MODUL PRAKTIKUM

PEMROGRAMAN KONTROLER

VI231420

VI231631

Dapartemen Teknik Instrumentasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
2025

DAFTAR ISI

PENDAHULUAN

Praktikum Pemrograman Kontroler merupakan salah satu kegiatan pembelajaran berbasis laboratorium yang bertujuan untuk memperkenalkan serta melatih mahasiswa dalam mengaplikasikan konsep-konsep dasar hingga lanjutan pada sistem tertanam (embedded system) berbasis mikrokontroler STM32. Praktikum Pemrograman Kontroler ini dirancang untuk memberikan pemahaman praktis kepada mahasiswa mengenai penggunaan mikrokontroler STM32 dalam berbagai skenario pengendalian sistem tertanam. Melalui serangkaian kegiatan praktik langsung, mahasiswa diajak untuk mengenal dan menguasai berbagai fitur dan periferal penting seperti GPIO, timer, interrupt, komunikasi serial, konversi sinyal analog-digital, hingga pengelolaan multitasking menggunakan sistem operasi real-time. Seluruh kegiatan praktikum disusun secara bertahap agar membentuk pemahaman yang menyeluruh dan aplikatif terhadap pengembangan sistem berbasis mikrokontroler. Dengan memanfaatkan tools modern seperti STM32CubeMX, Keil µVision, dan STM32CubeIDE, mahasiswa akan dilatih untuk merancang, memprogram, serta menguji sistem kendali secara langsung pada perangkat keras. Praktikum ini tidak hanya menekankan penguasaan teori, tetapi juga mendorong kemampuan implementasi, analisis, dan pemecahan masalah secara nyata, sehingga menjadi bekal penting bagi mahasiswa dalam menghadapi tantangan teknologi di dunia industri maupun riset sistem tertanam.

Aturan Kerja Laboratorium Instrumentasi Dan Kontrol

Peraturan ini disusun sebagai acuan untuk seluruh aktivitas operasional dan layanan di Laboratorium Instrumentasi dan Kontrol, Departemen Teknik Instrumentasi, Fakultas Vokasi-ITS. Setiap pengguna laboratorium wajib mengikuti dan melaksanakan ketentuan ini selama berada di laboratorium.

1. Gunakan APD
2. Perhatikan Tanda Bahaya
3. Penuhi Safety Funciton
4. Dilarang menjalankan alat laboratorium tanpa izin/pengawasan
5. Jaga kebersihan ruang kerja
6. Hati-hati dengan barang-barang pecah belah dan mudah terbakar
7. Jangan tinggalkan alat yang sedang berjalan tanpa pengawasan
8. Laksanakan kegiatan dengan sesuai prosedur yang berlaku
9. Mengembalikan peralatan dan bahan ke tempat semula

Aturan Keamanan Laboratorium Instrumentasi Dan Kontrol

Untuk memastikan keamanan selama beraktivitas di laboratorium, setiap pengguna laboratorium wajib mematuhi ketentuan berikut:

1. Wajib melaporkan segala bentuk kejadian kecelakaan, cedera, atau kerusakan alat harus segera dilaporkan kepada teknisi, asisten, atau koordinator laboratorium.
2. Wajib mengetahui lokasi fasilitas keselamatan/safety tools seperti kotak P3K, eye wash, safety shower, alat pemadam api ringan, spill kit, dsb.
3. Wajib memahami prosedur penggunaan alat sebelum mengoperasikannya.

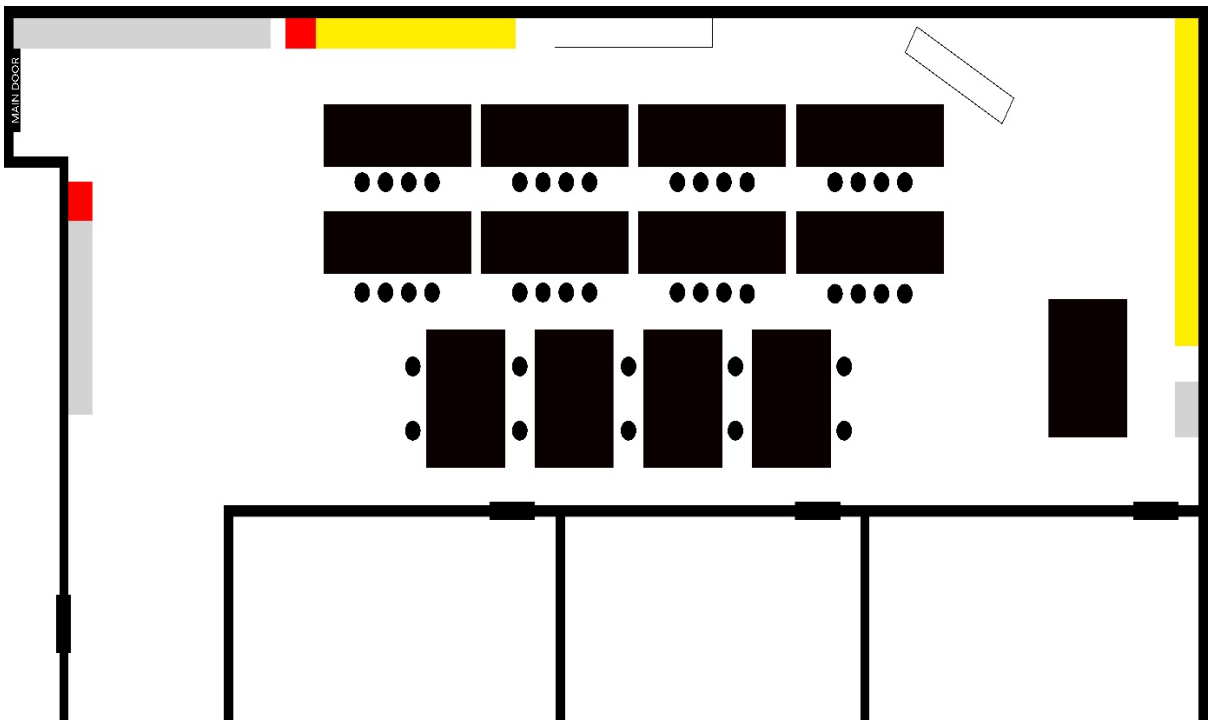
4. Wajib mengenakan pakaian yang aman dan alas kaki tertutup selama berada di ruang laboratorium.
5. Wajib menggunakan Alat Pelindung Diri (APD) sesuai kebutuhan eksperimen.
6. Wajib mengikat rambut agar tidak mengganggu aktivitas eksperimen bagi pengguna laboratorium dengan rambut panjang.
7. Dilarang melakukan aktivitas bercanda, bermain-main, atau tidur di dalam area laboratorium.
8. Dilarang mengonsumsi makanan dan minuman selama kegiatan di dalam laboratorium.
9. Dilarang menggunakan peralatan laboratorium tanpa izin atau sepengetahuan teknisi/koordinator.
10. Dilarang membuang limbah sembarangan. buanglah limbah pada tempat yang sesuai dan telah ditentukan.
11. Pengguna laboratorium yang memakai alat untuk jangka waktu lama diwajibkan meninggalkan penanda/pengingat/peringatan sebagai informasi bagi pengguna lain.
12. Bekerjalah dengan sikap tenang, berhati-hati, dan selalu waspada selama melakukan eksperimen.


PANDUAN
BERKEGIATAN DI LABORATORIUM
1. GUNAKAN APD
2. PERHATIKAN TANDA BAHAYA
3. PENUHI SAFETY FUNCTION
4. DILARANG MENJALANKAN ALAT LABORATORIUM TANPA IZIN / PENGAWASAN
5. JAGA KEBERSIHAN RUANG KERJA
6. HATI-HATI DENGAN BARANG-BARANG PECAH BELAH DAN MUDAH TERBAKAR
7. JANGAN TINGGALKAN ALAT YANG SEDANG BERJALAN TANPA PENGAWASAN
8. LAKSANAKAN KEGIATAN SESUAI DENGAN PROSEDUR YANG BERLAKU
9. MENGEMBALIKAN PERALATAN DAN BAHAN KE TEMPAT SEMULA
SOP

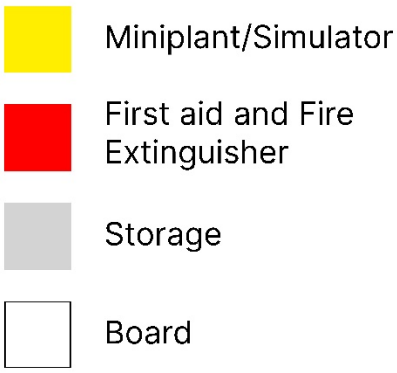
Sanksi Pelanggar Aturan

Pengguna laboratorium yang melanggar aturan ini akan dikenakan sanksi berupa pencabutan hak akses penggunaan laboratorium.

Denah Laboratorium Instrumentasi Dan Kontrol



Keterangan



Prosedure Kedatangan

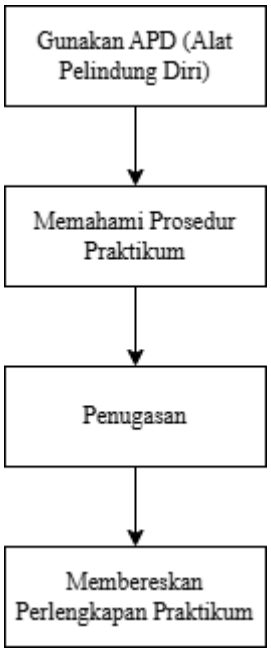
Kebutuhan Saat di Lab IC

1. Berada dalam kondisi kesehatan yang optimal. Urungkan niat untuk datang ke laboratorium jika merasa tidak sehat, beristirahatlah di rumah dan/atau periksakan diri ke dokter terdekat.
2. Membawa keperluan praktikum yang telah ditentukan

3. Mengikuti safety briefing yang diberikan oleh asistem laboratorium dan/atau laboran dengan cermat. Seluruh praktikan WAJIB mengikuti safety briefing sebelum melakukan praktikum. Asisten praktikum wajib memastikan seluruh praktikan sudah mengikuti safety briefing sebelum melaksanakan praktikum

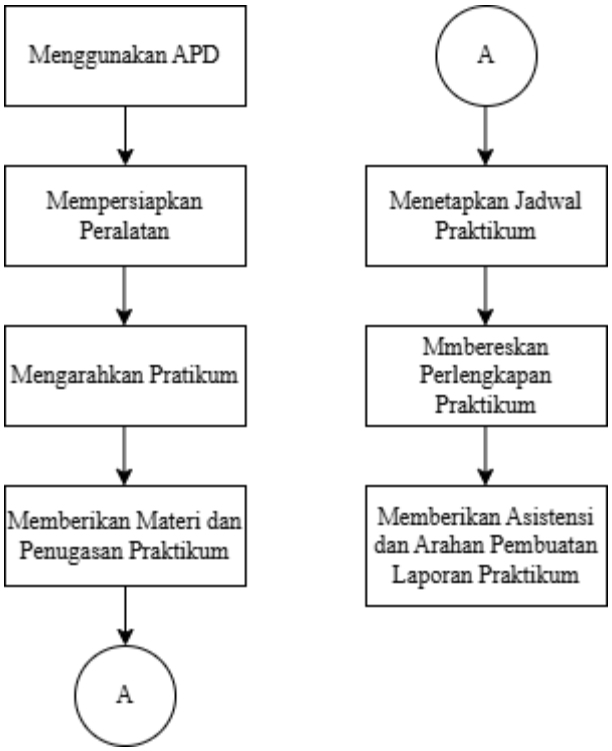
Prosedur Kedatangan Di Lab IC

A. Untuk Praktikan



1. Gunakan APD yang memiliki kondisi masih layak pakai dengan benar. Tanyakan kepada asisten laboratorium/laboran untuk tempat penyimpanan APD berada.
2. Pahami prosedur praktikum yang digunakan sebelum melakukan praktikum demi keamanan pelaksanaan praktikum
3. Saat menuju area praktikum, pastikan alat dan bahan praktikum telah lengkap tersedia. Jika alat dan bahan praktikum belum tersedia, segera tanyakan kepada asisten laboratorium/laboran untuk tempat penyimpanan alat dan bahan praktikum berada
4. Laksanakan praktikum dengan cermat, disiplin dan waspada. Patuhi aturan yang diberikan demi keamanan pelaksanaan praktikum
5. Dengarkan arahan/penugasan dari asisten laboratorium/laboran dengan cermat sehingga dapat meningkatkan produktivitas saat pelaksanaan asistensi praktikum
6. Bersihkan area praktikum ketika selesai melakukan praktikum dengan hati-hati

B. Untuk Asisten Laboratorium



1. Sediakan dan gunakan APD yang memiliki kondisi masih layak pakai dengan benar.
2. Pastikan alat dan bahan yang digunakan untuk praktikum dapat digunakan.
3. Berikan arahan dan dampingan saat melaksanakan praktikum dengan benar dan disiplin.
4. Berikan penjelasan mengenai materi praktikum/penugasan pasca melakukan praktikum dengan jelas.
5. Sebelum mengakhiri praktikum, tetapkan jadwal kapan perlu melakukan asistensi data.
6. Setelah praktikum selesai, bersihkan dan rapikan alat serta bahan praktikum. Pastikan alat tidak mengalami kerusakan dan bahan praktikum kembali ke tempat penyimpanan yang tepat.
7. Berikan arahan yang jelas saat melakukan asistensi dan pembuatan laporan.

PROSEDUR PRAKTIKUM

Pengetahuan Penunjang

1. STM32CubeMX

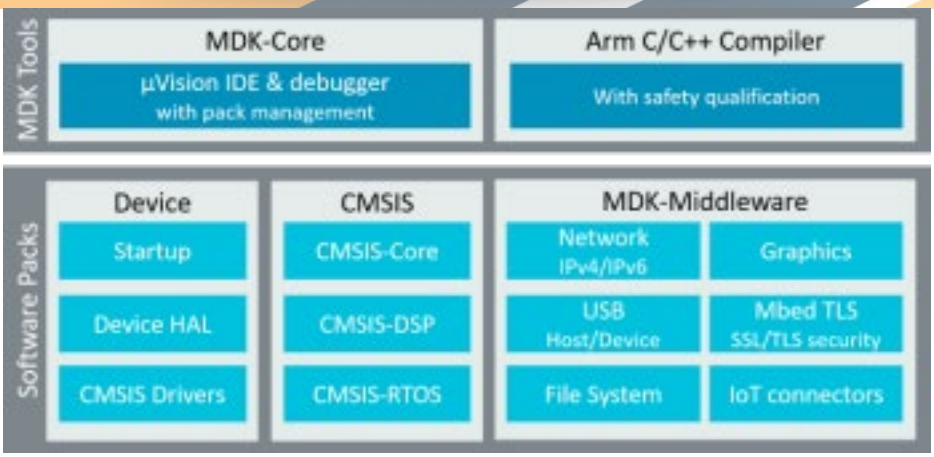
STM32CubeMX adalah perangkat lunak yang dikembangkan oleh STMicroelectronics untuk membantu dalam konfigurasi dan pengkodean awal mikrokontroler STM32. Perangkat ini memungkinkan pengguna untuk secara visual mengatur pin, periferal, clock, dan middleware melalui antarmuka grafis sebelum mengekspor kode yang dapat digunakan langsung dalam STM32CubeIDE atau lingkungan pengembangan lainnya.

Fitur	Deskripsi
Konfigurasi Periferal	Memudahkan konfigurasi GPIO, UART, SPI, I2C, ADC, DAC, TIM, dll.
Manajemen Clock	Memberikan tampilan grafis untuk mengonfigurasi sistem clock dengan presisi.
Middleware Support	Mendukung FreeRTOS, USB, TCP/IP, dan stack lainnya.
Generator Kode	Menghasilkan kode C/C++ untuk inisialisasi mikrokontroler.
Pembuatan File HAL	Mempermudah pemrograman dengan abstraksi perangkat keras STM32.
Simulasi Daya	Memberikan estimasi konsumsi daya berdasarkan konfigurasi perangkat keras.

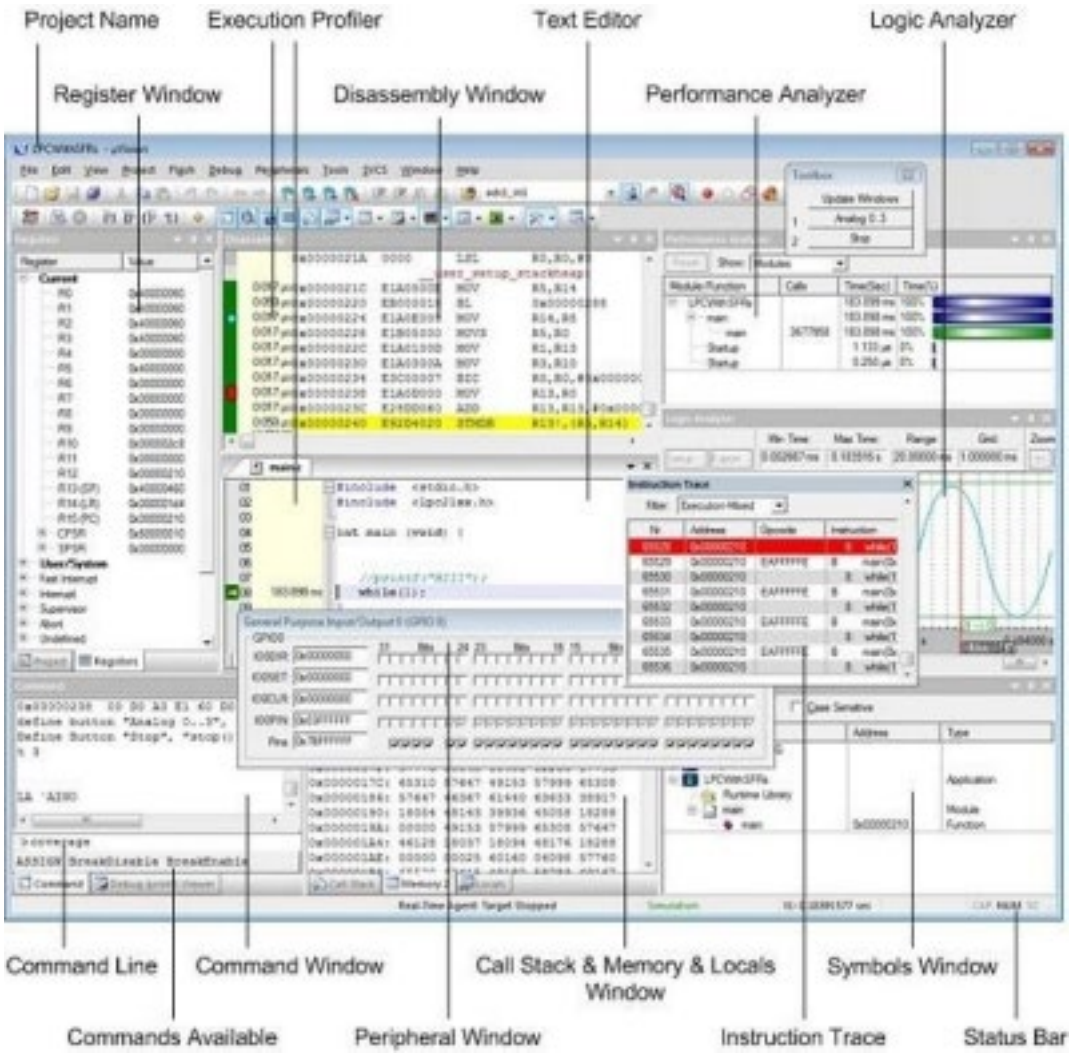
Table 1. Main fiture STM32CubeMX

2. KEIL µVision ARM

Keil MDK adalah pengembangan perangkat lunak terlengkap untuk berbagai keluarga mikrokontroler STM32 dan menyediakan semua yang Anda butuhkan untuk membuat, membangun, dan men-debug aplikasi tertanam. MDK mencakup Arm Compiler asli dan Keil uVision IDE/Debugger yang mudah digunakan yang terhubung ke STM32CubeMX dan *Software Packs*.



Gambar 1. MDK Tools & Software Packs

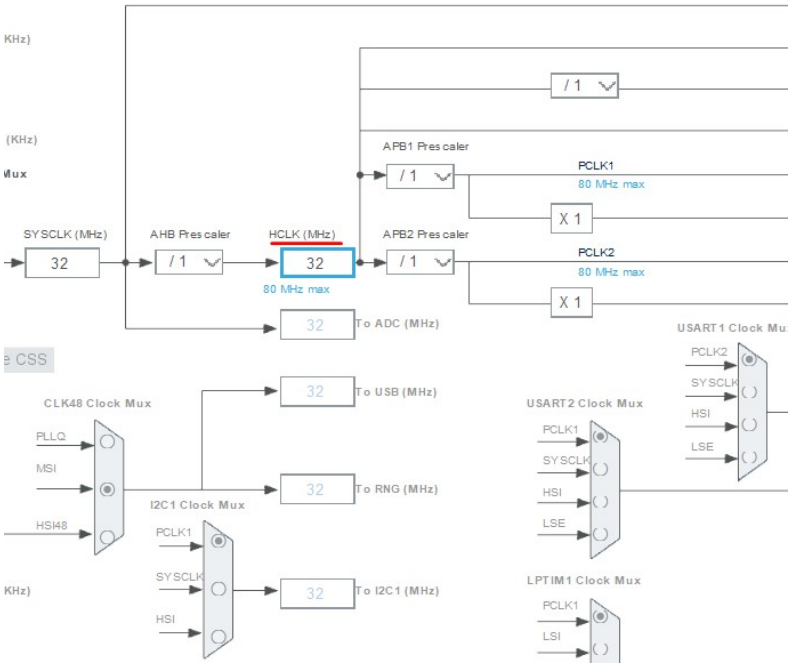


Gambar 2. Fitur Keil MDK

3. STM32 Timer

Timer adalah periferal penting dalam mikrokontroler. Timer memungkinkan pelacakan waktu, menghasilkan sinyal digital, menjalankan kode secara periodik, dan

banyak fungsi lainnya. Sekilas, konsep **clock** dan **timer** mungkin membingungkan, tetapi keduanya adalah hal yang sepenuhnya berbeda. **Clock** adalah bagian fisik yang bertanggung jawab untuk menyinkronkan semua komponen dalam mikrokontroler, sedangkan **timer** adalah periferal yang digunakan bersama periferal lain untuk menjalankan kode secara periodik dan menghasilkan sinyal digital.



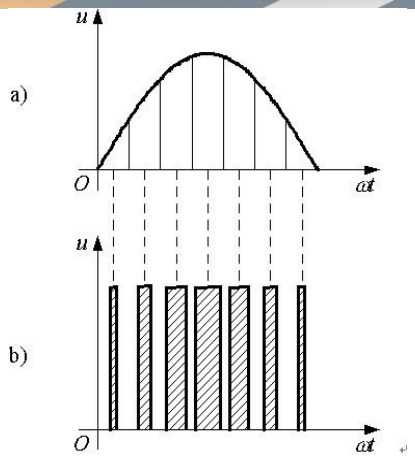
Gambar 3. Contoh Setting Timer STM32

4. Interrup dalam STM32

Interrupt dalam STM32 CubeMX adalah mekanisme yang memungkinkan mikrokontroler menanggapi peristiwa eksternal atau internal tanpa harus terus- menerus memeriksa statusnya (polling). Dengan menggunakan interrupt, mikrokontroler dapat menjalankan tugas lain dan hanya merespons ketika terjadi peristiwa tertentu, seperti timer overflow, input dari sensor, atau komunikasi serial.

5. Pulse Width Modulation

PWM (Pulse Width Modulation) adalah teknik modulasi yang digunakan untuk mengontrol rangkaian analog melalui keluaran digital dari MCU.

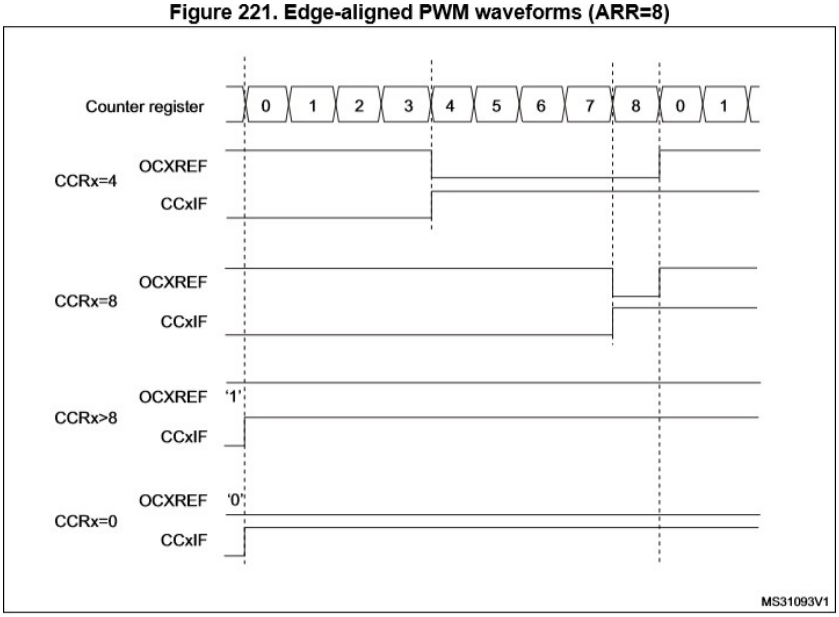


Gambar di atas menunjukkan sinyal PWM. Gambar (b) menunjukkan sinyal digital dari MCU, sedangkan gambar (a) menunjukkan sinyal analog yang dihasilkan ketika keluaran digital dihubungkan ke perangkat daya, seperti motor.

Sebagai contoh, pulsa dengan keluaran PWM pada *duty cycle* 50%, frekuensi 10Hz, dan level tinggi 3.3V, dapat menghasilkan sinyal analog dengan tegangan keluaran sekitar 1.65V.

