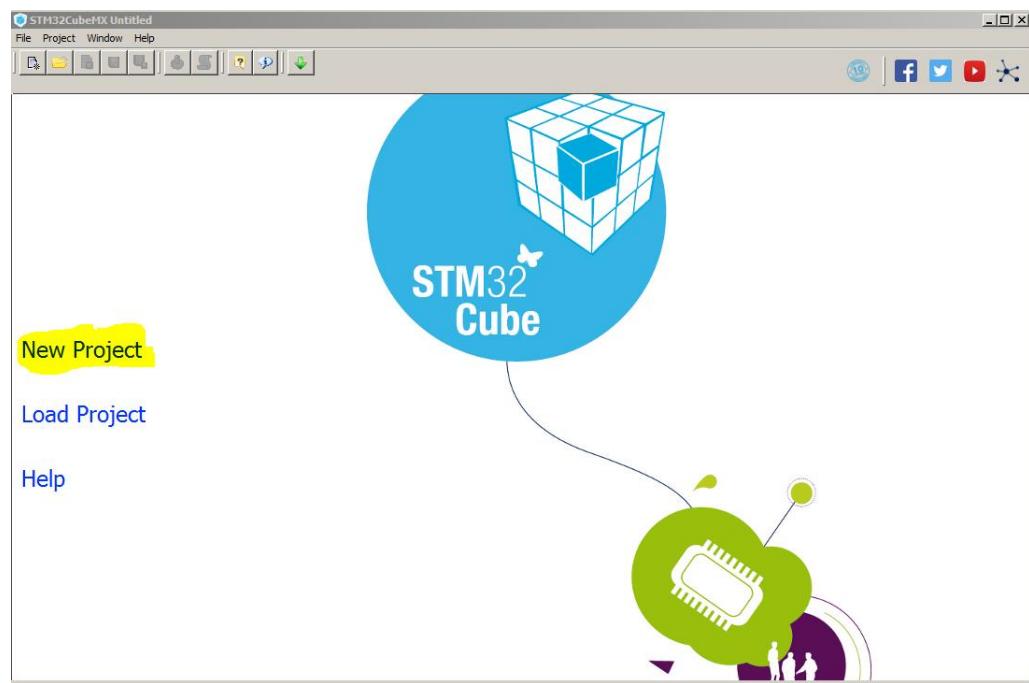
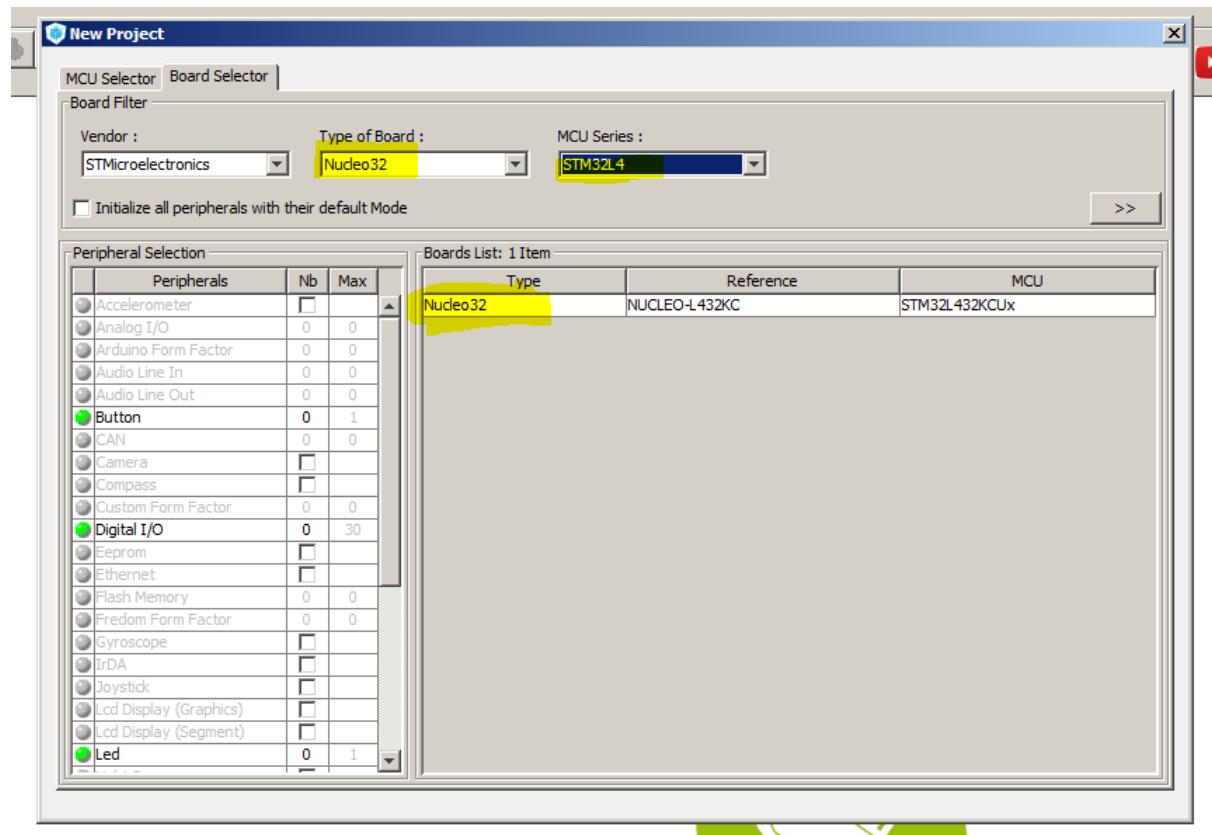


