

Politechnika Poznańska
Wydział Automatyki, Robotyki i Elektrotechniki
Systemy mikroprocesorowe

PROJEKT
**STEROWANIE OBROTAMI SILNIKA ZA POMOCĄ
STEROWNIKA PID**

Autor:

WIKTOR ROSIŃSKI
144576
PAWEŁ KWIATKOWSKI
139654

1 Opis działania

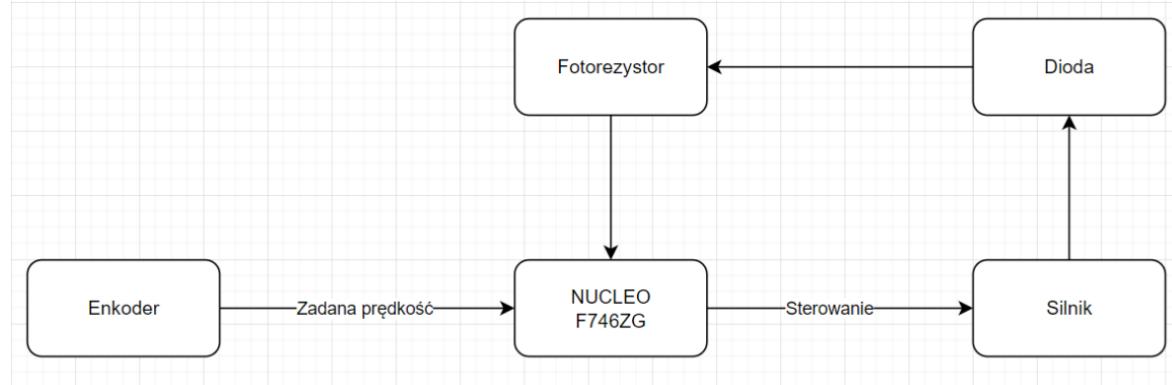
Projekt zakłada sterowanie obrotami silnika po przez sterownik PID realizowanego za pomocą modułu NUCLEO F746ZG.

Wartość prędkości zadanej ustawiamy za pomocą enkodera lub za pośrednictwem programu wykonanego w języku Phyton. Następnie moduł nukleo nastawia sterowanie na silnik by kręcił się zadaną prędkością.

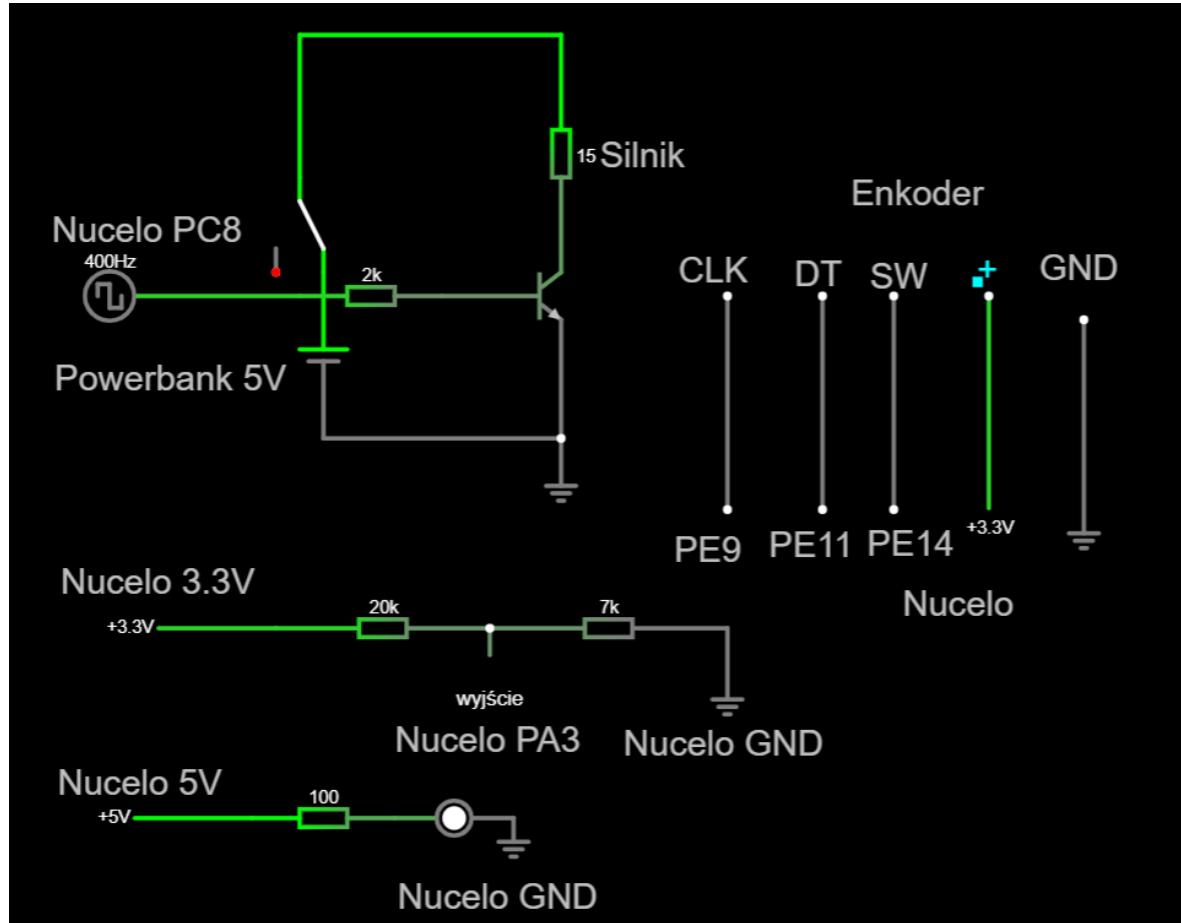
Pomiar prędkości silnika realizowany jest po przez pomiar wzrostów rezystancji na fotorezystorze spowodowany przysłonięciem diody umieszczonej za silnikiem co jeden obrót. Dodaliśmy również funkcjonalność obserwacji pracy silnika na wykresie w czasie rzeczywistym w programie w języku Phyton. Silnik wyposażony jest w śmigło, które skonstruowane jest tak by tylko raz na obrót było w stanie zasłonić diodę. Jest to nasza pętla ujemnego sprzężenia zwrotnego.

2 Schemat górnny projektu

Schemat przedstawia logikę układu i oddziaływanie poszczególnych komponentów na siebie.



3 Schemat elektryczny

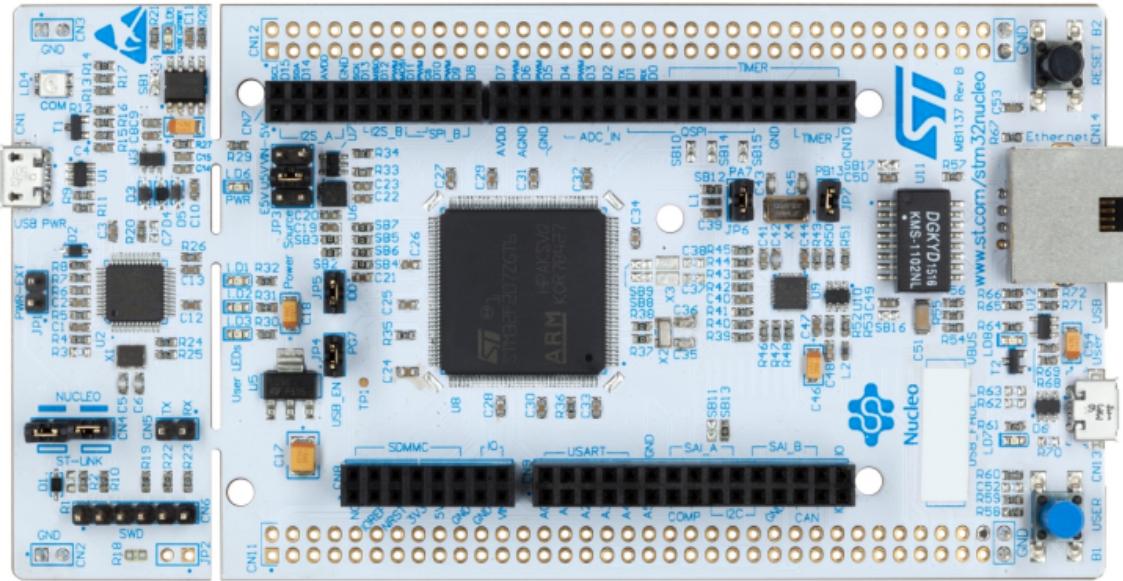


Dodatkowo można pobrać plik txt z naszego [githuba](#) i sprawdzić działanie układu w internetowym programie [falstad](#).