PWM memiliki dua parameter utama: frekuensi keluaran dan *duty cycle*. *Duty cycle* digunakan untuk mengubah tegangan keluaran sinyal analog. Frekuensi yang lebih tinggi akan menghasilkan sinyal analog yang lebih halus, sedangkan *duty cycle* yang lebih besar akan menghasilkan tegangan analog yang lebih tinggi.

Figure 221. Edge-aligned PWM waveforms (ARR=8)



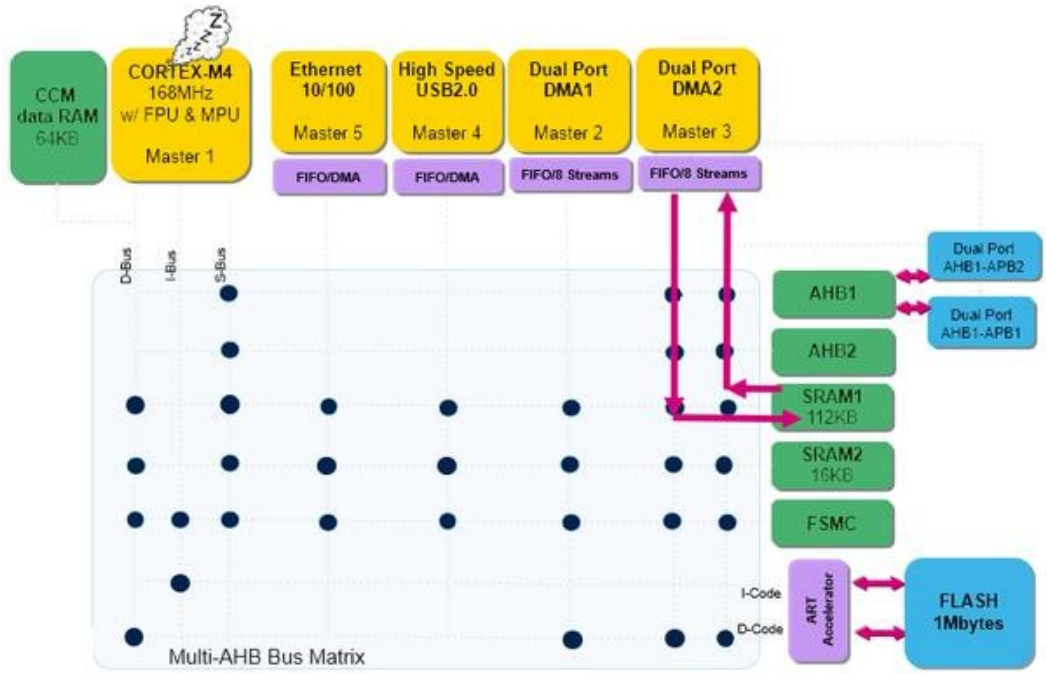
Gambar 4. Contoh *Edge-aligned PWM waveforms* (ARR=8)

6. Direct Memory Access

Direct Memory Access adalah bus master dan peripheral sistem yang menyediakan

transfer data berkecepatan tinggi antara peripheral dan memori, serta dari memori ke memori. Data dapat dipindahkan dengan cepat oleh DMA ***tanpa tindakan CPU apa pun*** , sehingga sumber daya CPU tetap bebas untuk operasi lainnya.

Saluran DMA dapat beroperasi tanpa dipicu oleh permintaan dari peripheral. Mode ini disebut mode memori-ke-memori, dan dimulai oleh perangkat lunak. Mode ini memungkinkan transfer dari satu lokasi alamat ke lokasi lain tanpa permintaan perangkat keras. Setelah saluran dikonfigurasi dan diaktifkan, transfer segera dimulai.



Gambar 5. *Memory to Memory Mode*

7. Analog Digital Converter (ADC)

Analog to Digital Converter (ADC) adalah salah satu periferal penting dalam mikrokontroler STM32 yang berfungsi untuk mengubah sinyal analog menjadi sinyal digital yang dapat diproses oleh unit pemroses pusat (CPU). Sinyal analog adalah sinyal kontinyu, seperti tegangan dari sensor suhu, sensor cahaya, atau potensiometer, yang nilainya dapat berubah-ubah secara halus dalam rentang tertentu. Karena mikrokontroler hanya bisa memahami data dalam bentuk digital (diskrit), maka dibutuhkan ADC untuk melakukan proses konversi tersebut.

Pada mikrokontroler STM32, ADC biasanya memiliki resolusi 12-bit, yang artinya sinyal analog akan dikonversi menjadi angka digital dalam rentang 0 hingga 4095. Nilai ini merepresentasikan besarnya tegangan input relatif terhadap tegangan referensi (biasanya 3.3V). Misalnya, jika tegangan masukan sebesar 1.65V dan referensinya 3.3V, maka nilai digital hasil ADC akan mendekati 2048.

8. Digital Analog Converter (DAC)

Digital to Analog Converter (DAC) adalah periferal pada mikrokontroler STM32 yang berfungsi untuk mengubah nilai digital menjadi sinyal analog berupa tegangan yang kontinyu. DAC digunakan saat sistem digital perlu menghasilkan sinyal analog, misalnya untuk mengontrol aktuator analog, menghasilkan suara/audio, atau menyimulasikan keluaran sensor.

Mikrokontroler STM32 memiliki modul DAC dengan resolusi 12-bit, yang berarti nilai digital 0 akan menghasilkan tegangan mendekati 0V dan nilai maksimum(4095) akan menghasilkan tegangan mendekati tegangan referensi (misalnya 3.3V). Dengan mengubah nilai digital yang dikirim ke DAC, pengguna dapat menghasilkan berbagai bentuk gelombang analog seperti sinus, segitiga, dan persegi.

9. Serial (I2C)

I2C (Inter-Integrated Circuit) merupakan salah satu protokol komunikasi serial yang umum digunakan dalam sistem mikrokontroler, termasuk STM32. I2C dirancang untuk komunikasi antar perangkat dalam jarak pendek menggunakan dua jalur utama, yaitu SDA (Serial Data Line) dan SCL (Serial Clock Line). Tidak seperti UART atau USART yang menggunakan prinsip komunikasi point-to-point, I2C mendukung komunikasi multi-master dan multi-slave, sehingga satu master dapat berkomunikasi dengan banyak perangkat slave menggunakan hanya dua kabel. Pada STM32, komunikasi I2C difasilitasi oleh periferal I2C yang dapat dikonfigurasi melalui kode langsung menggunakan HAL Library atau secara grafis menggunakan STM32CubeMX. Pengguna dapat mengatur kecepatan komunikasi (standard mode 100 kHz, fast mode 400 kHz, atau bahkan high-speed mode hingga 3.4 MHz), alamat slave (7-bit atau 10- bit), serta mode operasi (master atau slave). Salah satu keunggulan I2C adalah efisiensinya dalam penggunaan jalur komunikasi dan kemampuannya untuk menghubungkan banyak perangkat tanpa memerlukan banyak pin tambahan. Hal ini menjadikannya sangat cocok untuk menghubungkan mikrokontroler dengan sensor sensor digital seperti accelerometer, RTC (Real-Time Clock), EEPROM, dan lain sebagainya.

10. FreeRTOS

FreeRTOS (Free Real-Time Operating System) adalah sistem operasi real-time open source yang dirancang untuk digunakan pada mikrokontroler dan sistem embedded. FreeRTOS dikembangkan oleh Richard Barry dan saat ini dikelola oleh Amazon melalui proyek Amazon FreeRTOS. Sistem operasi ini memungkinkan pengembang untuk membuat aplikasi multitasking, di mana berbagai proses atau tugas (task) dapat berjalan secara bersamaan dalam satu sistem mikrokontroler. Hal ini sangat berguna untuk sistem yang memiliki banyak fungsi yang harus berjalan secara paralel, seperti membaca sensor, mengendalikan output, berkomunikasi melalui serial, dan melakukan logging data secara bersamaan.

Dalam sistem tanpa RTOS (bare-metal), semua proses biasanya dijalankan dalam satu loop besar (infinite loop) dan dikelola secara manual. Namun, dengan FreeRTOS,

pengembang dapat membagi aplikasi menjadi beberapa task yang independen, dan sistem operasi akan menangani penjadwalan serta prioritas eksekusi masing-masing task secara otomatis menggunakan algoritma preemptive atau cooperative scheduling. FreeRTOS juga menyediakan fitur-fitur penting lainnya seperti queue, semaphore, mutex, dan timers, yang sangat berguna dalam mengelola komunikasi antar task dan sinkronisasi data. Pada STM32, FreeRTOS sangat mudah diintegrasikan melalui STM32CubeMX, di mana pengguna cukup mengaktifkan middleware FreeRTOS dan mengatur task, stack size, serta prioritasnya melalui antarmuka grafis. Setelah konfigurasi selesai, kode inisialisasi dan struktur task akan otomatis dihasilkan, sehingga pengguna bisa langsung mengisi fungsi-fungsi spesifik untuk masing-masing task.

Prosedur Praktikum 1 - Kontrol LED Flip-Flop dengan Push Button Menggunakan STM32

Overview

Praktikum ini berfokus pada pemrograman dasar mikrokontroler STM32 menggunakan STM32CubeMX dan Keil µVision ARM untuk mengontrol LED dan push button. Praktikan dapat mempelajari cara membuat LED berkedip, menyalakan LED dengan tombol, menggunakan fungsi dalam program, serta melakukan debugging untuk memantau kondisi tombol secara real-time.

Tahap Persiapan Praktikum

Sebelum kegiatan praktikum dilakukan, praktikan dapat mempersiapkan alat dan bahan sebagai berikut:

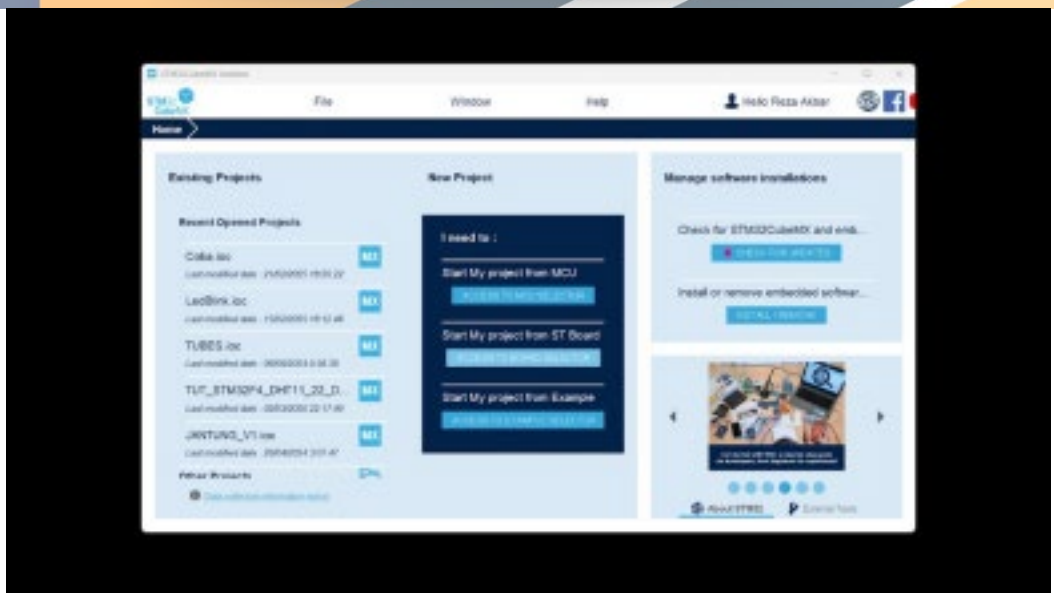
a. PC/ Laptop untuk melakukan praktikum
b. Software STM32CubeMX
c. Software KEIL µVision ARM
d. STM32 Nucleo F446re

Tahap Praktikum

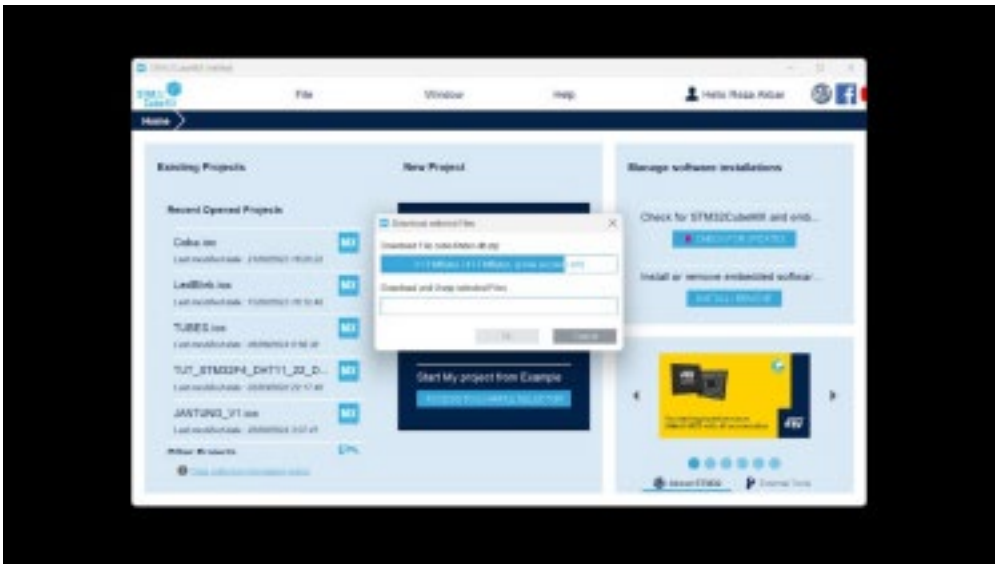
P1 **"Kontrol LED Flip-Flop dengan Push Button Menggunakan STM32"**

1. Buka Program STM32 CubeMX

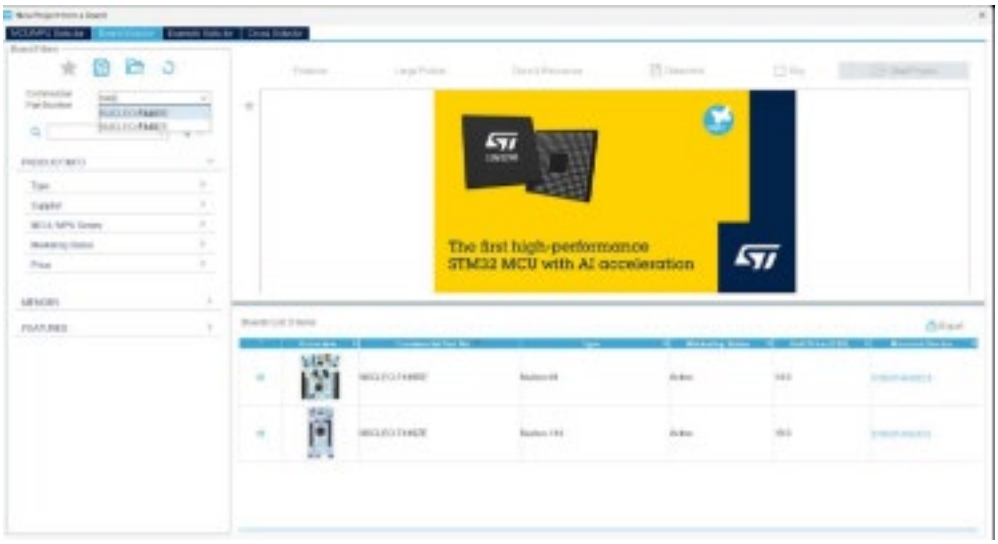
2. Pilih access to board selector



3. Tunggu proses download file Modul Praktikum Pemrograman Kontroler

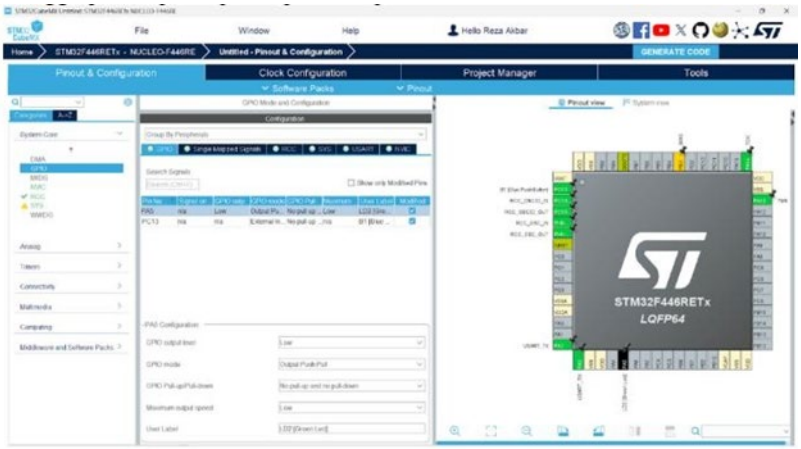


4. Cari board berdasarkan nama dengan type “NUCLEO-F446RE”

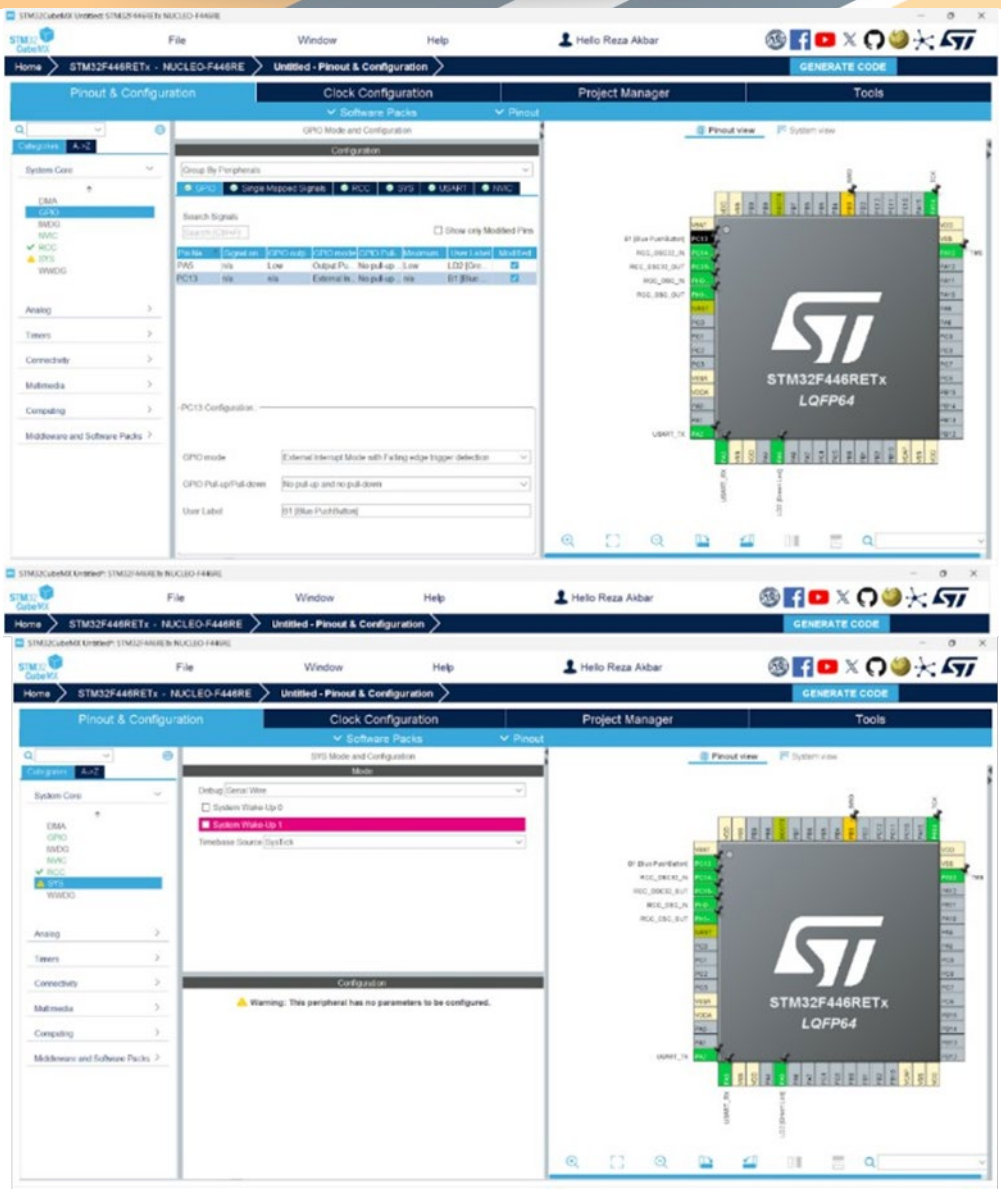


5. Klik “NUCLEO-F446RE” hingga berubah berwarna biru > Klik Start Project > Klik “Yes”

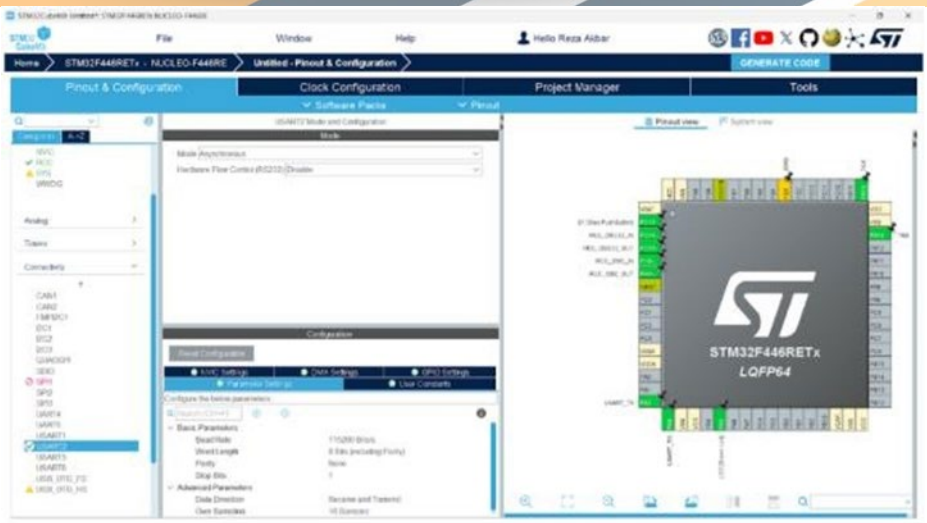
6. Tunggu proses persiapan seperti tampil berikut

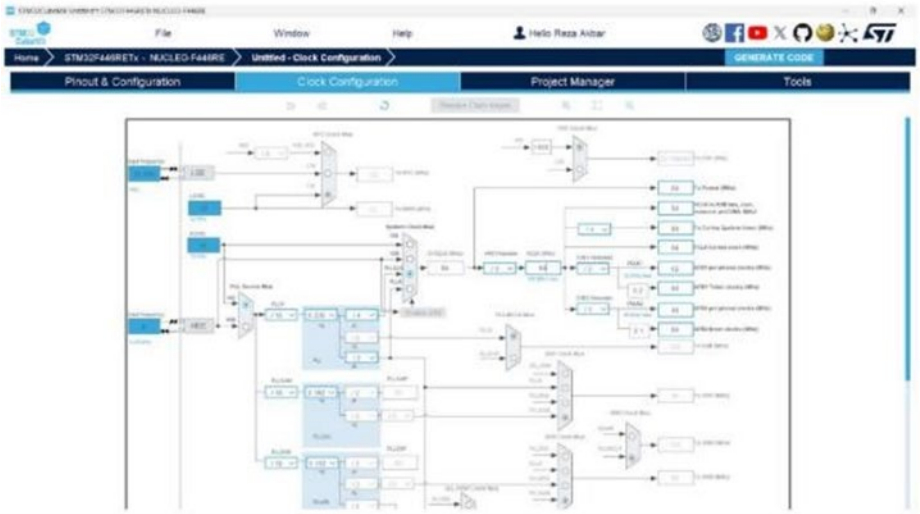


7. Setting GPIO dengan ceklist “PA5” & “PC13”

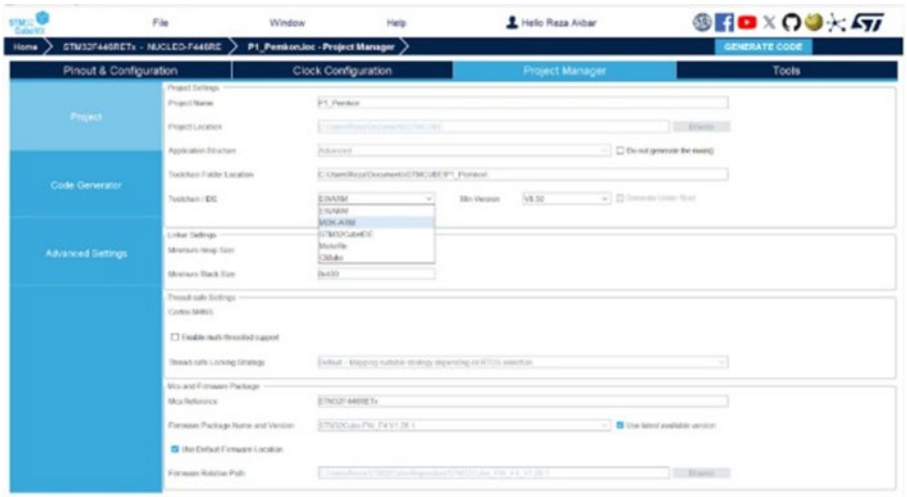


8. Connectivity > “USART2”

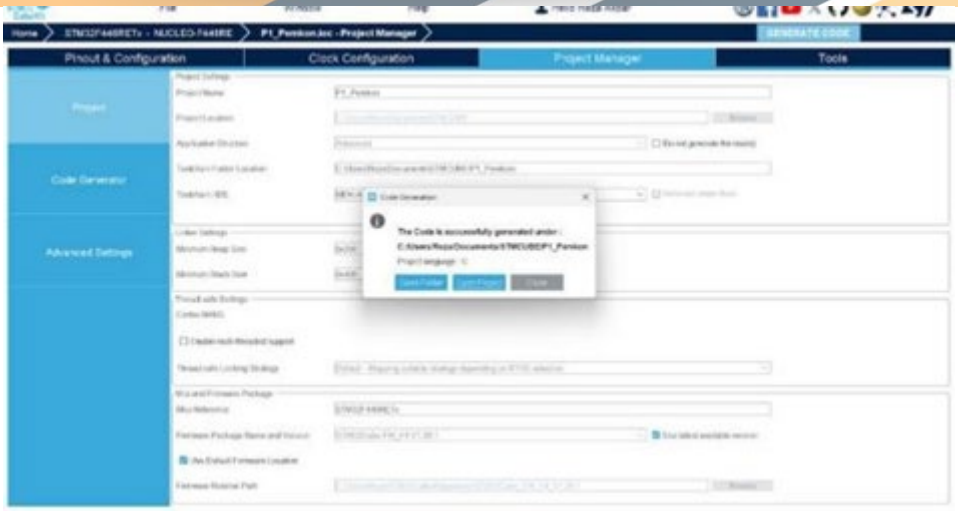




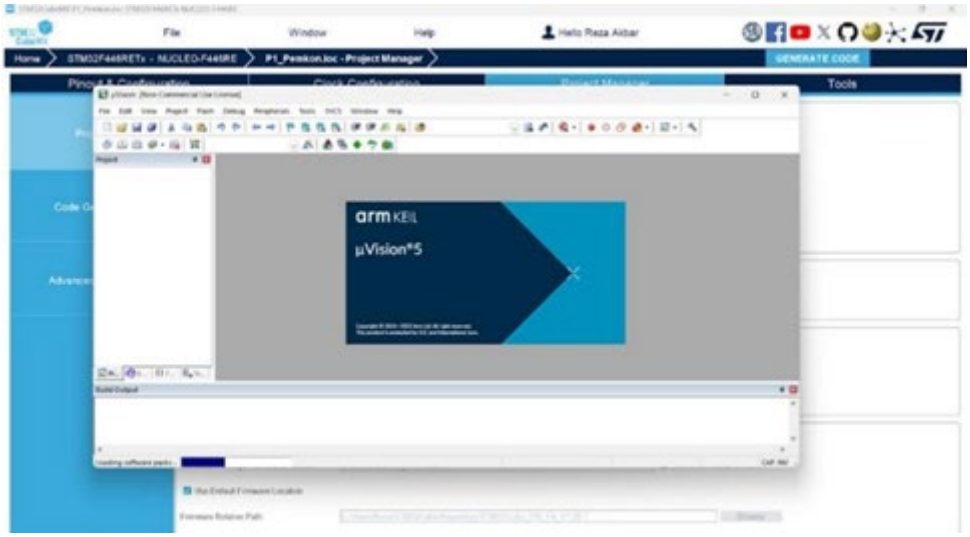
9. Beri nama projek lalu pilih “Toolchain/IDE” > “MDK-ARM”



