

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



VI XỬ LÝ - VI ĐIỀU KHIỂN
(CO3009)

LAB 1: LED ANIMATION

Giảng viên hướng dẫn: TS Lê Trọng Nhân
Th.S Phan Văn Sỹ

Sinh viên: Trương Thiên Ân

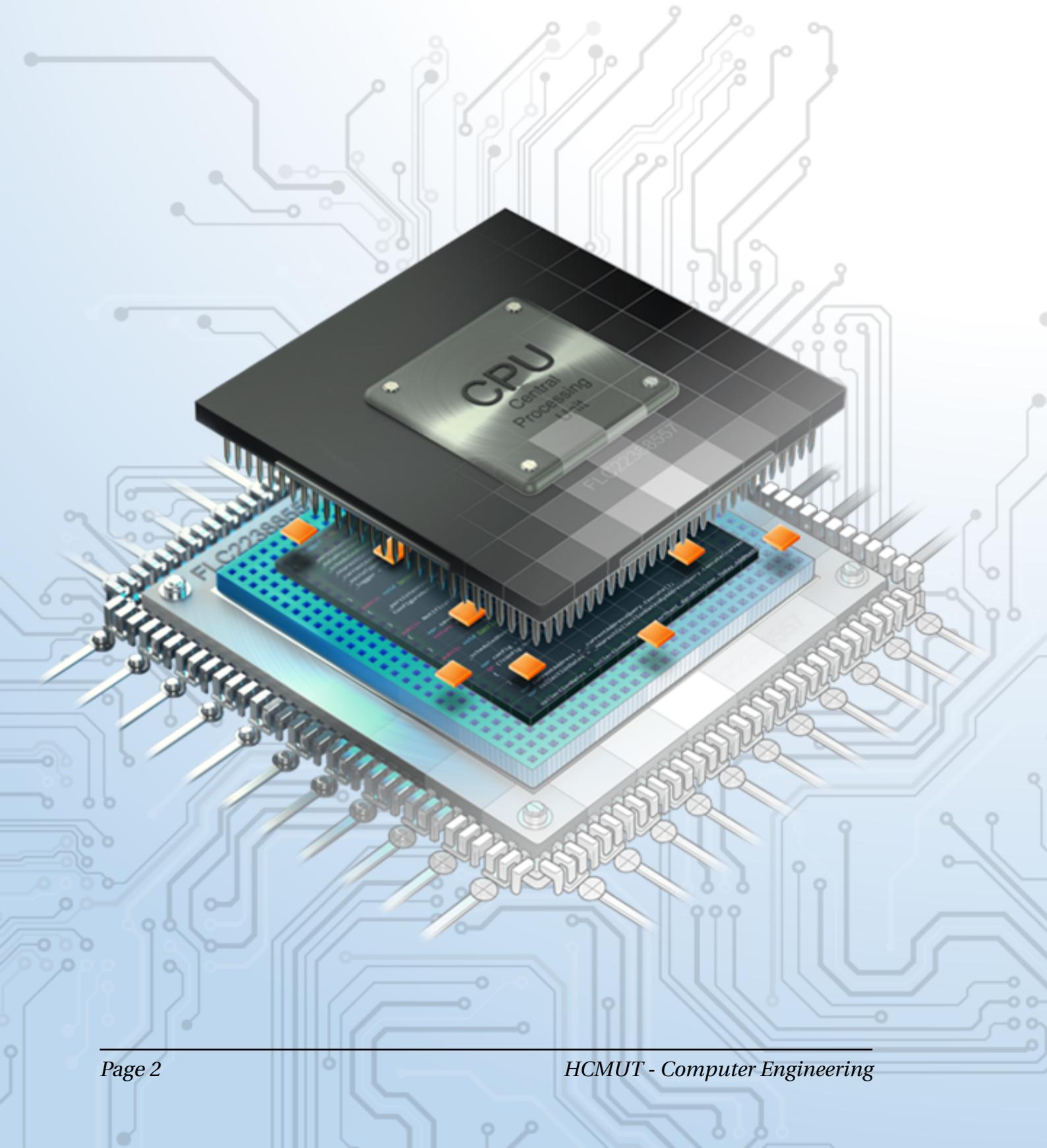
MSSV: 2310190

Lớp: TN01

Repo các bài lab: HCMUT_STM32_LAB

TP. HỒ CHÍ MINH, THÁNG 11/2024

Microcontroller

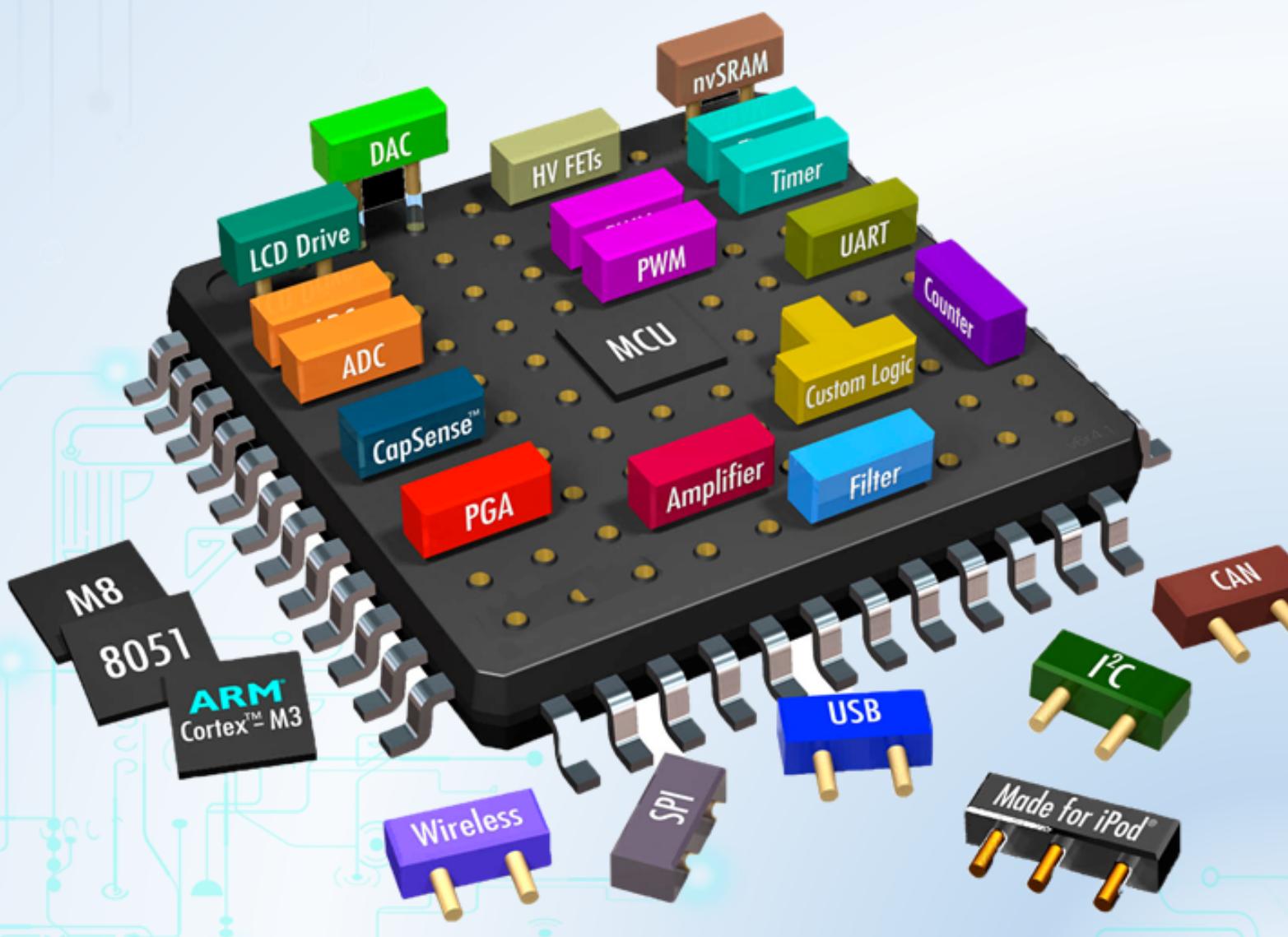


Mục lục

Chapter 1. LED Animations	5
1 Introduction	6
2 First project on STM32Cube	7
3 Simulation on Proteus	13
4 Exercise and Report	20
4.1 Exercise 1	20
4.2 Exercise 2	22
4.3 Exercise 3	23
4.4 Exercise 4	26
4.5 Exercise 5	28
4.6 Exercise 6	31
4.7 Exercise 7	33
4.8 Exercise 8	33
4.9 Exercise 9	33
4.10 Exercise 10	33
5 Source code	36

CHƯƠNG 1

LED Animations



1 Introduction

In this manual, the STM32CubeIDE is used as an editor to program the ARM microcontroller. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors.



Hình 1.1: STM32Cube IDE for STM32 Programming

The most interest of STM32CubeIDE is that after the selection of an empty STM32 MCU or MPU, or preconfigured microcontroller or microprocessor from the selection of a board, the initialization code generated automatically. At any time during the development, the user can return to the initialization and configuration of the peripherals or middleware and regenerate the initialization code with no impact on the user code. This feature can simplify the initialization process and speedup the development application running on STM32 micro-controller. The software can be downloaded from the link bellow:

https://ubc.sgp1.digitaloceanspaces.com/BKU_Softwares/STM32/stm32cubeide_1.7.0.zip

Moreover, for a hangout class, the program is firstly simulated on Proteus. Students are also supposed to download and install this software as well:

https://ubc.sgp1.digitaloceanspaces.com/BKU_Softwares/STM32/Proteus_8.10_SP0_Pro.exe

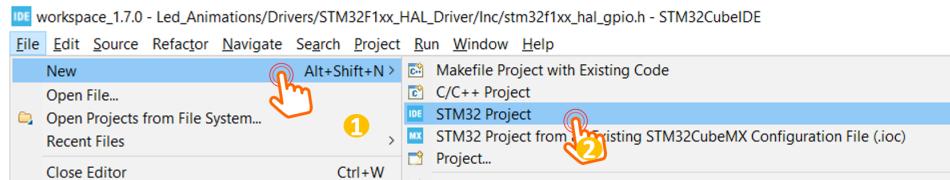
The rest of this manual consists of:

- Create a project on STM32Cube IDE
- Create a project on Proteus
- Simulate the project on Proteus

Finally, students are supposed to finish 10 different projects.

