1 چکیده

2 مقدمه

3 آشنایی با میکروکنترلر

در این بخش میکروکنترلر های خانواده STM بررسی و سپس انواع برنامه نویسی و محیط های برنامه نویسی مناسب برای STM32F407 بررسی می‌شود.

3.1 آشنایی با میکروکنترلر های ARM

خانواده STM32F یکی از محبوب‌ترین خانواده‌های میکروکنترلرهای STM32 است که توسط شرکت STMicroelectronics تولید می‌شود. این خانواده بر اساس معماری ARM Cortex-M طراحی شده و به دلیل ویژگی‌های متنوع و عملکرد بالا، در بسیاری از کاربردها مورد استفاده قرار می‌گیرد. از جمله :

- سیستم‌های کنترل صنعتی
- تجهیزات پزشکی
- دستگاه‌های مصرفی
- سیستم‌های ارتباطی
- رباتیک
- اینترنت اشیاء (IoT)

ویژگی‌های کلیدی خانواده STM32F

1. معماری ARM Cortex-M:

- میکروکنترلرهای STM32F معمولاً بر پایه ARM Cortex-M0، M3، M4 و M7 ساخته شده‌اند که هر کدام ویژگی‌ها و قابلیت‌های خاص خود را دارند.

نسخه پیشنویس(ارسال اشکال فنی و نگارشی به zmohammadzadeh@of.iut.ac.ir)

2. عملکرد بالا:

- این میکروکنترلرها دارای فرکانس‌های کاری متنوعی هستند که می‌توانند تا ۲۴۰ مگاهرتز برسند (برای مدل‌های M7).

3. حافظه:

- خانواده STM32F شامل مقادیر مختلفی از حافظه فلش و SRAM است. برخی مدل‌ها دارای حافظه فلش تا ۲ مگابایت و SRAM تا ۵۱۲ کیلوبایت هستند.

4. پروتکل‌های ارتباطی:

- این خانواده از پروتکل‌های ارتباطی متنوعی مانند I2C، SPI، USART، USB، CAN و Ethernet پشتیبانی می‌کند.

5. ADC و DAC:

- بسیاری از مدل‌های STM32F دارای مبدل‌های آنالوگ به دیجیتال (ADC) و مبدل‌های دیجیتال به آنالوگ (DAC) هستند که برای کاربردهای سنجش و کنترل بسیار مفیدند.

6. تجهیزات جانبی:

- این خانواده شامل تایمرها، PWM، ورودی‌های دیجیتال و آنالوگ، و همچنین کنترلرهای DMA است که به بهبود عملکرد و کارایی کمک می‌کند.

7. مصرف انرژی:

- میکروکنترلرهای STM32F به گونه‌ای طراحی شده‌اند که می‌توانند در حالت‌های کم‌مصرف کار کنند، که برای برنامه‌های باتری‌خور بسیار مهم است.

زیرخانواده‌ها

خانواده STM32F به چندین زیرخانواده تقسیم می‌شود که هر کدام ویژگی‌های خاص خود را دارند:

1. STM32F0:

- مبتنی بر Cortex-M0، مناسب برای کاربردهای اقتصادی و کم‌مصرف.

2. STM32F1:

- مبتنی بر Cortex-M3، یکی از محبوب‌ترین میکروکنترلرها با قابلیت‌های خوب و عملکرد مناسب.

3. STM32F2:

نسخه پیشنویس(ارسال اشکال فنی و نگارشی به zmohammadzadeh@of.iut.ac.ir)

- مبتنی بر Cortex-M3، با عملکرد بالاتر و ویژگی‌های پیشرفته‌تر نسبت به STM32F1.

4. STM32F3:

- مبتنی بر Cortex-M4، مناسب برای کاربردهای سیگنال دیجیتال و پردازش سیگنال.

5. STM32F4:

- مبتنی بر Cortex-M4، با عملکرد بسیار بالا و قابلیت‌های پیشرفته، مناسب برای کاربردهای پیچیده.

6. STM32F7:

- مبتنی بر Cortex-M7، با بالاترین عملکرد در خانواده STM32F، مناسب برای کاربردهای پیشرفته و نیازمند پردازش بالا.

در این آزمایشگاه از STM32f407 استفاده شده است.

3.2 آشنایی با میکروکنترلر STM32F407

میکروکنترلر STM32F407 یکی از اعضای خانواده STM32F4 است که توسط شرکت STMicroelectronics تولید می‌شود. این میکروکنترلر بر پایه معماری ARM Cortex-M4 طراحی شده و به دلیل ویژگی‌های قوی و عملکرد بالا، در بسیاری از کاربردها مورد استفاده قرار می‌گیرد.

3.2.1 ویژگی‌های STM32F407

1. معماری پردازنده:

- Cortex-M4: دارای هسته ARM Cortex-M4 با فرکانس کاری تا ۱۶۰ مگاهرتز و پشتیبانی از DSP (پردازش سیگنال دیجیتال) و FPU (واحد پردازش شناور).

2. حافظه:

- حافظه فلش: تا ۱ مگابایت حافظه فلش برای ذخیره برنامه‌ها.

- حافظه SRAM: ۱۲۸ کیلوبایت SRAM برای ذخیره‌سازی داده‌ها.

3. پروتکل‌های ارتباطی:

نسخه پیشنویس(ارسال اشکال فنی و نگارشی به zmohammadzadeh@of.iut.ac.ir)

- پشتیبانی از پروتکل‌های مختلف شامل:
  - USART: برای ارتباط سریال.
  - I2C: برای ارتباط با دستگاه‌های جانبی.
  - SPI: برای ارتباط سریع با دستگاه‌های دیگر.
  - USB 2.0: پشتیبانی از USB OTG (On-The-Go).
  - CAN: برای ارتباط در شبکه‌های صنعتی.

4. ADC و DAC:
   - ADC: دارای ۱۲ کانال ADC با دقت ۱۲ بیتی و حداکثر نرخ نمونه‌برداری ۱٫۲ مگا نمونه در ثانیه.
   - DAC: دارای ۲ کانال DAC با دقت ۱۲ بیتی.

5. تجهیزات جانبی:
   - تایمرها: چندین تایمر ۱۶ و ۳۲ بیتی برای کنترل زمان و تولید سیگنال PWM.
   - کنترلر DMA: برای انتقال داده‌ها بدون دخالت CPU، که به بهبود عملکرد کمک می‌کند.

6. مصرف انرژی:
   - قابلیت کار در حالت‌های کم‌مصرف برای بهینه‌سازی مصرف انرژی در کاربردهای باتری‌خور.

7. پشتیبانی از توسعه:
   - پشتیبانی از ابزارهای توسعه مانند STM32CubeIDE و STM32CubeMX برای پیکربندی آسان و توسعه نرم‌افزار.

3.2.2 ساختار باس داخلی TBD

3.2.3 کاربردهای میکروکنترلر STM32F407

میکروکنترلر STM32F407 به دلیل ویژگی‌های قدرتمند و عملکرد بالا، در انواع مختلفی از کاربردها مورد استفاده قرار می‌گیرد، از جمله:

در زیر 200 پروژه با استفاده از STM32F407 به تفکیک دسته‌بندی‌های مختلف آورده شده است:

1. پروژه‌های GPIO

1. LED چشمک‌زن : چشمک زدن LED با زمان‌بندی مشخص.
2. کنترل موتور DC : کنترل سرعت و جهت موتور DC.
3. کنترل سروو موتور : حرکت سروو موتور به زوایای مختلف.
4. کنترل رله : روشن و خاموش کردن رله برای دستگاه‌های برقی.
5. دکمه فشار : شناسایی فشار دکمه و تغییر وضعیت LED.
6. کنترل RGB LED : تغییر رنگ LED RGB با ورودی‌های مختلف.
7. سیستم کنترل روشنایی : روشن و خاموش کردن چراغ‌ها با دکمه.
8. دستگاه بازی با دکمه‌ها : کنترل بازی با استفاده از دکمه‌ها.
9. کنترل پنکه با دکمه : روشن و خاموش کردن پنکه با دکمه.
10. سیستم کنترل ترافیک : کنترل چراغ‌های راهنمایی با دکمه‌ها.

2. پروژه‌های سنسور

11. سنسور دما و رطوبت (DHT11) : خواندن و نمایش دما و رطوبت.
12. سنسور PIR : تشخیص حرکت و فعال‌سازی آلارم.
13. سنسور نور (فتوسل) : روشن و خاموش کردن LED بر اساس نور محیط.
14. سنسور فاصله (Ultrasonic) : اندازه‌گیری فاصله و نمایش آن.
15. سنسور فشار : فعال‌سازی LED بر اساس فشار.

16. سنسور گاز : تشخیص وجود گاز و فعال‌سازی آلارم.
17. سنسور لرزش : تشخیص لرزش و فعال‌سازی آلارم.
18. سنسور باران : تشخیص بارش و فعال‌سازی آلارم.
19. سنسور رطوبت خاک : کنترل پمپ آب بر اساس رطوبت خاک.
20. سنسور دما (LM35) : خواندن دما و نمایش آن.

3. پروژه‌های ارتباطی

21. ارتباط سریال با PC : ارسال و دریافت داده‌ها از کامپیوتر.
22. ارتباط I2C با سنسور : خواندن داده‌ها از سنسور I2C.
23. ارتباط SPI با SD Card : خواندن و نوشتن داده‌ها بر روی کارت SD.
24. ارتباط RF : ارسال و دریافت داده‌ها با ماژول RF.
25. ارتباط بلوتوث : کنترل دستگاه‌ها از طریق بلوتوث.
26. ارتباط Wi-Fi با ماژول ESP8266 : اتصال به اینترنت و ارسال داده.
27. ارتباط GSM : ارسال پیامک با ماژول GSM.
28. ارتباط CAN : استفاده از پروتکل CAN برای ارتباط بین دستگاه‌ها.
29. ارتباط RS-485 : ارتباط با دستگاه‌های صنعتی.
30. ارتباط MQTT : ارسال داده‌ها به سرور MQTT.

4. پروژه‌های نمایشگر

31. نمایشگر LCD 16x2 : نمایش متن بر روی LCD.
32. نمایشگر OLED : نمایش اطلاعات بر روی OLED.
33. نمایشگر TFT : نمایش تصاویر و گرافیک بر روی TFT.
34. نمایش وضعیت سنسورها : نمایش وضعیت سنسورها بر روی LCD.
35. نمایش دما و رطوبت : نمایش داده‌های دما و رطوبت بر روی OLED.

36. نمایش زمان واقعی : استفاده از RTC برای نمایش زمان.

37. پروژه ساعت دیجیتال : نمایش ساعت و تاریخ بر روی LCD.

38. پروژه نمایشگر گرافیکی : طراحی گرافیک بر روی نمایشگر TFT.

39. پروژه نمایش داده‌های سنسور : نمایش داده‌های سنسور بر روی نمایشگر.

40. پروژه نمایش وضعیت باتری : نمایش وضعیت باتری بر روی LCD.

5. پروژه‌های کنترل

41. کنترل دما : کنترل فن بر اساس دما.

42. کنترل آبیاری خودکار : کنترل پمپ آب بر اساس رطوبت خاک.

43. کنترل چراغ‌های خیابانی : روشن و خاموش کردن چراغ‌ها بر اساس نور.

44. کنترل درب اتوماتیک : باز و بسته کردن درب با سنسور PIR.

