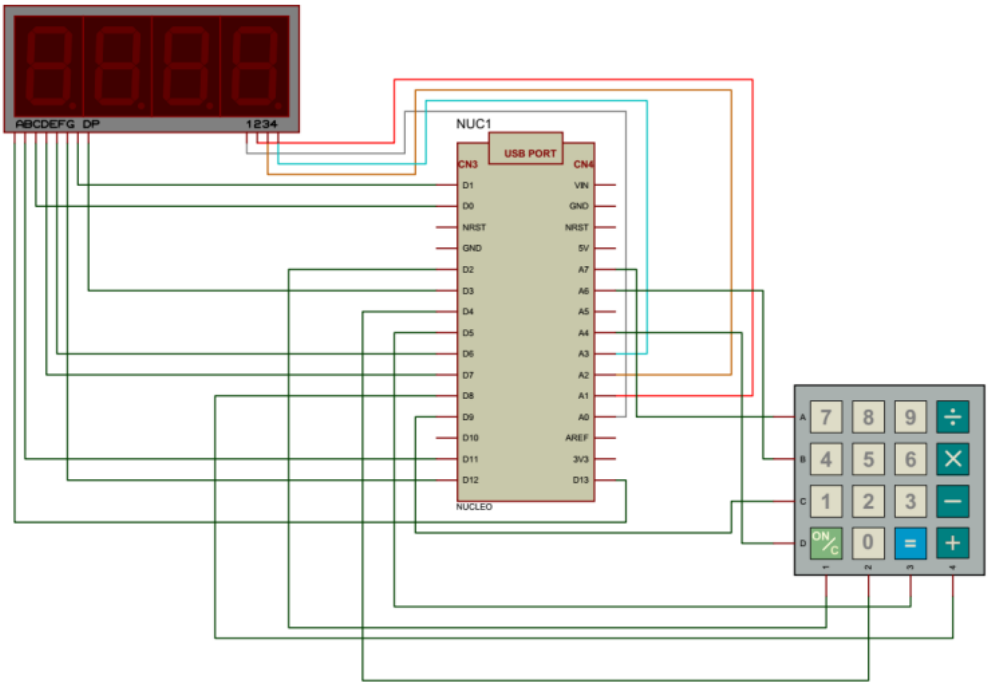
When we look at dc coupling we se pwm as expected but when we do ac coupling we see a form of modulation.

