



Uniwersytet Jagielloński w Krakowie
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Jakub Gącik

Nr. albumu 1113727

Opracowanie i implementacja algorytmu sterowania ramieniem robota

Praca licencjacka
na kierunku Informatyka

Praca wykonana pod kierunkiem
dr Grzegorza Zuzła
z Zakładu Doświadczalnej Fizyki Komputerowej

Kraków 2017

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Kraków, dnia

Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Kraków, dnia

Podpis kierującego pracą

Spis treści

Wstęp	4
Rozdział 1. Ogólne przedstawienie problemu	5
1.1. Budowa ramienia robota - założenia	5
1.2. Dobór podzespołów i technologii	5
Rozdział 2. Opis wybranych technologii	7
2.1. Zasada działania i specyfikacja serw	7
2.2. Pomiar różnicowy, przewaga nad innymi rozwiązaniami	10
2.3. Przygotowany rezystor bocznikujący - dobór parametrów części	11
Rozdział 3. Program dla mikrokontrolera	13
3.1. Środowisko ARDUINO	13
3.2. ATmega 2560 - wzmacniacze różnicowe	13
3.3. Transmisja szeregową ARDUINO \Leftrightarrow PC oraz przyjęty protokół danych	15
3.4. Brak pomiaru różnicowego w interfejsie programowym ARDUINO - działania na rejestrach mikrokontrolera	15
Rozdział 4. Aplikacja komputerowa GUI	17
4.1. Wybór języka programowania	17
4.2. Struktury użyte w aplikacji oraz wykrywanie przeciążeń serwomechanizmów	17
4.3. Inicjalizacja połączenia i sterowanie serwami - ciąg dalszy przyjętego protokołu danych	18
4.4. Prezentacja danych	19
4.5. Wykorzystanie kontrolera gier do sterowania ramieniem	20
Rozdział 5. Podsumowanie oraz przykłady wykorzystania projektu	22
Podziękowania	25
Bibliografia	26

Wstęp

Wybór tematu poniższej pracy jest uzasadniony zainteresowaniami autora w dziedzinie robotyki i elektroniki, jak również sugestiami osób promujących pracę oraz faktem, że roboramiona są powszechnie używane w projektach elektronicznych i maszynach przemysłowych. Podstawą w poprawnym działaniu takich ramion jest odpowiednie zaprojektowanie, ale nawet najbardziej dopracowana maszyna, może być bezużyteczna, jeśli byt odpowiadający za jej pracę jest nieprawidłowo przygotowany. Dlatego bez odpowiedniego programu obsługującego elektronikę żaden projekt nie ma sensu.

Najbardziej popularne i ogólnie dostępne części na rynku elektronicznym są zazwyczaj bardzo prymitywne w budowie, co oczywiście ma swoje zalety i wady. Do tych drugich należy awaryjność, ale nie spowodowana błędami projektowymi czy produkcyjnymi, a nieumiejętnością i brakiem rozważań użytkowników. Są to uszkodzenia mechaniczne, polegające na spalaniu czy połamaniu mechanizmów wewnętrznych, a powodem tego jest niedopasowanie tych elementów do warunków zewnętrznych, w jakich będą one pracować - nacisk elementów montażowych, zjawiska naturalne itd. Jednym z założeń projektu i kolejnym aspektem motywującym do wykonania opisywanej pracy jest uniknięcie wyżej opisanej sytuacji.

Kompletne źródła programów, napisane na potrzeby poniższej pracy, znajdują się na nośniku załączonym do pracy.

Autor pragnie zaprosić do czytania i refleksji nad opisywanymi problemami.

ROZDZIAŁ 1

Ogólne przedstawienie problemu

1.1. Budowa ramienia robota - założenia

W założeniach omawianego projektu, budowane ramię ma zostać wykorzystane do poruszania się obiektu, czyli do bazy urządzenia zostanie przymocowanych kilka sztuk prezentowanego ramienia oraz zostanie zaimplementowany algorytm synchronizujący ramiona. Projektowane ramię można porównać do odnóża pająka lub ludzkiej ręki bez dłoni - staw zawiasowy pozwalający na ruch przymocowanego dystansu w jednej płaszczyźnie oraz staw kulisty pozwalający na ruch całej konstrukcji w dwóch płaszczyznach.

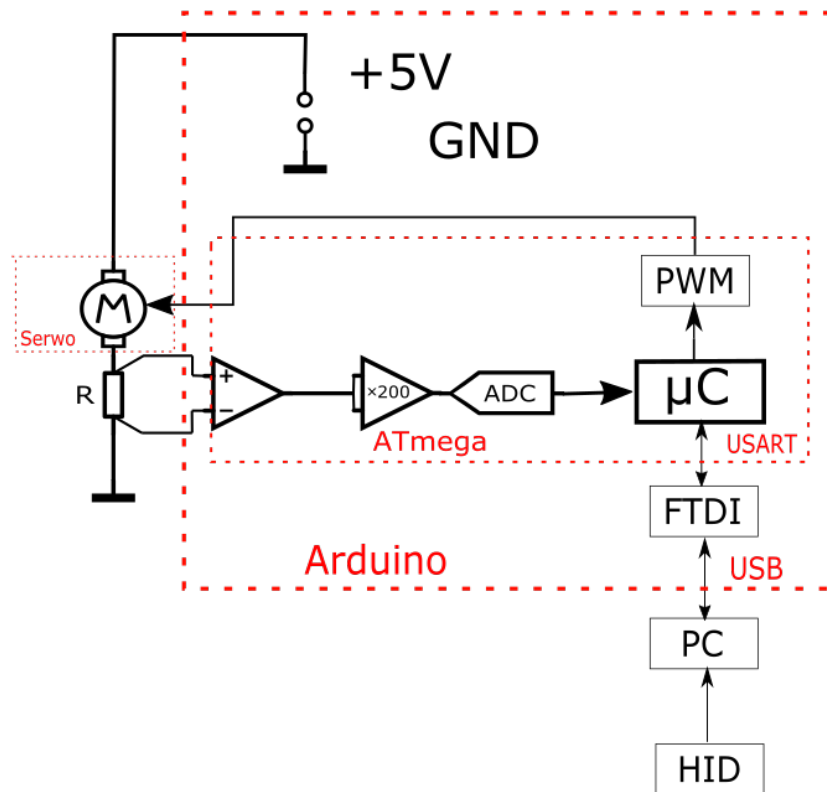
Kolejnym założeniem jest możliwość sterowania ramieniem za pomocą aplikacji komputerowej wykorzystując wybrany kontroler HID (mysz lub joystick często wykorzystywany w grach) oraz odporność na przeciążenia spowodowane napotkanymi przeszkodami.

1.2. Dobór podzespołów i technologii

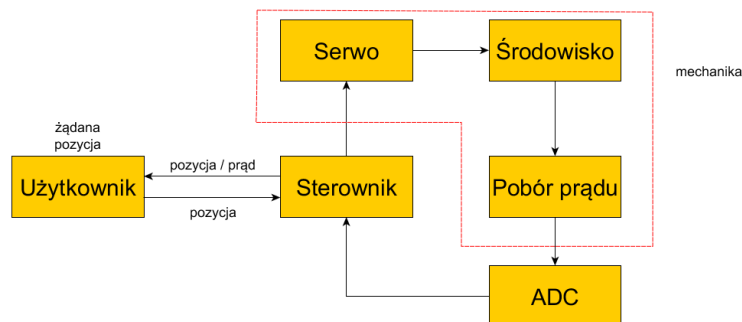
Do budowy stawów, o których mowa w rozdziale 1.1 wykorzystano serwa modelarskie SG90 - jedne z najbardziej popularnych i łatwo dostępnych serw na rynku, które charakteryzują się również niskimi cenami. Pośrednikiem pomiędzy serwami a komputerem, który zapewni odpowiednią kontrolę serw oraz transmisję szeregową będzie płytki prototypowa Arduino Mega, oparta o mikrokontroler ATmega 2560. Charakteryzuje się on zegarem o taktowaniu 16 MHz, 256 kB pamięci Flash oraz 8kB pamięci RAM. Każda płytka z rodziny Arduino posiada 10-cio bitowe przetworniki analogowo-cyfrowe, jednak tylko mikrokontroler w Arduino Mega posiada wbudowane wzmacniacze różnicowe pozwalające na różnicowy pomiar napięć, co jest podstawą uzyskania sygnału zwrotnego z serw i zabezpieczenia ich przed spalaniem. Opisywana płytka prototypowa posiada złącze USB typu B, które pozwala na podłączenie jej do komputera, w celu zaprogramowania oraz zapewnienia transmisji szeregową, dzięki czemu możemy w czasie rzeczywistym przysyłać instrukcje sterujące z komputera i informacje o przeciążeniach do komputera.

Za sterowanie serwami oraz analizę i prezentację informacji o przeciążeniach odpowiada program komputerowy napisany w języku C#. Jest to obiektowy język programowania zaprojektowany dla firmy Microsoft. Programy napisane w tym języku są kompilowane do języka CIL (Common

Intermediate Language) - jest to kod pośredniczący, który do wykonania wymaga środowiska uruchomieniowego np. .NET Framework.



RYSUNEK 1. Schemat opracowanego układu umożliwiającego pomiary prądów serwów.



RYSUNEK 2. Schemat blokowy prezentujący działanie układu.

Opis wybranych technologii

2.1. Zasada działania i specyfikacja serw

Serwa, a właściwie serwomechanizmy, są podstawowymi elementami wykorzystywanymi w modelarstwie. Dzięki nim możliwe są ruchy różnego typu części składowych modelu pozwalających na sterowanie, na przykład serwa realizują wychylenia osi w modelach pojazdów, co umożliwia ich skręcanie. Podstawowymi elementami, z których składa się serwo, są:

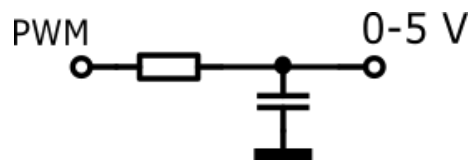
- Silnik napędowy sterowany prądem stałym
- Potencjometr (do pomiaru położenia przekładni)
- Wewnętrzny układ, który steruje pracą silnika
- Przekładnia zębata, pozwalająca uzyskać moment obrotowy

Do sterowania serwami wykorzystywany jest sygnał PWM (Pulse-Width Modulation) - jest to metoda regulacji prądu o stałej amplitudzie i częstotliwości, natomiast zmiane ulega wypełnienie. Częstotliwość tego sygnału w serwach analogowych, jakim jest SG90, to ok. 50 Hz. W serwach cyfrowych jest to 300 Hz i wpływa to na ich większą precyzję. Dzięki tranzystorom wbudowanym w mikrokontroler ATmega oraz funkcjom wbudowanym w Arduino wytworzenie takiego sygnału jest bardzo proste - inicjowany jest wartością jednobajtową czyli z zakresu 0-255 odpowiadające brakowi i pełnemu wypełnieniu (100%) przebiegu. Wykorzystując podane wyżej dane łatwo będzie zrozumieć poniższy przykład: Przy częstotliwości 50 Hz wykonywanych jest 50 zmian sygnału w ciągu sekundy, czas trwania sygnału po zmianie to około 20 ms. Sygnał jest inicjowany wartością 128 - około połowy z dostępnego zakresu. W rezultacie na wyjściu zostaje otrzymany cykliczny sygnał, gdzie na przemian w odcinkach 20 ms występuje 5 V i 0 V. Średnio jest to odpowiednik 50% dostępnego zakresu napięć.



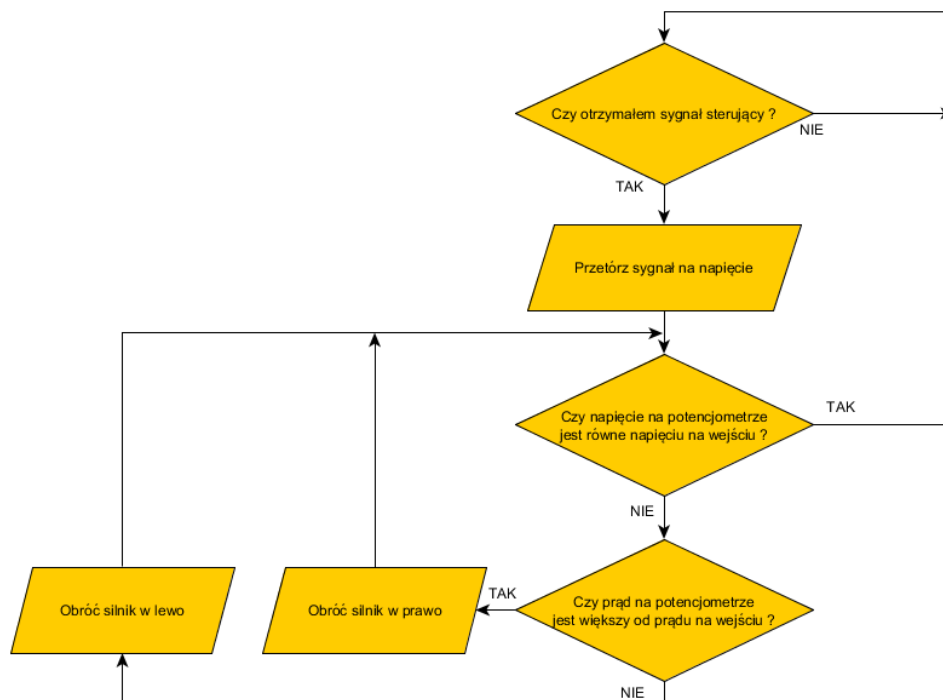
RYSUNEK 1. Wykres prezentujący sygnał PWM - napięcie od czasu.

Dopiero w serwie, za pomocą filtra dolnoprzepustowego, sygnał ten jest uśredniany na napięcie z zakresu 0-5V odpowiadające procentowemu wypełnieniu przebiegu.



RYSUNEK 2. Schemat filtra dolnoprzepustowego.

Do przekładni przymocowany jest potencjometr, który obraca się wraz z jej obrotem. Zmiana rezystancji jest informacją o aktualnym położeniu serwa. W oparciu o dzielnik napięcia powstaje sygnał napięciowy, który trafia do komparatora, gdzie jest porównywany z podanym napięciem wzorcowym (sygnałem sterującym PWM). Wytworzony w komparatorze sygnał to informacja o różnicy pomiędzy aktualną pozycją silnika, a pozycją zadaną. Ta informacja trafia do korektora, a jego zadaniem jest obrócenie silnika tak, aby zmniejszyć różnicę w pozycjach, aż do uzyskania pożądanego efektu.



RYSUNEK 3. Schemat blokowy opisujący działanie serwa.

Serwomechanizmy dzielimy na grupy, a podział ten jest realizowany ze względu na wagę oraz materiały, z których jest wykonane serwo. W opisywanym projekcie autor wykorzystał serwa SG90 firmy Tower Pro. Jest to serwo z grupy micro, czyli o wadze do 10 gram oraz o plastikowych przekładniach zębatych. Inne grupy serw to:

- Mini - o wadze około 25 g oraz metalowych lub karbonowych przekładniach, a tym samym generujące większy moment siły
- Standard - duże serwa o wadze około 50 g, występują w różnych odmianach, są bardzo silne

Dokładna specyfikacja wykorzystywanego serwa Tower Pro SG90 :

- Zasilanie: 4.8 V - 6.0 V
- Zakres ruchu: 180°
- Grupa: micro
- Moment 1.8 kg * cm
- Prędkość 0.10 s /60st
- Rodzaj: analogowe
- Wymiary 22x12x27 mm (dł x szer x wys)
- Masa: 9 g

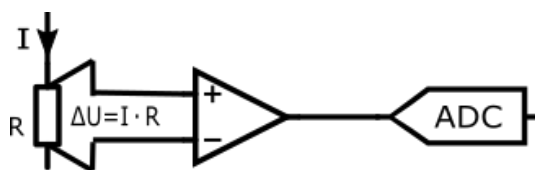


RYSUNEK 4. Serwo Tower Pro SG90

2.2. Pomiar różnicowy, przewaga nad innymi rozwiązaniami

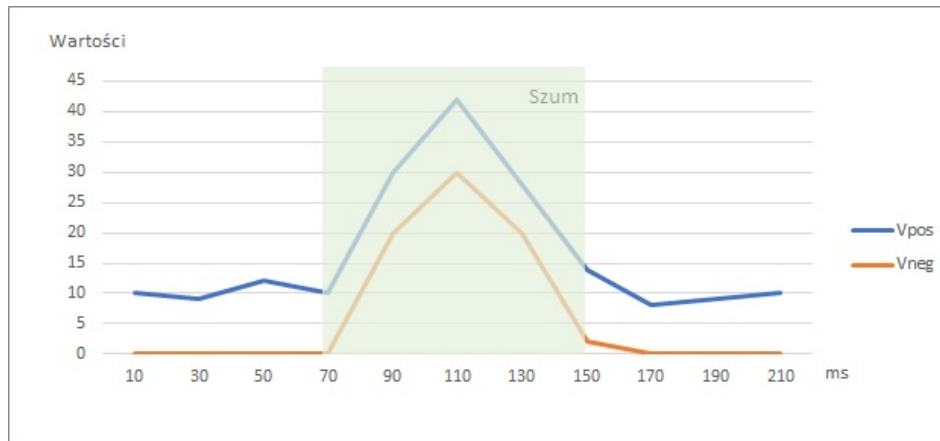
W podrozdziale 2.1 została opisana zasada ustalania i zmian pozycji serwa, która polega na porównywaniu napięć. Są to napięcia odpowiadające pozycji serwa. Niestety samo to rozwiązanie ma pewną wadę - jeśli serwo nie znajdzie się na zadanej pozycji korektor będzie usilnie chciał odpowiednio ustawić silnik. Takie działanie może prowadzić do spalenia silnika lub uszkodzenia przekładni zębatych. Założeniem projektu jest uniknięcie takich sytuacji. Wybrany rozwiązaniem tego problemu jest zastosowanie rezystora bocznikującego w obwodzie zasilania serwa oraz wykonywanie różnicowego pomiaru napięcia, które na nim panuje, a w konsekwencji monitorowanie poboru prądu przez serwo.

Różnicowy pomiar napięcia polega na podłączeniu dwóch bezpośrednich końców badanego elementu do wzmacniacza operacyjnego, a dopiero powstałe tam napięcie na wyjściu trafia bezpośrednio do przetwornika ADC (Analog->Digital Converter).



RYSUNEK 5. Schemat połączenia badanego elementu z przetwornikiem.

Zaletą tego rozwiązania jest odporność na ewentualne szумы, ponieważ przewody połączeniowe są blisko siebie, są tej samej jakości i długości, a więc posiadają ten sam opór oraz zewnętrzne oddziaływania wpływają tak samo na oba przewody, czyli różnica napięć w nich panujących będzie identyczna niezależnie od zaburzeń zewnętrznych.



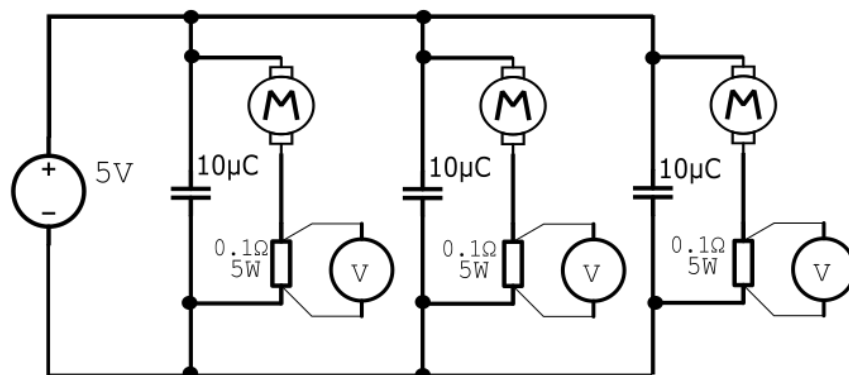
RYSUNEK 6. Wykres prezentujący wyniki pomiaru różnicowego, kiedy występują szumy.

Innym możliwym rozwiązaniem problemu jest rozebranie serwa i podłączenie się do sygnału z potencjometru lub komparatora, odpowiednia analiza w mikrokontrolerze i interakcja. Jednak w tym przypadku przeciążenie można mierzyć tylko na podstawie pomiaru czasu, w jakim serwo się przemieszcza do zadanej pozycji, a logicznym jest, że jeśli trwa to zbyt długo to oznacza, że silnik jest w jakiś sposób blokowany. To rozwiązanie, z pozoru łatwiejsze, ma jednak pewne wady - w pomiarze musi zostać uwzględniona ustalona prędkość serwa, co dodatkowo skomplikuje algorytm. Serwa przemieszczają się dość szybko, a przetworniki ADC wbudowane w mikrokontroler mają ograniczoną szybkość, zwłaszcza wtedy, kiedy ten sam mikrokontroler musi zapewnić również transmisję szeregową i sterowanie serwow. Ponadto w tym rozwiązaniu nie jest zapewniona odporność na szumy, dlatego wybrane do opisywanego projektu rozwiązanie polegające na różnicowym pomiarze napięcia wydaje się być jednym z najlepszych.

2.3. Przygotowany rezystor bocznikujący - dobór parametrów części

Jak opisano w rozdziale 2.2 pomiar różnicowy wykonywany jest na elemencie elektronicznym - w tym przypadku jest to rezystor bocznikujący, czyli taki o bardzo małej rezystancji i dużej precyzji przeznaczony do pomiaru dużych prądów oraz nie wprowadzający znaczących strat w układzie. Takie rezystory wykorzystywane są we wszelkiego rodzaju miernikach natężenia. [1] Na potrzeby projektu powstała płytką z prostym układem umożliwiającym opisywane pomiary. Zastosowano rezystor bocznikujący o oporze 0.1Ω , który został podłączony szeregowo z silnikiem. W trakcie wykonywania ruchów przez serwo, napięcie na rezystorze rośnie. Dodatkowo w układzie

zastosowano kondensatory $10\ \mu\text{F}$ 63V podłączone równolegle między napięciem a masą, aby zapewnić odporność na szumy, iskrzenia silnika, ponieważ są one elementami indukcyjnymi.



RYSUNEK 7. Schemat przygotowanego układu umożliwiającego pomiary.

Program dla mikrokontrolera

3.1. Środowisko ARDUINO

Arduino jest to elektroniczna platforma z otwartym źródłem, która pozwala na proste programowanie wcześniej przygotowanych mikrokontrolerów. Arduino udostępnia interfejs programowy, który w intuicyjny sposób pozwala wykonywać złożone operacje na rejestrach sterujących ATmegi. Jest to projekt stale rozwijany, posiadający wielu zwolenników, mnóstwo książek do nauki oraz niezliczoną ilość wciąż powstających modułów poszerzających funkcjonalność płytek prototypowych. Na płytce znajduje się konwerter USB \leftrightarrow RS-232 (o napięciach TTL), a twórcy udostępniają proste środowisko programistyczne z wbudowanym kompilatorem dzięki czemu szereg operacji wymaganych przy typowym programowaniu procesorów AVR sprowadza się do napisania kodu, podłączenia płytki do portu USB i kliknięcia przycisku 'PROGRAMUJ'.

Prostota obsługi i programowania wynika z wbudowanego bootloadera - programu ładującego skompilowany kod użytkownika do pamięci FLASH mikrokontrolera [2]



RYSUNEK 1. Logo projektu ARDUINO.

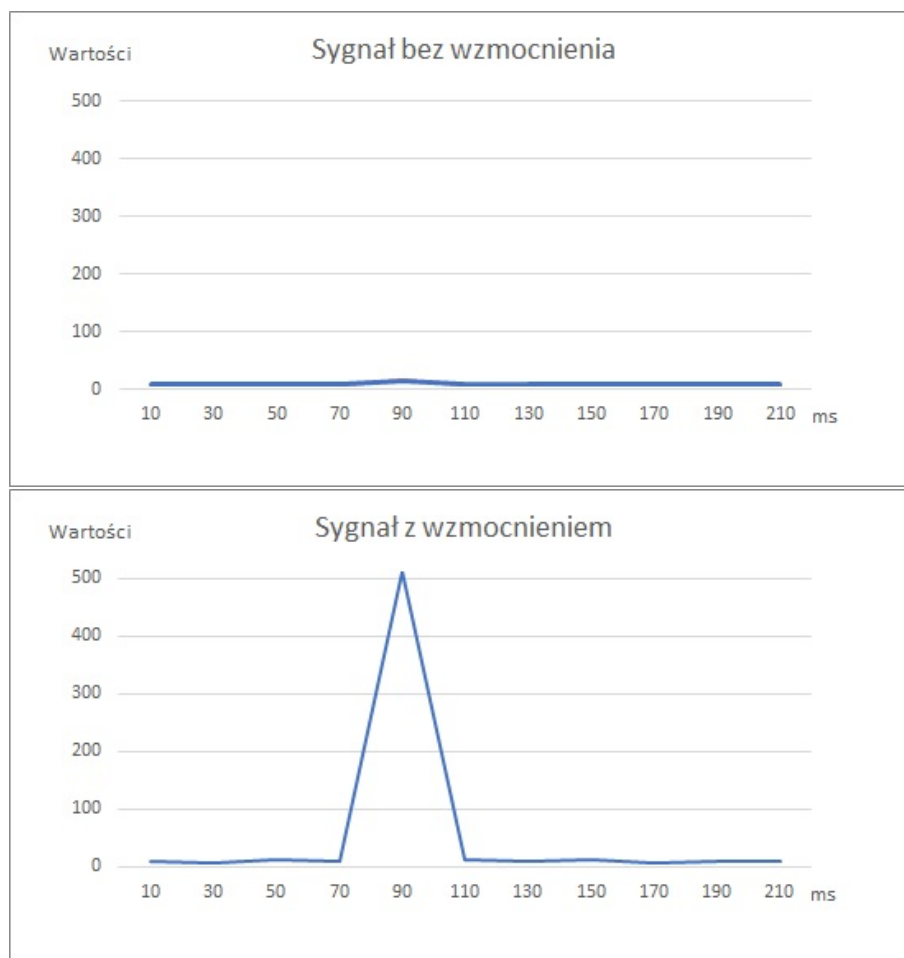
3.2. ATmega 2560 - wzmacniacze różnicowe

Spośród płytek z rodziny Arduino (m.in. Uno, Leonardo, Yum, Nano, Micro, Mega) tylko model Mega posiada mikrokontroler spełniający wymagania projektu. Jest to mikrokontroler ATmega 2650 posiadający wzmacniacze różnicowe. Przetworniki ADC wbudowane w mikrokontroler są 10-cio bitowe co oznacza, że zakres otrzymywanych, zdigitalizowanych wartości napięcia jest z przedziału 0-1023, a mierzone napięcie trafiające do przetwornika może być z zakresu 0V - 5V, czyli uzyskany pomiar ma dokładność

4.88mV, a w przypadku pomiaru różnicowego jest to 9.17mV, ponieważ zakres wartości z przetwornika to $[-512, +511]$. Opór opisanego w podrozdziale 2.3 rezystora bocznikującego jest bardzo mały, a to oznacza, że spadek napięcia mierzony na nim również jest niewielki, czyli rozbieżność wartości określająca czy serwo jest w trakcie ruchu, czy w spoczynku również jest niewielka i musi być wzmocniona tak, aby odpowiadała zakresowi dynamicznemu przetwornika ADC. Tu z pomocą przychodzi wspomniany wzmacniacz różnicowy, który dodatkowo ma możliwość wzmocnienia sygnału 10 lub 200 razy co oznacza, że powstała różnica jest wzmocniana i dopiero ta wartość trafia do 10-bitowego przetwornika. Dokładność pomiaru, w przypadku wzmocnienia $\times 200$, wzrasta do $24 \mu V$ w przedziale 0-25 mV.

Przykład wyliczenia ΔU dla rezystancji opornika $R = 0.1\Omega$ i prądu $I = 0.005A$:

$$\Delta U = R \cdot I = 0.1\Omega \cdot 0.005A = 0.0005V = 0.5mV$$



RYSUNEK 2. Wykresy prezentujące siłę sygnału ze wzmocnieniem i bez.

Wyżej opisana operacja powoduje diametralne zwiększenie rozbieżności między wartościami prądów w różnych stanach pracy serwomechanizmu.

3.3. Transmisja szeregową ARDUINO <=> PC oraz przyjęty protokół danych

Transmisja szeregową polega na szeregowym wysyłaniu i odbieraniu symboli, czyli jeden po drugim. W Arduino odbywa się to za pomocą wbudowanego portu USB. Zainicjowany w programie BaudRate (liczba symboli na sekundę) jest równa ilości bitów przesyłanych na sekundę, ponieważ standard USB 2.0 posługuje się tylko dwoma stanami 0 i 1 dla odpowiednich napięć. Została ustawiona jedna z domyślnych przepustowości i wynosi 57600 bps (bod per second).

W celu zapewnienia porozumienia między urządzeniami, została opracowana pewna konwencja przesyłania informacji. Przesyłane i odbierane są ciągi znaków, które kolejno są analizowane przez programy na urządzeniach. Informacje są przesyłane tylko podczas wykonywania interakcji tak, aby nie zapełnić buforów w płytce, ponieważ mikrokontroler ma ograniczone zasoby. W momencie wystąpienia przeciążenia Arduino wysyła sekwencję znaków: $D_n : val$, gdzie n jest numerem serwa, a val jest wartością odczytaną za pomocą różnicowego pomiaru na tym właśnie serwie. Arduino odbiera również sygnały sterujące z komputera w postaci: $S_n : pos$, gdzie n jest numerem wybranego serwa, a pos jest pozycją w jakiej serwo ma się ustawić i ze względu na ograniczenia, w postaci plastikowych elementów montażowych wartość ta jest z zakresu 10-140, które oznaczają kąt w stopniach.

3.4. Brak pomiaru różnicowego w interfejsie programowym ARDUINO - działania na rejestrach mikrokontrolera

W związku z występowaniem wzmacniaczy różnicowych tylko w modelu Mega, Arduino nie udostępnia w interfejsie funkcji do łatwego wykonywania takich pomiarów. Rozwiązaniem jest napisanie własnych funkcji, które bezpośrednio ustawiają odpowiednie bity w rejestrach mikrokontrolera, o których można przeczytać w dokumentacji ATmegi 2560 [3]. Należy ustawić porty, do których podłączone są elementy wymagające pomiaru, wartości referencyjne, z którymi sygnał będzie porównywany - tu został ustawiony na port sygnał z portu zewnętrznego, do którego nic nie jest podłączone. Bity w opisywanych rejestrach są też zmieniane przez mikrokontroler, co na przykład oznacza ukończony pomiar, gotowy do odczytu. Zamieszczony niżej fragment kodu zawiera funkcje do inicjalizacji i wykonywania różnicowych pomiarów napięcia:

```
1 void initdiff(){ // definition of initialization function
2   ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
3   ADMUX = (1<<REFS0)|(0<<REFS1); // set reference voltage
4   ADMUX |= (1<<MUX3)|(1<<MUX1)|(1<<MUX0); // set ports to read voltage (
      particular)
5 }
6 void adc1() {
7   ADCSRB = (0<<MUX5); // set read port on A0 and A1
```

```

8  }
9  void adc2() {
10     ADCSRB = (1<<MUX5);    // set read port on A8 and A9
11 }
12 uint16_t do_adc() {        // definition of read function
13     ADCSRA |= (1<<ADSC); //Start an ADC conversion by setting the ADSC bit.
14     while(!(ADCSRA & (1<<ADIF))); //Wait for the conversion to finish. The
        ADC signals that it's finished
15     ADCSRA |= (1<<ADIF);
16     return ADC; //Return the ADC result from the ADC register.
17 };

```


Aplikacja komputerowa GUI

4.1. Wybór języka programowania

Praktycznie każdy z powszechnie znanych języków oprogramowania udostępnia biblioteki lub funkcje wbudowane pozwalające na inicjowanie i komunikację z urządzeniami za pomocą transmisji szeregowej z wykorzystaniem portu szeregowego (emulowanego przez sterownik FTDI). Mnóstwo języków umożliwia też proste przygotowanie graficznej reprezentacji danych, dlatego pojawia się oczywiste logiczne pytanie: Dlaczego .NET C#?

Składnia programu pisanego w środowisku Arduino do złudzenia przypomina składnię C++ - jest to właściwie C++ z dodanymi funkcjami do obsługi mikrokontrolera - dlatego dla zachowania jednolitości i prostego zrozumienia kodu naturalnym byłoby wybranie właśnie tego języka dla napisania aplikacji komputerowej. Ponadto C# posiada wiele wbudowanych modułów do transmisji szeregowej, rysowania wykresów i prostego tworzenia GUI, co znacznie ułatwia implementację [4].



RYSUNEK 1. Logo Microsoft Visual C#.

4.2. Struktury użyte w aplikacji oraz wykrywanie przeciążeń serwomechanizmów

C# jest językiem wysokopoziomowym, obiektowym więc każda użyta struktura jest klasą lub należy do klasy. Napisany program komputerowy zbiera informacje o aktualnych przeciążeniach serw oraz poddaje je analizie. Otrzymane sygnały są często skrajne - dla braku przeciążeń pojawiające się wartości poboru prądu są mniejsze od 10-ciu jednostek ADC, natomiast w momencie pojawienia się samego ruchu serwa wartości te przekraczają 500 [a.u.]. Sposobem wykrycia przeszkody i blokady serwa, a nie samego ruchu, jest zebranie informacji z pewnego odcinka czasu i obliczenie kroczącej średniej arytmetycznej odczytów, ponieważ po wykonaniu ruchu otrzymywane wartości ponownie maleją. Do zmagazynowania informacji nadchodzących z

mikrokontrolera wykorzystano zmodyfikowaną kolejkę FIFO - domyślna kolejka zaimplementowana w C# jest oparta o listę wskaźnikową, czyli teoretycznie o nieskończonej pojemności i braku dostępu do konkretnej liczby elementów wstecz. Spersonalizowana kolejka posiada konkretnie zdefiniowaną wielkość, co oznacza, że w przypadku przepełnienia kolejki, kolejny element nie zostanie bezpośrednio dodany - wcześniej zostanie skasowany najstarszy element. Średnią arytmetyczną ostatnich wartości można obliczyć dodając do wcześniej obliczonej średniej wszystkich przechowywanych elementów iloraz $\frac{\text{New sample} - \text{Oldest sample}}{\text{Deep of queue}}$ - jest to tzw. średnia krocząca. Wielkość ta, jest zależna od szybkości napływania danych z mikrokontrolera oraz zamierzonej ilości ostatnich sekund, z których jest obliczana wspomniana średnia.

```

1 public void Add(int val) // methods to adding data to queue
2 {
3     int small_temp = 0;
4     while (smallQ.Count > small_size)
5     {
6         smallQ.Dequeue();
7     }
8     if (smallQ.Count == small_size) small_temp = (int)smallQ.Dequeue();
9     smallQ.Enqueue(val);
10    small_average = small_average + (val - small_temp) / smallQ.Count;
11 }

```

Jak opisano w podrozdziale 3.3 informacje o przeciążeniach są wysyłane tylko w momencie ich występowania, czyli wysokich wartości napięcia zmierzonego na boczniku, natomiast informacje do kolejki w aplikacji komputerowej muszą trafiać stale, dlatego mikrokontroler wysyła też jedną nieprzeciążoną wartość, a wątek w programie periodycznie ją kopiuje i wstawia do kolejki, a tym samym obliczana średnia spada zgodnie z założeniem, nie nadużywając buforów mikrokontrolera.

4.3. Inicjalizacja połączenia i sterowanie serwami - ciąg dalszy przyjętego protokołu danych

W programie wykorzystano domyślny moduł do komunikacji szeregowej. Funkcje w nim zawarte pozwalają w łatwy sposób otworzyć port komunikacyjny, zainicjować połączenie z odpowiednimi parametrami oraz odczytywać i wysyłać dane. Jako że to aplikacja sprawdza występowanie przeciążeń na serwach, nie mogą one zostać podłączone przez układ mikrokontrolera jeśli nie zostanie zainicjowanie połączenie z programem na PC. Zostało to zrealizowane w następujący sposób:

1. Mikrokontroler po uruchomieniu wysyła periodycznie specjalne hasło *IamGOD*, które zostało podane w implementacji.
2. To samo hasło zostało podane w kodzie programu komputerowego - program po otwarciu portu sprawdza, czy otrzymywane hasło zgadza się z podanym w kodzie.

- Po zweryfikowaniu urządzenia program wysyła do mikrokontrolera swój kod (ten sam), a mikrokontroler porównuje hasło wbudowane i otrzymane.
- Po zweryfikowaniu kodu, mikrokontroler uruchamia sterowanie serwami.

Po zamknięciu aplikacji oraz wykryciu przeciążenia program wysyła do mikrokontrolera polecenie inicjujące odłączenie serw *exit* oraz *break*, aby uniknąć ich zniszczenia. Gdy średnia prądów mierzonych na boczniku spadnie poniżej pewnego, zadanego poziomu zostaje wysłane polecenie *continue*, które wymusza ponowne podłączenie serw.

4.4. Prezentacja danych

Aplikacja komputerowa została napisana wykorzystując środowisko Visual Studio 2015. Jedną z klas wbudowanych w C# oraz narzędziem oferowanym w "Visualowym ToolBoxie" jest Chart, czyli z języka angielskiego "Wykres". Udostępnia on intuicyjny interfejs, w którym wystarczy wybrać typ wykresu, linii łączącej jego punkty (np. spline), kolorystykę, legendę, a w trakcie działania programu dodawane są wartości do aktualnego czasu i wpisywane w wykres. Dodatkowo za pomocą wbudowanej w klasę funkcji wykres jest przeskalowywany względem osi Y dla lepszej czytelności w przypadku dłuższego pojawiania się podobnych wartości.



RYSUNEK 2. Zrzut interfejsu programu. Na osi X oznaczane są ostatnie próbki; na osi Y oznaczona jest wartość przeciążenia w jednostkach ADC.

Interwał czasowy z jakim wprowadzane są dane do wykresu musi być równy interwałowi z jakim przychodzą dane z mikrokontrolera, a właściwie z jakim dane wpływają do kolejki opisanej w podrozdziale 5.2 tak, aby wprowadzane wartości nie dublowały się oraz nie zostały pominięte.

```

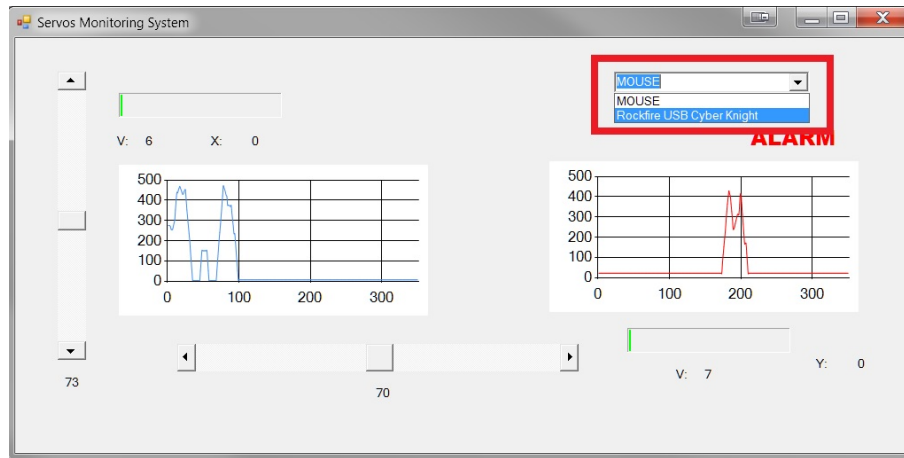
1  const int SecInChart = 7;           // last seconds in charts
2  const int NewValInterval = 20;      // Time interval in refreshing charts
3  int ValPerSec;                      // is calculated based on const above
4  ValPerSec = 1000 / NewValInterval;

5
6  private void SetChart(Chart ch, String series, float val)
7  {
8      if (ch.InvokeRequired)
9      {
10         SetChartPoint d = new SetChartPoint(SetChart); //need delegate
11         //function
12         this.Invoke(d, new object[] { ch,series,val });
13     }
14     else
15     {
16         if (ch.Series[series].Points.Count >= SecInChart*ValPerSec)
17             // delete the oldest values from chart if is full
18             ch.Series[series].Points.RemoveAt(ch.Series[series].Points.Count
19             - SecInChart * ValPerSec);
20         ch.Series[series].Points.AddY(val);           // add new value
21         ch.ChartAreas[0].RecalculateAxesScale();      // recalculate for
22                                                         resizing chart
23     }
24 }

```

4.5. Wykorzystanie kontrolera gier do sterowania ramieniem

Do aplikacji zaimplementowano dodatkową funkcjonalność jaką jest możliwość sterowania ramieniem robota za pomocą kontrolera do gier. W tym celu wykorzystano otwarte źródłowe API (Application Programming Interface) do DirectX o nazwie SlimDX, które jest przeznaczone głównie do tworzenia gier pod platformą .NET, ale w tym przypadku zostało użyte do oprogramowania kontrolerów. Interfejs, który udostępnia SlimDX, jest uniwersalny i implementacja jednego kodu pozwala na używanie różnego typu kontrolerów do gier - joysticków, gamepadów, kierownic itd. Do użycia wystarczy załadować i podłączyć kontroler do komputera wraz z zainstalowanymi sterownikami do urządzenia, uruchomić aplikację i spośród listy dostępnych urządzeń wybrać to interesujące.



RYSUNEK 3. Zrzut interfejsu programu z wyszczególnieniem opcji wyboru urządzenia sterującego.

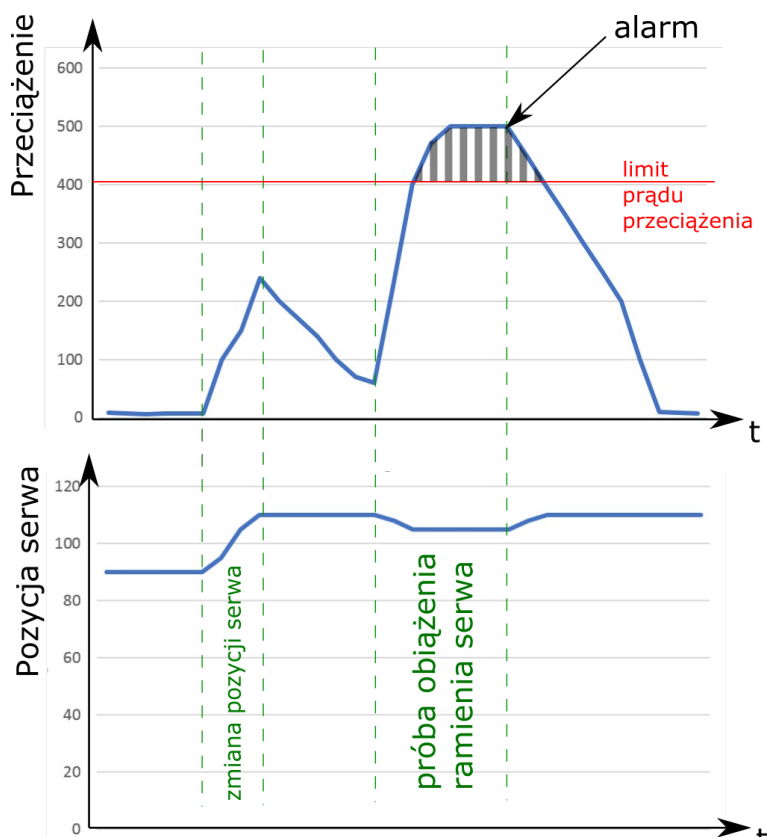
Do prac nad wykorzystaniem kontrolera do gier jako urządzenia sterującego użyto wiekowego, bezprzewodowego Joysticka firmy Rockfire model RF-626u



RYSUNEK 4. Wykorzystany kontroler do gier - RockFire RF-626u.

Podsumowanie oraz przykłady wykorzystania projektu

Realizacja opisywanego projektu przebiegła pomyślnie - opracowano układ i algorytm pozwalający na odpowiednie sterowanie ramieniem robota zabezpieczając przy tym serwa przed przeciążeniami. Program alarmuje użytkownika o zaistniałych, groźnych przeciążeniach - są one rozróżniane od przeciążeń powstających przy zmianie pozycji serwa.



RYSUNEK 1. Wykresu prezentujące pozycję serwów i występujące przeciążenia w czasie.

Technologię wykrywania przeciążeń opisaną w powyższej pracy, przy odpowiednim doborze parametrów części elektronicznych, można zastosować niemal w każdym urządzeniu wymagającym takiej kontroli, dlatego ilość zastosowań jest praktycznie nieskończona. Mogą to być nie tylko serwa, ale również standardowe silniki, siłowniki, pompy itp. Do najbardziej oczywistych zastosowań prezentowanego projektu należy system poruszania się i przesuwania przeszkód w eksploratorach obszarów czy łazikach pozaziemskich - poruszanie się na np. sześciu kończynach podobnych do pajęczych jest dużo bardziej odporne na utrudnienia terenu, niż koła czy gąsienice.



RYSUNEK 2. Przykładowy model robota, w którym można zastosować serwa z pomiarem przeciążeń.

Innym przykładem są modele samolotów, w których układ sterowania lotek często realizowany jest za pomocą serwomechanizmów. Silne opory powietrza mogą zniszczyć mechanizm, a więc za pomocą systemu użytkownik dostanie ostrzeżenie o takiej możliwości i sugestię zaprzestania lotów.



RYSUNEK 3. Model samolotu, wymagającego serw do sterowania.

Prototypy linii montażowych w fabrykach, a właściwie ramion przy nich pracujących to również przykład, w którym wymagane jest kontrolowanie

znaczącej liczby serw i silników. W przypadku awarii firma poniesie ogromne straty produkcyjne, jak również ucierpieć mogą pracownicy obsługujący maszyny - odpowiedni system może temu przeciwdziałać. Identyczna sytuacja jak i konsekwencje mogą się pojawić w przypadku braku kontroli "łyżek" w koparkach.

W pracy wykorzystano informacje i wiedzę z zakresu:

- **elektroniki** (pomiar natężenia prądu z użyciem bocznika, pomiar różnicowy, przetwarzanie ADC)
- **programowania mikrokontrolerów** (dogłębne poznanie funkcji ADC w ATmegach, transmisja szeregową danych z potwierdzeniem, protokoły danych)
- **programowania GUI** (C#, średnie kroczące, HIDy, grafika, obsługa portów szeregowych)
- **mechaniki** (budowa serwomechanizmów, ramion robota itp.)

Podziękowania

Autor, pragnie złożyć ogromne wyrazy szacunku i podziękowania swojego promotora Pana doktora Grzegorza Zuzla, który sprawował pieczę nad poczynaniami w pracy licencjackiej. Twór ten nie byłby też możliwy bez pozostałych pracowników Zakładu Doświadczalnej Fizyki Komputerowej z wyszczególnieniem Pana doktora Krzysztofa Pelczara. Cały ten skład zaangażował się w rozwiązywanie napotkanych problemów i to właśnie dzięki nim prezentowana praca została ukończona. Podziękowania należą się również Garażowi Złożoności, który pomimo licznych opracowywanych projektów, udostępnił opisywany w pracy mikrokontroler należący do garażowego asortymentu.

Przez okres ostatnich trzech lat autorowi było dane osiąść ogromną wiedzę, dlatego dziękuje on wszystkim pracownikom wydziału, a zwłaszcza tym, którzy potrafili rozbudzić w nim pasję do rzeczy, o których wcześniej nie miał pojęcia, takie które posłużą mu w przyszłości i wyróżnią go na rynku pracy. Nie zabrnąłby on do miejsca, w którym aktualnie się znajduje, gdyby nie dobrobyt i odpowiednie warunki do nauki, które zapewnili mu rodzice, dlatego z całego serca dziękuje również im i obiecuje, że zawsze będzie pracował tak, aby byli z niego dumni. Autor jeszcze raz wszystkim ogromnie DZIĘKUJE.

Bibliografia

- [1] B.M. Oliver, J.M. Cage, *Pomiary i przyrządy elektroniczne*, Wydawnictwa Komunikacji i Łączności, Warszawa 1978
- [2] <https://www.arduino.cc/>
- [3] *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V DATASHEET*, Atmel Corporation, 2014
- [4] I. Griffiths, M. Adams, J. Liberty, *C#. Programowanie. Wydanie VI*, Helion, 2012

Załączniki

Nośnik danych dołączony do pracy zawiera:

- Program napisany dla mikrokontrolera Arduino
- Program kontrolujący napisany w języku C#
- Instalatory dla SlimDX