STM32L04 – Prva vaja (Blinky – utripanje LED)

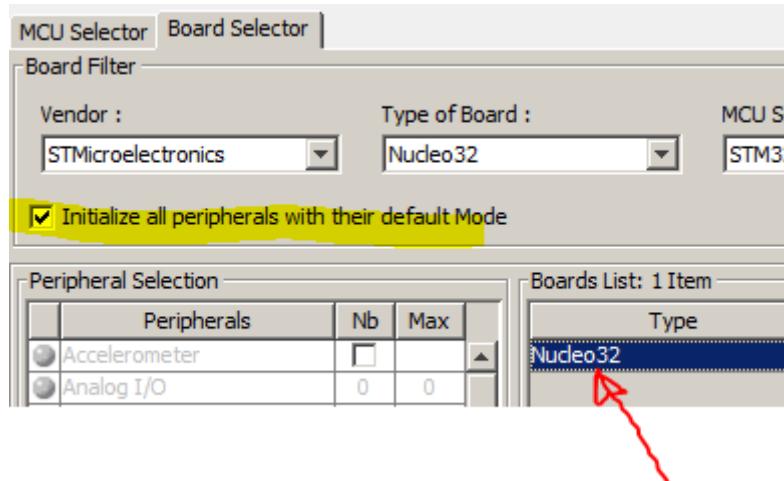
Poženi STM32 Cube MX in klikni na **New Project**



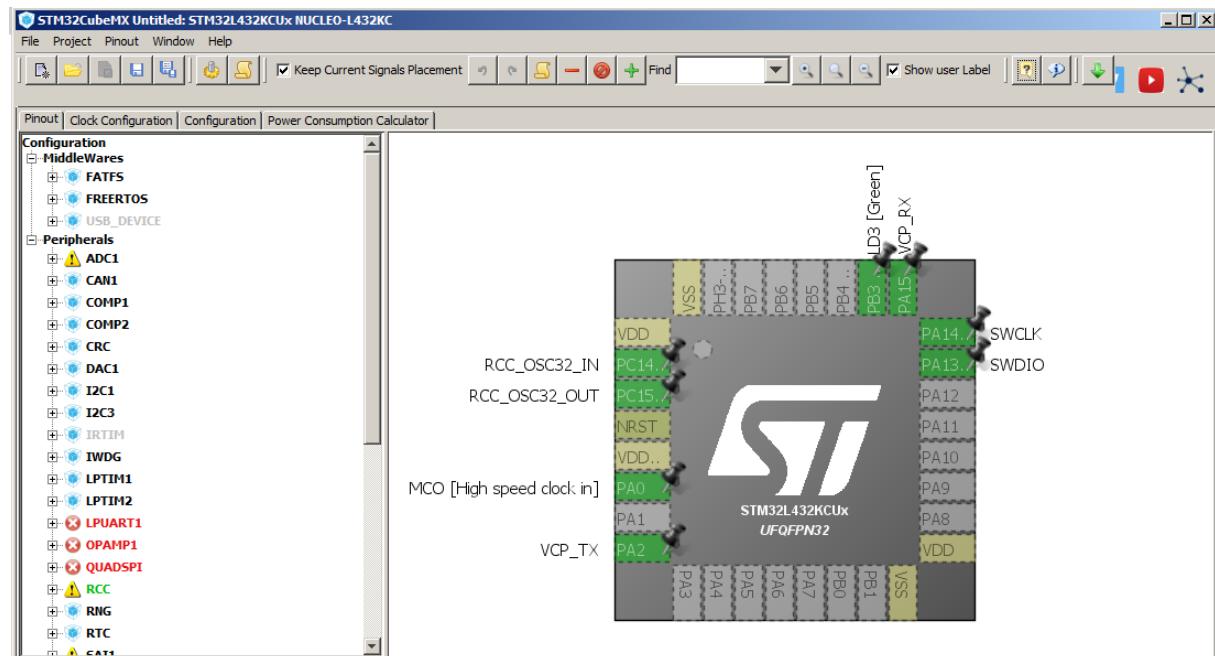
Izberi zavihek **Board Selector**, pod Type of board izberi **Nucleo 32**, in MCU Series izberi **STM32L4**



Obkljukaj še opcijo **Initialize all peripherals with their default Mode** in potrdi z dvoklikom na desni v seznamu »Board List« **NUCLEO-L432KC**



Pojavi se predogled razporeda priključkov na mikrokontrolerju (Pinout).



Za nadaljevanje moraš vedeti, kaj je kje priključeno na sami ploščici. Potrebuješ oznako GPIO priključka, kjer je priključena LED

Odpriš dokumentacijo na naslovu:

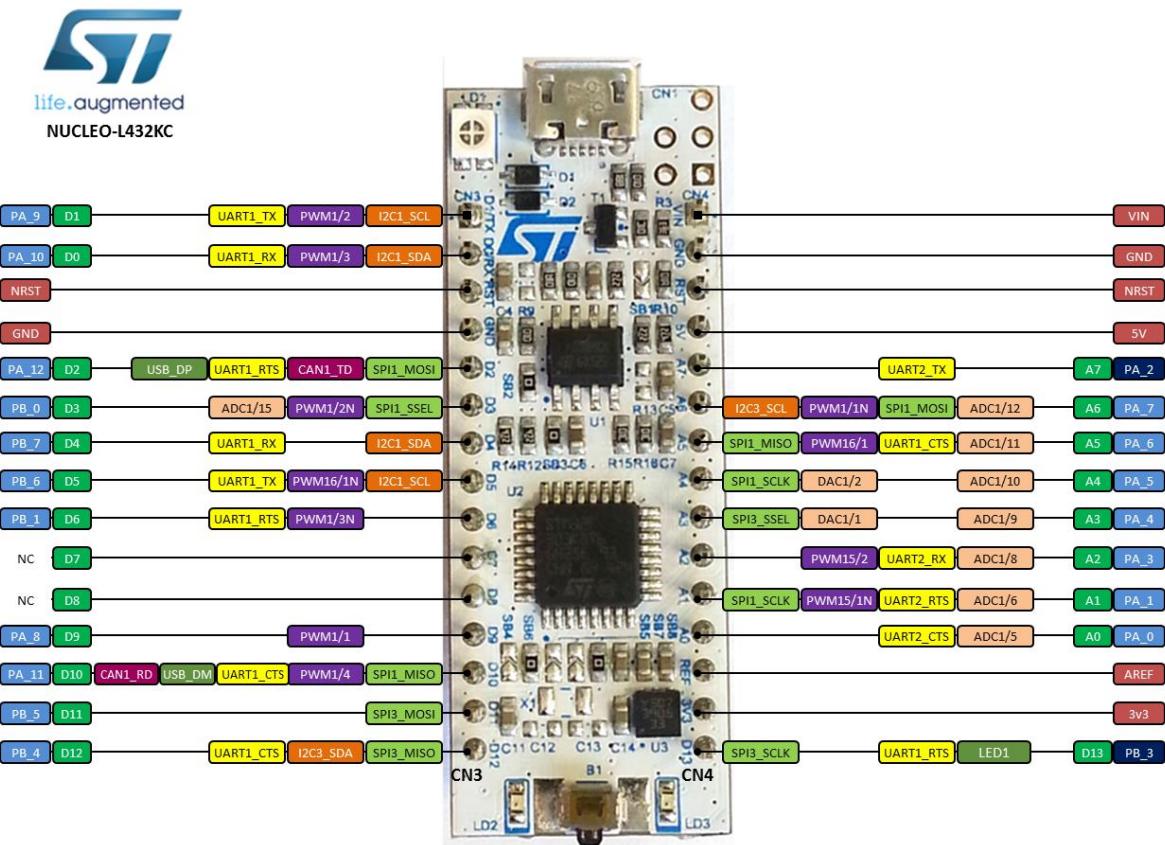
http://www.st.com/content/ccc/resource/technical/document/user_manual/e3/0e/88/05/e8/74/43/a0/DM00231744.pdf/files/DM00231744.pdf/jcr:content/translations/en.DM00231744.pdf

ali <http://goo.gl/jGR5RW>

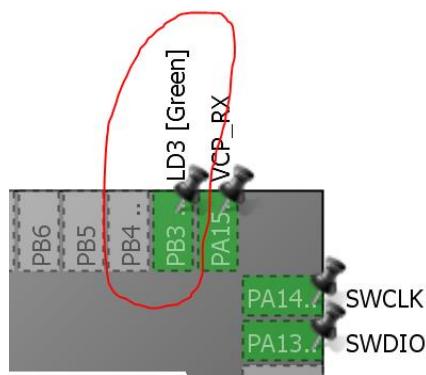
in poiščeš v poglavju 6.5, kjer piše:

User LD3: the green LED is a user LED connected to Arduino Nano signal D13 corresponding to the STM32 I/O PB3 (pin 26)

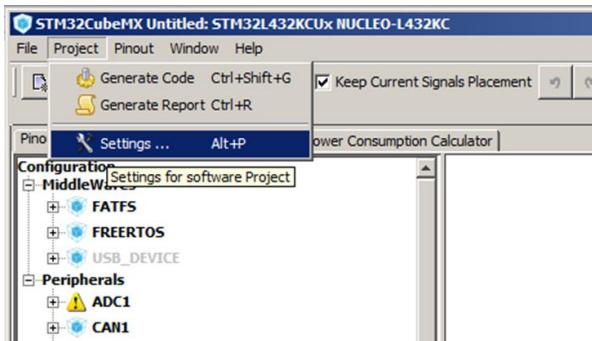
Zelena LED je torej na GPIO portu PB3. Pomagaš si lahko tudi s tole pregledno shemo, kjer je desno spodaj isti podatek:



Na pinout-u vidiš, da je ta signal že označen z **LD3 [Green]**

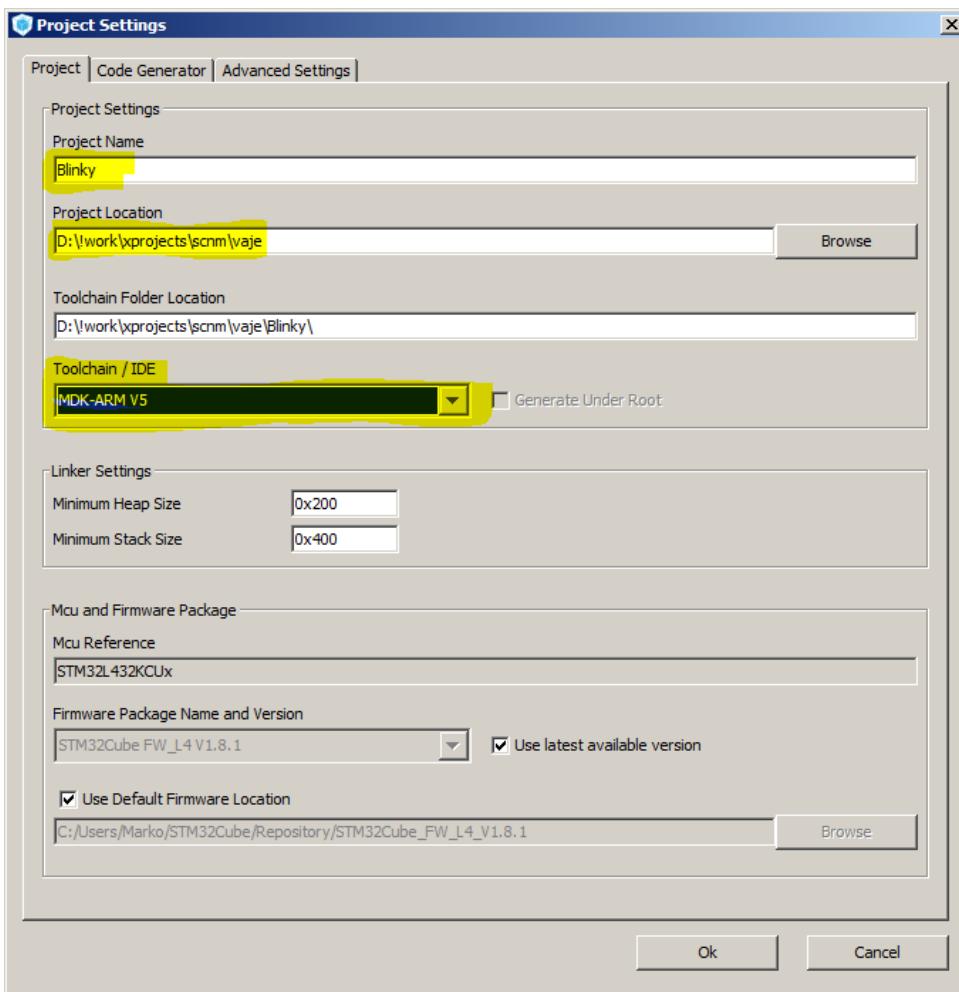


V Cube MX greš v menu **Project** in izbereš **Settings**:



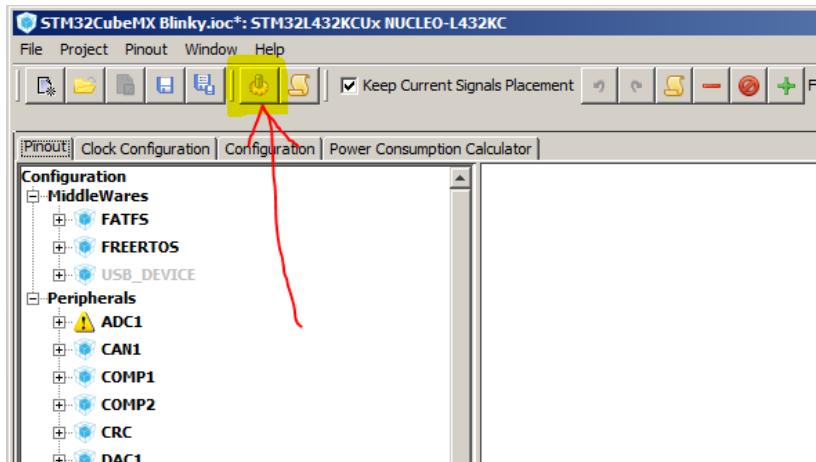
Pojavi se okno z opcijami za tvoj nov projekt, kjer moraš vnesti tri stvari:

1. Ime projekta (**Project name: Blinky**)
2. Mesto na disku, kjer boš imel shranjen projekt (**Project Location**: izbereš en direktorij na disku)
3. Izberi razvojno orodje (**Toolchain/IDE: MDK-ARM V5**)



Pritisni **OK**, da potrdiš nastavitev.

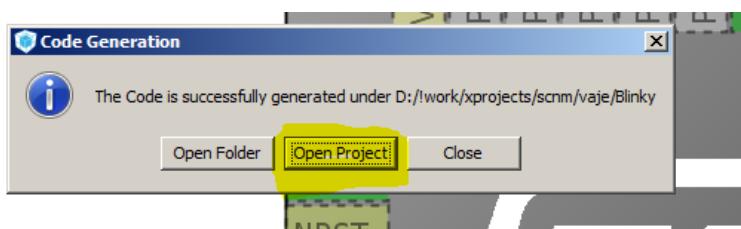
Sedaj klikni na ikono za generiranje kode tvojega novega projekta:



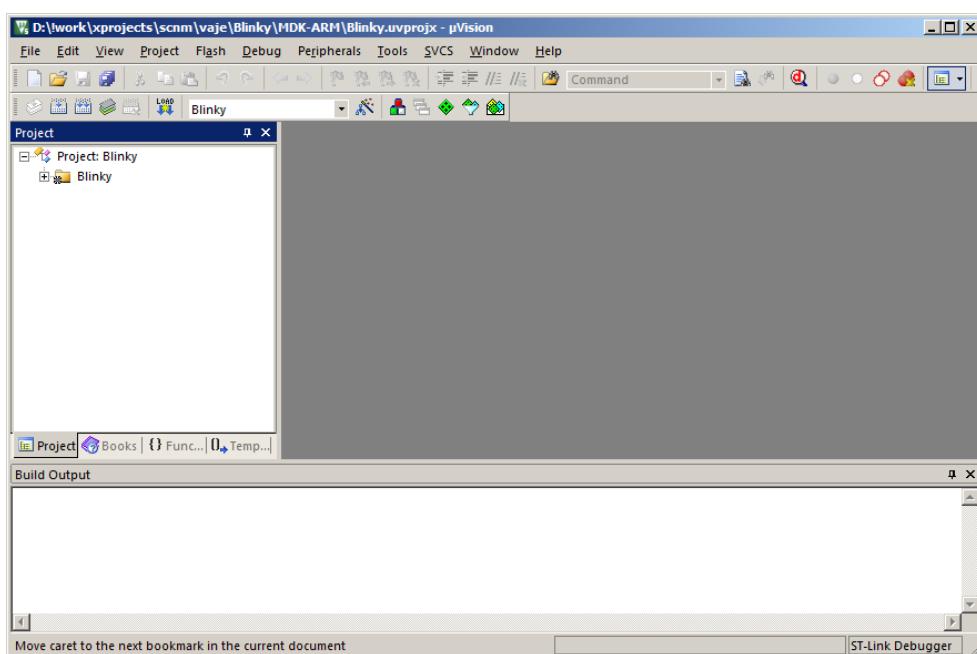
Ikona je simbol zobnika (pete z leve):



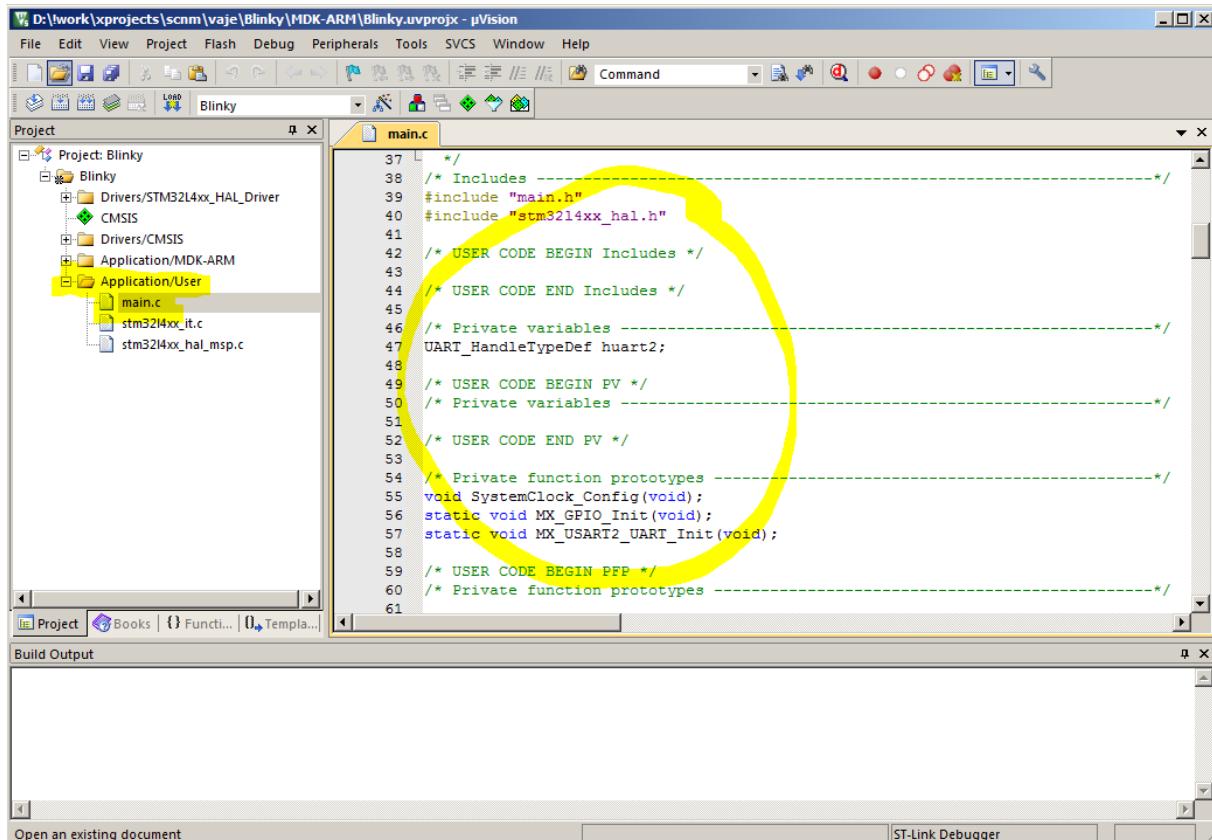
Pojavi se pogovorno okno, kjer te sprašuje, kako naprej. Izberi srednji gumb: Open Project



Odpalo se bo novo okno z uVision razvojnim okoljem:



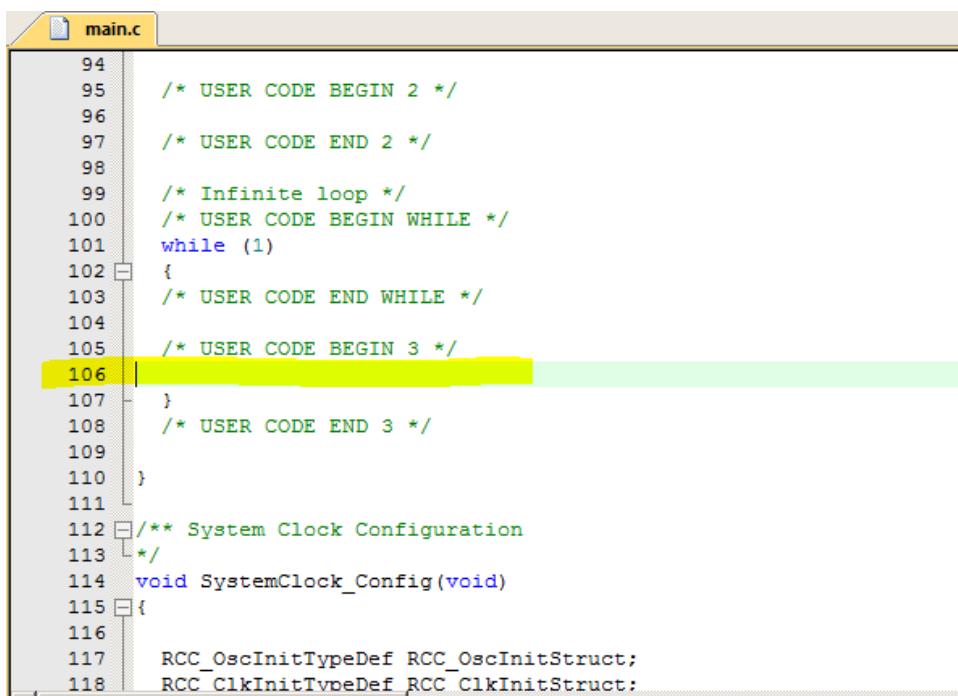
Na levi strani je struktura projekta z vsemi datotekami. Drevo razširiš s pritiskom na »+«. Izberi skupino Application/User in dvakrat klikni na main.c, da se na desni strani v urejevalniku pojavi izvorna koda te datoteke main.c:



The screenshot shows the pVision IDE interface. On the left, the Project Explorer window displays the project structure for 'Blinky'. The 'Application/User' folder is selected and highlighted with a yellow circle. Inside it are 'main.c' and two other source files. The main workspace shows the content of 'main.c'. The code is annotated with several dashed green boxes and a large yellow circle highlighting specific sections. The code itself includes comments like /* USER CODE BEGIN */ and prototypes for functions like SystemClock_Config.

```
37  */
38  /* Includes -
39  #include "main.h"
40  #include "stm32l4xx_hal.h"
41
42  /* USER CODE BEGIN Includes */
43
44  /* USER CODE END Includes */
45
46  /* Private variables -
47  UART_HandleTypeDef huart2;
48
49  /* USER CODE BEGIN PV */
50  /* Private variables -
51
52  /* USER CODE END PV */
53
54  /* Private function prototypes -
55  void SystemClock_Config(void);
56  static void MX_GPIO_Init(void);
57  static void MX_USART2_UART_Init(void);
58
59  /* USER CODE BEGIN PFP */
60  /* Private function prototypes -
61
```

Poskrolaj do vrstice 106, kjer je koda:



The screenshot shows a code editor window for 'main.c'. Line 106 is highlighted with a yellow background and a green selection bar. The code on line 106 is empty, indicated by three dots (...). The rest of the code is visible, including the infinite loop and system clock configuration.

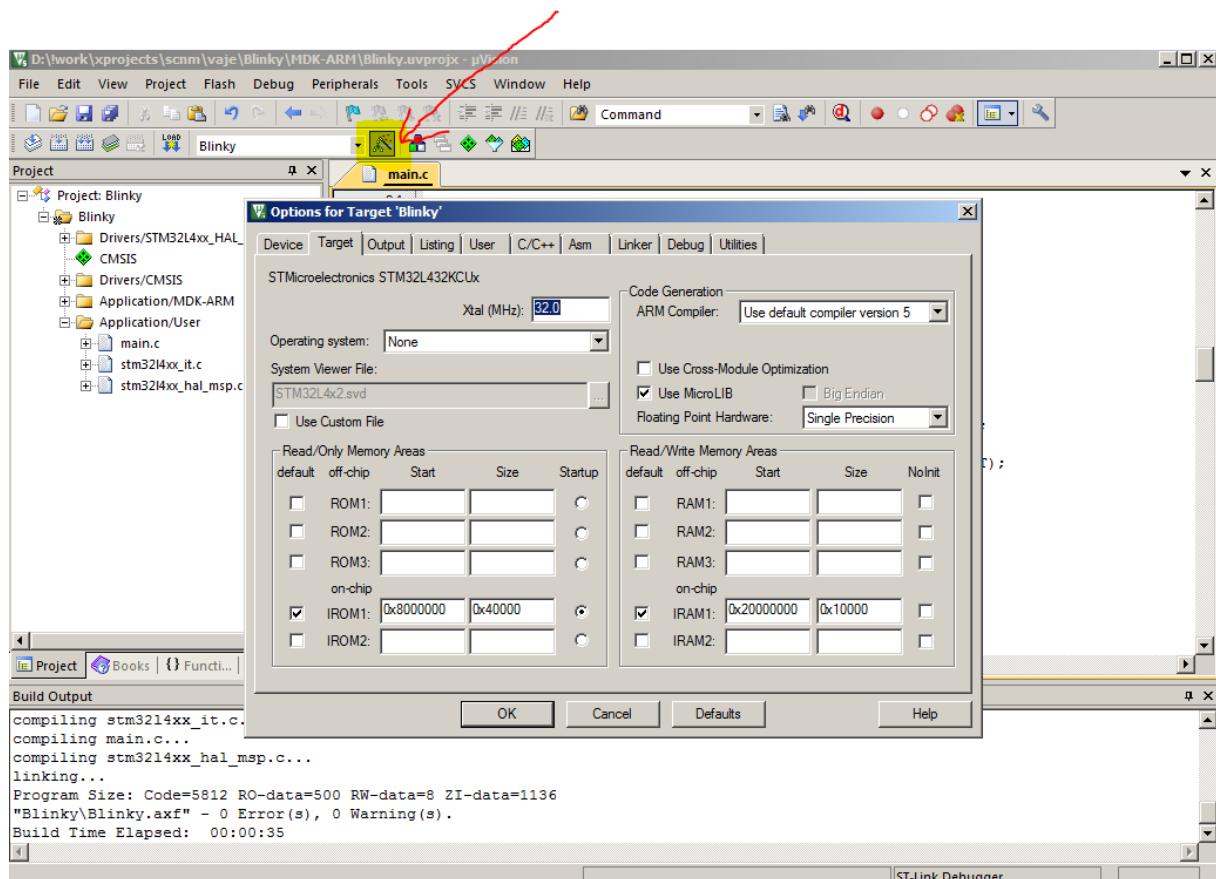
```
94
95  /* USER CODE BEGIN 2 */
96
97  /* USER CODE END 2 */
98
99  /* Infinite loop */
100 /* USER CODE BEGIN WHILE */
101 while (1)
102 {
103  /* USER CODE END WHILE */
104
105  /* USER CODE BEGIN 3 */
106  ...
107  /* USER CODE END 3 */
108
109
110 }
111
112 /** System Clock Configuration
113 */
114 void SystemClock_Config(void)
115 {
116
117  RCC_OscInitTypeDef RCC_OscInitStruct;
118  RCC_ClkInitTypeDef RCC_ClkInitStruct;
```