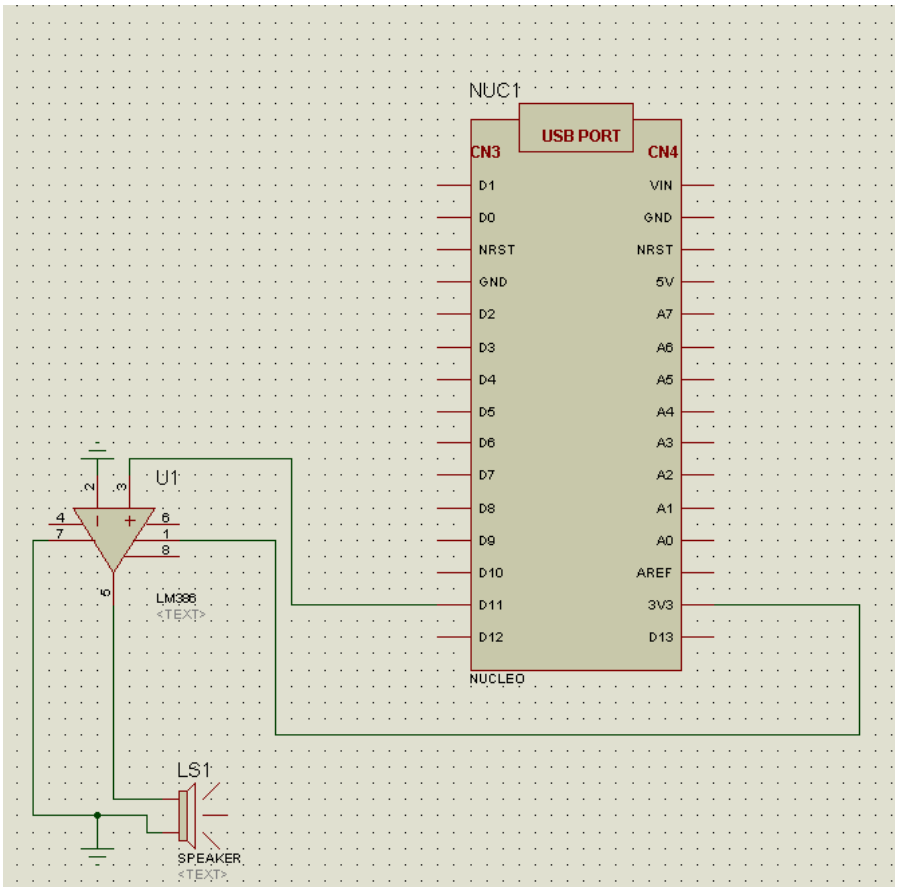
Video Link

https://youtu.be/rlTIdEFrL4A

Problem 4

Block Diagram





Code

main.c

```
#include "bsp.h"    //Board support package
#include "stm32g0xx.h"
#include "syd.h" // Library for calculator and SDD functions



int main(void) { //Empty main function it only calls init_xxx functions then rest at infinite
loop

BSP_system_init();
speaker_init();
SDD_init();
init_timer3();
membrane_init();
while(1){SDD_display();}

}
```

bsp.c

```
#include "bsp.h"
#include "stm32g0xx.h"

static volatile unsigned int tick ;
void BSP_system_init(){
	__disable_irq();
	BSP_sled_init();
	SysTick_Config(SystemCoreClock/1000);
	__enable_irq();
}

//SystemTick Functions

void delay(volatile unsigned int s){
	for(;s>0;s--);
}
//----------------------------


//GENERAL INPUT OUTPUT FUNCTIONS
void BSP_input_init(int gpio, int port){
//Set first variable to
// 0 ---> GPIOA
// 1 ---> GPIOB
// 2 ---> GPIOC
// 3 ---> GPIOD
// 5 ---> GPIOF
RCC->IOPENR |= (1U << gpio);
/* Setup as output */
switch (gpio) { // Sets the specific port as input according to its parameters. gpio value
determines which port, port value determines which pin
	case 0:
		GPIOA->MODER &= ~(3U << 2*(port));
		GPIOA->PUPDR &= ~(3U << 2*(port));
		GPIOA->PUPDR |= (2U  <<  2*(port));
		break;
	case 1:
```

```
            GPIOB->MODER &= ~(3U << 2*(port));
            GPIOB->PUPDR |= (2U  <<  2*(port));
            break;
      case 2:
            GPIOC->MODER &= ~(3U << 2*(port));
            GPIOC->PUPDR |= (2U  <<  2*(port));
            break;
      case 3:
            GPIOD->MODER &= ~(3U << 2*(port));
            GPIOD->PUPDR |= (2U  <<  2*(port));
            break;
      case 5:
            GPIOF->MODER &= ~(3U << 2*(port));
            GPIOF->PUPDR |= (2U  <<  2*(port));
      break;
default:
      break;
} }


void BSP_output_init(int gpio, int port){
RCC->IOPENR |= (1U << gpio);
switch (gpio) {    // Sets the specific port as output according to its parameters. gpio value
determines which port, port value determines which pin
      case 0:
            GPIOA->MODER &= ~(3U << 2*(port));
            GPIOA->MODER |= (1U << 2*port);
            break;
      case 1:
            GPIOB->MODER &= ~(3U << 2*(port));
            GPIOB->MODER |= (1U << 2*port);
            break;
      case 2:
            GPIOC->MODER &= ~(3U << 2*(port));
            GPIOC->MODER |= (1U << 2*port);
            break;
      case 3:
            GPIOD->MODER &= ~(3U << 2*(port));
            GPIOD->MODER |= (1U << 2*port);
