

Temat pracy:

Odczyt oraz analiza danych w czasie rzeczywistym.

Cel pracy:

Stworzenie odpowiednika telegrafo, rejestrującego impulsy elektryczne, które dalej będą zamieniane na odpowiednie znaki, zapisywane jako tekst i przesyłane do komputera.

Realizacja:

1. Impulsy rejestrowane będą na podstawie aktywności bicepca. Dzięki temu będę w stanie określić dwa stany:
 - 1 – stan, w którym biceps jest napięty
 - 0 – stan, w którym nie jest
2. Do reprezentacji poszczególnych znaków wykorzystam alfabet Morse'a.
3. Obsługa danych oraz ich zamiana na znaki zrealizowana będzie na płytce Arduino UNO

Plan pracy:

1. Przygotowanie układu będącego w stanie rejestrować impulsy elektryczne.
2. Napisanie programu będącego w stanie przetwarzać sygnały, a co za tym idzie:
 - rozróżnić alfabet stosowany w kodowaniu Morse'a:
 - kropka – stan „1” trwający krócej od kreski
 - kreska – stan „1” trwający dłużej od kropki
 - odróżnić właściwy sygnał od zakłóceń
 - określić, kiedy zaczyna się kodowanie danego symbolu, a kiedy kończy

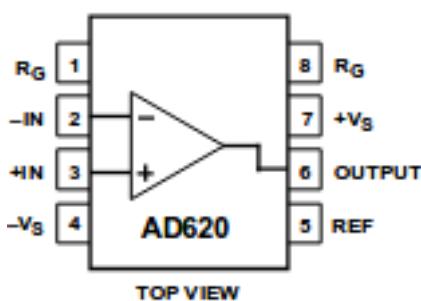
Część pierwsza – budowa układu odbierającego impulsy elektryczne:

Słowo wstępne:

Układ ten zbudowałem na podstawie poradnika. Wstępnie projektowany był on do badania sygnałów eeg, jednakże po przetestowaniu go okazało się, że można nim również badać impulsy mięśniowe.

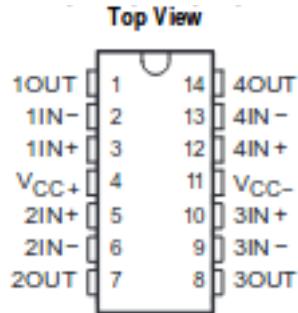
Wykorzystane materiały:

- Wzmacniacze operacyjne
 - 1x AD620A
 - Opis pinów



- Przystosowanie wzmacniacza do wykorzystania go na płytce uniwersalnej: Wzmacniacz kupiłem w wersji A, która nie posiadała pinów dostosowanych do zamontowania jest na płytce uniwersalnej. W celu jego zaadaptowania, przyspawiałem go do płytki wraz z odpowiednimi goldpinami.

