Dokumentacja

Oprogramowanie maszyny sortującej kolorowe cukierki

Skład sekcji:
Tomasz Andrzejewski, Tomasz Budzyński, Adam Sankowski,
Nicola Fuchs, Remigiusz Oczko

Spis treści

1 Specyfikacja sprzętu

1.1 Minimalne wymagania sprzętowe

- Kamera - przechwytująca kolory z rozdzielczością 300x300 i framewrite 30fps
- Mikrokontroler - dowolny z rodziny STM32, posiadający timery oraz obsługujący komunikację USART
- Serwomechanizm - Dowolny, kompatybilny z STM32 oraz obsługujący zakres obrotu 0-180°
- Opcjonalnie zasilanie zewnętrzne dla serwomechanizmu

1.2 Wymagane oprogramowanie

- Python w wersji 3.4+, 32bit
- OpenCV w wersji 32bit
- LabView w wersji 32bit
- ST-Link

2 Przypadek użycia

2.1 Sprzęt wykorzystany do przeprowadzenia testów

- Kamera - Logitech HD Webcam C310
- Mikrokontroler - STM32 NUCLEO F303RE, 72MHz
- SERVO SERWO Tower Pro SG90

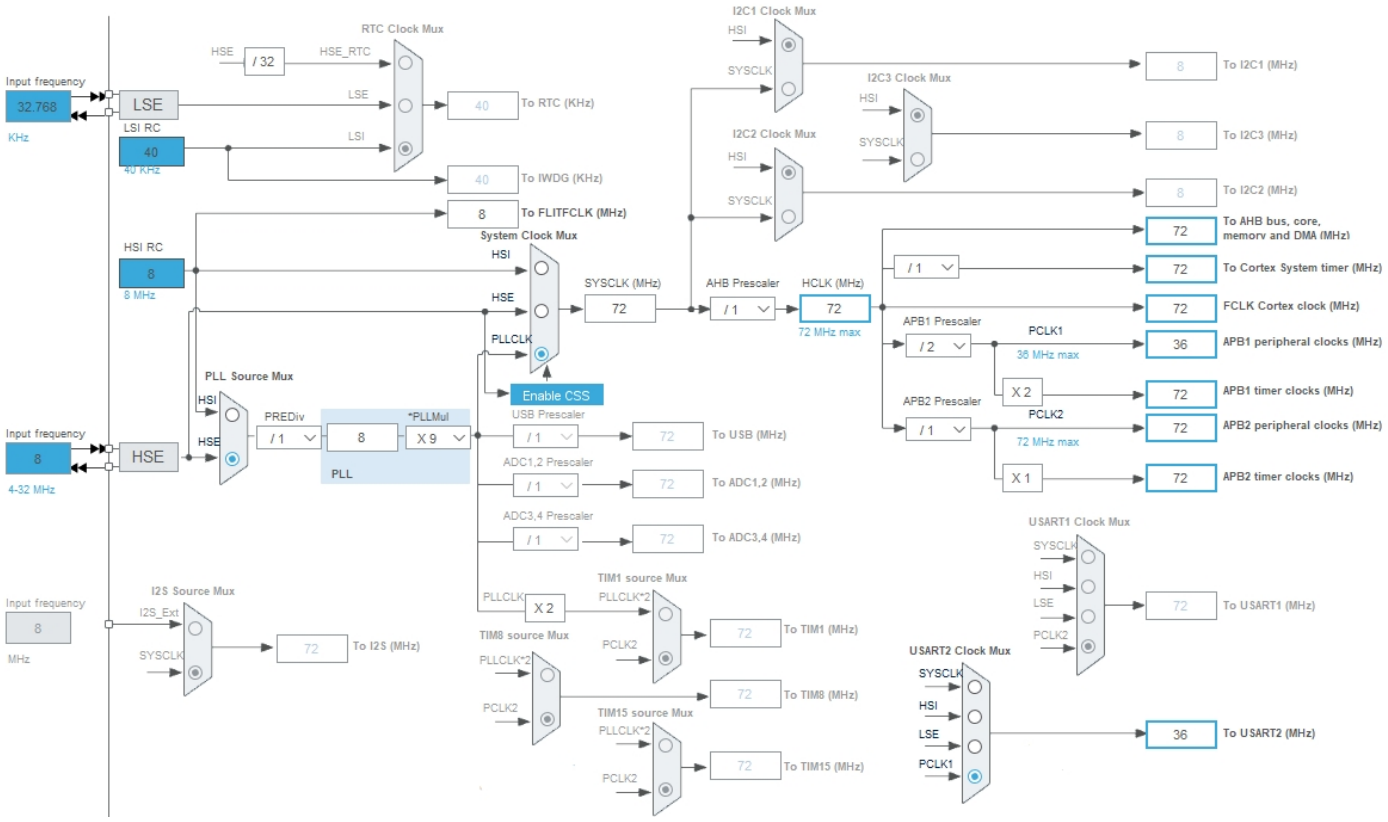
2.2 Oprogramowanie

- Python 3.8.8 32bit
- OpenCV 4.3 32bit
- STM32CubeIDE 1.5.0 32bit
- LabView 20.0 32bit

2.3 Konfiguracja STM32

2.3.1 Konfiguracja Timera

W celu zapewnienia większej stabilności sygnału taktującego powinniśmy wybrać zewnętrzne źródło taktowanie (HSE). Wówczas niezbędne jest wybranie opcji Crystal/Ceramic resonator. W kolejnym etapie konfiguracji musimy przejść do panelu Clock Configuration (Rysunek 1) i w tabeli Input Frequency ustawić wartość 8MHz, kolejnym istotnym krokiem jest wybranie w pętli PLL Source Mux wejścia HSE, natomiast w System Clock Mux wejścia PLLCLK. Następnie konfigurując dzielniki i mnożniki powinniśmy ustawić w oknie HCLK wartość 72MHz (w przypadku mikrokontrolera 84MHz wartość ta powinna wynosić 84MHz) oraz maksymalną wartość na magistrali odpowiedzialnej za obsługe timera generującego sygnał PWM (w naszym przypadku jest to magistrala APB1). Pozostale parametry należy pozostawić bez zmian.



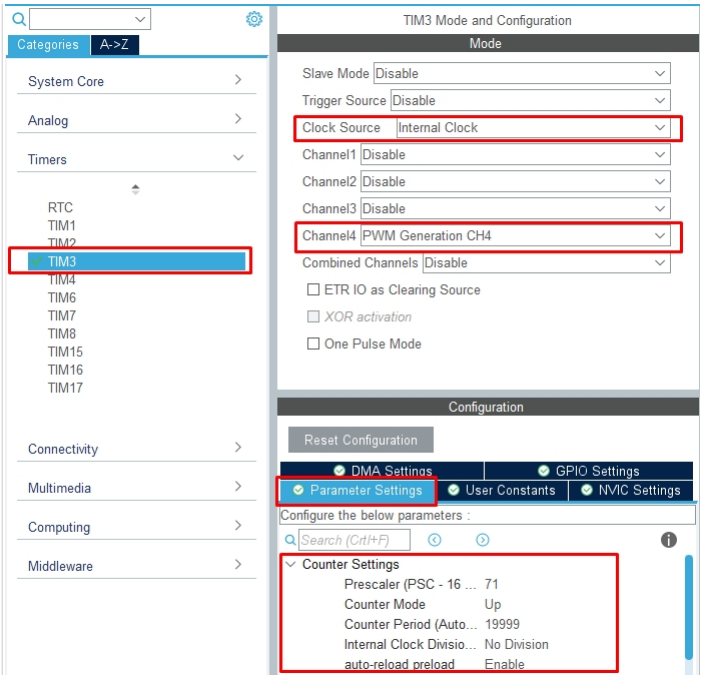
Rysunek 1: Clock Configuration

Następnie należy przystąpić do konfiguracji Timera. Po wybraniu odpowiedniej zakładki w kategorii Timers przechodzimy do okna Parameter Settings. Aby poprawnie ustawić następne zmienne musimy skorzystać z poniższego wzoru:

$$FREQ = \frac{TIM_{CLK}}{(ARR+1)(PSC+1)(CKD+1)}$$

gdzie:
FREQ - częstotliwość generowanego przerwania
TIM_{CLK} - częstotliwość taktowania magistrali, na której umieszczony jest timer
ARR - AutoReload Register
PSC - Prescaler
CKD - Internal Clock Division

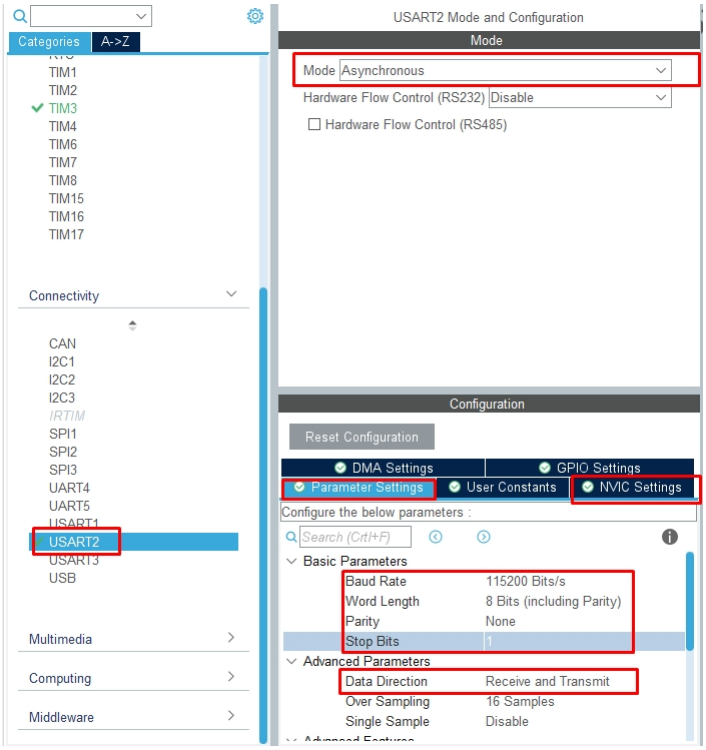
Można również skorzystać z przygotowanego przez nas pliku Excel, który oblicza nam wartość FREQ w sposób zautomatyzowany. Arkusz dostępny jest pod adresem: Plik excel. W celu zapewnienia poprawności działania układu powinniśmy wartość FREQ ustawić na 50Hz, PSC=19999, CKD=1, ARR=1. Pozostałe parametry pozostawiamy bez zmian.



Rysunek 2: Ustawienie kluczowych parametrów konfiguracji Timera

2.3.2 Konfiguracja USART

Aby prawidłowo przystąpić do transmisji danych, należy w właściwy sposób skonfigurować port USART na naszym mikrokontrolerze, w tym celu przechodzimy do zakładki Connectivity i wybieramy wybrane przez nas piny, które umożliwiają przesył danych. Jeśli nie wiemy, który wybrać należy skorzystać z oficjalnej dokumentacji dostępnej na stronie producenta lub w zakladce Connectivity wybrać dowolny wiersz o nazwie USART. Następnie w oknie Mode wybieramy tryb transmicji, zalecamy wybranie trybu Asynchronicznego. Kolejnym oknem które należy skonfigurować jest Parameter Settings, okno to pozwala nam wybrać prędkość transmisji (Baud Rate). Tą wartość ustawiamy na 115200Bits/s. Ważnymi parametrami jest również długość przesyłanej ramki (zalecamy ramkę o długości 8), występowanie bitu parzystości (zalecamy wybrać wartość None) oraz ilość bitów stopu - tutaj sugerujemy wybrać 1. Jednym z najważniejszych elementów konfiguracji, z punktu widzenia naszego zadania jest kierunek transmisji. W naszym przypadku KONIECZNE jest ustawienie transmisji dwukierunkowej (Receive and Transmit). Ostatnim z etapów konfiguracji jest ustawienie USART w trybie przerwaniowym, możemy to wykonać w zakładce NVIC Settings. Należy również pamiętać, że konfiguracja na tym poziomie, wpływa na kompatybilność z LabView i każde odstępstwo od podanych przez nas parametrów musi być również ustawione w konfiguracji subVI o nazwie "ToSTM".

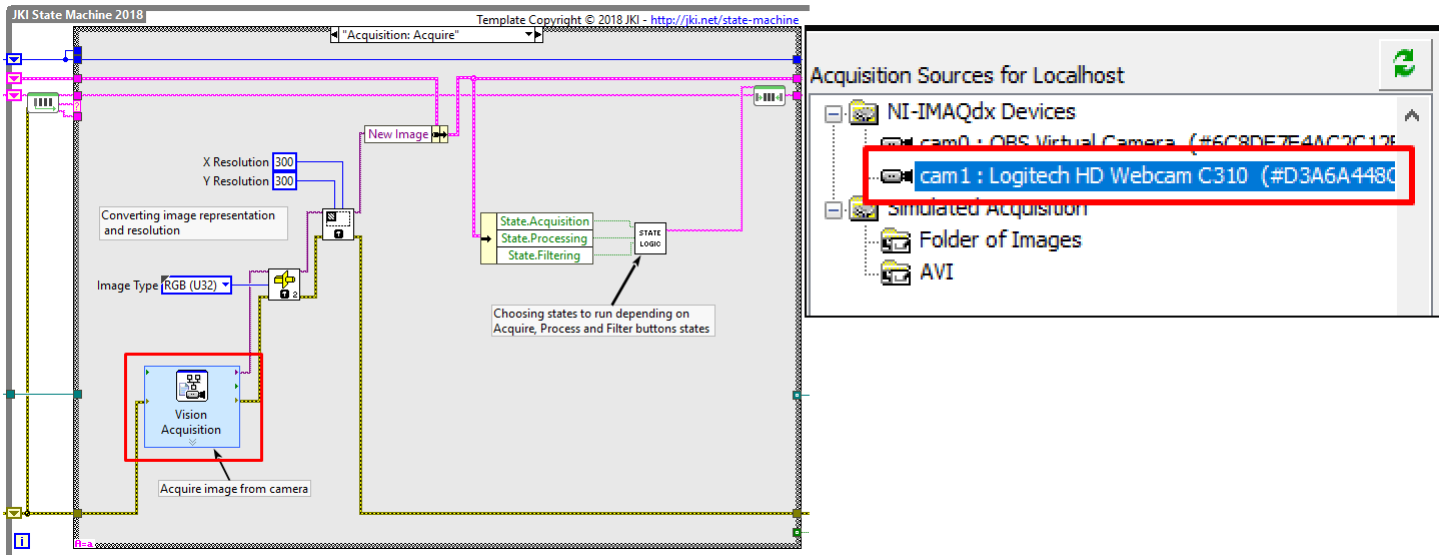


Rysunek 3: Ustawienie kluczowych parametrów konfiguracji USART

2.4 Konfiguracja LabView

2.4.1 Konfiguracja Vision Acquisition

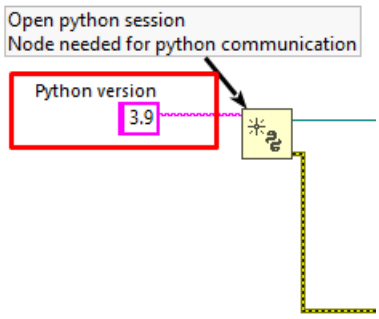
Aby nasz program działał poprawnie, musimy skonfigurować kamerę. W tym celu przechodzimy do okna Block Diagram a następnie w Event Structure wybieramy zakładkę Acquisition Acquire. Kolejnym krokiem jest wejście w blok Vision Acquision gdzie wybieramy dostępną kamerę. Pozostałe opcje pozostawiamy domyślnie.



Rysunek 4: Konfiguracja Vision Acquisition

2.4.2 Wybór odpowiedniej wersji Pythona

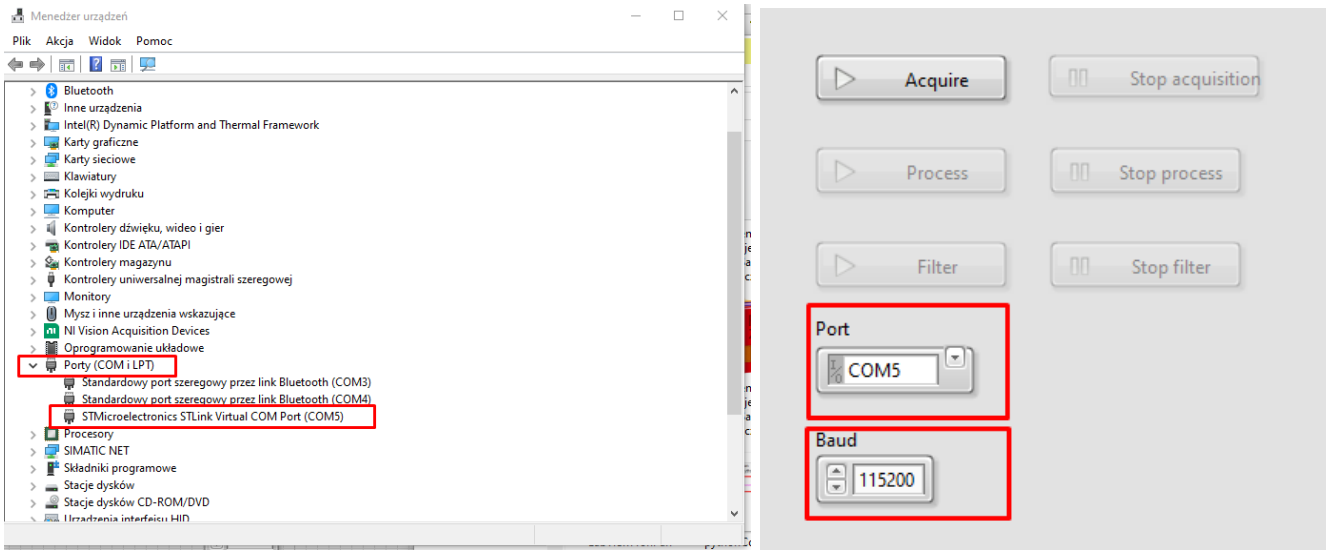
Aby wybrać posiadaną przez nas wersję Pythona, należy przejść do bloku Python version, a następnie wpisać odpowiednią wartość.



Rysunek 5: Konfiguracja wersji Pythona

2.4.3 Konfiguracja parametrów potrzebnych do nawiązania połączenia z STM32

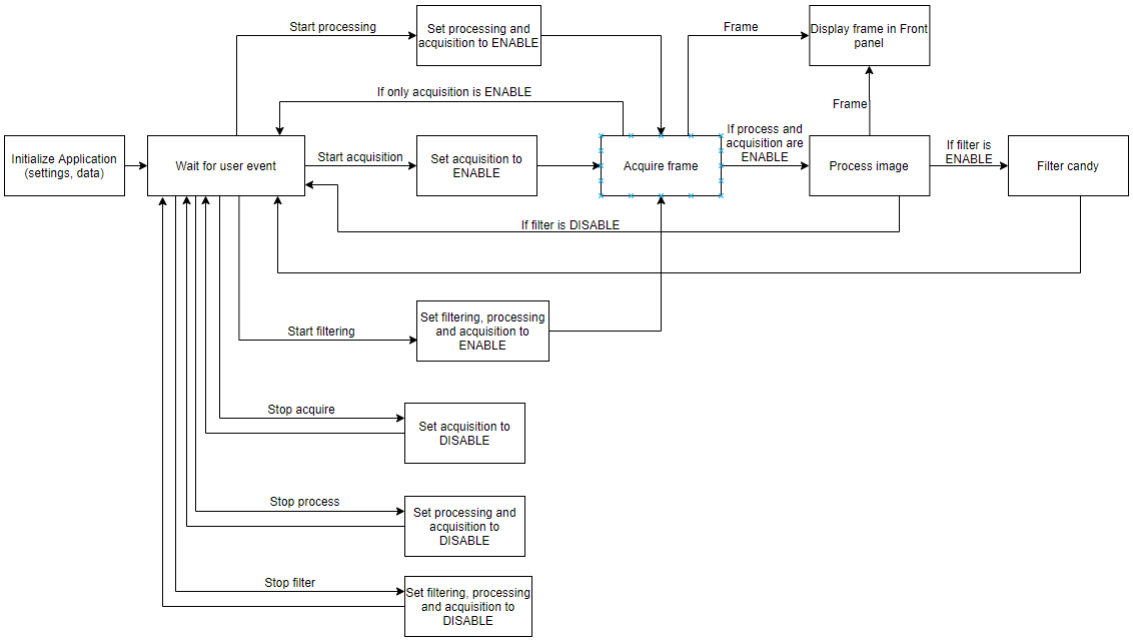
Aby nawiązać połączenie z mikrokontrolerem, za pomocą USART musimy skonfigurować wartości Baud Rate oraz Port. Konfiguracja tych zmiennych znajduje się na panelu użytkownika. Baud Rate wybieramy zgodnie z wartością zadeklarowaną na etapie konfiguracji STM32 (Rysunek3). Aby poprawnie skonfigurować Port musimy wejść do menadżera urządzeń i w zakładce Porty(COM I LPT) wybrać odpowiedni port wirtualny.



Rysunek 6: Konfiguracja Baud Rate oraz Portu COM

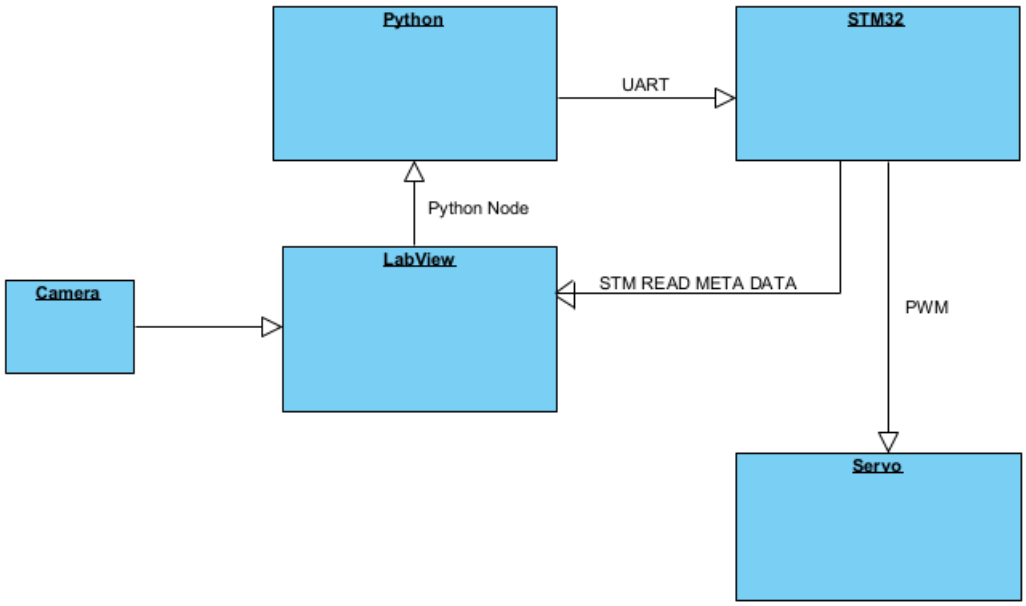
3 Schemat blokowy aplikacji, diagram przepływu danych

3.1 Schemat blokowy aplikacji



Rysunek 7: Schemat blokowy aplikacji

3.2 Diagram przepływu danych



Rysunek 8: Diagram przepływu danych

4 Opis poszczególnych stanów

4.1 Event Structure

Stan ten używany jest do reagowania na używanie przycisków, umieszczonych na front panelu oraz, do obsługi niektórych zdarzeń takich jak kliknięcie w przycisk X na front panelu w trakcie działania aplikacji.

4.2 Core

To grupa stanów, które zostały zbudowane przez firmę JKI i są stanami bazowymi dla frameworku JKI State Machine. Dzięki nim, developer nie musi zajmować się budowaniem niektórych stanów od zera, co pozwala skupić się na tworzeniu funkcjonalności i ewentualnie, w razie potrzeby, dostosowaniu tych bazowych stanów.

4.2.1 Default

Gdy zachodzi niechciana sytuacja przejścia do stanu, który nie jest zdefiniowany głównym Case Structure, stan Default zostaje wywołany. Pokazuje on developerowi (stan ten nie powinien być możliwy do wywołania przez końcowego użytkownika), że aplikacja chciała wywołać stan nieistniejący stan i że należy naprawić taki stan rzeczy.

4.2.2 Initialize Core Data

Jest to stan wywoływany przy uruchamianiu aplikacji. Tworzy on dwie zmienne referencyjne, które zostają przekazane do głównego klastra danych.

4.2.3 Error Handler

Jak nazwa wskazuje, stan ten odpowiada za obsługę błędów. Do niego trafiamy zawsze, gdy w dowolnym stanie ścieżka błędu przyjmie kod niezerowy. Zazwyczaj błędy są obsługiwane poprzez powiadomienie o wystąpieniu błędu czytelnym dla przeciętnego użytkownika komunikatem, zatrzymaniem działania pewnej funkcjonalności lub całej aplikacji oraz ewentualnie poprzez wybór, jak użytkownik chce obsłużyć dany błąd.

4.3 Data

Grupa stanów Data odpowiada za dane, które krążą pomiędzy stanami w trakcie działania aplikacji.

4.3.1 Initialize

Odpowiada on za inicjalizację danych w aplikacji. Dzięki temu każdy typ danych, używany w dalszej części aplikacji, może zostać w tym stanie utworzony, a następnie jedynie wyciągany z głównego klastra danych za pomocą funkcji BBN lub UBN. Konwencja używania frameworku JKI SM mówi, aby wszystkie dane, które są wielokrotnie potrzebne w różnych stanach, były tworzone właśnie w tym stanie.

4.3.2 Cleanup

Stan ten zamyka referencję do obecnego VI po wykonaniu akcji naciśnięcia przycisku X. Jest on potrzebny, ponieważ zamykanie referencji w LabView jest bardzo ważne i należy to robić, gdy tylko nie będziemy już więcej korzystać z nich.

4.4 UI

Stany w grupie UI odpowiadają za większość obsługi front panelu w aplikacji.

4.4.1 Initialize

Aby aplikacja przy każdym uruchomieniu wyglądała tak samo, należy zainicjalizować początkowy stan front panelu. Właśnie za to odpowiada stan Initialize. W obecnej wersji ustawia on początkową dostępność przycisków.

4.4.2 Update UI

W tym stanie, w zależności od tego, czy uruchomiono funkcjonalność odbierania obrazu z kamery, aktualizowany jest obraz klatka po klatce.

4.4.3 Cursor Set

Stan ten odpowiada za ustawienie wyglądu kursora w zależności od obecnego stanu aplikacji (busy, idle). W obecnej wersji aplikacji nie jest on wywoływany.

4.4.4 Front Panel State

Ten stan obsługuje różne stany front panelu, np.: minimalizację, zamknięcie.

4.5 Macro

Stany typu Macro są tworzone tylko dla czytelności kodu. Gdy zachodzi potrzeba wykonania kilku stanów jeden po drugim, można stworzyć stan typu Macro, którego zadaniem będzie wprowadzenie do kolejki odpowiedniej listy stanów. W ten sposób kod staje się bardziej czytelny.

4.5.1 Initialize

Stan ten wywoływany jest zawsze przy uruchamianiu aplikacji. Wywołuje on stany odpowiedzialne za inicjalizację aplikacji.

4.5.2 Exit

Te macro jest wywoływane po wciśnięciu przycisku X na front panelu w trakcie działania aplikacji. Wywołuje ono stany, które powinny zostać wykonane, aby aplikacja została zamknięta w sposób kontrolowany.

4.6 Acquisition

Grupa Acquisition zajmuje się funkcjonalnością polegającą na poprawym odbieraniu klatek z kamery.

4.6.1 Start

Jak sama nazwa wskazuje, stan ten jest wywoływany, gdy użytkownik chce rozpocząć odbieranie klatek z kamery. Można go wywołać poprzez naciśnięcie przycisku Acquire na front panelu. Powoduje on wyszarzenie oraz udostępnienie odpowiednich ikon oraz zmienia zmienną stanu funkcjonalności akwizycji na TRUE.

4.6.2 Acquire

W tym stanie następuje akwizycja klatek z kamery. Dostosowywana jest również rozdzielczość obrazu. Oprócz akwizycji, stan ten odpowiada za wpisanie do kolejki odpowiednich stanów, w zależności od tego, jakie funkcjonalności są obecnie włączone.

4.6.3 Stop

Po kliknięciu w przycisk Stop acquisition, wywoływany jest ten stan. Zmienia on zmienną stanu ten funkcjonalności na FALSE oraz wyszarza i udostępnia odpowiednie ikony. Dodatkowo, ponieważ kolejne funkcjonalności nie działają bez akwizycji obrazu, automatycznie zostają one wyłączone.

4.6.4 Clear

Aby obraz na front panelu po zatrzymaniu aplikacji nie był ostatnią klatką uzyskaną przed zatrzymaniem, należy po wykonaniu procedury zatrzymania funkcjonalności, powrócić do stanu Update UI. Spowoduje to obsługę zamknięcia aplikacji i wyczyszczenia miejsca na obraz.

4.7 Processing

Ta grupa stanów obsługuję funkcjonalność przetwarzania obrazu poprzez OpenCV zaprogramowanie w Pythonie. W ten sposób aplikacja wykrywa kolorowe cukierki i wysyła wiadomość, jak należy obsłużyć konkretny kolor, który pojawił się w polu widzenia kamery.

4.7.1 Start

Stan ten jest wywoływany, gdy użytkownik chce rozpocząć przetwarzanie obrazu z kamery. Można go wywołać poprzez naciśnięcie przycisku Process na front panelu. Powoduje on wyszarzenie oraz udostępnienie odpowiednich ikon oraz zmienia zmienną stanu funkcjonalności przetwarzania na TRUE.

4.7.2 Process

Tutaj obsługiwane jest przetwarzanie obrazu. Ostatnia odebrana klatka zostaje wysłana do Pythona, który po przetworzeniu zwraca obraz z zaznaczonymi znalezionymi kolorami oraz z numerem znalezionego koloru. Numer pozwala w łatwy sposób sterować filtracją, która jest ostatnią funkcjonalnością.

4.7.3 Stop

Po kliknięciu w przycisk Stop process, wywoływany jest ten stan. Zmienia on zmienną stanu tej funkcjonalności na FALSE oraz wyszarza i udostępnia odpowiednie ikony. Dodatkowo, ponieważ funkcjonalność filtracji nie działa bez przetwarzania obrazu. Dlatego też automatycznie zostaje ona wyłączona.

4.8 Filtering

Ostatnia grupa stanów odpowiada za obsługę funkcjonalności filtracji za pomocą układu mikrokontrolera oraz serwa.

4.8.1 Start

Stan ten jest wywoływany, gdy użytkownik chce rozpocząć filtrację cukierków. Można go wywołać poprzez naciśnięcie przycisku Filter na front panelu. Powoduje on wyszarzenie oraz udostępnienie odpowiednich ikon oraz zmienia zmienną stanu funkcjonalności filtrowania na TRUE.

4.8.2 Filter

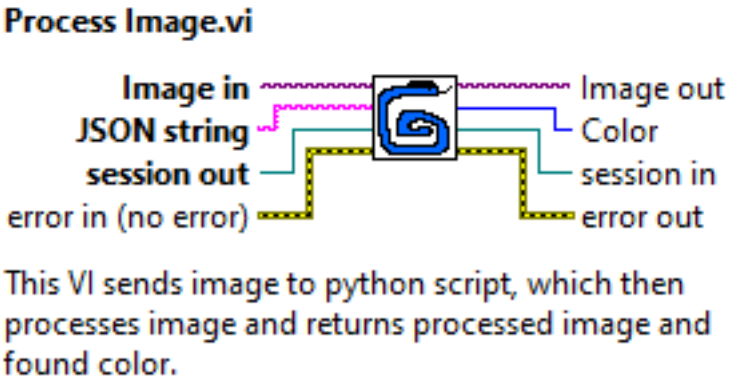
W tym stanie aplikacja komunikuje się z mikrokontrolerem poprzez UART w celu określenia, jaki sygnał mikrokontroler musi wysłać do serwa. Po otrzymaniu wiadomości STM wysyła wiadomość zwrotną, aby zasygnalizować, że sygnał został odczytany poprawnie.

4.8.3 Stop

Po kliknięciu w przycisk Stop filter, wywoływany jest ten stan. Zmienia on zmienną stanu tej funkcjonalności na FALSE oraz wyszarza i udostępnia odpowiednie ikony.

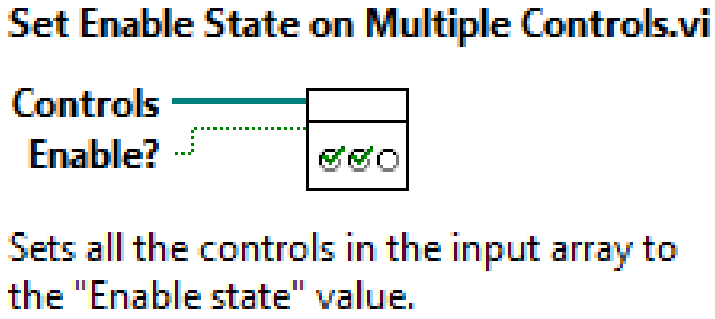
5 Opis stworzonych subVIs

5.1 Process Image



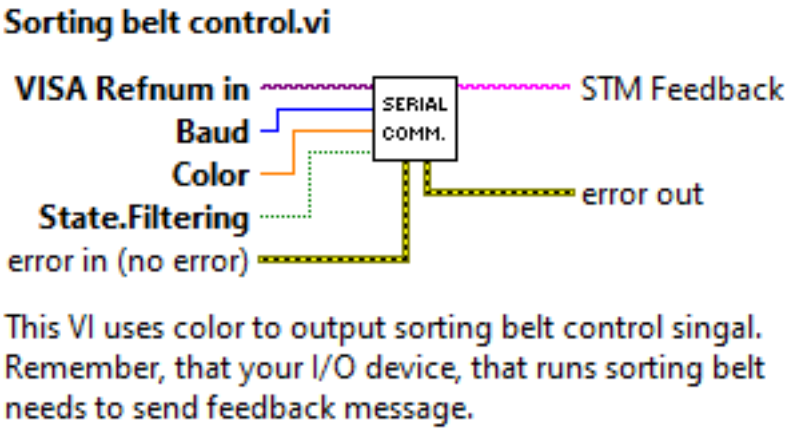
Rysunek 9: Process Image

5.2 Enable State



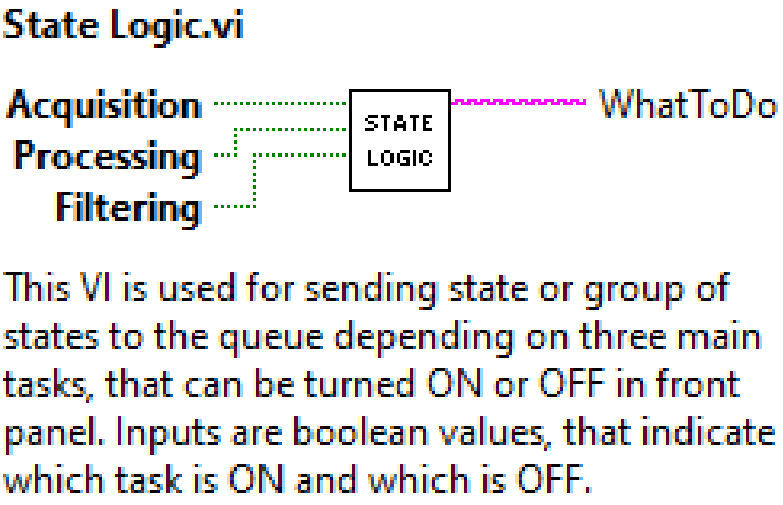
Rysunek 10: Enable state

5.3 Sorting belt control



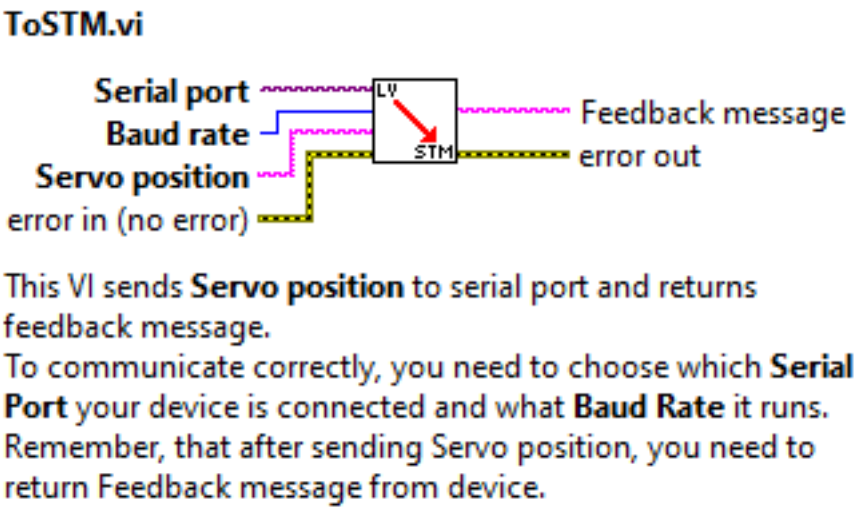
Rysunek 11: Enable state

5.4 State logic



Rysunek 12: State logic

5.5 Przesyłanie danych na stm32



Rysunek 13: ToSTM subVIs