**Szybka, Inteligentna kamera z interfejsem Ethernet**

W poniższej pracy zaprezentowany jest projekt szybkiej, inteligentnej kamery opartej na systemie wbudowanym Raspberry Pi 2 z układem SoC BCM2836 firmy Broadcom oraz modułem kamery RaspiCam - sensor ov5647.

Dzięki wydajnym podzespołom płyty Raspberry Pi 2 możliwe jest szybkie przetworzenie obrazu w celu detekcji wymaganego wzorca i wykonanie określonej akcji.  
Moduł Ethernet umożliwia komunikację między użytkownikiem, a płytą za pomocą sieci internetowej.

Płyta pracuje pod kontrolą systemu Linux, dystrybucji Raspbian , na którym uruchamiana jest aplikacja napisana w języku C++.

W pracy przedstawiona jest jedna z wielu koncepcji wykorzystania inteligentnej kamery oraz jej praktyczna realizacja.

Słowa kluczowe: **systemy wbudowane, wizja komputerowa, Ethernet,** Internet of Things

Rozdział 1

Wstęp

Rozwój elektroniki, a szczególnie systemów mikroprocesorowych w ostatnich latach ma niebagatelny wpływ na niemal każdy aspekt naszego życia. Systemy te wraz z ewolucją są wykorzystywane już nie tylko do specjalnych zastosowań w wąskich dziedzinach, ale także jako elementy ułatwiające nam codzienne życie. Do jednych z wielu dziedzin, w których są kluczowe można zaliczyć min. : robotykę, motoryzację, medycynę, sport, tzw. inteligentne systemy.   
Pojęcie inteligentnych systemów w ostatnim czasie jest bardzo powszechne. Systemy te wyróżniają się tym, że zamiast wykonywania z góry założonego schematu działania są wyposażone w dodatkową inteligencję. Inteligencja ta daje dużo większe możliwości wykorzystania urządzenia, gdyż odpowiednio zastosowana daje możliwość odciążenia użytkownika z części zadań, lub nawet wykonania ich dużo sprawniej i dokładniej.

Jako, że wraz z rozwojem technologii możliwości układów SoC wzrastają, przy niewielkich wciąż wymiarach, pozwala to zawrzeć w chipie module wiele podzespołów, takich jak dedykowane układy graficzne, moduły komunikacji sieciowej, oraz inne podsystemy rozszerzające funkcjonalność układu.  
Ponadto systemy wbudowane odgrywają znaczącą rolę szczególnie tam, gdzie niewielkie gabaryty urządzenia i niski pobór mocy są kluczowe.

Przykładem takiego systemu może być inteligenta kamera, której zadaniem jest detekcja danego obiektu i podjęcie określonej przez projektanta akcji np. sterowania innym urządzeniem lub podzespołem, czy też tworzeniem statystyk.

* Internet of Things

Temat niniejszej pracy świetnie wpasowuje się w koncepcję tzw. Internetu Rzeczy (ang. Internet of Things- skr. IoT). Według której wszelkiego rodzaju urządzenia mogą stanowić jedną wielką sieć, pozwalającą sprawniej ze sobą wydajnie współpracować komunikując się ze sobą, wspólnie przetwarzając i wymieniając dane za pomocą sieci komputerowej.

Pod hasłem systemy inteligentne jak widać kryje się wiele różnych zagadnień, w których sprawne poruszanie się i umiejętność zintegrowania ich w jeden system jest dla projektanta kluczowe.

W niniejszej pracy zawarty jest opis projektu szybkiej inteligentnej kamery z interfejsem Ethernet, która wykrywa ludzi zbliżających się do niej, po czym otwiera drzwi i umieszcza wpis na serwerze http wraz z datą wykrycia i zdjęciem danej osoby.

1.1 Przegląd dostępnych komercyjnych rozwiązań

Prace techniczne i koncepcyjne poprzedziło badanie rynku. Za pomocą porównywarek cenowych zacząłem oceniać i porównywać oferty systemów monitoringu z różnych zakresów cenowych i jakościowych. Przyznaję, że w najniższej półce cenowej odnalazłem oferty o akceptowalnych parametrach. Jako przykład zostanie podany system monitorowania bezprzewodowego Conrad 8103J, kameraIR, odbiornik 4-kanałowy. Producent zachwala rozwiązanie jako idealne do indywidualnego monitoringu wideo. W opisie produktu odnajdujemy informację, że komunikacja odbywa się po drodze radiowej, istnieje możliwość oglądania obrazu na ekranie monitora, oraz automatyczne diody podczerwieni umożliwiają transmisję czarno-białego obrazu w nocy. Koszt zestawu wynosi 275 PLN.

|  |  |
| --- | --- |
| Dodatkowe światło IR | Tak |
| Komunikacja | Analogowa, Radiowa |
| Mikrofon | Tak |
| Napięcie robocze | 8 V/DC z zasilacza |
| Rodzaj kamery CCTV | Bezprzewodowy system monitorujący |
| Rozdzielczość (TVL) | 720 x 480 p |
| Wymiary (odbiornik) | 24 x 78 x 107 mm |
| Wymiary nadajnika | Ø 26 x 35 mm |
| Zakres temperatury roboczej | od -10 do +50 °C |
| Zasięg maksymalny | 100 m |

Dane techniczne i specyfikacja

Z drugiej strony porównano rozbudowany system monitoringu IP: „Rejestrator BCS-NVR0802, 8 x Kamera BCS-TIP7300IR, Dysk 500GB, Akcesoria” firmy ivel electronics. Koszt zestawu 25 849,00 zł . W specyfikacji urządzania można odnaleźć bardzo wydajny sprzęt:

Rejestrator BCS-NVR0802

8 kanałowy sieciowy rejestrator cyfrowy z kompresją obrazu H.264 , standard PAL

Nagrywanie do 8 kamer IP w D1 (25 kl/s), 720P (25 kl/s), 1080P (12 kl/s)

Rysunek 1.1 Rejestrator BCS-NVT0802

Prędkość zapisu rejestratora wynosi 200 klatek na sekundę (100 klatek w 1080P)  
Wydajny procesor Dual-Core z systemem Embedded Linux

8x Video, 8 x Audio, VGA, BNC, USB, HDMI, PTZ,  
 RS485, Wyjścia/Wejścia alarmowe

W rejestratorze można zamontować 2 dyski HDD SATA.  
Możliwy podgląd przez internet na komputerze, smartfonie i tablecie.

Kamera BCS-TIP7300IR

Rysunek 1.2 Kamera sieciowa BCS-TIP7300IR

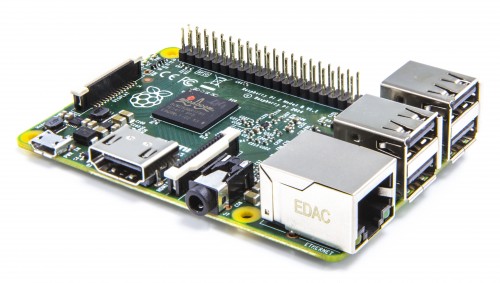
Kamera sieciowa IP 3MPix FULL HD  
20 kl/s @3MPix (2048 x 1536), 25 kl/s @1080p  
Przetwornik 1/2.8" SONY Progressive Scan CMOS  
Obiektyw 8 - 16 mm/F1.6 CS Auto Iris  
PoE, mechaniczny filtr podczerwieni  
Zasięg podczerwieni 30 metrów  
Hermetyczna  obudowa  klasy IP66

Po przeanalizowaniu dostępnych rozwiązań zaskakujący wniosek nasuwa się sam. Zauważamy sprzęt o bardzo dobrych parametrach, dostępny cenowo, ale niepraktyczny dla zwykłego użytkownika. Monitorując w dużej ilości zastosowań nie potrzebujemy nagrywać pełnometrażowego filmu z dużą dokładnością, który monitoruje obszar, w którym tak naprawdę niewiele się zmienia. Znacząca część zastosowań nie wymaga archiwizowania dużej ilości danych. Podgląd ze smartfona jest cennym dodatkiem aczkolwiek chcemy aby system sam informował o interesujących nas zdarzeniach i nie absorbował użytkownika w celu monitorowania obszaru. Konkluzją analizy dostępnych komercyjnych rozwiązań jest koncepcja zaprojektowania „Szybkiej, Inteligentnej kamery z interfejsem Ethernet”. Spośród istniejących rozwiązań kamera ma wyróżniać się właśnie na polu zastosowania. Tani wydajny sprzęt w połączeniu z algorytmami rozpoznawania obrazu możliwi opracowanie kamery bezkonkurencyjnej pod względem użyteczności, elastyczności zastosowania oraz pełnego wykorzystania potencjału sprzętowego.

1.2 Przegląd dostępnych systemów wbudowanych

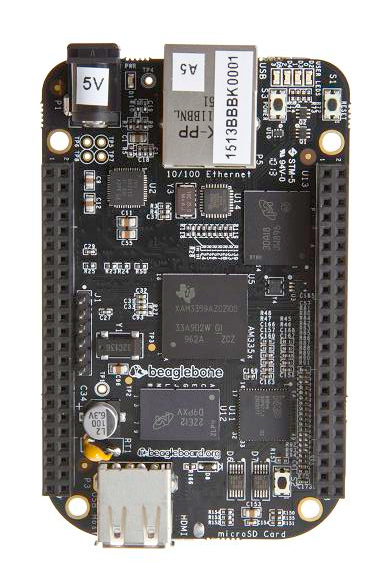
Poniżej przedstawiono kilka z wielu dostępnych na rynku urządzeń, które mogą być zastosowane np. do rozwiązań typu „smart”.

**Raspberry Pi 2 firmy *Raspberry Pi Foundation*** jest minikomputerem wyposażonym w wydajną, 4-rdzeniową jednostkę SoC **BCM 2836 firmy *Broadcom***. Posiada min. złącze HDMI, 4 złącza USB i jedno Ethernet’owe. Ponadto kontroler kart SD, pozwala uruchomić system operacyjny z karty microSD. Płyta posiada także złącza dla dedykowanej kamery RaspiCam (sensor ov5xx) oraz złącze Display port, dla wyświetlacza. Cena : 175 zł



Rysunek 1.1. Raspberry Pi w wersji 2

**BeagleBone Black Rev. C firmy *BeagleBoard***wyposażony w wydajny procesor **Sitara AM335x firmy *Texas Instruments*** taktowany zegarem 1GHz, 512 MB pamięci RAM i akcelerator grafiki 3D, a także złącza USB, HDMI i 92 porty GPIO. Cena: 249zł.



Rysunek 1.2. BeagleBone Black Rev. C

**OlinuXino Micro A13 firmy *Olimex*** z jednostką SoC Allwinner A13 i rdzeniem A13 Cortex A8 taktowanym zegarem 1GHz oraz jedostką GPU 3D Mali400 i 256 MB RAM stanowi tańszą alternatywę dla ww. urządzeń. Płyta posiada złącza USB, kontroler kart SD oraz wyjście VGA. Ponadto ma też osobne złącze dla wyświetlacza LCD i 40 portów GPIO. Cena: 35 EUR.



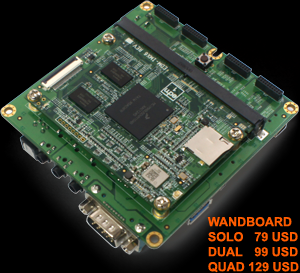
Rysunek 1.3. OlinuXino Micro A13

Poniżej zestawiono w tabeli poszczególne urządzenia dla porównania parametrów i ceny.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Urządzenie | Procesor | Pamięć RAM | GPU | Interfejs kamery  (CSI) | Ethernet | Cena |
| Raspberry Pi 2 | BCM 2386 900MHz  quad-core ARM Cortex-A7 CPU | 1GB DDR2 SDRAM | VideoCore IV | Tak | 10/100M | 175zł |
| Beagle Bone Black Rev. C | [AM335x 1GHz ARM Cortex-A8](http://www.ti.com/product/am3358) | 512MB DDR3 RAM | PowerVR SGX530 | Nie | 10/100M | 249zł |
| OlinuXino Micro A13 | Allwinner A13 ARM Cortex A8 @1GHz | 256MB DDR2 RAM | 3D Mali400 | Tak | Nie (dostępny moduł USB-Ethernet) | ok. 140zł |

Tabela 1.1 : Zestawienie popularnych systemów wbudowanych

Rysunek 1.4 Płyta Wandboard Solo



Do zestawienia dodano rozwiązanie typowo z zastosowań przemysłowych: Wandboard Solo (model jednordzeniowy).

Processor: Freescale i.MX6 Solo

Cores: Cortex-A9 Single core

Graphic engine: Vivante GC 880 + Vivante GC 320

Memory: 512 MB DDR3

Warto wspomnieć o tym rozwiązaniu z powodu jego rosnącej popularności. Mały komputer jednopłytkowy wyróżnia solidne wykonanie, niezawodność oraz możliwość własnej adaptacji nakładek edm1 wykorzystującej standardowe złącze EDM.

Do płytek nakładkowych typu edm (firmy Wandboard) lub rozwiązań SOM firmy olimex można opracować płytkę bazową, która będzie zinegrowana z kamerką i peryferiami komunikacji. Umożliwi to zmniejszenie rozmiaru, wykorzystanie tylko tych elementów, które są niezbędne. Zmniejszy to koszt projektu. Będziemy mieli również gwarancję, że jak zmieni się procesor będzie można wymienić płytkę nakładkową z procesorem (bez konieczności zmiany projektowej płytki bazowej).

Przyszłościowe koncepcyjne myślenie o rozwoju produktu skłoniło do wyboru dwóch rozwiązań, mniej popularnego tańszego układu firmy olimex i popularnych układów raspbery pi (z proc. BCM 2386). Oba wybrane zestawy demonstracyjne posiadają swoje odpowiedniki w płytach bazowych do rozwoju urządzeń tzw SOM (System on Module).

1.3. Architektura systemu Linux

Obecnie Linux w systemach embedded jest jednym z najpopularniejszych systemów operacyjnych. Dzięki uniwersalności i elastyczności można go spotkać w wielu urządzeniach dostępnych na rynku, takich jak : smartfony, tablety, odtwarzacze etc. Ważnym zagadnieniem w pracy z systemami opartymi na jądrze Linux jest znajomość jego architektury, a także połączenie wiedzy z jego działania, administracji, konfiguracji oraz programowania w środowisku systemowym. W podrozdziale tym scharakteryzowane zostały podstawowe elementy każdego systemu opartego na jądrze Linux.

Główny system plików

Aplikacje

Biblioteki

Jądro

Mechanizmy i pozostałe moduły

Sterowniki i moduły zależne od architektury

Sprzęt

Rysunek 1.1 Architektura systemu Linux

* Jądro systemu – centralny element systemu operacyjnego, uprzywilejowany egzekutor zarządzający wszystkimi zasobami systemu.

-Moduły jądra – działają w uprzywilejowanym trybie; pozwalają na rozszerzanie funkcjonalności jądra bez ingerencji w kod źródłowy systemu.

-Sterowniki – pośredniczą w komunikacji jądra ze sprzętem, oraz urządzeń wirtualnych (abstrakcyjnych)

* Główny system plików – drzewo katalogów i plików

-Biblioteki – używane przez niemal każdą aplikację; umożliwiają rozszerzenie funkcjonalności aplikacji działającej pod kontrolą systemu oraz stanowią interfejs do usług systemowych jądra.

-Aplikacje – przenośne na poziomie kodu. Do podstawowych należą : powłoka, programy do operacji na plikach i katalogach, procesach, mechanizmach sieciowych, a także środowiska graficzne, kodery audio/wideo oraz wiele innych standardowych aplikacji.

1.4. Dystrybucje systemu Linux

Jako dystrybucję systemu Linux można określić kompletny system operacyjny, w skład którego oprócz samego jądra wchodzą dodatkowe usługi i aplikacje. To co odróżnia poszczególne dystrybucje, to jednolita organizacja plików konfiguracyjnych i mechanizm instalacji i zarządzania pakietami.

Istnieje kilka kryteriów klasyfikacji dystrybucji:

* Komercyjne lub niekomercyjne
* Przeznaczone dla określonej grupy odbiorców : użytkowników zwykłych lub biznesowych
* Wieloplatformowe, lub optymalizowane pod wybraną platformę sprzętową
* Wyspecjalizowane lub ogólnego przeznaczenia
* O określonym priorytecie np. przenośności czy bezpieczeństwa

Wśród systemów wbudowanych opartych na platformie Raspberry można wyróżnić następujące dystrybucje:

-Raspbian – najpopularniejsza dystrubucja, oparta na Debianie

-ArchLinux – pozwala w pełni dostosować system do wymagań użytkownika; pozbawiona GUI

-OpenELEC – dystrybucja typu Media Center

W projekcie wykorzystana zostanie dystrybucja Raspbian .

1.3 Konkurencyjne rozwiązania



Konkurencyjnym rozwiązaniem wchodzącym dynamicznie na rynek jest windows embedded. Z każdą wersją obecnego systemu Windows udostępnia okrojoną wersję embedded. Rozwiązanie ma swoje zalety np. support techniczny, kompatybilność z aktualnymi desktopowymi systemami windows. Jako ciekawostkę można podać, że wymagania desktopowego systemu windows xp spełnia większość systemów wbudowanych, jak czyamy w wymaganiach systemu [odnośnik: <https://support.microsoft.com/pl-pl/kb/314865>]. Za wyborem linuxowego rozwiązania decydowało otwarte oprogramowanie, możliwość edycji ustawień systemowych z pozmiou kompilacji systemu. Pomijając indywidualne preferencje autora nie należy również zapominać, że systemy linux charakteryzują się bezpieczeństwem i jest to sprawdzone rozwiązanie dla systemów wbudowanych.

1.5 Podsumowanie

Na podstawie zestawienia popularnych systemów wbudowanych można stwierdzić, że każdy z nich oferuje różne konfiguracje i peryferia, co zmusza projektanta do optymalnego wyboru platformy sprzętowej pod kątem projektu. Analiza dostępnych urządzeń i dobranie optymalnego i niezawodnego rozwiązania stanowi ważny aspekt niniejszej pracy.  
Najlepszy stosunek cena/jakość ma Raspberry Pi 2. Czterordzeniowy procesor i 1GB pamięci RAM daje dużo większe możliwości względem konkurencyjnych produktów, co jest kluczowe przy wymagających aplikacjach wykorzystujących przetwarzanie obrazu.

Jako dystrybucję systemu uruchamianą na płycie wybrano Raspbian.

Rozdział 2

Cel i założenia pracy

Praca ma na celu zbadania możliwości wykonania projektu Szybkiej, Inteligentnej kamery z interfejsem Ethernet na systemie wbudowanym (ang. embedded system).  
Kamerę stosować można będzie np. do monitorowania przestrzeni przemysłowej, a także w komunikacji miejskiej, czy też w zastosowaniach domowych. Docelowo użytkownikiem mają być osoby indywidualne, aczkolwiek system jest elastyczny, wobec czego łatwo można go zmodyfikować także pod kątem bardziej specjalistycznych zastosowań.  
Projekt stanowi podstawę do realizacji dużo większego systemu – sieci kamer połączonych ze sobą, np. pod kontrolą serwera i umożliwiających realizację bardziej złożonych zadań, jak np. śledzenie trasy danej osoby, czy też kontrolowanie jednocześnie kilku obszarów w hali produkcyjnej.

Założenia projektu :

* Maksymalizacja prędkości działania
* Inteligencja dostosowana do wymogów monitoringu – zaimplementowana logika pozwala kamerze rozpoznać określone obiekty oraz wykonywać samodzielnie akcje, sterować układem zewnętrznym
* Modularność – struktura oprogramowania ma być łatwa do rozbudowy i debugowania
* Komunikacja z użytkownikiem

Do komunikacji z użytkownikiem wykorzystany zostanie serwer http, na którym będą umieszczane przez aplikację logi (zdjęcie osoby i data wykonania akcji). Natomiast sterowanie innymi urządzeniami zrealizowane jest poprzez wystawienie stanów HIGH/LOW na wyjściach portów GPIO.

Na wykonanie projektu składa się:

* Praca koncepcyjna
* Konfiguracja systemu
* Napisanie aplikacji uruchamianej na płycie
* Skonfigurowanie serwera http i przygotowanie skryptu PHP/jQuery
* Wykonanie testów działania projektu
* Wykonanie obudowy urządzenia

Rozdział 3

Koncepcja projektu

W rozdziale omówiona zostanie koncepcja i zasada działania inteligentnej kamery.

3.1. Maksymalizacja szybkości

Szybkość działania kamery jest jednym z kluczowych założeń działania systemu. W zależności od przeznaczenia systemu może się znacznie różnić. W dziedzinach wyspecjalizowanych, takich jak robotyka, czy

Czynniki składające się na ogólną szybkość działania kamery:

1.Przetwarzanie obrazu przez sensor oraz protokół komunikacyjny między sensorem, a płytą.

Według danych producenta czas potrzebny na wygenerowanie jednej klatki wynosi xx s. Protokół zastosowany w kamerze, to (USB/równoległy ?), cechujący się czasem przesyłu na poziomie xx s.

2.Przetwarzanie obrazu przez sterownik obrazu (kompresja danych)

Czas ten jest uwarunkowany wydajnością akceleratora graficznego i ze względu na wydajną jednostkę GPU VideoCore IV uznawany jest w projekcie za pomijalny.

3.Przetwarzanie danych przez aplikację.

Głównym czynnikiem mającym wpływ na ten czas jest algorytm zastosowany do detekcji twarzy. Dokładny jego opis znajduje się w rozdziale 3.6.1

4.Szybkość przesyłu danych przez interfejs Ethernet

Protokół Ethernet jest jednym z podstawowych interfejsów komunikacji urządzeń w lokalnych sieciach komputerowych. Zastosowana w płycie Raspberry Pi 2 wbudowana karta sieciowa wg informacji producenta umożliwia transfer danych z przepustowością 10 lub 100 Mb/s.

Ważnym aspektem są czas odpowiedzi systemu dla każdej z ww. operacji oraz ogólne czasy odpowiednio między wykryciem twarzy a :

-akcją sprzętową (np. otwarciem drzwi, włączeniem alarmu)

-umieszczeniem logu na serwerze

W projekcie przyjęto założenie, że odstęp czasowy między zdarzeniami określony będzie przez użytkownika. Rozdziały 4. i 5. aspekt ten opisują dokładniej.

3.2 Inteligencja dostosowana do wymogów monitoringu

Inteligentny system monitorujący wymaga, aby oprócz dostarczania obrazu dokonać także jego przetworzenia – tj. za pomocą odpowiednich algorytmów dokonać detekcji żądanych obiektów.

Podrozdział 3.6.1 traktuje szczegółowo o tym zagadnieniu.

3.3 Modularność

Struktura modularna pozwala zapewnić dużą kontrolę przy projektowaniu i debugowaniu działania aplikacji. Aplikacja jest pisana zgodnie z techniką programowania obiektowo orientowanego (w skrócie OO), wyróżnić można kolejne kroki :

1.Rozpoznanie problemu

Przeprowadzona została analiza funkcjonalności systemu i wymagań, zgodnie z założeniami z rozdziału 3.1.

2.Projektowanie programu

Na ten etap składa się : zidentyfikowanie zachowań systemu i obiektów w nim występujących, a następnie określenie ich hierarchii ( dziedziczenie ) i sekwencji działania.

Obiekty występujące w systemie to :

**Module** – bazowa klasa abstrakcyjna

**Object Detect** – klasa implementująca detekcję obiektów

**Worker** – klasa odpowiedzialna za wszelkiego rodzaju akcje

**Logger** – klasa implementująca mechanizm generowania logów. Współpracuje bezpośrednio z **Object Detect**

**Controller** – główna klasa zarządzająca pozostałymi, implementuje „logikę” kamery.

Do konkretnych zachowań można zaliczyć : detekcję obiektów, wykonanie akcji (sterowanie portami GPIO), stworzenie pliku logów. Każda z wymienionych klas dziedziczy po **Module**.

Sekwencja działania jest zdefiniowana następująco :

1. **Object Detect** po wykryciu twarzy, wysyła sygnał do **Controllera**
2. **Controller** ustawia flagę *detected=true*, odczekuje ustalony czas *timeout = 2s*
3. Jeśli po upływie timeout’u **Controller** otrzyma ponownie sygnał, to zleca modułowi **Worker** i **Logger** pracę – odpowiednio otwarcie drzwi i wygenerowanie pliku z logiem.
4. Po wykonaniu pracy **Worker** wysyła do **Controllera** sygnał
5. Po odebraniu sygnału od **Worker’a** **Controller** zatrzymuje pracę modułu **Object Detect**
6. Po wykonaniu akcji **Worker** wysyła sygnał do **Controllera**
7. Odebranie sygnału od **Workera** powoduje wznowienie pracy **Object Detect**

3.Implementacja

Implementacja komunikacji między obiektami zrealizowana jest z wykorzystaniem mechanizmu sygnałów biblioteki Boost. Daje to możliwość przekazania sterowania danemu obiektowi poprzez wywołanie funkcji obiektu zlecanego - zbindowanej w slocie wraz z przekazanym argumentem określonego typu.

4.Testowanie

Ważnym aspektem weryfikacji poprawnej pracy systemu są testy. Dzięki technice OO stworzenie testów wymaga dużo mniejszego nakładu pracy niż pisanie specjalnych funkcji testujących.

Argumentem dla wyboru techniki OO jest fakt, iż system w przyszłości może realizować inne zadania/warunki, w konsekwencji aplikacja będzie potencjalnie modyfikowana i sukcesywnie rozbudowywana.

3.4 Interfejs użytkownika

Jako interfejs użytkownika służy serwer http – Apache 2. Na podstawie dostarczonych od modułu Logger zdjęć generuje odpowiednie wpisy i umieszcza je na stronie WWW. Za generację logów po stronie serwera odpowiedzialne są skrypty napisane z wykorzystaniem technologii PHP i jQuery.

Dane takie jak data, godzina są zawarte w nazwie plików, natomiast zostaną wyodrębnione przez ww. skrypty. Strona WWW odświeżana jest automatycznie co 1s. Szczegółowy opis działania skryptów opisany jest w podrozdziale 3.6.3.

3.5 Funkcje systemu

Funkcje systemu :

* Detekcja obiektów

Obiekty te mogą być w zasadzie dowolnego typu. Detekcja opiera się na wytrenowaniu algorytmu dlatego mogą to być zarówno obiekty rzeczywiste, takie jak: twarze ludzkie, litery, pojazdy itp., oraz specyficzne wzory pozwalające identyfikować jednoznacznie żądane obiekty.

* Sterowanie modułem zewnętrznym

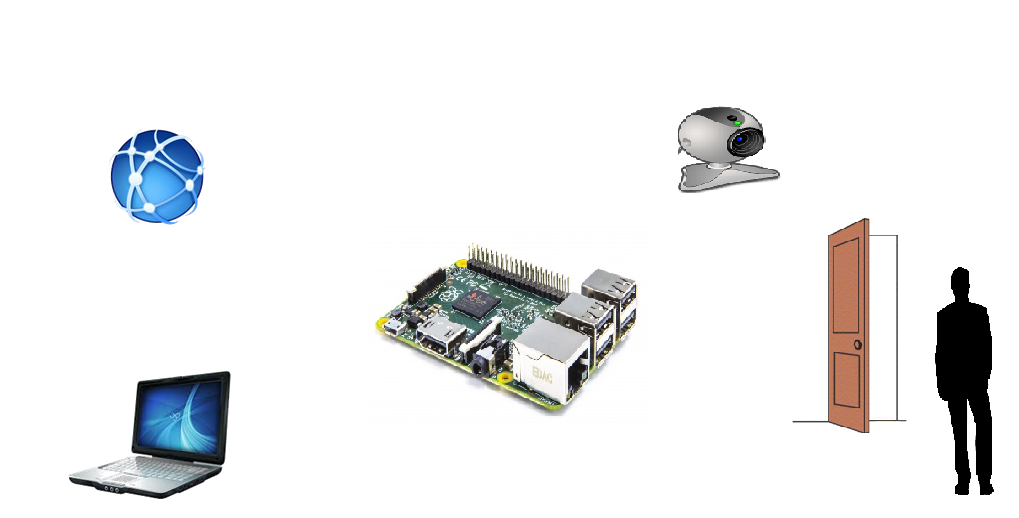
W zależności od wymagań mogą to być: manipulatory robotów, sterowniki bram, napędy, czujniki, a także inne aplikacje, które oczekują na informacje od kamery ( np. ilość obiektów i ich czasy poruszania się, czy też orientacja w przestrzeni).

* Logowanie zdarzeń na serwerze http

Dzięki interfejsowi Ethernet możliwa jest komunikacja systemu z użytkownikiem. Np. umieszczanie na serwerze logów, lub danych przetworzonych przez urządzenie w bazie danych.

Dodatkową funkcjonalnością przewidywaną przy dalszej rozbudowie systemu jest graficzny interfejs użytkownika GUI (ang. Graphic User Interface), pozwalający na szybszą i bardziej intuicyjną interakcję użytkownika z systemem.

W projekcie kamera łączy role systemu monitoringu i sterownika drzwi: ma za zadanie wykryć twarze osób, które pojawią się przed drzwiami, a następnie je otworzyć. Kolejnym krokiem jest umieszczenie na serwerze logu w postaci : zdjęcia danej osoby oraz daty i godziny pojawienia się. Dostęp do logów będzie możliwy dla każdego użytkownika posiadającego urządzenie posiadające przeglądarkę stron WWW ( czyli smartfony, tablety, komputery itd.) i będące w obrębie sieci, w której pracuje kamera.  
Ogólna idea zastosowania kamery jest zilustrowana na rysunku 3.1.



Rysunek 3.1. Koncepcja ogólna projektu

3.6 Szczegółowe omówienie funkcji

3.6.1 Detekcja obiektów

Do realizacji detekcji obiektów wykorzystana zostanie biblioteka OpenCV (Open Computer Vision). Bibliotekę tę cechuje wieloplatformowość, kilka interfejsów programistycznych, modularna struktura oraz wiele algorytmów z zakresu wizji komputerowej i uczenia maszynowego.   
W projekcie użyty został interfejs C++, natomiast platformą docelową jest Linux w dystrybucji Raspbian.

Detekcja obiektów jest zaawansowanym zagadnieniem wizji komputerowej. Biblioteka OpenCV dostarcza wiele funkcji umożliwiających łatwe użycie algorytmów. Poniżej zamieszczony jest opis działania algorytmu detekcji obiektów Viola-Jones.

* Cechy Haara  
    
  Główną ideą działania algorytmu jest wykorzystanie przesuwnego, skalowalnego okna, które posiada tzw. cechy Haara (ang. Haar features).

Cechy te nakładane są na dany obraz w celu określenia przynależności danego obszaru do klasy obiektów poszukiwanych. Dla każdej cechy obliczana jest różnica sumy wartości pikseli znajdujących się na obszarach białych i czarnych.



Rysunek 3.2 Cechy Haara Rysunek 3.3 Wykorzystanie cech Haara do detekcji twarzy

Uwzględniając wszystkie możliwe wymiary i położenia dla okna przesuwnego np. o wymiarach 24x24, liczba cech wynosi ponad 160 000. Ograniczone zasoby sprzętowe standardowych komputerów klasy PC, a tym bardziej systemów wbudowanych nie pozwalają na wykonanie tej skali obliczeń w sensownym czasie. Dlatego podstawową ideą jest wyeliminowanie ze zbioru cech, tych które są niestotne w procesie detekcji. Tutaj z pomocą przychodzą obrazy całkowe(ang. Integral images) i algorytm AdaBoost (Adaptive Boosting).

* Obrazy całkowe

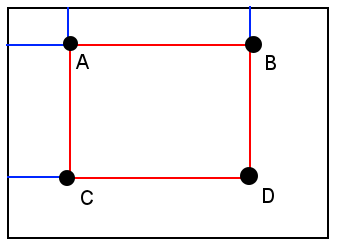
Pozwalają one na szybkie i efektywne obliczanie sumy wartości pixeli w określonym obszarze cechy.

Metoda ta wykorzystuje zależność 3.1, która pozwala obliczyć wartość sumy pixeli z obszaru ograniczonego przez zaledwie cztery punkty A=(x0,y0),B=(x1,y0),C=(x0,y1),D=(x1,y1)

(3.1)

gdzie:   
 (3.2)

Co zilustrowano na rysunku (3.4).



Rysunek 3.4 Obliczanie sumy wartości pixeli fragmentu obrazu

* Algorytm AdaBoost

Algorytm opracowany i zaprezentowany w 1997 r. przez Yoava Freunda i Roberta Schapire jest jednym z wielu realizujących tzw. boosting. W wyniku procesu uczenia algorytmu otrzymujemy silny klasyfikator binarny, który składa się ze słabszych klasyfikatorów z odpowiednimi wagami. Klasyfikator taki daje nam jednoznaczną informację czy dany obiekt należy do klasy poszukiwanych, czy też nie. Budowany jest w procesie uczenia, gdzie na wejście podaje się tzw. zestaw treningowy (ang. training set), a na wyjściu otrzymujemy gotowy klasyfikator zdolny do pracy z nowym zbiorem danych.

Klasyfikator słaby, to taki, którego błąd klasyfikacji jest mniejszy niż 0.5, czyli lepszy niż zwykłe zgadywanie.

Poniżej znajduje się opis algorytmu:

**Algorytm AdaBoost**

**Wejście:** zbiór przykładowych obrazów (x1,y1),..,(xn,yn), gdzie yi=0,1 dla odpowiednio negatywnych i pozytywnych próbek

**Wyjście:** silny klasyfikator H(x).

Kroki algorytmu :

1. Dla każdego z k elementów zestawu trenującego T przypisz identyczną wagę początkową.

W1 (i)= , i Liczba iteracji wynosi N.

2.Dla n=1,…,N

Wybierz słaby klasyfikator o najmniejszym błędzie n

(3.3)

3. Oblicz nową wagę

α n = (3.4)

4.Dla poprawnie sklasyfikowanych przykładów trenujących xi wagi są uaktualniane na podstawie zależności:

Wn+1= (3.5)

Dla niepoprawnie sklasyfikowanych przykładów :

Wn+1= (3.6)

gdzie Z- stały czynnik normalizujący

5. Klasyfikator końcowy:

H(x)= (3.7)

Podsumowując AdaBoost wykorzystuje te cechy, które są w stanie wykryć samodzielnie więcej niż połowe przypadków. Poprzez zmniejszanie wag cech poprawnie wykrywających obiekty i zwiększanie wag cech, które sklasyfikowały obiekty błędnie, algorytm potrafi „skupić się” na trudnych przypadkach.   
Klasyfikator zaproponowany przez autorów biblioteki OpenCV zawiera po wytrenowaniu około 6000 cech. Jest to ogromna redukcja względem wspomnianych 160 000, jednakże wciąż zbyt dużo, aby zapewnić detekcję obiektu w krótkim czasie. Rozwiązaniem jest kaskada klasyfikatorów.

* Kaskada klasyfikatorów

Większość z obszaru analizowanych obrazów nie zawiera twarzy. Stąd wyszedł pomysł, aby ocenić czy obszar może zawierać twarz już na samym początku. Jeśli nie, odrzucić ten region, a skupić się na tych, które faktycznie mogą zawierać twarz. Taki zabieg pozwala oszczędzić mnóstwo czasu i zasobów sprzętowych.  
Zamiast sprawdzać wszystkie 6000 cech autorzy proponują podzielić proces detekcji na wiele etapów. W każdym z etapów są zgrupowane określone cechy nakładane jedna po drugiej, przy czym ilość cech sprawdzanych w kolejnych etapach rośnie. Jeśli obraz nie przejdzie początkowych etapów jest odrzucany i sprawdzany jest kolejny. Natomiast jeśli przejdzie wszystkie etapy – klasyfikowany jest jako twarz. Zasadę działania kaskady ilustruje rysunek 3.5.

Etap N

Etap 2

Etap 1

T

T

T

N

N

N

Odrzuć obraz

Rysunek 3.5 Kaskada klasyfikatorów dla N etapów

Biblioteka OpenCV w pakiecie zawiera także narzędzia, które pozwalają samemu wytrenować klasyfikator zdolny do rozpoznania dowolnych obiektów. Aby jednak wytrenować dość silny klasyfikator potrzeba dużej ilości pozytywnych i negatywnych przykładów. Stąd też zdecydowano się użyć dostarczonych w pakiecie biblioteki gotowych klasyfikatorów. Proces trenowania przedstawiony jest na schemacie (3.5). W wyniku działania narzędzia HaarTraining otrzymujemy plik XML ze zdefiniowanym klasyfikatorem, gotowym do użycia w aplikacji.

Classifier.xml

HaarTraining

Pozytywne

próbki

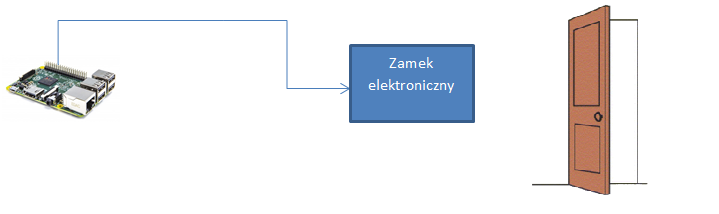
Negatywne

próbki

Rysunek 3.6. Schemat trenowania klasyfikatora

### 3.6.2 Sterowanie urządzeniem zewnętrznym

Sterowanie urządzeniami zewnętrznymi zrealizowane jest przez programowe wystawianie stanów logicznych na portach GPIO płyty. Pozwala to po zestawieniu z układem sterowanym na kontrolę urządzeniem. W projekcie płyta steruje układem z przekaźnikiem 12V. Jest to tylko przykład demonstracyjny czasu reakcji kamery. Sygnały cyfrowe można zastosować do dowolnego przeznaczenia np. włączenie alarmu, światła, klimatyzacji, powitania gościa...



GPIO : HIGH/LOW STATE

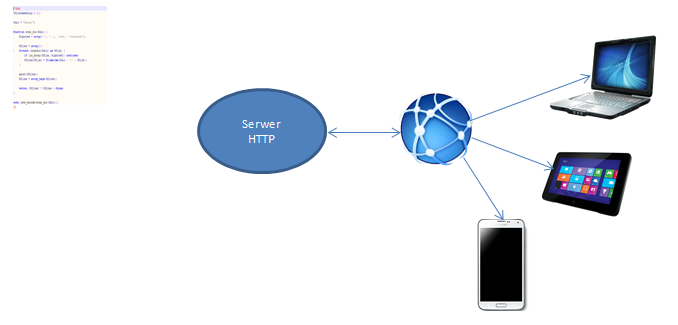
Rysunek 3.7 Koncepcja sterowania urządzeniem zewnętrznym

3.6.3 Logowanie zdarzeń na serwerze HTTP

Interfejs Ethernet zapewnia możliwość interakcji kamery z użytkownikiem. Jej realizacja opiera się o serwer http, który będzie wykonywał skrypt napisany w technologii PHP i jQuery i wynik działania umieszczał na stronie internetowej.  
Efektem czego użytkownik będzie mógł połączyć się z serwerem z dowolnego urządzenia, które posiada przeglądarkę stron WWW.

* Serwer http

Jako serwer wykorzystano Apache 2 – popularny, darmowy serwer http współpracujący z różnymi językami programowania oraz ze wsparciem dla baz danych – MySQL, PHP etc.  
Współpracuje z systemami Linux, Windows i Mac OS. W projekcie uruchamiany jest na Linuxie.



Ethernet

PHP + jQuery

Rysunek 3.8 Koncepcja interfejsu komunikacji z użytkownikiem

Rozdział 4

Realizacja projektu

Rozdział ten przedstawia opis szczegółowy koncepcji przedstawionych w rozdziale 3.

4.1 Oprogramowanie

4.1.1 Narzędzia i środowisko pracy

Jako środowisko robocze wykorzystano system Ubuntu 14.04. Do stworzenia aplikacji użyte zostało środowisko programistyczne Eclipse Kepler. Dzięki wielu wtyczkom dostępnym do tego IDE (ang. Integrated Development Environment), możliwa była min. wygodna współpraca z systemem kontroli wersji GIT. System kontroli wersji pozwolił na bezpieczne rozwijanie projektu, wraz z możliwością śledzenia istotnych zmian.

4.1.2 Aplikacja

Aby zapewnić elastyczność aplikacji i łatwość rozbudowy oraz debugowania zdecydowano się na modularną strukturę. Wyróżnić można w niej moduł odpowiedzialny za „logikę” – Controller oraz moduły wykonujące zadania – Object Detect, Worker, Logger.

Ważnym aspektem działania aplikacji jest komunikacja między modułami. W projekcie wykorzystano mechanizm sygnałów biblioteki Boost.

* **Controller** jest odpowiedzialny za przetwarzanie otrzymanych od innych modułów sygnałów i zlecanie im wykonania odpowiednich zadań. Po otrzymaniu sygnału od modułu Object Detect, Controller z wykorzystaniem mechanizmu sygnałów biblioteki Boost zleca modułowi Worker wykonanie akcji, zaś modułowi Logger przygotowanie pliku logu, który zostanie przetworzony następnie przez serwer HTTP.

Moduł ten odbiera też sygnał zwrotny od **Worker’a**, potwierdzający wykonanie akcji i pozwala wznowić pracę modułu **Object Detect.**

1. class Controller{
2. public:
3. typedef boost::signals::connection conn;
4. short int cam\_w,cam\_h,latency;
5. boost::signal <bool(void \*wsk)>SigC;
6. bool detected;
7. Module \* modules[3];
8. bool logicUnit(int nr,void\* wsk);
9. Controller(int w=320,int h=240,int l=2000);
10. ~Controller();
11. private:
12. conn c\_obd;
13. conn c\_wor;
14. conn c\_log;
15. };

Listing 4.1 Klasa Controller

Logger

Object Detection

Controller

HTTP Server

Worker

Rysunek 4.x Diagram współpracy obiektów

* **Object\_Detection** ma za zadanie wykryć obiekty ze strumienia video (dostarczanego przez kamerę podłączoną do płyty. Obiektami założonymi w projekcie są twarze ludzkie, jednak aplikacja jest pod tym względem elastyczna tj. wystarczy wytrenować klasyfikator dowolnego obiektu (narzędzie HaarTraining dostarczone w pakiecie z biblioteką OpenCV) i dołączyć wygenerowany plik XML. Po wykryciu twarzy wysyłany jest sygnał do Controllera wraz ze wskaźnikiem do struktury *frame* z wyodrębnionymi twarzami z klatki.

frame n

camera

frame 3

frame 2

frame 1

Obj\_Det. enabled= true?

N

T

detect

Signal

frame pointer

N

Controller

Wykryto obiekt?

T

Haarcascade\_frontalface.xml

Obj\_Det enabled

Rysunek 4.x schemat działania modułu detekcji obiektów

* **Worker** – zadaniem tego modułu jest wykonanie właściwej akcji zadanej przez kamerę.   
  W projekcie jest to wystawienie stanu logicznego wysokiego na porcie GPIO, który może przykładowo sterować układem z przekaźnikiem, lub innym układem.

Do implementacji użyto biblioteki **wiringPi** autorstwa Gordona Hendersona. Biblioteka ta dostarcza łatwy w obsłudze interfejs pozwalający sterować portami GPIO, generować przebiegi PWM, odmierzać czas, oraz wiele innych.

Istotnym zagadnieniem jest mapowanie portów\* wirtualnych stosowanych w bibliotece na rzeczywiste wyprowadzenia GPIO.

\*Opis mapowania portów znajduje się w dodatku

Na poniższym fragmencie kodu zaprezentowano sposób inicjalizacji biblioteki przez wywołanie funkcji **wiringPiSetup()**, a następnie zdefiniowanie portów w konstruktorze klasy Worker.

1. Worker::Worker(){
2. name="WORKER";
3. #ifdef BRD\_BUILD
4. wiringPiSetup();
5. pinMode (0,OUTPUT);//GPIO\_0 (BCM\_GPIO 17) (PHYS. HEADER -> 11)
6. pinMode (1,INPUT); //GPIO\_1 (BCM\_GPIO 18) (PHYS. HEADER -> 12)
7. #endif
8. }

Listing 4.1. Inicjalizacja portów GPIO

Funkcją odpowiedzialną za akcje wykonywane przez moduł Worker jest **work(void \* wsk)**. Jako argument przyjmuje wskaźnik typu void\*, który następnie jest konwertowany do zmiennej **int state** i dekodowany jest rozkaz od Controllera.

1. bool Worker::work(void\* wsk){
2. std::cout<<name<<" RUNNING"<<std::endl;
3. #ifndef TIME\_TEST
4. int state= \*(static\_cast<int\*>(wsk));
5. #else
6. int state = 0;
7. #endif
8. if(state == 0){
9. #ifdef BRD\_BUILD
10. digitalWrite(0,HIGH);
11. #endif
12. std::cout<<"WORKER sets HIGH state"<<std::endl; //otwórz drzwi
13. #ifndef TIME\_TEST
14. SigW(2,wsk);
15. #endif
16. }
17. else if (state == 2){
18. #ifdef BRD\_BUILD
19. digitalWrite(0,LOW);
20. #endif
21. std::cout<<"WORKER sets LOW state"<<std::endl; //zamknij drzwi
22. #ifndef TIME\_TEST
23. SigW(2,wsk);
24. #endif
25. }
26. else
27. {}
28. return false;
29. }

Listing 4.2 Funkcja work klasy Worker

Po wystawieniu stanu moduł przekazuje sterowanie **Controllerowi** wysyłając sygnał zwrotny **SigW** wraz z argumentem w postaci otrzymanego na początku wskaźnika **wsk**.

* **Logger**

4.2 Realizacja sprzętowa

4.2.1 Raspberry Pi 2 rev.B

4.2.2 Układ SoC BCM 2836

4.2.3 Moduł kamery

W podrozdziale omówione zostaną dostępne na rynku rozwiązania, które można wykorzystać w projekcie. Do kryteriów kluczowych w projekcie należy zaliczyć : szybkość kamery, interfejs komunikacji oraz cenę.

* Kamera z interfejsem USB

USB (ang. Universal Serial Bus) – uniwersalny sprzętowy interfejs komunikacyjny opracowany przez firmy Microsoft, Intel, Compaq, IBM i DEC. Wykorzystujący magistralę szeregową o prędkościach transmisji danych odpowiednio w standardach :

* USB 1.1 do 12Mbit/s (1,5 MB/s)
* USB 2.0 do 480 Mbit/s (60MB/s)
* USB 3.0 do 5 Gbit/s

Moduły z interfejsem USB są dosyć popularne zwłaszcza w rozwiązaniach desktopowych, gdzie szybkość przesyłu danych nie jest kluczowa. Świetnie sprawdzają się jako urządzenia dostarczające strumienia wideo dla aplikacji VoIP (ang. Voice over IP). Jako dodatkowy atut można wyróżnić często stosowane wbudowane mikrofony.

Do zestawienia wybrano popularną kamerę **HD Webcam C310** firmy ***Logitech***. Kamera należy do średniej półki cenowej, jej wybrane parametry przedstawiono poniżej.

|  |  |
| --- | --- |
| Standard transmisji danych | Full Speed USB 2.0 |
| Rozdzielczość Video | 320x240(QVGA), 640x480(VGA) |
| Maksymalna ilość klatek/s | 30fps@ 352x288(CIF)  15fps@640x480(VGA) |
| Pole widzenia FOV (ang. Field of View) | 50o |
| Wbudowany mikrofon | Tak |
| Cena | ok. 120zł |

Tabela 4.1 Podstawowe parametry kamery HD Webcam C310

* Dedykowany moduł RaspiCam z interfejsem MIPI CSI-2

MIPI CSI-2 (Mobile Industry Processor Interface Camera Serial Interface) definiuje jednokierunkowy interfejs między modułem kamery, a procesorem. Pozwala na uzyskanie transmisji danych do 4Gbps ( po 1Gbps na linię danych). Moduł wykorzystywany w projekcie posiada 2 linie, co daje łączną przepustowość na poziomie 2Gbps.

Innym rozwiązaniem jest dedykowany dla platformy Raspberry moduł RaspiCam z sensorem **OV5647** firmy ***OmniVision.***Moduł ten wysyła dane za pomocą szyny Camera Serial Interface (CSI-2) do procesora BCM2835. Wykorzystywana jest do tego złącze taśmowe 15-pinowe podłączane do gniazda CSI Camera płyty Raspberry Pi 2. Według danych producenta kamera jest w stanie dostarczyć strumień wideo o rozdzielczości 1920x1080 przy 30 klatkach na sekundę.

|  |  |
| --- | --- |
| Standard transmisji danych | MIPI CSI-2 |
| Rozdzielczość Video | 1080p,720p,640x480(VGA) |
| Maksymalna ilość klatek/s | 30fps@ 1080p  60fps@720p  90fps@640x480 |
| Pole widzenia FOV (ang. Field of View) | 53.5o (horizontal) 41.41o (vertical) |
| Wbudowany mikrofon | Nie |
| Cena | 85zł |

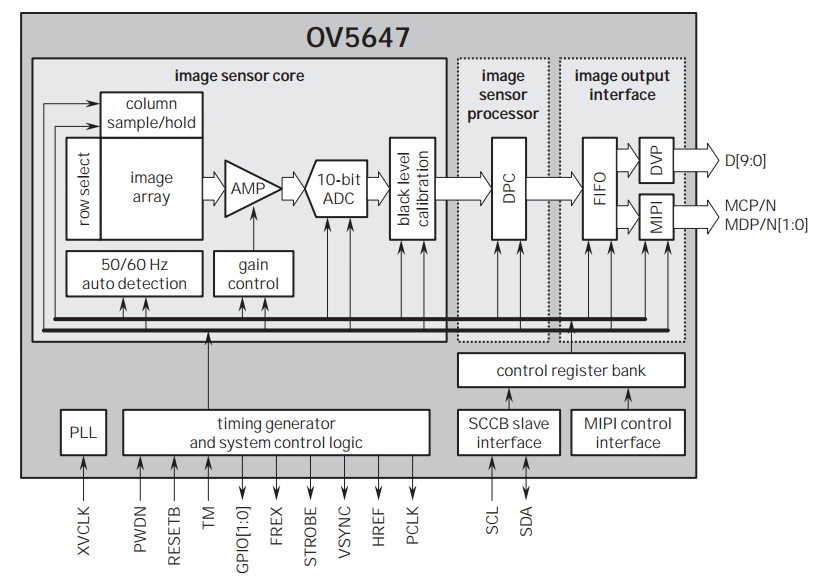
Tabela 4.2 Podstawowe parametry kamery RaspiCam

Jak widać na korzyść modułu dedykowanego działa nie tylko cena, ale także dużo większa wydajność, wsparcie sprzętowe platformy Raspberry, a przede wszystkim transmisja danych dużo lepsza niż w przypadku kamer z interfejsem USB, co ma wpływ na założoną szybkość kamery. Poniżej znajduje się dokładny opis wybranego modułu RaspiCam.

Sercem modułu jest sensor OV5647 w skład którego wchodzą takie bloki jak :

* rdzeń sensora optycznego z układami odpowiedzialnymi za przechwycenie obrazu i wstępne przetworzenie go
* procesor obrazu DPC dokonujący dalszej obróbki
* interfejs wyjściowy przekształcający dane do odpowiedniego formatu

Istotną rolę pełnią bloki generujące sygnały synchronizujące obrazu niezbędne układowi odbierającemu do poprawnego odtworzenia obrazu z danych dostarczonych przez sensor.  
Urządzenie komunikuje się z procesorem za pomocą interfejsu SCCB ( ang. Serial Camera Control Bus ). Interfejs ten opiera się na protokole komunikacyjnym I2C (znanym też jako Two Wire Interface) wykorzystującym dwie linie : danych - SDA i zegara – SCL. Urządzenie Master – w tym przypadku procesor BCM2836 wysyła do urządzenia Slave - sensora odpowiednie komunikaty, za pomocą których możliwa jest zmiana jego parametrów.



Rysunek4.x.x. Schemat funkcjonalny modułu RaspiCam

**Rozdział 5**

**Testowanie**

Jednym z kluczowych zagadnień każdego projektu jest weryfikacja poprawności działania urządzenia. Oprócz weryfikacji funkcjonalnej w pracy tej istotne są pomiary czasów odpowiedzi poszczególnych modułów, a przede wszystkim modułu Object\_Detection odpowiedzialnego za obsługę kamery i przetwarzanie obrazu. Pomiary te pozwolą oszacować całkowity czas odpowiedzi urządzenia, a także dzięki nim będzie można porównać działanie układu opartego na platformie Raspberry i kamerze dedykowanej oraz komputerze klasy PC z kamerą USB.

**5.1Implementacja modułu testowego**

Do przetestowania działania funkcji oraz pomiaru czasu stworzony został oddzielny moduł **Time\_Test.** Moduł ten wykorzystuje implementację obiektowo-orientowaną aplikacji głównej, co pozwala w łatwy i przejrzysty sposób dokonać testu i pomiaru czasu działania każdego modułu roboczego.

1. class Time\_Test{
2. private:
3. int module\_nr;
4. std::vector <Module\*> wektor;
5. public:
6. void add(Module \*\*);
7. void measure\_time();
8. void display\_results();
9. };

Listing 5.1 Klasa testująca

Praca modułu opiera się na trzech funkcjach:

* void add(Module\*\*) – metoda ta przyjmuje jako argument tablicę wskaźników na obiekty typu Module i umieszcza je w składowej **wektor**.
* bool measure\_time() – metoda odpowiedzialna za pomiar czasów poszczególnych modułów oraz całkowitego.

Metoda ta wykorzystuje dostarczone wraz z biblioteką standardową języka C++ bibliotekę chrono, która opiera się na 3 głównych obiektach, jak :

* *duration* – mierzące okres czasu – w projekcie są to globalne zmienne : **time\_span** (czas działania modułu), **total\_time** (czas całkowity) oraz wektor czasów **times** przechowujący poszczególne czasy działania modułów.
* *time\_point* – odniesienie do punktu w czasie – w projekcie są to zmienne **t1** oraz **t2**
* *clock* – framework, odpowiadający za odniesienie czasu określonego przez obiekty typu *time\_point* względem czasu rzeczywistego. – w projekcie są to obiekty typu  *time\_point* oraz metoda *now()* zwracająca aktualny czas frameworku *high\_resolution\_clock*, korzystającego z zegara wysokiej precyzji.

1. void Time\_Test::measure\_time(int n){
2. printf("Start of time tests \n");
3. high\_resolution\_clock::time\_point t1;
4. high\_resolution\_clock::time\_point t2 ;
5. duration<double> time\_span,total\_time;
6. std::vector <std::chrono::duration<double>> times;
7. n=N;
8. printf("\nNumber of iterations : %d",n);
9. for (int i=0; i<module\_nr; ++i){
10. t1 = high\_resolution\_clock::now();
11. c=0;
12. while(c<N){
13. assert(! wektor[i]->work());
14. c++;
15. }
16. t2 = high\_resolution\_clock::now();
17. time\_span = duration\_cast<nanoseconds>(t2 - t1);
18. times.push\_back(time\_span);
19. total\_time += time\_span;
20. }

5.2 Listing Implementacja funkcji testującej

Funkcja przedstawiona na listingu 5.2 posiada pętlę główną for – w ramach której następuje iteracja po kolejnych modułach i wywołanie polimorficzne funkcji **work()** dla każdego z nich. Wywołanie to jest objęte dodatkową pętlą while, która jest odpowiedzialna za uśrednienie wyników pomiaru czasu. W każdej iteracji pętli for funkcja ta zapisuje w zmiennych **t1** i **t2** punkty czasowe odpowiednio przed i po wywołaniu N-razy (n jest argumentem wejściowym wywołania programu głównego) funkcji składowej **work()** badanego modułu, a następnie zapisuje wartość ich różnicy do zmiennej **time\_span**. Czas zmierzony dla każdego modułu jest umieszczony w wektorze **times**. Jednocześnie w zmiennej **total\_time** przechowywany jest czas całkowity wykonania sekwencji modułów.

**5.2Wyniki pomiarów**

W tabeli 5.1 znajduje się zestawienie pomiarów czasów wykonanych na platformie Raspberry Pi 2 z modułem RaspiCam oraz na komputerze klasy notebook: ASUS X53S z modułem kamery USB.

Specyfikacja platformy ASUS:

* CPU: Inter Core i5 2410M - 2 cores @ 2,3 GHz
* RAM: 8GB 1333MHz DDR3
* GPU: Nvidia GeForce GT 540M 1GB DDR3
* Dysk twardy 750GB SATA 7200 obr./min.

Systemem wykorzystanym na platformie ASUS jest Linux Ubuntu 14.04 uruchamianym w maszynie wirtualnej programu VirtualBox (wersja 4.3.30).   
Zasoby przydzielone maszynie wirtualnej :

* 4GB pamięci RAM
* 2 wirtualne rdzenie CPU
* 128 MB Video Memory

Specyfikacja platformy Raspberry :

* CPU: BCM 2836 - quad core @ 1GHz
* RAM: 1GB pamięci DDR2
* GPU: VideoCore IV (przydzielone 128 MB pamięci RAM z zasobów wspólnych)
* Karta pamięci microSD class 10 16GB

Poniżej znajdują się wyniki dla N = 10, 100 i 500 powtórzeń wywołania funkcji roboczych poszczególnych modułów oraz dla rozdzielczości strumienia: 320x240.

* Ilość iteracji N = 10

|  |  |  |
| --- | --- | --- |
|  | Platforma testowa | |
| Moduł | ASUS x53S | Raspberry Pi 2 |
| Object\_Detection [ms] | 254,762741 | 549,6572 |
| Worker [us] | 832,679 | 98,599 |
| Logger [ms] | 0,712403 | 6,425799 |
| Całkowity średni czas [ms] | 256 | 556 |

* Ilość iteracji N = 100

|  |  |  |
| --- | --- | --- |
|  | Platforma testowa | |
| Moduł | ASUS x53S | Raspberry Pi 2 |
| Object\_Detection [ms] | 137,7835103 | 523,30284 |
| Worker [us] | 387,455 | 80,37 |
| Logger [ms] | 1,907358 | 6,440570 |
| Całkowity średni czas [ms] | 140 | 529 |

* Ilość iteracji N = 500

|  |  |  |
| --- | --- | --- |
|  | Platforma testowa | |
| Moduł | ASUS x53S | Raspberry Pi 2 |
| Object\_Detection [ms] | 118,555288 | 525,432592 |
| Worker [us] | 95,112 | 77,03 |
| Logger [ms] | 1,339017 | 6,264304 |
| Całkowity średni czas [ms] | 119 | 531 |

Tabela 5.2.1 Zestawienie wyników pomiarów czasu dla różnych N i rozdzielczości 320x240

5.3 Uruchomienie testów

Na proces testowania składa się :

* Zbudowanie projektu
* Załadowanie modułu sterownika V4L2
* Uruchomienie aplikacji testowej z odpowiednimi argumentami

Na zbudowanie projektu składają się min. Kompilacja warunkowa ( zbędne do testowania bloki aplikacji głównej są pomijane), oraz wywołanie skryptu budującego makefile.

Załadowanie sterownika możliwe jest z poziomu linii komend i musi być koniecznie wykonane przed uruchomieniem aplikacji. Minimalna wersja komendy ta jest następująca:  
uv4l - -driver raspicam - -video\_nr *nr*

Gdzie nr to index pod jakim zostanie zarejestrowane urządzenie RaspiCam przez sterownik V4L2 w systemie (np. dla nr = **0** będzie to : /dev/video**0**).  
Dodatkowe argumenty pozwalające konfigurować sensor z poziomu sterownika to :

--encoding yuv420

--width 320

--height 240

Atrybut - -encoding pozwala wymusić format kodowania obrazu, w tym przypadku zastosowany jest format YUV420. Kolejne dwa argumenty są odpowiedzialne za szerokość i wysokość obrazu strumienia(w projekcie można je pominąć z uwagi na to, że z poziomu aplikacji także są konfigurowalne, co jest opisane niżej oraz w rozdziale 4.).

Uruchomienie wymaga podania ścieżki pliku binarnego cam wraz z argumentami oddzielonymi spacjami:

./cam [width] [height] [N]

[width] – szerokość klatki strumienia video

[height] – wysokość klatki strumienia video

[N] – ilość iteracji ( opisane w paragrafie 5.2)

Przykładowe wywołanie : ./cam 320 240 100

Uruchomi w rezultacie aplikację testową z parametrami strumienia : rozdzielczość 320x240 oraz wykona uśrednione na 100 iteracjach testy.

**5.4 Wnioski**

Na podstawie pomiarów stwierdzić można, że kamera zrealizowana na platformie Raspberry ustępuje urządzeniu o dużo większej mocy obliczeniowej jakim jest laptop ASUS x53S.

Jak widać przy większej ilości iteracji średnie czasy są znacznie mniejsze. W przypadku platformy ASUS wyraźnie widać względny spadek czasów. Najprawdopodobniej wynika to z optymalizacji sprzętowej procesora i jednostki GPU przy wielokrotnym odnoszeniu się do tego samego fragmentu pamięci.

W obu przypadkach na całkowity czas odpowiedzi kamery ma wpływ głównie działanie modułu Object\_Detection. Nie jest to zaskakujący rezultat, gdyż moduł ten wykonuje dwa ważne zadania tj. pośredniczy w przechwyceniu strumienia video oraz dokonuje przetworzenia każdej przez algorytm wykorzystujący kaskady Haar’a. Zdecydowana przewaga platformy ASUS wynika z dużo większej wydajności jednostki GPU i CPU względem platformy Raspberry. Dla pomiaru czasu działania modułu Logger’a kluczowym aspektem jest nośnik na jakim dokonywany jest zapis zdjęcia wykrytego obiektu.   
Raspberry Pi 2 wykorzystuje kartę Micro SD class 10, co wg specyfikacji powinno gwarantować minimalny transfer na poziomie 10MB/s. W przypadku ASUS’a jest to dużo szybszy dysk twardy wykorzystujący magistralę SATA, zapewniającą wg specyfikacji transfer na poziomie ok. 179MB/s. Stąd też widoczna jest róznica w czasach.

Jako, że laptop ASUS nie posiada portów GPIO, test modułu Worker w zasadzie pokazuje tylko czas wywołania funkcji roboczej.

Podsumowując – platforma Raspberry pod względem wydajności ustępuje rozwiązaniom desktopowym, jednak różnica nie jest ogromna. Czas reakcji na poziomie 0,5s z powodzeniem pozwala na wykorzystanie urządzenia w wielu dziedzinach. Ponadto uwzględniając atuty takie jak: niewielkie gabaryty, niskie zużycie energii oraz możliwość sterowania urządzeniami zewnętrznymi Raspberry Pi 2 zdecydowanie wygrywa z platformą ASUS.

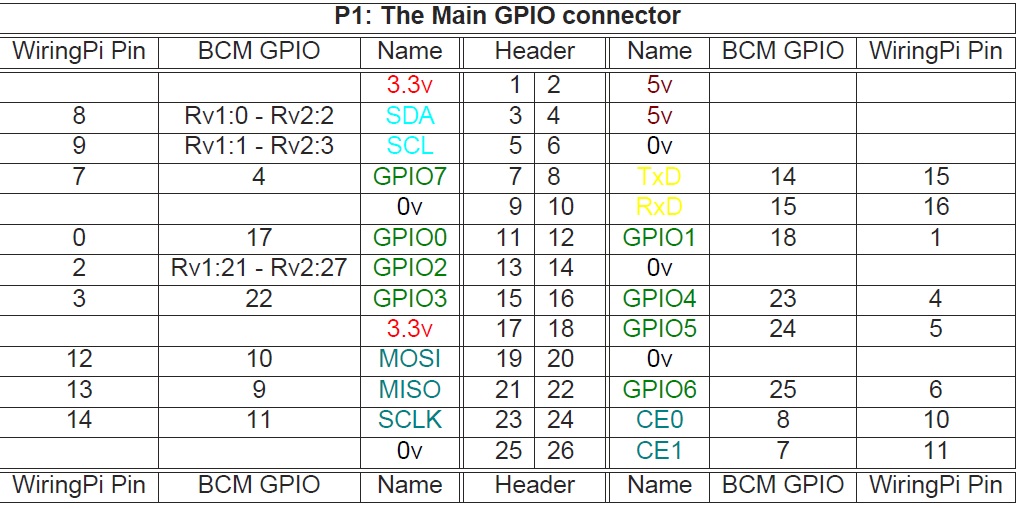
**Dodatek X**

**Mapowanie portów procesora BCM2836**

Wyprowadzenia portów GPIO na płycie są identyczne w przypadku płyt Raspberry Pi, Raspberry Pi rev. B i Raspberry Pi 2 rev. B – jest to 40 złączy typu goldpin. Jednak wraz ze zmianą jednostek SoC w poszczególnych wersjach płyt zmianie uległy też rozkład i funkcje odpowiednich pinów.

Biblioteka **wiringPi** dostarcza wygodny interfejs obsługi portów GPIO układów z rodziny Raspberry Pi, dzięki któremu programista nie musi sprawdzać tych portów, zapewniając przenośność kodu między różnymi wersjami urządzeń.

Autor biblioteki proponuje własny schemat numeracji pinów rozpoczynający się od GPIO0 do GPIO7 na głównym złączu P1. Oprócz tego dostępne są też porty specjalnego przeznaczenia jak np. złącza SDA i SCL interfejsu I2C.



Rys. X.1 Mapowanie portów GPIO biblioteki wiringPi