**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI

Praca dyplomowa magisterska

# *Algorytm sterowania wykorzystujący sztuczne sieci neuronowe dla bezzałogowego statku latającego typu TRICOPTER*

Autor: *Rafał Włodarz*
Kierunek studiów: *Automatyka i robotyka*
Opiekun pracy: *dr hab. Adam Piłat*

Kraków, 2015

*Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.*

*Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.*

# Spis treści

# 1. Wstęp

## 1.1. Cele pracy

## 1.2. Zawartość pracy

# 2. Sztuczne sieci neuronowe

Rozdział ten zawiera informacje na temat sieci neuronowych, ich architektury, zasady działania oraz algorytmów uczenia.

## 2.1. Początki sieci neuronowych

Początki prac nad poznaniem procesów zachodzących w mózgu datuje się na rok 1943. W pracy McCulloch'a oraz Pitts'a przedstawiono matematyczny model neuronu, który zapoczątkował badania związane z tym tematem. W 1949 roku Donald Hebb odkrył, iż informacje przechowywane w sieci neuronowej są reprezentowane jako wartości wag pomiędzy poszczególnymi neuronami. Na podstawie tych informacji zaproponował on pierwszy algorytm uczenia sieci neuronowej, który został nazwany regułą Hebba. Już wtedy odkryto, iż bardzo dużą zaletą sieci jest równoległy sposób przetwarzania informacji oraz metodologia uczenia, która zastępuje tradycyjny proces programowania.

# 3. Architektura statku latającego typu tricopter

Poniższy rozdział przedstawia zbiór podstawowych zagadnień związanych z konstrukcją wirnikowca typu tricoper oraz zawiera informacje na temat zasad sterowania układem.

## 3.1. Konstrukcja tricopter-a

## 3.2. Budowa modelu

# 4. Aplikacja sterująca

W niniejszym rozdziale przedstawiono informacje na temat aplikacji sterującej. Zagadnienia teoretyczne konieczne do zrozumienia problemu, poszczególne etapy przygotowania platformy sprzętowej, oraz proces tworzenia systemu czasu rzeczywistego.

## 4.1. System czasu rzeczywistego

System czasu rzeczywistego (ang. real-time system) to system, który przetwarza każdy rodzaj informacji i który musi reagować na sygnały wejściowe - bodźce generowane z zewnątrz w skończonym i określonym czasie. Jego poprawne działanie zależy zarówno od prawidłowych rezultatów logicznych, jak również od czasu reakcji. Na podstawie tych kryteriów są one dzielone na:

– Systemy o ostrych wymaganiach czasowych (ang. hard real-time) - wymagania czasowe muszą być skrupulatnie przestrzegane, naruszenie ram czasowych może wpłynąć na życie ludzkie, środowisko czy też sam system,

– Systemy o słabych wymaganiach czasowych (ang. soft real-time) – głównym kryterium oceny tych jest średni czas odpowiedzi. Sporadyczne opóźnienie nie powoduje zagrożenia lecz jedynie wpływa negatywnie na ocenę całego systemu,

– Systemy o solidnych wymaganiach czasowych (ang. firm real-time) - są one kombinacją systemów o wymaganiach ostrych oraz słabych. Naruszenie kryterium czasowych może pojawiać się okazjonalnie. Często dla lepszej oceny systemu stosuje się ograniczenia czasowe o charakterze słabym- krótsze, których przekroczenie nie powoduje katastrofy oraz óstrym- dłuższe, których naruszenie oznacza nieprawidłowe działanie systemu.

Bez względu na to do której grupy systemów zalicza się aplikację musi ona charakteryzować się następującymi cechami:

– Ciągłość działania - powinny działać nieprzerwanie w okresie od uruchomienia systemu do jego wycofania,

– Zależność od otoczenia - zachowanie opiera rozpatruje się w kontekście otoczenia. Prowadzone obliczenia zależą od zdarzeń oraz danych pochodzących z zewnątrz układu,

– Współbieżność - struktura systemu narzuca, aby jednoczesne zdarzenia były obsługiwane równocześnie przez szereg procesów,

– Przewidywalność - zdarzenia i dane generowane przez otoczenie pojawiają się przypadkowo co nie narusza deterministycznego zachowania systemu,

– Punktualność - odpowiedź systemu na bodźce zewnętrzne powinna być dostarczona w odpowiednich momentach - wymaganych ramach czasowych.

Z pewnością aplikacja sterująca obiektami latającymi powinna spełniać wszystkie powyższe założenia. Gdyby któreś z nich nie zostało spełnione jakiekolwiek próby sterowania zakończyły by się porażką.

Dynamika wielokomórkowców wymaga od aplikacji bardzo szybkiego czasu reakcji na zewnętrzne impulsy. Opóźnienie sterowania w takim przypadku powoduje bardzo negatywne skutki do których zaliczamy: brak kontroli nad obiektem, utrata stabilności w powietrzu oraz niekontrolowane zetknięciu się z przeszkodą. Wszystkie wymienione zachowania mogą powodować bardzo duże zniszczenia dla modelu, jego otoczenia oraz zagrażać zdrowiu operatora oraz innych osób znajdujących się w zasięgu działania modelu.

Na podstawie definicji oraz powyższej analizy skutków określono, iż aplikacja sterującą zalicza się do systemu hard-real time.

## 4.2. Konfiguracja beaglebone black

Ze względu na kontynuację prac nad modelem bazowa konfiguracja beaglebone black wraz z dodatkowymi czujnikami oraz urządzeniami peryferyjnymi opisano w pracy

W trakcje powstawania niniejszej pracy na rynku dostępne są nowsze czujniki, które mogłyby podnieść jakość oraz częstość pomiarów, jednak ingerencja w konfigurację uniemożliwiła by porównanie algorytmów tradycyjnych z sieciami neuronowym stąd wszystkie testy zostały przeprowadzone na tej samej konfiguracji.

## 4.3. Przygotowanie systemu operacyjnego

Ze względu na wcześniej wspominaną specyfikę systemu sterującego oraz zastosowanie platformy sprzętowej typu mini PC wraz z systemem operacyjnym typu UNIX, ważne jest, aby wyeliminować wszelkie możliwe przerwania oraz inne aspekty, które wpływają na płynność oraz czas wykonywania się aplikacji sterującej.

Wprowadzono usprawnienie w postaci instalacji systemu operacyjnego wyposażonego w jądro czasu rzeczywistego (ang. real-time kernel). Jądro to określane jest również jako w pełni wywłaszczalne. Oznacza to, iż dopuszczane jest wywłaszczenie procesu działającego w trybie jądra. Cecha ta wraz z wcześniej narzuconymi priorytetami dla poszczególnych procesów gwarantuje, iż aplikacja sterująca, uruchomiona z wysokim priorytetem nie zostanie wywłaszczona na zbyt długi czas przez inne procesy systemowe.

Zabieg ten pozwala nam spełnić podstawową cechę systemów czasu rzeczywistego, którą jest punktualność.

## 4.4. Tworzenie aplikacji niezależnej od platformy sprzętowej

Dobrze zaprojektowana aplikacja sterująca obiektami latającymi powinna mieć możliwa uruchomienia na różnorodnych platformach sprzętowych, bez względu na architekturę wykorzystanych do ich budowy procesorów.

Podstawowa wersja aplikacji została poddana testom, które porównały wyniki działania sieci neuronowej na 2 całkowicie odmiennych platformach.

### 4.4.1. Najważniejsze aspekty

Pierwszym, a zarazem najważniejszym aspektem jest problem który może powstać podczas mnożenia liczb zmiennoprzecinkowych. W przypadku wykonywania tej operacji

Konfiguracja miała na celu uwydatnienie problemu błędu obcięcia liczb zmiennoprzecinkowych

### 4.4.2. Testy

Do testów wykorzystano komputer stacjonarny z 64 bitowym procesorem Intel i5-4670 3.40 Ghz oraz układ beaglebone black, który został wyposażony w procesor 32 bitowy ARM Cortex-A8 o częstotliwości taktowania 1GHz.

W celu uwydatnienia problemu stworzono sieć neuronową posiadającą po jednej z warstw wejściowej, ukrytej, wyjściowej. Wagi poszczególnych neuronów zostały dobrane w sposób losowy tak, jednak wszystkich z nich wykorzystują maksymalną dostępną dokładność. Następnie na wejście sieci podawana jest stała wartość. Po jej przetworzeniu otrzymujemy wynik

## 4.5. Architektura systemu sterującego

# 5. Testy systemu sterującego

# 6. Podsumowanie

# Bibliografia