- 2x TL084CN
 - Opis pinów

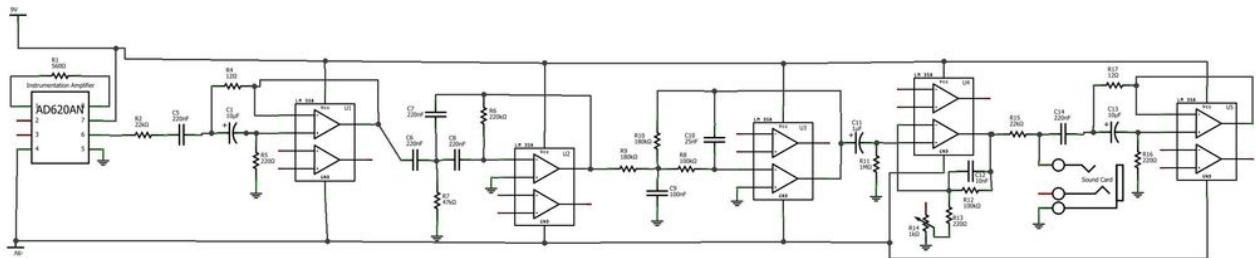


- Kondensatory
 - 1x 10 nF, ceramic
 - 1x 20 nF, ceramic
 - 1x 100nF, tantalum
 - 5x 220nF, tantalum
 - 1x 1uF, electrolytic
 - 2x 10uF, electrolytic
- Rezystory
 - 2x 12Ω
 - 1x 220Ω
 - 1x 560Ω
 - 2x 22kΩ
 - 1x 47kΩ
 - 2x 100kΩ
 - 2x 180kΩ
 - 1x 220kΩ
 - 2x 270kΩ
 - 1x 1MΩ
- Potencjometr
 - Rodzaj
 - 1x 1kΩ
 - Oznaczenia:
 - szary: wejście
 - brązowy: wyjście
 - czarny: masa
- Połączenia
 - płytka uniwersalna
 - okablowanie
 - przewody do elektrod biomedycznych
 - <https://botland.com.pl/pl/czujniki-medyczne/2605-przewody-do-elektrod-biomedycznych.html>
 - część kabli dostosowana do płytki uniwersalnej
 - część kabli wycięta i oskurowana ze skrętki
- Baterie:
 - 2x 9V
- Sensory
 - elektrody biomedyczne

- <https://botland.com.pl/pl/czujniki-medyczne/2604-elektrody-biomedyczne-10szt.html>
- Płytki arduino UNO

Przygotowanie układu:

Schemat ogólny:

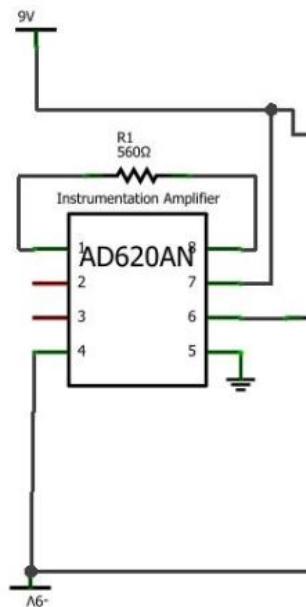


Opis poszczególnych części:

1. Wzmocnienie sygnału (90x)

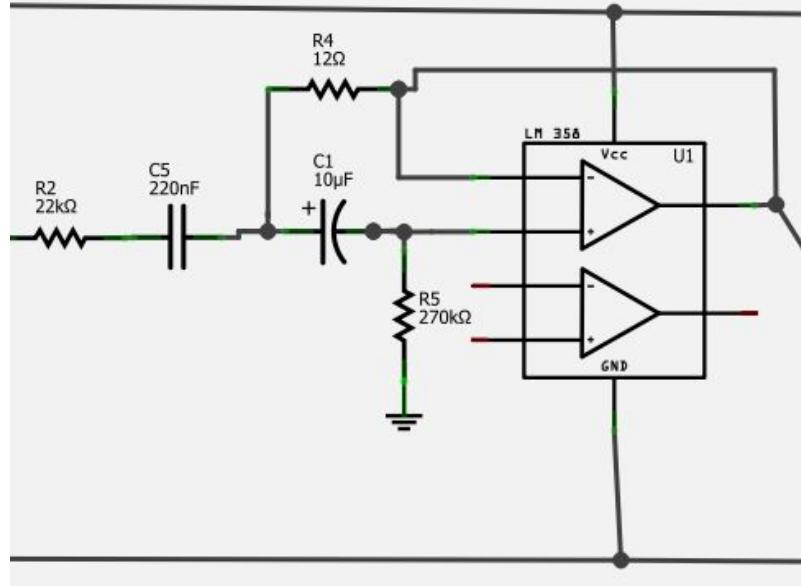
Oblicza on również różnicę sygnałów rejestrowanych na końcach bicepsu.

Po zastosowaniu opornika 560Ω uzyskane zostanie prawie 90x wzmocnienie (bazując na wzorze z dokumentacji: $G = 49,400 / [\text{opornik}] + 1$)