Dodaj naslednje 4 vrstice kode:

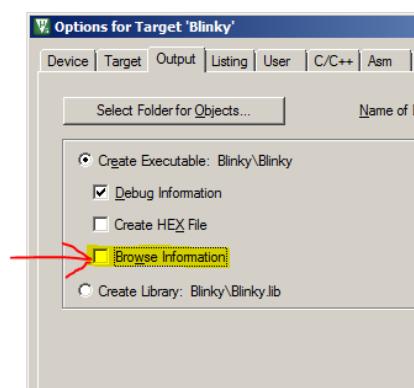
```
105  /* USER CODE BEGIN 3 */
106      HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
107      HAL_Delay(200);
108      HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
109      HAL_Delay(200);
110  }
111  /* USER CODE END 3 */
```

Sedaj priklopi razvojno ploščico na USB s pomočjo micro-USB kabla.

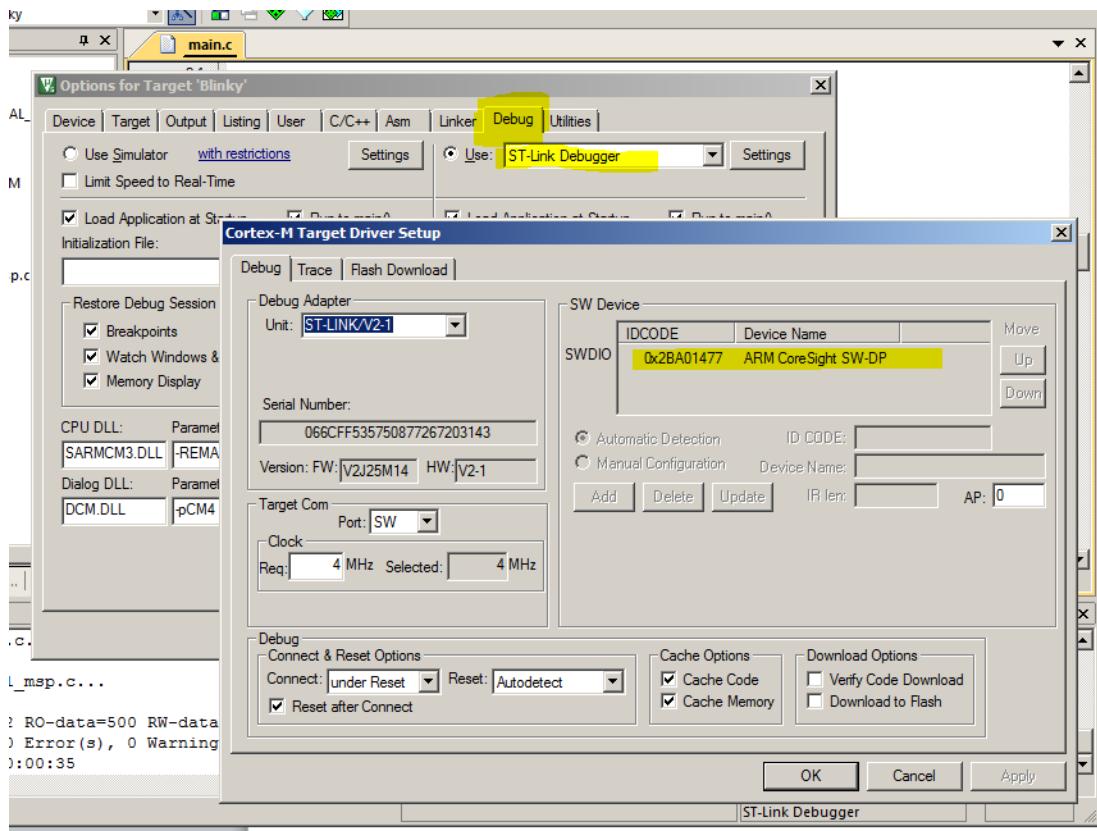
V uVision IDE-ju nastavi opcije projekta tako, da klikneš na »čarobno paličico«:



Pojavi se okno. Izberi zavihek »Output« in pri Browse information zbrisi kljukico:

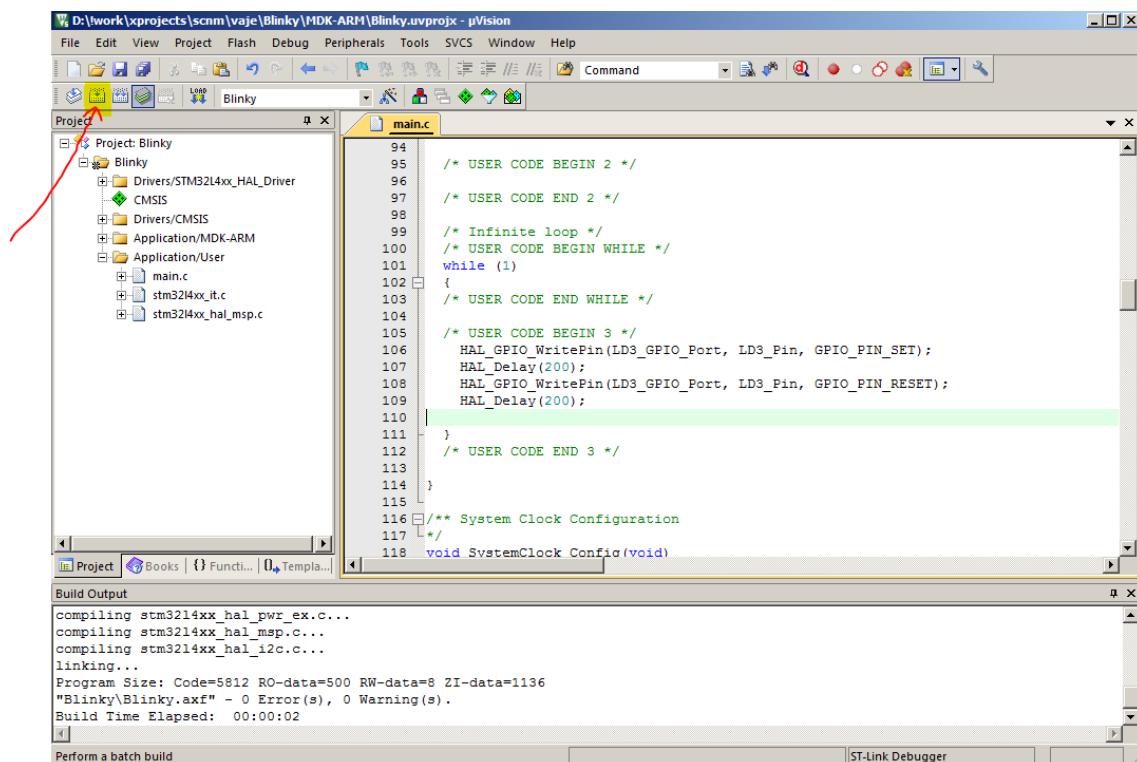


Izberi še zavihek »Debug« in preveri, da je izbran (Use) debugger ST-Link Debugger. Klikni na Settings in preveri, da na desni pri SW Device piše ARM CoreSight SW-DP.



S tem preveriš, da imaš vse pravilno nameščeno in da povezava s ploščico deluje.

Potrdi z OK in še enkrat OK, da se zapre okno z opcijami projekta. Sedaj lahko prevedeš svoj projekt tako, da pritisneš na ikono »Build« ali pritisneš tipko F7.



V spodnjem delu okna je panel Build output, kjer mora pisati

"Blinky\Blinky.axf" - 0 Error(s), 0 Warning(s).

```
compiling stm3214xx_hal_pwr_ex.c...
compiling stm3214xx_hal_msp.c...
compiling stm3214xx_hal_i2c.c...
linking...
Program Size: Code=5812 RO-data=500 RW-data=8 ZI-data=1136
"Blinky\Blinky.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:02
```

To pomeni, da je program preveden brez napak in da bo deloval tako, kot si napisal v kodi.

Sedaj lahko svoj program naložiš v pomnilnik mikrokontrolerja s pritiskom na ikono »Load« ali pa na tipko F8.

```
"Blinky\Blinky.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:02
Load "Blinky\\Blinky.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 20:11:36
```

LED seveda ne utripa, ker se program še ni zagnal, ampak je samo naložen. Pritisni tipko Reset (na nasprotni strani od USB konektorja). Sedaj mora LED utripati.

Čestitke! Napisal si svoj prvi program za 32 bitni ARM mikrokontroler.

Kaj lahko še na hitro poskusiš?

Za začetek spreminja vrednosti v vrstici

HAL_Delay(200);

in opazuj, kaj se zgodi. Ne pozabi prevesti (F7) in naložiti (F8) programa, ko spremeniš vrednosti v urejevalniku.

Da se program prične samodejno izvajati lahko nastaviš v opcijah projekta (Čarobna paličica), kjer v zavihku Debug nastaviš, da se po nalaganju sam resetira in požene:

