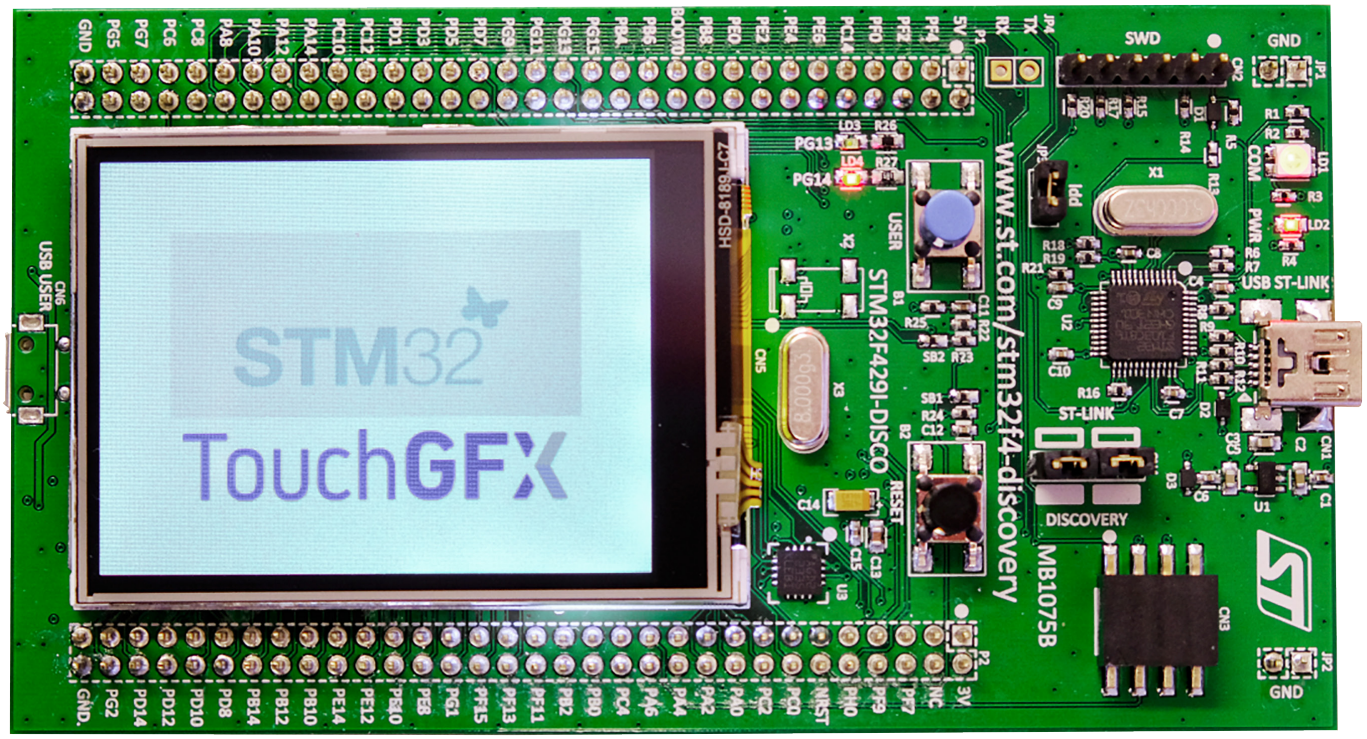


Projektowanie interfejsów graficznych z użyciem TouchGFX (1)

Standardy komunikacji pomiędzy urządzeniem a operatorem są wyznaczane przez producentów smartfonów. Wielu użytkowników całkiem słusznie postrzega stosowane tam rozwiązania za atrakcyjne i wygodne. TouchGFX oferuje możliwość tworzenia podobnych rozwiązań: przeciągania, przewijania, wstawiania efektów 3D, przezroczystości, mieszania czcionek itd., a dodatkowo położono nacisk na bardzo szybką reakcję aplikacji na dotyk ekranu wyświetlacza.

Problem prawidłowej interakcji operator–maszyna zaprzątał głowy projektantów od zarania ery przemysłowej. Początkowo stosowane ręczne dźwignie sterujące i mechaniczne wskaźniki położenia były stopniowo zastępowane elektrycznymi układami kontrolno-pomiarowymi. Nadzór nad procesami przemysłowymi zapewniały tablice kontrolne z wieloma przełącznikami, lampkami i miernikami wskazówkowymi nazywanymi potocznie zegarami. Były one postrzegane przez współczesnych jako przejaw postępu technicznego i nowoczesnej techniki. Pierwsze generacje komercyjnych komputerów również miały panele kontrolne z lampkami sygnalizacyjnymi i przyciskami sterującymi. Takie rozwiązania nikogo nie dziwiły nawet w latach 80. XX wieku.

Obsługa urządzeń bardziej lub mniej skompilowanych ewoluowała razem z rozwojem techniki i mocno przyspieszyła wraz z rozpowszechnianiem się mikroelektroniki. Komputery wyposażano w monitory ekranowe, najpierw tylko znakowe, potem również graficzne, klawiatury, później myszki, czy zapomniane już trackballe. W mikroprocesorowych układach wbudowanych królowały proste klawiatury numeryczne, przyciski i 7-segmentowe wyświetlacze LED. Dużym skokiem jakościowym było pojawienie się alfanumerycznego wyświetlacza LCD wyposażonego w sterownik HD44780. To do dzisiaj chętnie stosowany element w budżetowych rozwiązaniach.

Kolejny skok w rozwoju interfejsów użytkownika w układach sterowników mikroprocesorowych umożliwił następny element: wyświetlacz graficzny. Początkowo był to bardzo drogi element, najczęściej monochromatyczny, z niewielką rozdzielczością i przekątną ekranu. Obecnie takie komponenty są stosunkowo tanie i dostępne z kolorowymi matrycami o szerokim zakresie rozdzielczości, najczęściej wyposażane w ekran dotykowy rezystancyjny lub w najnowszych rozwiązaniach – pojemnościowy.

Równolegle z możliwościami sprzętowymi ewaluowała idea interfejsu użytkownika. Kiedy wydawało się, że maksimum możliwości daje konfiguracja z kolorowym monitorem, klawiaturą i myszką, pojawił się interfejs dotykowy. Idea sterowania urządzeniem za pomocą przyciskania ikon na ekranie wydawała się od początku uszyta na miarę potrzeb systemów wbudowanych. Wielka elastyczność interfejsów zbudowanych z ekranów z dedykowanymi elementami sterującymi rekompensowała niezbyt komfortowe działanie pierwszych rezystancyjnych paneli dotykowych. Współczesne panele pojemnościowe pracują nieporównywalnie bardziej precyzyjnie, ale rewolucja w branży smartfonów spowodowała, że znacznie wzrosły oczekiwania

użytkowników w zakresie estetyki, szybkości działania i ergonomii interfejsów użytkownika w systemach wbudowanych. Żeby sprostać takim wymaganiom, projektanci oprogramowania muszą stosować narzędzia umożliwiające szybkie tworzenie graficznych interfejsów użytkownika. Jednym z nich są gotowe biblioteki z funkcjami graficznymi. Kiedyś potrafiły być bardzo drogie i stać na nie było tylko bardziej zamożne firmy. Dlatego, podobnie jak z podstawowymi narzędziami: środowiskami IDE i kompilatorami C/C++, producenci mikrokontrolerów zaczęli dostarczać nieodpłatnie biblioteki wspierane konfiguratorami grafiki. Jednym z takich rozwiązań jest STM32 TouchGFX przeznaczony do projektowania interfejsów graficznych dla systemów wyposażonych w mikrokontrolery rodziny STM32.

TouchGFX jest dystrybuowanym bezpłatnie zaawansowanym programem przeznaczonym do projektowania interfejsów graficznych zoptymalizowanym dla mikrokontrolerów STM32. Jego podstawowym zadaniem jest wydajne wspieranie projektowania interfejsów użytkownika w urządzeniach wbudowanych opartych na mikrokontrolerach STM32. Zależnie od wymagań interfejsy te mogą wykorzystywać zarówno wyświetlacze o małej rozdzielczości i małej głębi koloru, jak i zaawansowane technicznie wyświetlacze z dużą głębią koloru, dużą rozdzielczością i wspomaganiem (akceleracją) sprzętową. TouchGFX jest zintegrowany z całym ekosystemem STM32 i zapewnia użytkownikom łatwy i szybki proces rozwoju aplikacji z użyciem znanych narzędzi projektowych.

Zaczynamy

Każdy, kto próbował samodzielnie programować graficzne interfejsy wie, że jest to żmudna i trochę niewdzięczna praca. Jeżeli do uzyskania nawet najprostszego efektu trzeba napisać sporo kodu i często dodatkowo przekopać się przez sporą dokumentację wyświetlacza i układów peryferyjnych mikrokontrolera, to jest to najlepszy sposób do zniechęcenia i w konsekwencji szukania innych rozwiązań. Z drugiej strony, kiedy pierwsze efekty przychodzą prawie natychmiast, to nabieramy ochoty na dalsze eksperymentowanie i eksplorowanie coraz bardziej zaawansowanych możliwości.

W przypadku programu TouchGFX przyjęto zasadę, że pierwsze działające aplikacje można sobie zaprojektować i wykonać szybko i przy początkowej nikłej wiedzy na temat całego środowiska projektowego. Jest jeden warunek – jeżeli program ma działać na sprzęcie, musimy dysponować jednym z wielu firmowych modułów ewaluacyjnych z mikrokontrolerami STM32, wyposażonych w kolorowe wyświetlacze LCD. Zwalnia to nas na początku z konieczności zaprojektowania, sprawdzenia i uruchomiania swojej platformy sprzętowej. Wydaje się, że jest to dość niska cena, którą przychodzi nam zapłacić za możliwość szybkiego opanowania podstaw projektowania własnych interfejsów i ocenienia, czy takie rozwiązanie jest dla nas przydatne.

Dla niecierpliwych przewidziano symulacje działania projektowanego interfejsu na ekranie komputera, ale jest to ograniczone tylko do warstwy wizualnej i z oczywistych względów nie obejmuje symulowania rozwiązań sprzętowych na przykład obsługi interfejsów komunikacyjnych czy szeroko rozumianej interakcji z pozostałymi elementami systemu wbudowanego.

Od wersji 4.13.0 TouhGFX, jest dystrybuowany jako pakiet X-CUBE-TOUCHGFX zawierający 2 elementy:

1. Genrator TouchGFX przeznaczony do tworzenia niestandardowych bibliotek HAL TouchGFX za pomocą CubeMX.
2. TouchGFX Designer przeznaczony do budowania projektu użytkowania za pomocą programu Windows GUI Builder.

Oprogramowanie można pobierać poprzez stronę st.com lub wykorzystując do tego celu CubeMX. Ponieważ my będziemy testować działanie TouchGFX Designer na jednej z płyt ewaluacyjnych ST, wykonamy instalację przez pakiet CubeMX.

Po otwarciu CubeMX trzeba wejść w zakładkę *Help → Manage Embedded Software Package → STMicroelectronics*, wybrać *X-Cube-TouchGFX* i kliknąć na *Install*. CubeMX zainstaluje TouchGFX Generator oraz pobierze wersję instalacyjną Touch GFX Designer i umieści ją w katalogu: *C:\Użytkownik\STM32Cube\repository\packs\STMicroelectronics\X-Cube-TouchGFX\3.13.0\Utilites\PC_software\TouchGFXDesigner*.

Instalowanie TouchGFX Designer przebiega standardowo i nie wymaga komentarza.

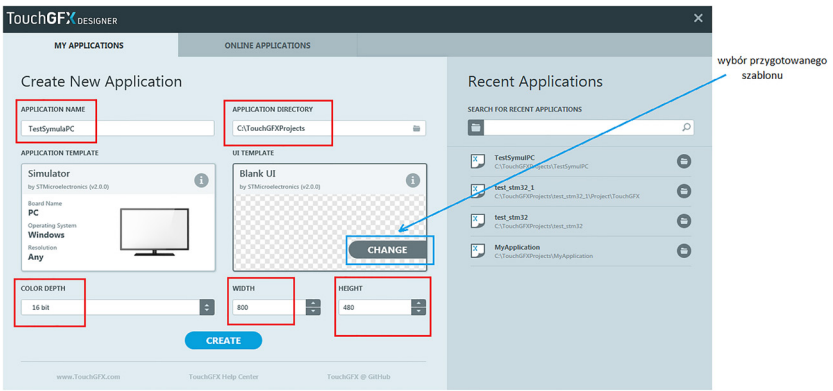
W tym momencie nie będziemy się zajmować generatorem TouchGFX przeznaczonym do generowania bibliotek warstwy HAL łączących aplikacje Touch GFX ze sprzętem. Dla modułów ewaluacyjnych STM32 takie biblioteki są przygotowane. Możemy ten etap pominąć i skupić się na projektowaniu interfejsu graficznego.

TouchGFX Designer

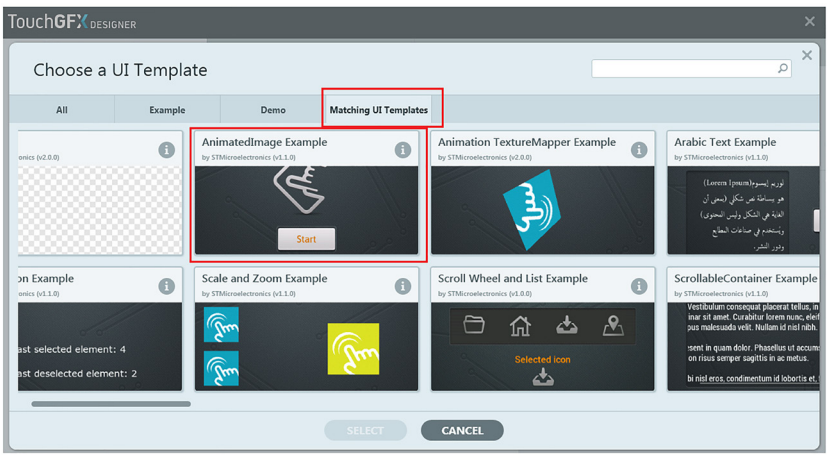
Przykładowe aplikacje interfejsu użytkownika można uruchamiać i testować na dwa sposoby. Pierwszy z nich to symulowanie projektowanych ekranów w symulatorze uruchamianym na komputerze pracującym w systemie Windows. Nie jest wymagany żaden dodatkowy sprzęt, ale jest to tylko symulacja. Drugi sposób polega na wygenerowaniu kodu na podstawie projektu interfejsu. Potem ten kod jest kompilowany i przesyłany do pamięci mikrokontrolera modułu ewaluacyjnego. W tym przypadku interfejs użytkownika jest uruchamiany i pracuje na docelowym sprzęcie.

Przykładowa aplikacja w symulatorze

Po uruchomieniu TouchGFX Designer rozpoczynamy nowy projekt. W oknie *Application Name* nadajemy mu nazwę na przykład



Rysunek 1. Okno kreatora nowego projektu



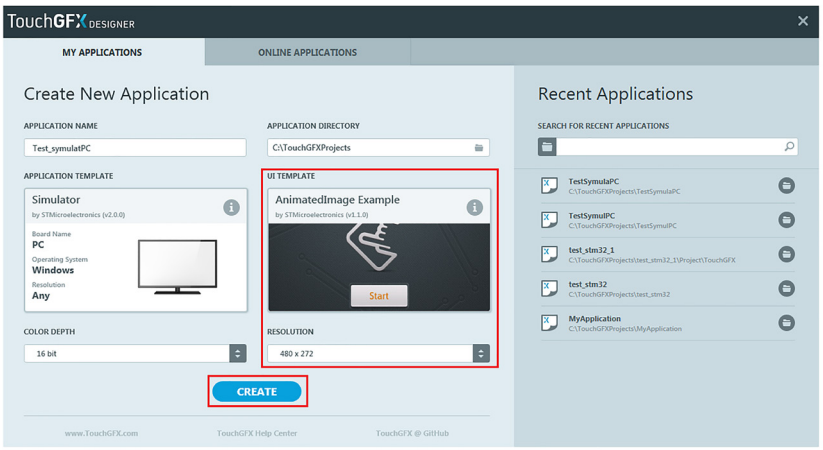
Rysunek 2. Okno wyboru przykładowych interfejsów

TestSymulatPC. Ścieżka dostępu do folderów z plikami projektów jest określana w oknie *Application Directory*. Foldery mają taką samą nazwę jak projekty. Kreator nowej aplikacji umożliwia na tym etapie określenie bitowej głębi kolorów i rozmiaru wyświetlacza (**rysunek 1**).

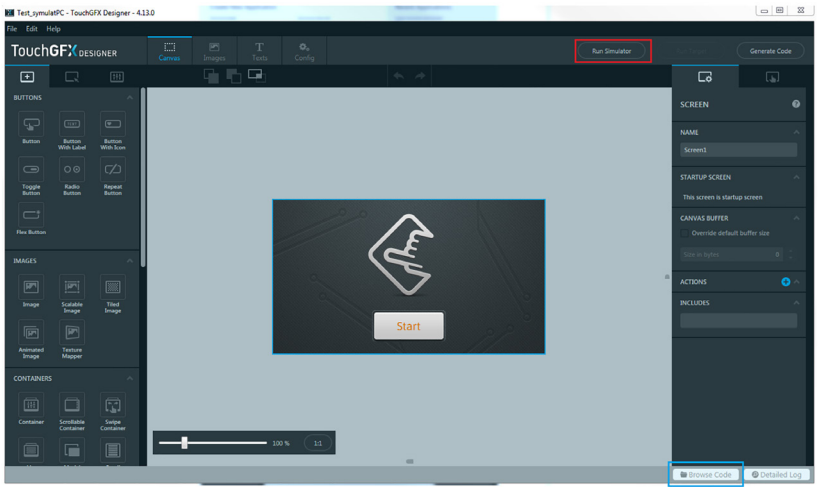
Po kliknięciu na okno szablonu *UI Template* pojawia się przycisk *CHANGE* pozwalający wybrać przygotowane przykładowe szablony interfejsy użytkownika (**rysunek 2)**. Z listy u góry wybieramy opcję *MatchnigUITemplate*, po , by na liście były tylko szablony pasujące do ustawień projektu (rozdzielczości wyświetlacza). Wybieramy pierwszy z listy *AnimatedImage Example* i klikamy na *Select*. Teraz wybrany szablon pojawi się w okienku *Template* okna *Create New Application* (**rysunek 3**).

Po kliknięciu na przycisk *CREATE* TouchGFX komunikuje się z serwerami ST, łączy wybrany szablon interfejsu z domyślnym szablonem aplikacji i tworzy przykładowy gotowy projekt. Okno projektu zostało pokazane na **rysunku 4**. U góry okna z prawej strony jest umieszczony przycisk *Run Simulator*. Po jego przyciśnięciu TouchGFX kompiluje nasz projekt i uruchamia symulację w oknie systemu Windows (**rysunku 5**).

Po kliknięciu na przycisk *Start* jest wyświetlana animacja ruchu ikony umieszczonej nad przyciskiem *Start*. Po kliknięciu na przycisk *Browse Code* otwiera się katalog z plikami projektu wygenerowanymi przez TouchGFX.



Rysunek 3. Okno kreatora projektu z wybranym przykładowym szablonem



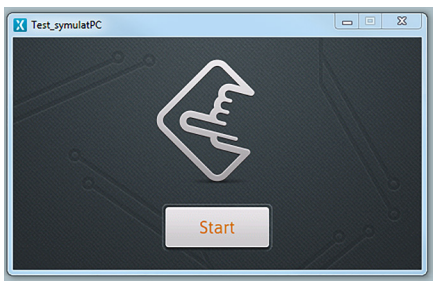
Rysunek 4. Okno przykładowego projektu

Przykładowa aplikacja działająca na module ewaluacyjnym

Symulowanie działania przykładowych interfejsów w okienku Windows pozwala na szybkie poznanie możliwości przygotowanych przykładów. Funkcja symulatora może też pomagać przy projektowaniu grafiki na własne potrzeby bez powiązania z konkretnym sprzętem. Jednak dla konstruktora ważniejsze jest uruchomienie aplikacji na konkretnej platformie sprzętowej. Jak już wspomniałem, TouchGFX umożliwia tworzenie interfejsów użytkownika na modułach ewaluacyjnych oferowanych przez ST.

Kreator nowego projektu automatycznie ustawia domyślnie w oknie *APPLICATION TEMPLATE* opcję symulatora PC, a w oknie *UI TEMPLATE BLANK UI*. W poprzednim przykładzie zmieniliśmy *UI TEMPLATE* na *AnimatedImage Example* tak jak to zostało pokazane na rysunku 2. Żeby możliwe było wygenerowanie szablonu projektu współpracującego z jednym z modułów ewaluacyjnych, trzeba go wybrać w oknie *APLICATION TEMPLATE* (**rysunek 6**). Klikamy na przycisk *CHANGE* w oknie *APPLICATION TEMPLATE* i z listy *Choose an Application Template* wybieramy jeden z modułów.

Niestety w trakcie pisania tego artykułu była dostępna wersja TouchGFX 4.13, w której generowanie szablonów nie działało dla wszystkich modułów z listy. Na stronie http://bit.ly/3rGaeXr jest umieszczona informacja, że poprawnie działają szablony w wersji V3.0 i wyższej. Program jest ciągle rozwijany i kolejne moduły są prawidłowo wspierane. Dostępne wersje szablonu można wybrać, klikając na ikonkę *i* wybieranego modułu (**rysunek 7**).



Rysunek 5. Okno symulacji ekranu interfejsu użytkownika

W trakcie prób dysponowałem modułem STM32F7508-DK. Początkowo można go było wybrać, program wygenerował szablon, ale projektu nie dało się skompilować, bo były błędy w poleceniach kompilacji oraz w samym wygenerowanym kodzie. Nie można było użyć STM32F7508-DK i TouchGFX Designera ani do uruchomienia gotowych przykładów, ani do tworzenia własnej aplikacji z użyciem *UI TEMPLATE BLANK UI*. W kolejnej wersji TouchGFX szablon dla STM32F7508-DK otrzymał wersję V3.0.0 i wszystko zaczęło działać.

Po tych niepowodzeniach postanowiłem początkowo spróbować z dość starym modułem STM32F429I Discovery również umieszczonym na liście modułów, ale w starej wersji V1.1.0. I tu się okazało, że dla tego modułu TouchGFX Designer generuje działające projekty zarówno przykładowe, jak i własne, wykonane na podstawie pustego szablonu *BLANK UI*. Na początek postanowiłem uruchomić program demonstracyjny przygotowany przez twórców TouchGFX. Ponieważ wyświetlacz zamontowany w STM32F429I Discovery ma relatywnie małą rozdzielczość, okazało się, że dla niego można wybrać tylko szablon *BLANK UI* i program demonstracyjny TouchGFX Demo3 w wersji V1.1.0 (**rysunek 8**).

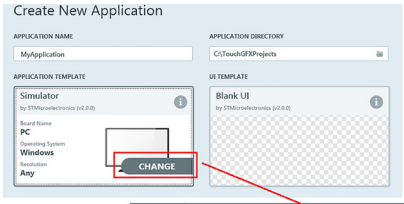
Po kliknięciu na przycisk *CREATE* TouchGFX komunikuje się z serwerami ST i generuje projekt widoczny na ekranie Designera. W tym momencie powinniśmy móc skompilować wygenerowany kod i po kompilacji przesłać go do pamięci mikrokontrolera. Projekt jest kompilowany po kliknięciu na przycisk *Generate Code*. Przebieg kompilacji możemy obserwować w oknie otwieranym po kliknięciu na *Detailed Log* (**rysunek 9**).

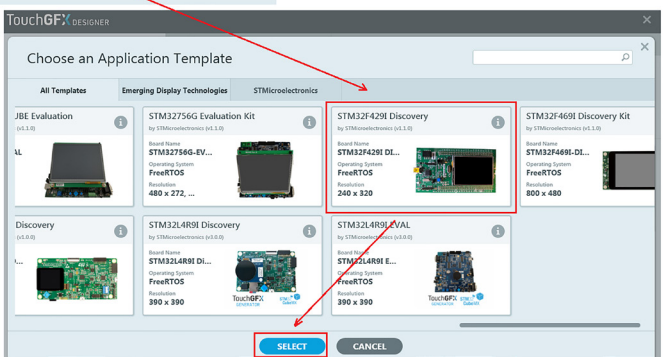
Jeżeli kod zostanie prawidłowo skompilowany, a w przypadku tego programu demonstracyjnego tak się dzieje, to można go przesłać do pamięci modułu, klikając na przycisk *Run Target*. Kod jest ponownie kompilowany, a wynik jest przesyłany do pamięci modułu. Do programowania pamięci TouchGFX Designer uruchamia program STM32 ST_LINK Utility. Program ten musi być wcześniej zainstalowany w domyślnej lokalizacji. W przeciwnym przypadku programowanie pamięci nie może się

wykonać i zgłaszany jest błąd. Ten sposób programowania pamięci Flash nie sprawdził się w dalszych testach modułu, który oprócz pamięci Flash mikrokontrolera wymagał zaprogramowania wbudowanej zewnętrznej pamięci Flash używanej przez TouchGFX do przechowywania swoich danych, na przykład krojów czcionek, bitmap itp.

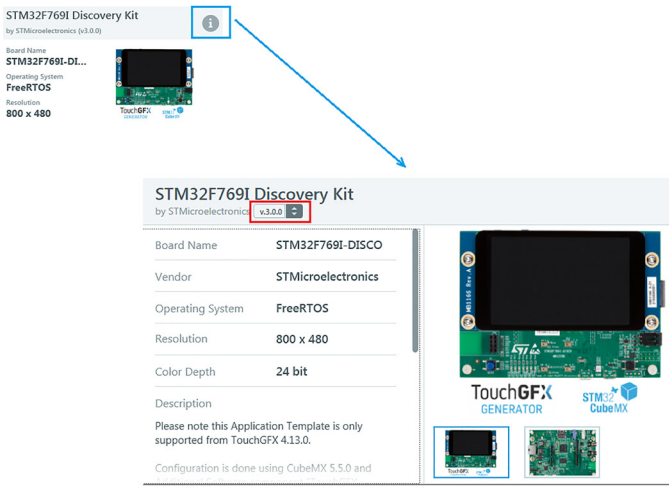
Po załadowaniu i uruchomieniu projektu można ustawiać ile „bąbelków" będzie wyświetlanych przez program testowy (**fotografia 1**). Uruchomienie aplikacji testowej pozwoliło nam na sprawdzenie poprawności wyboru obsługiwanego modułu testowego, co jak się okazało, nie jest takie oczywiste. Poza tym sprawdziliśmy kompilowanie projektu i ładowanie pliku wynikowego do pamięci. Przyczyna początkowych niepowodzeń wynikająca z użycia modułu, dla którego nie przygotowano jeszcze gotowego szablonu, została usunięta w kolejnych wersjach programu TouchGFX Designer.

Kolejnym krokiem będzie zaprojektowanie i wykonanie własnego prostego interfejsu w programie TouchGFX Designer. W oknie *Application Template* wybieramy testowy moduł STM32F429I Discovery, a w oknie *UI TEMPALTE* czysty szablon *Blank UI* i klikamy na przycisk *CREATE* (**rysunek 10**). W głównym oknie edytora wyświetlany jest pierwszy ekran interfejsu (można potem dodawać następne) o rozdzielczości równej rozdzielczości wyświetlacza. Umieszczony w lewym dolnym rogu pasek narzędzia powiększania pozwala powiększyć wyświetlany ekran. Powiększenie pomaga przy projektowaniu, na przykład przy precyzyjnym umieszczaniu widżetów na ekranie. Po kliknięciu w pasku narzędzi przycisku *Config* można wybrać opcję *Display* (**rysunek 11**). Jest tu wyświetlana rozdzielczość ekranu (nie można jej zmienić), orientacja, do wyboru pozioma lub pionowa, i bitowa głębia koloru. To dobre miejsce wyboru orientacji wyświetlacza, bo obie opcje pionowa i pozioma mogą być stosowane zależnie od potrzeb. My zmienimy orientację na poziomą.





Rysunek 6. Wybór modułu ewaluacyjnego

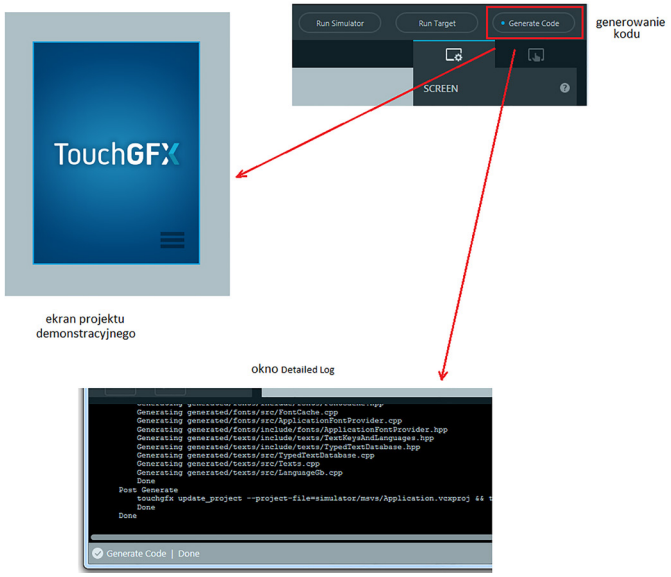


Rysunek 7. Sprawdzenie i wybór wersji szablonu

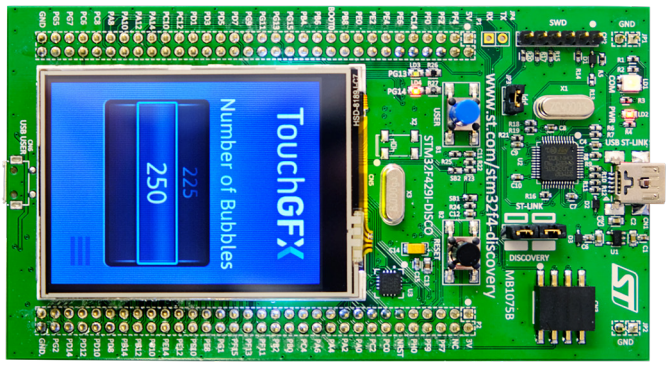
Rysunek 8. Wybór aplikacji demonstracyjnej dla modułu STM32F429I

Każdy ekran jest identyfikowany przez swoją nazwę. Kreator pustego szablonu generuje jeden ekran startowy o domyślnej nazwie *Screen1*. Ponieważ jest to ekran, który będzie wyświetlany po uruchomieniu aplikacji (startup screen) zmienimy jego nazwę na *main*. Na **rysunku 12** zostało pokazane zarządzanie ekranami. Okno *Screens* z lewej strony ekranu głównego wyświetla wszystkie ekrany projektu (w tym momencie tylko jeden główny *main*). Ikona + *(plus)* umożliwia dodawanie do listy kolejnych ekranów według potrzeb. Okno *SCREEN* z prawej strony ekranu głównego TouchGFX Designer umożliwia zmianę nazwy okna (pole *NAME*). Na dole okna *SCREEN* umieszczono pole *ACTIONS*, w którym dodaje się akcje powiązane z ekranem.

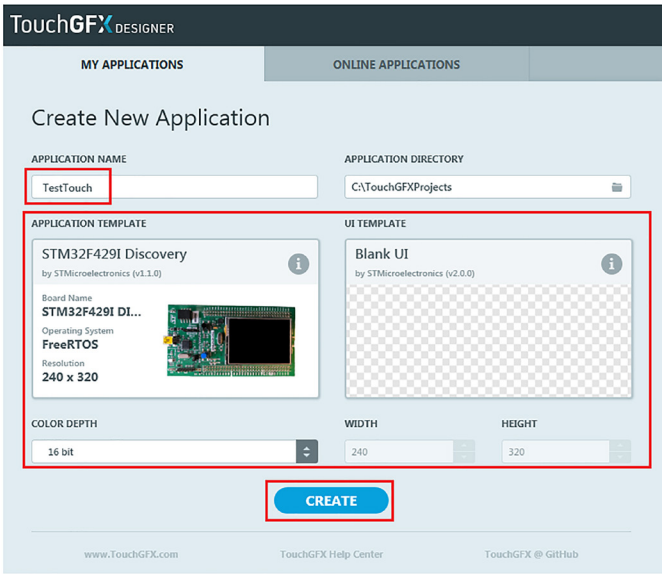
Kolejnym krokiem będzie dodanie tła ekranu. Możemy użyć do tego celu obrazka o rozmiarze w pikselach nie większym niż rozdzielczość ekranu LCD, który jest zapisany w formacie *.png* (tylko taki format jest akceptowany przez TouchGFX Designer). Istnieje też możliwość wyboru jednolitego tła (stylu), czyli przygotowanego pliku *.dng* jednolicie białego. Obrazek o żądanej rozdzielczości przygotowujemy sobie wcześniej i zapisujemy w folderze na dysku. Z otwartego okna *Sreens* przechodzimy do okna dodawania widżetów z lewej strony ekranu głównego i z listy *Images* wybieramy *Image*. W oknie ekranu pojawi się ikona, którą możemy przesuwać po ekranie (**rysunek 13**). Dodaliśmy widżet obrazka i w kolejnym kroku trzeba określić, co to za obrazek, czyli wybrać jego lokalizację



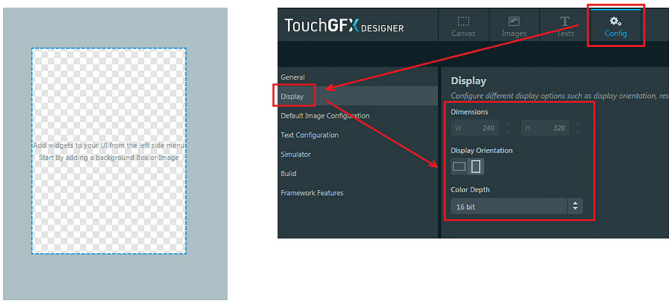
Rysunek 9. Kompilowanie kodu wygenerowanego przez TouchGFX Designer



Fotografia 1. Program testowy uruchomiony na module STM32F429I Discovery



Rysunek 10. Tworzenie własnego pustego projektu

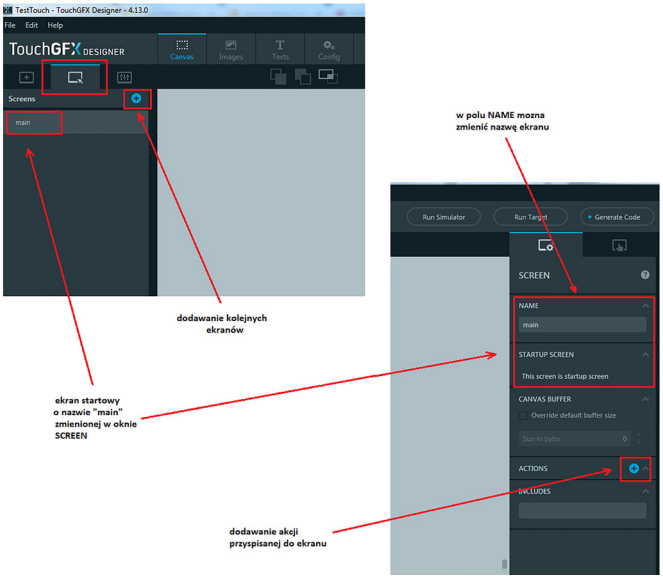


Rysunek 11. Konfiguracja orientacji wyświetlania i bitowej głębi kolorów TouchGFX Designer

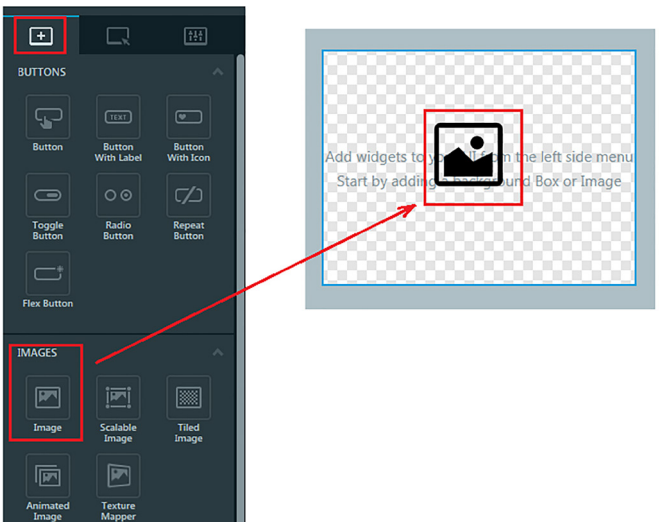
na dysku. Klikamy na ikonę obrazka i z prawej strony okna Designera pojawia się okno właściwości *IMAGE*.

W polu *Name* można zmienić nazwę widżetu. Pole *Style* służy do wybrania przygotowanego białego tła (alternatywnie dla własnego obrazka). W polu *Image* wybieramy własny obrazek (bitmapę) tła o rozdzielczości równej rozdzielczości ekranu LCD, wcześniej zapisany na dysku. Można tu dodawać wiele bitmap, nie tylko służących jako tło, ale na przykład wykorzystywanych do wyświetlania widżetów przycisków (button). Kolejne obrazki dodajemy, klikając na ikonę + (plus) pola *Image*, a potem wybieramy jeden z nich. Na **rysunku 14** zostało pokazane okno właściwości widżetu *Image* z dodawaniem własnego obrazka tła. Efekt dodania tła na ekranie został pokazany na **fotografii tytułowej artykułu.**

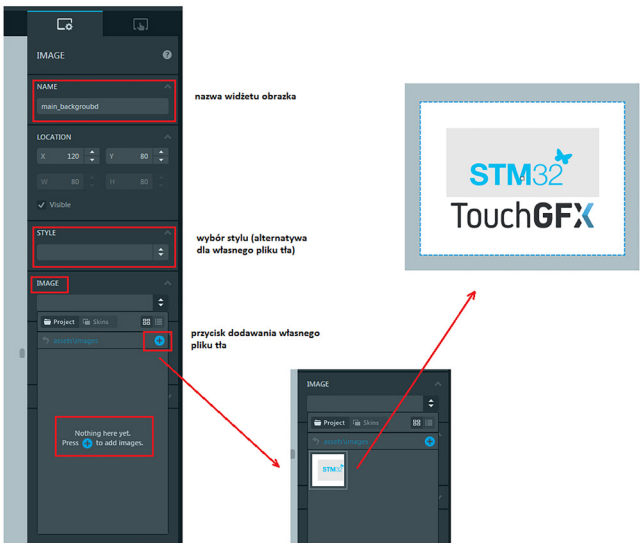
W ten sposób możemy ustawiać dowolne tło ekranu. Na potrzeby dalszej części przykładowej aplikacji zmieniłem tło na bardziej jednolite, żeby kolejne elementy dodawane do interfejsu były lepiej widoczne. Kolejną czynnością w budowaniu naszego prostego przykładowego interfejsu



Rysunek 12. Zarządzanie ekranami: zmiana nazwy i dodawanie kolejnych ekranów



Rysunek 13. Dodanie obrazka jako tła



Rysunek 14. Dodawanie obrazka – pliku .png

będzie dodanie dwóch widżetów przycisków (button), nadanie im odpowiedniego wyglądu i przypisanie wykonywanej akcji. Od tego zaczniemy kolejną część artykułu.

Tomasz Jabłoński, EP