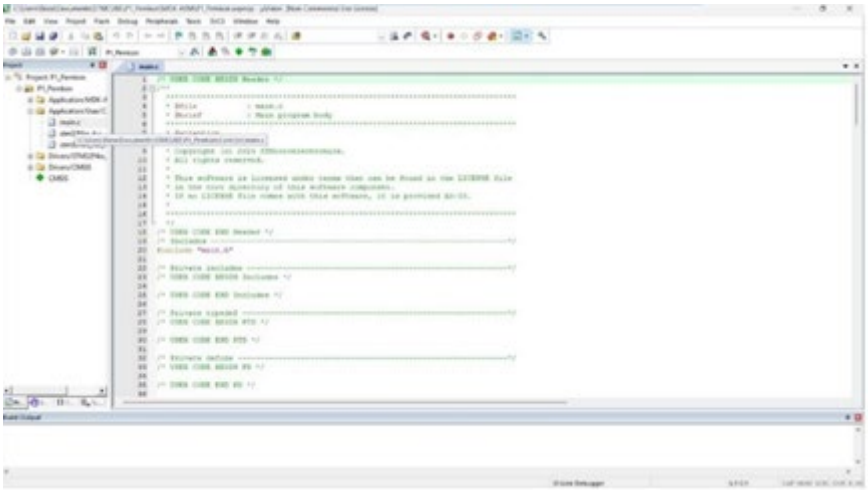
10. Setelah itu klik “Generate Code”, setelah generate code berhasil maka akan muncul pop-up seperti berikut lalu klik “Open Project”



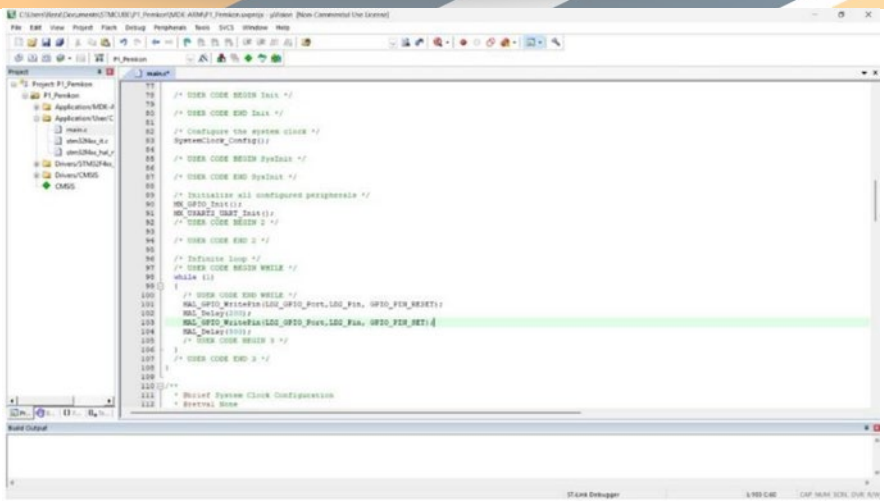
11. Laman otomatis akan membuka “KEIL ARM”



12. Cari file main.c yang berfungsi sebagai main program STM32

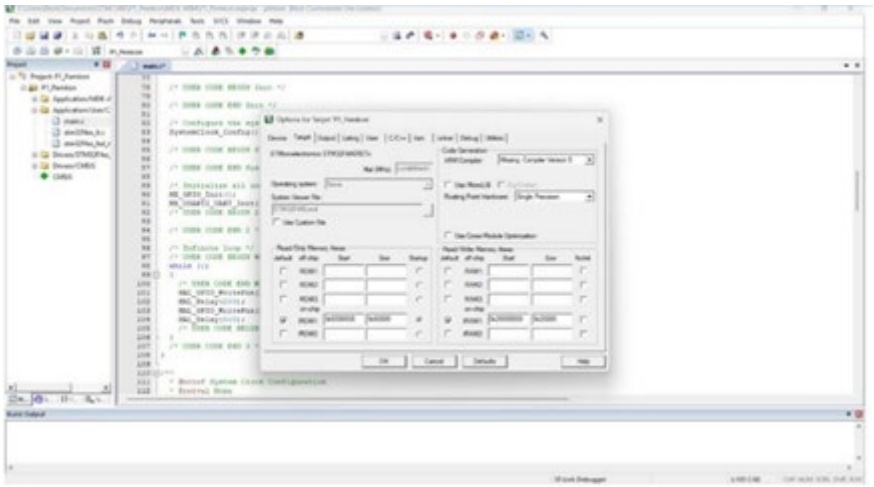


13. Buat main Program Percobaan Pertama LED Blink seperti berikut :

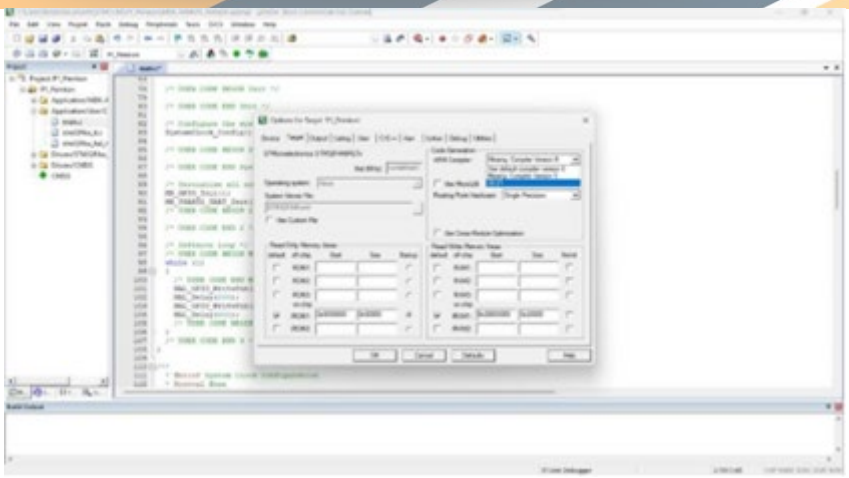


```
        // Blink LED
        while (1)
         {
         /* USER CODE END WHILE */
               HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPI
               O_PIN_RESET); HAL_Delay(200);
               HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN_SET);
               HAL_Delay(500);
         /* USER CODE BEGIN 3 */
}
```

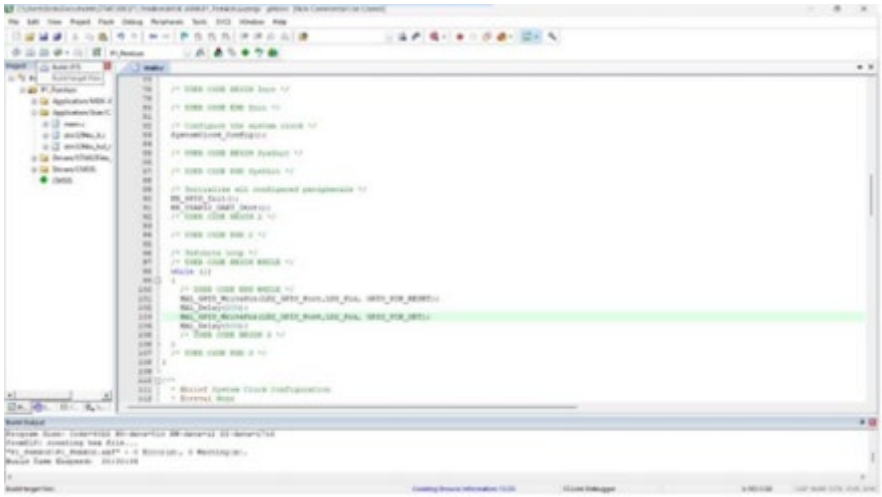
14. Buka option for target untuk memastikan bahwa source code di compile atau dijalankan pada software KEIL, tombol berada di sebelah kanan pilihan select target



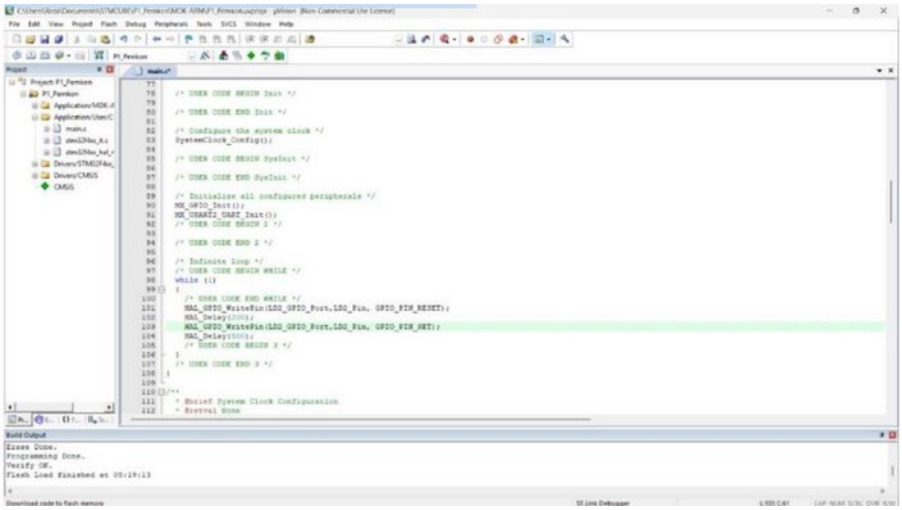
15. Jika ARM Compiler tertulis missing maka ganti ke pilihan yang lain bisa menggunakan “use default compiler version” atau menggunakan sesuai versi KEIL spesifik untuk contoh menggunakan KEIL “Version V6.21” disesuaikan dengan versi KEIL yang telah terpasang pada komputer masing-masing



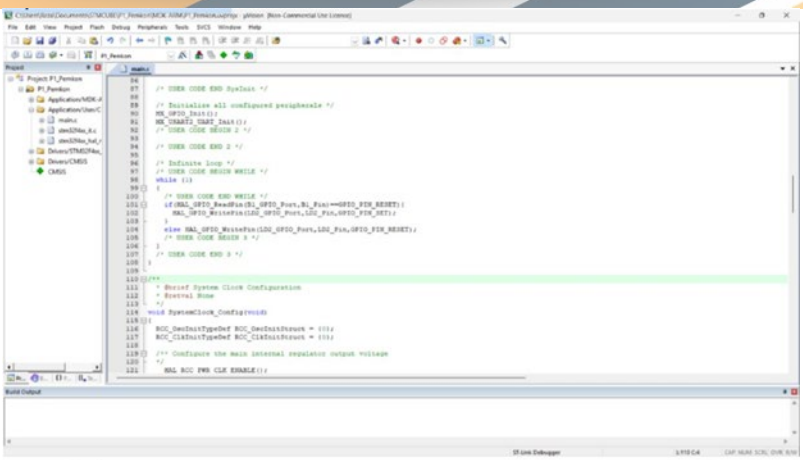
16. Kemudian klik build dan pastikan tidak ada error



17. Setelah dipastikan tidak ada error klik load

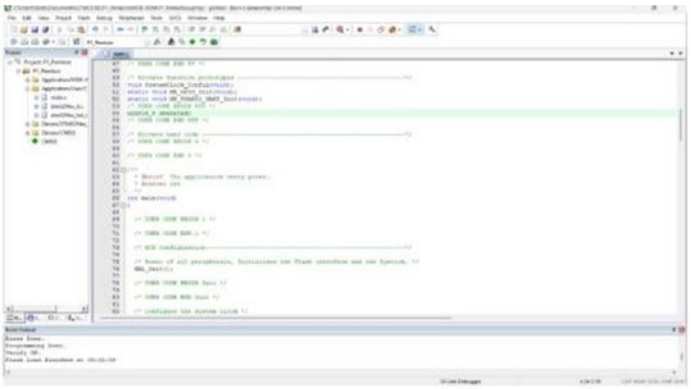


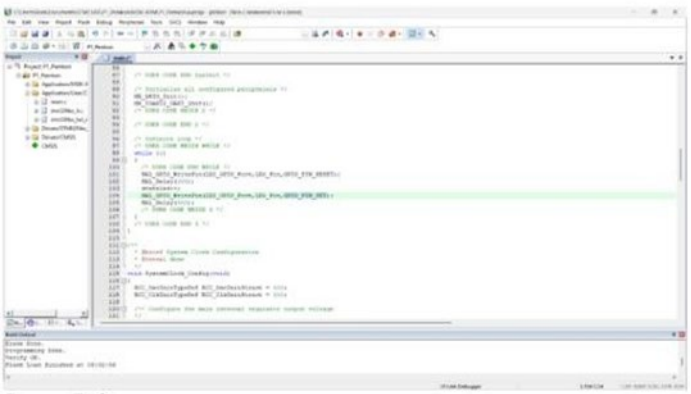
18. Setelah load dilakukan led tidak akan langsung berkedip melainkan tombol reset pada board harus ditekan terlebih dahulu, setelah di tekan baru program akan berjalan sepenuhnya.
19. Lanjut ke percobaan ke-2 yaitu push button, untuk kali ini tidak perlu membuat dari awal main program, lakukan cara nomor 16 hingga 18 kembali lalu ganti code program seperti contoh dibawah



```
        // Button only
        while (1)
         {
         /* USER CODE END WHILE */
        if(HAL_GPIO_ReadPin(B1_GPIO_Port,B1_Pin)==GPIO_PIN_RES
              ET){
              HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN
              _SET);
                    }
              else
        HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN_RESET)
        ;  /* USER CODE BEGIN 3 */
         }
```

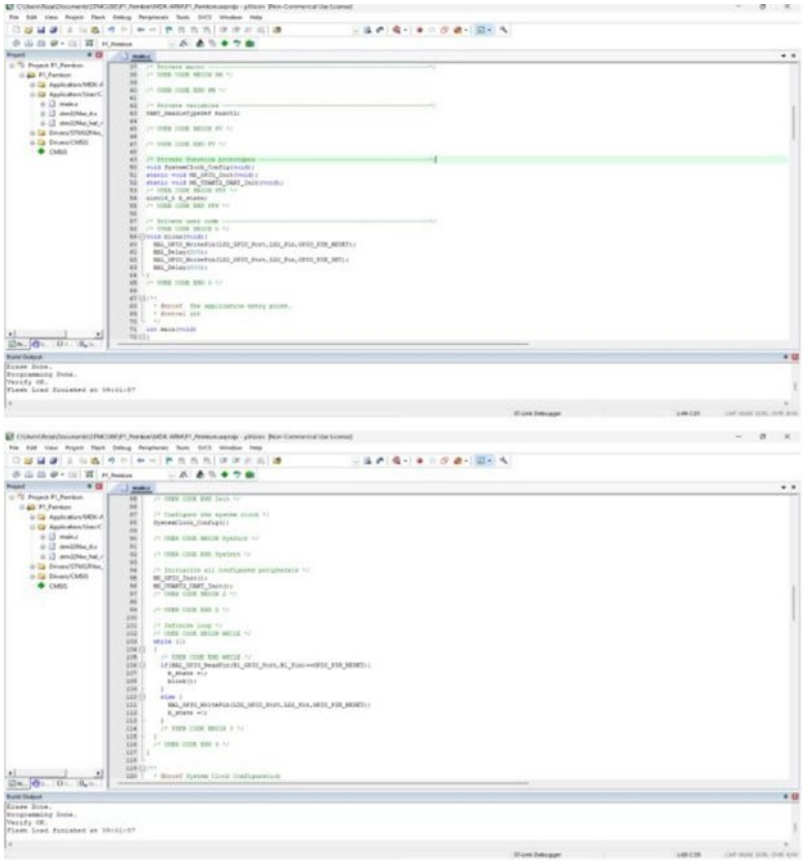
20. Lanjut ke percobaan ke-3 yaitu kita menambahkan fungsi button yang dimana jika button ditekan maka led akan berkedip dan jika button di lepas maka led akan mati. Sebelumnya kita akan membuat function juga jadi di while kita tinggal memanggil function blink tersebut, dan lakukan cara nomor 16 hingga 18 kembali.





```
      // Button with function
      /* USER CODE BEGIN 0 */
      void blink(void){
            HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPI
            O_PIN_RESET); HAL_Delay(200);
            HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,G
            PIO_PIN_SET); HAL_Delay(500);
      }
      /* USER CODE END 0 */

  while (1)
   {
   /* USER CODE END WHILE */
  if(HAL_GPIO_ReadPin(B1_GPIO_Port,B1_Pin)==GPIO_PIN_RESET){   blink();
              }
              else
      HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN_RESET)
      ;  /* USER CODE BEGIN 3 */
         }
```

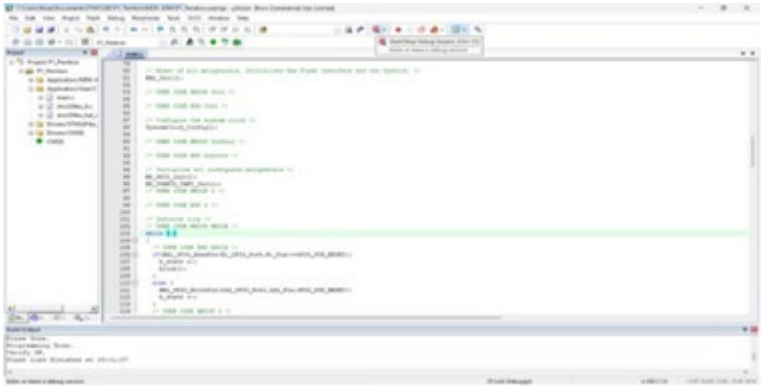
21. Lanjut ke percobaan ke-4 kita akan mencoba melakukan debugging dimana kita akan melihat state dari push button buat variable b_state seperti di bawah dan biarkan function blink, serta tambahkan untuk statement di bagian while seperti gambar dibawah



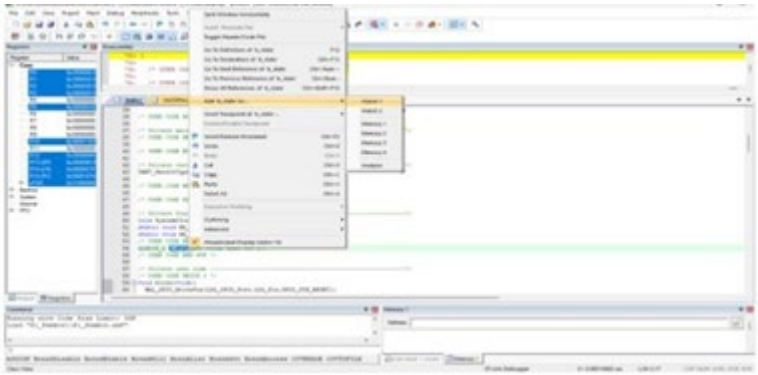
```
// Debug
/* USER CODE BEGIN PFP */ uint16_t b_state;
/* USER CODE END PFP */
/* Private user code ---------------------------------------------*/
/* USER CODE BEGIN 0 */
void blink(void){
```

```
HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN_RESET);
HAL_Delay(200);
HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN_SET); HAL_Delay(500);
}
/* USER CODE END 0 */

while (1)
 {
 /* USER CODE END WHILE */
if(HAL_GPIO_ReadPin(B1_GPIO_Port,B1_Pin)==GPIO_PIN_RESET){ b_state =1;
blink();
}
else {
HAL_GPIO_WritePin(LD2_GPIO_Port,LD2_Pin,GPIO_PIN_RESET); b_state =0;
}
```

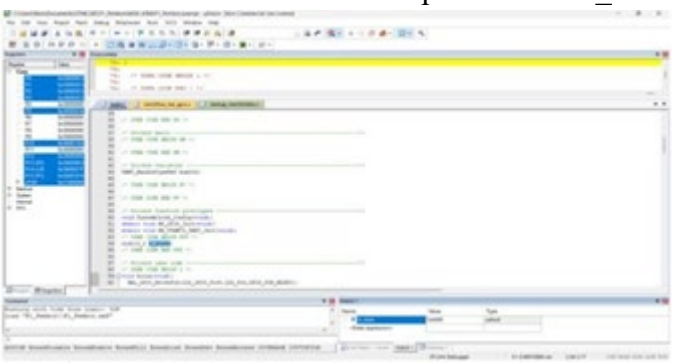
22. Lakukan kembali cara nomor 16 hingga 17, kemudian klik bagian debugging seperti di Langkah bawah



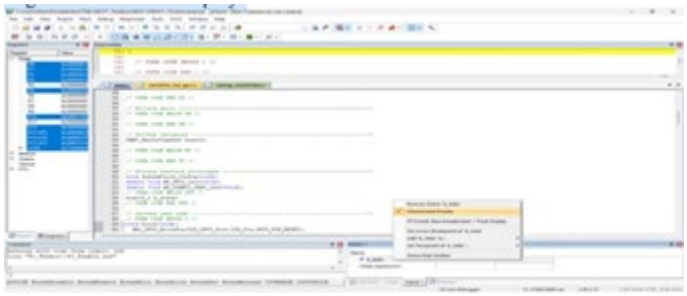
23. Add variable b_state ke watch 1 dengan blok b_state > klik kanan lalu pilih seperti yang ada di gambar bawah



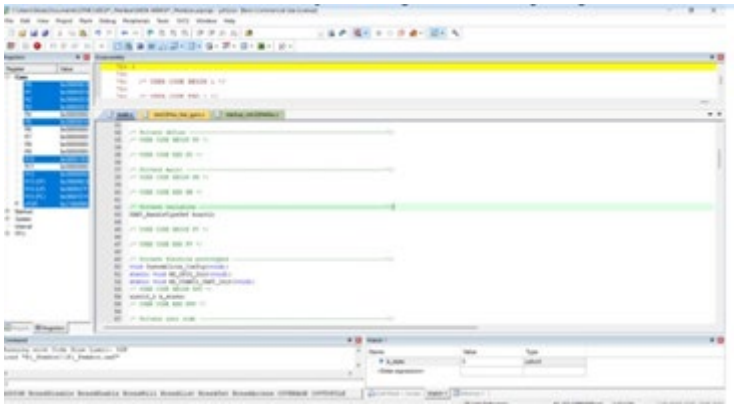
24. Window watch 1 akan muncul di bawah dan terdapat variable b_state yang terisi



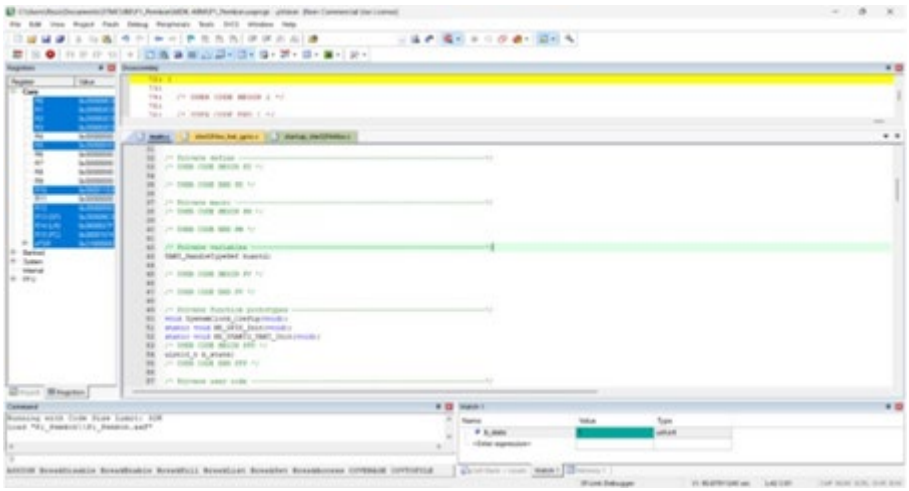
25. Kemudian kita juga dapat mengubah value yang awalnya hexadesimal ke bilangan decimal, biasa dengan klik kanan b_state yang ada di watch kemudian unchecklist bagian hexadesimal display



26. Setelah itu maka akan berubah menjadi bilangan desimal seperti gambar di bawah



27. Untuk running maka tekan tombol reset dan coba tekan tombol warna biru yang ada pada board untuk melihat nilai b_state berubah atau tidak, jika program telah sesuai maka jika tombol biru ditekan maka value b_state akan berubah.