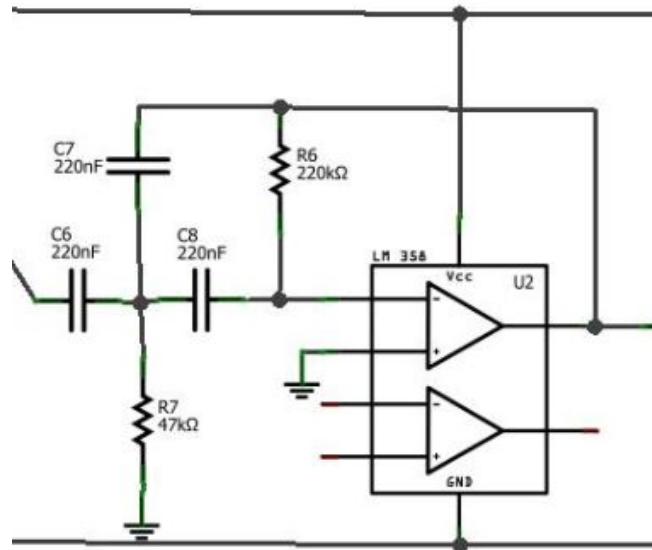
2. 60Hz filtr Notcha

Filtr ten wykorzystywany jest w celu usunięcia zakłóceń tworzących się na okablowaniu. Stosując rezystor 12Ω zapewnione jest odfiltrowywanie częstotliwości oscylujących w okolicy 60Hz, czyli takich, jakie wytwarzane są przez sam układ podczas jego pracy.



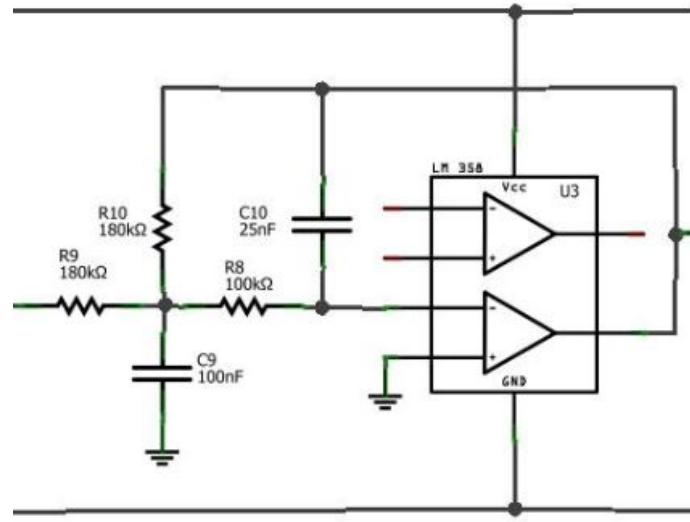
3. 7Hz filtr górnoprzepustowy

Jego zadaniem jest wyeliminowanie zakłóceń biorących się z powierzchni skóry. W tym celu będzie odfiltrowywał wszystkie częstotliwości będące poniżej 7Hz.



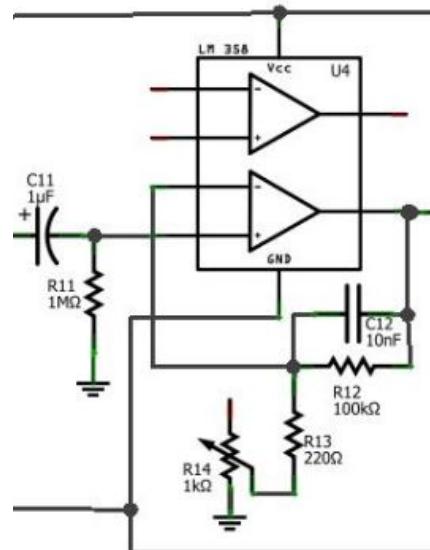
4. 31Hz filtr dolnoprzepustowy

Jest to pozostałość po wstępnej wersji wzmacniacza, mającego rejestrować aktywność mózgu. Zważywszy na to, że mózgowe fale beta mają częstotliwość mniejszą niż 30Hz, wszelkie wartości, które są powyżej niej, są nieistotne i mogą zostać pominięte.



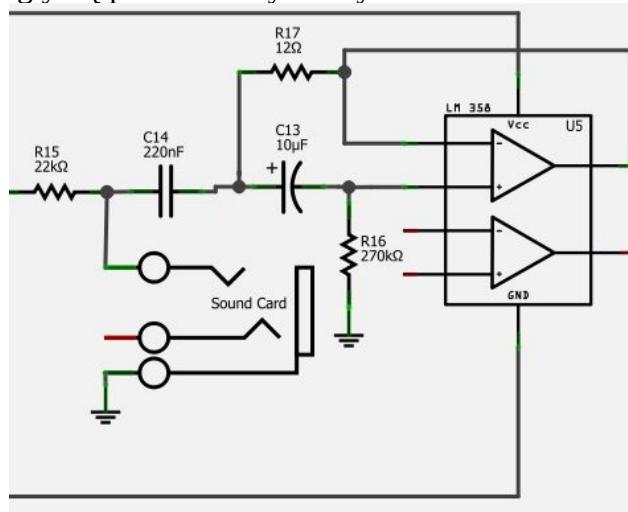
5. Wzmocnienie sygnału (83x – 455x)

Wzmocnienie to jest zmienne, dzięki czemu można je w łatwy sposób dostosowywać. Uzyskane jest to za pomocą potencjometru, który w tym przypadku działa jak rezystor, którego oporność można zmieniać.



6. 60Hz filtr Notcha

Przed wysłaniem informacji do układu, jest ona jeszcze raz odfiltrowywana z ewentualnych zakłóceń, jakie mogły się po drodze wytworzyć.

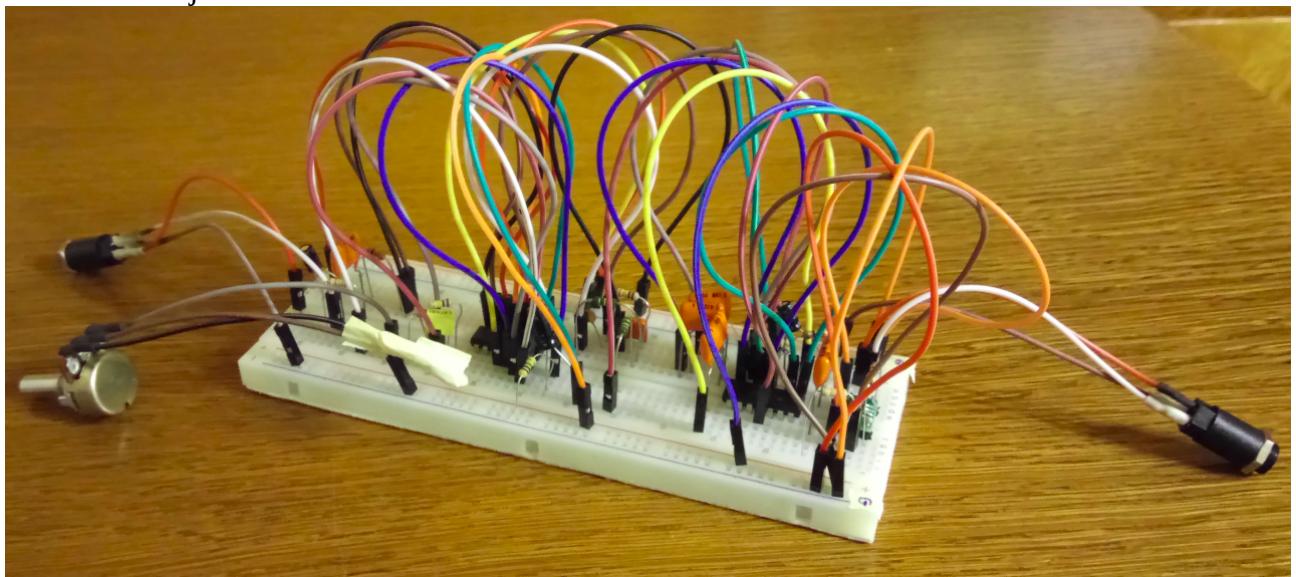


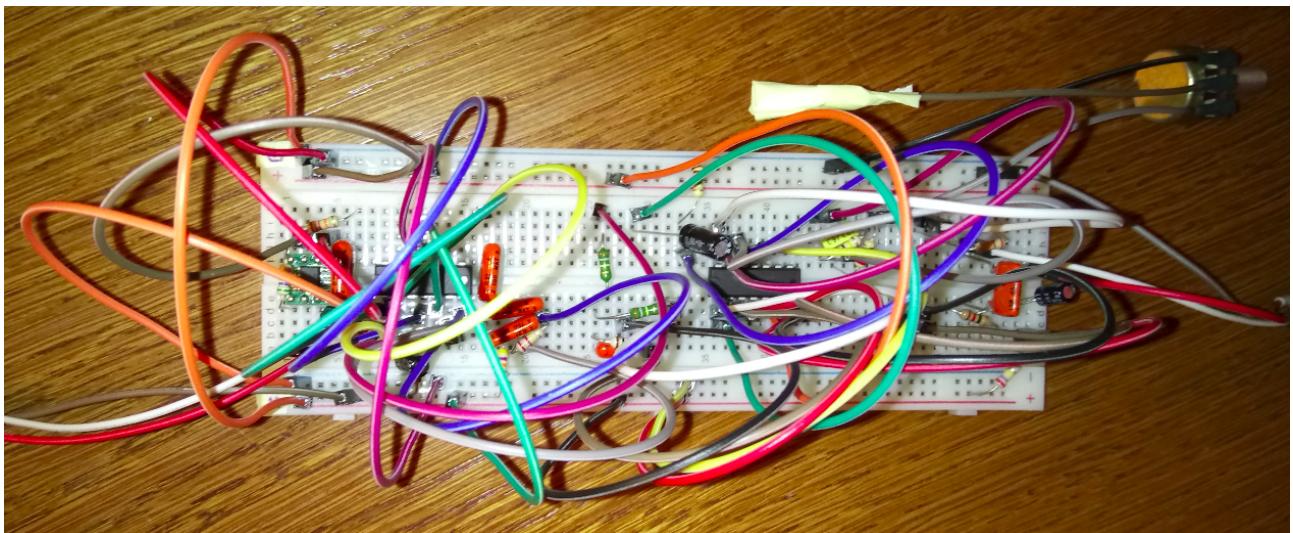
Realizacja układu

Układ ten zbudowałem dwa razy. Przy swojej pierwszej próbie zastosowałem zbyt długie przewody, które musiałem wymienić, ponieważ:

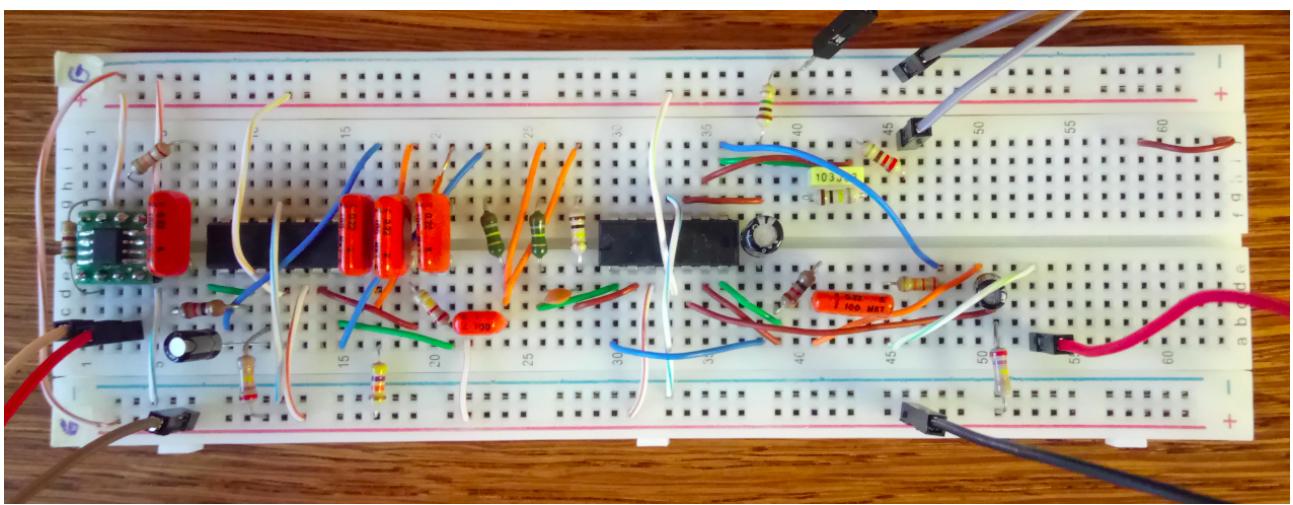
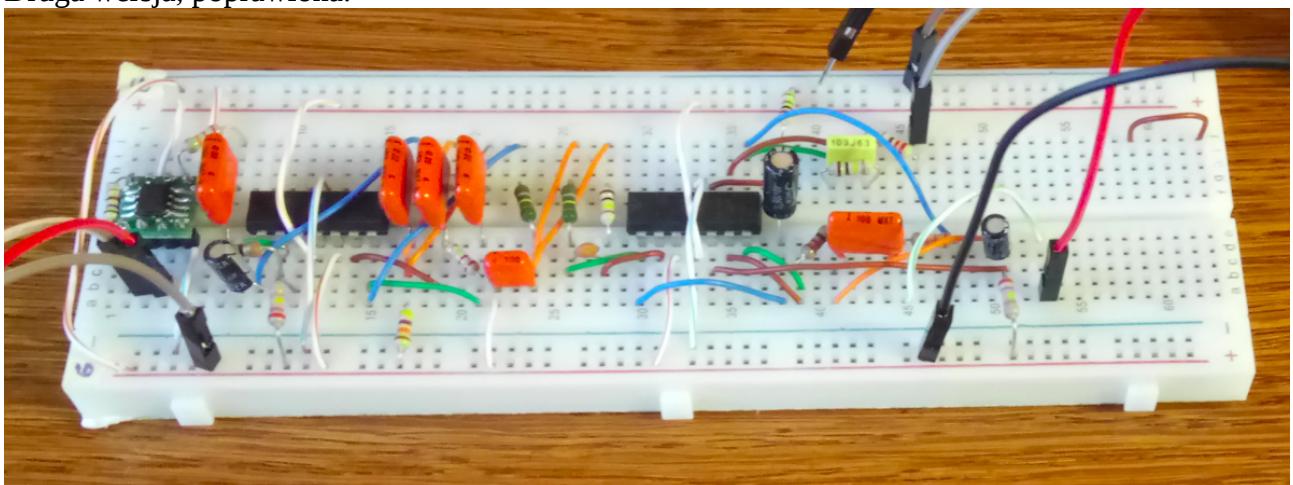
- ciężko się było zorientować co jest z czym podpięte
- ciężko było sprawdzić czy wszystko jest dobrze podłączone
- kable te mogły niekorzystnie wpływać na układ, dodając niepotrzebnych zakłóceń (przy pierwszym testowaniu po pewnym czasie układ przestawał działać – nie wiem czy wynikło to z samych kabli, czy też inne czynniki (jakość, stan oraz sposób podłączenia baterii) miały na to większy wpływ – w drugiej wersji zmieniłem baterie na inne, dokupiłem na nie nakładkę do podłączenia do płytki (przedtem obkręcone były drutem i „zabezpieczone” gumą do żucia) oraz, a może prze wszystkim, skróciłem długość kabli)

Pierwsza wersja:





Druga wersja, poprawiona:

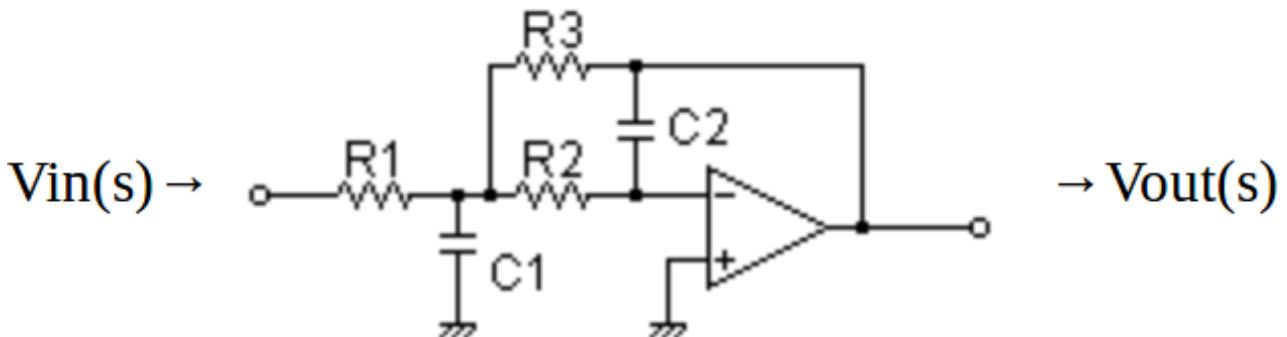


Wprowadzenie poprawek

Pierwsza wersja wzmacniacza dostosowana była do odbioru impulsów mózgowych. Jednakże, w tym projekcie operacje będą wykonywane na impulsach mięśniowych, których charakterystyka jest trochę inną i tak jak zakres impulsów mózgowych wynosi (0Hz – 30Hz) tak częstotliwość wytwarzana przez impulsy mięśniowe rozciąga się w przedziale (0Hz - 500Hz).

