

**WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI**

KATEDRA ELEKTRONIKI

**PRACA DYPLOMOWA INŻYNIERSKA**

*Przetwarzanie obrazów wideo na mikrokontrolerze z rodziny ARM Cortex-M*

*Video image processing on ARM Cortex-M family microcontroller*

Autor: *Mateusz Stechura*

Kierunek studiów: Elektronika i Telekomunikacja

Opiekun pracy: dr hab. inż. Paweł Russek

Kraków, 2018

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631   
z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

*<podpis dyplomanta>*

**Spis treści**

[Wstęp 5](#_Toc501744310)

[Cel pracy 7](#_Toc501744311)

[Rozdział 1 Podstawowe zagadnienia 8](#_Toc501744312)

[1.1 Formatowanie RGB-565 8](#_Toc501744313)

[1.2 Matryca CMOS 9](#_Toc501744314)

[Rozdział 2 Komercyjne systemy analizy obrazu 11](#_Toc501744315)

[2.1 Inteligentna analiza wizyjna BOSH IVA 11](#_Toc501744316)

[2.2 Microsoft KINECT 11](#_Toc501744317)

[Rozdział 3 Część projektowa 12](#_Toc501744318)

[3.1 Opis peryferiów 12](#_Toc501744319)

[3.1.1 Interfejs I2C 12](#_Toc501744320)

[3.1.2 Interfejs SPI 14](#_Toc501744321)

[3.1.3 Interfejs DCMI 16](#_Toc501744322)

[3.1.4 DMA – bezpośredni dostęp do pamięci 17](#_Toc501744323)

[3.2 Opis sprzętu 18](#_Toc501744324)

[3.2.1 Płytka ewaluacyjna STM32F429 Discovery 18](#_Toc501744325)

[3.2.2 Moduł kamery MT9D111 18](#_Toc501744326)

[3.3 Projekt 21](#_Toc501744327)

[3.3.1 Plan testów 23](#_Toc501744328)

[3.3.2 Testy 23](#_Toc501744329)

[Podsumowanie i wnioski 39](#_Toc501744330)

[Streszczenie 40](#_Toc501744331)

[Summary 41](#_Toc501744332)

[Słowa kluczowe 42](#_Toc501744333)

[Keywords 43](#_Toc501744334)

[Bibliografia 44](#_Toc501744335)

[Dodatek A. Spis zawartości dołączonej płyty CD 45](#_Toc501744336)

[Spis ilustracji 46](#_Toc501744337)

[Spis tabel 47](#_Toc501744338)

[Spis stosowanych skrótów 48](#_Toc501744339)

# Wstęp

Motywacją do wybrania tematu przetwarzania obrazów na mikrokontrolerze była chęć przetestowania wydajności nowoczesnego 32-bitowego procesora w przetwarzaniu obrazów. Dokładniejsze poznanie zasady działania peryferiów jakimi dysponuje, zapoznanie się z ich poprawną obsługą na poziomie programowym. Dodatkowym czynnikiem, który wpłynął na wybór pracy było moje głębokie zainteresowanie programowaniem systemów wbudowanych oraz przetwarzaniem obrazów. W efekcie czego powstał pomysł na pracę inżynierską która łączy obie rzeczy ze sobą.

Obecnie praktycznie na każdym kroku towarzyszy nam przetwarzanie obrazów na różnych platformach sprzętowych. Coraz częściej polączenie kamer cyfrowych oraz dodatkowego sprzętu który analizuje zbierane przez kamerę dane realizuje się skomplikowane układy które potrafią przetwarzać obrazy w taki sposób, że są w stanie wykrywać różne anomalie, zmiany, specyficzne zachowania np ludzi, ich gesty czy też mimikę twarzy. Zagadnienie to jest obecne praktycznie wszędzie tam gdzie mamy do czynienia z nowoczesną elektroniką. Telefony komórkowe posiadają aparaty fotograficzne z wykrywaniem twarzy, postacie w grach komputerowych mogą być sterowane na podstawie ruchów naszych ciał, telewizory w domach, lub elektronika w samochodach jest sterowana poprzez proste gesty dłoni. To są przykłady komercyjne, dostępne dla wszystkich mające na celu ułatwienie lub urozmaicenie użytkowania sprzętu elektronicznego. Jednak przetwarzanie obrazów wykorzystywane jest również przez sektory bezpieczeństwa publicznego jak np. wojsko lub policja które wykorzystują zdjęcia satelitarne lub nagrania z monitoringów do zwalczania przestępczośi. Medycyna jest kolejnym sektorem który na szeroką skalę wykorzystuje techniki obrazowe, do analizy stanu zdrowia pacjentów np. wszystkim znany rezonans magnetyczny czy też USG. Sektor przemysłowy podczas wytwarzania elementów nośnych optycznie sprawdza wykonane elementy pod kątem ich jakości oraz wytrzymałości, wyszukuje się mikrouszkodzeń które mogą dyskwalifikować wykonany element z użycia gdyż może zagrażać bezpieczeństwu. Rownież w samochodzach coraz częściej montuje się inteligętne systemy które potrafią zaparkować samochód lub utrzymać go na drodzę na swoim pasie ruchu na podstawie danych które zbierają kamery.

Zagadnienie przetwarzania obrazów jest jak widać bardzo rozległe, często wykorzystuje ono bardzo skomplikowane algorytmy bazujące na sieciach neuronowych. Jednak ta praca będzie się skupiać głównie na pomiarze wydajności sprzętowej zaproponowanego mikroprocesora przy prostym zaproponowanym przez autora algorytmie przetwarzania obrazu. Praca jest traktowana jako wstęp do późniejszej rozbudowy oprogramowania o bardziej złożone metody analizy obrazu.

Praca została podzielona na 3 rozdziały. W pierwszym rozdziale omawiane są podstawowe zagadnienia umożliwiające zrozumienie działania kamer cyfrowych oraz formatowania używanego w aplikacji. Znajduje się tam opis formatu RGB oraz sposób reprezentacji poszczególnych pikseli w postaci binarnej. Kolejnym elementem jest opis działania matrycy CMOS w kamerach cyfrowych oraz krótkie wyjaśnienie w jaki sposób analogowa wartość natężenia światła zamieniana jest na wartość cyfrową rozłożoną na 3 podstawowe barwy jakimi są czerwony, zielony oraz niebieski. W drugiej części natomiast zaprezentowane zostały komercyjne układy które w działaniu bazują na przetwarzaniu obrazów otrzymywanych z kamer. Ostatnim, trzecim rozdziałem pracy jest opis oraz dokładna analiza wykonanego projektu. Przedstawione są tam wyniki pomiarów, charakterystyki poboru prądu przez układ w zależności od różnej konfiguracji, analiza szybkości wyświetlania obrazów (klatki na sekundę) oraz wydajność w przetwarzaniu otrzymywanych obrazów z czujnika CMOS. Dodatkowo w tym rozdziale omówione również zostały podstawowe interfejsy używane w aplikacji takie jak SPI, I2C, DCMI oraz DMA.

# Cel pracy

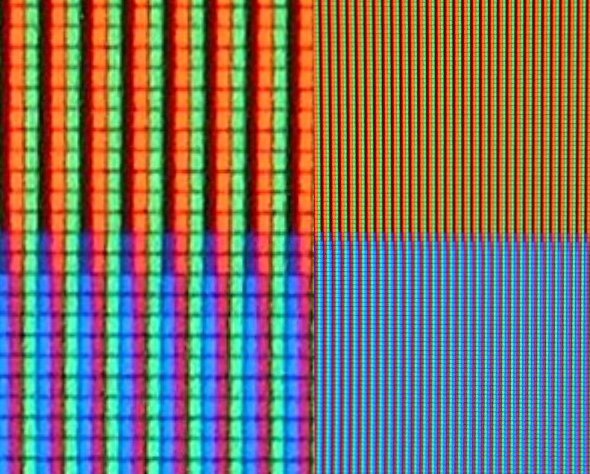
Celem pracy jest zaimplementowanie w języku C prostej aplikacji przetwarzania obrazu na mikrokontroler STM32F429ZIT. Zapoznanie sie z zasadą działania wszystkich interfejsów komunikacyjncyh wykorzystywanych przez układ oraz poprawnym ich oprogramowaniu w środowisku *MDK Keil uVision 5*. Sprawdzenie wydajności mikrokontrolera pod kątem przetwarzania obrazów. Pomiar zużycia energii przez układ w różnych wariantach konfiguracji taktowania procesora oraz różnych trybów pracy peryferiów. Spośród kilku wariantów zostanie przestawiona najlepsza konfiguracja układu która zapewni możliwie najwyższą szybkość działania przy optymalnym poziomie zużycia energii.

# Podstawowe zagadnienia

## Formatowanie RGB-565

Aby zrozumieć istotę przetwarania i analizy obrazow która jest wykorzystywana w tej pracy należy przede wszystkim poznać format w którym przechowywane są obrazy dostarczone przez kamerę cyfrową.

Format RGB jest modelem przestrzeni barw który opisują 3 składowe R-red, G-green oraz B-blue. Z angielskiego są to odpowiednio składowe czerwona, zielona oraz niebieska. W zależności od poziomu każdej składowej barwy możemy uzyskać dowolny kolor. Zwykle element odbiorczy jak np wyświetlacz LCD składa się z pikseli, w którym każdy piksel posiada 3 elementy swiecące w kolorach czerwonym, zielonym oraz niebieskim. Przykładową budowe matrycy pokazuje poniższa ilustracja:



Rysunek 1: Rozkład pikseli na matrycy

Źródło: https://pl.wikipedia.org/wiki/RGB

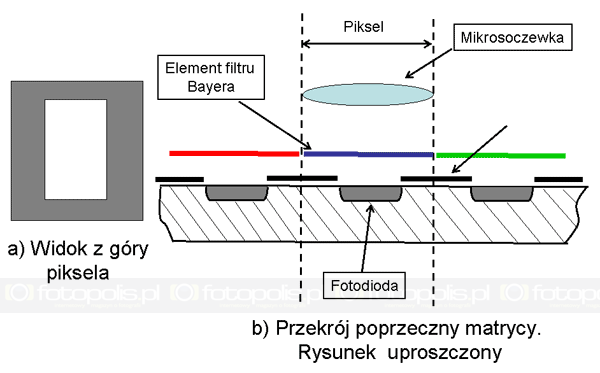
W systemie mikroprocesorowym stosuje się zapis w formacie binarnym czyli za pomocą 0 i 1. Każdy piksel składa się z 3 odcieni R-G-B i zapisany jest w zmiennej zawierającej 16 bitów (przy formatowaniu RGB 565). Aby to zrobić 16 bitów zostało rozdzielone na każdy kolor w ilościach po 5 bitów na kolor niebieski i czerwony oraz 6 bitów na kolor zielony. Użycie 6 bitów na kolor zielony podyktowane jest faktem, że ludzkie oko jest bardziej czułe na stopniowanie koloru zielonego niż niebieskiego czy czerwonego. W aplikacji więc w każdej zmiennej najstarsze 5 bitów odpowiada za nasycenie koloru czerwonego, 5 najmłodszych za nasycenie koloru niebieskiego a pozostałe 6 za kolor zielony.

Tabela 1: Rozmieszczenie bitów w formacie RGB-565

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Format RGB-565 | | | | | | | | | | | | | | | |
| B15 | B14 | B13 | B12 | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| R4 | R3 | R2 | R1 | R0 | G5 | G4 | G3 | G2 | G1 | G0 | B4 | B3 | B2 | B1 | B0 |

## Matryca CMOS

Matryca zbudowana jest z wielu gęsto rozmieszczonych elementów światłoczułych, na ktore naklada się odpowiednie filtry przepuszczające tylko określoną długość światła uzyskując w ten sposób elementy które reagują tylko na wybraną barwę światła.



Rysunek 2: Budowa czujnika matrycy CMOS

Źródło: https://www.fotopolis.pl

Światło padając na fotodiode powoduje wydzielanie się ładunków elektrycznych, ładunek jest większy im większe natężenie światła o danej długości fali pada na fotodiode. Przetwornik następnie konwertuje odczytany poziom natężenia barwy światła z wielkości analogowej na wielkość cyfrową, proces ten zwany jest próbkowaniem i kwantyzacją. Tak przygotowane paczki danych przesyła dalej do mikroprocesora w formacie RGB-565. Ilość danych jest podyktowana rozdzielczością z jaką matryca pracuje.

Dla roździelczości 240x320 pikseli będzie to 76800 zmiennych 16 bitowych czyli 19200 bajtów danych na jeden obraz.

# Komercyjne systemy analizy obrazu

## Inteligentna analiza wizyjna BOSH IVA

Jest to bardzo zaawansowany system analizy wizyjnej. Posiada wiele funkcji które mają na celu poprawę bezpieczeństwa i prywatności.

System ten jest w stanie wykrywać obiekty wchodzące lub opuszczające pewien zdefiniowany obszar, podejrzane zachowania określonego typu, pozostawione nieruchome obiekty np bagaż na lotnisku lub samochód na parkingu, analiza obiektów pod kątem ich rozmiaru, poruszania się, prędkości, zliczanie osób przechodzących przez obszar, wykrywanie wcześniej zdefiniowanego obiektu i wiele innych.

## Microsoft KINECT

Urządzenie wyprodukowane do współpracy z konsolą XBOX 360 oraz XBOX ONE, obecnie nie rozwijane już przez Microsoft, jednak jest nadal wykorzystywane i rozwijane przez inne sektory gospodarki. Organizacja Institute of Creative Technology wykorzystuje urządzenie kinect do wykrywania objawów stersu pourazowego u żołnierzy. Podstawowymi elementami czujnika KINECT są 2 kamery oraz promiennik podczerwieni. Jedna z kamer jest kamerą RGB o roździelczości 640x480. Druga z kamer wraz z promiennikiem podczerwieni odpowiada za pomiar odległości obiektu od kamery. Promiennik podczerwieni wysyła siatkę punktów a kamera z filtrem podczerwonym rejestruje te punkty z których następnie na poziomie programowym można odczytać odległość, kontury postaci oraz ich ruchy.

# Część projektowa

Rozdział 3 został poświęcony dokładnej analizie zrealizowanego projektu przetwarzania obrazu na mikrokontrolerze STM32F429ZIT. W pierwszej części zostały opisane interfejsy mikroprocesora wykorzystywane przy realizacji pracy, znaleźć można w nich podstawowe informacje przybliżające czytelnikowi zasadę działania oraz obsługi peryferiów takich jak I2C, SPI oraz DCMI. Kolejnym etapem jest opis sprzędu używanego w projekcie jak moduł kamery oraz płytka STM32F429 Discovery. Ostatni element części projektowej to przedstawienie oprogramowania, idei oraz doświadczeń przeprowadzonych przez autora.

## Opis peryferiów

### Interfejs I2C

Interfejs I2­C jest interfejsem szeregowym obsługującym transmisję dwukierunkową w trubie half-duplex. W praktyce oznacza to, że w jednym momencie można transmitować tylko w jednym kierunku, bit po bicie w kolejności od najbardziej znaczącego bitu (MSB) do najmniej znaczącego bitu (LSB). Transmisja wykorzustuje 2 linie SDA (Serial Data Line) która odpowiada za przesył danych oraz linię SCL (Serial Clock Linie) która jest linią zegarową. Taktowana jest ona ze z góry określoną częstotliwością która to definiuje prędkość transmisji.

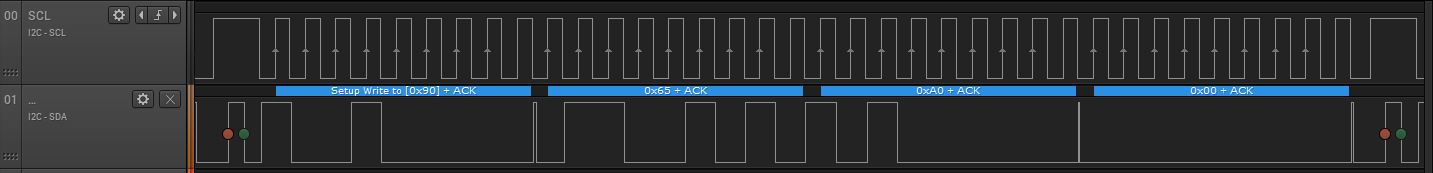


Rysunek 3 Struktura Ramki I2C

Żródło: https://pl.wikipedia.org/wiki/I%C2%B2C

Na rysunku została przedstawiona ramka sygnału I2C. Analizując ramkę można dostrzec pewien schemat przesułu danych. Urządzenie typu Master rozpoczyna transmisję wysyłając bit START czyli podczas trawnia stanu wysokiego na lini SCL następuje zmiana stanu z wysokiego na niski na lini SDA. Następnie przesyłanych jest kolejne 8 bitów z zastrzeżeniem, że zmiany stanu na lini SDA mogą następować jedynie w momencie gdy na lini SCL występuje stan niski. Koniec transmisji poprzedzony jest wysłaniem bitu STOP który jest analogiczny do bitu START z tym, że podczas stanu wysokiego na lini SCL linia danych SDA zmienia stan z niskiego na wysoki.

W praktyce komunikacja jest troszkę bardziej skomplikowana, poniżej przykładowa ramka zapisu wartości

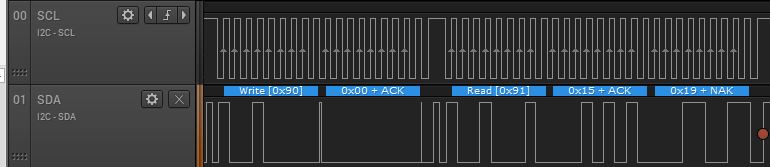


Rysunek 4 Zapis poprzez I2C wartości 0xA000 do rejestru 0x65

Transmisja rozpoczyna się od przesłania przez urządzenie master bitu start. Następnie przesyłany jest 8-bitowy adres urządzenia na magistrali I2C do którego zapis ma być wykonany, jeżeli adres jest poprawny to ostatni 9 bit ma wartość 1, jest to tak zwany bit ACK potwierdzający poprawność przesłania danych do urządzenia jest on również generowany przez urządzenie do którego ma nastąpić transmisja. Stan niski oznaczałby błąd – NACK. Kolejnym etapem jest przesłanie 8 bitowego adresu w pamięci urządzenia pod który chcemy wpisać pewną wartość, transmisja ta również potwierdzana jest komunikatem ACK. Teraz następuje przesył danej 16 bitowej. Transmisja ta ze względu na charakterystykę pracy interfejsu została pdozielona na 2 osobne transmisje 8 bitowe. Wartość 0xA000 binarnie wynosi 1010 0000 0000 0000 co po przeliczeniu zgadza się z wartościami odczytanymi przy kolejnych stanach wysokich lini SCL. Na koniec linia SDA przechodzi w stan wysoki podczas trwania na lini SCL stanu wysokiego co oznacza bit STOP i transmisja kończy się.

Wspomniany wcześniej adres urządzenia zapisywany jest na 7 bitach co oznacza, że na jednym interfejsie I2C możemy umieścić aż 127 różnych urządzeń. Proces zapisu lub odczytu definiujemy dopisując do adresu urządzenia na pozycji najmniej znaczącego bitu wartość 0 jeżeli chcemy wykonać operację zapisu do urządzenia lub wartość 1 jeżeli chcemy z urządzenia odczytać jakieś dane.

Proces odczytu jednak wygląda inaczej. W pierwszej kolejności należy do urządzenia operacją zapisu wpisać adres pamięci z której będziemy odczytywać a następnie przesłać komendę odczytu czyli najczęściej adres urządzenia zwiększony o 1, teraz urządzenie typu slave odeśle nam dane które znajdują się pod żądanym adresem w jego pamięci. Co ważne, przy tym kierunku transmisji przesłanie przez urządzenie slave ostatniej paczki danych zakończy się komędą NACK + bit STOP.

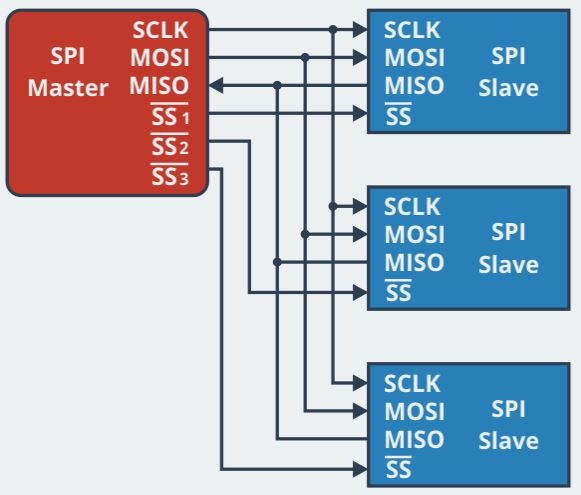


Rysunek 5 Proces odczytu poprzez I2C z adresu 0x00

Warto zaznaczyć, że przykład oczytu jak i zapisu powyżej zostały oparte o proces zpisu dla modułu kamery MT9D111 użytej w projekcie. Dla innych urządzeń proces ten może być inaczej zdefiniowany w dokumentacji konkretnego czujnika, lecz ogólna zasada działania magistrali pozostaje bez zmian.

### Interfejs SPI

Szeregowy interfejs komunikacyjny który w odróżnieniu od I2C może pracować w trybie Full-duplex a więc możliwa jest równoczesna transmisja w dwuch kierunkach (nadawanie oraz odbieranie danych). Urządzenia na magistrali SPI dzielą się na urządzenia typu master oraz slave (nadrzędne oraz podrzędne). Urządzenie nadrzędne zapewnia sygnał zegarowy (SCK) dla zewnętrznych urządzeń typu slave. Wyborem urządzenia z którym ma nastąpić transmisja sterujemy odpowiednim stanem na wyjściu CS (chip select, czasem równie oznaczane jako SS ), jeżeli wyjście to jest w stanie niskim oznacza transmisję do urządzenia slave który jest podpięty pod ten pin. Mikroprocesor STM32F429 pin ten oznacza jako NSS i może być on sterowany na poziomie programowym jak i sprzętowym. W interfejsie obecne są jeszcze 2 linie danych oznaczane jako MISO (master input slave output) oraz MOSI (master output slave input) i są to odpowiednio linie do odbioru oraz nadawania danych dla urządzenia master.



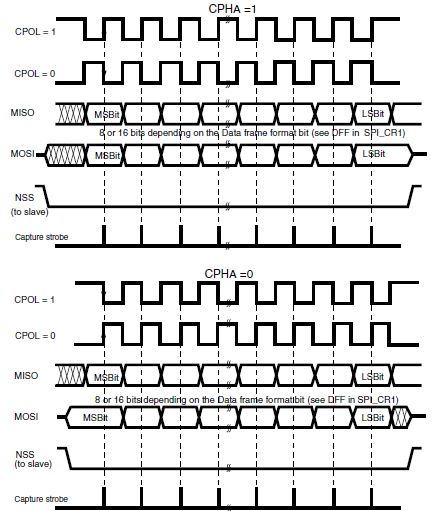
Rysunek 6 Podłączenie urządzeń przy użyciu interfejsu SPI

Źródło: https://forbot.pl/blog/kurs-stm32-f4-10-obsluga-spi-wyswietlacz-oled-id13475

Transmisja danych odbywa w zależności od konfiguracji interfejsu przy narastajacym lub opadającym zboczu lini SCK. Pracę układu konfigurujemy bitami oznaczanymi jako CPOL oraz CPHA.

Parametr CPOL definiuje w jakim stanie jest linia zegarowa jeżeli układ jest w stanie bezczynności, jeżeli parametr ten wynosi 0 oznacza to, że w trakcie bezczynności synał SCK jest w stanie niskim, dla wartości 1 jest odwrotnie.

Parametr CPHA określa przy którym ze zboczy sygnału SCK odbywa się transmisja danych. Są 2 możliwości, transmisja na zboczu które przechodzi ze stanu aktywnego w stan bezczynności (CPHA = 1) oraz transmisja na zboczu przechodzącym ze stanu bezczynności w stan aktywny. Dokładnie przedstawia tą zależność rysunek 7.

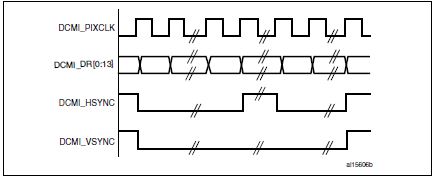


Rysunek 7 Przebiegi czasowe komunikacji SPI

Źródło: STM32F429 Reference Manual

### Interfejs DCMI

Jest to szeregowy oraz równoległy interfejs mogący pracować w konfiguracji 8,10,12 lub 14 bitów oraz z różnymi formatami obrazu takimi jak YcbCR4:2:2, RGB565 oraz JPEG. Kamera dołączona do interfejsu powinna dysponować co najmniej 8 liniami danych lub maksymalnie 14. Synchronizacja pracy kamery oraz interfejsu jest zapewniona dzięki lini sygnalizującej nadawanie kolejnych pikseli z kamery (PIXCLK), lini sygnalizującej nadawanie każdej kolejnej lini obrazu (HSYNC), która jest opcjonalna i nie jest konieczna do poprawnej pracy. Ostatnią linią synchronizującą pracę jest sygnał VSYNC który informuje o rozpoczęciu nadawania każdej kolejnej ramki obrazu.



Rysunek 8 Ramka DCMI

Źródło: STM32F429 Reference Manual

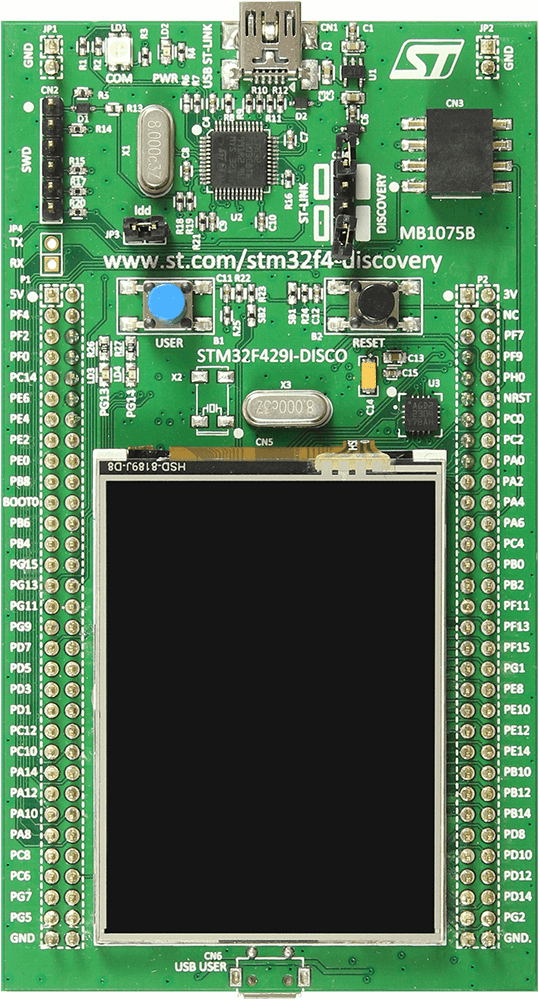
Zmiana stanu sygnału VSYNC na niski informuje interfejs, że kamera zaczyna nadawanie kolejnej ramki obrazu, sygnał HSYNC w stanie niskim informuje, że nadawana jest kolejna linia obrazu, natomiast każda zmiana sygnału PIXCLK informuje o kolejnym nadawanym pikselu obrazu. Oczywiście polaryzacja syngałów jest opcją konfigurowalną i w niektórych przypadkach stan wysoki może oznaczać stan aktywny. Do dyspozycji mamy 2 podstawowe tryby pracy interfejsu czyli tryb wykonania zdjęcia w którym odbierana jest jedna ramka obrazu lub tryb ciągły, w nim interfejs pozostaje aktywny i odbiera dane do momentu gdy sami go wyłączymy.

### DMA – bezpośredni dostęp do pamięci

DMA czyli Direct Memory Access jest techniką która umożliwia dostęp do pamięci RAM bez użycia rdzenia procesora, czyli na pracę jakby w tle. Odbierając dane z przykładowo interfejsu SPI w tradycyjnym przypadku odbiór ten musiałby obsłużyć procesor, wykonać kopiowanie odebranych danych z rejestru interfejsu do pamięci RAM. Przy użyciu DMA operacja kopiowania danych do pamięci wykonywana jest bezpośredniu czyli bez uzycia procesora którego moc obliczeniowa może być wykorzystana w innym celu. Dodatkowo szybkość działania peryferiów w mikroprocesorze jest zazwyczaj dużo mniejsza niż prędkość procesora a więc nie wykorzystując techniki DMA bardzo duzo czasu procesor marnuje na czekanie aż jakieś dane zostaną odebrane zanim będzie mógł zacząć je kopiować do pamięci. Interfejs ten może kopiować dane między peryferiami a pamięcią w obu kierunkach a także między pamięcią z jednego w drugie miejsce.

## Opis sprzętu

### Płytka ewaluacyjna STM32F429 Discovery

Sercem modułu jest mikrokontroler STM32F429ZIT którego najważniejsze parametry to:

Rysunek 9 STM32F429-DISCOVERY

Źródło: www.st.com

* Architektura 32 bitowa
* Taktowanie rdzenia 180MHz
* 2MB pamięci flash
* 256+4kB pamięci SRAM
* 168 portów wejścia/wyjścia
* Szeroka gama interfejsów w tym między innymi I2C, SPI, DMA, UART

Sama płytka zawiera w sobie wiele dodatkowych modułow takich jak wbudowany programator oraz debugger ST-LINK, wyświetlacz TFT LCD o przekątnej 2.4 cala z panelem dotykowym i roździelczości 240x320 pikseli, czujnik żyroskopowy oraz pamięć SDRAM o pojemności 64MB.

### MT9D111Moduł kamery MT9D111

Rysunek 10 Moduł MT9D111

Źródło: http://ultimumvitae.com

Moduł MT9D111 jest cyfrową kamerą o maksymalnej roździelczości UXGA tj. 1600x1200 pikseli. Układ pracuje w opraciu o interfejs DCMI o 8 bitowej szynie danych dzięki czemu według zapewnień producenta maksymalna przepustowość danych wynosi 80MB/s czyli dla roździelczości 1600x1200 15klatek/s lub dla roździelczości 800x600 30 kl/s. Moduł posiada zintegrowany procesor obrazu, mikrokontroler, automatyczną regulację ostrości oraz wbudowany koder JPEG. Przesyłanie danych z czujnika CMOS do mikrokontrolera odbywa się przez interfejs DCMI, konfigurację czujnika oraz dostęp do rejestrów zapewnia komunikacja poprzez I2C.

Układ można konfigurować na bardzo wiele sposobów, przede wszystkim konfigurowalna jest rozdzielczość obrazu oraz format danych wyjściowych. Szybkość przepływu danych zależy od szybkości taktowania, układ można taktować sygnałem pochodzącym z mikroprocesora lub skonfigurować wewnętrzny generator VCO do taktowania układu. Maksymalna częstotliwość pracy kamery to 80MHz co oznacza, że z taką częstotliwością kamera będzie nadawać kolejne piksele obrazu. Częstotliwość ta jednak w projekcie nie będzie mogła zostać zastosowana z uwagi na ograniczenia modułu DCMI w mikrokontrolerze który narzuca maksymalną częstotliwosć 54MHz.

Oczywiście dzięki zintegrowanemu procesorowi obrazu możemy również zapisując odpowiednie rejestry w czujniku zdefiniować różne parametry obrazu jak jasność, kontrast, nasycenie, skorygować poszczególne barwy a nawet ustawić szybkość migawki.

Za częstotliwość taktowania modułu odpowiadają następujące parametry: fPFD, fVCO oraz fOUT, ostatnia wartość oznacza częstotliwość taktowania pętli PLL. Do konfigulacji bloku PLL trzeba doprowadzić zewnętrzny sygnał zegarowy oznaczany jako fIN. Zależności przedstawia tabela 2.

Tabela 2 Taktowania syngałów MT9D111

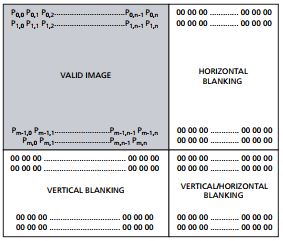
|  |  |  |  |
| --- | --- | --- | --- |
| Częstotliwość | Definicja | Wartość minimalna (MHz) | Wartość maksymalna (MHz) |
|  |  | 2 | 13 |
|  |  | 110 | 240 |
|  |  | 6 | 80 |
|  |  | 6 | 64 |

Wartości N, M oraz P są wartościami ustawianymi w rejestrach kamery. Operując na nich oraz sterując częstotliwością wejściową można ustawić żądaną częstotlwiosć.

Kolejne rejestry odpowiadają za ustawianie roździelczości obrazu, oraz korzystanego trybu decymacji czyli zmniejszenia fizycznej roździelczości obrazu. Termin decymacja oznacza wycinanie pewnych wartości lub pomijanie któryś pikseli. Jednak tutaj termin decymacji oznacza troszkę coś innego. Kamera w zależności od ustawionego wyjściowego rozmiaru obrazu dzieli oryginalny obraz na żądaną ilość pikseli, z sąsiednich pikseli które są nadmiarowe oblicza wartość średnią i zapisuje ją jako jeden piksel. Dzięki takiej operacji nie tracimy pierwotnego pola widzenia oraz nie wprowadza się aliasingu. Okresla się wagę decymacji według wzoru:

Oraz wprowadza tak wyliczoną wartość w odpowiednie rejestry kamery.

Kolejnym krokiem jest ustawienie parametrów HORIZONTAL\_BLANKING oraz VERTICAL\_BLANKING. Są to dodatkowe linie oraz wiersze w obrazie kamery które nie niosą żadnej informacji lecz obniżają one częstotliwość wychodzących ramek obrazu z kamery oraz przez wszystkim dają matrycy czas na przetworzenie obrazu. Po każdej nadanej lini obrazu następuje przerwa w nadawaniu oznaczona jako HORIZONTAL\_BLANKING, po nadaniu wszystkich lini obrazu następuje przerwa czasowa oznazona jako VERTICAL\_BLANKING. Długość przerw jest konfigurowana w rejestrach do których wpisuje się wartości liczbowe odpowiadające pustym liniom oraz wierszą, czas można obliczyć znając taktowanie lini PIXCLK.



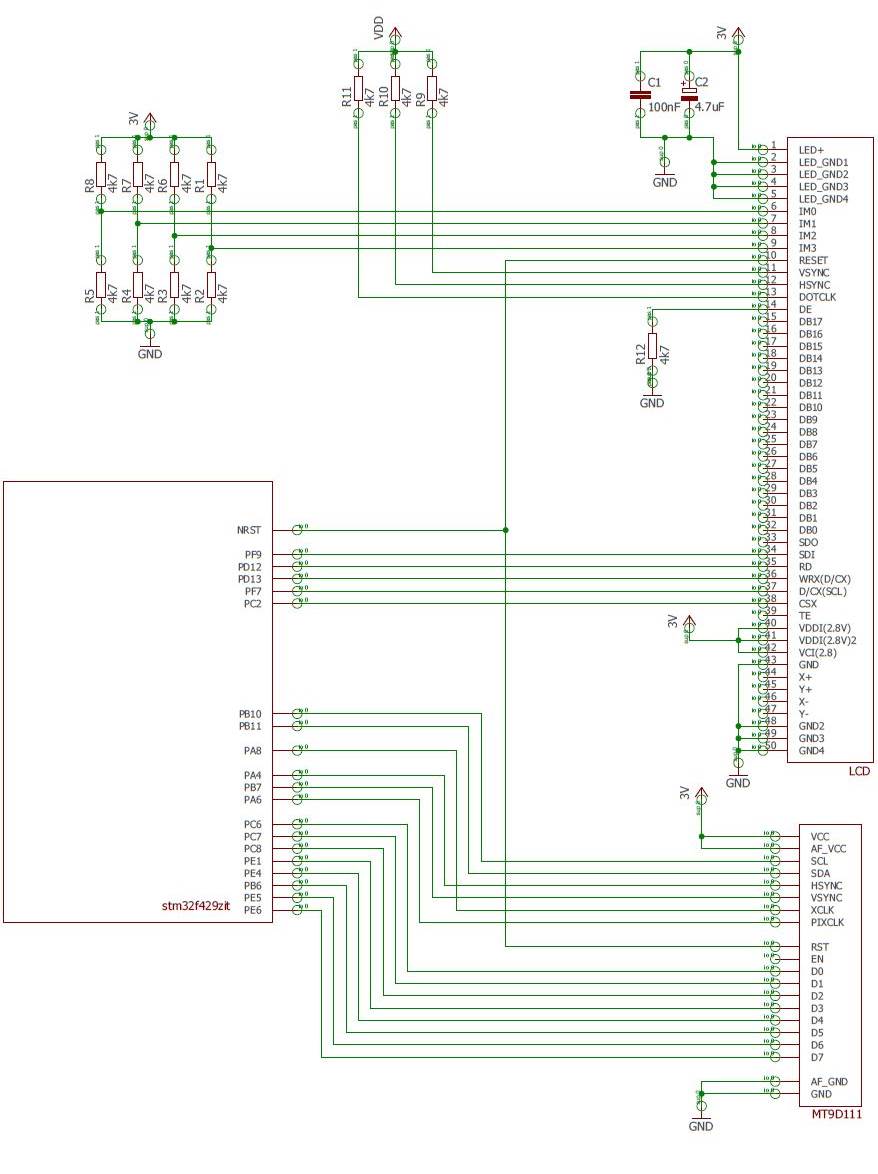
Rysunek 11 Odczyt ramki obrazu

Źródło: MT9D111 Datasheet

## Projekt

Podstawowa funkcjonalność modułu DISCOVERY została powiększona o sensor obrazu MT9D111 wpięty do interfejsu DCMI. Połączenia wyświetlacza LCD do mikrokontrolera pozostały bez zmian. Wyświetlacz obsługuje sterownik ILI9341 który może działać w oparciu o równoległy interfejs o maksymalnej szerokości magistrali 18 bitów. Niestety okazało się, że interfejs DCMI do obsługi strumienia danych z kamery dzieli niektóre piny mikroprocesora z modułem kontrolera wyświetlacza o nazwie LTDC. Z Uwagi na priorytet obsługi kamery zdecydowano na obsługę wyświetlacza poprzez szeregowy interfejs SPI.

Częściowy schemat układu przedstawia rysunek 9 na którym pominięto szczegółowe połączenia oraz elementy dostępne na płytce STM32F429-DISCOVERY a skupiono się na elementach wykorzystywanych w projekcie oraz połączeniami występującymi między nimi a mikrokontrolerem.



Rysunek 12 Schemat układu

### Plan testów

Testy mają wykazać wydajność procesora w operacjach obrazowych. Uznano, że najlepszą jednostką będzie wydajność wyrażona w klatkach na sekundę wyświetlanych na wyświetlaczu. Pomiar dokonywany jest na poziomie programowym zliczając ilość przetworzonych klatek obrazu. W przerwaniu sprawdzany jest warunek czy upłynęła pełna sekunda od poprzedniego pomiaru jeżeli warunek jest prawdziwy następuje odczyt zmiennej która jest inkrementowana po każdej przesłanej ramce obrazu, wartość tej zmiennej to prędkość w klatkach na sekundę. Wartosć ta następnie jest zerowana aby poprawnie zmierzyć prędkość w kolejnej sekundzie. Przerwania generowane są w oparciu o skonfigurowany timer zwany SysTick który generuje przerwania co 1ms.

Wydajność będzie zależała od częstotliwości taktowania rdzenia głównego mikroprocesora oraz konfiguracji generatora VCO w module kamery MT9D111. Taktowanie z kolei będzie znacznie wpływać na pobór energii. Przetestowana będzie wydajność oraz zużycie energii w zależności od taktowania rdzenia głównego oraz pętli PLL kamery. Dla najlepszej konfiguracji zostanie dodana implementacja przetwarzania obrazu oraz zbadana różnica w wydajności. Aby znaleźć element najbardziej ograniczający prędkość pracy układu zostanie również osobno zbadana prędkość transmisji poprzez SPI oraz realna prędkość generowania obrazu z kamery.

### Testy

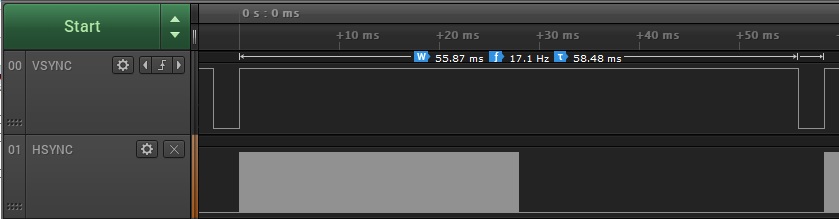
#### Test wydajności modułu kamery

Pierwszym krokiem było przetestowanie możliwości samego modułu kamery. W tym celu zgodnie z notą katalogową zostały skonfigurowane odpowiednie rejestry w kamerze ustawienie maksymalnego możliwego taktowania jakim jest 80MHz oraz minimalnych wartości HORIZONTAL, oraz VERTICAL\_BLANKING (odpowiednio 348 oraz 32) w celu skrócenia do minimum czasu oczekiwania między kolejnymi liniami obrazu oraz między ramkami. Jako sygnał zegarowy kamery służy wyjście mikrokontrolera PA8, pin ten został skonfigurowany do obsługi funkcji systemowej oznaczonej jako MCO1. Na pinie dostępna jest częstotliwość zegara podzielona przez preskaler o zdefiniowanej przez użytkownika wartości. W przykładzie preskaler został ustawiony na 3, więc dzieli częstotliwość zegara CPU przez 3.

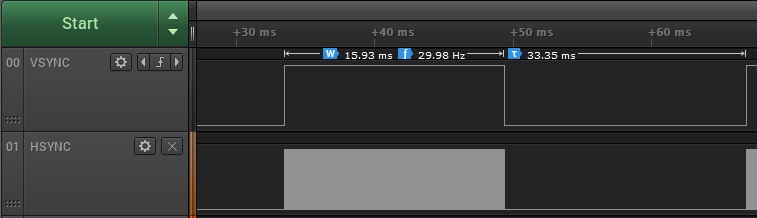
Sygnał ten jest wykorzystywany do konfiguracji bloku PLL kamery (patrz [3.2.2 Moduł kamery MT9D111](#_Moduł_kamery_MT9D111))

Roździelczość obrazu wyjściowego ustawiona na 1600x1200 pikseli, wyłączona opcja decymacji obrazu. Pomiar powtórzony dla roździelczości 800x600.

Częstotliwość wysyłanych przez kamerę ramek obrazu zmierzona została jako wartość częstotliwości lini VSYNC.



Rysunek 13 Prędkość transmisji dla roździelczości 1600x1200 oraz taktowania PLL 80MHz



Rysunek 14 Prędkość transmisji dla roździelczości 800x600 oraz taktowania PLL 80MHz

Jak widać przy maksymalnym taktowaniu bloku PLL udalo się osiągnąć wartości 17 klatek na sekundę dla roździelczości 1600x1200 oraz prawie 30 klatek dla roździelczości 800x600. Są to wartości zgodne z dokumentacją czujnika.

Niestety takiej prędkości nie jest w stanie obsłużyć moduł DCMI mikrokontrolera. W tym przypadku maksymalna częstotlwiość taktowania wynosi 54MHz. W tym celu skonfigurowano kamerę następującymi wartościami:

A wartość preskalera częstotliwości taktowania CPU zmieniono na 4 co dało wyjściową częstotliwość 45 MHz. Powtarzając obliczenia częstotliwość jaką udalo się uzyskać to 53MHz co ostatecznie przy roździelczości 800x600 umożliwiło otrzymanie prędkości nadawania na poziomie 20 klatek na sekundę.

#### Wpływ pomniejszania roździelczości opcją decymacji na pracę układu.

Decymacja polega na tworzeniu 1 piksela obrazu z kilku sąsiednich wyliczając ich średnią wartość która interpretowana jest następnie jako jeden piksel a następnie moduł kamery przesyła takie dane poprzez interfejs DCMI. Aby tą średnią wartość można wyliczyć moduł kamery musi jednak odczytać z matrycy wszystkie piksele z pierwotnego rozmiaru co zajmuje czas. Wyjaśnia to fakt, że przy ustawieniu roździelczości 1600x1200 w rejestrach kamery oraz zdefiniowaniu decymacji obrazu do roździelczośći 320x240 nie zwiększymy szybkości nadawania kolejnych ramek obrazu ponieważ czujnik z matrycy i tak musi odebrać wszystkie piksele a następnie je konwertuje do innej roździelczości.

Z rozważań z poprzedniego akapitu wydajność kamery dla roździelczości 800x600 okazała się najbardziej zadowalająca i taka roździelczość została wybrana jako podstawowa do dalszych rozważań wraz z uruchomioną opcją decymacji obrazu do roździelczości 320x240 pikseli jaką wspiera wyświetlacz LCD.

#### Wpływ taktowania magistrali SPI na pobór energii oraz pracę układu.

Maksymalna częstotliwość taktowania modułu SPI w mikrokontrolerze STM32F429 wynosi 90MHz. Do pomiaru wydajności zastosowano pomiar zaimplementowany w programie, który po każdej przesłanej ramce obrazu do wyświetlacza LCD inkrementuje wartość odpowiedniej zmiennej. Następnie w przerwaniu generowanym przez zegar SysTick w momencie gdy upłynie 1 sekunda zmienna jest odczytywana oraz wyświetlana na wyświetlaczu. Testy tu zawarte nie pokazują jednak realnej wydajności samego modułu SPI. Trzeba pamiętać, że przed przesłaniem kolejnych pikseli, które są przechowywane przez zmienne 16 bitowe należy je odpowiednio przygotować do transmisji przesyłając najpierw starsze 8 bitów a następnie młodszą ich część. Operacje te zajmują pewien okres czasu który również jest zawarty w pomiarze.

Tabela 3 Wpływ taktowania SPI na wydajność

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Częstotliwość SPI [MHz] | Częstotliwość CPU [MHz] | Częstotliwość PLL [MHz] | I[mA] | Prędkość transmisji SPI [kl/s] |
| 90 | 180 | 53 | 300 | 28 |
| 45 | 290 | 14 |
| 22.5 | 270 | 7 |
| 11.25 | 260 | 4 |

Z tabeli można zauważyć zależność wydajności od taktowania, Oczywiście zmniejszając prędkość taktowania SPI o połowę czas przesyłu takiej samej paczki danych musiał ulec dwukrotnemu wydłużeniu, tak też się dzieję. Jednak im częstotliwość magistrali jest mniejsza tym pobór energii maleje. Przy obniżeniu taktowania z 90 na 45MHz pobór prądu obniżony jest o jedynie 10mA a dodatkowo spada dwukrotnie prędkość transmisji z 28 na 14 klatek na sekundę. Na potrzeby projektu ustawiono więc taktowanie 90MHz jako opcję najbardziej. Prędkość SPI wynosi więc 28kl/s a prędkość z jaką kamera dostarcza obraz wynosi 20kl/s. W takim wypadku transmisja SPI nie będzie negatywnie skutkować na pracę układu oraz go spowalniać, a płynność obrazu będzie zależała głównie od pracy kamery.

#### Badanie wpływu częstotliwości pętli PLL kamery na prędkość transmisji

W tej części zostanie zmierzony wpływ taktowania pętli PLL kamery na prędkość nadawanych ramek oraz na pobor energii przez układ. Pomiar wykonywany przy pomocy analizatora stanów logicznych. Mierzona była czestotliwość lini VSYNC która jak wiemy odpowiada za synchronizacje w nadawanych kolejnych ramkach brazu. Gdy jest w stanie wysokim nadawana jest ramka obrazu. Mierząc częstotliwość tej lini w dostajemy tak na prawdę zarówno prędkość w klatkach na sekundę.

Tabela 4 Pomiary wpływu pętli PLL na układ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Częstotliwość PLL [MHz] | Częstotliwość CPU [MHz] | Częstotliwość SPI [MHz] | I[mA] | VSYNC[Hz] |
| 53 | 180 | 90 | 300 | 19.88 |
| 45 | 300 | 17.4 |
| 37 | 290 | 13.9 |
| 31 | 285 | 11.6 |
| 26 | 280 | 9.9 |
| 18 | 270 | 7 |
| 10 | 260 | 3.8 |
| 6 | 255 | 3 |
| Camera Off | 220 | 0 |

Pobór energii przez sam moduł kamery można obliczyć wykorzystując ostatni wspis z tabeli. Jego pomiar został przeprowadzony przy odłączonej lini zasilającej kamery. Nota katalogowa kamery mówi o zużyciu energii na poziomie 348mW przy pełnej roździelczości oraz maksymalnym taktowaniu 80MHz. Kamera zasilana jest napięciem 3V stąd:

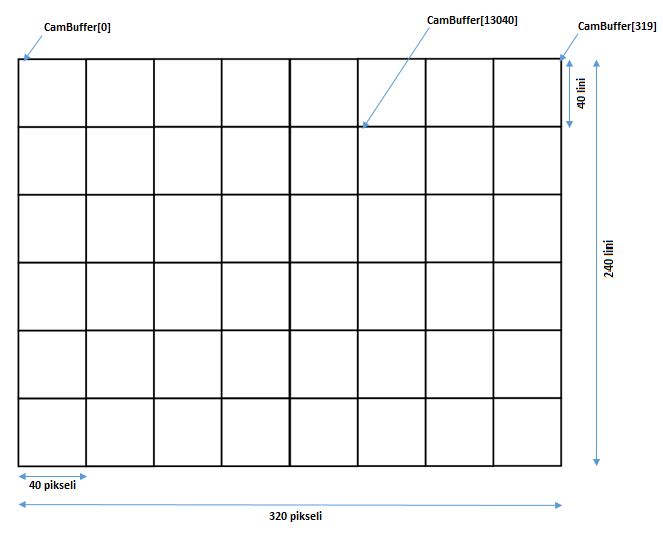
Obliczając pobór prądu przy naszej konfiguracji otrzymujemy:

Troszkę niższa wartość spowodowana jest niższym taktowaniem pętli PLL kamery oraz roździelczością pracy 800x600 co powoduje obniżenie poboru energii.

Analizując wyniki jedyną akceptowalną wartością taktowania pętli PLL jest 53MHz. Spowodowane jest to faktem, że oko ludzkie powyżej 24kl/s widzi obraz jako płynny, niestety mniejsze prędkości są już zauważalne dlatego też zdecydowano na uzycie maksymalnej częstotliwości pracy aby dostarczyć jak najbardziej płynny obraz.

#### Implementacja algorytmu przetwarzania obrazu

Jest to stosunkowo prosty algorytm który działa po odebraniu ramki obrazu i przekopiowaniu go poprzez DMA do pamięci mikroprocesora. Algorytm ma za zadanie znaleźć najjaśniejszy element w obrazie i zaznaczyć go czerwonym prostokątem. Zadanie to realizuje funkcja *findWhiteZone* w programie. Działa ona w oparciu o 2 tablice, jedna z nich to tablica 76800 elementowa o nazwie *CamBuffer* w której przechowywane są piksele obrazu. Drugą tablicą jest tablica o wymiarach 8x6 o nazwie *searchZone[8][6]* która definiuje obszary w których może znajdować się szukany element.



Rysunek 15 Podział ramki obrazu na obszary

Algorytm sprawdza każdy kolejny piksel z tablicy czy jego wartość jest większa od wartości 0xCD14 która stanowi około 80% maksymalnej wartości czyli 0xFFFF czyli wartości odpowiadającej maksymalnemu natężeniu świecenia każdej składowej barwy z palety RGB co w efekcie daje kolor biały. Jeżeli piksel jest jaśniejszy to algorytm oblicza w którym prostokącie piksel się znajduje i inkrementuje zmienną w tablicy searchZone która odpowiada sprawdzanemu pikselowi. W przypadku gdy sprawdzany jest piksel z tablicy CamBuffer[13040] jeżeli jest on jasniejszy od wyznaczonego poziomu to inkrementowana będzie zmienna w tablicy searchZone[6][2]. Kolejnym etapem jest przeszukanie tablicy searchZone w poszukiwaniu prostokąta w którym znalazło się najwięcej pikseli o odpowiedniej jasności. Ostatnim krokiem jest zaznaczenie tego obszaru prostokątem do czego służy funkcja *putrectangle(uint16\_t a, uint16\_t b, uint8\_t size),* która przyjmuje jako swoje argumenty środek prostokąta do zaznaczenia oraz długość boku w pikselach. Ciało funkcji zamieszczone zostało poniżej. Tablice CamBuffer oraz searchZone to tablice zdefiniowane jako zmienne globalne.

void findWhiteZone(void)

{

uint16\_t\* bitptr = CamBuffer;

uint8\_t maxX = 0;

uint8\_t maxY = 0;

uint8\_t xC = 0;

uint8\_t yC = 0;

uint8\_t tempLine;

// Clear searchZone tab

for(int x = 0; x < 8; x++)

{

for(int y = 0; y < 6; y++)

{

searchZone[x][y] = 0;

}

}

// Increment corresponding matrix zone if pixel inside is greater than some brightest level

for(int i = 0; i < 76800; i++)

{

if(\*(bitptr+i) > 0xCD14)

{

tempLine = i/320;

yC = (tempLine/40);

xC = (i-320\*(tempLine))/40;

searchZone[xC][yC] += 1;

}

}

//Check witch zone of picture is brightest

for(int x = 0; x < 8; x++)

{

for(int y = 0; y < 6; y++)

{

if(searchZone[maxX][maxY] < searchZone[x][y])

{

maxX = x;

maxY = y;

}

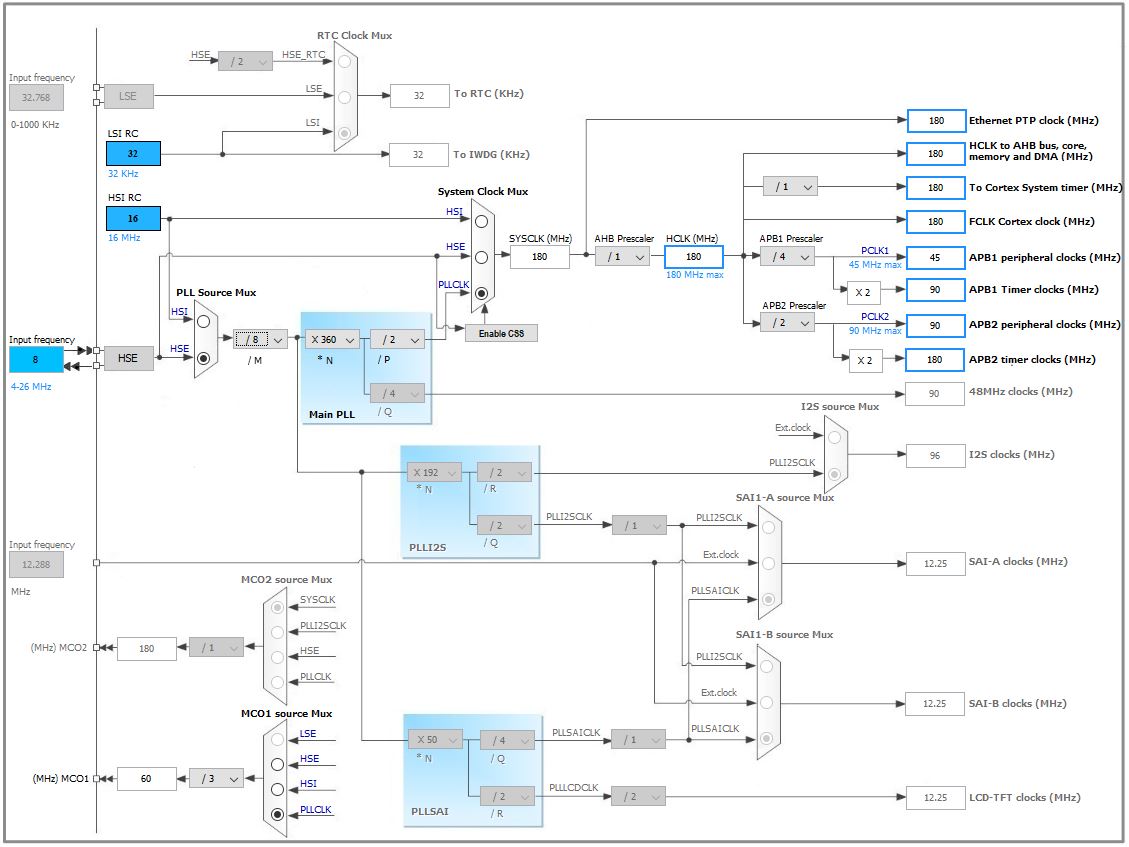
}

}

putrectangle(maxX\*40+20,maxY\*40+20,40);

}

#### Badanie wpływu taktowania rdzenia mikrokontrolera na prędkość przetwarzania obrazu oraz pobór energii



Rysunek 16 Konfiguracja zegarów STM32F429ZIT

Źródło: STM32CubeMX

Badany będzie wpływ taktowania głównego zegara oznaczonego jako SYSCLK na schemacie powyżej na pracę układu w szczególności na pobór energii. Z uwagi na fakt, że zegar modułu SPI wykorzystywanego w projekcie podpięty jest do magistrali APB2 której taktowanie wyliczane jest na podstawie zegara głównego CPU (SYSCLK) niemożliwym będzie utrzymanie taktowania SPI na stałym poziomie 90MHz. Szyna APB2 Peripherial Clocks może być taktowana maksymalnie z częstotliwością 90MHz, Dostępne dzielniki częstotliwości to wartości 1,2,4,8,16 co oznacza, że dla częstotliwości SYSCLK mniejszych od 90MHz magistrala APB2 a tym samym moduł SPI taktowany może być z maksymalną częstotliwością równą częstotliwości SYSCLK. Dodatkowo należy pamiętać, że taktowanie pętli PLL obliczane jest na podstawie częstotliwości generowanej przez mikrokontroler na wyjściu PA8 według wzoru:

Zmieniając częstotliwość zmieniać się będzie również taktowanie pętli PLL kamery. Dla każdego pomiaru będzie więc konieczne obliczanie nowych wartości M, N oraz P które konfigurują pętle PLL i są wpisywane do rejestrów kamery przy konfiguracji aby utrzymać stałą wartość taktowania pętli na poziomie 53MHz. Dzięki temu zabiegowi utrzymana zostanie stała prędkość nadawania przez kamerę obrazu tj prędkości niecałych 20 klatek/s. Mierzona prędkość w jednostce kl/s opisuje czas od momentu otrzymania ramki obrazu, jej przetworzenie i przesłanie wszystkich danych do wyświetlacza. Prędkość ta będzie więc zależała od taktowania CPU gdyż im mniejsza tym wolniej wykona się algorytm przetwarzania obrazu przez co spadnie ilość klatek przesyłanych do wyświetlacza w jednostce sekundy.

Tabela 5 Pobór energii oraz prędkość działania układu w zależności od taktowania rdzenia mikrokontrolera przy stałym taktowaniu bloku PLL kamery

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SYSCLK [MHz] | Przetwarzanie obrazu | PLL [MHz] | f SPI [MHz] | I [mA] | Prędkość transmisji SPI [kl/s] |
| 180 | NIE | 53 | 90 | 310 | 28 |
| TAK | 310 | 24 |
| 160 | NIE | 80 | 300 | 25 |
| TAK | 300 | 21 |
| 140 | NIE | 70 | 290 | 22 |
| TAK | 290 | 18 |
| 120 | NIE | 60 | 290 | 19 |
| TAK | 290 | 13 |
| 100 | NIE | 50 | 285 | 16 |
| TAK | 285 | 12 |
| 80 | NIE | 40 | 270 | 12 |
| TAK | 270 | 9 |
| 60 | NIE | 30 | 270 | 9 |
| TAK | 270 | 7 |
| 40 | NIE | 20 | 260 | 7 |
| TAK | 260 | 5 |

Jak się okazało podczas pomiarów najniższa możliwa prędkość taktowania CPU wyniosła 140 MHz. Po skonfigurowaniu przypadku dla prędkości 120MHz procesor przestał nadążać z odbieraniem danych z kamery przez co tracił również on synchronizację i obraz na wyświetlaczu LCD przestał być rozpoznawalny. Dzieje się tak ponieważ interfejs DCMI oraz DMA które pracują przy odbiorze danych z kamery taktowane są z prędkością równą prędkości SYSCLK czyli taktowaniu procesora (Tylko dla przypadku gdy dzielnik AHB wynosi 1 tak jak podczas pomiarów). Kamera pracuje więc z częstotliwością 53MHz a magistrala AHB z prędkością 120MHz, jednak w nocie katalogowej można doczytać, że do poprawnej pracy interfejsu DCMI musi zostać zachowany minimalny stosunek AHB/PIXCLK:

Zdecydowano więc powtórzyć pomiary tym razem manipulując jedynie różnymi wartościami taktowania rdzenia mikrokontrolera w wyniku czego zmieniać się będzie zarówno taktowanie SPI jak i taktowanie bloku PLL kamery.

Tabela 6 Pobór energii oraz prędkość działania układu w zależności od taktowania rdzenia mikrokontrolera przy proporcjonalnym taktowaniu bloku PLL kamery

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SYSCLK [MHz] | Przetwarzanie obrazu | PLL [MHz] | f SPI [MHz] | I [mA] | Prędkość transmisji SPI [kl/s] | VSYNC [Hz] |
| 180 | NIE | 53 | 90 | 310 | 28 | 19.88 |
| TAK | 310 | 24 |
| 160 | NIE | 47.1 | 80 | 290 | 25 | 17.67 |
| TAK | 290 | 21 |
| 140 | NIE | 41.25 | 70 | 280 | 22 | 15.46 |
| TAK | 280 | 19 |
| 120 | NIE | 35.3 | 60 | 270 | 19 | 13.25 |
| TAK | 270 | 16 |
| 100 | NIE | 29.5 | 50 | 250 | 16 | 11 |
| TAK | 250 | 13 |
| 80 | NIE | 23.5 | 40 | 240 | 13 | 8.8 |
| TAK | 240 | 11 |
| 60 | NIE | 17.6 | 30 | 220 | 10 | 6.6 |
| TAK | 220 | 8 |
| 40 | NIE | 11.2 | 20 | 210 | 6 | 4.2 |
| TAK | 210 | 5 |

Tym razem przy około 100MHz taktowania CPU obraz na wyświetlaczu również zaczął się dziwnie zachowywać. Tym razem wina leżała po stronie modułu kmaery MT9D111. Producent podaje w dokumentacji minimalne oraz maksymalne taktowania poszczególnych elementów pętli PLL. ( patrz Tabela 2 Taktowania sygnałów MT9D111)

Okazało się, że przy częstotliwości CPU wynoszącej 100MHz parametry zbytnio zbliżyły się do wartości minimalnych co spowodowało błędną pracę modułu. Częstotliwość wyjściowa PIXCLK w tym przypadku powinna wynosić około 29.5MHz. Przekonfigurowano więc parametry M,N, P oraz dzielnik częstotliwości CPU tak by otrzymać tą samą częstotliwość wyjściową oddalając tym samym częstotliwości sygnałów od wartości minimalnych operacja ta przywróciła prawidłowe funkcjonowanie modułu kamery.

# Podsumowanie i wnioski

Wydajność układu jaką udalo się uzyskać przy prostym przetwarzaniu obrazu pozwoliła stwierdzić, że układ ten jest wystarczająco wydajny do dalszego rozwoju oprogramowania o bardziej skomplikowane przetwarzanie obrazu. Kamera jest w stanie generować obraz z prędkością 20kl/s przy dostępnym mikroprocesorze, jest to wartość wystarczająca by obraz oglądany nie męczył oka, był płynny oraz wyraźny. Algorytm zaimplementowany wyszukuje na obrazie punkt o największej jasności, taki podstawowy algorytm mógłby być użyty w prostym robocie który mógłby znajdywać biały punkt a następnie go śledzić. Podczas pracy nad projektem zostały poruszone podstawowe zagadnienia z programowania mikrokontrolerów, zasady działania peryferiów w tym przez wszystkim modułów komunikacyjncyh. Podczas pracy pojawiło się wiele problemów przy poprawnej obsłudze interfejsów co zmusiło do sięgnięcia po analizator stanów logicznych i przeanalizowania przebiegów na poszczególnych liniach interfejsów komunikacyjnych czy aby dane poprawnie są przesyłane. Wymagało to precyzyjnej znajomości ramek danych jakimi operują interfejsy. Dodatkowo autor dokładnie poznał architekturę procesora Cortex M4 oraz jego zasadę działania, peryferia jakimi dysponuję oraz obsługę na poziomie programowym takich elementów jak przerwania oraz konfigurację peryferiów. Firma STM dostarcza gotowe biblioteki do obsługi interfejsów między innymi do obsługi SPI, I2C, DCMI lub przerwań. Praca na niektórych bibliotekach przygotowanych przez producenta wymagała od autora umiejętności analizy kodu, poznania funkcji oraz kodu napisanego przez inną osobę. Jednak podczas pracy okazało się, że biblioteka do obsługi SPI okazała się słabo zoptymalizowana pod kątem szybkości działania interfejsu przez co autor zdecydował się korzystać z biblioteki tylko podczas inicjalizacji, do transmisji słuzy funkcja przygotowana przez autora. Podobnie biblioteki do obsługi przerwań okazały się na tyle rozbudowane że spowalniały one pracę urządzenia na tyle, że zrezygnowano z ich uzycia a w ich miejscu pracują funkcję znacznie lżejsze obudowane tylko w podstawową funkcjonalność. Doświadczenia te pozwoliły autorowi na udoskonalenie swoich umiejętności programistyznych a konieczność dokładnej analizy czyjegoś kodu oraz obsługi w nim peryferiów wymagała dokładnego czytania dokumentacji procesorów.

# Streszczenie

Projekt ten zakłada przetestowanie mikroprocesora STM32F429ZIT zbudowanego w oparciu o rdzeń ARM Cortex M4 32bitowy taktowany prędkością 180MHz. Mikrokontroler należy do serii wysokiej wydajności. Dzięki wysokiemu taktowaniu przepustowość interfejsów komunikacyjnych powinna być na tyle wysoka by nie wpływały one negatywnie na całą procedurę obsługi obrazu w tym jego przesył między modulami. Praca ma więc w celu przetestowanie wydajności rdzenia w przetwarzaniu obrazu jak i również wydajności całego systemu zbudowanego w projekcie w oparciu o wspomniany mikrokontroler. W tym celu konieczne było napisanie programu oraz opracowanie własnych scenariuszy testowych które miały by pokazać faktyczną wydajność systemu od momentu zarejestrowania obrazu na matrycy CMOS, przesłania obrazu do mikroprocesora, przetworzeniu obrazu aby na końcu móc wyświetlić go na wyświetlaczu LCD. Dodatkowo w celu lepszego zrozumienia problematyki w pracy zawarte zostały krótkie wprowadzenai do używanych interfejsów, ich zasada działania oraz wygląd ramek danych wysyłanych interfejsami. Od uzyskanych wyników zależeć będzie dalszy rozwój oprogramowania w tym o rozbudowę przetwarzania obrazu i wykorzystaniu systemu w bardziej skomplikowanych algorytmach przetwaraznia obrazu.

# Summary

The main purpose of this project is to test STM32F429ZIT microcontroller based on ARM Cortex M4 32bit core with 180MHz clock. This microcontroller belongs to high-performance series. Because of high core frequency, throughput of communication interfaces should be enough to carry all data without any speed lost due to all image processing steps including forwarding image frame between peripheral and memory. Purpose of this thesis is to perform few test which will show Cortex M4 Core performance and also all embedded system performance built on this project. To had it done author had to write program and devise own tests scenario which will show system real performance from moment when CMOS sensor capture image, forward it to microcontroller memory, process image and finally display processed image on LCD. Additional part of this thesis is to describe how works all fundamental peripheral used in project, how data is send through them and how data frame looks. Future of this embedded system depends on performance result from test scenario. If Performance will be satisfying, in the future system will be develop with more complex image processing algorithm.

# Słowa kluczowe

* Peryferia
* Interfejs
* Synchronizacja
* Ramka obrazu
* Przetwarzanie obrazu
* Matryca CMOS
* Pętla PLL

# Keywords

* Peripheral
* Interface
* Synchronization
* Image frame
* Image processing
* CMOS Image Sensor
* Phase-locked loop oscilator (PLL)

# Bibliografia

1. ST, *RM0090 Reference manual*, Lipiec 2017
2. ST, *STM32F429xx Data Sheet*, Styczeń 2016
3. ST, *STM32F429I-DISCO schematics*, Wersja 1.0
4. ST, *Digital Camera Interface (DCMI) for STM32 MCUs*, <http://www.st.com/content/ccc/resource/technical/document/application_note/group0/c0/ef/15/38/d1/d6/49/88/DM00373474/files/DM00373474.pdf/jcr:content/translations/en.DM00373474.pdf>
5. STM32.eu, *Obsluga interfejsu DCMI,* [*https://stm32.eu/2013/07/29/obsluga-interfejsu-dcmi-kamery-ccd-przyklad-na-stm32f2/*](https://stm32.eu/2013/07/29/obsluga-interfejsu-dcmi-kamery-ccd-przyklad-na-stm32f2/)
6. ST, Dokumentacja, <http://www.st.com/en/evaluation-tools/32f429idiscovery.html>
7. Bosh Security, *Inteligentna analiza wizyjna,* <http://resource.boschsecurity.com/documents/Operation_Guide_enUS_7603462027.pdf>
8. Wikipedia, *Microsoft Kinect*, <https://pl.wikipedia.org/wiki/Kinect>
9. Microsoft, *Microsoft Kinect*, <https://developer.microsoft.com/pl-pl/windows/kinect>
10. Wikipedia, *Magistrala I2C,* [*https://pl.wikipedia.org/wiki/I%C2%B2C*](https://pl.wikipedia.org/wiki/I%C2%B2C)
11. Forbot, *Obsługa SPI,* <https://forbot.pl/blog/kurs-stm32-f4-10-obsluga-spi-wyswietlacz-oled-id13475>
12. Micron, *1/3.2-Inch System-On\_A-Chip (SOC) CMOS Digital Image Sensor MT9D111,* [*http://www.uctronics.com/download/cam\_module/MT9D111\_SOC2010DS.pdf*](http://www.uctronics.com/download/cam_module/MT9D111_SOC2010DS.pdf)

# Dodatek A. Spis zawartości dołączonej płyty CD

mateusz\_stechura\_280487.doc – praca w formacie MS office

mateusz\_stechura\_280487.pdf – praca apisana w formacie PDF

./Program – kompletny kod źródłowy programu zapisany jako projekt programu uVision Keil.

./dokumenty – kompletne dokumenty wykorzystywane w projekcie

# Spis ilustracji

[Rysunek 1: Rozkład pikseli na matrycy 9](#_Toc502076244)

[Rysunek 2: Budowa czujnika matrycy CMOS 10](#_Toc502076245)

[Rysunek 3 Struktura Ramki I2C 13](#_Toc502076246)

[Rysunek 4 Zapis poprzez I2C wartości 0xA000 do rejestru 0x65 14](#_Toc502076247)

[Rysunek 5 Proces odczytu poprzez I2C z adresu 0x00 15](#_Toc502076248)

[Rysunek 6 Podłączenie urządzeń przy użyciu interfejsu SPI 16](#_Toc502076249)

[Rysunek 7 Przebiegi czasowe komunikacji SPI 17](#_Toc502076250)

[Rysunek 8 Ramka DCMI 18](#_Toc502076251)

[Rysunek 9 STM32F429-DISCOVERY 19](file:///C:\Users\Mateusz\Desktop\AGH\4%20rok\Inżynierka\praca_inzynierka_V2.docx#_Toc502076252)

[Rysunek 10 Moduł MT9D111 19](file:///C:\Users\Mateusz\Desktop\AGH\4%20rok\Inżynierka\praca_inzynierka_V2.docx#_Toc502076253)

[Rysunek 11 Odczyt ramki obrazu 21](#_Toc502076254)

[Rysunek 12 Schemat układu 23](#_Toc502076255)

[Rysunek 13 Prędkość transmisji dla roździelczości 1600x1200 oraz taktowania PLL 80MHz 25](#_Toc502076256)

[Rysunek 14 Prędkość transmisji dla roździelczości 800x600 oraz taktowania PLL 80MHz 25](#_Toc502076257)

[Rysunek 15 Podział ramki obrazu na obszary 32](#_Toc502076258)

[Rysunek 16 Konfiguracja zegarów STM32F429ZIT 35](#_Toc502076259)

# Spis tabel

[Tabela 1: Rozmieszczenie bitów w formacie RGB-565 10](#_Toc502076260)

[Tabela 2 Taktowania syngałów MT9D111 20](#_Toc502076261)

[Tabela 3 Wpływ taktowania SPI na wydajność 27](#_Toc502076262)

[Tabela 4 Pomiary wpływu pętli PLL na układ 29](#_Toc502076263)

[Tabela 5 Pobór energii oraz prędkość działania układu w zależności od taktowania rdzenia mikrokontrolera przy stałym taktowaniu bloku PLL kamery 36](#_Toc502076264)

[Tabela 6 Pobór energii oraz prędkość działania układu w zależności od taktowania rdzenia mikrokontrolera przy proporcjonalnym taktowaniu bloku PLL kamery 37](#_Toc502076265)

# Spis stosowanych skrótów

LCD – Liquid Crystal Display

SPI – Serial Peripherial Interface

I2C – Inter-Integrated Circuit

DCMI – Ditital Camera Interface

DMA – Direct Memory Access

CMOS – Complementary Metal-Oxice-Semiconductor

MSB – Most Significant Bit

LSB – Less Significant Bit

SDA – Serial Data Line

SCL – Serial Clock Line