Tahap Pasca Praktikum

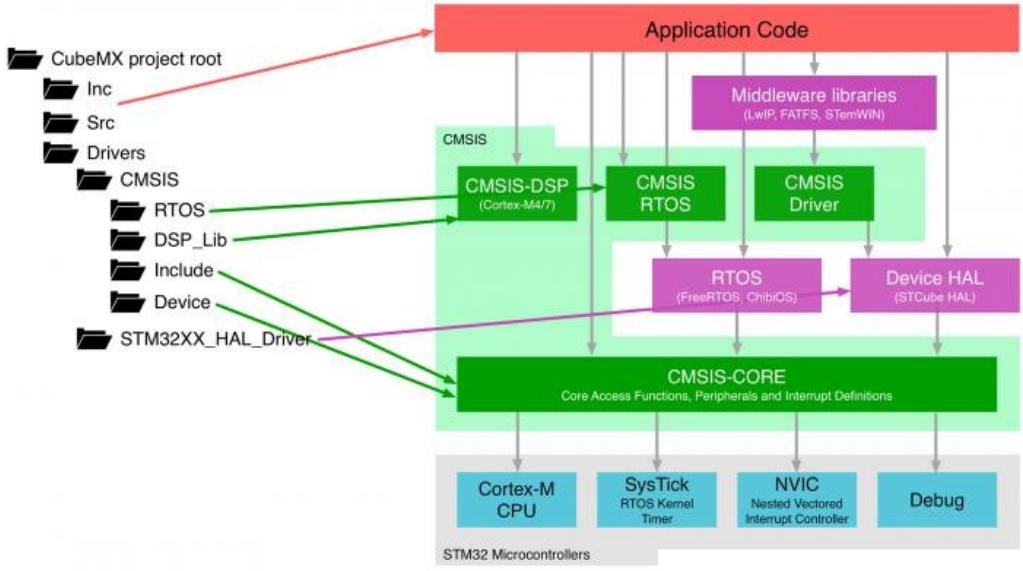
1. Coba analisis perbedaan program blink atau led kedip diatas dengan program blink yang ada di bawah ini apa perbedaannya dan apa keunggulan masing" statement:
   1. HAL_GPIO_WritePin
   2. HAL_GPIO_TogglePin
2. Menjelaskan apa yang dimaksud dengan function program seperti blink() pada percobaan ke-2

3. Mengetahui kenapa pada statement jika tombol ditekan di state RESET kenapa tidak SET
4. Mengetahui fungsi dari debugging pada percobaan kali ini
5. Tuliskan laporan praktikum sesuai dengan format template pada Lampiran 6

Prosedur Praktikum 2 - Timer, Interrupt, PWM, dan DMA Menggunakan STM32

Overview

Pada percobaan kali ini, mahasiswa mencoba timer, mengatur PWM dan DMA menggunakan STM32 dengan hasil code dari STM32CubeMX dapat diilustrasikan dengan diagram sebagai berikut.



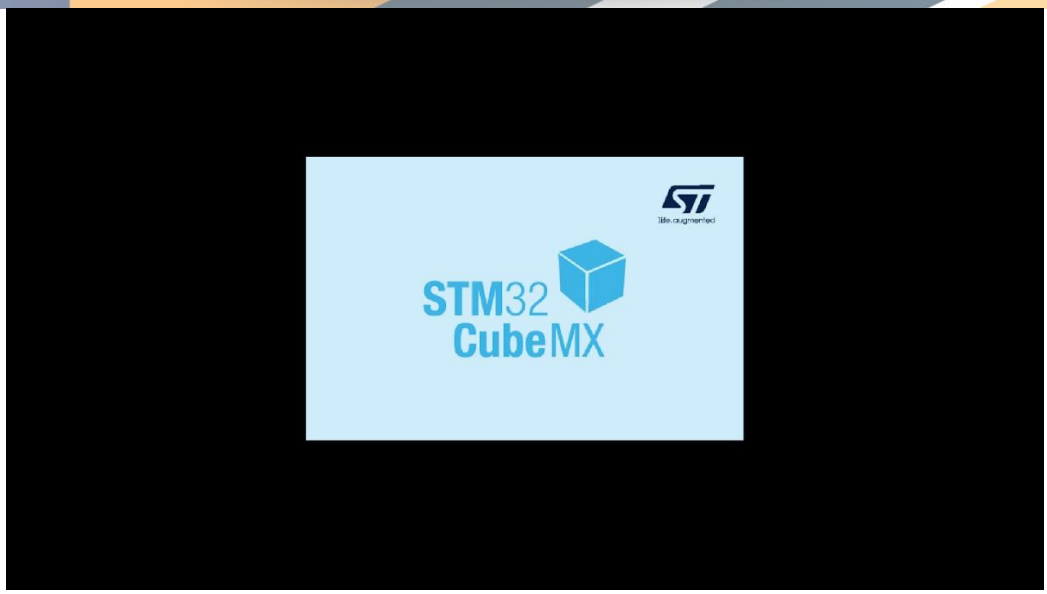
Gambar 1. Struktur Code STM32CubeMX

Tahap Persiapan Praktikum

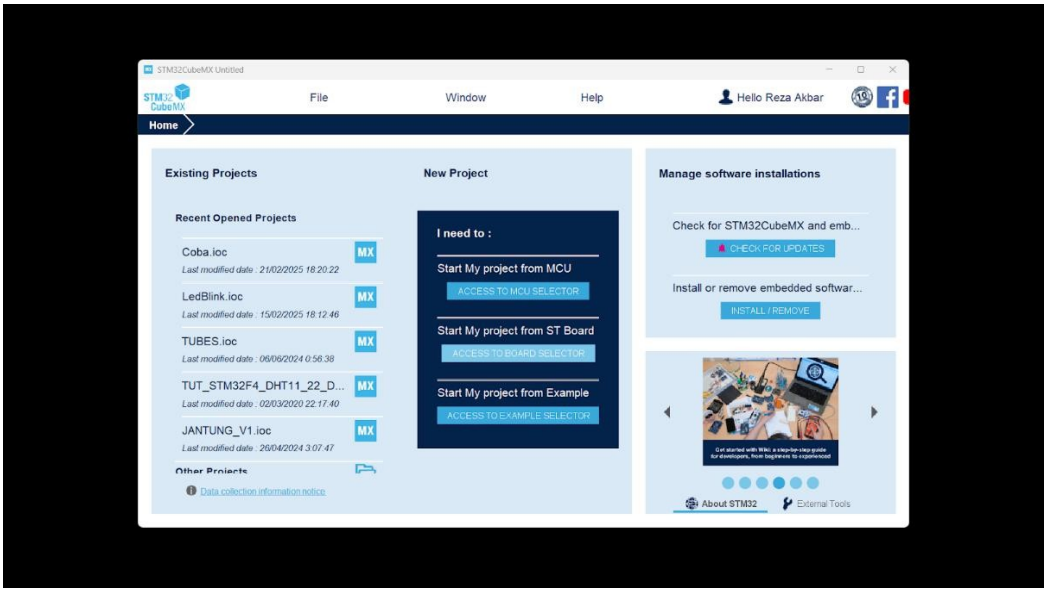
1. PC/ Laptop untuk melakukan praktikum
2. Software STM32CubeIDE
3. Software KEIL µVision ARM
4. STM32 Nucleo F466RE
5. Jumper f. LED + Resistor
6. Potensiometer
7. LCD I2C 16x2

Tahap Praktikum

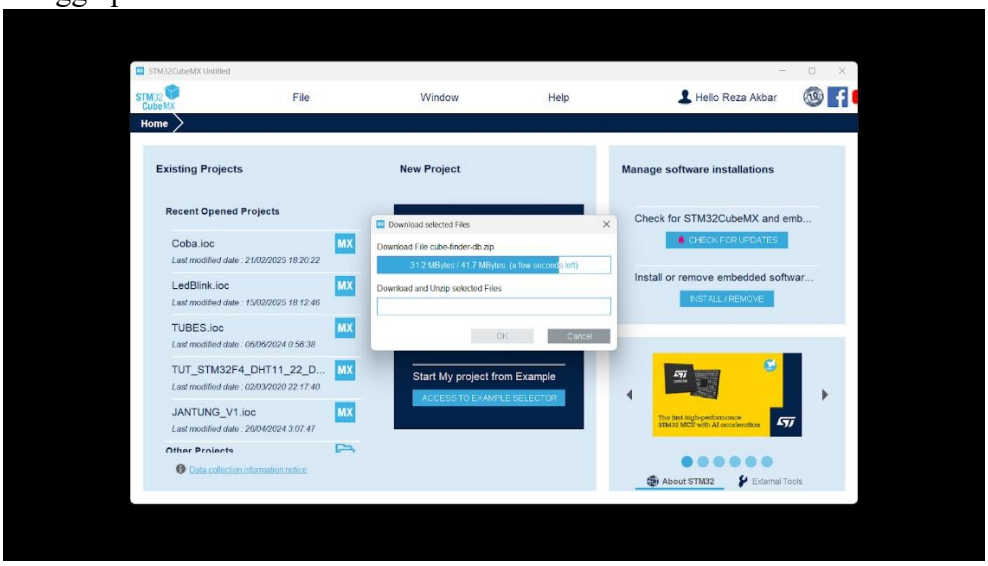
1. Buka Program STM32 CubeMX



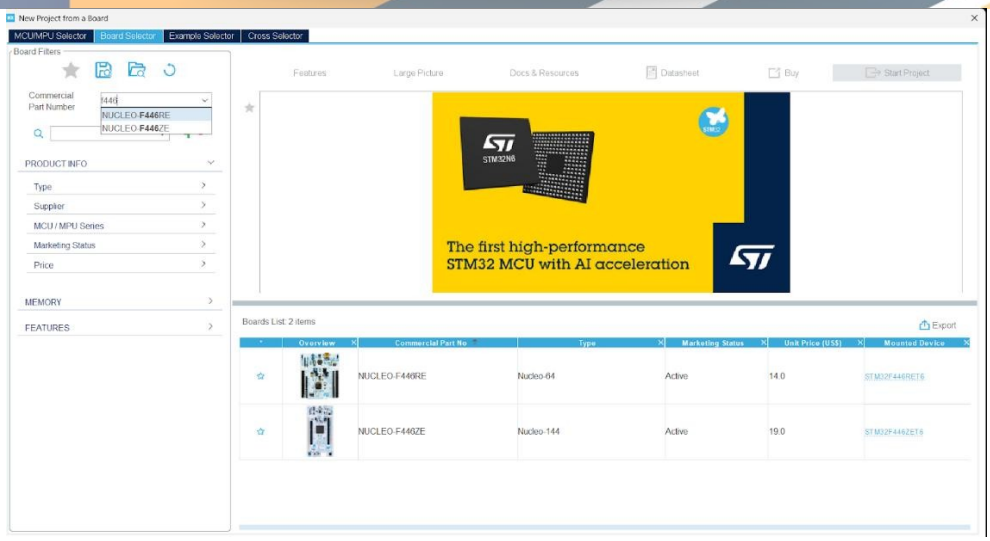
2. Pilih access to board selector



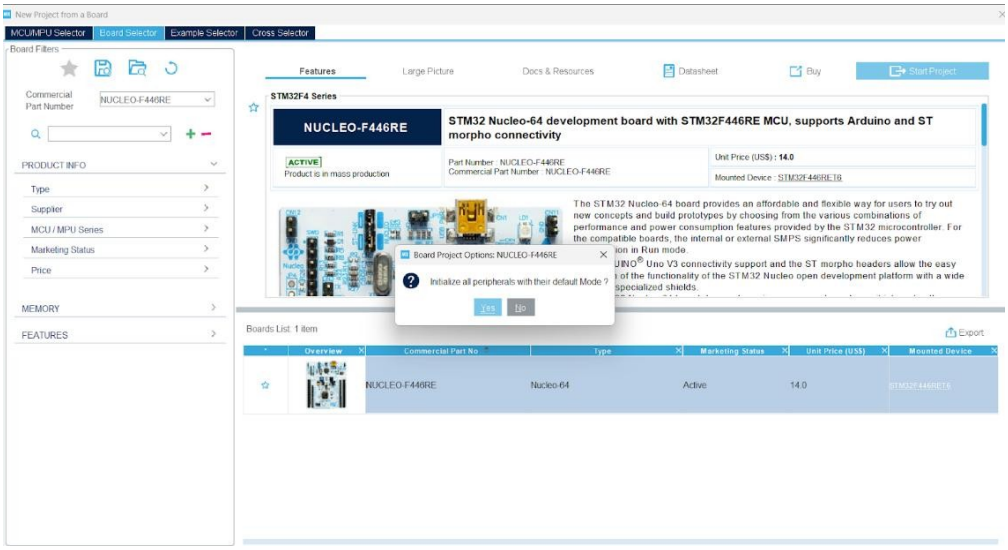
3. Tunggu proses download file



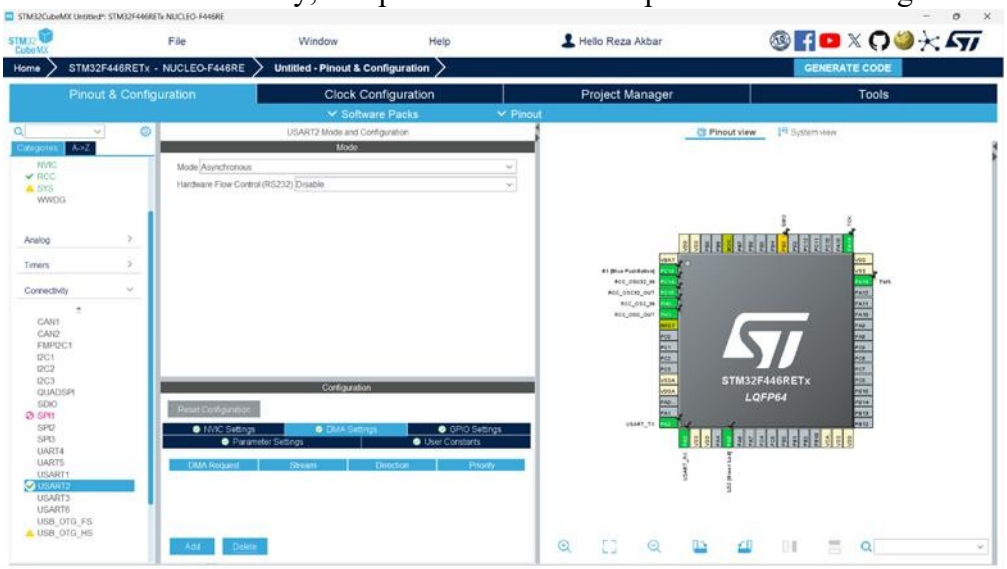
4. Cari board berdasarkan nama dengan type “NUCLEO-F446RE”



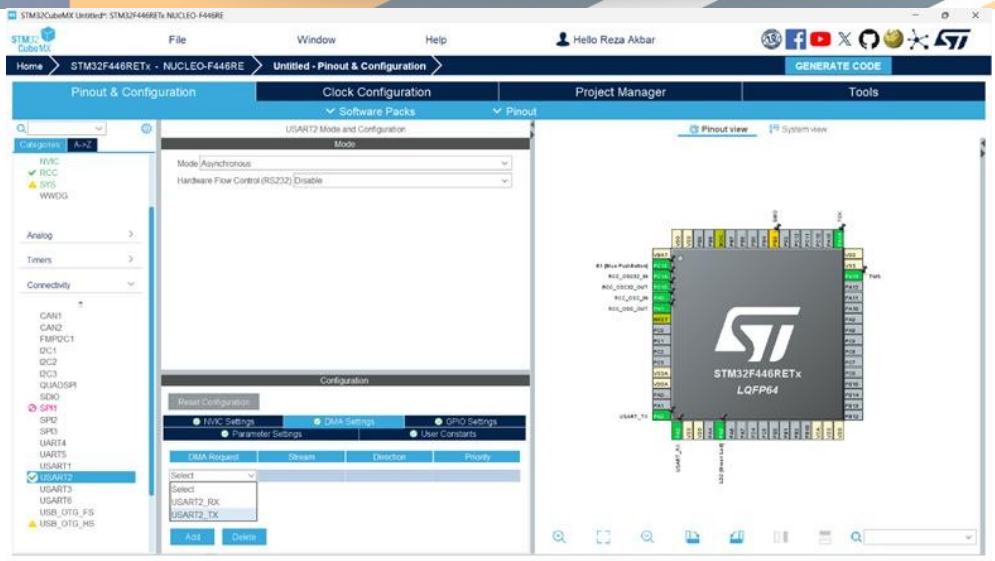
5. Klik “NUCLEO-F446RE” hingga berubah berwarna biru > Klik Start Project > Klik “Yes”



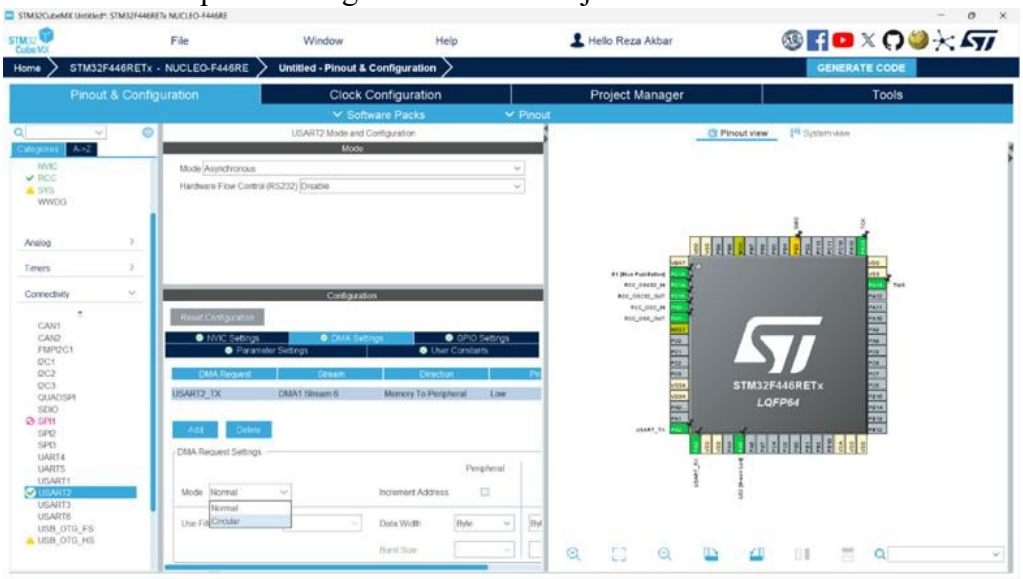
6. Pilih menu connectivity, dan pilih USART2 buka pilihan DMA Settings



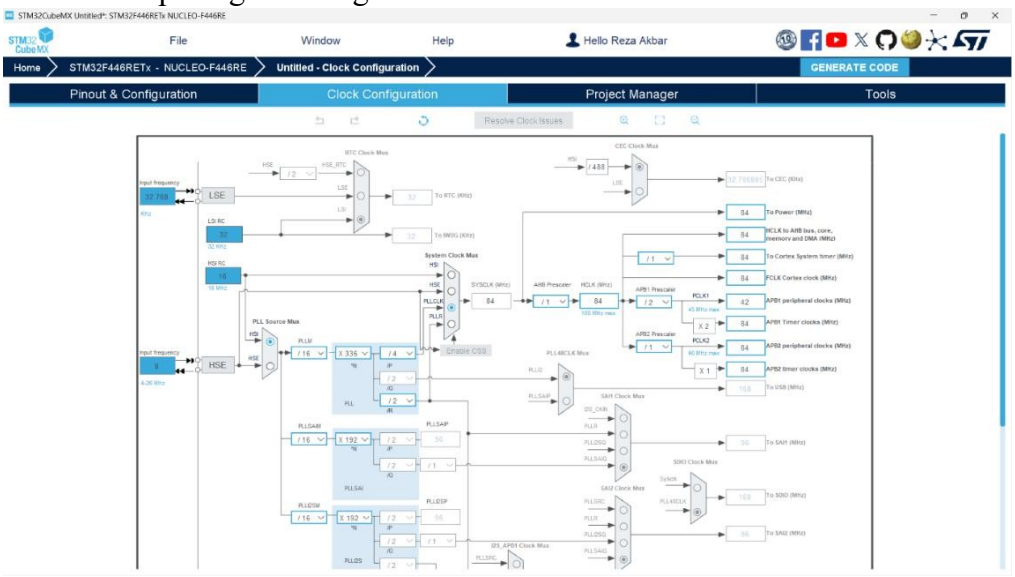
7. Klik Add dan select DMA Request menjadi USART2_TX



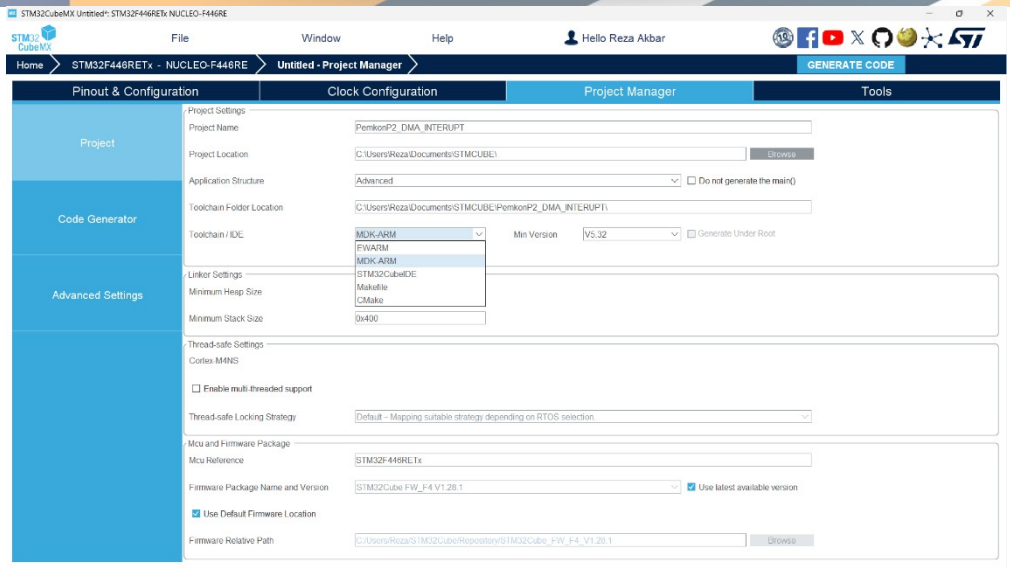
8. Pada DMA Request Settings ubah mode menjadi circular



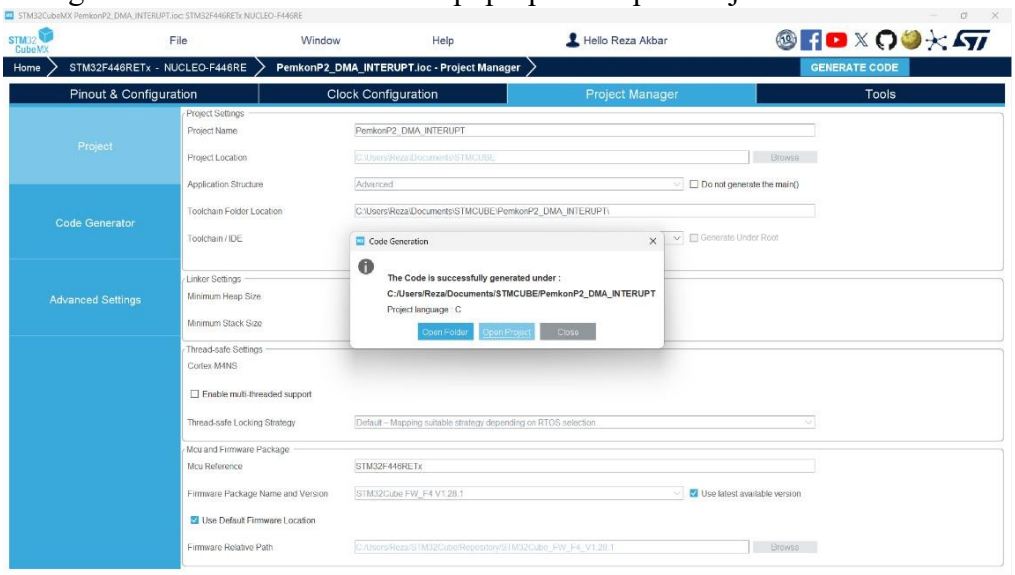
9. Biarkan tetap dengan settingan default