2 First project on STM32Cube

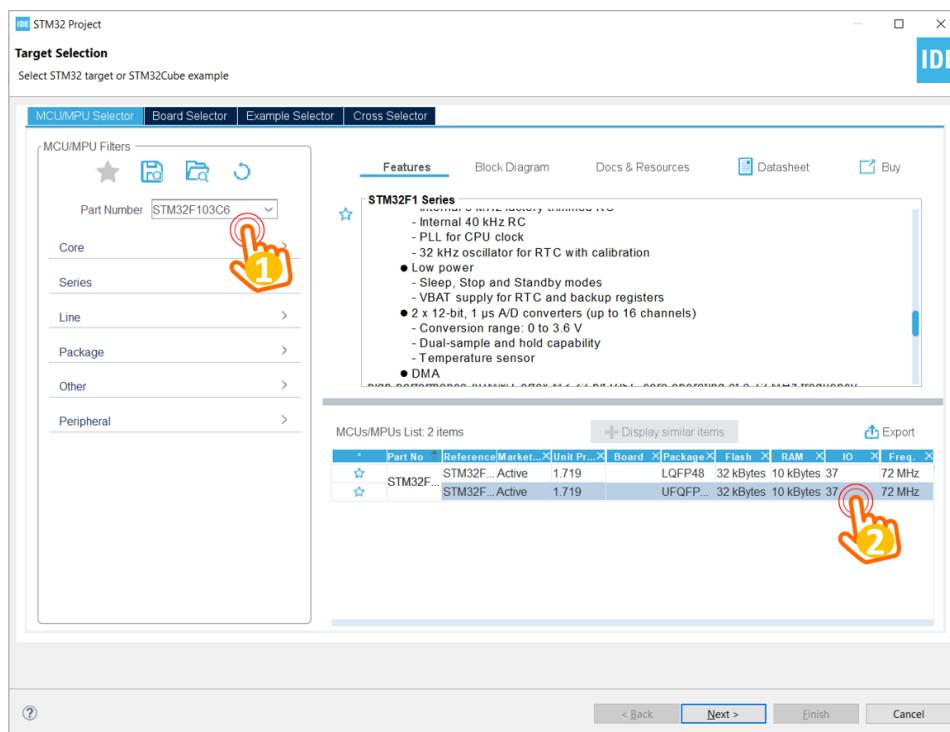
Step 1: Launch STM32CubeIDE, from the menu **File**, select **New**, then chose **STM32 Project**



Hình 1.2: Create a new project on STM32CubeIDE

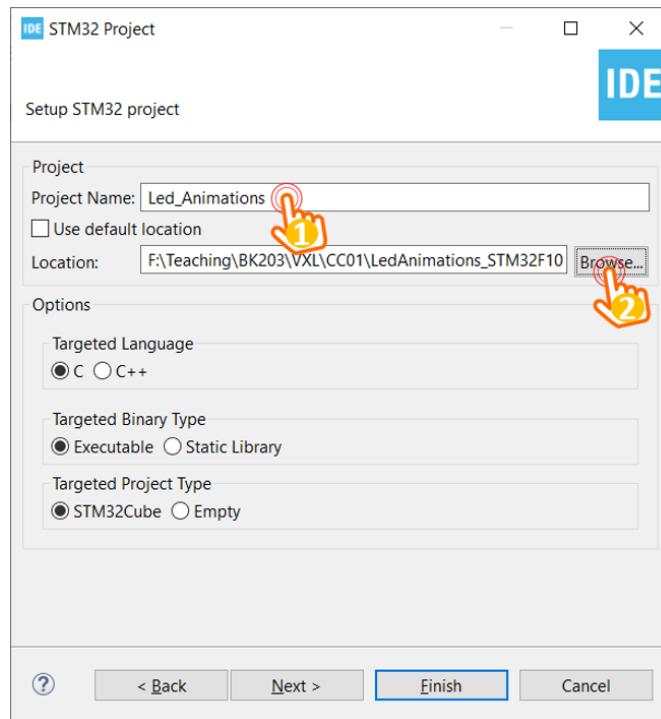
The IDE needs to download some packages, which normally takes time in this first time a project is created.

Step 2: Select the STM32F103C6 in the following dialog, then click on **Next**



Hình 1.3: Select the target device

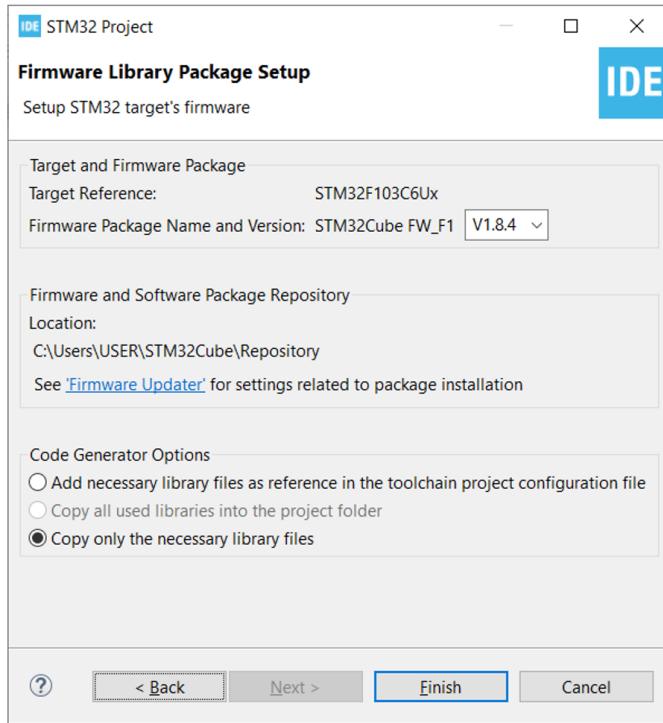
Step 3: Provide the **Name** and the **Location** for the project.



Hình 1.4: Select the target device

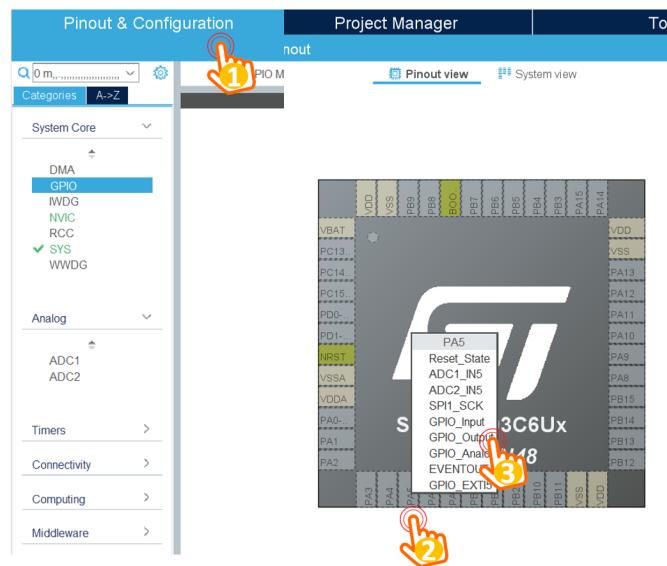
It is important to notice that the **Targeted Project Type** should be **STM32Cube**. In the case this option is disable, step 1 must be repeated. The location path should not contain special characters (e.g. the space). Finally, click on the **Next** button.

Step 4: On the last dialog, just keep the default firmware version and click on **Finish** button.



Hình 1.5: Keep default firmware version

Step 5: The project is created and the wizard for configuration is display. This utility from CubeIDE can simplify the configuration process for an ARM micro-controller like the STM32.

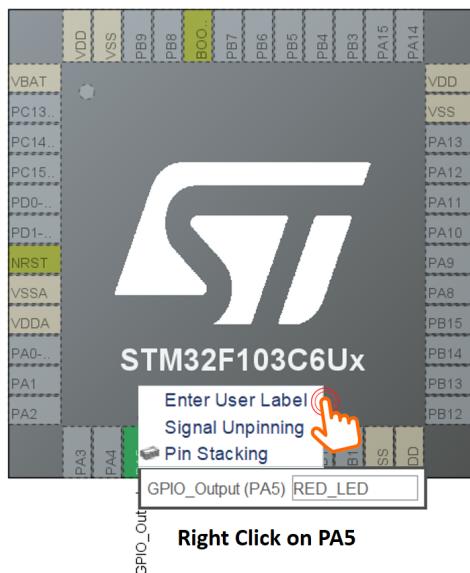


Hình 1.6: Set PA5 to GPIO Output mode

From the configuration windows, select **Pin configuration**, select the pin **PA5** and set to **GPIO Output** mode, since this pin is connected to an LED in the STM32 development kit.

Step 6: Right click on PA5 and select **Enter user label**, and provide the name for

this pin (e.g. **LED_RED**). This step helps programming afterward more memorable.



Hình 1.7: Provide a name for PA5

Finally, save the configuration process by pressing **Ctrl + S** and confirm this step by clicking on **OK** button. The code generation is started.