4 Użyte części w projekcie

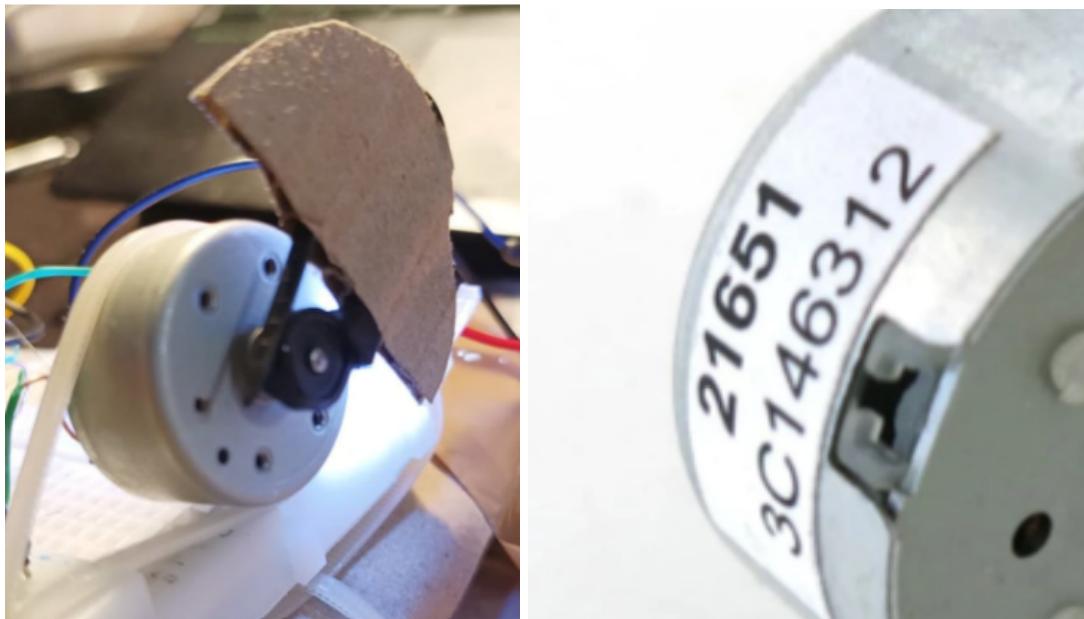
4.1 NUCLEO F746 ZG

Moduł Nucleo jest sterownikiem jak i komunikatorem pomiędzy wszystkimi częściami naszego układu. Za jego pośrednictwem będziemy odbierać sygnały z poszczególnych części jak i nadawać sygnały sterujące. Służyć też będzie nam jako źródło zasilania. W poniższej tabeli zostały wypisane wykorzystywane piny i ich zastosowanie.



Nazwa i numer pinu	zastosowanie
PC8	Podajemy napięcie na bazę tranzystora
3V	zasilanie enkodera i fotorezystora
5V	zasila diodę
GND	Używane do uziemienia oraz diody
PA3	Sczytuje sygnał z fotorezystora
GND	Uziemienie silnika, tranzystora
PE9	Obsługuje wejście CLK enkodera
PE11	Obsługuje wejście DT enkodera
PE14	Obsługuje wejście SW enkodera

4.2 Silnik



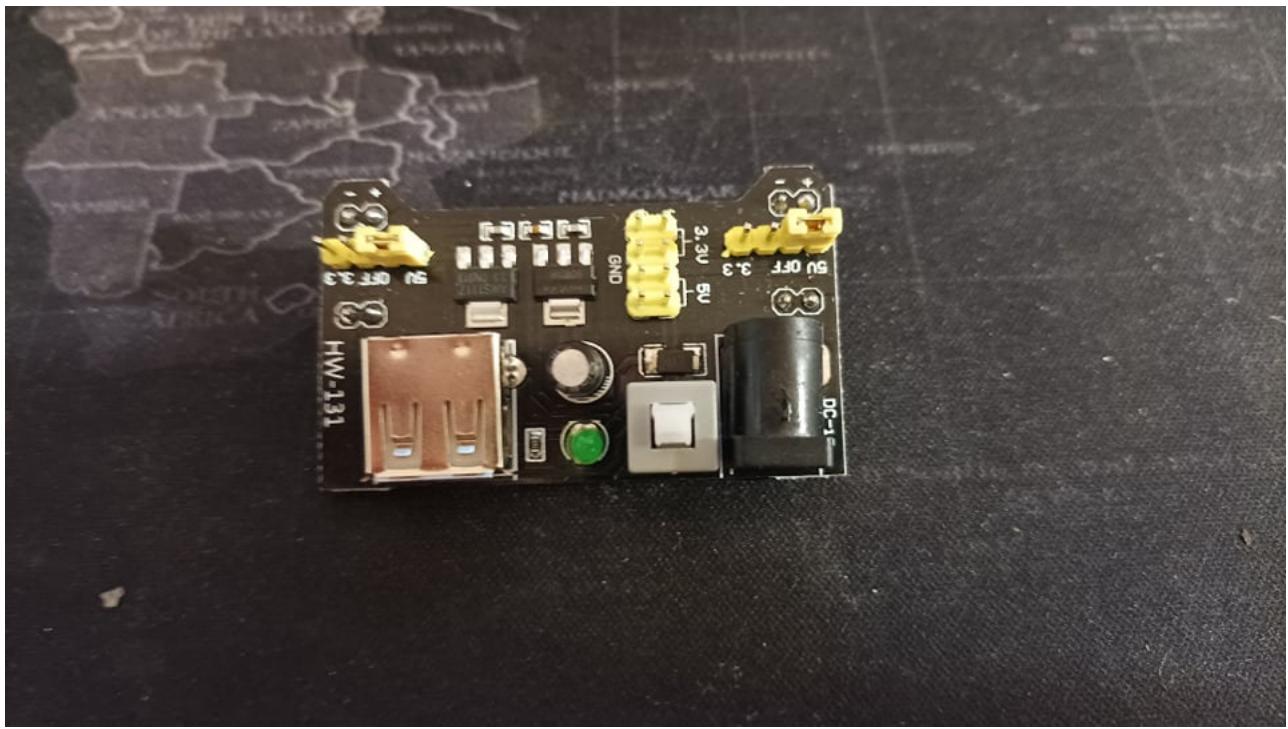
Obiekt regulacji.

Silnik został wymontowany z odtwarzacza CD jego numer seryjny to: 3C146312.

Staraliśmy się znaleźć dokumentacje do niego lub podobny model i jedyne co znaleźliśmy to silnik o numerze seryjnym: rc300-ft-08800. Próbowaliśmy na podstawie dołączonych do niego danych za modelować nasz silnik jednak nie dawało to wspólniernych wyników. Dlatego samodzielnie staraliśmy się wyznaczyć jego wartości i model.

Wartość rezystancji silnika to 15Ω w stanie spoczynku.

