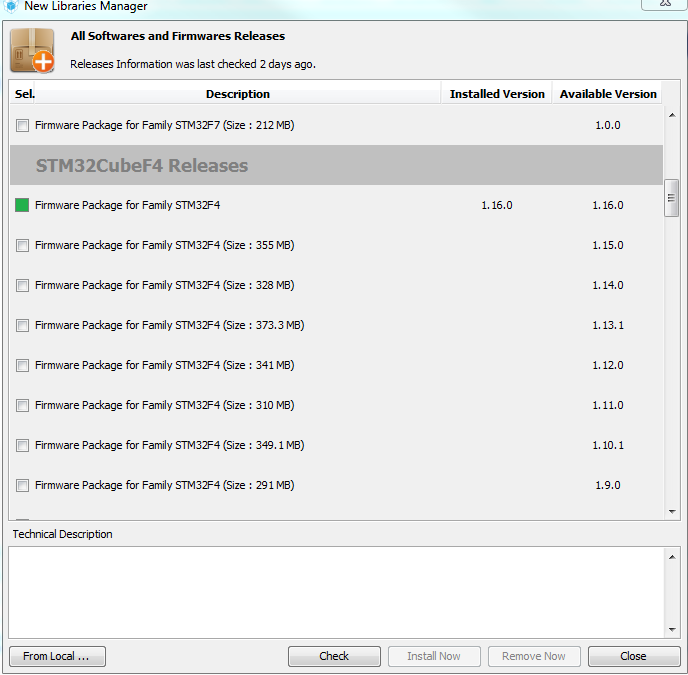
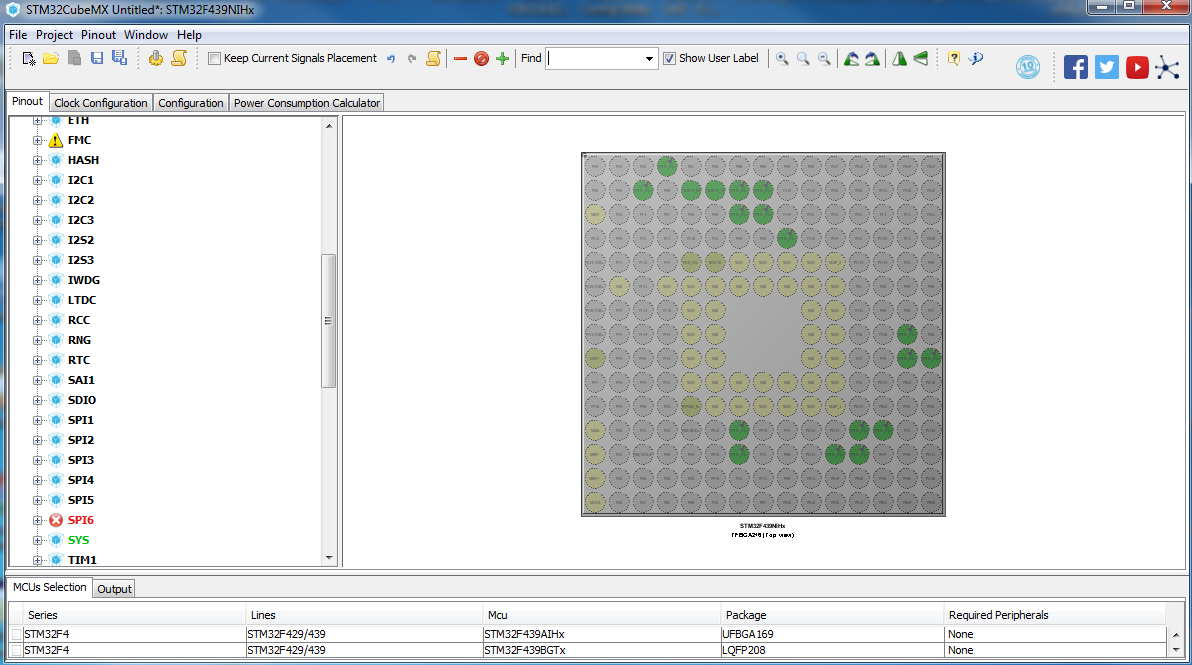
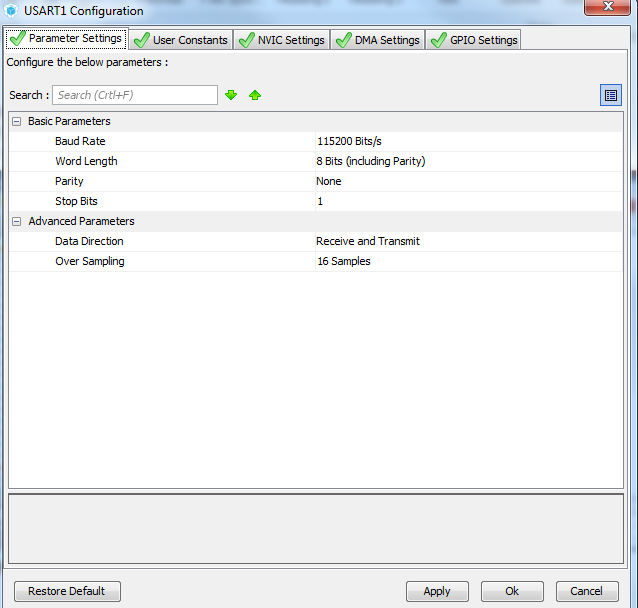
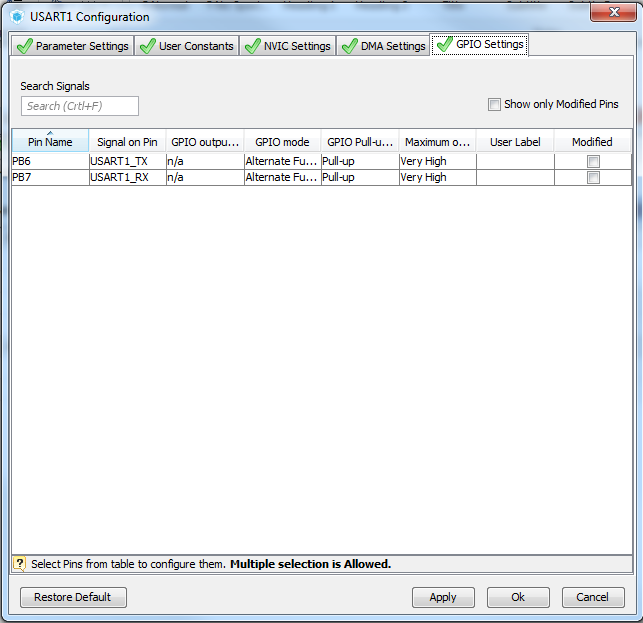
1. Nakon instalacije STM32CubeMX softverskog paketa download-ovan je STM32CubeF4 1.16.0
2. 
3. Odabran je čip STM32F439NIHx sa BGA kućištem.
4. Enable-ovan je FreeRTOS I odabrana je samo USART1 periferija u asinhronom modu, preko koje će ići tražena UART komunikacija.
5. Pinovi PG0 do PG15 odabrani su za GPIO output pinove zadatka. Zbog selekcije tih pinova za GPIO onemogućen je rad periferije SPI6, a FMC ne sme da radi u isgto vreme sa konfliktnim GPIO output pinom. Ove dve periferije nisu potrebne u zadatku, tako da je sve u redu