            break;
      case 5:
            GPIOF->MODER &= ~(3U << 2*(port));
            GPIOF->MODER |= (1U << 2*port);
            break;
      default:
            break;
} }


int BSP_input_read(int gpio, int port){      //It reads the specific port according to its
parameters.
      int value=0;
      switch (gpio){
            case 0:
                  value = ((GPIOA->IDR >> port ) & 0x1);
                  break;
            case 1:
                  value = ((GPIOB->IDR >> port) & 0x1);
                  break;
            case 2:
                  value = ((GPIOC->IDR >> port ) & 0x1);
                  break;
            case 3:
                  value = ((GPIOD->IDR >> port ) & 0x1);
                  break;
```

```
		case 5:
			value = ((GPIOF->IDR >> port ) & 0x1);
		break;
	default:
		break;
	}
	if(value == 1){
		return port;
	}
	else
		return value;
}



//-----------------------------------------

//ON BOARD BUTTON AND LED FUNCTIONS

void BSP_sled_init(){
	GPIOC->MODER &= ~(3U << 2*6);
	GPIOC->MODER |= (1U << 2*6);
}

void BSP_sled_toggle(){
		GPIOC->ODR ^= (1U << 6);
}
void BSP_sled_set(){
	  GPIOC->ODR |= (1U << 6);
}
void BSP_sled_clear(){
	GPIOC->ODR  &= ~(1U << 6);
}
////-------------------------------------



////USART FUNCTIONS
void BSP_UART_init(unsigned int baud){
	//Enable IOA and USART2 clocks
	RCC->IOPENR |= (1U << 0);
	RCC->APBENR1 |=(1U << 17);
	//Setting PA2 as alternative function mode "10"
	GPIOA->MODER  &= ~(3U << 2*2);
	GPIOA->MODER |= (2U << 2*2);

	GPIOA->AFR[0] &= ~(0xFU << 2*4);
	GPIOA->AFR[0] |= (1 << 2*4);
	//Setting PA3 as alternative function mode "10"
	GPIOA->MODER  &= ~(3U << 2*3);
	GPIOA->MODER |= (2U << 2*3);
	//Choosing alternative functions from MUX

	GPIOA->AFR[0] &= ~(0xFU << 4*3);
	GPIOA->AFR[0] |= (1 << 3*4);

	//Setup USart2
	USART2->CR1 = 0;
	USART2-> CR1 |= (1<<3);  //Transmitter enabled
	USART2-> CR1 |= (1<<2);   //Reciever enabled
	//USART2-> CR1      |= (1<<5);   //Reciever enabled
```

```
	USART2->BRR = (uint16_t)(SystemCoreClock/baud);

	USART2-> CR1 |= (1 << 0); //Lowpower usart enabled


}

int _print(int fd,char *buf, int len){
	(void)fd;
	for(int i=0; i<len;++i){
		uart_tx(buf[i]);
	}
	return len;
}

void print(char *s) {
	int c = 0;
	while( s[c] != '\0'){
		c++;
	}
	 c = _print(0, s, c);
}

void uart_tx(unsigned char c){
		USART2->TDR = (uint16_t)c;
		while(! ( USART2->ISR & (1 << 6)));
}

unsigned char uart_rx(void){
	uint8_t data = (uint8_t)USART2->RDR;
	return data;
}
```

bsp.h

```
#ifndef BSP_H_
#define BSP_H_

//Board Support for Nucleo G031K8.
#define PoC  2
#define PoA  0
#define PoB  1
#define PoD  3
#define PoF  5

void BSP_system_init();
//SystemTick Functions
void delay(volatile unsigned int);
void SysTick_Handler(void);


//GENERAL INPUT OUTPUT FUNCTIONS
void BSP_input_init(int gpio, int port);
void BSP_output_init(int gpio, int port);
int BSP_input_read(int gpio, int port);


//ON BOARD BUTTON AND LED FUNCTIONS

void BSP_sled_set();
void BSP_sled_clear();
void BSP_sled_init();
void BSP_sled_toggle();
void BSP_sbutton_init();

//USART FUNCTIONS
```

```
void BSP_UART_init(unsigned int baud);
unsigned char uart_rx(void);
int _print(int fd,char *buf, int len);
void print(char *s);
void uart_tx(unsigned char c);



#endif
```

syd.h

```
#ifndef SYD_H_
#define SYD_H_

//Random frequencies determined as notes. These notations means nothing
#define cs 600U
#define ds 230U
#define es 390U
#define fs 304U
#define gs 202U
#define as 440U
#define bs 409U
#define Cs 200U
#define Ds 50U
#define Es 100U
#define Fs 208U
#define Gs 502U


void SDD_init(void); //Initialize SSD pins
void membrane_init(void); //Initialize Membrane pins and interrupts
void speaker_init(void);

void SysTick_Handler(void); //Systick handler

void arrange_sound(unsigned int note); //Reset Pwm module
void EXTI4_15_IRQHandler(void); //Interrupt handler

void SDD_clean();  //Clean SDD
void SDD_idle(void); //Idle mode displays 1807 (1801022007)
void SDD_display(void); //Non flickering 4 digit display value
void SDD_disp_num(int digit);//Display a digit
void SDD_set(int display_value); //Shows display_value on SDD

void init_timer3(); //Timer initialization
void TIM3_IRQHandler(); //Timer exception handler

void init_timer2(); //Timer initialization
void TIM2_IRQHandler(); //Timer exception handler

void arrange_sdd(int data);

void flag_handler(void);
void open_pad(void); //open all columns
void close_pad(void); //close all columns
int control_rows(void);   //control which row is pressed
void press_detect(void);  //detect which button is pressed
void rows_columns(void);  //Calls the necessary function according to presssed button

float float_handler(float ansfer); //Converts float values to int to print on SDD
void dot_add(int digit);  //Enables dot for float values
```

```
void words(int a);  //Overflow and Invalid
//Basic math functions
void k_add(void);
void k_sub(void);
void k_divide(void);
void k_multiply(void);
//Scientific mode
void k_log(void);
void k_ln(void);
void k_sqrt(void);
void k_sqr(void);
//Trigonometric functions
void k_sin(void);
void k_cos(void);
void k_cot(void);
void k_tan(void);
void pi(void);

#endif
```