Poprawka ta nie będzie jednak uciążliwa, gdyż sprowadzi się jedynie do wymiany podzespołów z filtru dolnoprzepustowego.

Filtr, pierwotnie wykorzystany w układzie, nosi nazwę „Multiple Feedback Low-pass filter”. Jego schemat wygląda następująco:



Moim celem jest teraz takie dobranie podzespołów, aby odfiltrować wszystkie częstotliwości powyżej 500Hz

Do przeprowadzenia obliczeń wykorzystam stronę:
<http://sim.okawa-denshi.jp/en/OPtazuLowkeisan.htm>

Dla ustalonych parametrów:

- próg: ~ 500Hz
- quality factor [Q] (czyli jak wąskie są przejścia graniczne): 0.7

Otrzymałem rezultat:

- $R_1 = 3.3\text{k}\Omega$
- $R_2 = 13\text{k}\Omega$
- $R_3 = 3.3\text{k}\Omega$
- $C_1 = 0.15\mu\text{F}$
- $C_2 = 0.015\mu\text{F}$

Dla takiej konfiguracji otrzymam:

- próg: 512Hz
- quality factor: 0.7
- damping ratio [ζ] (czyli jak szybko funkcja jest „wyciszana”): 0.7

Przy sprawdzaniu poprawności układu zauważylem, że wcześniejszy filtr dolnoprzepustowy miał inne właściwości niż były podane, mianowicie odcinał on częstotliwości powyżej 23Hz, a nie jak to było podane, 31Hz. Dla filtra górnoprzepustowego (7Hz) wszystko było w porządku.

Oprócz tego pozbyłem się potencjometru, którego nieobecność zrekompensowałem oprogramowaniem, odpowiednio ustalając próg rozgraniczający stan aktywny mięśnia od nieaktywnego.

Część druga – analiza oraz przetwarzanie danych:

W tym rozdziale opiszę proces powstawania oprogramowania. Podzielony on będzie na mniejsze etapy:

- stworzenie odpowiedniej struktury danych
- rozróżnienie kiedy mięsień jest aktywny, a kiedy nie
- interpretacja wyników, w celu zamiany impulsów na alfabet Morse'a
- zamiana alfabetu Morse'a na alfabet łaciński

Wstęp

Do kodowania poszczególnych znaków wykorzystany jest alfabet Morse'a. Poniżej przedstawiona jest tabela ukaująca przyporządkowanie alfabetu Morse'a do alfabetu łacińskiego, wraz z liczbami i paroma znakami specjalnymi.

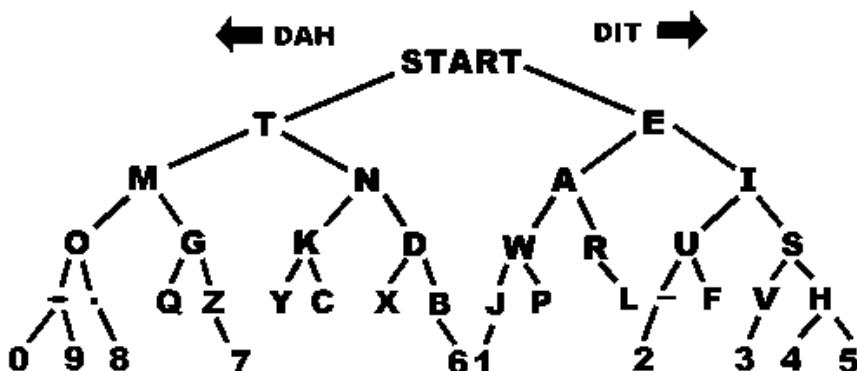
A	N	0	:	^
B	O	1	-	%
C	P	2	- -	\$
D	Q	3	- - -	#
E	R	4	- - - -	@
F	S	5	- - - - -	
G	T	6	- - - - - -	~
H	U	7	- - - - - - -	-
I	V	8	- - - - - - - -	?
J	W	9	- - - - - - - - -	
K	X)		
L	Y	(
M	Z	"		

Międzynarodowe kodowanie Morse'a składa się z pięciu elementów:

- kropki: trwającej 1 jednostkę czasową (j. cz.)
- kreski: trwające 3 j. cz.
- przerwy między kropkami i kreskami: trwającej 1 j. cz.
- przerwy między literami alfabetu łacińskiego: trwającej 3 j. cz.
- przerwy między wyrazami z języka łacińskiego: trwającej 7 j. cz.