1. Podešavanja UART parametara:

1. Proverena konfiguracija pinova I clock konfiguracija, ništa nije menjano.
2. Konfiguracija FreeRTOS-a zadržana default.
3. Projekat sačuvan pod imenom ElektronikaUART, za toolchain odabran EWARM, jer sam download-ovao IAR trial verziju. Nakon toga generisan kod.
4. Kod sadrži nekoliko foldera sa .c i .h fajlovima, gde se u jendom folderu nalaze fajlovi za FreeRTOS, u jednom je .out fajl za spuštanja na čip, postoji folder sa HAL drajverima I folder application gde se nalazi main u kome ću ja iskucati program, koristeći funkcije HAL drajvera.
5. U stm32f4xx\_hal\_uart.c postoji definicija funkcije

HAL\_StatusTypeDef HAL\_UART\_Receive(UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout) sa sledećim komentarom:

/\*\*

\* @brief Receives an amount of data in blocking mode.

\* @param huart: pointer to a UART\_HandleTypeDef structure that contains

\* the configuration information for the specified UART module.

\* @param pData: Pointer to data buffer

\* @param Size: Amount of data to be received

\* @param Timeout: Timeout duration

\* @retval HAL status

\*/

Planiram da ovakva funkcija stalno očitava podatke sa Rx pina i ukoliko naiđe na prvi preamble bajt počeće da parsira poruku. Ukoliko poruka ne bude ispravna nastaviće se čitanje narednog bajta dok se ne naiđe na ispravnu poruku nakon čega se pristupa setovanju vrednosti GPIO traženog GPIO pina.

U kodu je automatski izgenerisan UART\_HandleTypeDef promenljiva I nazvana huart1.

U kodu su automatski odrađene inicijalizacije. U sledećoj tački biće šta je ručno ddoato.

Svi pinovi G porta koji su odabrani za GPIO na po;etku su automatski postavljeni na nulu.

U kodu je automatski kreiran i pokrenut task:

osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128);

defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

U definiciju funkcije StartDefaultTask pri dnu main-a fajla biće iskucana funkcija čitanja sa UART periferije.

Deklarisana promenljiva uint8\_t RxBuffer[1];

Deklarisani su I preamble1, preamble2 I trailer istog tipa kao RxBuffer, sa tra\enim vrednostima iz yadatka.Deklarisana je promenljiva redniBrojGPIO I stanjeLSB;

Iskucana je sledeća funkcija (Task) koja izvršava traženo iz zadatka:

void StartDefaultTask(void const \* argument)

{

/\* USER CODE BEGIN 5 \*/

preamble1=0xAA;

preamble2=0x55;

trailer=0xEE;

/\* Infinite loop \*/

for(;;)

{

HAL\_UART\_Receive(&huart, (uint8\_t\*)RxBuffer, 1, 50);//cita se samo jedan bajt

if(preamble1 ^ RxBuffer[0])//ukoliko se razlikuju vrace se na pocetak petlje

continue;

else{ //ukoliko su jednaki cita se naredni bajt i ispituje sa preamble2

HAL\_UART\_Receive(&huart1, (uint8\_t\*)RxBuffer, 1, 50);

if(preamble2 ^ RxBuffer[0])//ukoliko se razlikuju vrace se na pocetak petlje

continue;

else{ //ukoliko su jednaki citaju se naredna dva bajta i bajt po bajt i ;uvaju se informacije

HAL\_UART\_Receive(&huart1, (uint8\_t\*)RxBuffer, 1, 50);

redniBrojGPIO=RxBuffer[0];

HAL\_UART\_Receive(&huart1, (uint8\_t\*)RxBuffer, 1, 50);

stanjeLSB=0x01&RxBuffer[0];

HAL\_UART\_Receive(&huart1, (uint8\_t\*)RxBuffer, 1, 50);

if(trailer ^ RxBuffer[0])//ukoliko se razlikuju vrace se na pocetak petlje i sve pada u vodu, poruka je ipak lažna

continue;

else { //ukoliko su jednaki poruka je ta;na i pristupa se dodeljivanju vrednosti pina

if(stanjeLSB){

switch(redniBrojGPIO){

case 0: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 1: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 2: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 3: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 4: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 5: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 6: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 7: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 8: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 9: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 10: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 11: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 12: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 13: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 14: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

case 15: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_SET); break;

default: break;//ukoliko nije nijedan od postojecih GPIO pinova poruka se ignorise

}

}

else{

switch(redniBrojGPIO){

case 0: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 1: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 2: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 3: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 4: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 5: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 6: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 7: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 8: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 9: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 10: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 11: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 12: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 13: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 14: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

case 15: HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_0, GPIO\_PIN\_RESET); break;

default: break;//ukoliko nije nijedan od postojecih GPIO pinova poruka se ignorise

}

}

}

}

}

osDelay(1);

}

/\* USER CODE END 5 \*/

}

Zbog prvog susretanja sa CubeMX softverskim paketom pri izradi zadatka korišćen primer inicijalizovanja ETH periferije sa adrese:

http://www.freertos.org/FreeRTOS-Plus/BSP\_Solutions/st/creating\_an\_STM32CubeMX\_project.html