syd.c

```
#include "bsp.h"
#include "syd.h"
#include "stm32g0xx.h"
#include "math.h"



static volatile  int loops[4]={0,0,0,0};//Intermediate value holder to easily change display
values
//Display Parameters           0     1    2   3     4    5    6    7    8   9      -    I    N
A    D    O   U    F    L  Blank
static uint16_t  numbers[21]={0x60,0x7E,0x60,0x32,0x2E,0x23,0x21,0x76,0x20,0x22};
static uint16_t digits[4]={0x32,0x31,0x23,0x13}; //HEX values to enable each digit
static volatile uint16_t   idle_timer=0; //idle timer

static volatile unsigned int tick; //Systick Handler

static volatile int rows; //Parameter to store the pressed row's pin value
static volatile int columns; //Parameter to store the pressed columns's pin value

static volatile unsigned int ansf;     //Stores float version of the answer
static volatile int disp; //Value to display
static volatile int disp_ans;   // Buffer parameter to change display value


static volatile int flags = 1; // Parameter to keep track of functional button pressed
static volatile int float_flags = 0; // Parameter to help displaying float values on SDD;

void SDD_init(void){
      //For my circuit
            /* Enable GPIOB and GPIOA clock */
       RCC->IOPENR |= (0x7);
       /* Setup PA0-PA1-PA4-PA5 pins as output */
    GPIOA->MODER &= ~(12586767U);
    GPIOA->MODER |= (0x400505);
    /* Setup PB0-7 pins as output */

       BSP_output_init(PoB,0);  //Setting PB9 as output
       BSP_output_init(PoB,1);  //Setting PB9 as output
       BSP_output_init(PoB,2);  //Setting PB9 as output
       BSP_output_init(PoB,3);  //Setting PB9 as output
       BSP_output_init(PoB,4);  //Setting PB9 as output
```

```
        BSP_output_init(PoB,6);  //Setting PB9 as output
        BSP_output_init(PoB,7);  //Setting PB9 as output
}


void SDD_clean(){  //clear the shown number and open digit to prevent shadows /silhouettes .
       GPIOB->ODR &= ~(223U<<0);
       GPIOA->ODR &= ~(51U<<0);
}

void SDD_idle(void){ //Idle mode shows firt and last 2 digits of 18010220007
       loops[0]=1;
       loops[1]=8;
       loops[2]=0;
       loops[3]=7;
       disp = 0;
       flags = 1;
       float_flags=0;
       disp_ans = 0;
       ansf = 0;
}

void SysTick_Handler(void){     //Systick handler
       tick++;
}




void SDD_display(void){         //Display the current values at each digit. Digits are arrayed
in mixed order to prevent one side being too bright
       volatile  uint32_t i=60; //Each time this function is called each digit gets toggled 60
times.
                                                //My logic was to create something similar to
60FPS to prevent flickering

for(;i>0;--i){
       SDD_disp_num(1); //Calls disp_num function for each digit
       SDD_disp_num(2);
       SDD_disp_num(3);
       SDD_disp_num(0);
       }
}


void SDD_disp_num(int digit){
       SDD_clean(); //Turns off the activated digit and old number value to prevent shadows
appearing between digit switches
        GPIOA->ODR |=  digits[digit];  //Turn on the specific digit by which the function is
called
        GPIOB->ODR |=  numbers[loops[digit]]; //loops[digit]  chooses the current cycle for the
specific digit

       //numbers[loops[digit]] chooses the correct hex value to show at the given digit
}



void system_core_update(){
       RCC->CR |= (0x1 << 24);   //PLL is enabled
       RCC->PLLCFGR |= (8U << 8); // PLL N factor is set to 8
       RCC->PLLCFGR |= (0x001U << 29); // PLL R factor is set to 2
       RCC->PLLCFGR |= (0x001U << 25); // PLL Q factor is set to 2
}
```

```
void TIM2_IRQHandler(){
	arrange_sound(ansf);
	TIM2->SR &=  ~(1U << 0); //Reset

}


void init_timer2(){ //Set to create exception each 0.0001 second
	RCC->CCIPR |= (0U << 1);  //Enabling TIM3
	TIM2->CR1 = 0;                  //RESET TIM3_CR1 register
	TIM2->CR1 |= (1 << 7); //AUTO RELOAD ENABLED
	TIM2->DIER |=(1 << 0); //UPDATE INTERRUPT ENABLED
	TIM2->CNT = 0 ;           //RESET COUNTER

	TIM2->PSC = 199;          //PRESCALER SET to 9
	TIM2->ARR =10;            //AUTORELOAD VALUE  (PSC+1*ARR)/SystemCoreClock=0.0001
	TIM2->CR1 |= (1 << 0);    //Counter enabled

	NVIC_SetPriority(TIM2_IRQn , 4); //Set to the lowest priority level
	NVIC_EnableIRQ(TIM2_IRQn);            //Enable interrupt

}
void speaker_init(void){

	RCC->IOPENR |= (7U << 0);       //GPIO clocks enabled

	GPIOB->MODER &= ~(3U << 2 * 5);        //GPIOB Port 5 selected as output
	GPIOB->MODER |= (2U << 2 * 5);


	GPIOB->AFR[0] &= ~(0xFU << 5*4); //PB5 selected as Alternate function 2
	GPIOB->AFR[0] |= (0001U << 5*4);
}



void init_timer3(){ //Set to create exception each 0.0001 second
	RCC->APBENR1 |= (1U << 1);      //Enabling TIM3

	TIM3->CR1 = 0;                  //RESET TIM3_CR1 register
	TIM3->CR1 |= (1U << 7);
	TIM3->CNT = 0 ;           //RESET COUNTER
	TIM3->PSC = 199;          //PRESCALER SET to
	TIM3->ARR =ansf;          //AUTORELOAD VALUE  SystemCoreClock/ARR*(1+PSC) = 250kHz

	TIM3->DIER |= (1U << 0);


	TIM3->CCMR1 &= ~(0X7U << 12);   //Resetting values
	TIM3->CCMR1 &= ~(0X1U << 24);   //Resetting values
	TIM3->CCMR1 |= (0X6U << 12);    // Channel 2 is selected as PWM

	TIM3->CCMR1 |= (1U << 11); // Channel 2 pre-load enabled
	TIM3->CCER  |= (1 << 4);//Channel 2 Capture Compare enabled

	TIM3->CCR2 = ((uint32_t)ansf)/2; //Half duty cycle
	TIM3->CR1 |= (1 << 0);    //Counter enabled


	NVIC_SetPriority(TIM3_IRQn, 1);
	NVIC_EnableIRQ(TIM3_IRQn);


}
```

```
void TIM3_IRQHandler(void) {
      if(float_flags == 1){

                  TIM3->ARR = (uint32_t) ansf;
                  init_timer3();
                  float_flags = 0;

      }

      TIM3->SR &= ~(1U << 0);
}

void membrane_init(void){
       /* Setup PA7-PA6-PA11-PA12 pins as input */
       BSP_input_init(PoA, 7);    //Setting PA7 as input
       BSP_input_init(PoA, 6);   //Setting PA6 as input
       BSP_input_init(PoA,8);       //Setting PA8 as input
       BSP_input_init(PoA,12);   //Setting PA12 as input

       BSP_output_init(PoA,15);  //Setting PA15 as output
       BSP_output_init(PoA,10);   //Setting PA10 as output
       BSP_output_init(PoA,9);   //Setting PA9 as output
       BSP_output_init(PoB,8);  //Setting PB9 as output

            EXTI->EXTICR[1] |= (0U << 8*2); //Setting PA6 as interrupt
            EXTI->RTSR1 |= (1U << 6);       //Rising edge triggered
            EXTI->IMR1  |= (1U << 6);       //Mask 1

            EXTI->EXTICR[1] |= (0U << 8*3); //Setting PA7 as interrupt
            EXTI->RTSR1 |= (1U << 7);       //Rising edge triggered
          EXTI->IMR1  |= (1U << 7);          //Mask 1


            EXTI->EXTICR[2] |= (0U << 8*0); //Setting PA8 as interrupt
            EXTI->RTSR1 |= (1U << 8);       //Rising edge triggered
            EXTI->IMR1  |= (1U << 8);       //Mask 1

          EXTI->EXTICR[3] |= (0U << 8*0);    //Third mux is selected for Pin 12-15 , 0U is used
to select PA 8*0 is used to select pin 12
          EXTI->RTSR1 |= (1U << 12);         //Rising edge triggered
          EXTI->IMR1  |= (1U << 12);         //Mask 1

          NVIC_SetPriority(EXTI4_15_IRQn , 0);     //Set to the highest priority level
          NVIC_EnableIRQ(EXTI4_15_IRQn);                 //Enable interrupt
}

void open_pad(void){
       GPIOA->ODR |= (1U << 15);      //Opening membrane columns
       GPIOA->ODR |= (1U << 10);      //Opening membrane columns
       GPIOA->ODR |= (1U << 9); //Opening membrane columns
       GPIOB->ODR |= (1U << 8); //Opening membrane columns
}

void close_pad(void){
            GPIOA->ODR &= ~(1U << 15);      //Closing membrane columns
            GPIOA->ODR &= ~(1U << 10);      //Closing membrane columns
            GPIOA->ODR &= ~(1U << 9); //Closing membrane columns
            GPIOB->ODR &= ~(1U << 8); //Closing membrane columns
}

void press_detect(void){
      close_pad();//Close all columns
      int buf_rows=0;
      GPIOA->ODR |= (1U << 15); //Opening a sing membrane column
      buf_rows = control_rows();      //Determining which row is pressed
```

```
	if (buf_rows != 0){			//If buf_rows value is changed
		rows = buf_rows;		//Set new rows and columns values and return
		columns = 15;
		return;}

	close_pad();
	GPIOA->ODR |= (1U << 10); //Opening membrane column
	buf_rows = control_rows();	//Determining which row is pressed
	if (buf_rows != 0){			//If buf_rows value is changed
		rows = buf_rows;		//Set new rows and columns values and return
		columns = 10;
		return;}

	close_pad();
	GPIOA->ODR |= (1U << 9);  //Opening membrane column
	buf_rows = control_rows();	//Determining which row is pressed
	if (buf_rows != 0){			//If buf_rows value is changed
		rows = buf_rows;		//Set new rows and columns values and return
		columns = 9;
		return;}

	close_pad();				//Close all pads
	GPIOB->ODR |= (1U << 8);  //Opening membrane column
	buf_rows = control_rows();	//Determining which row is pressed
	if (buf_rows != 0){			//If buf_rows value is changed
		rows = buf_rows;		//Set new rows and columns values and return
		columns = 8;
		return;}
}

int control_rows(void){
	int row = 0;
	//Checks each input ports value.
	//Control it one by one if any data is being read leave any other than 0 leave
immediately
	if (row == 0){
		row = BSP_input_read(PoA,7);}
	if (row == 0){
		row = BSP_input_read(PoA,6);}
	if (row == 0){
		row = BSP_input_read(PoA,8);}
	if (row == 0){
		row = BSP_input_read(PoA,12);}
	return row;
}


void arrange_sound(unsigned int note){
	//AUTORELOAD VALUE  SystemCoreClock/ARR*(1+PSC) = 250kHz
	//Resetting PWM for new frequency
	ansf = note;
	TIM3->ARR = (uint32_t) ansf;
	init_timer3();
	delay(80000);

}



void arrange_sdd(int data){
	switch (data) { // Sets the specific port as input according to its parameters. gpio
value determines which port, port value determines which pin


		case 0:
```

```
                        arrange_sound(Es);
                        SDD_set(Es);
                        break;
            case 1:
                  arrange_sound(cs);
                  SDD_set(cs);
                  break;
            case 2:
                              arrange_sound(ds);
                              SDD_set(ds);
                              break;
            case 3:
                              arrange_sound(es);
                              SDD_set(es);
                              break;
            case 4:
                              arrange_sound(fs);
                              SDD_set(fs);
                              break;
            case 5:
                              arrange_sound(gs);
                              SDD_set(gs);
                              break;
            case 6:
                              arrange_sound(bs);
                              SDD_set(bs);
                              break;
            case 7:
                              arrange_sound(as);
                              SDD_set(as);
                              break;
            case 8:
                              arrange_sound(Cs);
                              SDD_set(Cs);
                              break;
            case 9:
                              arrange_sound(Ds);
                              SDD_set(Ds);
                              break;
            case 14:
                              arrange_sound(0);
                              SDD_set(0000);
                              break;
            case 24:
                                          arrange_sound(Fs);
                                          SDD_set(Fs);
                                          break;
            case 34:
                                          arrange_sound(Gs);
                                          SDD_set(Gs);
                                          break;
      default:
            break;
      }

}

void rows_columns(void){
      open_pad(); // Open all membrane columns
      //Call arange SDD function according to button pressed
      //If its numerical calls function by its value
      //If its alphabetical call function by its XY value
      if(rows == 7){
            if(columns == 15){
                  arrange_sdd(1);
```

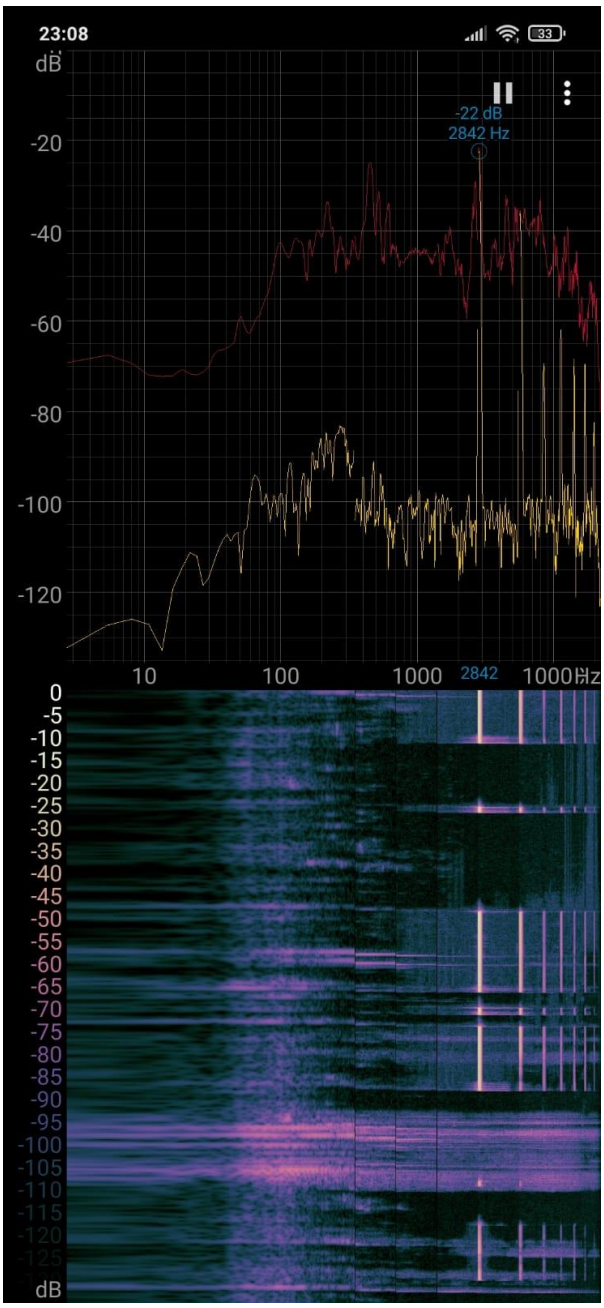
```
		}
		if(columns == 10){
				arrange_sdd(2);
			}
		if(columns == 9){
				arrange_sdd(3);
			}
		if(columns == 8){
				arrange_sdd(14);
			}
		}
	if(rows == 6){
		if(columns == 15){
			arrange_sdd(4);
		}
		if(columns == 10){
				arrange_sdd(5);
			}
		if(columns == 9){
				arrange_sdd(6);
			}
		if(columns == 8){
				arrange_sdd(24);
			}
		}
	if(rows == 8){
		if(columns == 15){
			arrange_sdd(7);
		}
		if(columns == 10){
				arrange_sdd(8);
			}
		if(columns == 9){
				arrange_sdd(9);
			}
		if(columns == 8){
				arrange_sdd(34);
			}
		}
	if(rows == 12){
		if(columns == 15){
			arrange_sdd(41);
		}
		if(columns == 10){
				arrange_sdd(0);
			}
		if(columns == 9){
				arrange_sdd(43);
			}
		if(columns == 8){
				arrange_sdd(44);
			}
		}
	return;
}

void SDD_set(int display_value){
	//Determine if its below zero or not
	if((display_value  < 0 )){ //if yes
		display_value*=-1; //make it positive
		loops[3]=display_value%10;	//Find its first decimal value
		loops[2]=((display_value%100)-loops[3])/10;  //Find its second decimal value
		loops[1]=((display_value%1000)-loops[2]-loops[3])/100;    //Find third decimal
value
```

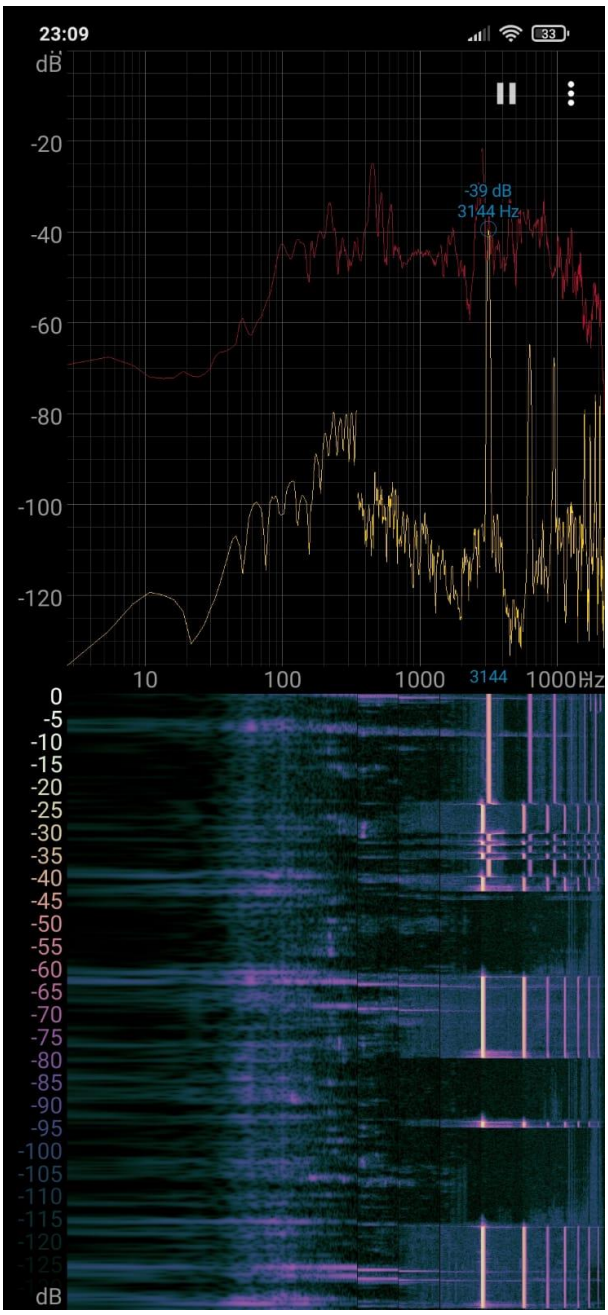
```
            loops[0]=((display_value%10000)-loops[1]-loops[2]-loops[3])/1000;      //Find
fourth decimal value
            if(loops[0] == 0){  //If fourth decimal is zero
                        loops[0]=10; //Make it "-" sign
                        if(loops[1] == 0){ //if third decimal is zero too close fourth and
make third decimal "-" sign
                                    loops[1]=10;
                                    loops[0]=19;
                                    if(loops[2] == 0){//if second decimal is zero too close third
and make second decimal "-" sign
                                                loops[2]=10;
                                                loops[1]=19;
                                                }
                                    }
                        }
      }
      else{ //if its positive
            loops[3]=display_value%10;      //Find its first decimal value
            loops[2]=((display_value%100)-loops[3])/10;  //Find its second decimal value
            loops[1]=((display_value%1000)-loops[2]-loops[3])/100;    //Find third decimal
value
            loops[0]=((display_value%10000)-loops[1]-loops[2]-loops[3])/1000;      //Find
fourth decimal value
            if(loops[0] == 0){ //If fourth decimal is zero
                  loops[0]=19; //close blank
                  if(loops[1] == 0){  //if third digit is zero too close that one too
                        loops[1]=19;
                        if(loops[2] == 0){  // if second digit is zero too close second digit
too
                                    loops[2]=19;
                                                      }
                                                }
                                          }
            }
}

void EXTI4_15_IRQHandler(void){
      //Small delay introduced to prevent bouincing
      delay(300);
      press_detect(); //Detect pressed button's row and column pins
      rows_columns(); //Arange SDD according to detected values
      EXTI->RPR1 |= (1U << rows);     //Set hardware raised flag to zero by software
                                                      //Sets only the interrupted pin to zero
      rows = 0;                                 //Reset row and column
      columns = 0;
}
```

Spectroid Görüntüleri



2842 Hz



3144 Hz

Comments and Questions

This code pretty hard to implement. We were not able to figure out why it didn't work when it didn't and why did it work when it did. The frequency values are not working on SSD. Timer 3 Channel 2 was implemented as PWM output. PB5 alternate function 2 is timer3 channel2. I disabled dot led on SSD , changed its pin with SSD B port. Which freed (D11) and I used freed port to
use PWM.

Video Link

https://youtu.be/x9psB07PPZg