Step 7: Implement the first blinky project in the main function as follow:

```

1 int main(void)
2 {
3     /* USER CODE BEGIN 1 */
4
5     /* USER CODE END 1 */
6
7     /* MCU Configuration
8     * -----
9     */
10    /* Reset of all peripherals, Initializes the Flash
11       interface and the Systick. */
12    HAL_Init();
13
14    /* USER CODE BEGIN Init */
15
16    /* USER CODE END Init */
17
18    /* Configure the system clock */
19    SystemClock_Config();
20
21    /* USER CODE BEGIN SysInit */
22
23    /* USER CODE END SysInit */
24
25    /* Infinite loop */
26
27    /* USER CODE BEGIN 2 */
28
29    /* USER CODE END 2 */
30
31    /* Infinite loop */
32
33    /* USER CODE BEGIN 3 */
34
35    /* USER CODE END 3 */
36
37    /* Infinite loop */
38
39    /* USER CODE BEGIN 4 */
40
41    /* USER CODE END 4 */
42
43    /* Infinite loop */
44
45    /* USER CODE BEGIN 5 */
46
47    /* USER CODE END 5 */
48
49    /* Infinite loop */
50
51    /* USER CODE BEGIN 6 */
52
53    /* USER CODE END 6 */
54
55    /* Infinite loop */
56
57    /* USER CODE BEGIN 7 */
58
59    /* USER CODE END 7 */
60
61    /* Infinite loop */
62
63    /* USER CODE BEGIN 8 */
64
65    /* USER CODE END 8 */
66
67    /* Infinite loop */
68
69    /* USER CODE BEGIN 9 */
70
71    /* USER CODE END 9 */
72
73    /* Infinite loop */
74
75    /* USER CODE BEGIN 10 */
76
77    /* USER CODE END 10 */
78
79    /* Infinite loop */
80
81    /* USER CODE BEGIN 11 */
82
83    /* USER CODE END 11 */
84
85    /* Infinite loop */
86
87    /* USER CODE BEGIN 12 */
88
89    /* USER CODE END 12 */
90
91    /* Infinite loop */
92
93    /* USER CODE BEGIN 13 */
94
95    /* USER CODE END 13 */
96
97    /* Infinite loop */
98
99    /* USER CODE BEGIN 14 */
100
101   /* USER CODE END 14 */
102
103   /* Infinite loop */
104
105   /* USER CODE BEGIN 15 */
106
107   /* USER CODE END 15 */
108
109   /* Infinite loop */
110
111   /* USER CODE BEGIN 16 */
112
113   /* USER CODE END 16 */
114
115   /* Infinite loop */
116
117   /* USER CODE BEGIN 17 */
118
119   /* USER CODE END 17 */
120
121   /* Infinite loop */
122
123   /* USER CODE BEGIN 18 */
124
125   /* USER CODE END 18 */
126
127   /* Infinite loop */
128
129   /* USER CODE BEGIN 19 */
130
131   /* USER CODE END 19 */
132
133   /* Infinite loop */
134
135   /* USER CODE BEGIN 20 */
136
137   /* USER CODE END 20 */
138
139   /* Infinite loop */
140
141   /* USER CODE BEGIN 21 */
142
143   /* USER CODE END 21 */
144
145   /* Infinite loop */
146
147   /* USER CODE BEGIN 22 */
148
149   /* USER CODE END 22 */
150
151   /* Infinite loop */
152
153   /* USER CODE BEGIN 23 */
154
155   /* USER CODE END 23 */
156
157   /* Infinite loop */
158
159   /* USER CODE BEGIN 24 */
160
161   /* USER CODE END 24 */
162
163   /* Infinite loop */
164
165   /* USER CODE BEGIN 25 */
166
167   /* USER CODE END 25 */
168
169   /* Infinite loop */
170
171   /* USER CODE BEGIN 26 */
172
173   /* USER CODE END 26 */
174
175   /* Infinite loop */
176
177   /* USER CODE BEGIN 27 */
178
179   /* USER CODE END 27 */
180
181   /* Infinite loop */
182
183   /* USER CODE BEGIN 28 */
184
185   /* USER CODE END 28 */
186
187   /* Infinite loop */
188
189   /* USER CODE BEGIN 29 */
190
191   /* USER CODE END 29 */
192
193   /* Infinite loop */
194
195   /* USER CODE BEGIN 30 */
196
197   /* USER CODE END 30 */
198
199   /* Infinite loop */
200
201   /* USER CODE BEGIN 31 */
202
203   /* USER CODE END 31 */
204
205   /* Infinite loop */
206
207   /* USER CODE BEGIN 32 */
208
209   /* USER CODE END 32 */
210
211   /* Infinite loop */
212
213   /* USER CODE BEGIN 33 */
214
215   /* USER CODE END 33 */
216
217   /* Infinite loop */
218
219   /* USER CODE BEGIN 34 */
220
221   /* USER CODE END 34 */
222
223   /* Infinite loop */
224
225   /* USER CODE BEGIN 35 */
226
227   /* USER CODE END 35 */
228
229   /* Infinite loop */
230
231   /* USER CODE BEGIN 36 */
232
233   /* USER CODE END 36 */
234
235   /* Infinite loop */
236
237   /* USER CODE BEGIN 37 */
238
239   /* USER CODE END 37 */
240
241   /* Infinite loop */
242
243   /* USER CODE BEGIN 38 */
244
245   /* USER CODE END 38 */
246
247   /* Infinite loop */
248
249   /* USER CODE BEGIN 39 */
250
251   /* USER CODE END 39 */
252
253   /* Infinite loop */
254
255   /* USER CODE BEGIN 40 */
256
257   /* USER CODE END 40 */
258
259   /* Infinite loop */
260
261   /* USER CODE BEGIN 41 */
262
263   /* USER CODE END 41 */
264
265   /* Infinite loop */
266
267   /* USER CODE BEGIN 42 */
268
269   /* USER CODE END 42 */
270
271   /* Infinite loop */
272
273   /* USER CODE BEGIN 43 */
274
275   /* USER CODE END 43 */
276
277   /* Infinite loop */
278
279   /* USER CODE BEGIN 44 */
280
281   /* USER CODE END 44 */
282
283   /* Infinite loop */
284
285   /* USER CODE BEGIN 45 */
286
287   /* USER CODE END 45 */
288
289   /* Infinite loop */
290
291   /* USER CODE BEGIN 46 */
292
293   /* USER CODE END 46 */
294
295   /* Infinite loop */
296
297   /* USER CODE BEGIN 47 */
298
299   /* USER CODE END 47 */
299
300   /* Infinite loop */
301
302   /* USER CODE BEGIN 48 */
303
304   /* USER CODE END 48 */
305
306   /* Infinite loop */
307
308   /* USER CODE BEGIN 49 */
309
310   /* USER CODE END 49 */
311
312   /* Infinite loop */
313
314   /* USER CODE BEGIN 50 */
315
316   /* USER CODE END 50 */
317
318   /* Infinite loop */
319
320   /* USER CODE BEGIN 51 */
321
322   /* USER CODE END 51 */
323
324   /* Infinite loop */
325
326   /* USER CODE BEGIN 52 */
327
328   /* USER CODE END 52 */
329
330   /* Infinite loop */
331
332   /* USER CODE BEGIN 53 */
333
334   /* USER CODE END 53 */
335
336   /* Infinite loop */
337
338   /* USER CODE BEGIN 54 */
339
340   /* USER CODE END 54 */
341
342   /* Infinite loop */
343
344   /* USER CODE BEGIN 55 */
345
346   /* USER CODE END 55 */
347
348   /* Infinite loop */
349
350   /* USER CODE BEGIN 56 */
351
352   /* USER CODE END 56 */
353
354   /* Infinite loop */
355
356   /* USER CODE BEGIN 57 */
357
358   /* USER CODE END 57 */
359
360   /* Infinite loop */
361
362   /* USER CODE BEGIN 58 */
363
364   /* USER CODE END 58 */
365
366   /* Infinite loop */
367
368   /* USER CODE BEGIN 59 */
369
370   /* USER CODE END 59 */
371
372   /* Infinite loop */
373
374   /* USER CODE BEGIN 60 */
375
376   /* USER CODE END 60 */
377
378   /* Infinite loop */
379
380   /* USER CODE BEGIN 61 */
381
382   /* USER CODE END 61 */
383
384   /* Infinite loop */
385
386   /* USER CODE BEGIN 62 */
387
388   /* USER CODE END 62 */
389
390   /* Infinite loop */
391
392   /* USER CODE BEGIN 63 */
393
394   /* USER CODE END 63 */
395
396   /* Infinite loop */
397
398   /* USER CODE BEGIN 64 */
399
400   /* USER CODE END 64 */
401
402   /* Infinite loop */
403
404   /* USER CODE BEGIN 65 */
405
406   /* USER CODE END 65 */
407
408   /* Infinite loop */
409
410   /* USER CODE BEGIN 66 */
411
412   /* USER CODE END 66 */
413
414   /* Infinite loop */
415
416   /* USER CODE BEGIN 67 */
417
418   /* USER CODE END 67 */
419
420   /* Infinite loop */
421
422   /* USER CODE BEGIN 68 */
423
424   /* USER CODE END 68 */
425
426   /* Infinite loop */
427
428   /* USER CODE BEGIN 69 */
429
430   /* USER CODE END 69 */
431
432   /* Infinite loop */
433
434   /* USER CODE BEGIN 70 */
435
436   /* USER CODE END 70 */
437
438   /* Infinite loop */
439
440   /* USER CODE BEGIN 71 */
441
442   /* USER CODE END 71 */
443
444   /* Infinite loop */
445
446   /* USER CODE BEGIN 72 */
447
448   /* USER CODE END 72 */
449
450   /* Infinite loop */
451
452   /* USER CODE BEGIN 73 */
453
454   /* USER CODE END 73 */
455
456   /* Infinite loop */
457
458   /* USER CODE BEGIN 74 */
459
460   /* USER CODE END 74 */
461
462   /* Infinite loop */
463
464   /* USER CODE BEGIN 75 */
465
466   /* USER CODE END 75 */
467
468   /* Infinite loop */
469
470   /* USER CODE BEGIN 76 */
471
472   /* USER CODE END 76 */
473
474   /* Infinite loop */
475
476   /* USER CODE BEGIN 77 */
477
478   /* USER CODE END 77 */
479
480   /* Infinite loop */
481
482   /* USER CODE BEGIN 78 */
483
484   /* USER CODE END 78 */
485
486   /* Infinite loop */
487
488   /* USER CODE BEGIN 79 */
489
490   /* USER CODE END 79 */
491
492   /* Infinite loop */
493
494   /* USER CODE BEGIN 80 */
495
496   /* USER CODE END 80 */
497
498   /* Infinite loop */
499
500   /* USER CODE BEGIN 81 */
501
502   /* USER CODE END 81 */
503
504   /* Infinite loop */
505
506   /* USER CODE BEGIN 82 */
507
508   /* USER CODE END 82 */
509
510   /* Infinite loop */
511
512   /* USER CODE BEGIN 83 */
513
514   /* USER CODE END 83 */
515
516   /* Infinite loop */
517
518   /* USER CODE BEGIN 84 */
519
520   /* USER CODE END 84 */
521
522   /* Infinite loop */
523
524   /* USER CODE BEGIN 85 */
525
526   /* USER CODE END 85 */
527
528   /* Infinite loop */
529
530   /* USER CODE BEGIN 86 */
531
532   /* USER CODE END 86 */
533
534   /* Infinite loop */
535
536   /* USER CODE BEGIN 87 */
537
538   /* USER CODE END 87 */
539
540   /* Infinite loop */
541
542   /* USER CODE BEGIN 88 */
543
544   /* USER CODE END 88 */
545
546   /* Infinite loop */
547
548   /* USER CODE BEGIN 89 */
549
550   /* USER CODE END 89 */
551
552   /* Infinite loop */
553
554   /* USER CODE BEGIN 90 */
555
556   /* USER CODE END 90 */
557
558   /* Infinite loop */
559
560   /* USER CODE BEGIN 91 */
561
562   /* USER CODE END 91 */
563
564   /* Infinite loop */
565
566   /* USER CODE BEGIN 92 */
567
568   /* USER CODE END 92 */
569
570   /* Infinite loop */
571
572   /* USER CODE BEGIN 93 */
573
574   /* USER CODE END 93 */
575
576   /* Infinite loop */
577
578   /* USER CODE BEGIN 94 */
579
580   /* USER CODE END 94 */
581
582   /* Infinite loop */
583
584   /* USER CODE BEGIN 95 */
585
586   /* USER CODE END 95 */
587
588   /* Infinite loop */
589
590   /* USER CODE BEGIN 96 */
591
592   /* USER CODE END 96 */
593
594   /* Infinite loop */
595
596   /* USER CODE BEGIN 97 */
597
598   /* USER CODE END 97 */
599
599
600   /* Infinite loop */
601
602   /* USER CODE BEGIN 603 */
603
604   /* USER CODE END 603 */
605
606   /* Infinite loop */
607
608   /* USER CODE BEGIN 609 */
609
610   /* USER CODE END 609 */
611
612   /* Infinite loop */
613
614   /* USER CODE BEGIN 616 */
616
617   /* USER CODE END 616 */
618
619   /* Infinite loop */
620
621   /* USER CODE BEGIN 623 */
623
624   /* USER CODE END 623 */
625
626   /* Infinite loop */
627
628   /* USER CODE BEGIN 635 */
635
636   /* USER CODE END 635 */
637
638   /* Infinite loop */
639
640   /* USER CODE BEGIN 646 */
646
647   /* USER CODE END 646 */
648
649   /* Infinite loop */
650
651   /* USER CODE BEGIN 661 */
661
662   /* USER CODE END 661 */
663
664   /* Infinite loop */
665
666   /* USER CODE BEGIN 678 */
678
679   /* USER CODE END 678 */
680
681   /* Infinite loop */
682
683   /* USER CODE BEGIN 695 */
695
696   /* USER CODE END 695 */
697
698   /* Infinite loop */
699
700   /* USER CODE BEGIN 707 */
707
708   /* USER CODE END 707 */
709
710   /* Infinite loop */
711
712   /* USER CODE BEGIN 724 */
724
725   /* USER CODE END 724 */
726
727   /* Infinite loop */
728
729   /* USER CODE BEGIN 741 */
741
742   /* USER CODE END 741 */
743
744   /* Infinite loop */
745
746   /* USER CODE BEGIN 758 */
758
759   /* USER CODE END 758 */
760
761   /* Infinite loop */
762
763   /* USER CODE BEGIN 775 */
775
776   /* USER CODE END 775 */
777
778   /* Infinite loop */
779
780   /* USER CODE BEGIN 792 */
792
793   /* USER CODE END 792 */
794
795   /* Infinite loop */
796
797   /* USER CODE BEGIN 813 */
813
814   /* USER CODE END 813 */
815
816   /* Infinite loop */
817
818   /* USER CODE BEGIN 830 */
830
831   /* USER CODE END 830 */
832
833   /* Infinite loop */
834
835   /* USER CODE BEGIN 847 */
847
848   /* USER CODE END 847 */
849
850   /* Infinite loop */
851
852   /* USER CODE BEGIN 864 */
864
865   /* USER CODE END 864 */
866
867   /* Infinite loop */
868
869   /* USER CODE BEGIN 881 */
881
882   /* USER CODE END 882 */
883
884   /* Infinite loop */
885
886   /* USER CODE BEGIN 898 */
898
899   /* USER CODE END 899 */
900
901   /* Infinite loop */
902
903   /* USER CODE BEGIN 915 */
915
916   /* USER CODE END 916 */
917
918   /* Infinite loop */
919
920   /* USER CODE BEGIN 927 */
927
928   /* USER CODE END 927 */
929
930   /* Infinite loop */
931
932   /* USER CODE BEGIN 944 */
944
945   /* USER CODE END 945 */
946
947   /* Infinite loop */
948
949   /* USER CODE BEGIN 961 */
961
962   /* USER CODE END 962 */
963
964   /* Infinite loop */
965
966   /* USER CODE BEGIN 978 */
978
979   /* USER CODE END 979 */
980
981   /* Infinite loop */
982
983   /* USER CODE BEGIN 995 */
995
996   /* USER CODE END 996 */
997
998   /* Infinite loop */
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949

```

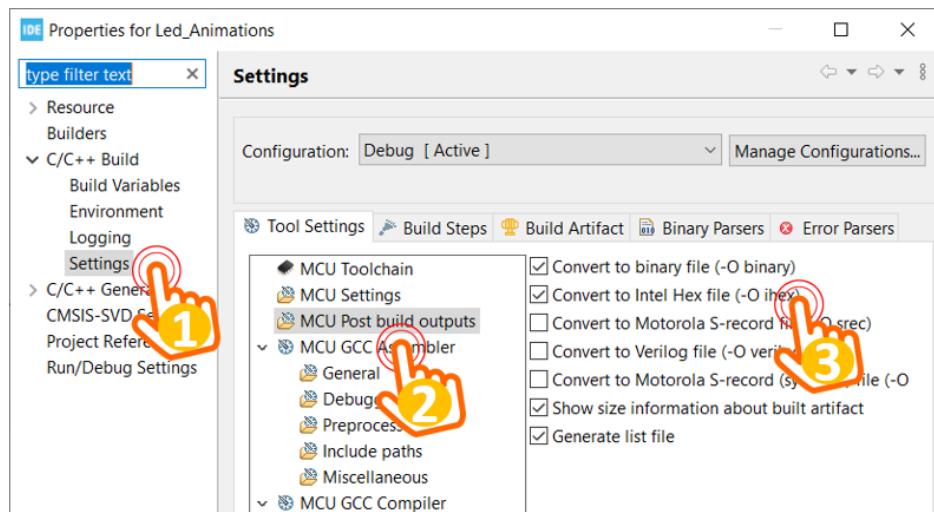
```

21  /* USER CODE END SysInit */
22
23  /* Initialize all configured peripherals */
24  MX_GPIO_Init();
25  /* USER CODE BEGIN 2 */
26
27  /* USER CODE END 2 */
28
29  /* Infinite loop */
30  /* USER CODE BEGIN WHILE */
31
32  while (1)
33  {
34      HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
35      HAL_Delay(1000);
36      /* USER CODE END WHILE */
37
38      /* USER CODE BEGIN 3 */
39  }
40  /* USER CODE END 3 */
41 }
```

Program 1.1: First blinky LED project

Actually, what is added to the main function is line number 34 and 35. Please put your code in a right place, otherwise it can be deleted when the code is generated (e.g. change the configuration of the project). When coding, frequently use the suggestions by pressing **Ctrl+Space**.

Step 7: Due to the simulation on Proteus, the hex file should be generated from STM32Cube IDE. From menu **Project**, select **Properties** to open the dialog bellow:



Hình 1.8: Config for hex file output

Navigate to **C/C++ Build**, select **Settings**, **MCU Post build outputs**, and check to the **Intel Hex file**.

Step 8: Build the project by clicking on menu **Project** and select **Build Project**. Please check on the output console of the IDE to be sure that the hex file is generated, as follow:

```
22:36:06 **** Incremental Build of configuration Debug for project Led_Animations ****
make -j8 all
arm-none-eabi-size   Led_Animations.elf
    text      data      bss      dec      hex filename
    4596        20     1572     6188    182c Led_Animations.elf
Finished building: default.size.stdout

22:36:06 Build Finished. 0 errors, 0 warnings. (took 272ms)
```

Hình 1.9: Compile the project and generate Hex file

The hex file is located under the **Debug** folder of your project, which is used for the simulation in Proteus afterward. In the case a development kit is connected to your PC, from menu **Run**, select **Run** to download the program to the hardware platform.

In the case there are multiple project in a work-space, double click on the project name to activate this project. Whenever a project is built, check the output files to make sure that you are working in a right project.

3 Simulation on Proteus

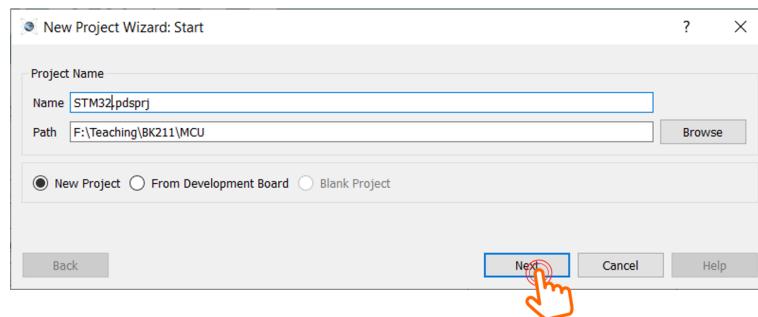
For an online training, a simulation on Proteus can be used. The details to create an STM32 project on Proteus are described below.

Step 1: Launch Proteus (**with administration access**) and from menu **File**, select **New Project**.



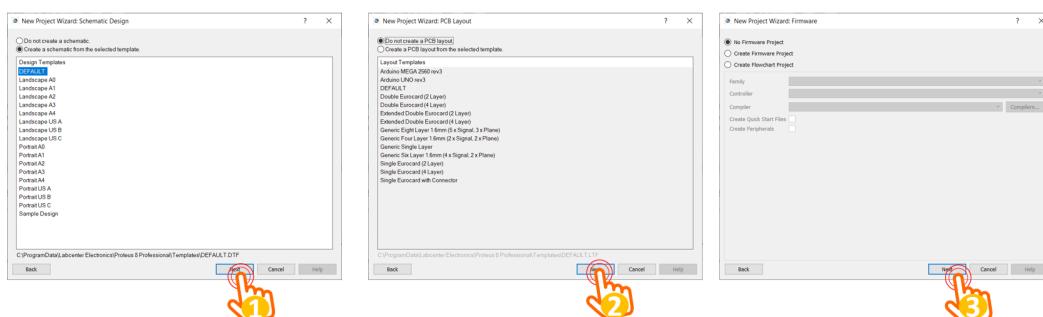
Hình 1.10: Create a new project on Proteus

Step 2: Provide the name and the location of the project, then click on **Next** button.



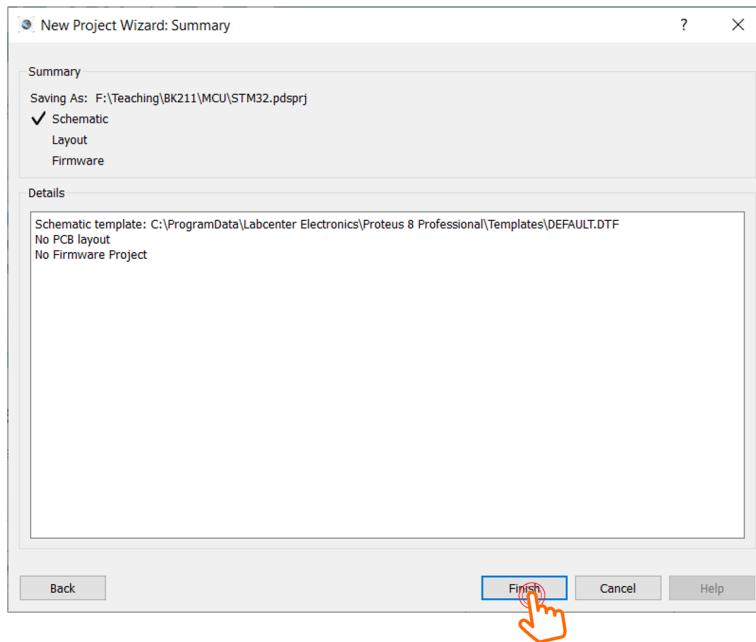
Hình 1.11: Provide project name and location

Step 3: For following dialog, just click on **Next** button as just a schematic is required for the lab.



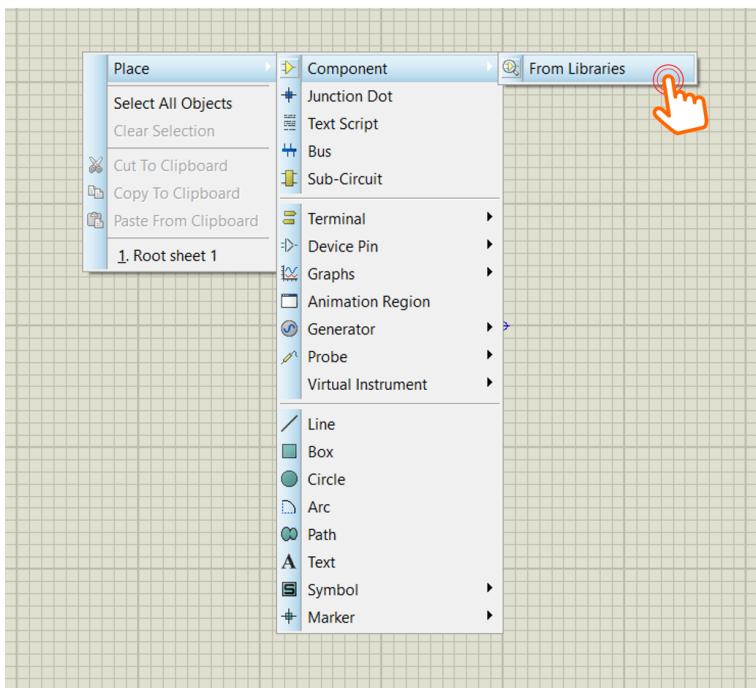
Hình 1.12: Keep the default options by clicking on Next

Step 4: Finally, click on **Finish** button to close the project wizard.



Hình 1.13: Finish the project wizard

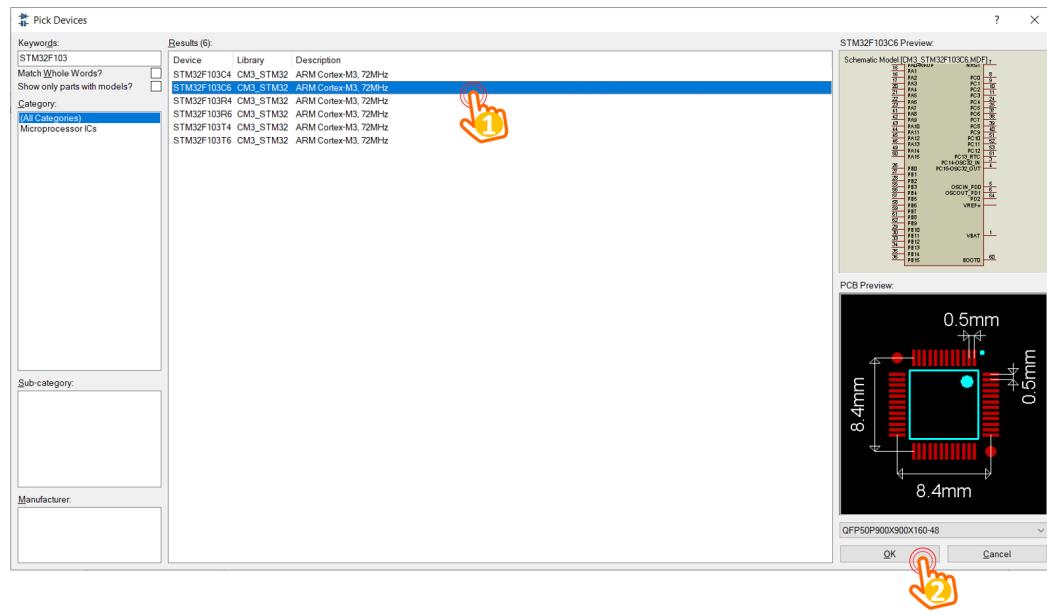
Step 5: On the main page of the project, right click to select **Place, Components, From Libraries**, as follows:



Hình 1.14: Select a component from the library

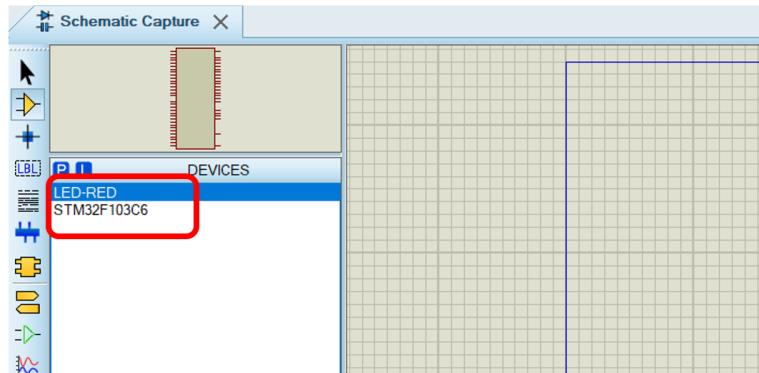
If there is an error with no library found, please restart the Proteus software with Run as administrator option.

Step 6: From the list of components in the library, select STM32F103C6, as follows:



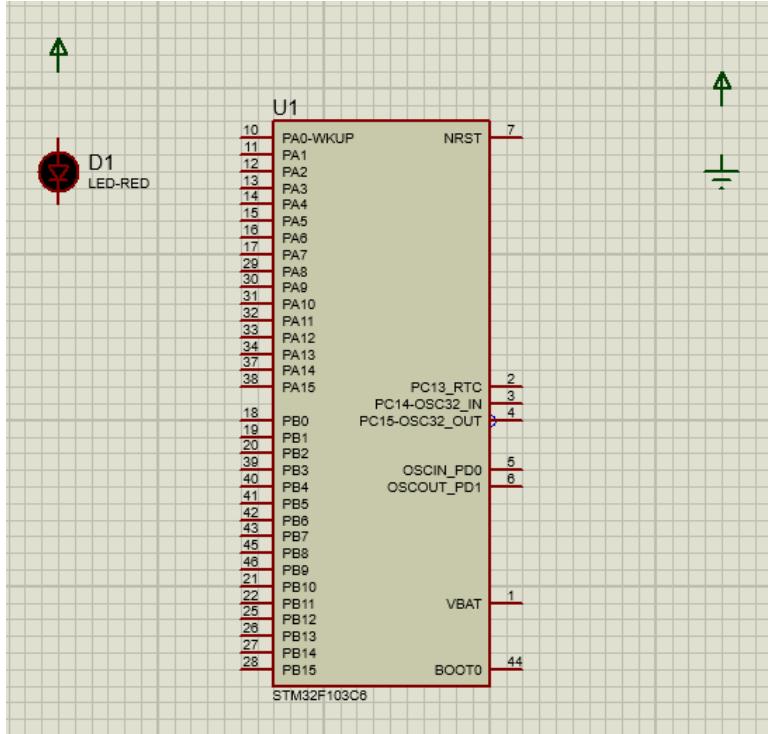
Hình 1.15: Select STM32F103C6

Repeat step 5 and 6 to select an LED, named **LED-RED** in Proteus. Finally, these components are appeared on the DEVICES windows, which is on left hand side as follows:



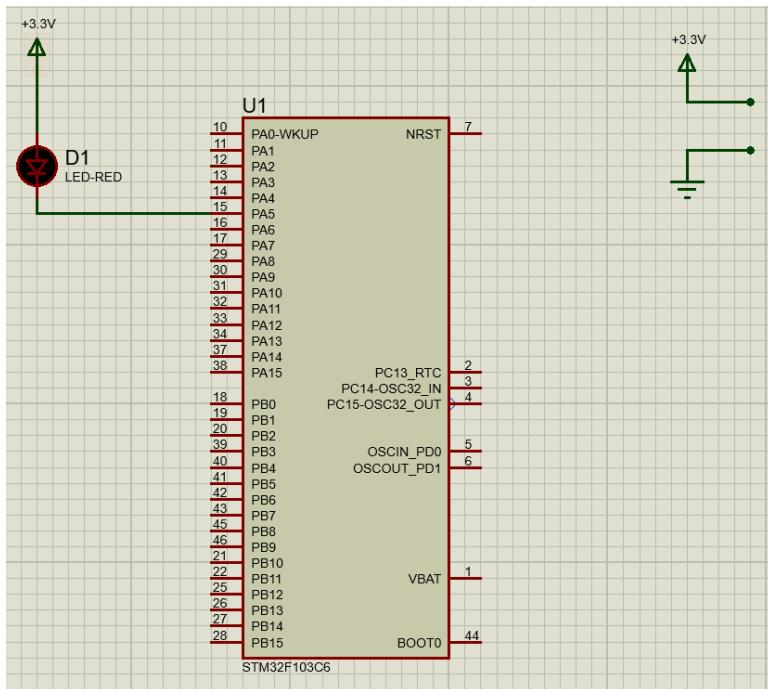
Hình 1.16: STM32 and an LED in the project

Step 7: Place the components to the project: right click on the main page, select on **Place, Component**, and select device added in Step 6. To add the Power and the Ground, right click on the main page, select on **Place, Terminal**. The result in this step is expected as follows:



Hình 1.17: Place components to the project

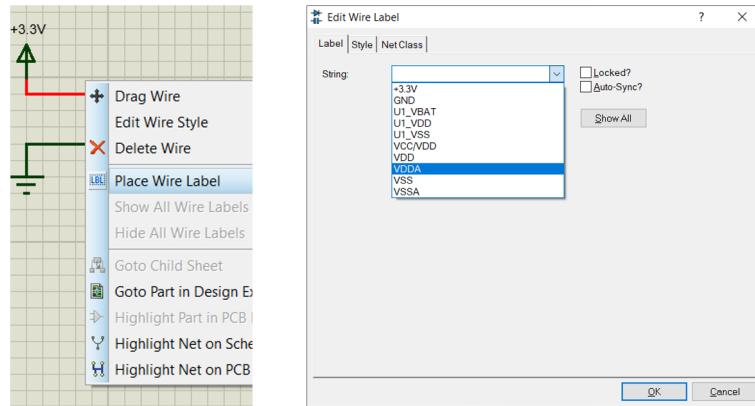
Step 8: Start wiring the circuit. The negative pin of the LED is connected to PA5 while its positive pin is connected to the power supply. For the power and the ground on the right, just make a short wire, which will be labeled in the next step.



Hình 1.18: Connect components and set the power to 3.3V

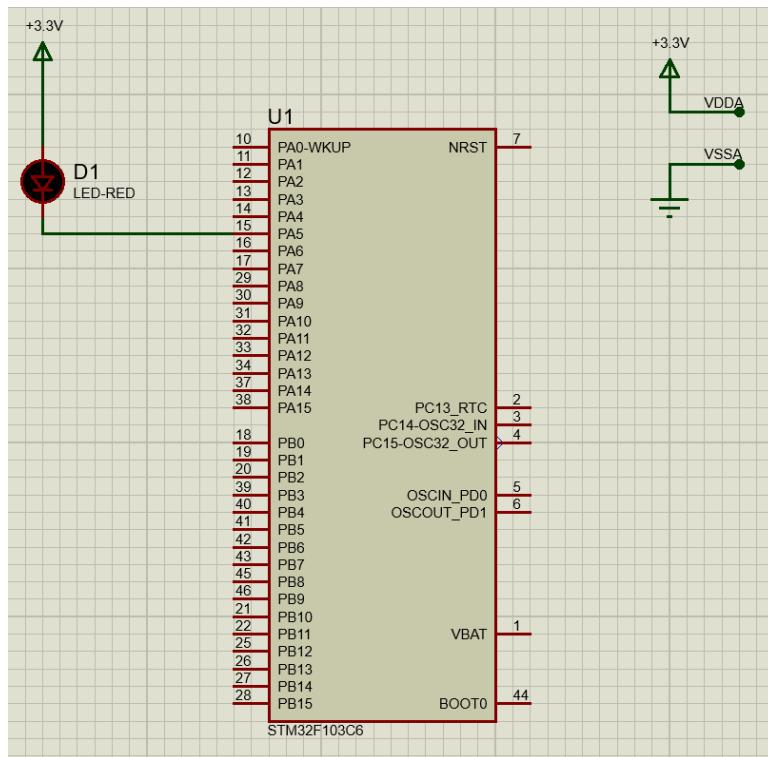
In this step, also double click on the power supply in order to provide the String property to **+3.3V**.

Step 8: Right click on the wire of the power supply and the ground, and select Place wire Label



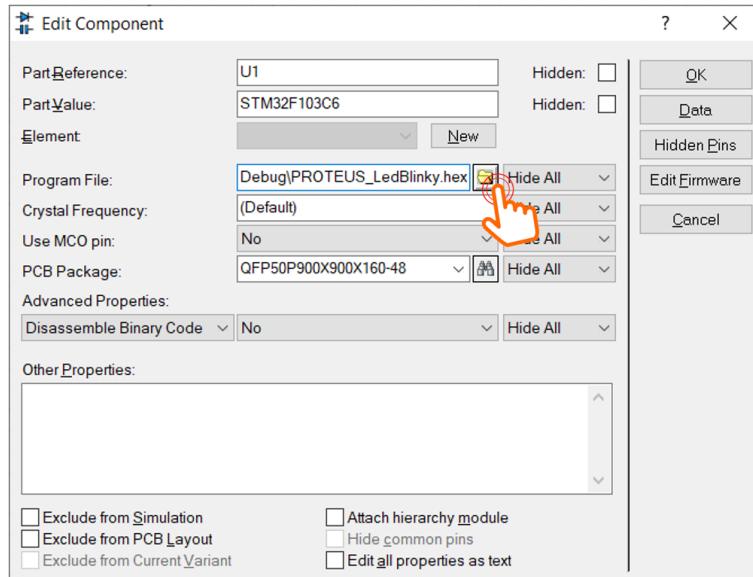
Hình 1.19: Place label for Power and Ground

This step is required as VDDA and VSSA of the STM32 must be connected to provide the reference voltage. Therefore, VDDA is connected to 3.3V, while the VSSA is connected to the Ground. Finally, the image of our schematic is shown bellow:



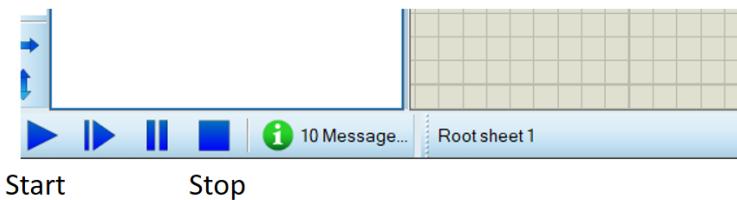
Hình 1.20: Finalize the schematic

Step 9: Double click on the STM32, and set the **Program File** to the Hex file, which is generated from Cube IDE, as following:



Hình 1.21: Set the program of the STM32 to the hex file from Cube IDE

From now, the simulation is ready to start by clicking on the menu **Debug**, and select on **Run simulation**. To stop the simulation, click on **Debug** and select **Stop VMS Debugging**. Moreover, there are some quick access bottom on the left corner of the Proteus to start or stop the simulation, as shown following:



Hình 1.22: Quick access buttons to start and stop the simulation

If everything is success, students can see the LED is blinking every second. Please stop the simulation before updating the project, either in Proteus or STM32Cube IDE. However, the step 9 (set the program file for STM32 in Proteus) is required to do once. Beside the toggle instruction, student can set or reset a pin as following:

```

1 while (1){
2     HAL_GPIO_WritePin(LED_RED_GPIO_Port , LED_RED_Pin ,
3         GPIO_PIN_SET);
4     HAL_Delay(1000);
5     HAL_GPIO_WritePin(LED_RED_GPIO_Port , LED_RED_Pin ,
6         GPIO_PIN_RESET);
7     HAL_Delay(1000);

```

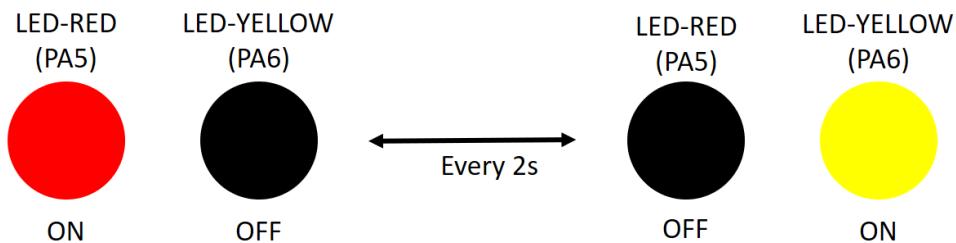
Program 1.2: An example for LED blinky

4 Exercise and Report

4.1 Exercise 1

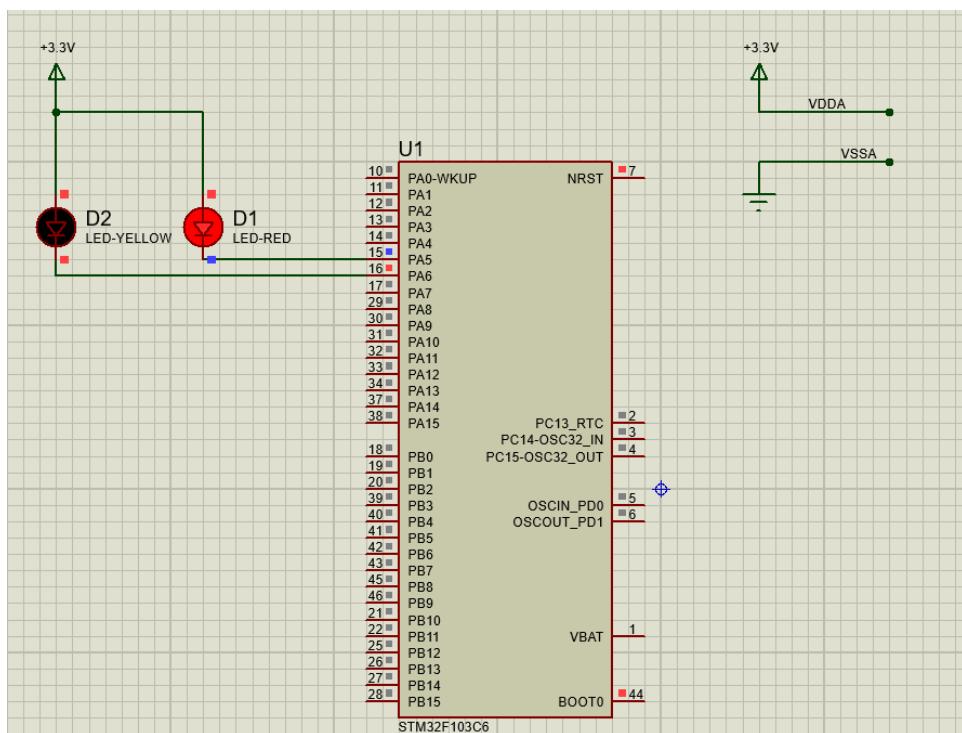
From the simulation on Proteus, one more LED is connected to pin **PA6** of the STM32 (negative pin of the LED is connected to PA6). The component suggested in this exercise is **LED-YELLOW**, which can be found from the device list.

In this exercise, the status of two LEDs are switched every 2 seconds, as demonstrated in the figure below.



Hình 1.23: State transitions for 2 LEDs

Report 1: Depict the schematic from Proteus simulation in this report. The caption of the figure is a downloadable link to the Proteus project file (e.g. a github link).



Hình 1.24: LAB1_exercise1

Report 2: Present the source code in the infinite loop while of your project. If a

user-defined functions is used, it is required to present in this part. A brief description can be added for this function (e.g. using comments).

```
1 void red_led (uint8_t n){  
2     HAL_GPIO_WritePin(RED_LED_GPIO_Port , RED_LED_Pin , n);  
3 }  
4 void yellow_led (uint8_t n){  
5     HAL_GPIO_WritePin(YELLOW_LED_GPIO_Port , YELLOW_LED_Pin , n  
6 );  
7 }
```

Program 1.3: Function to turn on LED

```
1 while (1)  
2 {  
3     if (counter > 0){  
4         counter--;  
5     }  
6     HAL_Delay(10);  
7     switch (state){  
8         case INIT:  
9             red_led(OFF);  
10            yellow_led(OFF);  
11            counter = 200;  
12            state = RED;  
13            red_led(ON);  
14            break;  
15         case RED:  
16             if (counter <= 0){  
17                 red_led (OFF);  
18                 counter = 200;  
19                 state = YELLOW;  
20                 yellow_led(ON);  
21             }  
22             break;  
23         case YELLOW:  
24             if (counter <= 0){  
25                 yellow_led(OFF);  
26                 counter = 200;  
27                 state = RED;  
28                 red_led(ON);  
29             }  
30             break;  
31         default:  
32             break;  
33     }  
34     /* USER CODE END WHILE */  
35  
36     /* USER CODE BEGIN 3 */  
37 }
```

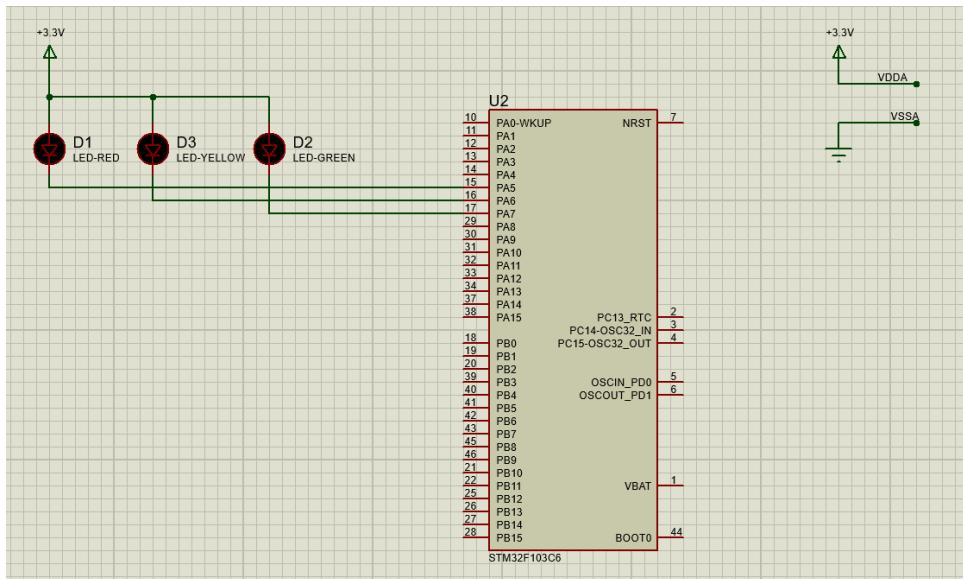
Program 1.4: Code for exercise 1

4.2 Exercise 2

Extend the first exercise to simulate the behavior of a traffic light. A third LED, named **LED-GREEN** is added to the system, which is connected to **PA7**. A cycle in this traffic light is 5 seconds for the RED, 2 seconds for the YELLOW and 3 seconds for the GREEN. The LED-GREEN is also controlled by its negative pin.

Similarly, the report in this exercise includes the schematic of your circuit and a your source code in the while loop.

Report 1: Present the schematic.



Hình 1.25: LAB2_exercise2

Report 2: Present the source code in while.

```
1 void green_led (uint8_t n){  
2     HAL_GPIO_WritePin(LED_GREEN_GPIO_Port , LED_GREEN_Pin , n);  
3 }  
4 void red_led (uint8_t n){  
5     //same as exercise 1  
6 }  
7 void yellow_led (uint8_t n){  
8     //same as exercise 1  
9 }
```

Program 1.5: Function to control LED

```
1 while (1)  
2 {  
3     if (counter > 0){  
4         counter--;  
5     }  
6     HAL_Delay(10);
```

```

7   switch (state){
8     case INIT:
9       red_led(OFF);
10      yellow_led(OFF);
11      green_led(OFF);
12      counter = 500;
13      state = RED;
14      red_led(ON);
15      break;
16    case RED:
17
18      if (counter <= 0){
19        red_led (OFF);
20        counter = 300;
21        state = GREEN;
22        green_led(ON);
23      }
24      break;
25    case GREEN:
26      if (counter <= 0){
27        green_led (OFF);
28        counter = 200;
29        state = YELLOW;
30        yellow_led(ON);
31      }
32      break;
33    case YELLOW:
34      if (counter <= 0){
35        yellow_led(OFF);
36        counter = 500;
37        state = RED;
38        red_led(ON);
39      }
40      break;
41    default:
42      break;
43  }
44  /* USER CODE END WHILE */
45
46  /* USER CODE BEGIN 3 */
47}

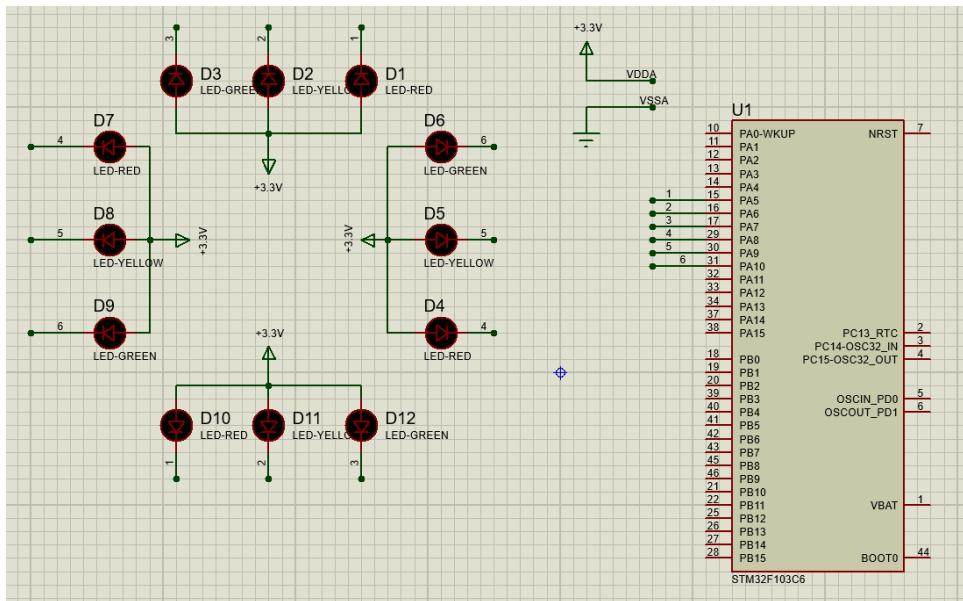
```

Program 1.6: Code for exercise 1

4.3 Exercise 3

Extend to the 4-way traffic light. Arrange 12 LEDs in a nice shape to simulate the behaviors of a traffic light. A reference design can be found in the figure bellow.

Report 1: Present the schematic.



Hình 1.26: LAB1_exercise3

Report 2: Present the source code.

```
1 void red_led (uint8_t state, uint8_t way){
2     if (way == 1){
3         HAL_GPIO_WritePin(LED_RED_1_GPIO_Port, LED_RED_1_Pin,
4                           state);
5     } else if (way == 2){
6         HAL_GPIO_WritePin(LED_RED_2_GPIO_Port, LED_RED_2_Pin,
7                           state);
8     } else{
9         return;
10    }
11 }
12
13 void yellow_led (uint8_t state, uint8_t way){
14     if (way == 1){
15         HAL_GPIO_WritePin(LED_YELLOW_1_GPIO_Port,
16                           LED_YELLOW_1_Pin, state);
17     } else if (way == 2){
18         HAL_GPIO_WritePin(LED_YELLOW_2_GPIO_Port,
19                           LED_YELLOW_2_Pin, state);
20     } else{
21         return;
22    }
23 }
24
25 void green_led (uint8_t state, uint8_t way){
26     if (way == 1){
27         HAL_GPIO_WritePin(LED_GREEN_1_GPIO_Port,
28                           LED_GREEN_1_Pin, state);
29     } else if (way == 2){
30         HAL_GPIO_WritePin(LED_GREEN_2_GPIO_Port,
31                           LED_GREEN_2_Pin, state);
32     } else{
33         return;
34    }
35 }
```

```

23 }else if (way == 2){
24     HAL_GPIO_WritePin(LED_GREEN_2_GPIO_Port,
25         LED_GREEN_2_Pin, state);
26 }else{
27     return;
28 }

```

Program 1.7: Function to control LED on both way

```

1   while (1)
2   {
3       HAL_Delay(10);
4       if (counter > 0){
5           counter --;
6       }
7
8       switch (state){
9           case INIT:
10          red_led(OFF, 1);
11          yellow_led(OFF, 1);
12          green_led(OFF, 1);
13          red_led(OFF, 2);
14          yellow_led(OFF, 2);
15          green_led(OFF, 2);
16          counter = 500;
17          state = STOP_1;
18          red_led(ON, 1);
19          green_led(ON, 2);
20          break;
21           case STOP_1:
22           if (counter <= 200){
23               green_led(OFF, 2);
24               yellow_led(ON, 2);
25           }
26           if (counter <= 0){
27               red_led (OFF, 1);
28               counter = 300;
29               state = GO_1;
30               green_led(ON, 1);
31               red_led (ON, 2);
32               yellow_led(OFF, 2);
33           }
34           break;
35           case GO_1:
36           if (counter <= 0){
37               green_led (OFF, 1);
38               counter = 200;
39               state = SLOW_1;
40               yellow_led(ON, 1);

```

```

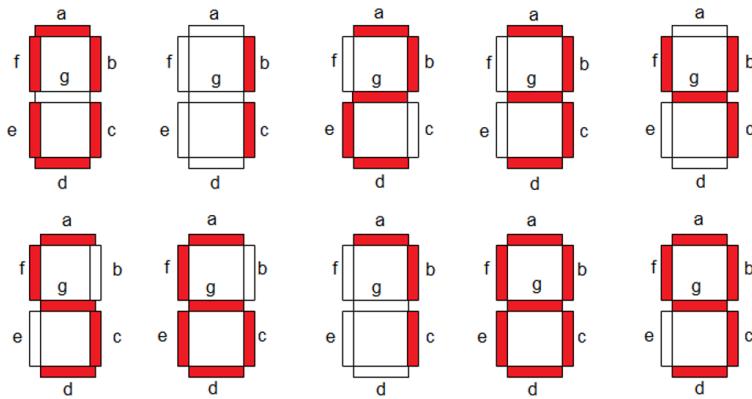
41     }
42     break;
43 case SLOW_1:
44     if (counter <= 0){
45         yellow_led(OFF, 1);
46         red_led(OFF, 2);
47         green_led(ON, 2);
48         counter = 500;
49         state = STOP_1;
50         red_led(ON, 1);
51     }
52     break;
53 default:
54     break;
55 }
56
57 /* USER CODE END WHILE */
58
59 /* USER CODE BEGIN 3 */
60 }
```

Program 1.8: Code for exercise 3

4.4 Exercise 4

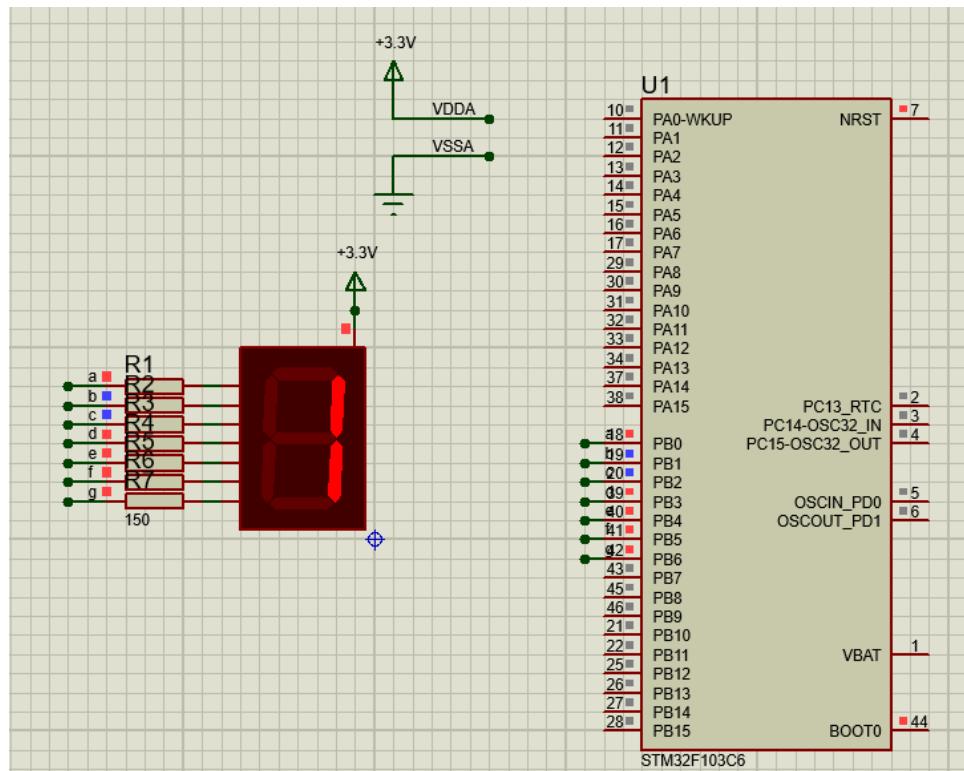
Add **only one 7 led segment** to the schematic in Exercise 3. This component can be found in Proteus by the keyword **7SEG-COM-ANODE**. For this device, the common pin should be connected to the power supply and other pins are supposed to be connected to PB0 to PB6. Therefore, to turn-on a segment in this 7SEG, the STM32 pin should be in logic 0 (0V).

Implement a function named **display7SEG(int num)**. The input for this function is from 0 to 9 and the outputs are listed as following:



Hình 1.27: Display a number on 7 segment LED

Report 1: Present the schematic.



Hình 1.28: LAB1_exercise4

Report 2: Present the source code.

```

1 void display7SEG(uint8_t num){
2     HAL_GPIO_WritePin(a_GPIO_Port, a_Pin, !((~
3         led7seg_num_display[num]) & (1 << 0)));
4     HAL_GPIO_WritePin(b_GPIO_Port, b_Pin, !((~
5         led7seg_num_display[num]) & (1 << 1)));
6     HAL_GPIO_WritePin(c_GPIO_Port, c_Pin, !((~
7         led7seg_num_display[num]) & (1 << 2)));
8     HAL_GPIO_WritePin(d_GPIO_Port, d_Pin, !((~
9         led7seg_num_display[num]) & (1 << 3)));
10    HAL_GPIO_WritePin(e_GPIO_Port, e_Pin, !((~
11        led7seg_num_display[num]) & (1 << 4)));
12    HAL_GPIO_WritePin(f_GPIO_Port, f_Pin, !((~
13        led7seg_num_display[num]) & (1 << 5)));
14    HAL_GPIO_WritePin(g_GPIO_Port, g_Pin, !((~
15        led7seg_num_display[num]) & (1 << 6)));
16 }
```

Program 1.9: Function display7SEG

```

1 while (1)
2 {
3     /* USER CODE END WHILE */
4     if (counter >= 10) {
```

```

5     counter = 0;
6 }
7 display7SEG(counter++);
8 HAL_Delay(1000);
9 /* USER CODE BEGIN 3 */
10}

```

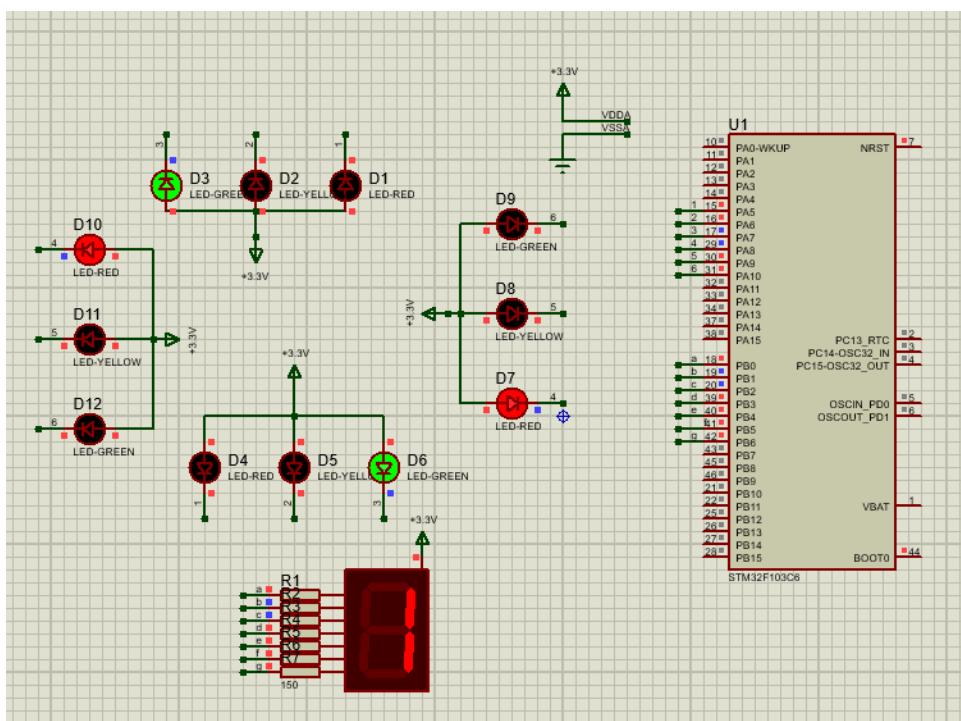
Program 1.10: Code for exercise 4

4.5 Exercise 5

Integrate the 7SEG-LED to the 4 way traffic light. In this case, the 7SEG-LED is used to display countdown value.

In this exercise, only source code is required to present. The function display7SEG in the previous exercise can be re-used.

Report 1: Present the schematic.



Hình 1.29: LAB1_exercise5

Report 2: Present the source code.

```

1 #define ON 0
2 #define OFF 1
3 #define INIT 0
4 //only define state of 1 way -> STOP_1 = GO_2 + SLOW_2 and
   GO_1 + SLOW_1 = STOP_2
5 #define STOP_1 1

```

```

6 #define SLOW_1 2
7 #define GO_1 3
8 uint8_t state = INIT;
9 uint16_t counter = 0;
10 uint8_t led7seg_num_display[] = {
11     /*0*/ 0x40,/*0b1000000*/
12     /*1*/ 0x79,/*0b11111001*/
13     /*2*/ 0x24,/*0b0100100*/
14     /*3*/ 0x30,/*0b0110000*/
15     /*4*/ 0x19,/*0b0011001*/
16     /*5*/ 0x12,/*0b00010010*/
17     /*6*/ 0x02,/*0b00000010*/
18     /*7*/ 0x78,/*0b1111000*/
19     /*8*/ 0x00,/*0b00000000*/
20     /*9*/ 0x10/*0b0010000*/
21 };
22 uint8_t prev_num = 255;

```

Program 1.11: Private define

```

1 void red_led (uint8_t state, uint8_t way){
2     //same as exercise 3
3 }
4 void yellow_led (uint8_t state, uint8_t way){
5     //same as exercise 3
6 }
7 void green_led (uint8_t state, uint8_t way){
8     //same as exercise 3
9 }
10 void display7SEG(uint8_t num){
11     //same as exercise 4
12 }

```

Program 1.12: Function control led light on both way

```

1 while (1)
2 {
3     HAL_Delay(10);
4     if (counter > 0){
5         counter--;
6     }
7
8     switch (state){
9     case INIT:
10        red_led(OFF, 1);
11        yellow_led(OFF, 1);
12        green_led(OFF, 1);
13        red_led(OFF, 2);
14        yellow_led(OFF, 2);
15        green_led(OFF, 2);
16        counter = 500;

```

```

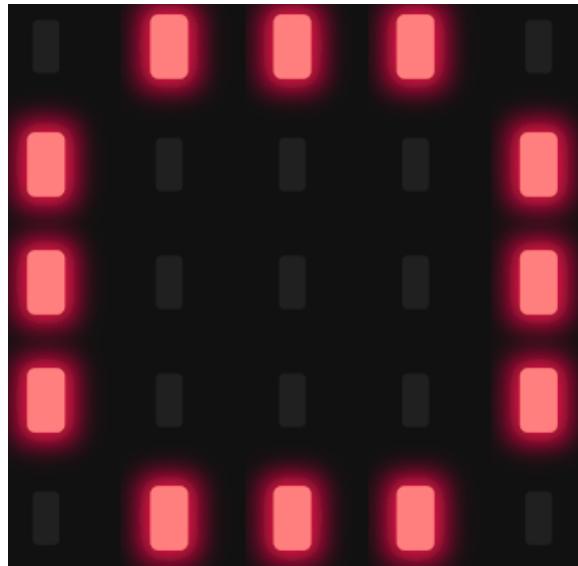
17     state = STOP_1;
18     red_led(ON, 1);
19     green_led(ON, 2);
20     break;
21 case STOP_1:
22     if (counter <= 200){
23         green_led(OFF, 2);
24         yellow_led(ON, 2);
25     }
26     if (counter <= 0){
27         red_led (OFF, 1);
28         counter = 300;
29         state = GO_1;
30         green_led(ON, 1);
31         red_led (ON, 2);
32         yellow_led(OFF, 2);
33     }
34     break;
35 case GO_1:
36     if (counter <= 0){
37         green_led (OFF, 1);
38         counter = 200;
39         state = SLOW_1;
40         yellow_led(ON, 1);
41     }
42     break;
43 case SLOW_1:
44     if (counter <= 0){
45         yellow_led(OFF, 1);
46         red_led(OFF, 2);
47         green_led(ON, 2);
48         counter = 500;
49         state = STOP_1;
50         red_led(ON, 1);
51     }
52     break;
53 default:
54     break;
55 }
56 uint8_t curr_num = (counter + 99)/ 100;
57 if (curr_num != prev_num){
58     prev_num = curr_num;
59     display7SEG(curr_num);
60 }
/* USER CODE END WHILE */

62 /* USER CODE BEGIN 3 */
63
64 }
```

Program 1.13: Code for exercise 5

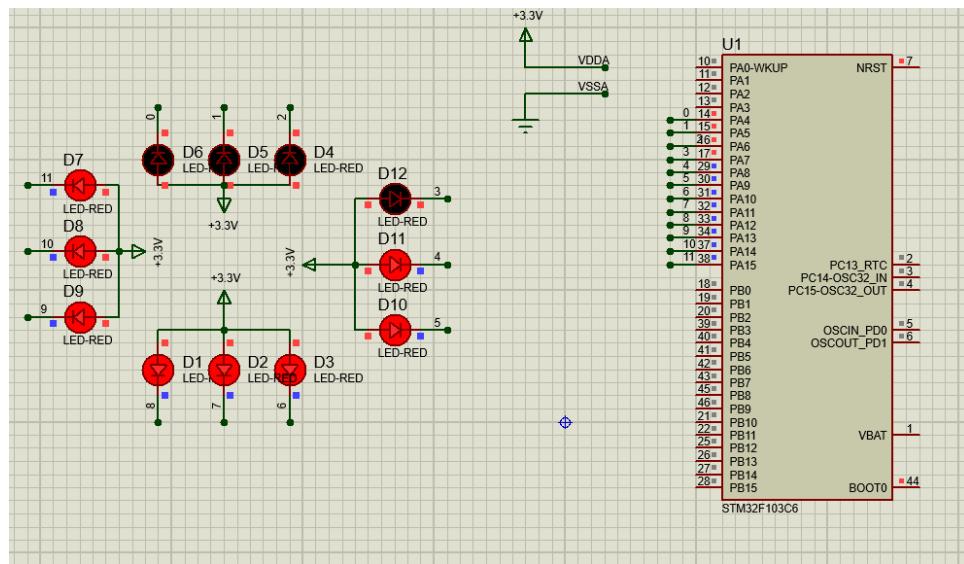
4.6 Exercise 6

In this exercise, a new Proteus schematic is designed to simulate an analog clock, with 12 different number. The connections for 12 LEDs are supposed from PA4 to PA15 of the STM32. The arrangement of 12 LEDs is depicted as follows.



Hình 1.30: 12 LEDs for an analog clock

Report 1: Present the schematic.



Hình 1.31: LAB1_exercise6

Report 2: Implement a simple program to test the connection of every single LED. This testing program should turn every LED in a sequence.

Function **stateUpdate()**: Used to control LEDs based on the *state* - 12 bits (0 - ON, 1 - OFF)

```

1 void stateUpdate(){
2     HAL_GPIO_WritePin(LED_0_GPIO_Port, LED_0_Pin, !(~state &
3         (1 << 0)));
4     HAL_GPIO_WritePin(LED_1_GPIO_Port, LED_1_Pin, !(~state &
5         (1 << 1)));
6     HAL_GPIO_WritePin(LED_2_GPIO_Port, LED_2_Pin, !(~state &
7         (1 << 2)));
8     HAL_GPIO_WritePin(LED_3_GPIO_Port, LED_3_Pin, !(~state &
9         (1 << 3)));
10    HAL_GPIO_WritePin(LED_4_GPIO_Port, LED_4_Pin, !(~state &
11        (1 << 4)));
12    HAL_GPIO_WritePin(LED_5_GPIO_Port, LED_5_Pin, !(~state &
13        (1 << 5)));
14    HAL_GPIO_WritePin(LED_6_GPIO_Port, LED_6_Pin, !(~state &
15        (1 << 6)));
16    HAL_GPIO_WritePin(LED_7_GPIO_Port, LED_7_Pin, !(~state &
17        (1 << 7)));
18    HAL_GPIO_WritePin(LED_8_GPIO_Port, LED_8_Pin, !(~state &
19        (1 << 8)));
20    HAL_GPIO_WritePin(LED_9_GPIO_Port, LED_9_Pin, !(~state &
21        (1 << 9)));
22    HAL_GPIO_WritePin(LED_10_GPIO_Port, LED_10_Pin, !(~state
23        & (1 << 10)));
24    HAL_GPIO_WritePin(LED_11_GPIO_Port, LED_11_Pin, !(~state
25        & (1 << 11)));
26 }

```

Program 1.14: Function to control 12 led

```

1 while (1)
2 {
3     if (counter >= 12){
4         counter = 0;
5         flag ^= 1; //toggle flag
6     }
7     if (flag == 0){
8         state = state | (1 << counter);
9     }else if (flag == 1){
10         state = state & ~(1 << counter);
11     }
12     stateUpdate();
13     counter++;
14     HAL_Delay(SPEED);
15     /* USER CODE END WHILE */
16
17     /* USER CODE BEGIN 3 */
18 }

```

Program 1.15: Code for exercise 6

4.7 Exercise 7

Implement a function named **clearAllClock()** to turn off all 12 LEDs. Present the source code of this function.

```
1 void clearAllClock(){
2     state = 0xFF;
3     stateUpdate();
4 }
```

Program 1.16: Function Implementation

4.8 Exercise 8

Implement a function named **setNumberOnClock(int num)**. The input for this function is from **0 to 11** and an appropriate LED is turn on. Present the source code of this function.

```
1 void setNumberOnClock(int num){
2     state &= ~(1 << num);
3     stateUpdate();
4 }
```

Program 1.17: Function Implementation

4.9 Exercise 9

Implement a function named **clearNumberOnClock(int num)**. The input for this function is from **0 to 11** and an appropriate LED is turn off.

```
1 void clearNumberOnClock(int num){
2     state |= (1 << num);
3     stateUpdate();
4 }
```

Program 1.18: Function Implementation

4.10 Exercise 10

Integrate the whole system and use 12 LEDs to display a clock. At a given time, there are only 3 LEDs are turn on for hour, minute and second information.

Function **stateUpdate()**: Used to control LEDs based on the *state - 12 bits* (0 - ON, 1 - OFF).

```
1 void stateUpdate() {
2     HAL_GPIO_WritePin(LED_0_GPIO_Port, LED_0_Pin, !(~state &
3         (1 << 0)));
4     HAL_GPIO_WritePin(LED_1_GPIO_Port, LED_1_Pin, !(~state &
5         (1 << 1)));
```

```

4   HAL_GPIO_WritePin(LED_2_GPIO_Port, LED_2_Pin, !(~state &
5     (1 << 2)));
6   HAL_GPIO_WritePin(LED_3_GPIO_Port, LED_3_Pin, !(~state &
7     (1 << 3)));
8   HAL_GPIO_WritePin(LED_4_GPIO_Port, LED_4_Pin, !(~state &
9     (1 << 4)));
10  HAL_GPIO_WritePin(LED_5_GPIO_Port, LED_5_Pin, !(~state &
11    (1 << 5)));
12  HAL_GPIO_WritePin(LED_6_GPIO_Port, LED_6_Pin, !(~state &
13    (1 << 6)));
14  HAL_GPIO_WritePin(LED_7_GPIO_Port, LED_7_Pin, !(~state &
15    (1 << 7)));
16  HAL_GPIO_WritePin(LED_8_GPIO_Port, LED_8_Pin, !(~state &
17    (1 << 8)));
18  HAL_GPIO_WritePin(LED_9_GPIO_Port, LED_9_Pin, !(~state &
19    (1 << 9)));
20  HAL_GPIO_WritePin(LED_10_GPIO_Port, LED_10_Pin, !(~state &
21    (1 << 10)));
22  HAL_GPIO_WritePin(LED_11_GPIO_Port, LED_11_Pin, !(~state &
23    (1 << 11)));
24 }

```

Program 1.19: Function stateUpdate()

Function `clockUpdate()`: This function is used to check and update if there is a new state (for example `prev_state[2]` (hour) = 1 and `curr_state[2]` (hour) = 2 -> update `prev_state`, display the new state on clock, clear the old state)

```

1 void clockUpdate(){
2   curr_state[0] = sec/5;
3   curr_state[1] = min/5;
4   curr_state[2] = hour;
5   for (int i = 0; i < 3; i++){
6     setNumberOnClock(curr_state[i]);
7   }
8   for (int i = 0; i < 3; i++){
9     int flag = 1;
10    for (int j = 0; j < 3; j++){
11      if (curr_state[j] == prev_state[i]){
12        flag = 0;
13        break;
14      }
15    }
16    if (flag == 1){
17      clearNumberOnClock(prev_state[i]);
18    }
19    prev_state[i] = curr_state[i];
20  }
21 }

```

Program 1.20: Function clockUpdate()

Function **writeClock()**: This is the function to calculate min and hour. It also calls `clockUpdate()` to check and update on every tick (10ms)

```
1 void writeClock(){
2     clockUpdate();
3     sec++;
4     if (sec >= 60){
5         sec = 0;
6     }
7     if (sec == 0){
8         min++;
9         if (min >= 60){
10            min = 0;
11        }
12        if (min == 0){
13            hour++;
14            if (hour >= 12){
15                hour = 0;
16            }
17        }
18    }
19}
20}
```

Program 1.21: Function `writeClock()`

Function **setTimer(int h, int m, int s)**: This function is used to set the timer at any time you want.

```
1 void setTime(int h, int m, int s){
2     hour = h;
3     min = m;
4     sec = s;
5     clockUpdate();
6 }
```

Program 1.22: Function `setTimer(int h`

Source code in while loop:

```
1 /* USER CODE BEGIN 2 */
2 clearAllClock();
3 setTime(2, 45, 0); //set timer at 2:45
4 /* USER CODE END 2 */

5
6 /* Infinite loop */
7 /* USER CODE BEGIN WHILE */
8 while (1)
9 {
10     writeClock();
11     HAL_Delay(SPEED);
12     /* USER CODE END WHILE */
13 }
```

```
14     /* USER CODE BEGIN 3 */  
15 }
```

Program 1.23: Code for exercise 10

Simulation for exercise 10: LAB1_exercise10

5 Source code

Thiên Ân - 2310190 - repo: HCMUT_STM32_LAB