45. کنترل ماشین رباتیک : کنترل حرکت ربات با دکمه‌ها.

46. کنترل سیستم امنیتی : استفاده از سنسور حرکتی و آلارم.

47. کنترل سیستم تهویه : کنترل فن‌ها بر اساس دما و رطوبت.

48. کنترل سیستم روشنایی هوشمند : کنترل چراغ‌ها با سنسور نور.

49. کنترل دستگاه‌های برقی با رله : کنترل دستگاه‌های برقی با رله.

50. کنترل دوربین مدار بسته : چرخش دوربین با استفاده از سروو موتور.

6. پروژه‌های صوتی

51. دستگاه بوق : فعال‌سازی بوق با فشار دکمه.

52. سیستم هشدار صوتی : فعال‌سازی آلارم صوتی در صورت تشخیص حرکت.

53. پروژه موزیک پلیر : پخش موزیک با استفاده از اسپیکر.

54. سیستم تشخیص صدا : فعال‌سازی LED در پاسخ به صدا.

55. کنترل دستگاه صوتی : کنترل پخش و توقف موزیک.

56. سیستم صدای اطلاع‌رسانی : پخش پیام‌های صوتی.

57. پروژه ضبط صدا : ضبط صدا و پخش آن.

58. پروژه صدای محیط : اندازه‌گیری سطح صدا و نمایش آن.

59. پروژه موزیک با سنسور : پخش موزیک با تشخیص حرکت.

60. سیستم هشدار با صدا : فعال‌سازی آلارم صوتی در شرایط خاص.

7. پروژه‌های آموزشی

61. آزمایشگاه الکترونیک : استفاده از دکمه‌ها و LEDها برای آموزش.

62. پروژه‌های DIY با میکروکنترلر : ترکیب پروژه‌های مختلف برای یادگیری.

63. کنترل سیستم‌های الکتریکی : آموزش کنترل دستگاه‌های الکتریکی.

64. پروژه‌های هنری با LED : طراحی الگوهای نوری با LED.

65. دستگاه تست سنسورها : خواندن و نمایش داده‌های سنسورهای مختلف.

66. پروژه‌های شبیه‌سازی : شبیه‌سازی سیستم‌های مختلف با میکروکنترلر.

67. آزمایشگاه رباتیک : طراحی و ساخت ربات‌های ساده.

68. پروژه‌های گرافیکی : طراحی گرافیک با نمایشگر TFT.

69. آموزش ارتباطات سریال : ارسال و دریافت داده‌ها از طریق UART.

70. آموزش پروتکل‌های ارتباطی : استفاده از I2C و SPI.

8. پروژه‌های اینترنت اشیاء (IoT)

71. سیستم مانیتورینگ دما و رطوبت : ارسال داده‌ها به سرور.

72. کنترل از راه دور با اپلیکیشن : کنترل دستگاه‌ها از طریق اینترنت.

73. پروژه خانه هوشمند : کنترل چراغ‌ها و دستگاه‌ها از طریق اینترنت.

74. پروژه مانیتورینگ کیفیت هوا : ارسال داده‌ها به سرور.

75. پروژه نظارت بر مصرف انرژی : ارسال داده‌های مصرف انرژی به سرور.

76. پروژه کنترل باغ هوشمند : کنترل آبیاری و نورپردازی از راه دور.

77. پروژه مانیتورینگ سلامت : ارسال داده‌های سلامت به اپلیکیشن.

78. پروژه هشدار ورود غیرمجاز : ارسال هشدار به موبایل.

79. پروژه کنترل دما از راه دور : تنظیم دما از طریق اینترنت.

80. پروژه سیستم امنیتی هوشمند : ارسال هشدار به موبایل در صورت تشخیص حرکت.

9. پروژه‌های رباتیک

81. ربات خط‌کش : حرکت ربات بر روی خط.

82. ربات جنگجو : کنترل ربات با استفاده از سنسورهای مختلف.

83. ربات خودران : استفاده از سنسورهای فاصله برای حرکت.

84. ربات جمع‌آوری اشیاء : جمع‌آوری اشیاء با استفاده از سنسور.

85. ربات پرنده : کنترل پرواز با استفاده از سنسورهای حرکتی.

86. ربات چرخشی : حرکت ربات به سمت چپ و راست.

87. ربات تعقیب‌گر : دنبال کردن اشیاء با استفاده از سنسور.

88. ربات هیدرولیکی : کنترل حرکت با استفاده از هیدرولیک.

89. ربات ردیاب صدا : حرکت به سمت منبع صدا.

90. ربات هوشمند : کنترل ربات با استفاده از اپلیکیشن موبایل.

10. پروژه‌های شبیه‌سازی

91. شبیه‌سازی سیستم‌های صنعتی : شبیه‌سازی ماشین‌های صنعتی.

92. شبیه‌سازی سیستم‌های الکتریکی : شبیه‌سازی مدارهای الکتریکی.

93. شبیه‌سازی سیستم‌های کنترل : شبیه‌سازی کنترل دما و رطوبت.

94. شبیه‌سازی سیستم‌های امنیتی : شبیه‌سازی سیستم‌های امنیتی.

95. شبیه‌سازی رباتیک : شبیه‌سازی حرکت ربات.

96. شبیه‌سازی شبکه‌های ارتباطی : شبیه‌سازی ارتباطات سریال و I2C.

97. شبیه‌سازی سیستم‌های هوشمند : شبیه‌سازی خانه‌های هوشمند.

98. شبیه‌سازی کنترل ترافیک : شبیه‌سازی چراغ‌های راهنمایی.

99. شبیه‌سازی سیستم‌های آبیاری : شبیه‌سازی آبیاری خودکار.

100. شبیه‌سازی سیستم‌های مانیتورینگ : شبیه‌سازی مانیتورینگ دما و رطوبت.

11. پروژه‌های کاربردی

101. سیستم کنترل دسترسی : استفاده از RFID برای کنترل دسترسی.

102. سیستم ثبت زمان : ثبت زمان ورود و خروج با دکمه.

103. سیستم مدیریت پارکینگ : کنترل ورود و خروج خودروها.

104. سیستم کنترل تردد : ثبت تردد افراد با استفاده از سنسور.

105. سیستم هشدار آتش‌سوزی : تشخیص دود و فعال‌سازی آلارم.

106. سیستم کنترل ترافیک هوشمند : کنترل چراغ‌های راهنمایی بر اساس ترافیک.

107. سیستم مدیریت انرژی : کنترل مصرف انرژی در ساختمان‌ها.

108. سیستم نظارت بر کیفیت آب : اندازه‌گیری کیفیت آب و ارسال داده‌ها.

109. سیستم کنترل دما و رطوبت گلخانه : کنترل دما و رطوبت در گلخانه‌ها.

110. سیستم مانیتورینگ سلامت بیمار : ارسال داده‌های سلامت به پزشک.

12. پروژه‌های چندرسانه‌ای

111. پروژه پخش موزیک : پخش موزیک با استفاده از اسپیکر.

112. پروژه ضبط صدا : ضبط صدا و پخش آن.

113. پروژه کنترل ویدئو : کنترل پخش ویدئو با دکمه‌ها.

114. پروژه نمایش تصاویر : نمایش تصاویر بر روی TFT.

115. پروژه نمایش انیمیشن : طراحی انیمیشن بر روی نمایشگر.

116. پروژه مانیتورینگ ویدئو : نمایش ویدئو از دوربین مدار بسته.

117. پروژه ضبط ویدئو : ضبط ویدئو و ذخیره آن روی SD Card.

118. پروژه پخش صدا با سنسور : پخش صدا با تشخیص حرکت.

119. پروژه پخش موزیک با بلوتوث : پخش موزیک از طریق بلوتوث.

120. پروژه نمایش متن و انیمیشن : طراحی متن و انیمیشن بر روی نمایشگر.

13. پروژه‌های امنیتی

121. سیستم امنیتی با دوربین : استفاده از دوربین برای نظارت.

122. سیستم هشدار ورود غیرمجاز : فعال‌سازی آلارم در صورت ورود غیرمجاز.

123. سیستم نظارت بر محیط : نظارت بر محیط با استفاده از سنسور.

124. سیستم کنترل دسترسی با QR Code : کنترل دسترسی با کد QR.

125. سیستم هشدار آتش‌سوزی : تشخیص آتش و فعال‌سازی آلارم.

126. سیستم نظارت بر خانه : کنترل و نظارت بر خانه از راه دور.

127. سیستم تشخیص حرکت با دوربین : فعال‌سازی آلارم در صورت تشخیص حرکت.

128. سیستم امنیتی با سنسور مغناطیسی : تشخیص باز و بسته شدن درب.

129. سیستم هشدار با سنسور صدا : فعال‌سازی آلارم در پاسخ به صدا.

130. سیستم کنترل تردد با سنسور : کنترل تردد افراد با سنسور.

14. پروژه‌های صنعتی

131. سیستم کنترل خط تولید : کنترل دستگاه‌های خط تولید.

132. سیستم مانیتورینگ صنعتی : نظارت بر دستگاه‌های صنعتی.

133. سیستم کنترل موتورهای صنعتی : کنترل موتورهای صنعتی با PWM.

134. سیستم اندازه‌گیری فشار : اندازه‌گیری فشار در سیستم‌های صنعتی.

135. سیستم کنترل دما در کارخانه : کنترل دما در کارخانه.

136. سیستم کنترل رطوبت در انبار : کنترل رطوبت در انبارها.

137. سیستم کنترل کیفیت تولید : نظارت بر کیفیت تولید.

138. سیستم کنترل ماشین‌آلات : کنترل ماشین‌آلات صنعتی.

139. سیستم مدیریت انرژی در کارخانه : کنترل مصرف انرژی در کارخانه.

140. سیستم کنترل فرآیندهای صنعتی : کنترل فرآیندهای تولید.

15. پروژه‌های تحقیقاتی

141. پروژه تحقیقاتی در زمینه IoT : بررسی کاربردهای IoT.

142. پروژه تحقیقاتی در زمینه رباتیک : بررسی تکنیک‌های رباتیک.

143. پروژه تحقیقاتی در زمینه شبکه‌های بی‌سیم : بررسی پروتکل‌های ارتباطی.

144. پروژه تحقیقاتی در زمینه امنیت سایبری : بررسی امنیت در سیستم‌های IoT.

145. پروژه تحقیقاتی در زمینه کنترل خودکار : بررسی سیستم‌های کنترل خودکار.

146. پروژه تحقیقاتی در زمینه سنسورها : بررسی انواع سنسورها و کاربردهای آن‌ها.

147. پروژه تحقیقاتی در زمینه هوش مصنوعی : بررسی کاربردهای هوش مصنوعی در رباتیک.

148. پروژه تحقیقاتی در زمینه انرژی‌های تجدیدپذیر : بررسی سیستم‌های انرژی خورشیدی.

149. پروژه تحقیقاتی در زمینه پزشکی : بررسی کاربردهای میکروکنترلرها در پزشکی.

150. پروژه تحقیقاتی در زمینه سیستم‌های هوشمند : بررسی سیستم‌های هوشمند در زندگی روزمره.

16. پروژه‌های سرگرمی

151. پروژه ساعت زنگ‌دار : طراحی ساعت زنگ‌دار با استفاده از LCD.

152. پروژه بازی با LED : طراحی بازی با استفاده از LEDها.

153. پروژه کنترل ماشین کنترلی : کنترل ماشین کنترلی با بلوتوث.

154. پروژه موزیک با دکمه : پخش موزیک با دکمه‌ها.

155. پروژه بازی حافظه : طراحی بازی حافظه با LEDها.

156. پروژه شبیه‌سازی بازی‌های کلاسیک : شبیه‌سازی بازی‌های کلاسیک با میکروکنترلر.

157. پروژه طراحی بازی‌های فکری : طراحی بازی‌های فکری با استفاده از سنسورها.

158. پروژه کنترل ربات با اپلیکیشن : طراحی اپلیکیشن برای کنترل ربات.

159. پروژه طراحی بازی‌های حرکتی : طراحی بازی‌های حرکتی با استفاده از سنسور.

160. پروژه طراحی دستگاه‌های سرگرمی : طراحی دستگاه‌های سرگرمی با میکروکنترلر.

17. پروژه‌های کاربردی در زندگی روزمره

161. سیستم کنترل روشنایی هوشمند : کنترل چراغ‌ها از طریق اپلیکیشن.

162. سیستم آبیاری هوشمند : کنترل آبیاری با سنسور رطوبت.

163. سیستم کنترل دما در خانه : کنترل دما با استفاده از سنسور.

164. سیستم مدیریت مصرف انرژی : نظارت بر مصرف انرژی در خانه.

165. سیستم کنترل سیستم‌های صوتی : کنترل سیستم‌های صوتی با بلوتوث.

166. سیستم کنترل دما و رطوبت گلخانه : کنترل دما و رطوبت در گلخانه.

167. سیستم مانیتورینگ سلامت : نظارت بر سلامت افراد در خانه.

168. سیستم کنترل دسترسی

169. سیستم کنترل دسترسی با کد QR : استفاده از کد QR برای کنترل دسترسی.

170. سیستم هشدار نشت گاز : تشخیص نشت گاز و ارسال هشدار.

171. سیستم کنترل روشنایی خودکار : روشن و خاموش کردن چراغ‌ها با سنسور نور.

172. سیستم مدیریت پارکینگ هوشمند : نظارت بر فضای پارکینگ و ثبت ورود و خروج.

173. سیستم آبیاری اتوماتیک باغ : کنترل آبیاری باغ با سنسور رطوبت.

174. سیستم کنترل دما و رطوبت در انبار : نظارت بر شرایط انبار.

175. سیستم هشدار سرقت : فعال‌سازی آلارم در صورت حرکت غیرمجاز.

176. سیستم نظارت بر مصرف آب : اندازه‌گیری و نمایش مصرف آب.

177. پروژه کنترل دستگاه‌های خانگی : کنترل دستگاه‌های خانگی از طریق اپلیکیشن.

178. سیستم مانیتورینگ کیفیت هوا : اندازه‌گیری و نمایش کیفیت هوا.

18. پروژه‌های خلاقانه

179. پروژه ساعت هوشمند : طراحی ساعت با قابلیت‌های هوشمند.

180. پروژه ربات پرنده : طراحی ربات پرنده با کنترل از راه دور.

181. پروژه ربات چرخشی : طراحی رباتی که می‌تواند بچرخد و موانع را دور بزند.

182. پروژه دستگاه بازی با سنسور حرکتی : طراحی بازی که بر اساس حرکات کاربر عمل می‌کند.

183. پروژه کنترل ربات با حرکات دست : کنترل ربات با استفاده از سنسورهای حرکتی.

184. پروژه طراحی دستگاه‌های هنری : طراحی دستگاهی که می‌تواند نقاشی کند.

185. پروژه ربات تعقیب‌گر نور : طراحی رباتی که به سمت نور حرکت می‌کند.

186. پروژه طراحی دستگاه‌های موسیقی : ساخت دستگاهی که می‌تواند موزیک تولید کند.

187. پروژه طراحی بازی‌های آموزشی : طراحی بازی‌هایی برای آموزش مفاهیم مختلف.

188. پروژه طراحی گجت‌های هوشمند : طراحی گجت‌هایی برای زندگی روزمره.

19. پروژه‌های تحقیقاتی و علمی

189. پروژه تحقیقاتی در زمینه سنسورهای زیستی : بررسی کاربرد سنسورهای زیستی.

190. پروژه تحقیقاتی در زمینه رباتیک هوش مصنوعی : بررسی تکنیک‌های هوش مصنوعی در رباتیک.

191. پروژه تحقیقاتی در زمینه انرژی‌های تجدیدپذیر : تحلیل سیستم‌های انرژی خورشیدی.

192. پروژه تحقیقاتی در زمینه کنترل خودکار : بررسی روش‌های کنترل خودکار.

193. پروژه تحقیقاتی در زمینه اینترنت اشیاء : بررسی کاربردهای IoT در صنایع مختلف.

194. پروژه تحقیقاتی در زمینه سیستم‌های هوشمند : تحلیل سیستم‌های هوشمند در زندگی روزمره.

195. پروژه تحقیقاتی در زمینه ارتباطات بی‌سیم : بررسی پروتکل‌های ارتباطی بی‌سیم.

196. پروژه تحقیقاتی در زمینه امنیت سایبری : بررسی تهدیدات امنیتی در سیستم‌های IoT.

197. پروژه تحقیقاتی در زمینه پزشکی : بررسی کاربردهای STM32 در پزشکی.

198. پروژه تحقیقاتی در زمینه سیستم‌های نظارتی : بررسی تکنیک‌های نظارتی و امنیتی.

20. پروژه‌های سرگرمی و تفریحی

199. پروژه ساخت دستگاه بازی با استفاده از LED : طراحی بازی‌های سرگرم‌کننده با LED.

200. پروژه طراحی یک گیم‌پد با STM32 : طراحی گیم‌پدی برای بازی‌های ویدئویی.

3.3 برنامه نویسی میکروکنترلر

برنامه‌نویسی HAL (Hardware Abstraction Layer) و LL (Low Layer) برای میکروکنترلرهایSTM32 ، دو رویکرد متفاوت برای توسعه نرم‌افزار هستند که هر کدام مزایا و معایب خاص خود را دارند.

3.3.1 HAL (Hardware Abstraction Layer)

HALیک لایه انتزاعی است که به برنامه‌نویس این امکان را می‌دهد تا بدون نیاز به درک عمیق از سخت‌افزار، با میکروکنترلر کار کند. این لایه شامل توابعی است که به راحتی می‌توانند برای پیکربندی و کنترل سخت‌افزار استفاده شوند.

مزایا:

- سهولت استفاده: توابع HAL معمولاً ساده و کاربرپسند هستند و به سرعت می‌توانند برای توسعه نرم‌افزار استفاده شوند.

- کاهش زمان توسعه: با استفاده ازHAL ، زمان توسعه نرم‌افزار به طور قابل توجهی کاهش می‌یابد.

- پشتیبانی از چندین میکروکنترلر: با استفاده ازHAL ، می‌توان برنامه‌ها را به راحتی بین میکروکنترلرهای مختلف STM32 منتقل کرد.

معایب:

- عملکرد پایین‌تر: به دلیل انتزاعی بودن، ممکن است عملکرد کمتری نسبت به کدهای نوشته شده به صورت مستقیم داشته باشد.

- حجم کد بیشتر: کدهای HAL معمولاً بزرگ‌تر و پیچیده‌تر از کدهای LL هستند.

3.3.2 LL (Low Layer)

LLیک لایه پایین‌تر است که به برنامه‌نویس این امکان را می‌دهد تا به صورت مستقیم با سخت‌افزار تعامل داشته باشد. این لایه شامل توابعی است که به طور خاص برای پیکربندی و کنترل سخت‌افزار طراحی شده‌اند.

مزایا:

- عملکرد بالاتر: کدهای LL به دلیل نزدیک‌تر بودن به سخت‌افزار، معمولاً عملکرد بهتری دارند.

- کنترل بیشتر: برنامه‌نویس کنترل بیشتری بر روی جزئیات سخت‌افزاری دارد و می‌تواند بهینه‌سازی‌های خاصی انجام دهد.

معایب:

- پیچیدگی بیشتر: استفاده از LL نیاز به دانش عمیق‌تری از سخت‌افزار و معماری میکروکنترلر دارد.

- زمان توسعه بیشتر: به دلیل پیچیدگی، زمان بیشتری برای توسعه نرم‌افزار لازم است.

انتخاب بین HAL و LL بستگی به نیازهای پروژه و تجربه برنامه‌نویس دارد. اگر به دنبال توسعه سریع و آسان هستید، HAL گزینه مناسبی است. اما اگر نیاز به عملکرد بالا و کنترل دقیق دارید، LLمی‌تواند انتخاب بهتری باشد. در این آزمایشگاه از هر دو روش استفاده شده است.

3.4 آشنایی با نرم افزارها

محیط‌های برنامه‌نویسی برای میکروکنترلرهای STM32 به توسعه‌دهندگان این امکان را می‌دهند که به راحتی نرم‌افزارهای خود را طراحی و پیاده‌سازی کنند. در اینجا به برخی از مهم‌ترین محیط‌های برنامه‌نویسی STM اشاره می‌کنیم:

3.4.1 STM32CubeIDE

STM32CubeIDE یک محیط توسعه یکپارچه (IDE) است که توسط STMicroelectronics ارائه شده و بر پایه Eclipse ساخته شده است.

ویژگی ها:

- پشتیبانی از HAL و LL: شامل ابزارهای لازم برای استفاده از هر دو لایه HAL و LL.

- مدیریت پروژه: امکان مدیریت آسان پروژه‌ها و تنظیمات مختلف.
- تنظیمات گرافیکی: ابزار پیکربندی سخت‌افزار (STM32CubeMX) به صورت گرافیکی که به راحتی می‌توان پین‌ها و تنظیمات دیگر را پیکربندی کرد.
- اشکال‌زدایی: ابزارهای قوی برای اشکال‌زدایی و مشاهده وضعیت متغیرها.

3.4.2 Keil MDK

Keil MDK (Microcontroller Development Kit) یک محیط توسعه معروف برای میکروکنترلرهای ARM است.

ویژگی ها:

- کامپایلر قدرتمند: شامل کامپایلر ARM C/C++ که بهینه‌سازی‌های بالایی دارد.
- کتابخانه‌های استاندارد: شامل کتابخانه‌های استاندارد برای توسعه نرم‌افزار.
- اشکال‌زدایی پیشرفته: ابزارهای اشکال‌زدایی قوی و شبیه‌سازی.

3.4.3 IAR Embedded Workbench

IAR Embedded Workbench یک IDE محبوب و قدرتمند برای توسعه نرم‌افزارهای میکروکنترلر است.

ویژگی ها:

- عملکرد بالا: کامپایلر بهینه‌سازی شده برای عملکرد و اندازه کد.
- ابزارهای اشکال‌زدایی: ابزارهای پیشرفته برای اشکال‌زدایی و تجزیه و تحلیل کد.
- پشتیبانی از چندین میکروکنترلر: پشتیبانی از انواع مختلف میکروکنترلرها از جمله STM32.

3.4.4 PlatformIO

PlatformIO یک محیط توسعه متن‌باز است که برای توسعه نرم‌افزارهای میکروکنترلر طراحی شده است.

ویژگی ها:

- پشتیبانی از چندین پلتفرم: پشتیبانی از انواع مختلف میکروکنترلرها و بردها.
- مدیریت کتابخانه: مدیریت آسان کتابخانه‌ها و وابستگی‌ها.
- یکپارچگی با ویرایشگرهای متن: قابلیت کار با ویرایشگرهای محبوب مانند Visual Studio Code.

3.4.5 Arduino IDE

Arduino IDE به طور خاص برای توسعه نرم‌افزارهای مبتنی بر Arduino طراحی شده، اما می‌توان از آن برای میکروکنترلرهای STM32 نیز استفاده کرد.

ویژگی ها:

- سادگی استفاده: رابط کاربری ساده و کاربرپسند.
- کتابخانه‌های آماده: دسترسی به کتابخانه‌های مختلف برای تسهیل توسعه.
- پشتیبانی از پروتکل‌های مختلف: پشتیبانی از پروتکل‌های مختلف ارتباطی.

انتخاب محیط برنامه‌نویسی مناسب بستگی به نیازهای پروژه، تجربه برنامه‌نویس و نوع میکروکنترلر دارد. هر یک از این محیط‌ها ویژگی‌ها و مزایای خاص خود را دارند که می‌توانند به تسهیل فرآیند توسعه کمک کنند. در این آزمایشگاه از Keil MDK در کنار STM32cubeMXاستفاده می‌شود.

4 نرم افزار STM32CubeMX

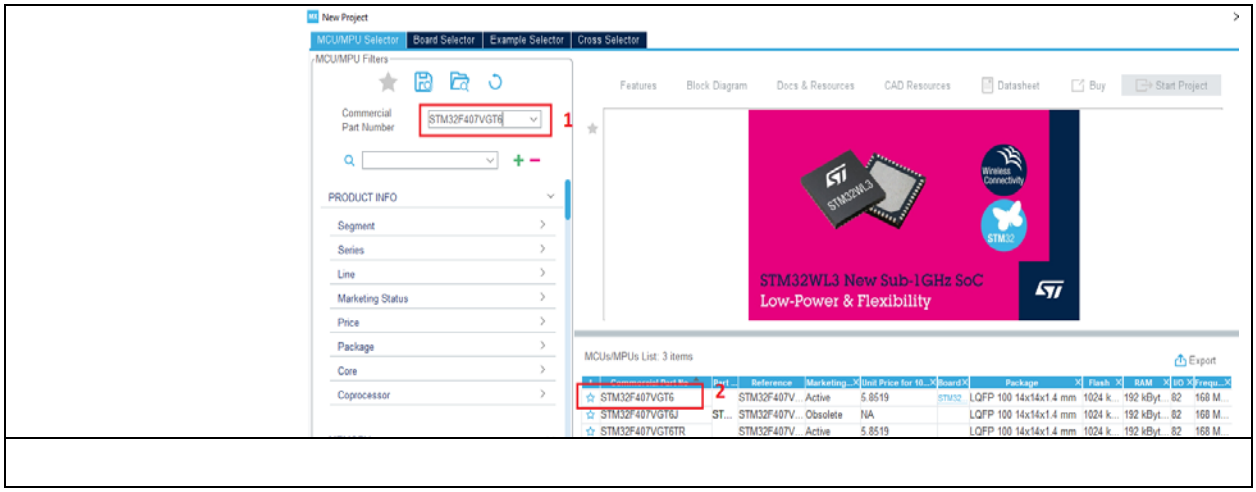
برنامه‌نویسی میکروکنترلر STM32F407 با استفاده از STM32CubeMX یک روش کارآمد و ساده برای توسعه نرم‌افزار است. STM32CubeMX یک ابزار گرافیکی است که به شما امکان می‌دهد تا پیکربندی سخت‌افزار و تنظیمات پروژه را به راحتی انجام دهید. در ادامه مراحل اصلی کار با STM32CubeMX برای STM32F407 را بررسی می‌کنیم.

4.1 نصب STM32CubeMX

- دانلود و نصب: ابتدا باید STM32CubeMX را از وب‌سایت STMicroelectronics دانلود و نصب کنید.

4.2 ایجاد یک پروژه جدید

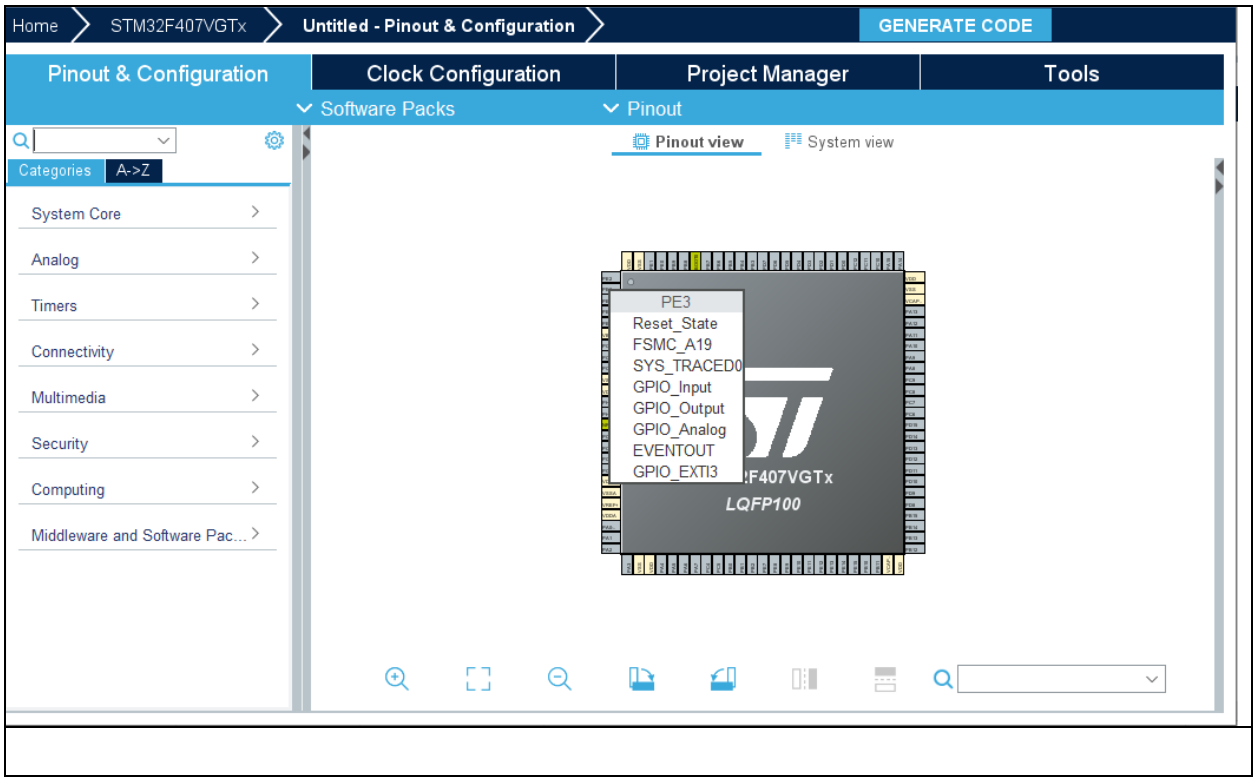
- شروع پروژه: پس از باز کردن STM32CubeMX، گزینه "File\New Project" را انتخاب کنید.
- انتخاب میکروکنترلر: در پنجره جستجو، STM32F407 را پیدا کرده و انتخاب کنید. می‌توانید از طریق شماره مدل یا سری میکروکنترلر جستجو کنید.



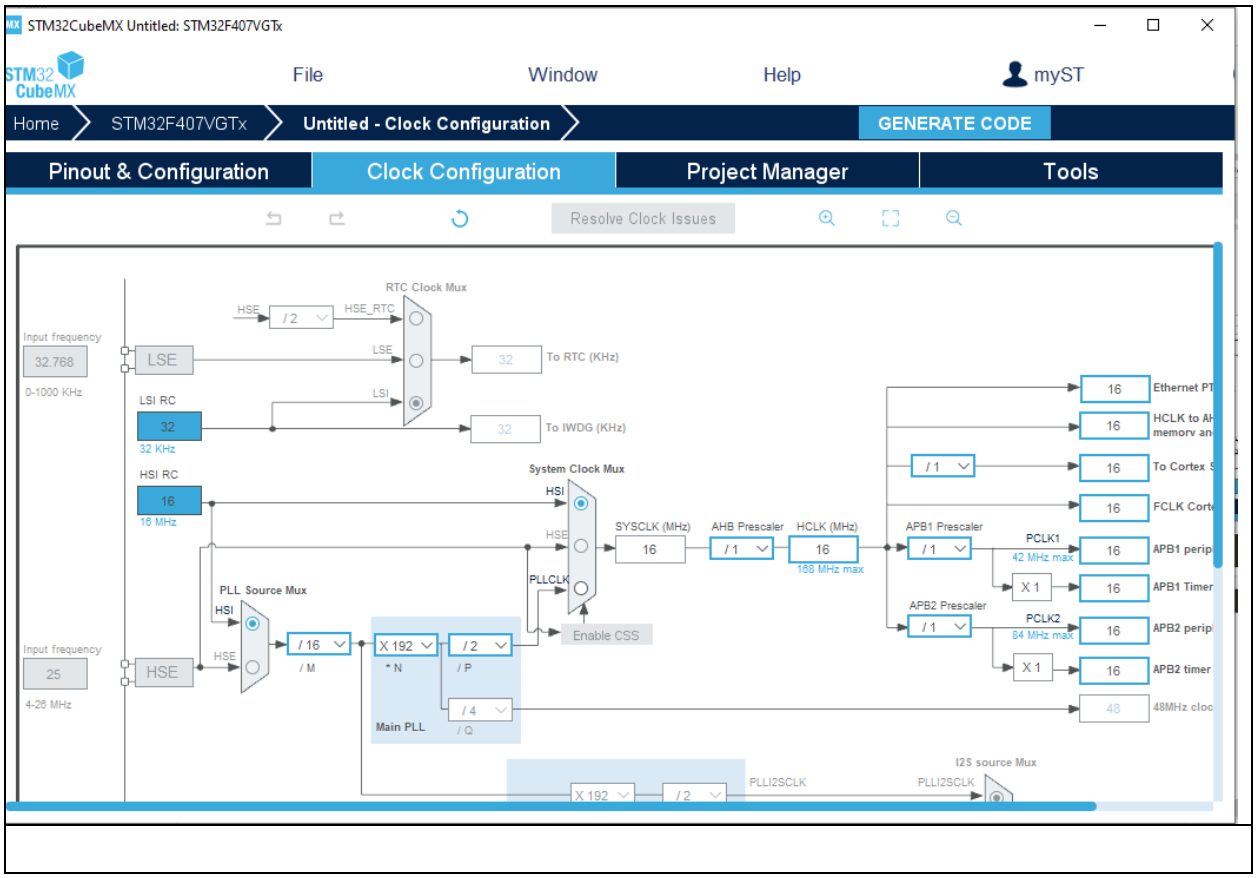
4.3 پیکربندی سخت‌افزار

- پیکربندی پین‌ها: با کلیک بر روی پین‌های مختلف، می‌توانید وظایف آن‌ها (مثل GPIO، UART، SPI و ...) را تنظیم کنید.

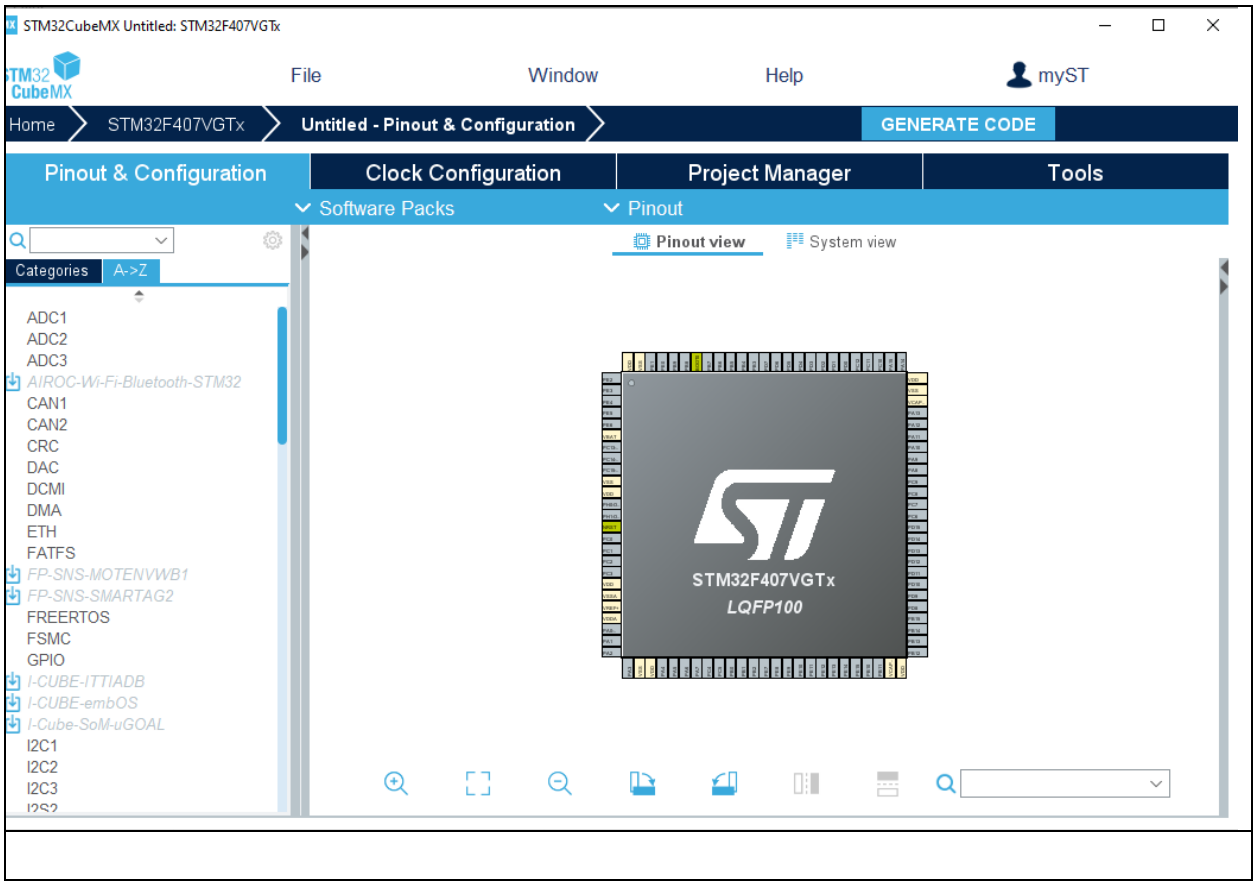
نسخه پیشنویس(ارسال اشکال فنی و نگارشی به zmohammadzadeh@of.iut.ac.ir)



- تنظیمات Clock: تنظیمات مربوط به کلاک میکروکنترلر را پیکربندی کنید تا فرکانس‌های مورد نیاز را تعیین کنید.

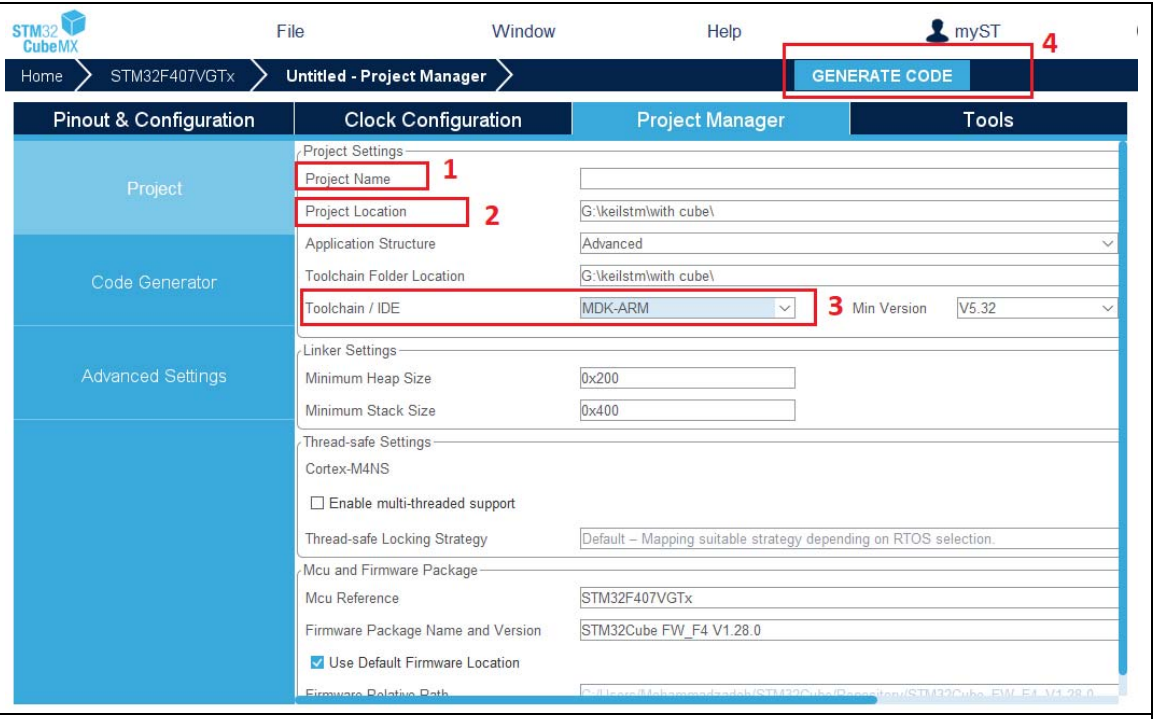


- پیکربندی پریفرال‌ها: از بخش "Peripherals" می‌توانید پریفرال‌هایی مانند ADC، DAC، Timer و غیره را فعال و پیکربندی کنید.



4.4 تنظیمات پروژه

- تنظیمات پروژه: در بخش "Project"، نام پروژه، مسیر ذخیره‌سازی و نوع IDE (مثل MDK-ARM) را انتخاب کنید.



4.5 تولید کد

- تولید کد: پس از انجام تنظیمات، بر روی دکمه "Project" و سپس "Generate Code" کلیک کنید. این کار کد پایه برای پروژه شما را تولید می‌کند.

4.6 توسعه نرم‌افزار

- باز کردن IDE: پروژه تولید شده را در Keil Uvision خود باز کنید.

- نوشتن کد: در فایل‌های تولید شده، می‌توانید کدهای خود را برای پیاده‌سازی منطق برنامه بنویسید. معمولاً کد اصلی در فایل `main.c` قرار دارد.

4.7 اشکال‌زدایی و تست

- بارگذاری کد: از طریق J-link یا ST-Link، کد را بر روی میکروکنترلر بارگذاری کنید.

- اشکال‌زدایی: با استفاده از ابزارهای اشکال‌زدایی موجود درKeil، می‌توانید برنامه را تست و اشکال‌زدایی کنید.

5 آشنایی با Keil MDK---TBD

7 تنظیمات ساعت

7.1 منبع کلاک

سه منبع ساعت مختلف می‌توانند برای راه‌اندازی ساعت سیستم (SYSCLK) استفاده شوند:

1- منبع ساعت سرعت بالا:

HSI (High-Speed Internal)

این منبع ساعت یک نوسان‌ساز داخلی است که به‌طور پیش‌فرض با فرکانس 16 مگاهرتز کار می‌کند HSI برای بیشتر کاربردهای عمومی مناسب است و نیازی به سخت‌افزار خارجی ندارد.

HSE (High-Speed External)

این منبع ساعت از یک کریستال خارجی یا نوسان‌ساز استفاده می‌کند که معمولاً دارای فرکانس 8 تا 26 مگاهرتز است . HSEاغلب برای کاربردهایی که به دقت بالا نیاز دارند، استفاده می‌شود.

2- منبع ساعت سرعت پایین

LSE (Low-Speed External):

این منبع ساعت بیشتر برای کاربردهایی مانند ساعت‌های RTC (Real-Time Clock) استفاده می‌شود. فرکانس معمولاً 32.768 کیلوهرتز است و از یک کریستال خارجی استفاده می‌کند.

LSI (Low-Speed Internal):

این منبع ساعت یک نوسان‌ساز داخلی است که معمولاً دارای فرکانس حدود 40 کیلوهرتز است LSI .برای کارکردهای ساده و مهم مانند RTC مناسب است.

3- ساعت PLL اصلی (PLL)

دارای یک مدار الکترونیکی است که برای تولید یک فرکانس خروجی با دقت بالا و هم‌فاز با یک فرکانس مرجع مورد استفاده قرار می‌گیرد. در میکروکنترلرSTM32F407 ، PLLبه‌عنوان ابزاری برای تولید فرکانس‌های بالاتر از منابع ساعت موجود) مانند HSE و (HSI استفاده می‌شود. این ویژگی به توسعه‌دهندگان اجازه می‌دهد تا فرکانس سیستم را بر اساس نیازهای خاص پروژه، افزایش دهند.

هر منبع ساعت می‌تواند به‌طور مستقل روشن یا خاموش شود زمانی که استفاده نمی‌شود، تا مصرف انرژی بهینه‌سازی شود.

7.2 رجیسترهای کلاک TBD

To be defiend(TBD)

7.3 نحوه راه‌اندازی کلاک در STM32F407 با استفاده از STM32Cube

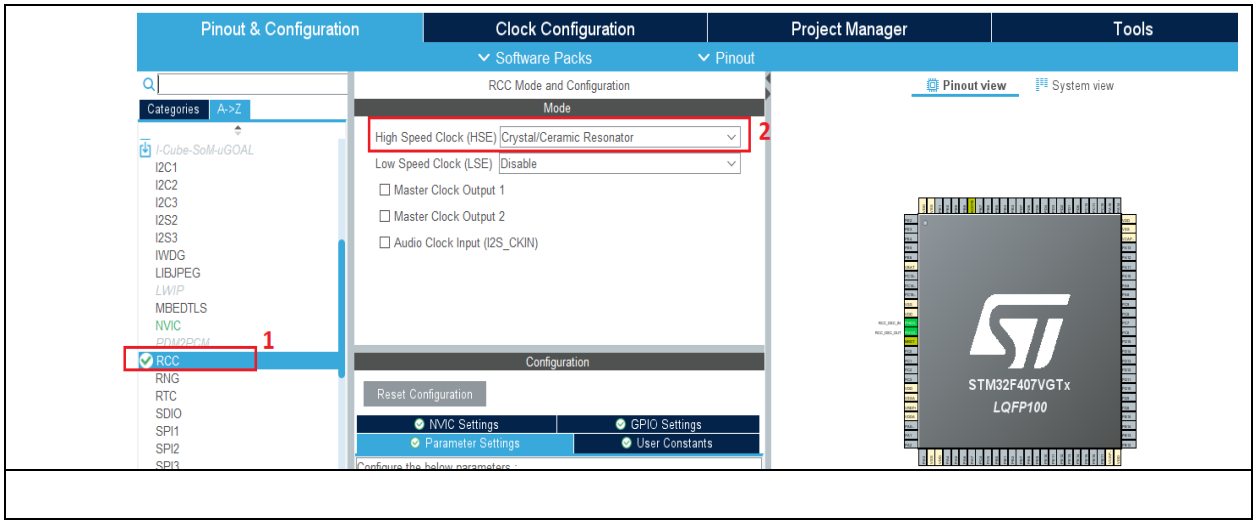
برای راه‌اندازی کلاک در میکروکنترلر STM32F407 با استفاده از STM32CubeMX، مراحل زیر را دنبال کنید:

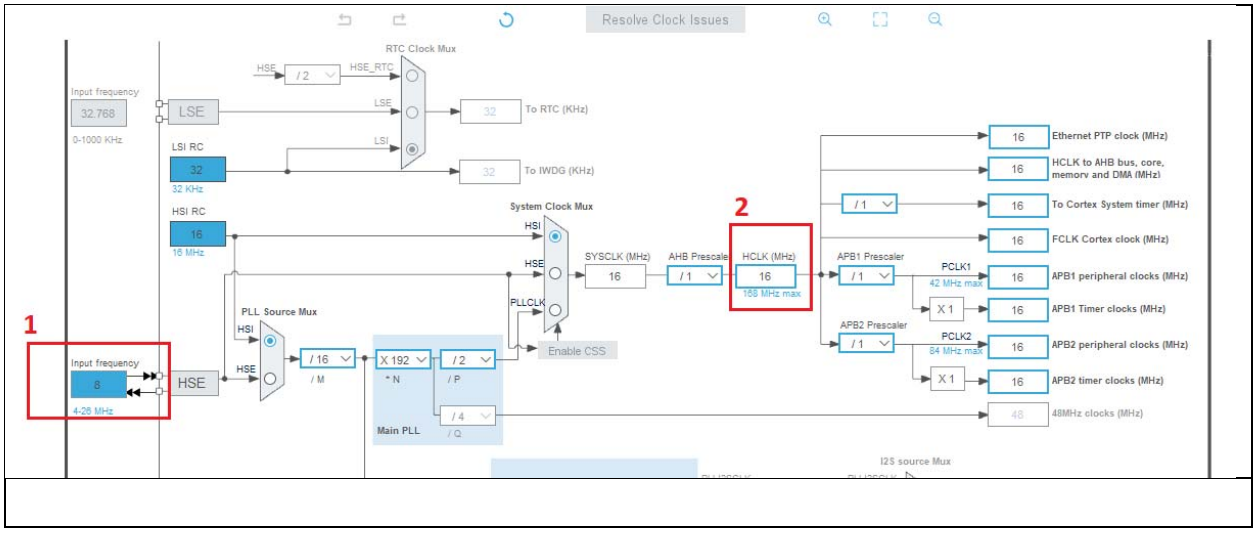
1. ایجاد پروژه جدید

- نرم‌افزار STM32CubeMX را باز کنید.
- یک پروژه جدید ایجاد کنید و میکروکنترلر STM32F407 را انتخاب کنید.

2. پیکربندی کلاک

- به تب "Clock Configuration" بروید.
- در این بخش، می‌توانید منابع کلاک مختلف را مشاهده کنید:
  - HSI (High-Speed Internal)
  - HSE (High-Speed External)
  - PLL (Phase-Locked Loop)
- منبع کلاک اصلی (SYSCLK) را انتخاب کنید. برای مثال، می‌توانید HSE را انتخاب کنید و فرکانس آن را تنظیم کنید.





3. تنظیم PLL

- اگر از PLL استفاده می‌کنید، پارامترهای مربوط به آن را تنظیم کنید:

- PLLM : تقسیم‌کننده ورودی PLL

- PLLN : ضرب‌کننده PLL

- PLLP : تقسیم‌کننده خروجی PLL

- اطمینان حاصل کنید که فرکانس نهایی SYSCLK در محدوده مجاز میکروکنترلر قرار دارد.

4. تنظیم کلاک‌های جانبی

- به تب "Pinout & Configuration" بروید و هر دستگاه جانبی که نیاز به کلاک دارد را فعال کنید.

- به تب "Configuration" بروید و تنظیمات مربوط به RCC (Reset and Clock Control) را بررسی کنید

نکات مهم

- همیشه فرکانس‌های مجاز برای میکروکنترلر STM32F407 را بررسی کنید تا از آسیب به دستگاه جلوگیری کنید.

- در صورت استفاده از HSE، اطمینان حاصل کنید که کریستال خارجی به درستی متصل شده است.

پس از تغییر کلاک پردازنده سایر کلاک ها به صورت اتوماتیک محاسبه می‌گردد.

5. نهایتا میتوانید کدهای مورد نظر را تولید و در محیط Keil زیربرنامه مربوطه با نام SystemClock_Config() را ملاحظه نمایید.

درگاههای STM32f407

درگاههای همه منظوره STM32f407 قابل دسترس روی پکیج آزمایشگاه در جدول نشان داده شده است.

PORTA	0,1,2,3,4,5,6,7,8,9,10,11,12
PORTB	0,1,5,6,7,8,9,10,11,12,13,14,15
PORTC	0,1,2,3,4,5,6,7,8,9,10,11,12,13
PORTD	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
PORTE	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
PORTH	0,1

7.4 معرفی رجیسترهای GPIO

در میکروکنترلر STM32F407 ، رجیسترهای GPIO (General Purpose Input/Output) به شما اجازه می‌دهند تا پین‌های ورودی و خروجی را پیکربندی و کنترل کنید. این رجیسترها شامل موارد زیر هستند:

7.4.1 پیکربندی حالت پین GPIOx_MODER

این رجیستر برای تنظیم حالت هر پین GPIO استفاده می‌شود. هر پین می‌تواند به یکی از حالت‌های زیر تنظیم شود:

-ورودی(00)

-خروجی(01)

-حالت چندمنظوره(10)

-حالت آنالوگ(11)

حالت پین‌های GPIO را می‌توانید با تنظیم رجیستر MODER پیکربندی کنید. این رجیستر ۲ بیت برای هر پین دارد.

GPIOA->MODER &= ~(0x3 << (2 * pin_number)); // پاک کردن حالت قبلی

GPIOA->MODER |= (0x1 << (2 * pin_number)); // تنظیم به حالت خروجی

7.4.2 پیکربندی نوع خروجی GPIOx_OTYPER

اگر پین را به عنوان خروجی تنظیم کرده‌اید، باید نوع خروجی را مشخص کنید (Push-Pull یا Open-Drain) با استفاده از رجیستر OTYPER.

-خروجی Push-Pull (0)

-خروجی Open-Drain (1)

```
GPIOA->OTYPER &= ~(0x1 << pin_number); // Push-Pull
// GPIOA->OTYPER |= (0x1 << pin_number); // Open-Drain
```

7.4.3 پیکربندی سرعت GPIOx_OSPEEDR

سرعت پین‌های GPIO را می‌توان با تنظیم رجیستر OSPEEDR مشخص کرد.

-سرعت پایین (00)

-سرعت متوسط (01)

-سرعت بالا (10)

-سرعت بسیار بالا (11)

```
GPIOA->OSPEEDR &= ~(0x3 << (2 * pin_number)); // پاک کردن سرعت قبلی
GPIOA->OSPEEDR |= (0x2 << (2 * pin_number));  // تنظیم به سرعت بالا
```

7.4.4 پیکربندی Pull-up/Pull-down GPIOx_PUPDR

برای تنظیم Pull-up یا Pull-down ، از رجیستر PUPDR استفاده کنید.

-بدون Pull (00)

- Pull-Up (01)

- Pull-Down (10)

```
GPIOA->PUPDR &= ~(0x3 << (2 * pin_number)); // پاک کردن پیکربندی قبلی
GPIOA->PUPDR |= (0x1 << (2 * pin_number));  // تنظیم به Pull-up
// GPIOA->PUPDR |= (0x2 << (2 * pin_number));  // تنظیم به Pull-down
```

7.4.5 نوشتن به پین خروجی GPIOx_ODR

این رجیستر برای نوشتن وضعیت به پین‌های خروجی استفاده می‌شود. هر بیت در این رجیستر نشان‌دهنده وضعیت یک پین خروجی است.

```
GPIOA->ODR |= (1 << pin_number); // تنظیم پین به HIGH
GPIOA->ODR &= ~(1 << pin_number); // تنظیم پین به LOW
```

7.4.6 خواندن از پین ورودی: GPIOx_IDR

```
if (GPIOA->IDR & (1 << pin_number)) {
    // پین HIGH است
} else {
    // پین LOW است
}
```

7.4.7 GPIOx_BSRR

این رجیستر برای تنظیم یا پاک کردن یک پین خروجی به‌طور مستقل استفاده می‌شود. با نوشتن یک بیت 1 در این رجیستر، پین مربوطه به حالت بالا (Set) می‌رود و با نوشتن یک بیت 1 در بخش بالایی، پین به حالت پایین (Reset) می‌رود.

7.4.8 GPIOx_LCKR

این رجیستر برای قفل کردن پیکربندی پین‌ها استفاده می‌شود. پس از قفل کردن، تنظیمات پین‌ها نمی‌توانند تغییر کنند تا زمانی که میکروکنترلر ریست شود.

7.4.9 GPIOx_AFRL و GPIOx_AFRH

این رجیسترها برای تنظیم عملکرد چندمنظوره پین‌ها استفاده می‌شوند. این امکان را می‌دهد که پین‌ها به عنوان یک عملکرد خاص) مثلUART ، SPIو (... تنظیم شوند.

7.5 زیربرنامه های موجود در کتابخانه HAL برای GPIO

کتابخانه HAL (Hardware Abstraction Layer) برای GPIO در میکروکنترلرهای STM32 شامل توابع متعددی است که برای پیکربندی و مدیریت پین‌های GPIO استفاده می‌شوند. در زیر فهرستی از توابع اصلی موجود در این کتابخانه آورده شده است:

7.5.1 پیکربندی GPIO

- `HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)`

  برای پیکربندی یک پورت GPIO.

7.5.2 نوشتن به GPIO

- `HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)`

  برای نوشتن وضعیت (HIGH یا LOW) به یک پین GPIO.

7.5.3 خواندن از GPIO

- `HAL_GPIO_ReadPin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)`

  برای خواندن وضعیت یک پین GPIO.

7.5.4 تغییر وضعیت GPIO

- `HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)`

برای تغییر وضعیت یک پین (از HIGH به LOW و بالعکس).

7.5.5 غیرفعال کردن GPIO

- `HAL_GPIO_DeInit(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)`

برای غیرفعال کردن یک پین GPIO.

7.5.6 ساختارهای داده GPIO_InitTypeDef

ساختاری برای تعریف ویژگی‌های پیکربندی پین GPIO، شامل:

- `Pin`: شماره پین.
- `Mode`: حالت (ورودی، خروجی، آنالوگ، و غیره).
- `Pull`: حالت Pull-up یا Pull-down.
- `Speed`: سرعت پین.
- `Alternate`: حالت جایگزین (برای پین‌های چندمنظوره).

در برنامه 7-1 تنظیمات و فرامین اولیه برای درگاهها نشان داده شده است.

```
GPIO_InitTypeDef GPIO_InitStruct = {0};

// پیکربندی پین 5 از پورت A به عنوان خروجی
__HAL_RCC_GPIOA_CLK_ENABLE(); // فعال‌سازی کلاک پورت A

GPIO_InitStruct.Pin = GPIO_PIN_5;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; // خروجی Push-Pull
GPIO_InitStruct.Pull = GPIO_NOPULL; // بدون Pull-up/Pull-down
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW; // سرعت پایین

HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); // پیکربندی GPIO

// نوشتن به GPIO
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); // روشن کردن LED
```

برنامه 7-1:کد های اولیه برای تنظیمات درگاهها

7.6 راه اندازی درگاه با استفاده از رجیسترها

برای پیکربندی GPIO در STM32F407 به روش رجیستریLow Level ، به صورت ورودی مراحل زیر را دنبال کنید:

```
.....
.....

    // 1. فعال‌سازی کلاک برای GPIOA
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; // روشن کردن کلاک GPIOA

    // 2. پیکربندی پین 1 (PA1) به عنوان ورودی
    GPIOA->MODER &= ~(0x3 << (1 * 2)); // پاک کردن تنظیمات قبلی (PINA1 را به حالت ورودی تنظیم می‌کند)

    // 3. خواندن مقدار از پین PA1
    while (1)
    {
        // 4. بررسی وضعیت ورودی
        if (GPIOA->IDR & (1 << 1)) // اگر PA1 HIGH باشد
        {
            // بودن پین اجرا شود HIGH در اینجا می‌توانید کدی بنویسید که در صورت
            // را روشن کنید یا به حالت دیگر بروید مثلاً می‌توانید یک LED
        }
        else
        {
            // بودن پین اجرا شود LOW در اینجا می‌توانید کدی بنویسید که در صورت
            // دیگری را روشن کنید یا کاری دیگر انجام دهید مثلاً می‌توانید LED
        }
    }
```

برای پیکربندی GPIO در STM32F407 به روش رجیستریLow Level ، به صورت خروجی مراحل زیر را دنبال کنید:

```
.....
.....
  // 1. فعال‌سازی کلاک برای GPIOA
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; // روشن کردن کلاک GPIOA

  // 2. پیکربندی پین 1 (PA1) به عنوان خروجی
  GPIOA->MODER &= ~(0x3 << (1 * 2)); // پاک کردن تنظیمات قبلی
  GPIOA->MODER |= (0x1 << (1 * 2));  // تنظیم به حالت خروجی (01)

  // 3. تنظیم سرعت پین به سرعت بالا (در صورت نیاز)
```

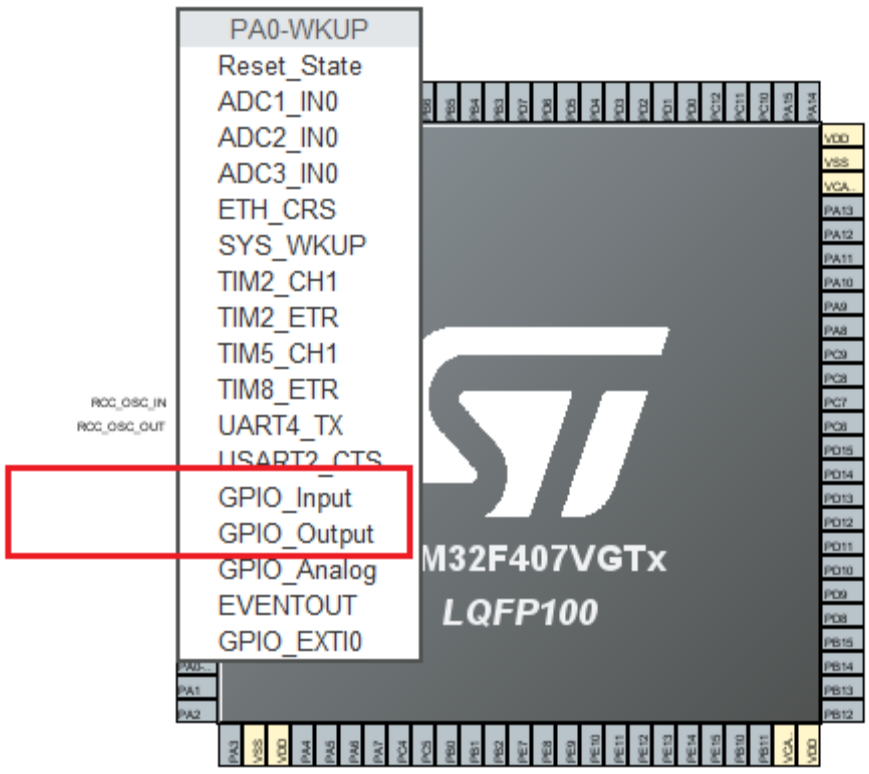
```
GPIOA->OSPEEDR |= (0x2 << (1 * 2)); // سرعت بالا (11)

while (1)
{
   // 4. 1 روشن کردن پین (PA1)
   GPIOA->ODR |= (1 << 1);  // تنظیم بیت 1 در رجیستر خروجی
   for (volatile int i = 0; i < 100000; i++); // تأخیر

   // 5. 1 خاموش کردن پین (PA1)
   GPIOA->ODR &= ~(1 << 1); // پاک کردن بیت 1 در رجیستر خروجی
   for (volatile int i = 0; i < 100000; i++); // تأخیر
}
```

راه‌اندازی GPIO (General Purpose Input/Output) در STM32 با استفاده از STM32CubeMX شامل چند مرحله است. در ادامه مراحل کلی را توضیح می‌دهم:

مراحل راه‌اندازی GPIO در STM32 با STM32CubeMX

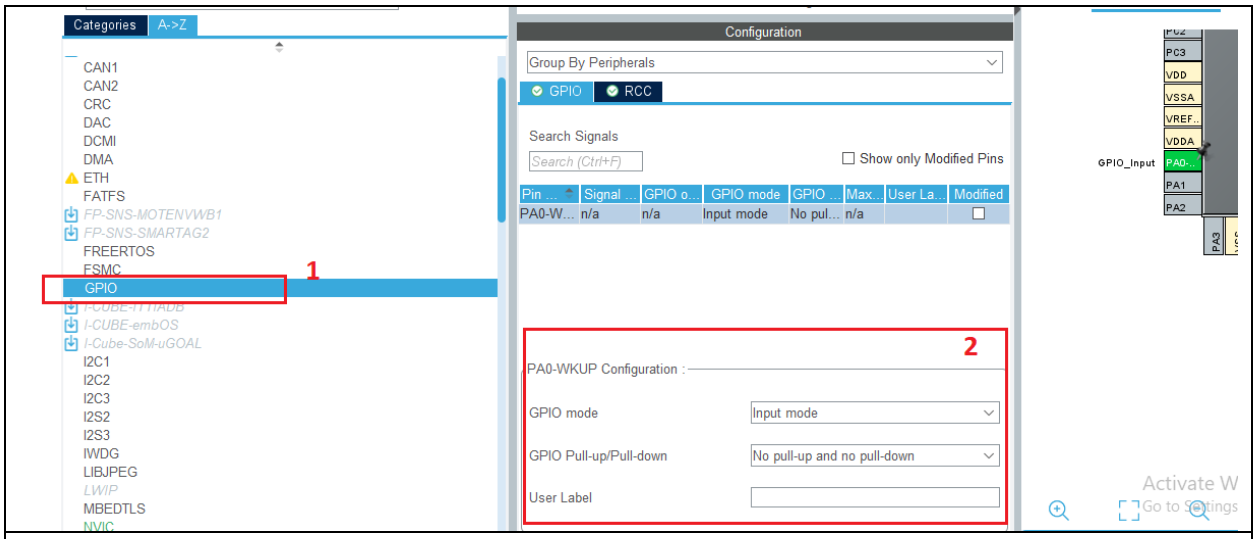
3. تنظیمات GPIO :

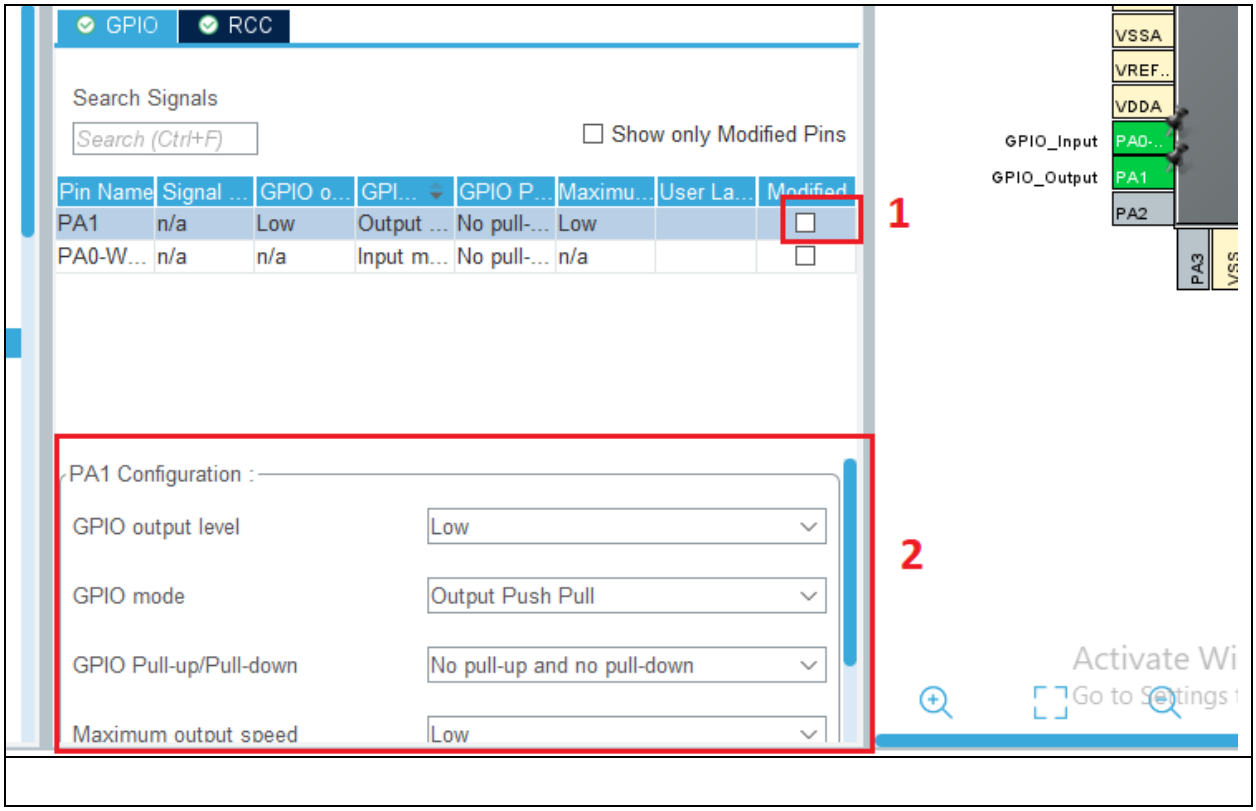
- در بخش "Pinout & Configuration"، پین‌های مورد نظر خود را برای ورودی یا خروجی انتخاب کنید.



- برای پین‌های خروجی، نوع خروجی (Push-Pull یا Open-Drain) و سرعت را تنظیم کنید.

- برای پین‌های ورودی، می‌توانید حالت (Pull-up, Pull-down یا Floating) را مشخص کنید.





4. تنظیمات Clock :

- اطمینان حاصل کنید که کلاک‌های مربوط به GPIO فعال شده‌اند. این معمولاً به‌طور خودکار انجام می‌شود.

پس از تنظیمات کدها را تولید و در keil ملاحظه نماید. تنظیمات GPIO در قالب زیربرنامه MX_GPIO_Init() قابل ملاحظه است.

7.7 برنامه های اجرایی در آزمایشگاه

7.7.1 پروژه LED چشمک زن

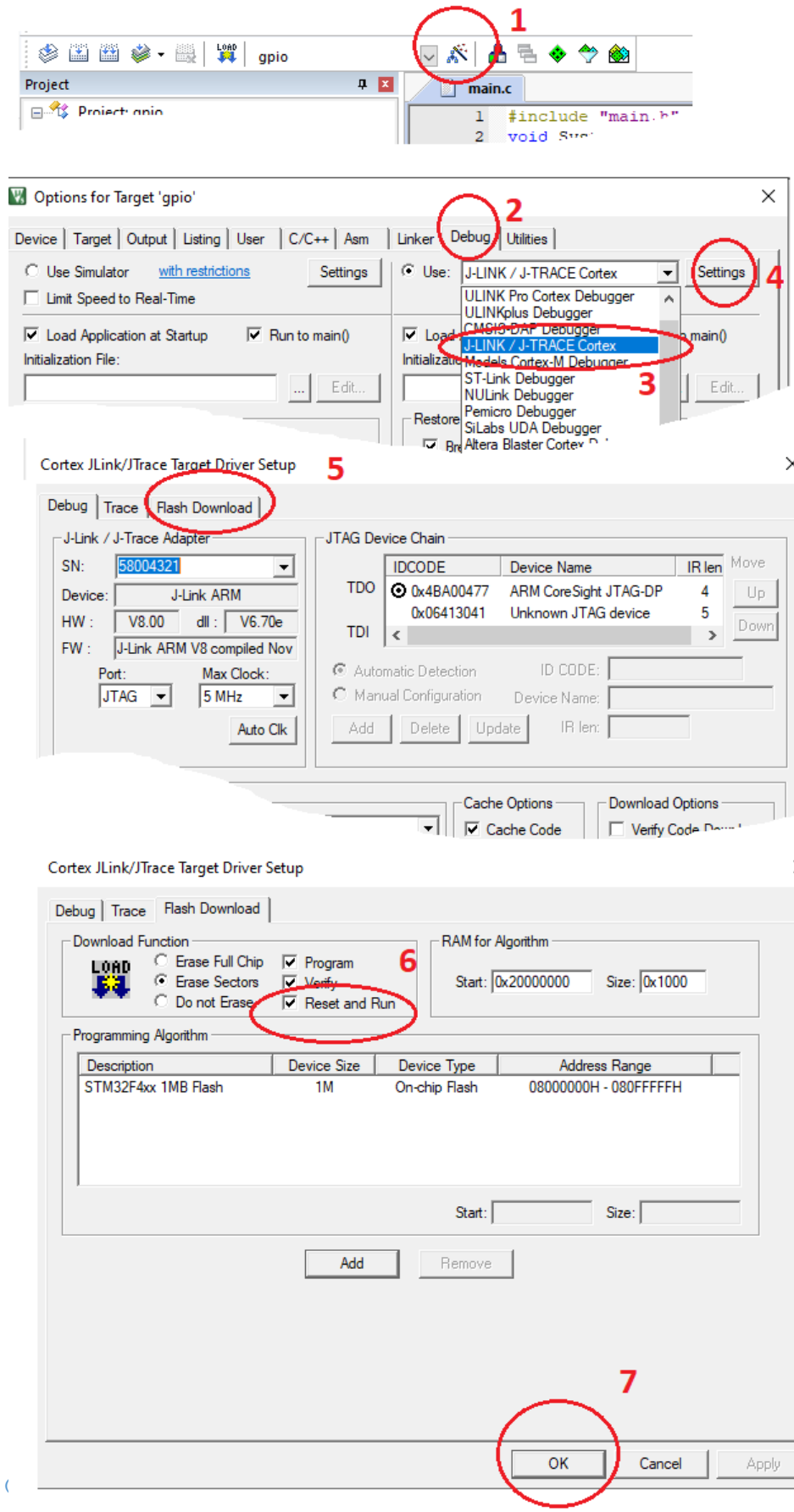
برنامه ای بنویسید که با استفاده از توابع HAL یک نشانگر LED چشمک زن طراحی نمایید.

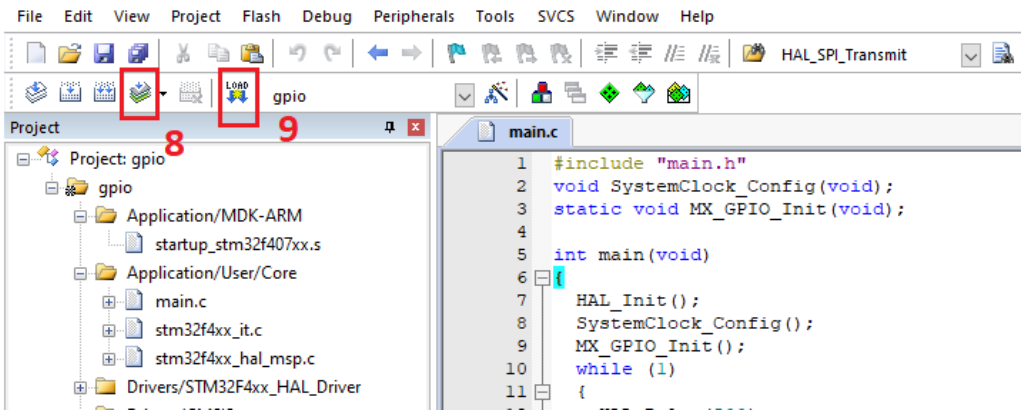
تنظیمات کلاک و پایه مورد نظر را در محیط STM32cubeMX انجام دهید و پروژه ایجاد شده را در محیط Keil ملاحظه نمایید. در برنامه 7-2 کدهای مورد نظر آورده شده است.

```
#include "main.h"
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  while (1)
  {
          HAL_Delay(500);
                  HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_1);

  }
}
```

برنامه 7-2: کدهای تهیه شده برای چشمک زنLED

برای اجرای برنامه در میکروکنترلر پروسه نشان داده شده در شکل 7-1 را دنبال نمایید.


1
gpio
Project
main.c
#include "main.h"
void Sys
Options for Target 'gpio'
2
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Use Simulator
with restrictions
Settings
Limit Speed to Real-Time
Use:
J-LINK / J-TRACE Cortex
Settings
4
ULINK Pro Cortex Debugger
ULINKplus Debugger
CMSIS-DAP Debugger
J-LINK / J-TRACE Cortex
Models Cortex-M Debugger
ST-Link Debugger
NULink Debugger
Pemicro Debugger
SiLabs UDA Debugger
Altera Blaster Cortex
3
Load Application at Startup
Run to main()
Initialization File:
Edit...
Restore
Cortex JLink/JTrace Target Driver Setup
5
Debug
Trace
Flash Download
J-Link / J-Trace Adapter
SN:
58004321
Device:
J-Link ARM
HW :
V8.00
dll :
V6.70e
FW :
J-Link ARM V8 compiled Nov
Port:
JTAG
Max Clock:
5 MHz
Auto Clk
JTAG Device Chain
IDCODE
Device Name
IR len
Move
TDO
TDI
0x4BA00477
ARM CoreSight JTAG-DP
4
0x06413041
Unknown JTAG device
5
Up
Down
Automatic Detection
Manual Configuration
ID CODE:
Device Name:
IR len:
Add
Delete
Update
Cache Options
Cache Code
Download Options
Verify Code
Cortex JLink/JTrace Target Driver Setup
Debug
Trace
Flash Download
Download Function
LOAD
Erase Full Chip
Erase Sectors
Do not Erase
Program
Verify
Reset and Run
6
RAM for Algorithm
Start:
0x20000000
Size:
0x1000
Programming Algorithm
Description
Device Size
Device Type
Address Range
STM32F4xx 1MB Flash
1M
On-chip Flash
08000000H - 080FFFFFH
Start:
Size:
Add
Remove
7
OK
Cancel
Apply

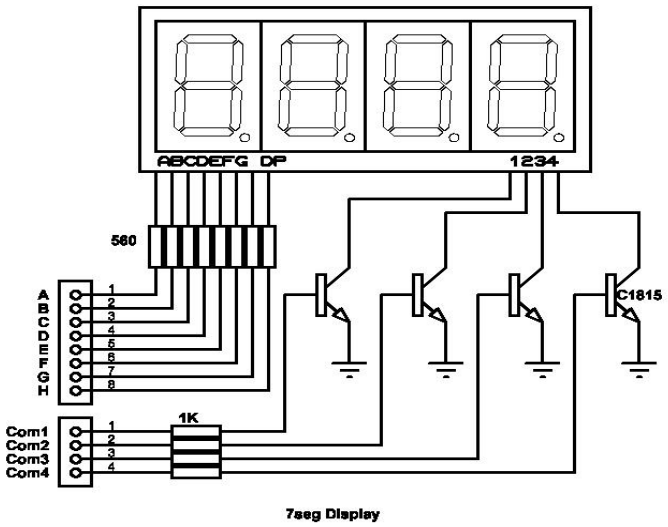


شکل 7-1: نحوه پروگرام کردن برنامه روی میکروکنترلر

در ادامه برای صحت عملکرد برنامه میتوانید خروجی پایه مورد نظر را به LED ها روی پکیج آموزشی متصل نمایید.

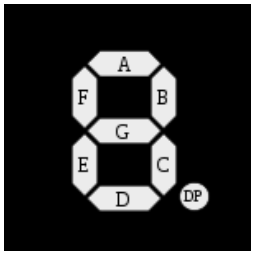
7.7.2 پروژه شمارنده چهار رقمی روی سون سگمنت

این نمایشگر ، برای نمایش اعداد و بعضی از حروف لاتین به کار می‌رود. مطابق شکل 7-2 بر روی بورد آموزشی یک نمایشگر چهار رقمی وجود دارد که در آن برای هر 7-Segment یک پایه‌ی Enable تعبیه شده است. علاوه بر آن 8 پایه‌ی داده به نام‌های A تا H نیز وجود داردکه برای هر چهار 7-Segment مشترک هستند.



شکل 7-2 نحوه اتصال پایه‌های 7-Segment

یک کردن هر کدام از پایه‌های داده، موجب روشن شدن Segment متناظر با آن خواهد شد. بنابراین برای نمایش یک عدد خاص، بر روی هر 7-Segment باید داده‌ی مناسب را بر روی پایه‌های داده ارسال کرده و پایه Enable آن را یک کرد. چنان‌چه این کار به صورت متناوب و با فرکانس مناسب انجام شود، می‌توان به صورت همزمان اعداد چهار رقمی‌دلخواه را روی چهار 7-Segment رؤیت کرد. با توجه به شکل 7-3 برای نمایش عدد '1' باید به Segmentهای B و C مقدار 1 منطقی اعمال کرد.



شکل 7-3 نامگذاری بخش‌های 7-Segment

داده ی یک از ارقام 0 تا 9 در آرایه ذیل مرتب شده است.

```
char digit[]={0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
```