4.3 Moduł zasilający do płyt ekranowych MB102



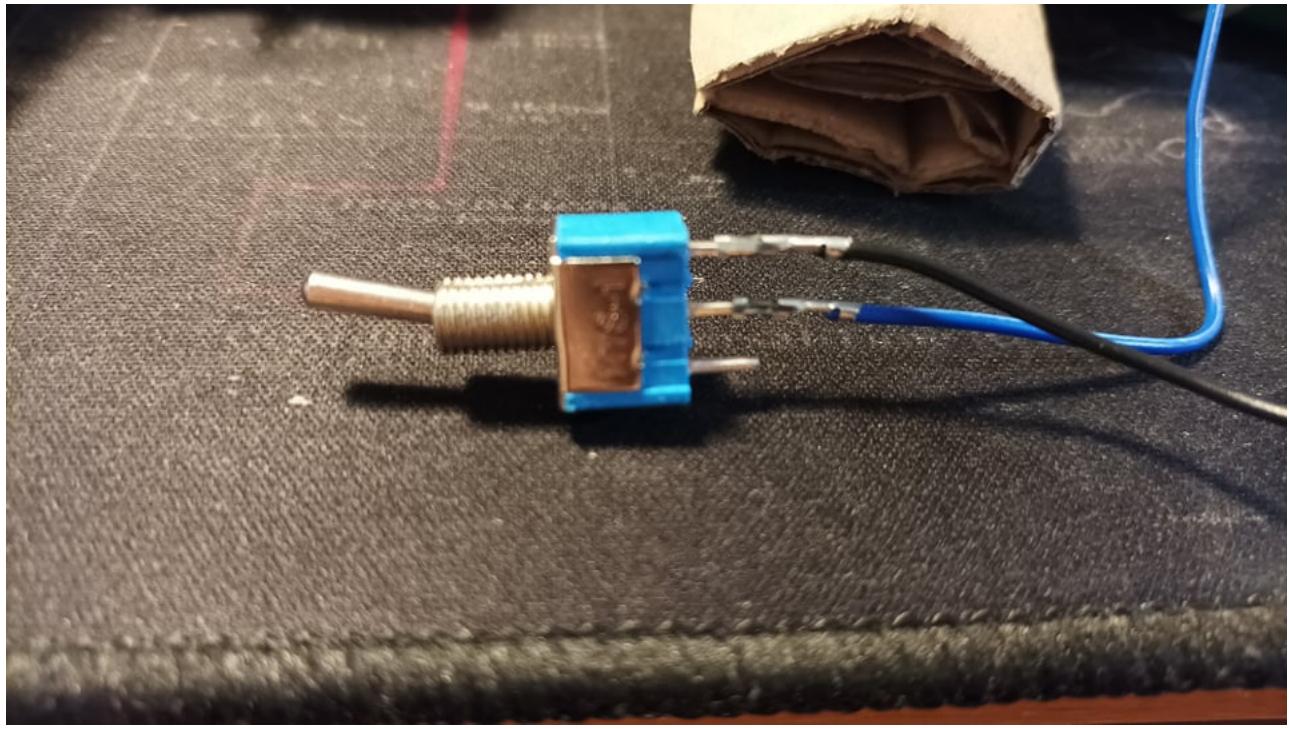
Początkowo silnik zasilany był z baterii 9V jednak z powodu użytkowania z czasem bateria szybko się rozładowywała zdecydowaliśmy się na zamianę baterii na moduł zasilający. Jest on bardziej efektywny i ekonomiczny w przypadku sterowania i ciąglej pracy z układem.

4.4 Enkoder z przyciskiem - Iduino SE055



Enkoderem zadajemy prędkość obrotową silnika.

4.5 Przełącznik



Używany jako włącznik silnika i wyłącznik. Działa jako zabezpieczenie w przypadku niepożądanego stanu silnika można automatycznie wyłączyć dopływ prądu dochodzącego do silnika.
Używaliśmy go również by sprawdzać działanie innych części programu bez uruchamiania silnika.

4.6 Tranzystor S9013

S9013



Używany do sterowania silnikiem.

4.7 Fotorezystor GL5528

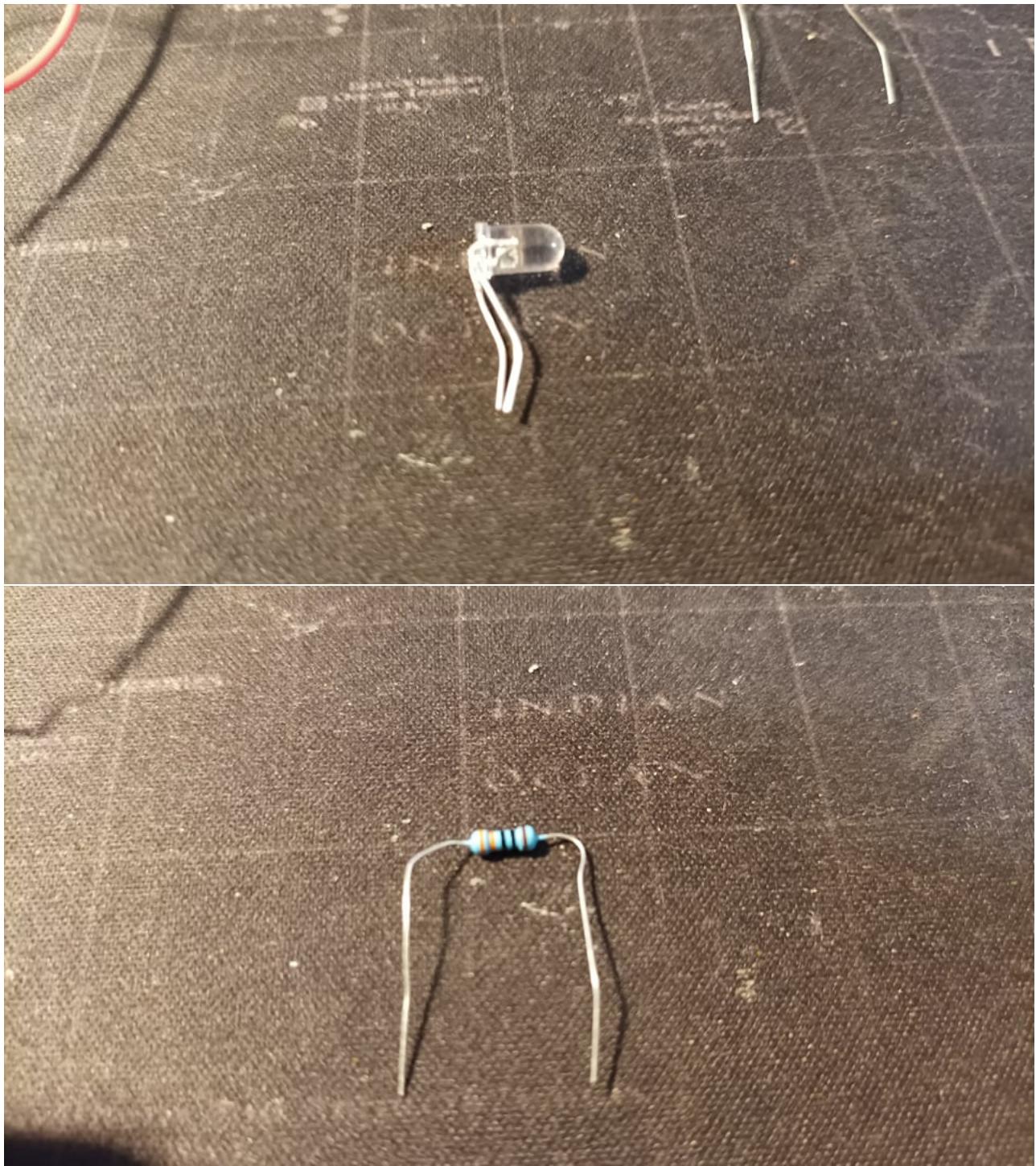


Przy jego pomocy mierzymy zadaną prędkość obrotów. Konkretnie w programie szczytujemy skoki rezystancji w chwili gdy silnik zasłoni diodę śmieglem.

Rozważaliśmy również użycie miernika światła jednak układ działał przez to za wolno z tego względu zdecydowaliśmy się na dzielnik napięcia z fotorezystorem.

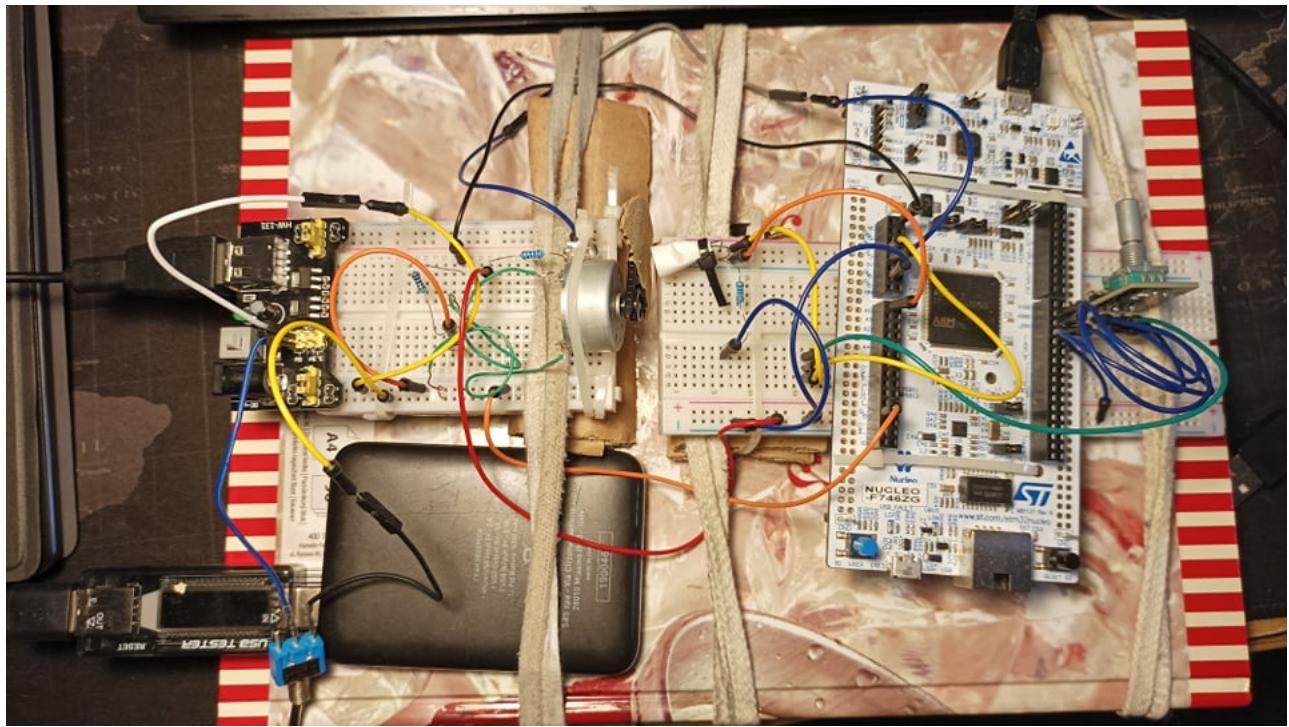
Dodatkowo by zredukować szum pomiarowy skupiliśmy pomiar jasności za pomocą przesłony. Zakres rezystancji możliwej do uzyskania fotorezystorze to od $10\ 000\ \Omega$ do $20\ 000\ \Omega$

4.8 Rezystory oraz Diody



Dioda używana do pomiaru prędkości.
Rezystory ustawiają napięcie na
bazie tranzystorze ($2\ 000\ \Omega$),
dzielnika napięć przy fotorezystorze ($7\ 000\ \Omega$)
oraz prąd na diodzie ($108\ \Omega$).

4.9 Połączony układ



Omówienie układu z prezentacją jego działania można obejrzeć w niniejszym [linku](#).

5 Program

Link do githaba z omawianymi materiałami dostępny jest w tym [linku](#).

W poniższych sekcjach omawiamy koncepcje i zastosowanie napisanego kodu. Wszystkie poniższe opisywane funkcjonalności znajdują się tam w znacznie obszerniejszej wersji i po pobraniu można przetestować ich działanie w praktyce (do czego serdecznie zachęcamy).

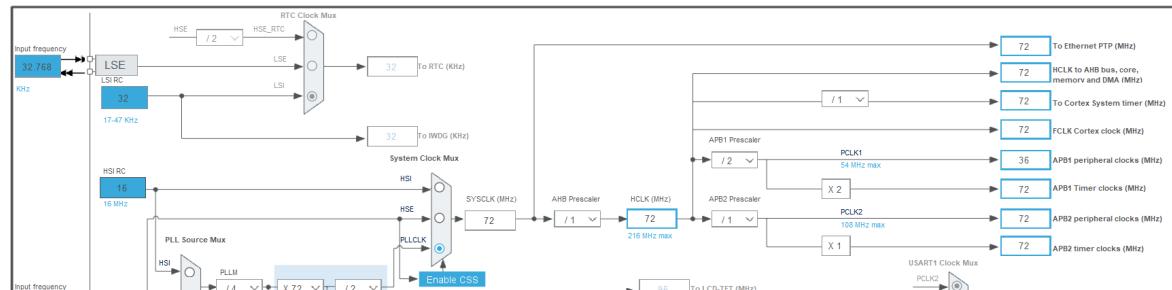
5.1 Program wgrany do modułu NUCLEO

5.1.1 Ustawienia programu STM

The screenshot displays five configuration panels from the STM32CubeMX software:

- TIM4 Mode and Configuration**: Shows Slave Mode Disable, Trigger Source Disable, Clock Source Internal Clock, and Channel1-4 Disable.
- USART3 Mode and Configuration**: Shows Mode Asynchronous, Hardware Flow Control (RS232) Disable, and Hardware Flow Control (RS485) Disable.
- TIM3 Mode and Configuration**: Shows Clock Source Internal Clock, Channel1-4 Disable, and Channel5-6 Disable.
- DAC Mode and Configuration**: Shows OUT1 Configuration checked, OUT2 Configuration and External Trigger disabled.
- TIM14 Mode and Configuration**: Shows Activated checked, Channel1 Enable, and One Pulse Mode disabled.

5.1.2 Nastawy zegarów modułu NUCLEO



5.1.3 Funkcje

```

713 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
714 {
715
716     if (htim == &htim14 )
717     {
718         /* Średnia z 2 ostatnich wyników obrótów*/
719         obrotys=(obroty*10+obroty_last)/2;
720         /* PID*/
721         e=impulsy-obroty;
722         obroty_last=obrotys;
723         obroty=0;
724
725         e_sum=e_sump+(e - e_last);
726         e_sump=e_sum;
727         /*Dodanie wartości stałej potrzebnej aby pokonać bezwładność*/
728         if(impulsy==0)
729         {
730             u_next=0;
731         }
732         else
733         {
734             u_next =465+( kp*e + ki*e_sum*(dt/2) + kd*(e - e_last)/dt);
735         }
736         e_last=e;
737         /*Saturacja */
738         if(u_next<0)
739         {
740             u_next=0;
741         }
742         else if(u_next>2499)
743         {
744             u_next=2499;
745         }
746         /* Filtr FIR*/
747         u_next=(u_next+u_last1+u_last2+u_last3)/4;
748         u_last3=u_last2;
749         u_last2=u_last1;
750         u_last1=u_next;
751         /* Zadanie wypełnienia PWM*/
752         __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, u_next);
753         /*Wysłanie informacji przez USART */
754         sprintf(msg,"%d %d ", obrotys,impulsy);
755         HAL_UART_Transmit(&huart3, (uint8_t*)msg, strlen(msg), 1000);
756     }
757 }
758 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
759 {
760
761     if (huart == &huart3)
762     {
763         /*konwersja*/
764         HAL_UART_Receive_IT(&huart3, key, 2);
765         koka=(key[0]-48)*10+(key[1]-48);
766         __HAL_TIM_SetCounter(&htim1,koka);
767
768     }
769 }
770 }
```

Implementacja sterowania PID zaimplementowana została w funkcji void HAL TIM PeriodElapsedCallback(TIM HandleTypeDef *htim). Jak widać zaimplementowany jest tu klasyczny układ regulacji na podstawie podstawowego równania opisującego regulator PID z filtrem FIR by uzyskać dokładniejsze sterowanie.

Funkcja void HAL UART RxCpltCallback(UART HandleTypeDef *huart) służy do odbierania wartości zadanej z programu Phyton

5.1.4 Sprawdzenie czy wystąpił obrót

```

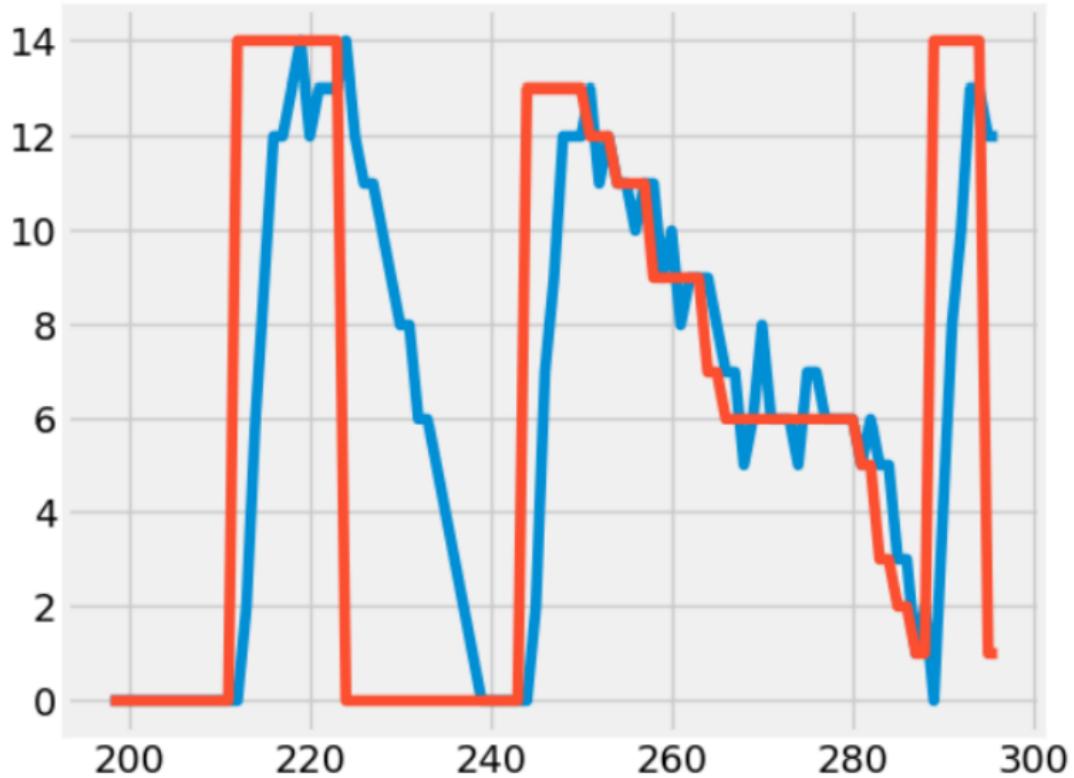
160    while (1)
161    {
162        /* USER CODE END WHILE */
163
164        /* USER CODE BEGIN 3 */
165        /* czytanie wartości z enkodera/
166        impulsy = __HAL_TIM_GET_COUNTER(&htim1);
167        /* czytanie wartości z dzielnika napięcia*/
168        HAL_ADC_Start(&hadc1);
169        HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
170        AdcValue = HAL_ADC_GetValue(&hadc1);
171        HAL_Delay(1);
172        wyniki[i]=AdcValue;
173        /* zobserwowanie obrotu*/
174        if(wyniki[(i-1)%rozmiar_dan+1])!=0
175        {
176            for(i2 = 1; i2 < rozmiar_dan; i2++)
177            {
178                /* zobserwowanie spadku*/
179                if(obr==0){
180                    if((wyniki[abs((i-i2)%rozmiar_dan+1))]-min_proc1)>wyniki[i])
181                    {
182                        okolica=wyniki[abs((i-i2)%rozmiar_dan+1));
183                        obr=1;
184                        break;
185                    }
186                }
187                /* zobserwowanie wzrostu*/
188                else if(obr==1)
189                {
190                    if((wyniki[abs((i-i2)%rozmiar_dan+1))]+min_proc1)<wyniki[i])
191                    {
192                        obr=2;
193                        memset(wyniki, 0, 1000 * sizeof(int));
194                        break;
195                    }
196                }
197                /* zobserwowanie osiągnięcia stanu wysokiego*/
198                else if(obr==2)
199                {
200                    if((wyniki[abs((i-i2)%rozmiar_dan+1))]+min_proc1)<wyniki[i])
201                    {
202                        obr=2;
203                        memset(wyniki, 0, 1000 * sizeof(int));
204                        break;
205                    }
206                }
207                /* zobserwowanie osiągnięcia stanu wysokiego*/
208                else
209                {
210                    if(okolica<wyniki[i]+200 )
211                    {
212                        obr=0;
213                        i-=1;
214                        obraty++;
215                        memset(wyniki, 0, 1000 * sizeof(int));
216                        break;
217                    }
218                }
219            }
220            i=(i+1)%rozmiar_dan;
221        }
222    }
223    /* USER CODE END 3 */
224 }
```

W sekcji while zaimplementowaliśmy zliczenie obrotów silnika na podstawie skoków wartości rezystancji fotorezystora. Dodatkowo dodaliśmy warunki by program drobnych spadków jasności wywołanych ruchem otoczenia nie wpływał na zliczanie obrotów.

5.2 Program wyświetlający prędkość obrotową silnika z pomiarów

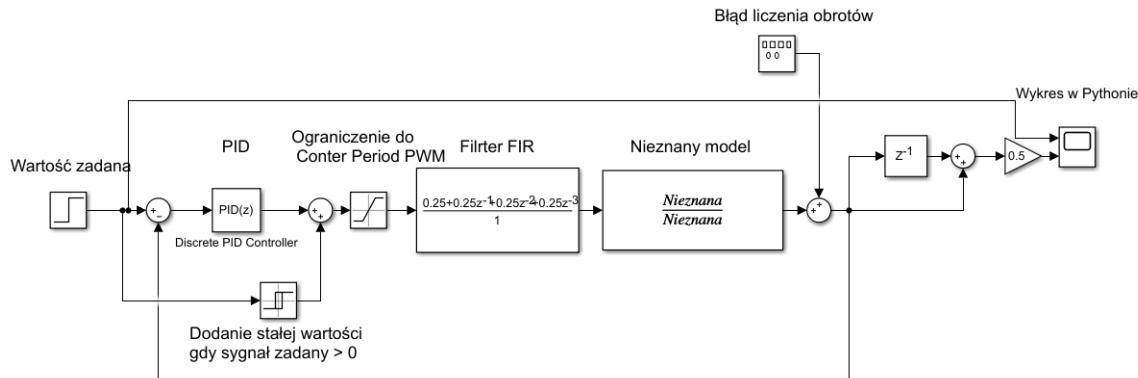
Program po uruchomieniu sczytuje prędkość obrotową silnika z modułu NUCLEO i wizualizuje dane w postaci wykresu. Wykres na osi y ilość obrotów zaś w osi x czas. Po uruchomieniu programu widoczne są na wykresie oscylacje. Wynikają one z szumu pomiarowego. Jednak nie utrudniają one oceny po prawnego działania układu regulacji.

Za pośrednictwem programu możemy również nastawić wartość zadaną.



6 Model regulatora

Przygotowaliśmy również prosty model działania naszego układu regulacji. Został on wykonany w programie Simulink. Można go pobrać z załączonego githaba i sprawdzić za jego pośrednictwem czy działanie układu rzeczywistego działa zgodnie z założeniami.



7 Analiza działania regulatora PID na podstawie przebiegów pokazanych w filmie

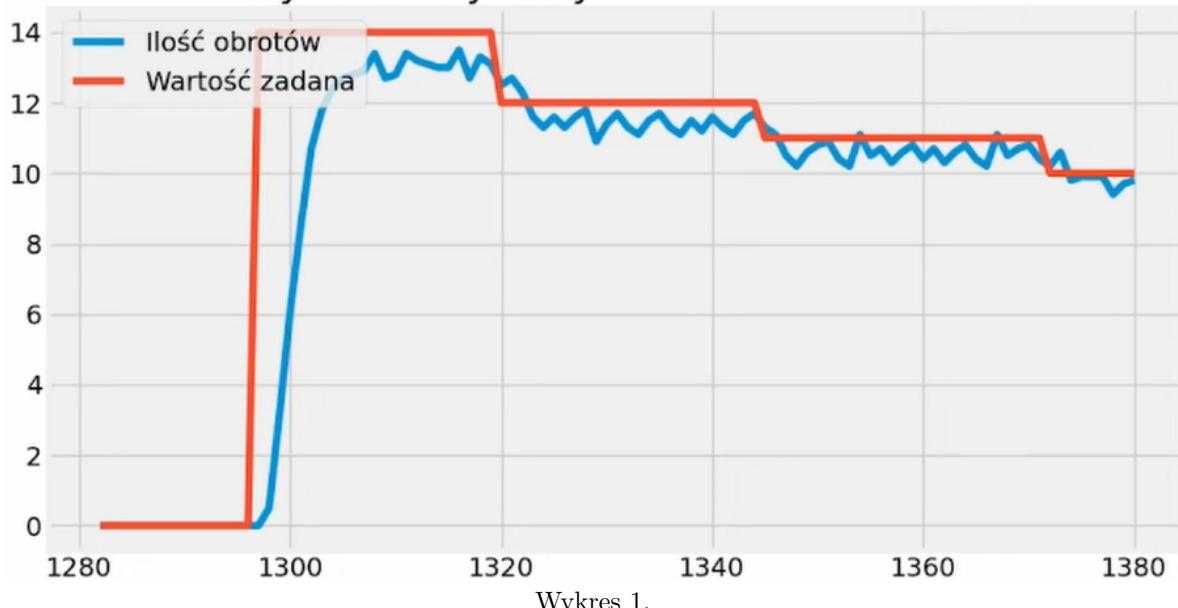
Na podstawie poniższych wykresów można stwierdzić, że regulator najlepiej działa dla wartości zadanej w okolicy 8 ponieważ dla tych wartości wyniki oscylują w wartości zadanej a dla wartości zadanych takich jak 11-14 widoczne jest niedoregulowanie a dla wartości 4-1 przeregulowanie. Obiekt osiąga wartość maksymalną od wartości minimalnej w ciągu ok 1 sekundy ale spadek trwa już ok 4 sekundy. Możliwy byłby szybszy spadek gdyby była możliwość załączenia przeciwnych obrotów.

Na wykresie 5 widoczne jest początkowe znaczne przeregulowanie wynikające z konieczności rozpoczęcia silnika ale po dłuższej chwili sygnał się reguluje w okolicach wartości zadanej (1.5 obrotu w ciągu 0.25s).

Na wykresie 9 można zaobserwować powolną utratę zgromadzonej energii a po dłuższej chwili (po 2380 próbce) większe skoki obrotów wynikające z konieczności dopędzenia w celu pokonania tarcia które ma większe znaczenie po utracie zgromadzonej energii.

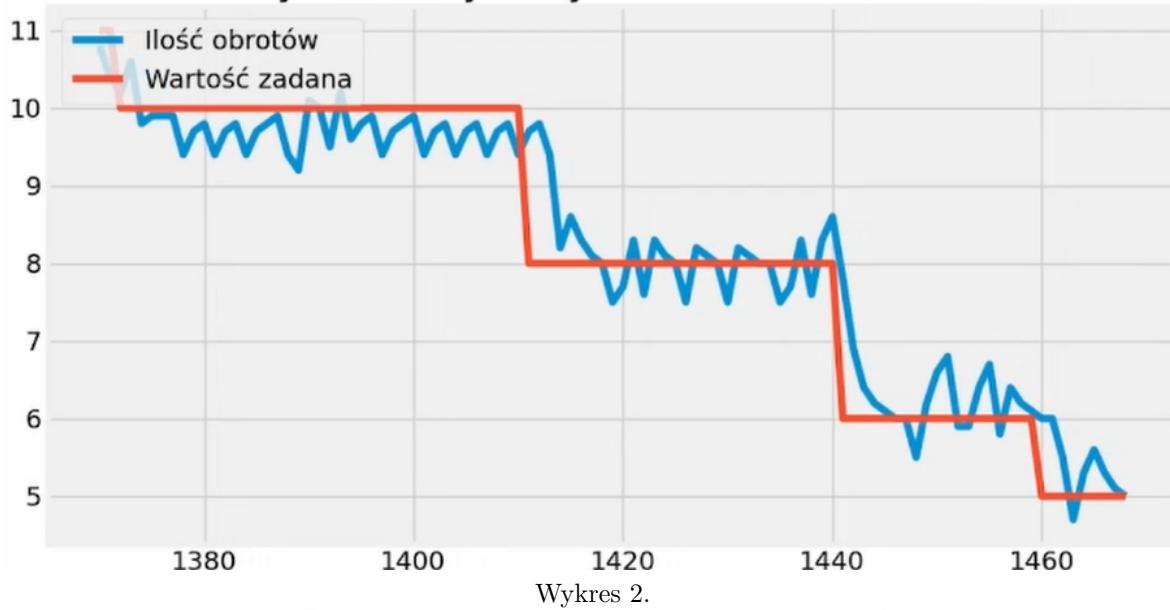
Należy zwrócić uwagę na błąd pomiaru, który może wynosić w granicach +/- 1 obrotu na próbkę.

Wykres otrzymanych wartości co 0.25s

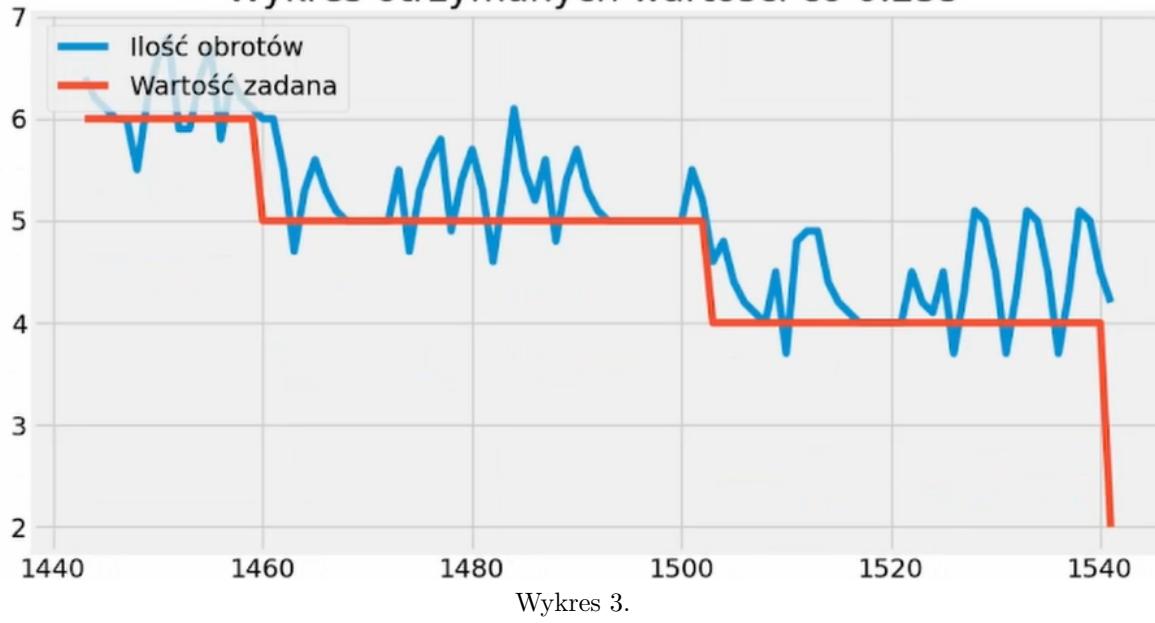


Wykres 1.

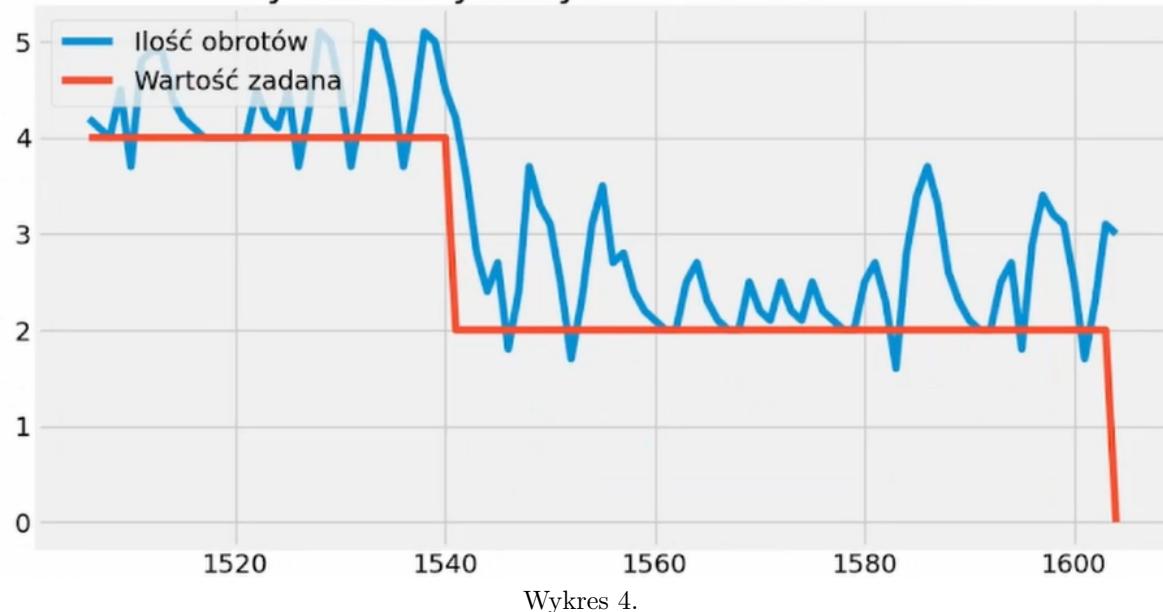
Wykres otrzymanych wartości co 0.25s



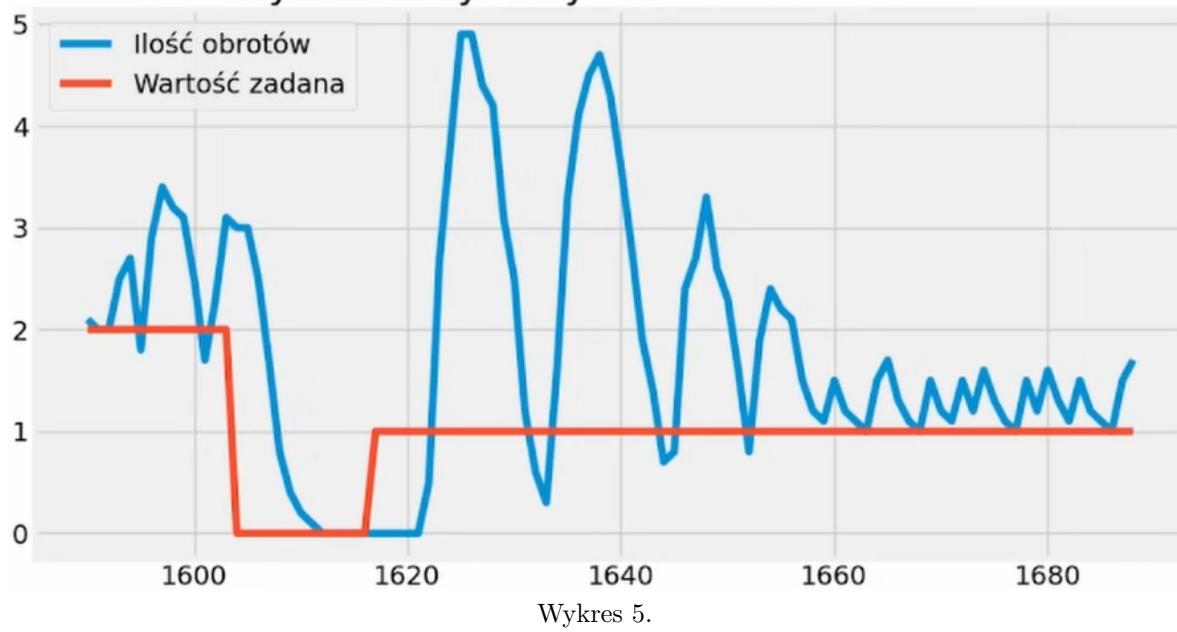
Wykres otrzymanych wartości co 0.25s



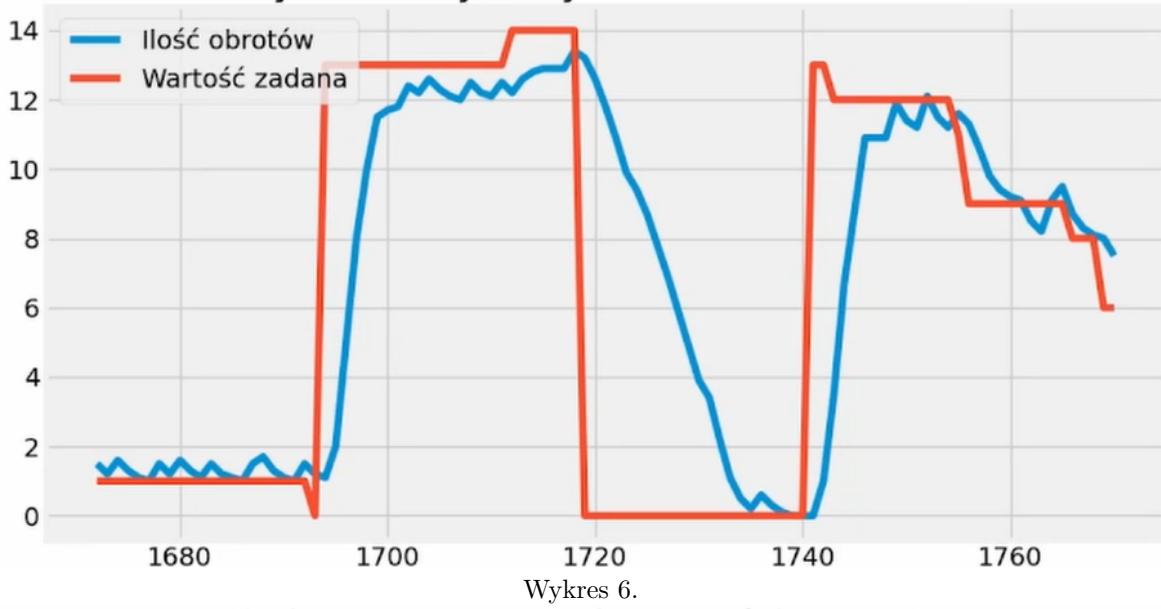
Wykres otrzymanych wartości co 0.25s



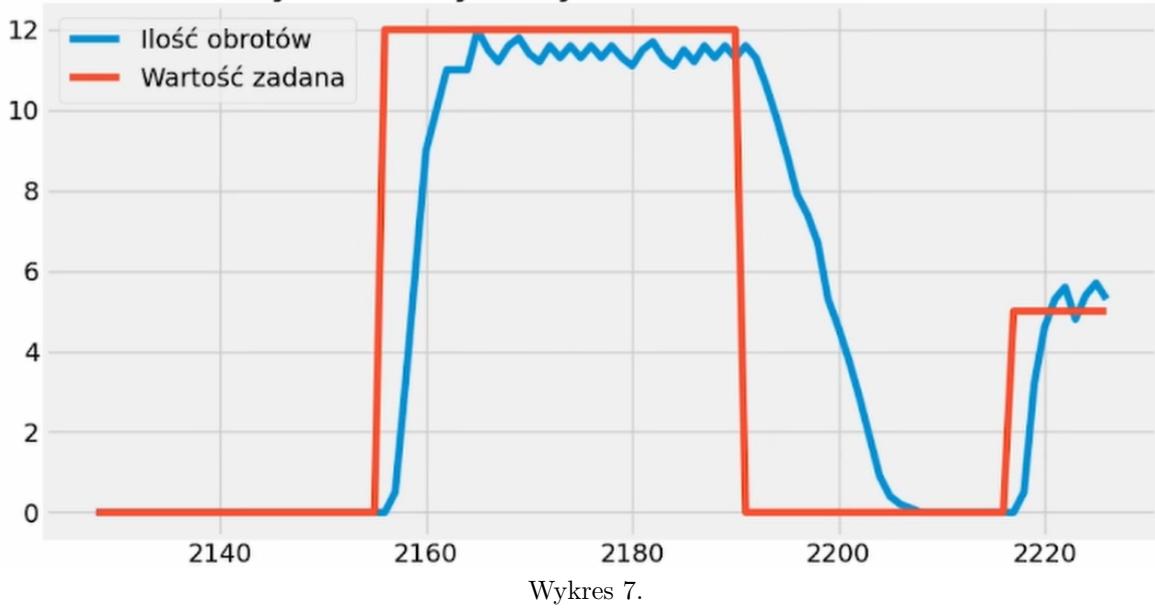
Wykres otrzymanych wartości co 0.25s



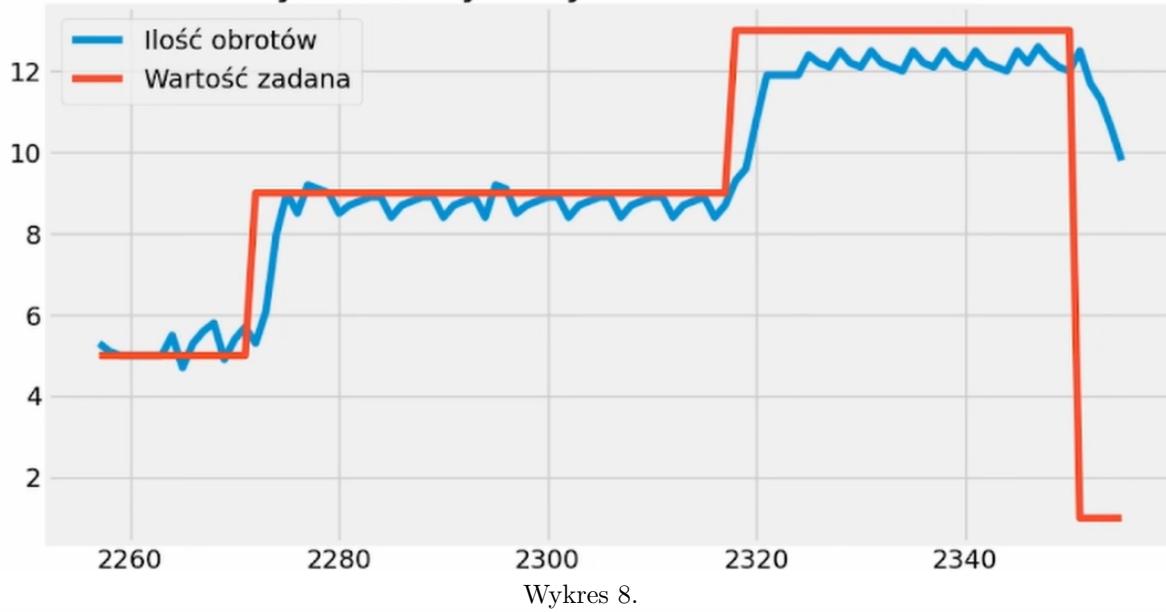
Wykres otrzymanych wartości co 0.25s



Wykres otrzymanych wartości co 0.25s

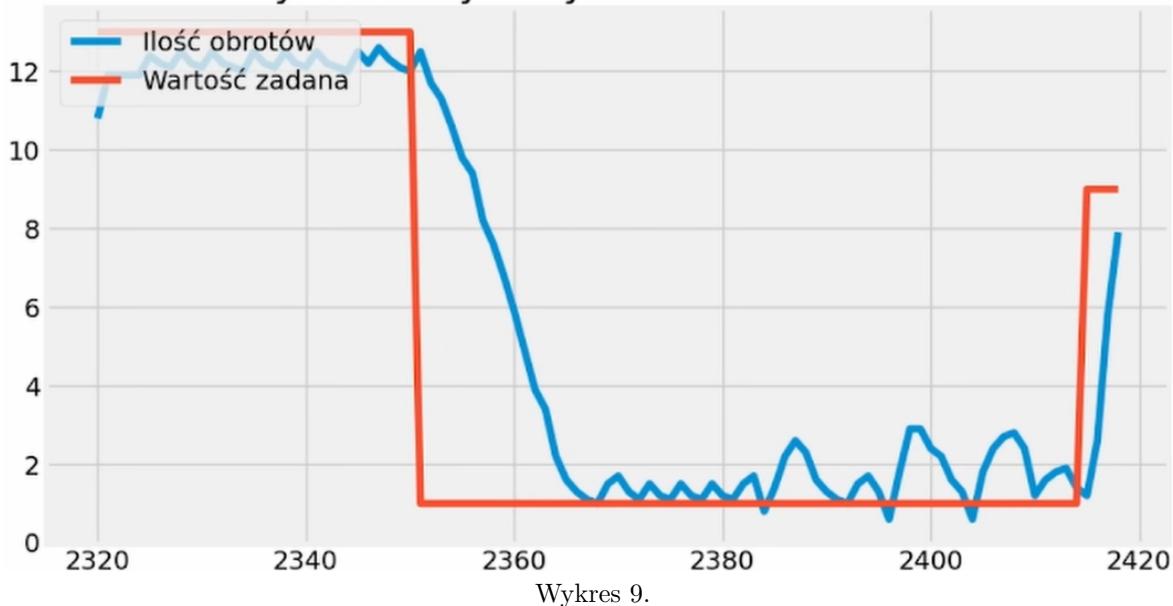


Wykres otrzymanych wartości co 0.25s



Wykres 8.

Wykres otrzymanych wartości co 0.25s



Wykres 9.