**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.

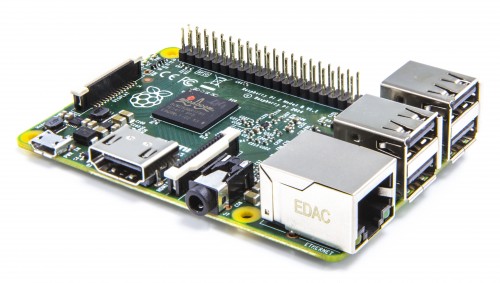
Pod hasłem systemy inteligentne jak widać kryje się wiele różnych zagadnień, w których sprawne poruszanie się i umiejętność zintegrowania ich w jeden system jest dla projektanta kluczowe.

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 systemów wbudowanych

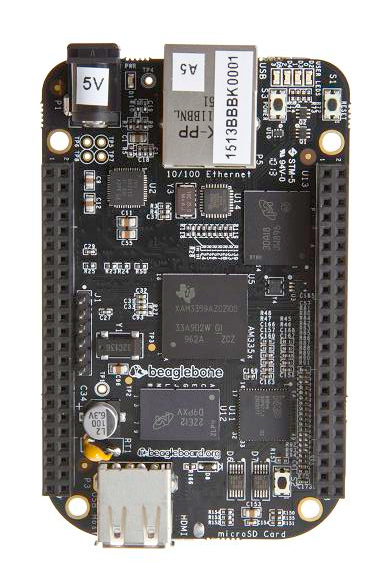
Poniżej przedstawiono kilka z wielu dostępnych na rynku urządzeń, które mogą być zastosowane np. do rozwiązań typu „smart”.

**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

1.2. 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.

1.2. 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 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
* Połączenie podzespołów : płyty z układem sterowanym
* Napisanie aplikacji uruchamianej na płycie
* Skonfigurowanie serwera http i przygotowanie skryptu PHP/jQuery
* Wykonanie testów działania projektu

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 :

-otwarciem drzwi

-umieszczeniem logu na serwerze

W projekcie przyjęto założenie, że przybliżony łączny czas między tymi zdarzeniami nie powinien przekraczać 2s. 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 wykryciu zamkniętych drzwi **Worker** wysyła sygnał do **Controllera**
7. Odebranie sygnału od **Workera** powoduje wznowienie pracy **Object Detect**

3.Implementacja

4.Testowanie

Ważnym aspektem poprawnej pracy systemu są testy. <cośtam cośtam>

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 bgeneruje 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

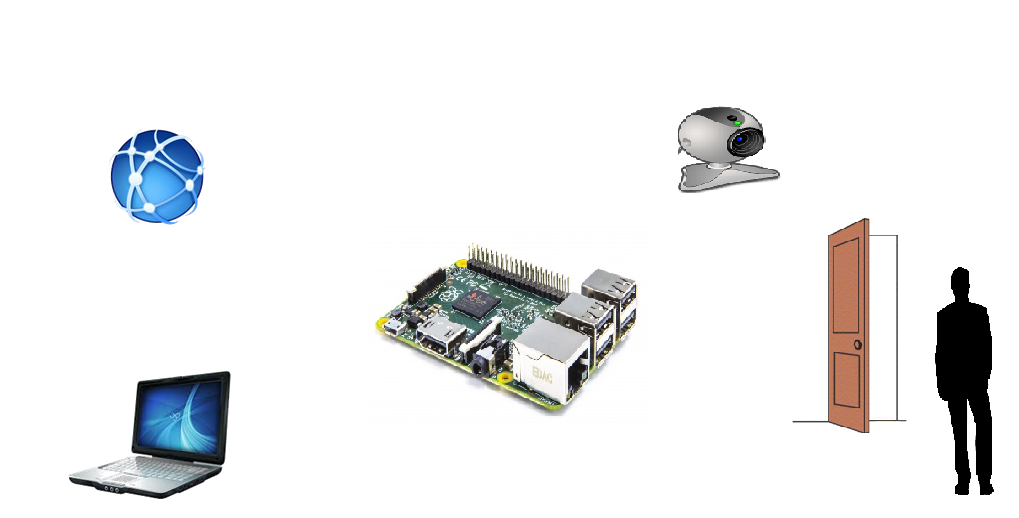
W zależności od wymagań mogą to być: manipulatory robotów, sterowniki bram, napędy, czujniki, a także inne aplikacje, które oczekują na informacje od kamery ( np. ilość obiektów i ich czasy poruszania się, czy też orientacja w przestrzeni).

* 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 pixeli 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

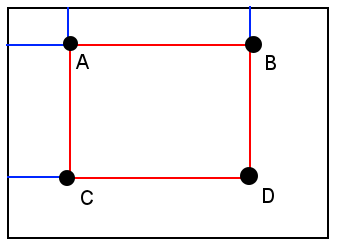
Pozwalają one na szybkie i efektywne obliczanie sumy wartości pixeli w określonym obszarze cechy.

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 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 t

(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

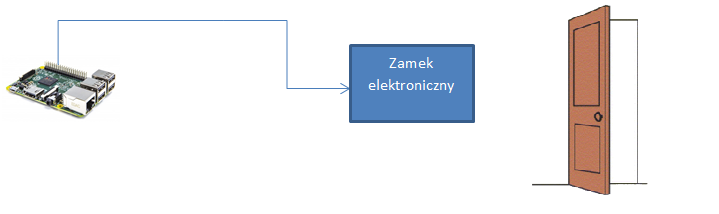
Negatywne

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.



GPIO : HIGH/LOW STATE

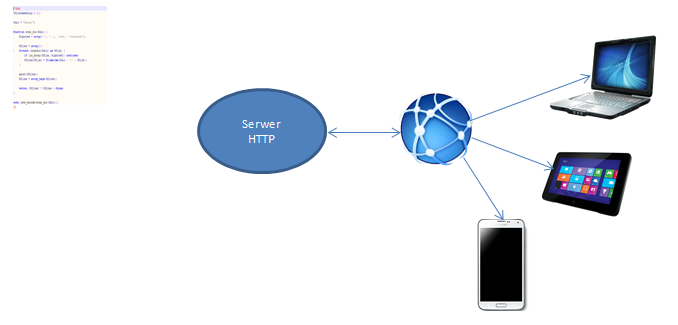
Rysunek 3.7 Koncepcja sterowania urządzeniem zewnętrznym

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 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**, co jest informacją o zamknięciu drzwi i pozwala wznowić pracę modułu **Object Detect.**

* **Object Detect** 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.

camera

frame n

frame 3

frame 2

frame 1

on=true?

N

T

Signal

frame pointer

N

detect

Wykryto obiekt?

T

Controller

Haarcascade\_frontalface.xml

on

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 steruje układem z przekaźnikiem.
* **Logger**