Przechowywanie danych

Dopasowanie alfabetu Morse'a do alfabetu łacińskiego będę trzymam w strukturze drzewiastej. Dzięki temu w łatwy sposób mogę przechodzić po kolejnych znakach odczytywanych z sensora.



Struktura pojedynczego węzła:

```
struct node {  
    node *dot;  
    node *dash;  
    int value;  
};
```

Rozróżnienie kiedy mięsień jest aktywny, a kiedy nie

Do określenia stanu mięśnia wykorzystałem metodę próbkowania. Co ustalony przedział czasu [PROBING_DELAY] odczytuję sygnał wysyłany przez wzmacniacz. Wyniki te zapisuję kolejno w tabeli o rozmiarze [PROBING_CYCLE].

Następnie, kiedy wypełnię całą tabelę, obliczam odchylenie standardowe i porównuję je z [MUSCLE_THRESHOLD]. Jeżeli wartość odchylenia standardowego jest większa od tej stałej, to przyjmuję, że w tym okresie czasu mięsień był aktywny.

Wybranie tej konwencji narzuca w tym rozwiązaniu przedział czasu, co jaki określany jest stan mięśnia. W tym przypadku jest to [PROBING_DELAY] * [PROBING_CYCLE].

Określenie [MUSCLE_THRESHOLD]

- odchylenie standardowe, kiedy mięsień nie jest aktywny: [15 - 20]
- odchylenie standardowe przy aktywności mięśnia: [80 - 120]

Dobór stałych:

- PROBING_DELAY = 5 ms
- PROBING_CYCLE = 40
- MUSCLE_THRESHOLD = 70
 - jest to zarazem najwrażliwsza część całego programu, także przy wystąpieniu jakichkolwiek błędów, warto zacząć od tej stałej
- Czas potrzebny do określenia stanu mięśnia = 200 ms

Określenie kodowania na podstawie aktywności mięśnia

Kodowanie Morse'a można reprezentować jako ciąg zer i jedynek z alfabetu $\Sigma = \{ 0, 1 \}$ następującym podstawieniem:

- | | |
|---|---------|
| • kropka: | 1 |
| • kreska: | 111 |
| • przerwa między kropkami i kreskami: | 0 |
| • przerwa między literami alfabetu łacińskiego: | 000 |
| • przerwa między wyrazami z języka łacińskiego: | 0000000 |

Przyjmuję, że czas trwania „1” jak i „0” jest taki sam i wynosi [DOT_CHUNK_DURATION]. W tym czasie zbierane są dane o aktywności mięśnia i jeżeli więcej w nim będzie stanów, w których mięsień był aktywny, to przyjmuję, że kodowany był znak „1”. W przeciwnym przypadku: „0”.

Wystąpienie tych samych odczytów pod rząd jest zapisywane i kiedy zostanie przerwane (po ciągu „1” odczytane zostało „0” lub na odwrót) to dokonywana jest ich interpretacja.

Wyniki z tej interpretacji zapisywane są do tablicy o rozmiarze [TEXT_SIZE] i są wypisywane po tym jak wykryte zostanie błędne kodowanie.

Dobór stałych:

- DOT_CHUNK_DURATION = 2000 ms
- TEXT_SIZE = 100

Przygotowanie do odczytu danych

Zanim program przejdzie w tryb odczytu danych, czeka aż przez cały czas trwania [DOT_CHUNK_DURATION] odczyty z sensoru będą wskazywały na aktywność mięśnia. Kiedy warunek ten zostanie spełniony, to czeka kolejne [DOT_CHUNK_DURATION], po którym następuje właściwy odczyt danych.

Kod programu

Jest on dostępny pod adresem: <https://github.com/Sokolnik21/ArduinoMuscleTelegraph>