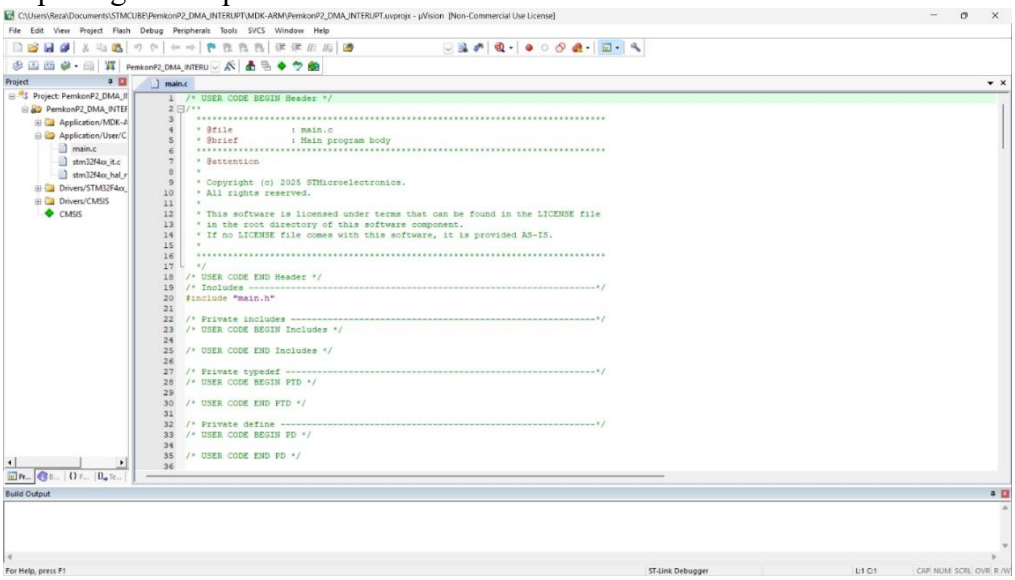
10. Beri nama project lalu pilih “Toolchain/IDE” > “MDK-ARM”



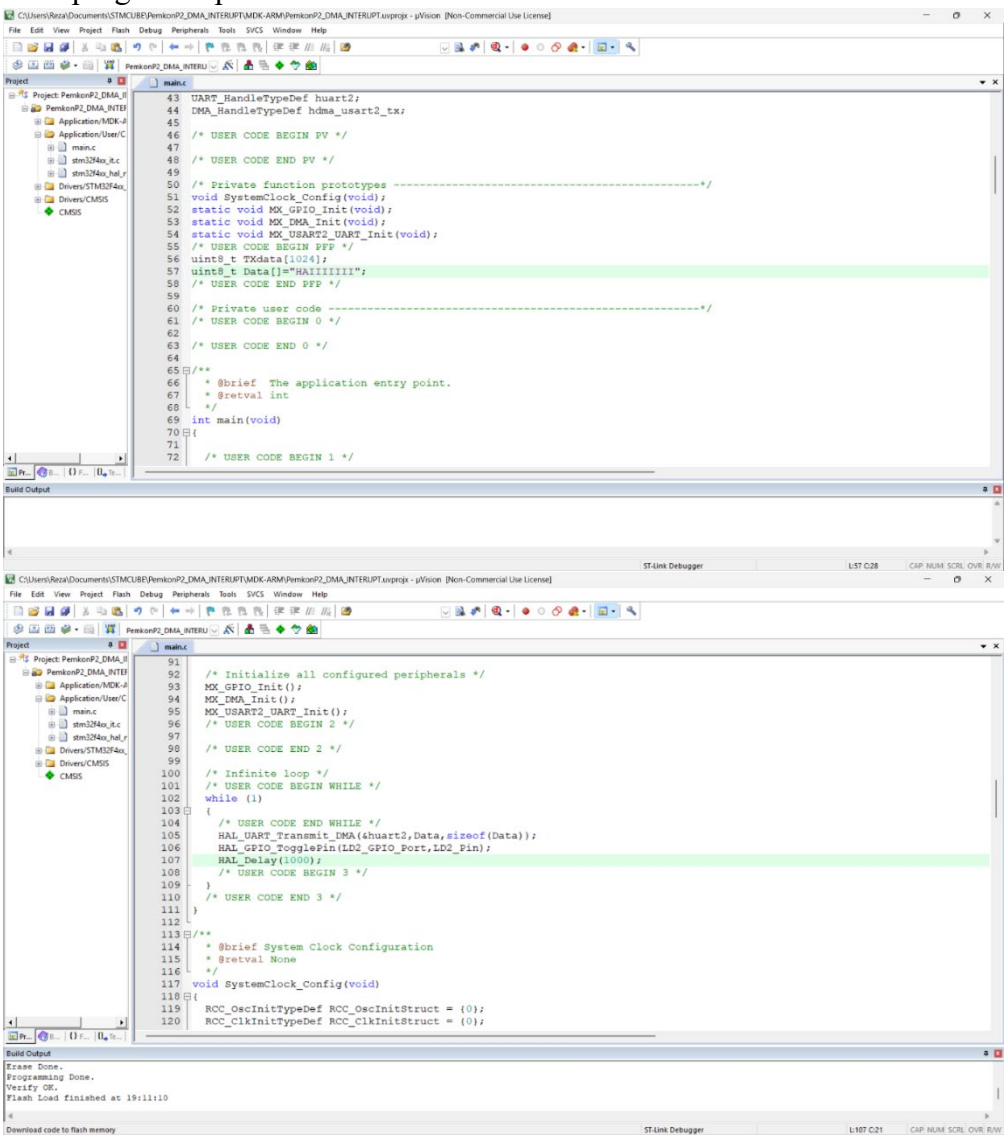
11. Klik generate code setelah muncul pop-up klik Open Project



12. Buka file main.c, serta pastikan target compiler telah sesuai dengan versi KEIL yang terpasang di computer.



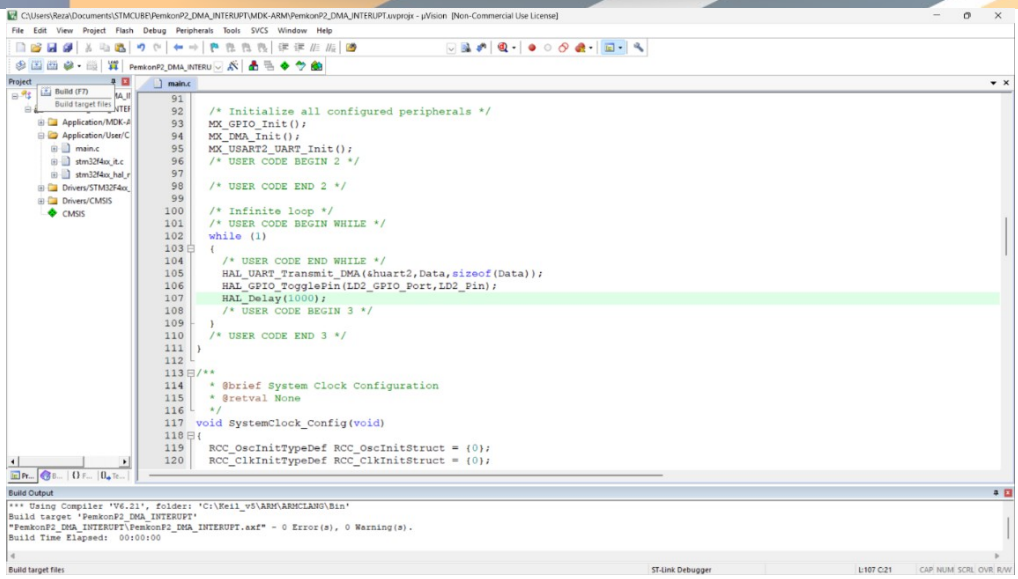
13. Buat program seperti di bawah ini



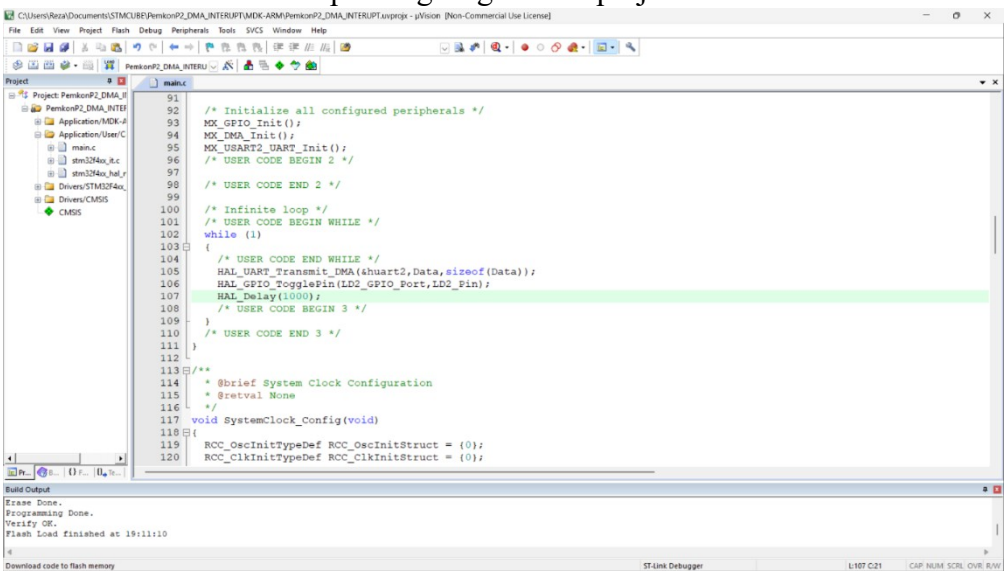
Source Code:

```
/* USER CODE BEGIN PFP */
uint8_t Data[]="HAIIIIIII";
/* USER CODE END PFP */

  while (1)
  {
    /* USER CODE END WHILE */
     HAL_UART_Transmit_DMA(&huart2,Data,sizeof(Data
     )); HAL_GPIO_TogglePin(LD2_GPIO_Port,LD2_Pin);
     HAL_Delay(1000);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

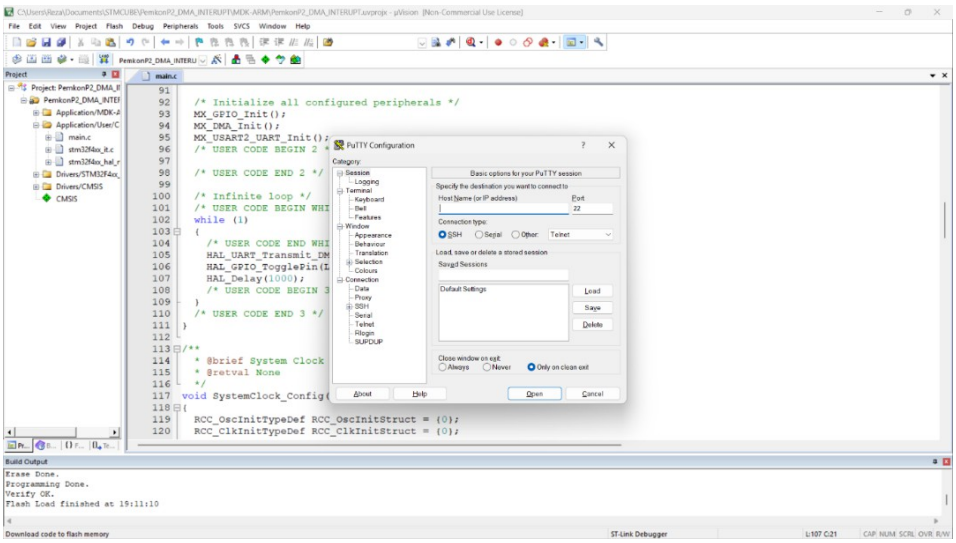
14. Build project pastikan tidak ada error, jika masih ada error silahkan di perbaiki kembali



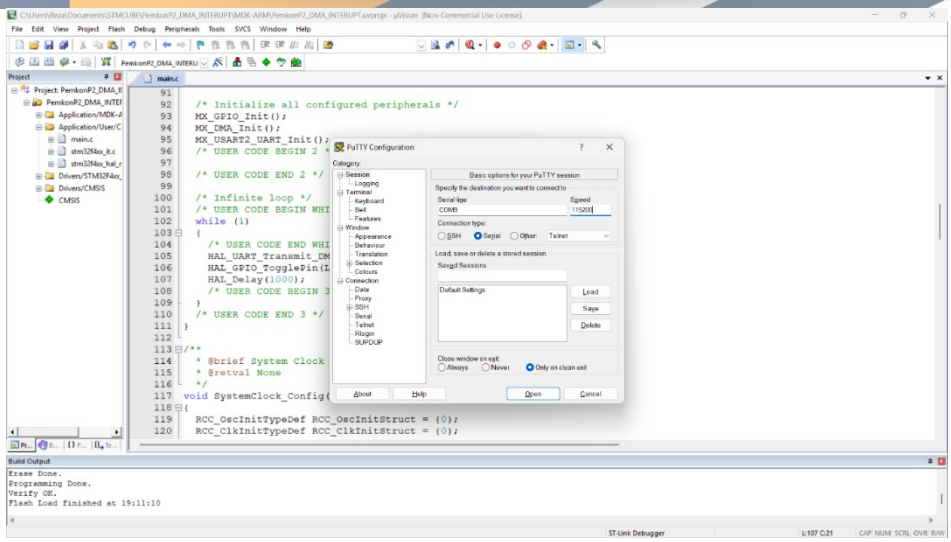
15. Setelah tidak ada error dapat langsung di load project



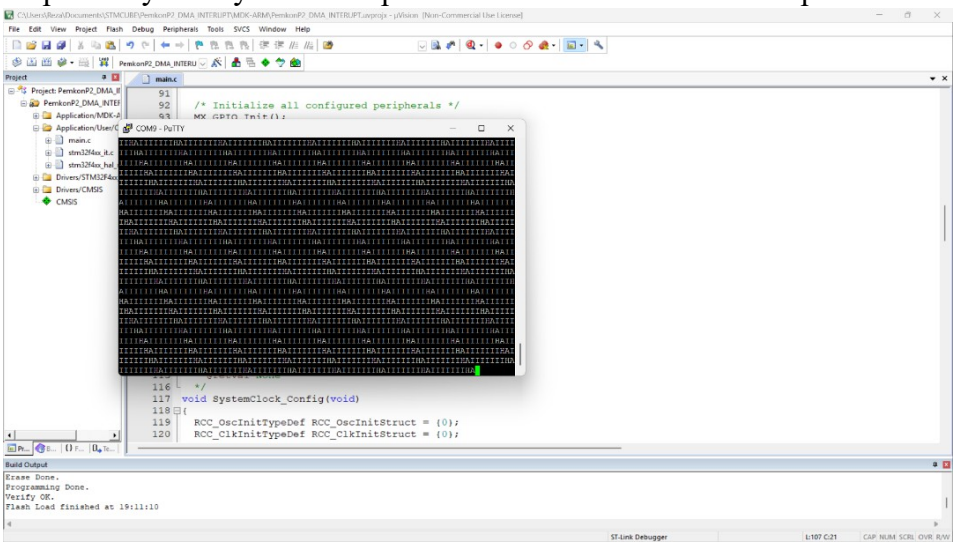
16. Kemudian buka software PuTTY



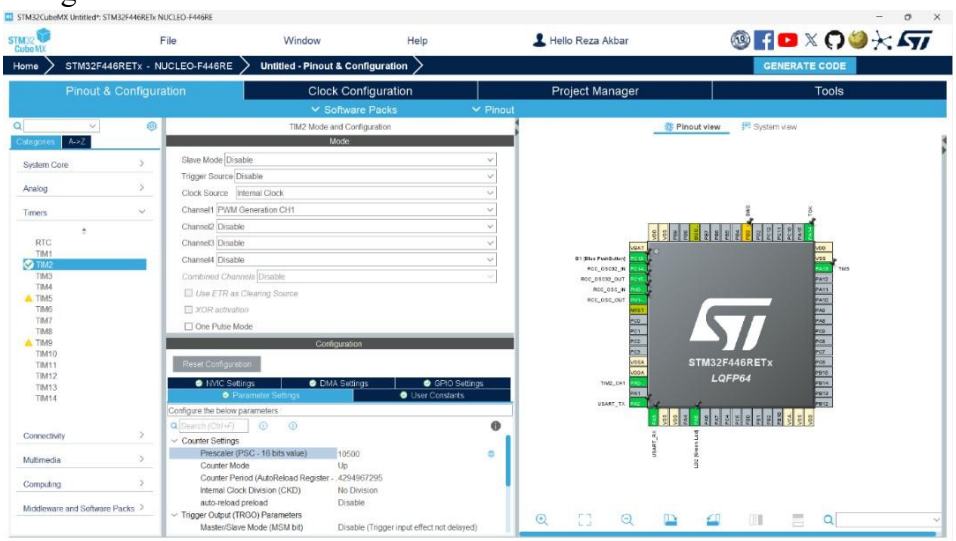
17. Pilih connection type ke Serial, isi serial COM sesuai COM yang terbaca di device manager computer, speed isi 115200 sesuai settingan STM32 dan open



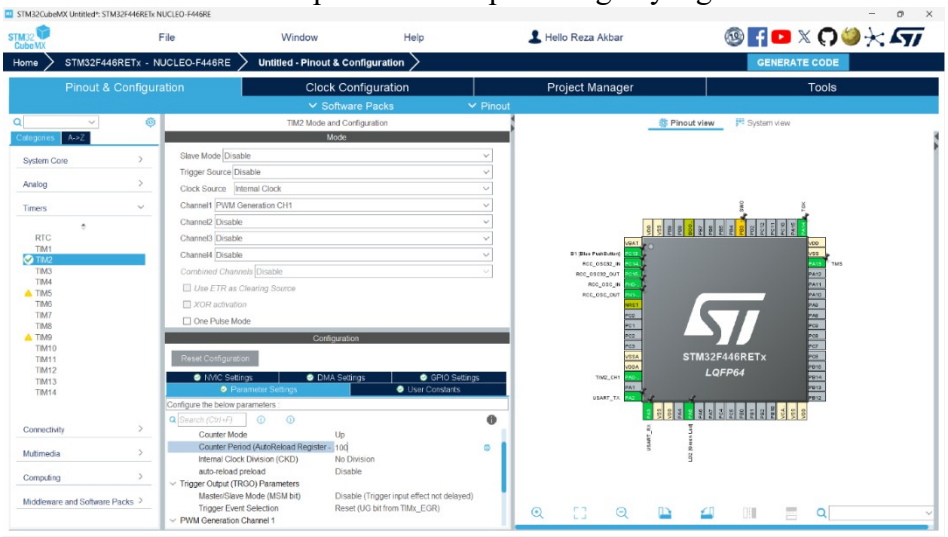
18. Jika percobaan berhasil maka di tampilan PuTTY akan menampilkan seperti di bawah tanpa adanya delay dan LED pada board STM32 akan berkedip



19. Untuk percobaan kedua ulangi Langkah 1 hingga 5 untuk membuat project baru, kemudian buka Timer klik TIM2 dan setting clock sour eke internal clock, channel1 ke PWM generation CH1, Prescale pada parameter setting bisa di isi dengan nilai 10500 sebagai contoh.



20. Masukkan nilai counter period sesuai perhitungan yang telah dilakukan.



21. Berikut rumus perhitungan frekuensi PWM

$$f_{PWM} = \frac{f_{Clock\ Source}}{(Prescale + 1)(CounterPeriod + 1)}$$

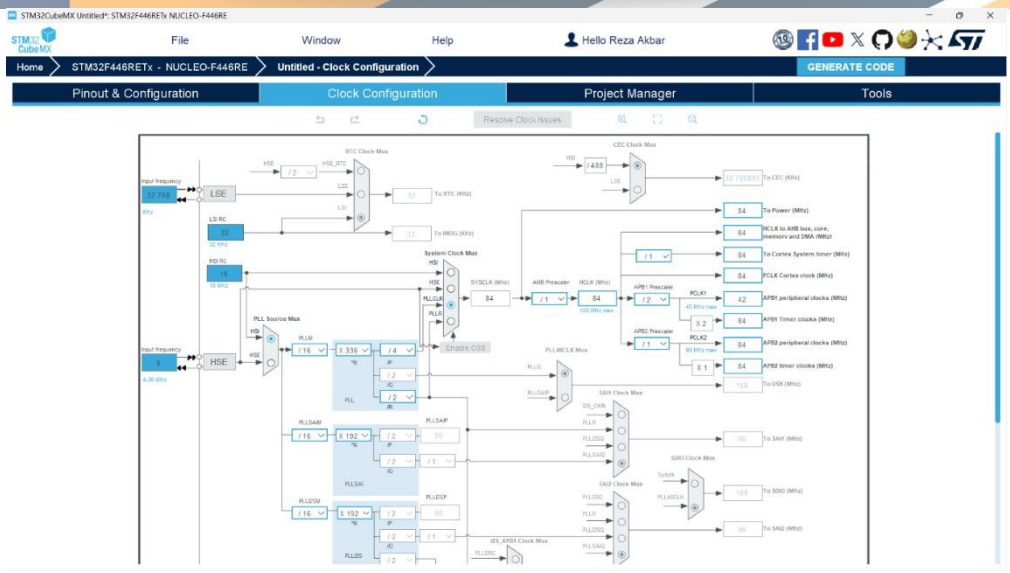
Sebagai contoh perhitungan frekuensi PWM

$$f_{PWM} = \frac{22.\ \ 84MHz}{(10500 + 1)(100 + 1)}$$

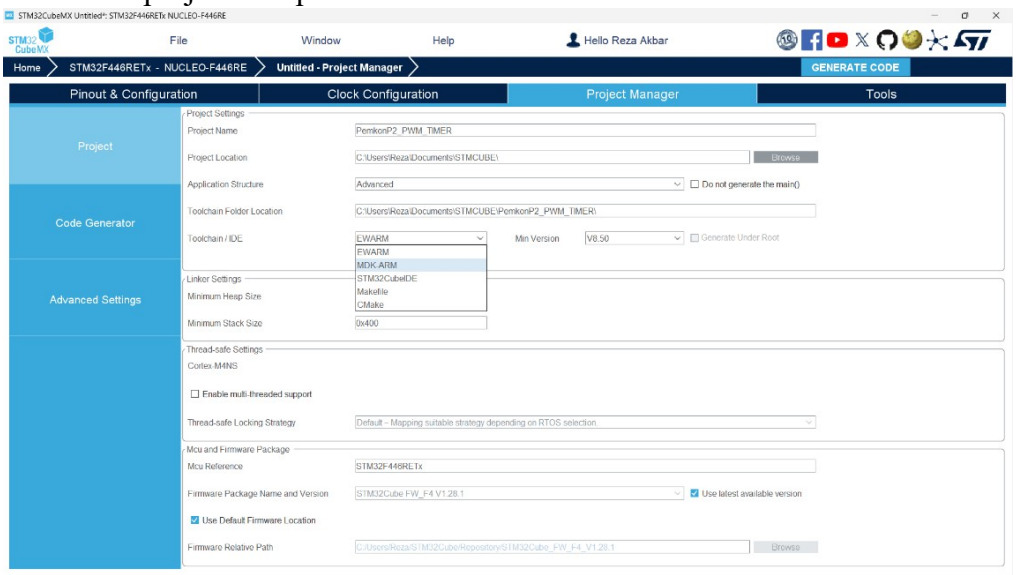
$$f_{PWM} = 79{,}2Hz$$

23. Buka NVIC settings enable TIM2 global interrupt

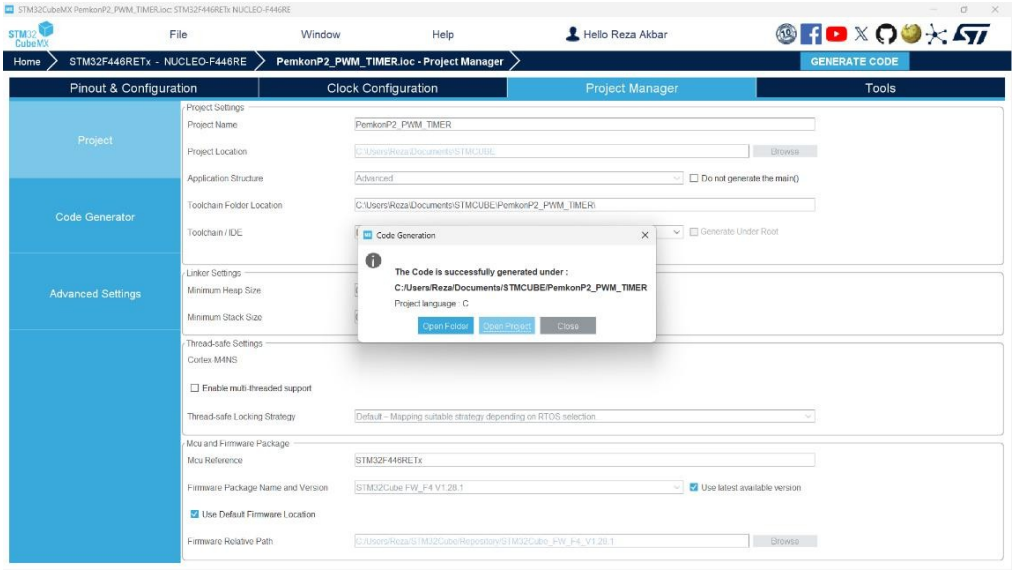
24. Untuk settingan clock biarkan default



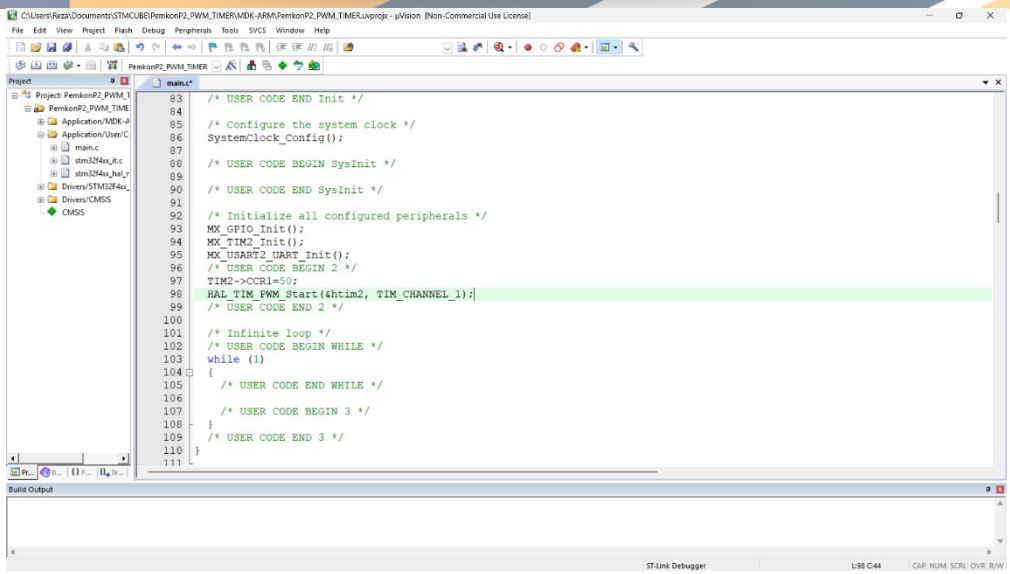
25. Beri nama project lalu pilih “Toolchain/IDE” > “MDK-ARM”



26. Setelah generate code maka akan muncul pop-up dan klik open project



27. Buat program seperti yang ada di bawah ini



Source Code:

```
/* Initialize all configured
peripherals */ MX_GPIO_Init();
MX_TIM2_Init
();
MX_USART2_UA
RT_Init();
/* USER
CODE BEGIN
2 */ TIM2-
>CCR1=50;
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

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

28. Setelah itu build project setelah dipastikan tidak ada error maka load project, kemudian tambahkan jumper pada pin A0 dan Gnd untuk di sambungkan ke osiloskop.
29. Cek hasil grafik pada osiloskop dan cek apakah hasil frekuensi yang terbaca di osiloskop telah sesuai dengan frekuensi yang diinginkan.

Tahap Pasca Praktikum

1. Jelaskan function DMA mengapa perlu dilakukan pada praktikum kali ini
2. Berikan analisa hubungan conectivity function USART dengan software PuTty
3. Pada function dibawah ini jelaskan pengaruh yang terjadi pada osiloskop

```
TIM2->CCR1=50;
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
```

Prosedur Praktikum 3 - ADC, DAC, Serial (I2C) Menggunakan STM32

Overview

Praktikum ini berfokus untuk memahami dan mengimplementasikan penggunaan ADC (Analog Digital Converter), DAC (Digital Analog Converter), dan komunikasi serial I2C pada mikrokontroler STM32.

Tahap Persiapan Praktikum

Sebelum kegiatan praktikum berlangsung, diharapkan praktikan dapat mempersiapkan beberapa hal seperti berikut.

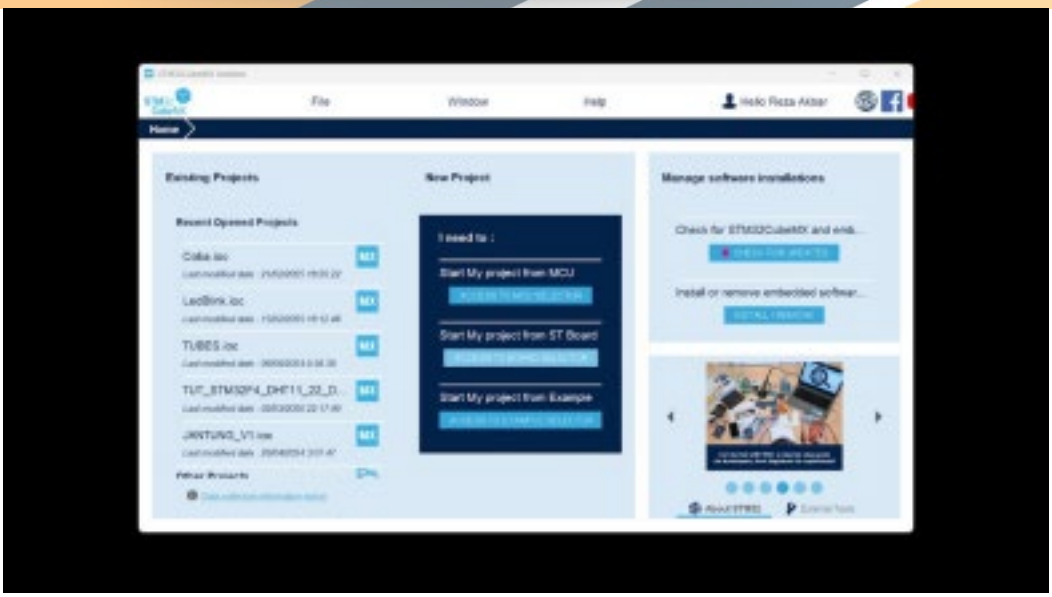
a. PC/ Laptop untuk melakukan praktikum
b. Software STM32CubeMX
c. Software KEIL µVision ARM
d. STM32 Nucleo F466RE
e. Osiloskop
f. Jumper
g. Potensiometer
h. LCD I2C 16x2

Tahap Praktikum

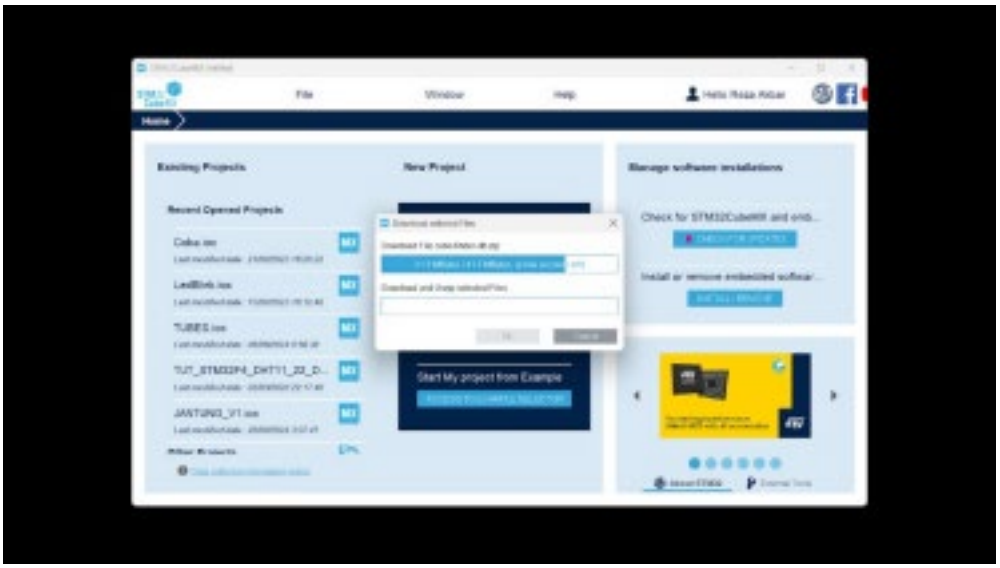
P3 “ADC, DAC, Serial (I2C) Menggunakan STM32”

1. Buka Program STM32 CubeMX

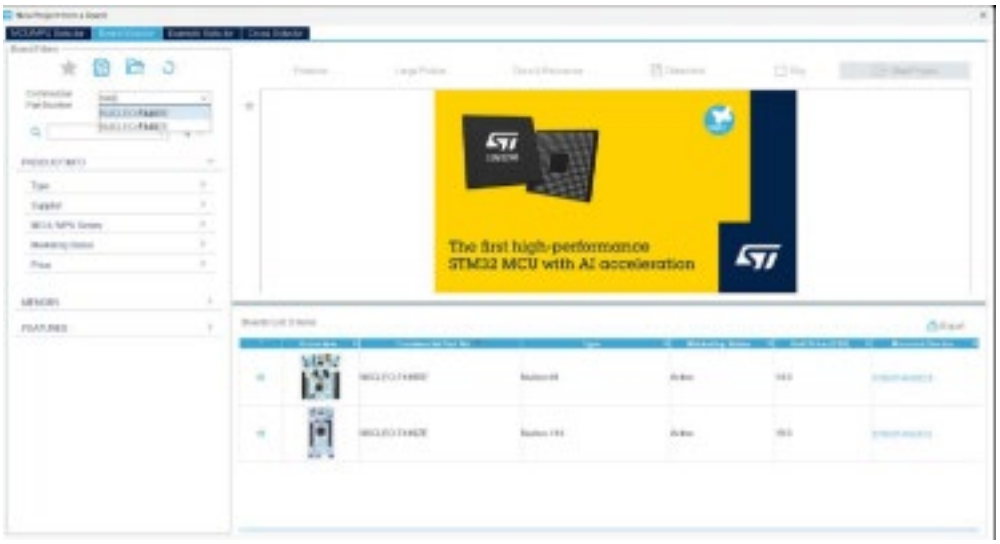
2. Pilih access to board selector



3. Tunggu proses download file Modul Praktikum Pemrograman Kontroler

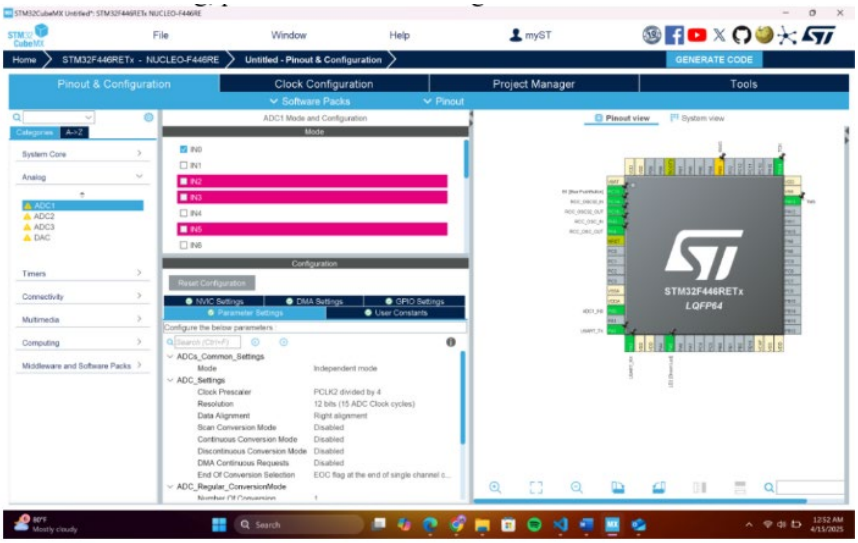


4. Cari board berdasarkan nama dengan type “NUCLEO-F446RE”

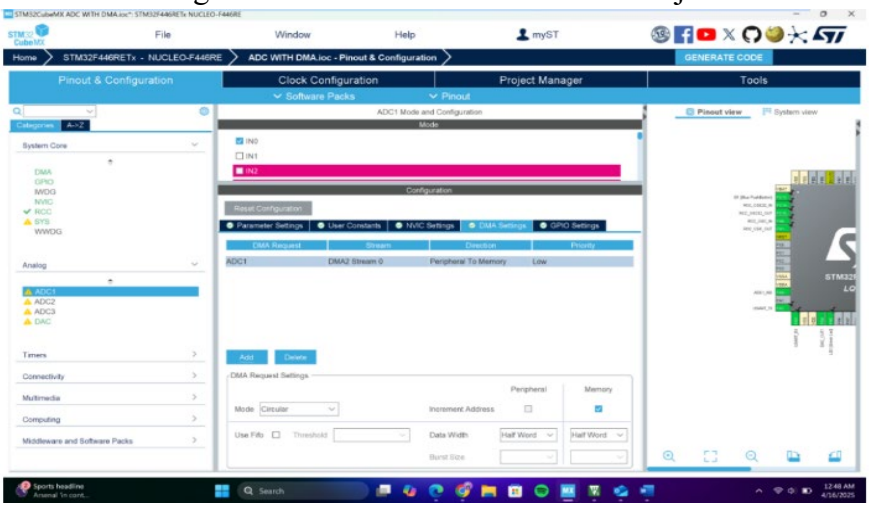


5. Klik “NUCLEO-F446RE” hingga berubah berwarna biru > Klik Start Project > Klik “Yes”

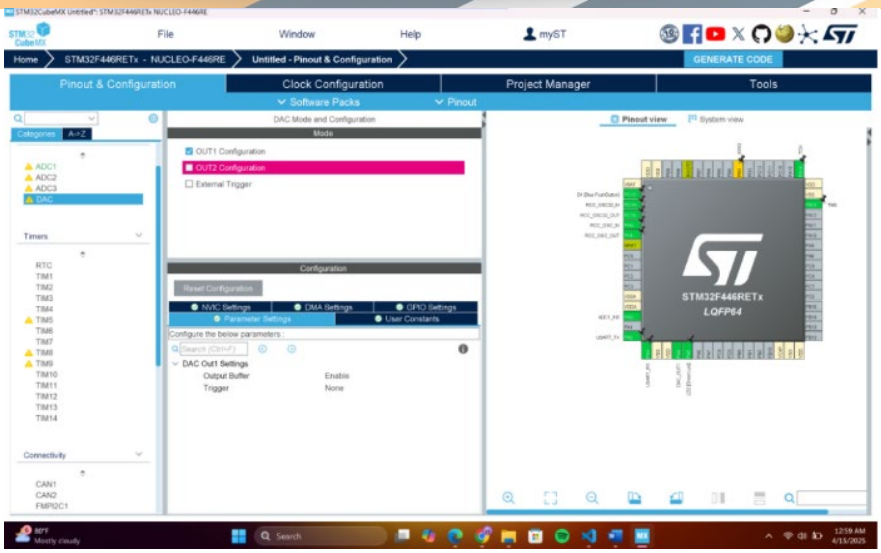
6. Pilih menu Analog, pilih ADC1 dan centang IN0.



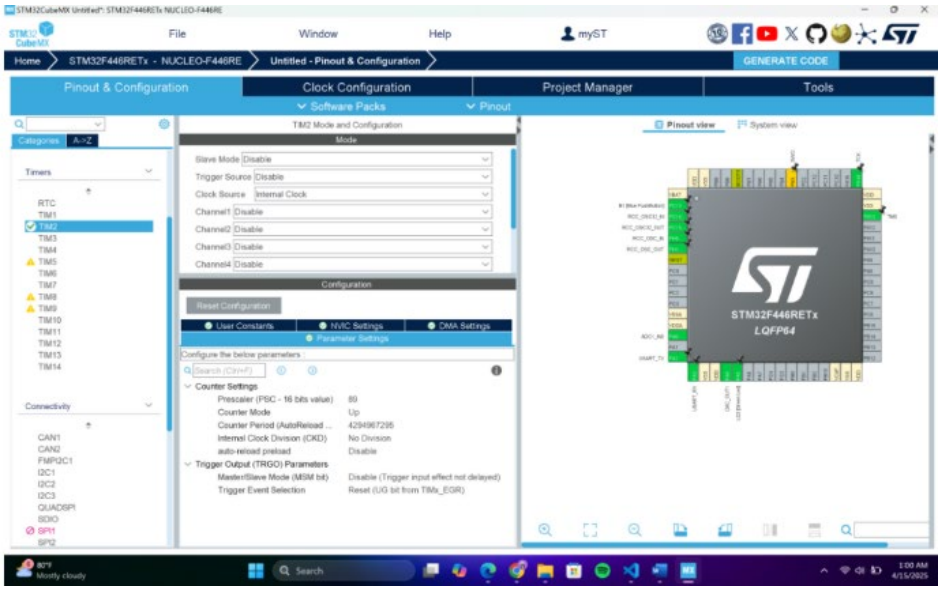
7. Pilih menu DMA setting dan add ADC1 dan ubah mode menjadi circular



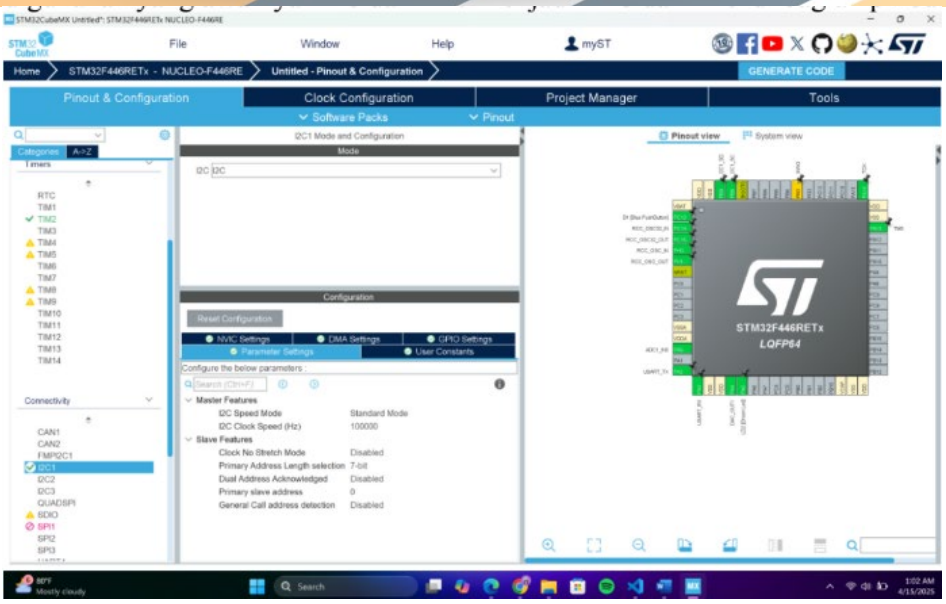
8. Klik DAC dan centang OUT1 Configuration.



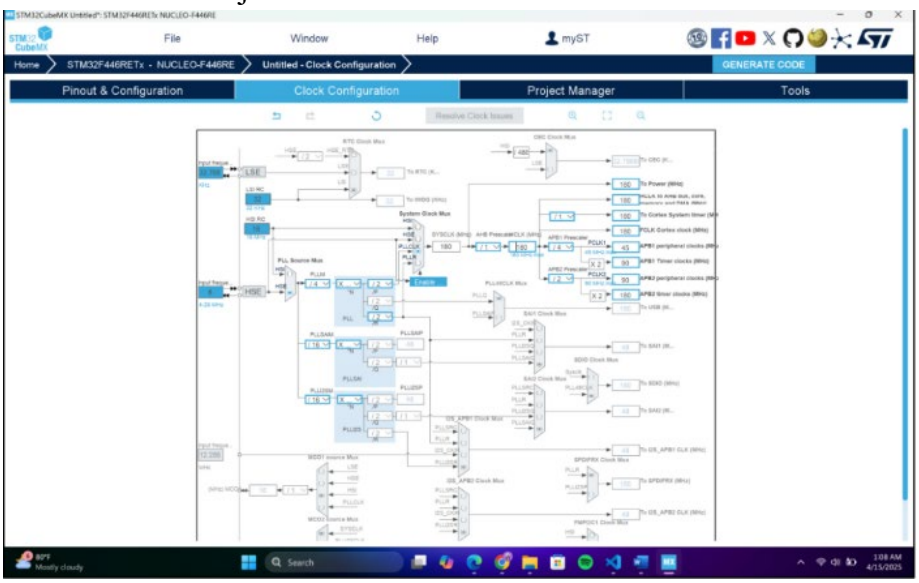
9. Pilih menu timer enable TIM2 serta ubah clock source menjadi Internal Clock dan ganti nilai prescaler menjadi 89



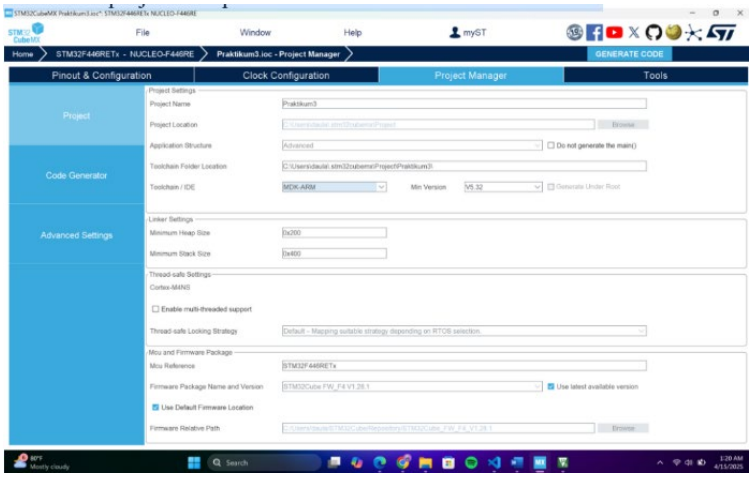
10. Pilih menu connectivity pilih I2C1 dan pilih I2C, kemudian ubah pin SDA SCL yang digunakan yang awalnya PB6 dan PB7 menjadi PB8 dan PB9 di bagian pinout view



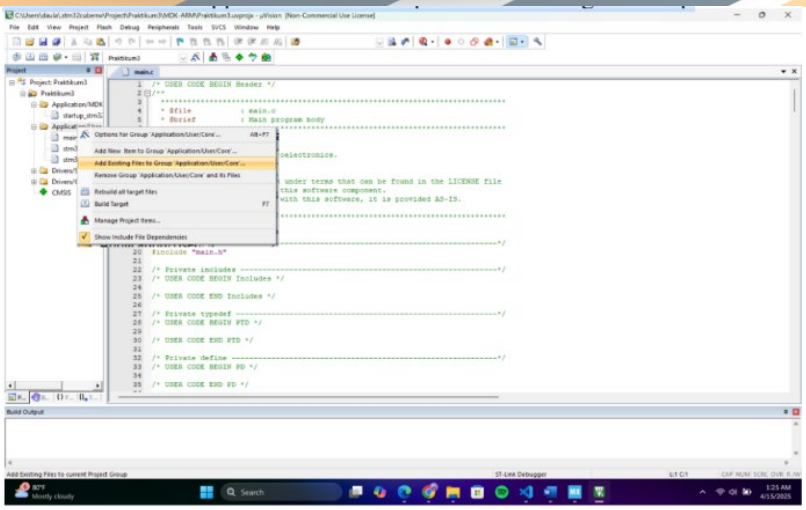
11. Buka clock configuration pindah pilihan PLL source yang awalnya di HS menjadi HSE serta ubah clock max menjadi 180



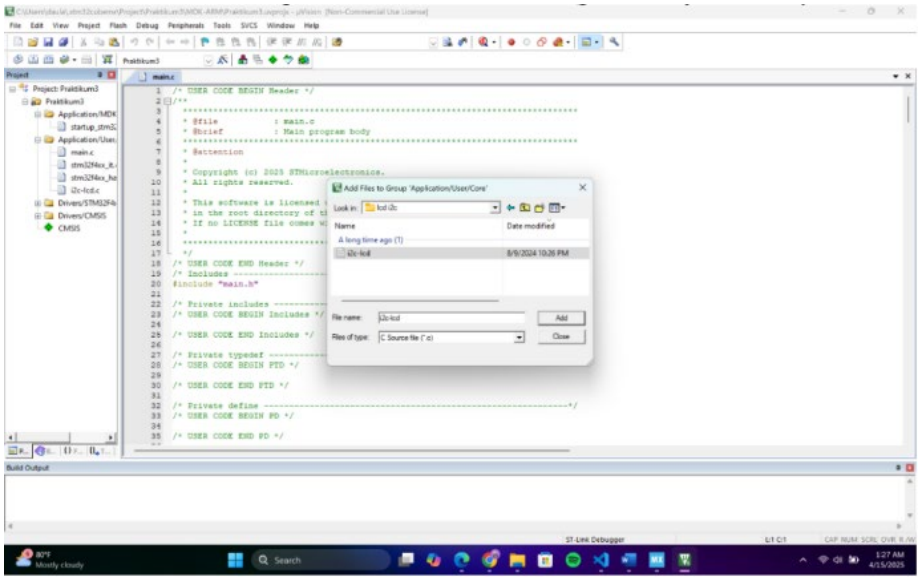
12. Beri nama project lalu pilih “Toolchain/IDE” > “MDK-ARM”



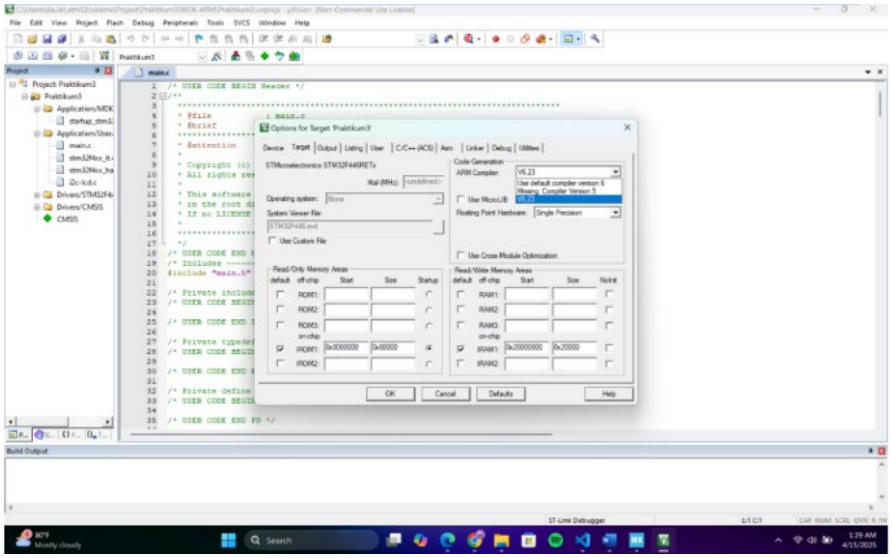
13. Klik kanan folder Application/user dan pilih Add Exiting to Group



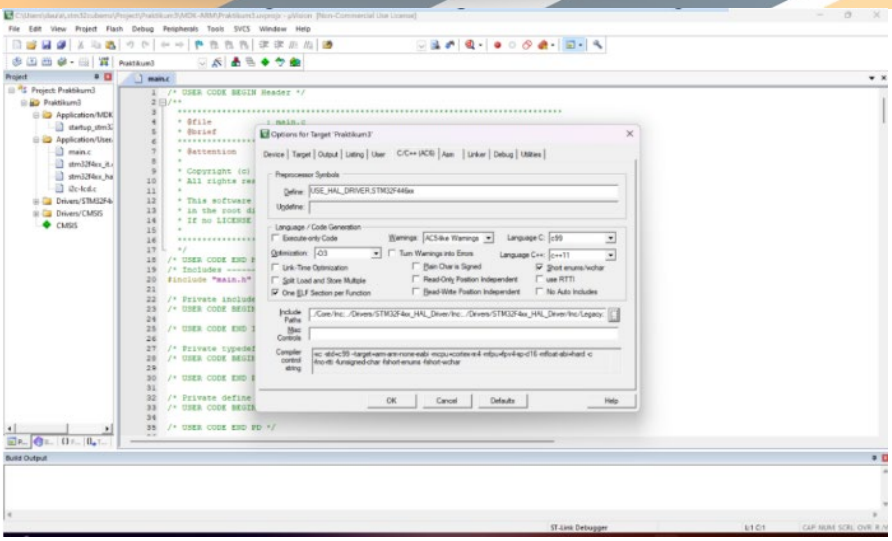
14. Cari Folder lcd i2c dan pilih i2c-lcd.c, klik Add (pastikan file library telah di extract)



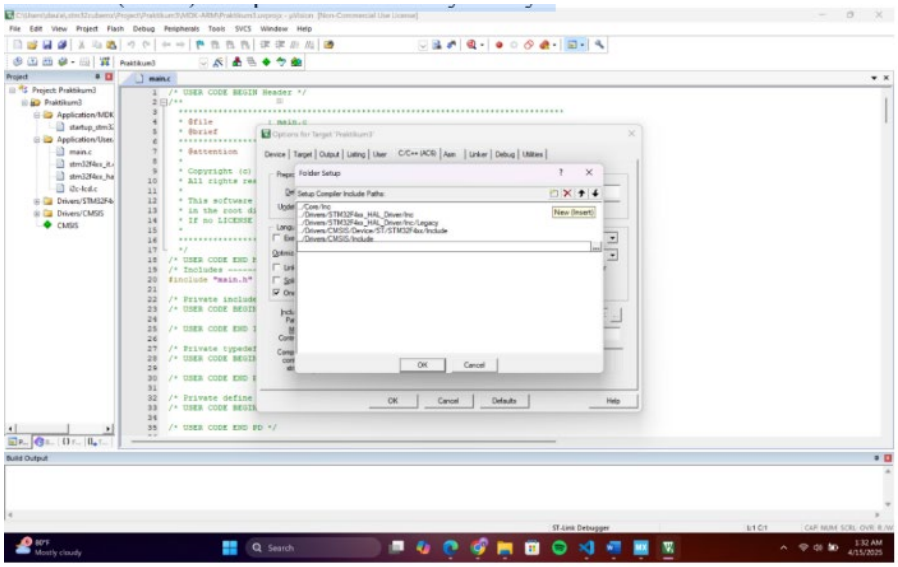
15. Klik Target Options ubah ARM Compiler sesuai versi Keil yang terinstall



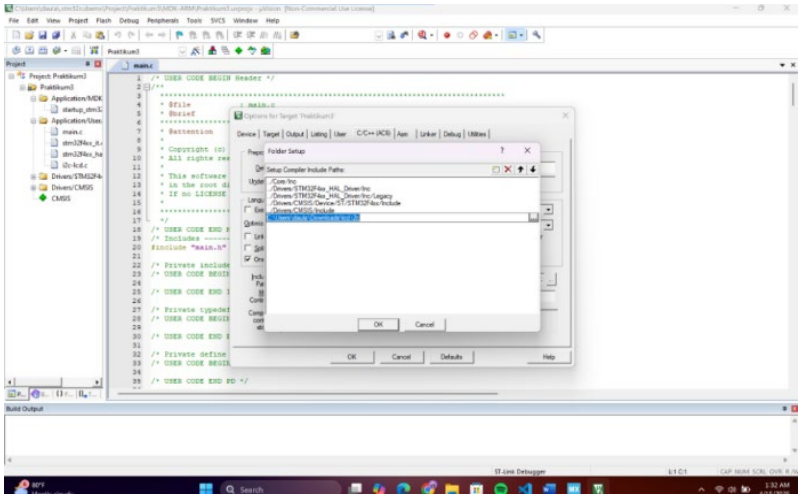
16. Pilih menu C/C++ dan pilih titik 3 di bagian kanan include path



17. Klik new (insert) dan pilih folder library i2c nya



18. Setelah di pilih folder kemudian klik “ok”



19. Tambahkan header file sesuai gambar dibawah ini dan program seperti dibawah ini

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "i2c-lcd.h"
#include "stdio.h"
/* Private includes ----------------------------------------------------------*/
  /* USER CODE BEGIN 2 */
  lcd_init();
  lcd_send_string("HAI");
  HAL_Delay(3000);
  lcd_put_cur(1,0);
  lcd_send_string("Elka");
  lcd_clear();
  /* USER CODE END 2 */
  while (1)
  {
    /* USER CODE END WHILE */
    float flt = 12.345;
    char fltChar[7];
    sprintf(fltChar, "%.3f", flt);
    lcd_put_cur(0, 0);
    lcd_send_string ("Value");
    lcd_put_cur(1,0);
    lcd_send_string(fltChar);
    HAL_Delay(500);
    flt=flt+1.23;
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

20. Kemudian compile dan upload code ke STM32 dan lihat hasilnya pada LCD
21. Untuk percobaan ke-2 yaitu read ADC Potensiometer dengan code seperti dibawah ini

```
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
   uint32_t value=0;
/* USER CODE END PD */

  while (1)
  {
    /* USER CODE END WHILE */
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,100);
    value=HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    char numChar[7];
    sprintf(numChar, "%d   ", value);
    lcd_put_cur(0, 0);
    lcd_send_string ("Value");
    lcd_put_cur(1,0);
    lcd_send_string(numChar);
    HAL_Delay(200);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

22. Untuk percobaan ke-3 yaitu read ADC Potensiometer dengan DMA code seperti dibawah ini

```
  /* USER CODE BEGIN 2 */
  lcd_init();
  lcd_send_string("HAI");
  HAL_Delay(3000);
  lcd_put_cur(1,0);
  lcd_send_string("Elka");
  lcd_clear();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    HAL_ADC_Start_DMA(&hadc1, &value, 10);
    HAL_Delay(10);
    HAL_ADC_Stop_DMA(&hadc1);
    char numChar[7];
    sprintf(numChar, "%d   ",value);
    lcd_put_cur(0,0);
    lcd_send_string("Value");
    lcd_put_cur(1,0);
    lcd_send_string(numChar);
    HAL_Delay(200);
    /* USER CODE BEGIN 3 */
  }
```

23. Kemudian compile dan upload code ke STM32 dan lihat hasilnya pada LCD
24. Untuk percobaan ke-4 yaitu DAC dengan hasil akhir generate sinewave, dengan code seperti dibawah ini

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
uint16_t sinewave[100] = {0, 4, 16, 36, 64, 100, 144, 195, 253, 319,
                          391, 470, 555, 646, 742, 844, 950, 1061, 1176, 1294,
                          1415, 1538, 1664, 1791, 1919, 2048, 2176, 2304, 2431, 2557,
                          2680, 2801, 2919, 3034, 3145, 3251, 3353, 3449, 3540, 3625,
                          3704, 3776, 3842, 3900, 3951, 3995, 4031, 4059, 4079, 4091,
                          4095, 4091, 4079, 4059, 4031, 3995, 3951, 3900, 3842, 3776,
                          3704, 3625, 3540, 3449, 3353, 3251, 3145, 3034, 2919, 2801,
                          2680, 2557, 2431, 2304, 2176, 2048, 1919, 1791, 1664, 1538,
                          1415, 1294, 1176, 1061, 950, 844, 742, 646, 555, 470,
                          391, 319, 253, 195, 144, 100, 64, 36, 16, 4};

void microDelay (uint16_t delay)
{
  __HAL_TIM_SET_COUNTER(&htim2, 0);
  while (__HAL_TIM_GET_COUNTER(&htim2) < delay);
}
  /* USER CODE BEGIN 2 */
  HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
  HAL_TIM_Base_Start(&htim2);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    for(uint8_t i=0; i<100; i++)
      {
          HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, sinewave[i]);
          microDelay(5);
      }
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

25. Kemudian compile dan upload program serta sambungkan pin DAC ke osiloskop serta analisa hasil di osiloskop.

Tahap Pasca Praktikum

1. Mengetahui kenapa nilai maksimal ADC memiliki nilai 4095
2. Mengetahui bagaimana cara kerja serial I2C
3. Mampu menjelaskan perbedaan ADC dan DAC
4. Mampu menganalisa apa yang diperoleh dari percobaan ketiga mengenai gelombang
5. Mengetahui apa perbedaan read ADC jika menggunakan PollForConversion dengan ADC menggunakan DMA

Prosedur Praktikum 4 - FREERTOS Menggunakan STM32

Overview

Praktikum ini berfokus untuk memahami dan mengimplementasikan penggunaan FreeRTOS pada mikrokontroler STM32, termasuk konfigurasi multitasking, penggunaan timer, komunikasi I2C, ADC, serta pengendalian output seperti LED dan LCD melalui sistem operasi real-time.

Tahap Persiapan Praktikum

Praktikum pemrograman kontroler dilakukan secara *offline*, sehingga peralatan yang diperlukan adalah :

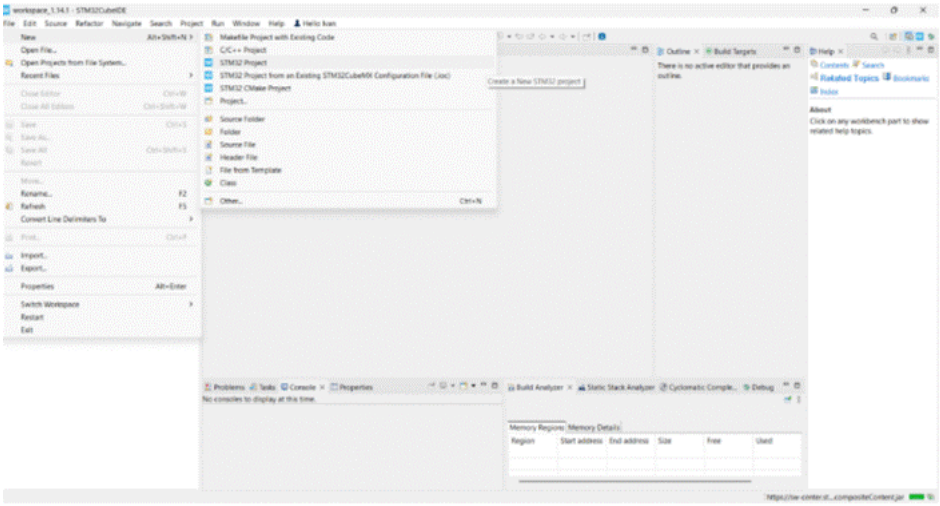
a. PC/ Laptop untuk melakukan praktikum
b. Software STM32CubeIDE
c. Software KEIL µVision ARM
d. STM32 Nucleo F466RE
e. Jumper
f. LED + Resistor
g. Potensiometer

h. LCD I2C 16x2

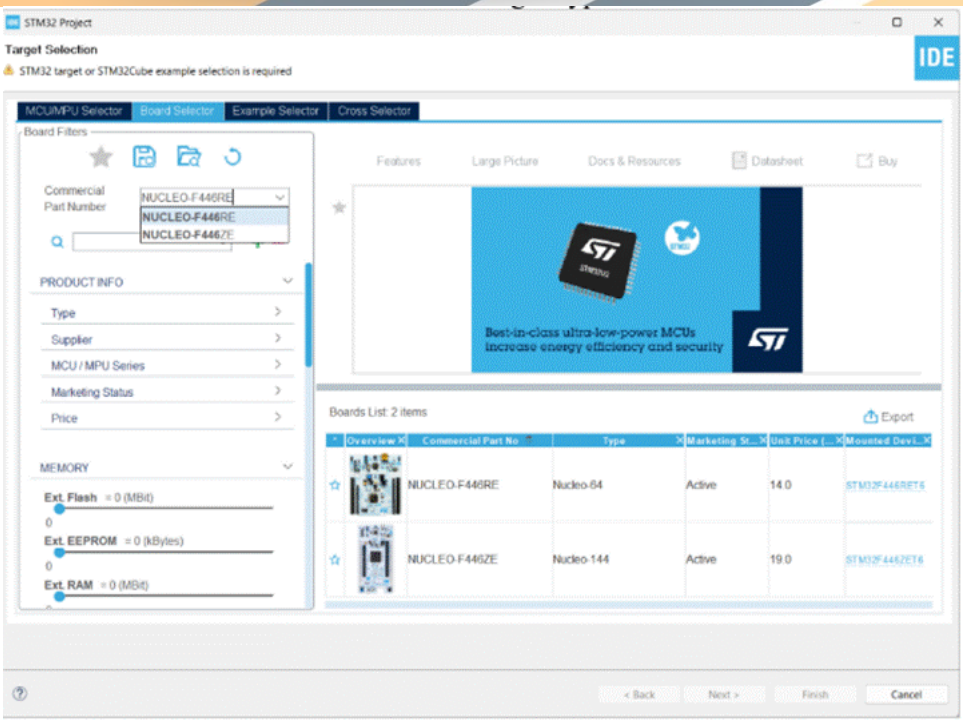
Tahap Praktikum

1. Buka Program STM32 CubeIDE

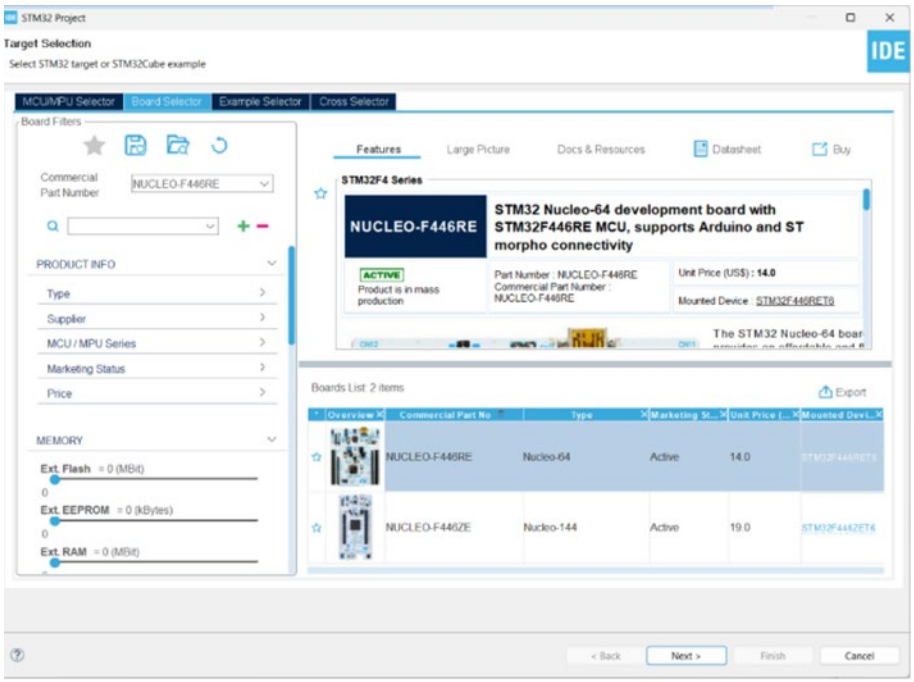
2. Pilih access to board selector



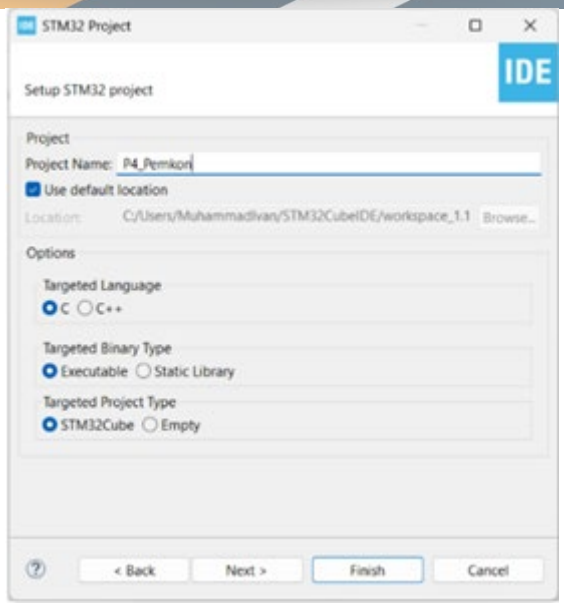
3. Kemudian cari board berdasarkan nama dengan type “NUCLEO-F446RE”



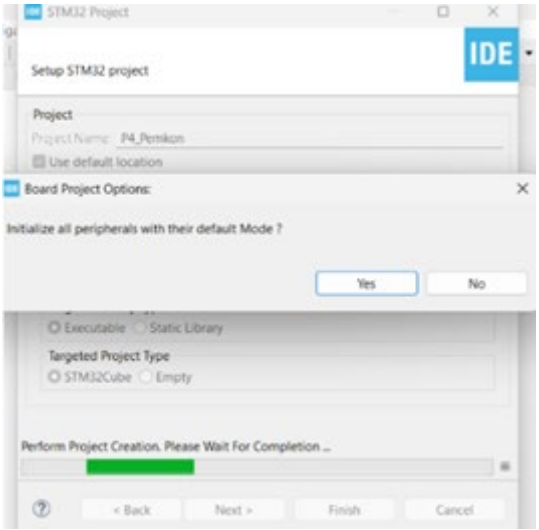
4. Klik “NUCLEO-F446RE” hingga berubah berwarna biru > Klik Next



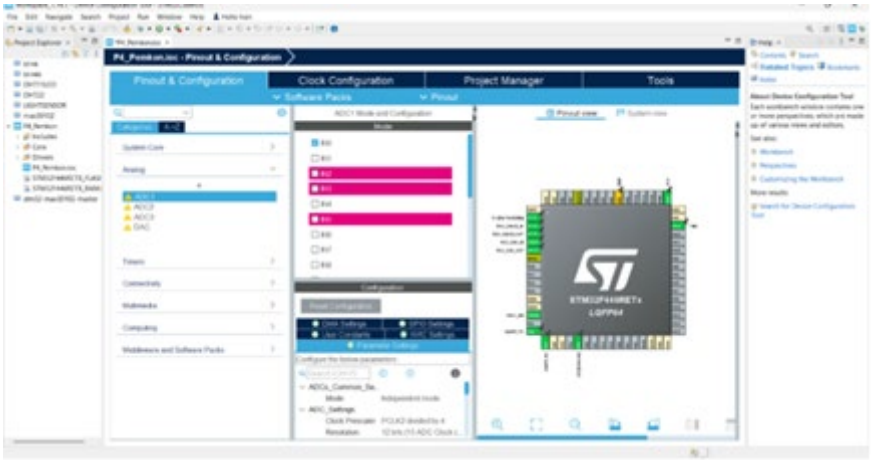
5. Masukkan Project Name kemudian klik



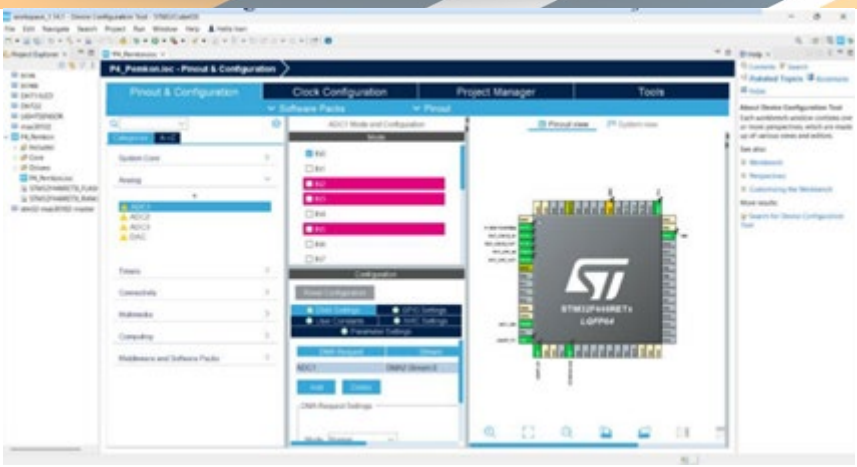
6. Klik yes dan tunggu hingga proses project creation selesai



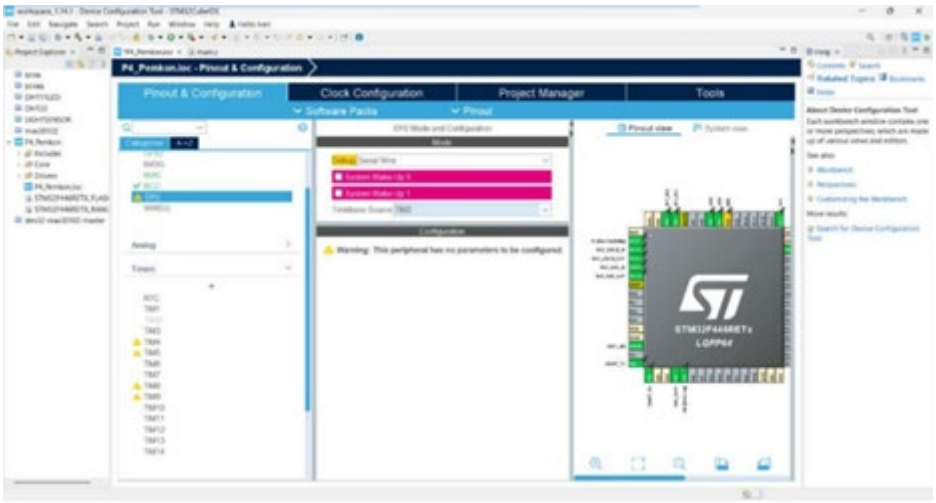
7. Pilih menu Analog, pilih ADC1 dan centang IN0.



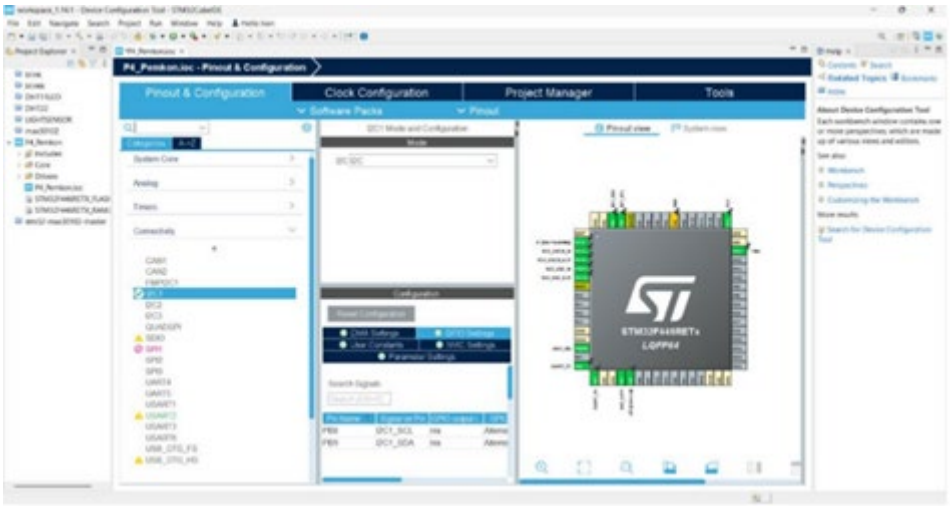
8. Pilih menu DMA setting dan add ADC1 dan ubah mode menjadi circular



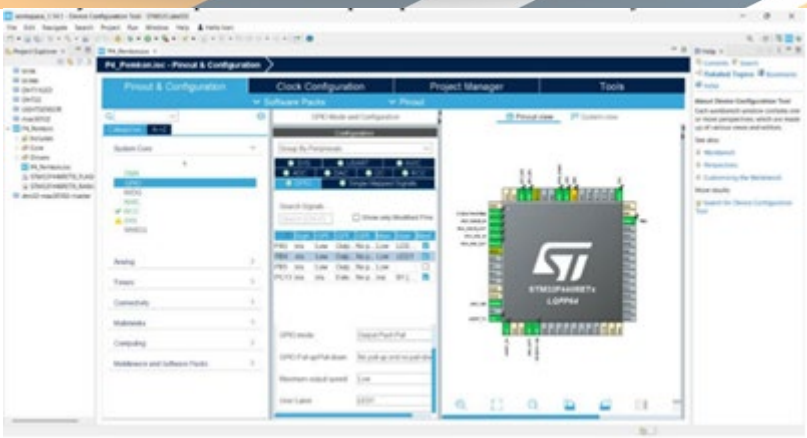
9. Pilih menu SYS kemudian pada Timebase Source pilih TIM2



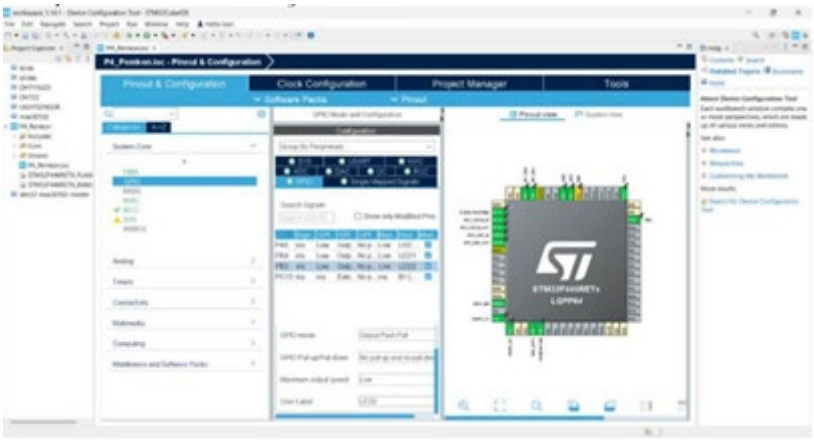
10. Pilih menu connectivity pilih I2C1 dan pilih I2C, kemudian ubah pin SDA SCL yang digunakan yang awalnya PB6 dan PB7 menjadi PB8 dan PB9 di bagian pinout view



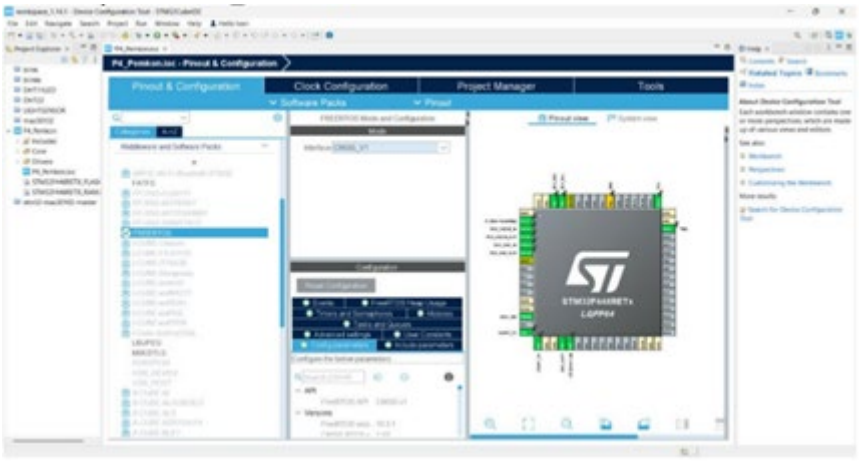
11. Kemudian pada PB4 dan PB5 ubah pin menjadi GPIO_Output, kemudian pada menu System Core pilih GPIO dan pada pin PB4 ubah label menjadi LED1



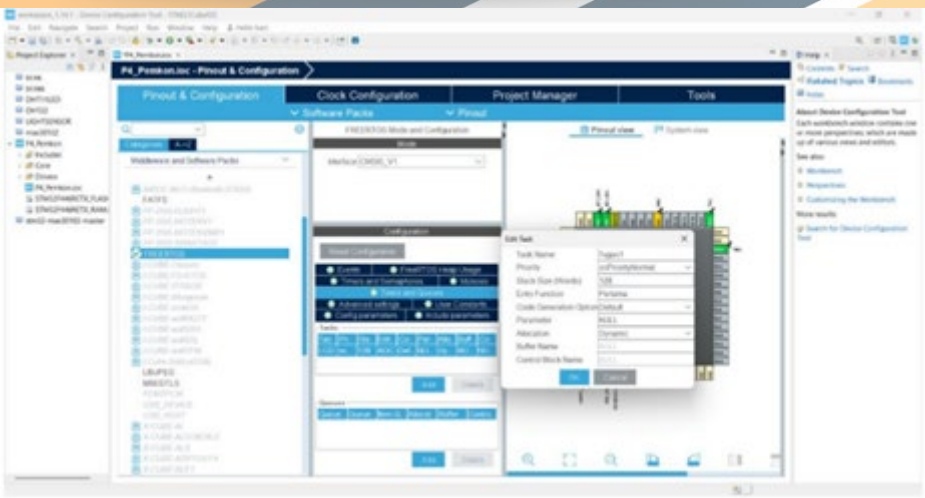
12. Pada pin PB5 ubah label menjadi LED2



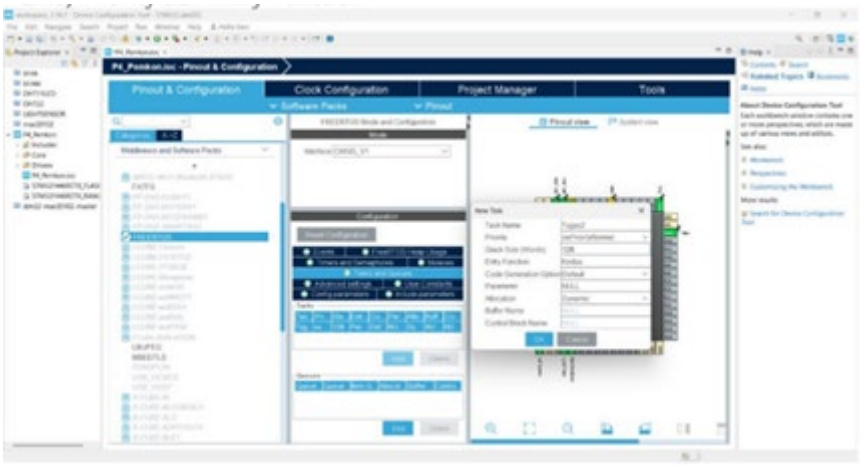
13. Pilih menu Middleware and Software Packs pilih FREERTOS, kemudian pada menu interface pilih CMSIS_V1



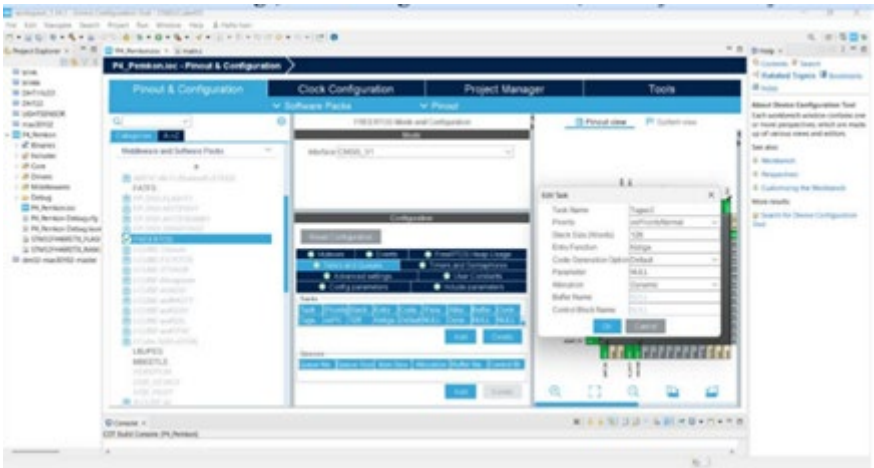
14. Pada configuration klik menu Task and Queues, pada menu Task klik 2x pada default task, kemudian ganti Task Name dan Entry Function



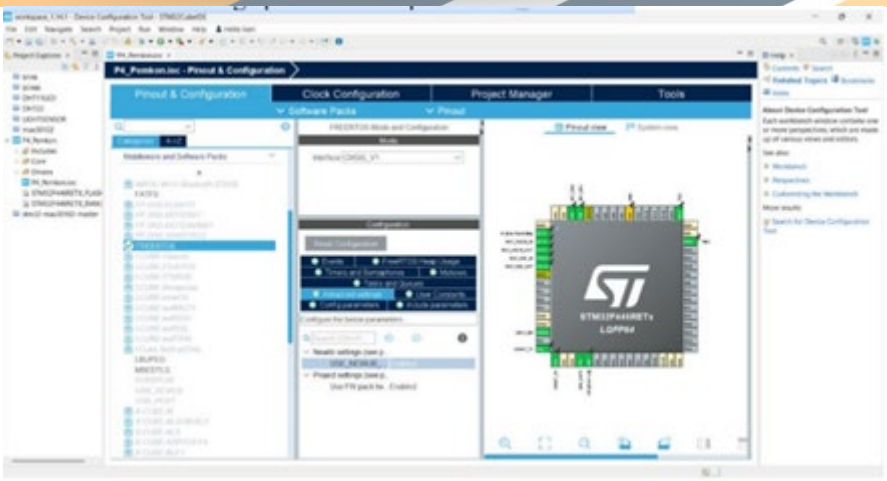
15. Kemudian pada menu Task klik Add untuk menambahkan New Task, ganti Task Name, Priority dan Entry Function



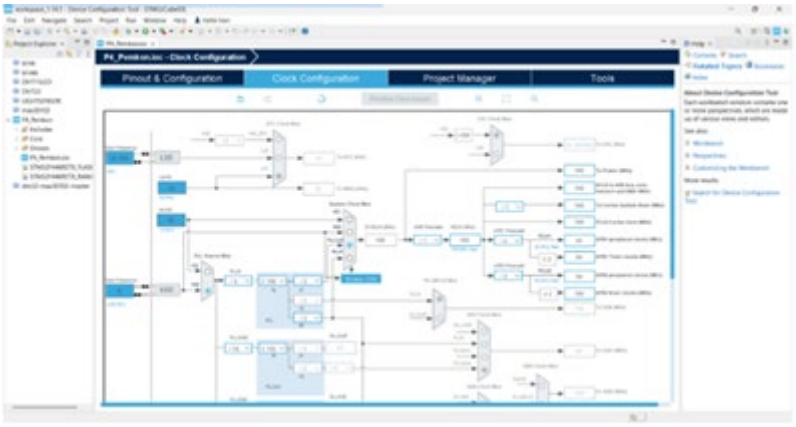
16. Tambahkan task baru lagi, kemudian ganti Task Name, Priority dan Entry Function



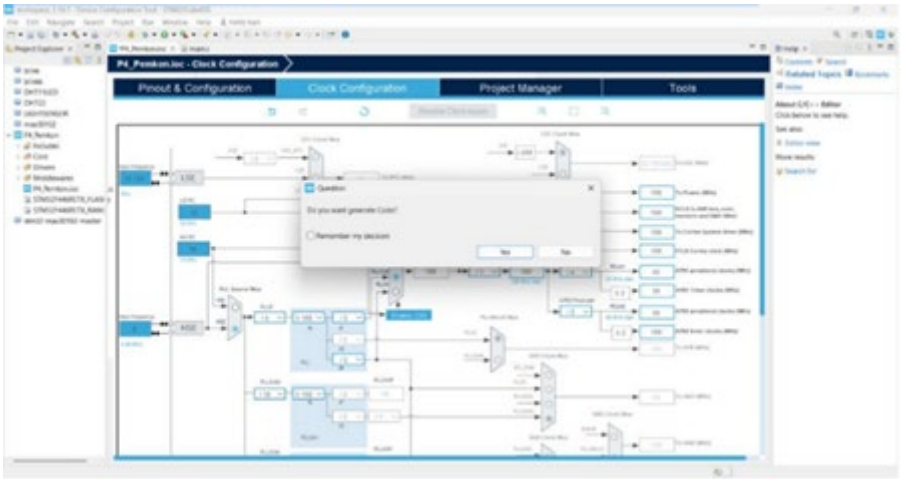
17. Pada Advanced settings pilih enabled pada USE_NEWLIB



18. Buka clock configuration pindah pilihan PLL source yang awalnya di HS menjadi HSE serta ubah clock max menjadi 180



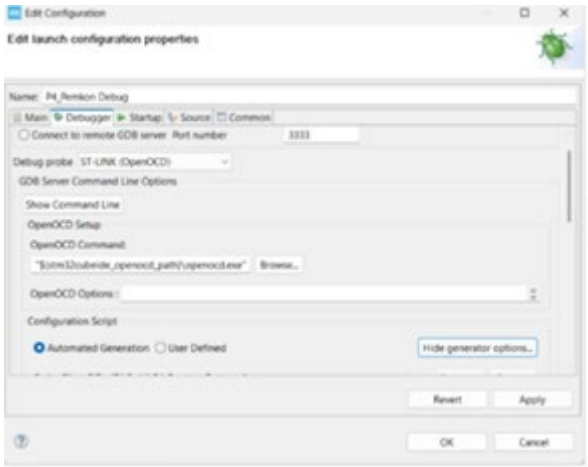
19. Setelah selesai mengatur semuanya, klik save dan pilih generate code



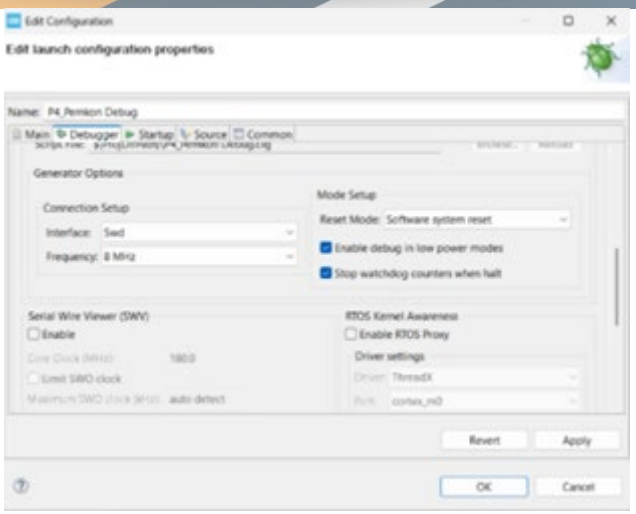
20. Pada main.c masukkan kode berikut

```
 * @retval None
 */
/* USER CODE END Header_Kedua */
void Kedua(void const * argument)
{
  /* USER CODE BEGIN Kedua */
  /* Infinite loop */
  for(;;)
  {
          // Menyalakan LED hijau
          HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_4);
//          HAL_Delay(500);
    // Menunda selama 1 detik
    osDelay(500);
  }
  /* USER CODE END Kedua */
}

/* USER CODE BEGIN Header_Ketiga */
/**
 * @brief Function implementing the Tugas3 thread.
 * @param argument: Not used
 * @retval None
 */
/* USER CODE END Header_Ketiga */
void Ketiga(void const * argument)
{
  /* USER CODE BEGIN Ketiga */
  /* Infinite loop */
  for(;;)
  {
      // Menyalakan LED hijau
          HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
//          HAL_Delay(1000);
    // Menunda selama 1 detik
    osDelay(1000);
  }
  /* USER CODE END Ketiga */
}

/**
  * @brief  Period elapsed callback in non blocking mode
```

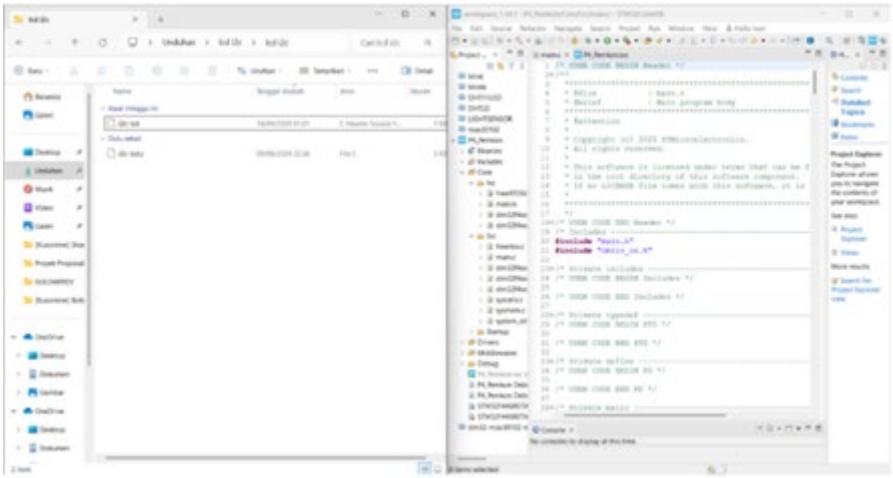
21. Kemudian build project dengan gambar palu dan tunggu hingga selesai dan tidak ada error
22. Kemudian klik debug project dengan gambar kumbang, dan atur Debug probe menjadi ST-LINK (OpenOCD) dan hide generator options



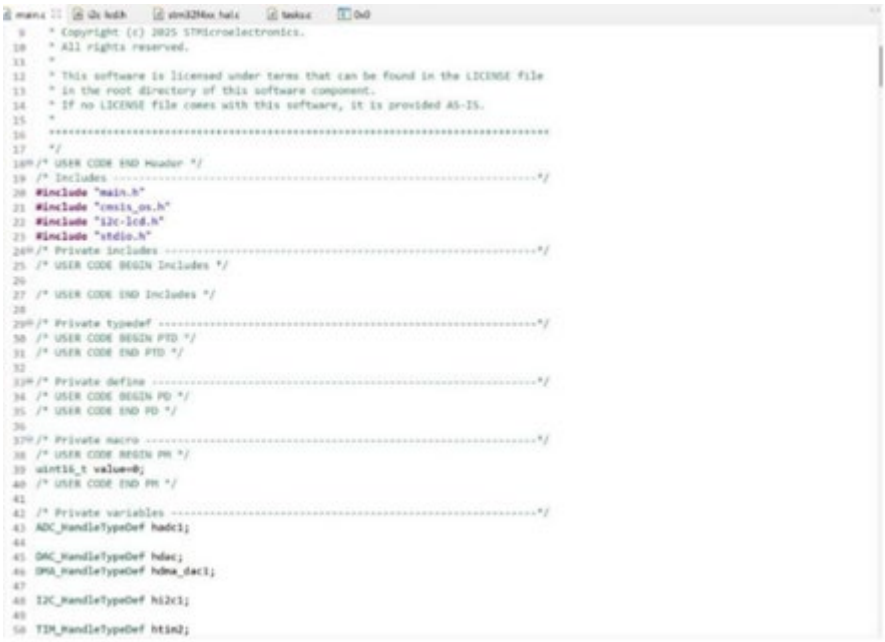
23. Kemudian scroll kebawah ubah Reset Mode menjadi Software system reset, kemudian klik Apply > OK



24. Untuk percobaan kedua tambahkan library I2C dengan cara mencopy atau mendrag file i2c-lcd.h ke folder Inc dan file i2c-lcd.c ke folder Src



25. Pada main.c tambahkan include I2C dan stdio.h, dan tambahkan uint16_t value=0



26. Kemudian tambahkan kodingan pada user code begin 2

27. Kemudian tambahkan kodingan pada void pertama

Tahap Pasca Praktikum

1. Praktikan mengetahui apa itu FreeRTOS dan dapat menjelaskan arsitektur dan tujuannya dalam sistem embedded.
2. Mengetahui apabila saat LED dan LCD berjalan dalam *task* terpisah, apa yang terjadi jika salah satu *task* memiliki *delay* lebih lama?
3. Mengetahuai mengapa pada menu konfigurasi SYS bagian Time Source diganti menjadi TIM2 tidak menggunakan default Systick

DAFTAR PUSTAKA

[1] Kurikulum Prodi Sarjana Terapan Rekayasa Teknologi Instrumentasi
[2] Modul HYSYS. Perancangan Sistem Kontrol.

LAMPIRAN

Lampiran 1. Safety Induction

A. Identifikasi Bahaya dan Pendendalian Risiko

1. Bahaya Umum
    - Paparan layar komputer dalam durasi panjang yang dapat menyebabkan kelelahan mata.
    - Risiko kehilangan data akibat kesalahan penyimpanan atau gangguan perangkat lunak.
    - Bahaya listrik dari perangkat keras seperti komputer dan adaptor.
    - Ketidaktepatan hasil analisis akibat kesalahan input parameter sistem kontrol.
2. Pengendalian Risiko
    - Gunakan Alat Pelindung Diri (APD) yang sesuai, seperti jas laboratorium dan sepatu tertutup.
    - Pastikan komputer, jaringan, dan perangkat lunak (Hysys & Matlab) berfungsi dengan baik sebelum digunakan.
    - Lakukan *double-check* terhadap input parameter dalam simulasi sistem kontrol untuk mencegah kesalahan proses.
    - Hindari menyalakan atau mematikan perangkat tanpa instruksi dari supervisor atau asisten.
    - Pastikan posisi duduk ergonomis dan atur jeda penggunaan komputer untuk menghindari kelelahan.

B. Prosedur Keamanan Darurat

1. Kebakaran
    - Tekan tombol alarm kebakaran terdekat.
    - Gunakan APAR yang sesuai (CO_2 untuk kebakaran akibat perangkat elektronik).
    - Lakukan evakuasi melalui jalur darurat, berkumpul di titik kumpul yang telah ditentukan.
2. Kecelakaan atau Cedera
    - Lakukan pertolongan pertama jika aman.
    - Hubungi petugas medis kampus atau layanan darurat jika diperlukan.
    - Segera laporkan kepada dosen/instruktur praktikum.
3. Gangguan atau Kerusakan Peralatan
    - Hentikan penggunaan komputer atau perangkat lunak jika terjadi gangguan/error.
    - Laporkan ke teknisi atau asisten laboratorium.
    - Tidak diperkenankan mencoba memperbaiki software atau hardware tanpa izin.

C. Penggunaan Peralatan Laboratorium

- Baca panduan penggunaan Hysys dan Matlab sebelum melakukan simulasi.

- Pastikan parameter input sistem kontrol (misal: *gain*, *delay*, *PID settings*) sesuai dengan skenario yang ditentukan.
- Jangan mengubah konfigurasi software tanpa sepengetahuan dosen/asisten.
- Simpan hasil kerja secara berkala di lokasi aman (cloud, flashdisk).
- Jangan meninggalkan komputer dalam kondisi menyala dan aktif tanpa pengawasan.
- Setelah selesai, tutup aplikasi sesuai prosedur dan matikan komputer jika diperintahkan.

Lampiran 2. Precautions

Lampiran 3. Job Safety Analysis

	JOB SAFETY ANALYSIS	NO. DOC	JSA/DOC/04/25
		NO. REVISI	1
		TANGGAL EFEKTIF	28 April 2025
		Hal	1 of 1

No. JSA : NRP-P1-PSKP

Nama Pekerjaan: (Judul Praktikum

User : (Nama Praktikan)

APD : 1. Helm Safety
2. Sepatu Safety
3. Masker
4. Sarung Tangan
5. Seragam Lengan Panjang

Lokasi	**Urutan Pekerjaan**	**Potensi Bahaya**	**Upaya Pengendalian**

Diketahui oleh

Laboran Laboratorium I&C

( )

Lampiran 4. Permit to Work

	INSTRUMENTATION AND CONTROL LABORATORY	

PERMIT TO WORK FOR AREA A

Doc No.	Rev.	*Healthy Safety Environment*	Page 1 of..

PERMIT TO WORK *Surat Izin Kerja*	***Date*** / Tanggal : ***Facility*** / Fasilitasl : ***No of work Permit*** :

• **Hot Work** Pekerjaan Panas	• **Cold Work** Pekerjaan Dingin	• **Enegry Isolation** Isolasi Energi	• **Confined Space** Bekerja di ruang terbatas	• **Critical Lift** Pengankatan Kritis

I. Description / Location of Job / Uraian/ Lokasi Pekerjaan

.....

.....

.....

No. of Worker (NRP)		**II. Validity of Work Permit / Masa Berlaku Izin Kerja**
Work Permit Applicant / Pemohon		**Date / Tanggal** From .....hrs To.....hrs
Work Permit Holder / Pemegang Lain Kerja		**Sign /** Tanda Tangan **Authorized Person /** Penanggung jawab
3. Isolation / Isolasi	**Electrical** **Mechanical**	**Check if applicaable and explain below** Jelaskan mohon dibawah ini :

.....

.....

4. Work of Permit Requirement/ Pernyataan lain Kerja	• **JSA & Risk Assesment** • **Work Method**	• **Drawing Activity** • **As Build Drawing**	• **Rigging Plan** • **Underground Composite**

5. Hazards **(Check The hazard and Required Precaution /** Beri Tanda Pada Bahaya dan Pencegahannya)

• Electrical Hazard • Working at Haight	• Chemical Hazard • Fire and Explotion Hazard • Mechanical Hazard • Slips, trips, and Fails	• Noise Hazard • Pressure hazard • Radiation Hazard • Biological Hazard	• Hot Work Hazard • Confined Space Hazard

Other Hazard/ Bahasa Lainnya
Precautions :
☐ Keep Hands Dry ☐ Permission before operating ☐ Don't touch cables or panels without approval ☐ Avoid flammable items ☐ No food or drinks ☐ Wear safety shoes
Other Precautions / Bahaya Lainnya

Lampiran 5. Minutes of Meeting (MOM)

Subject	
Vanue, Date & Time	
Attendee:	Division
Summary of Discussion	
Actions Points	

Lampiran 6. Format Laporan Praktikum

COVER
REVISION HISTORICAL SHEET
DAFTAR ISI
DASAR TEORI

- Berisi teori-teori yang relevan dengan topik praktikum.
- Setiap teori harus dijelaskan dengan bahasa ilmiah dan didukung oleh sumber referensi terpercaya.
- Penulisan kutipan atau referensi di dalam teks menggunakan **APA Style**.
- Paragraf rata kanan-kiri (**justify**), **spasi 1,15**, dan **font Times New Roman ukuran 12**.

2. METODE PERCOBAAN

- Jelaskan langkah-langkah praktikum secara sistematis dan urut.
- Sertakan informasi alat dan bahan yang digunakan.
- Format penulisan sama seperti poin sebelumnya (justify, spasi 1,15, font TNR 12).

3. HASIL PERCOBAAN

- Menyajikan data hasil pengamatan secara objektif.
- Gunakan tabel atau grafik jika diperlukan untuk memperjelas data.
- Setiap tabel/grafik diberi nomor dan judul.
- Penjelasan data ditulis di bawah tabel/grafik dengan format standar (justify, spasi 1,15, font TNR 12).

4. ANALISA

- Uraikan hasil praktikum berdasarkan teori yang telah dijelaskan di bagian dasar teori.
- Sertakan analisis penyebab hasil yang diperoleh, serta kemungkinan kesalahan dan faktor lain yang memengaruhi.
- Tulis dengan paragraf rapi (justify), spasi 1,15, dan font TNR 12.

5. KESIMPULAN

- Tulis poin-poin penting dari hasil analisis secara ringkas dan jelas.
- Hindari memasukkan informasi baru yang belum dibahas sebelumnya.
- Format penulisan tetap sama (justify, spasi 1,15, font TNR 12).

DAFTAR PUSTAKA

- Memuat semua sumber yang dirujuk dalam laporan.
- Ditulis sesuai dengan **APA Style** (contoh: buku, jurnal, dan situs web ilmiah).
- Diurutkan secara alfabetis berdasarkan nama belakang penulis pertama.
- Spasi 1,15, font Times New Roman 12, paragraf justify.

Link Kebutuhan Praktikum: