# Politechnika Warszawska
# Wydział Mechatroniki

Dokumentacja

# SLIDER

Zrealizowany w ramach projektu z przedmiotu
Urządzenia Multimedialne

Jarosław Affek
Daniel Fiedosiuk

Warszawa 2018

## Cel projektu:

Stworzenie mobilnego slidera fotograficznego do tworzenia timelapsów sterowanego zdalnie pilotem na podczerwień, zasilanego przenośnymi źródłami energii.

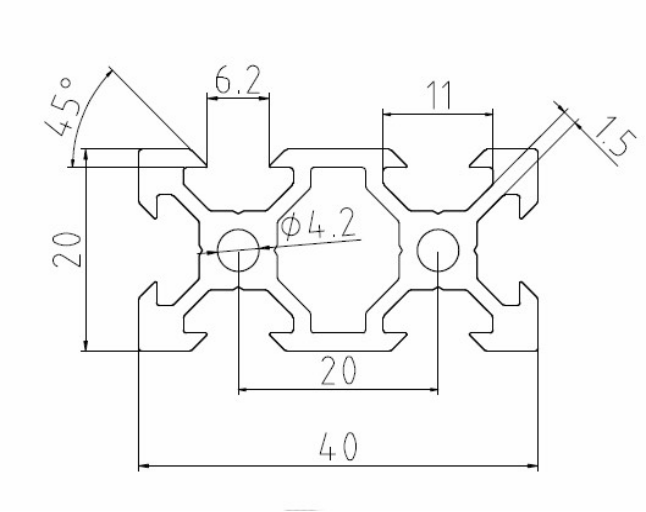
## Założenia:

- długość szyny slidera - 1,5 m
- możliwość zamontowania lustrzanki
- możliwość regulacji prędkości slidera
- zdalne sterowanie sliderem za pomocą pilota (na IR)
- ograniczenie ruchu czujnikami mechanicznymi
- przenośne zasilanie bateryjne/z akumulatora

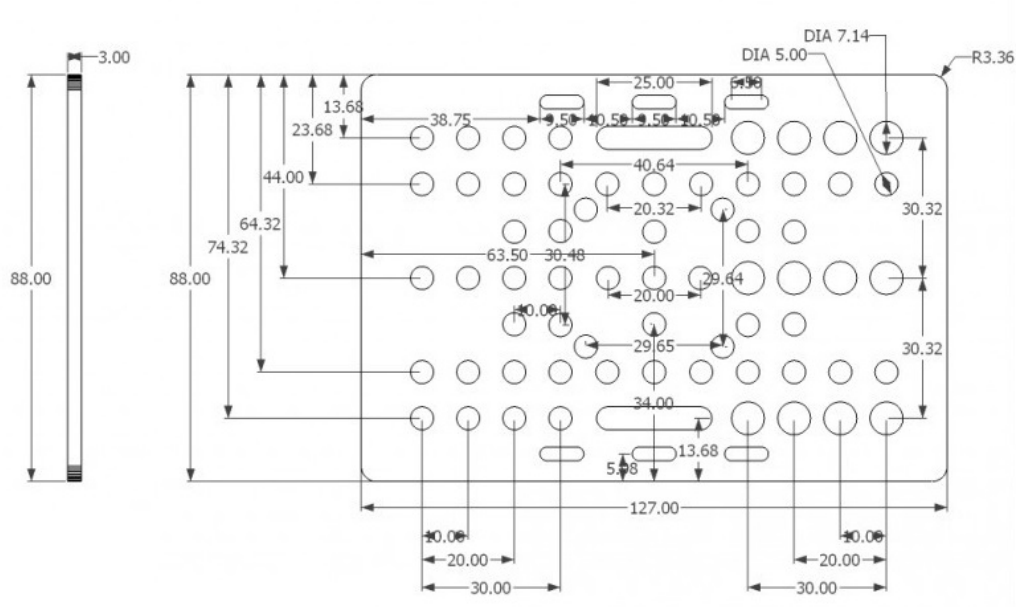
### 1. Budowa i wykorzystane elementy

Elementy mechaniczne:

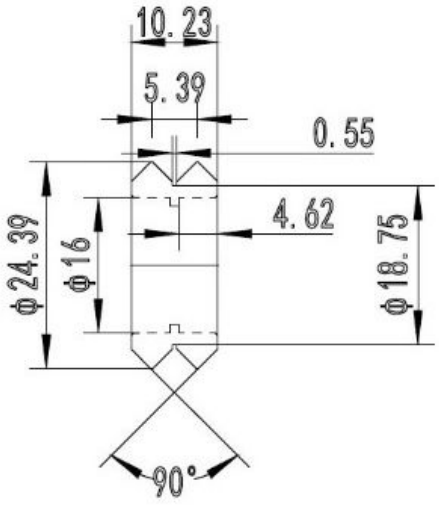
- Profil aluminiowy V-slot 20x40 mm, długość: 1500 mm



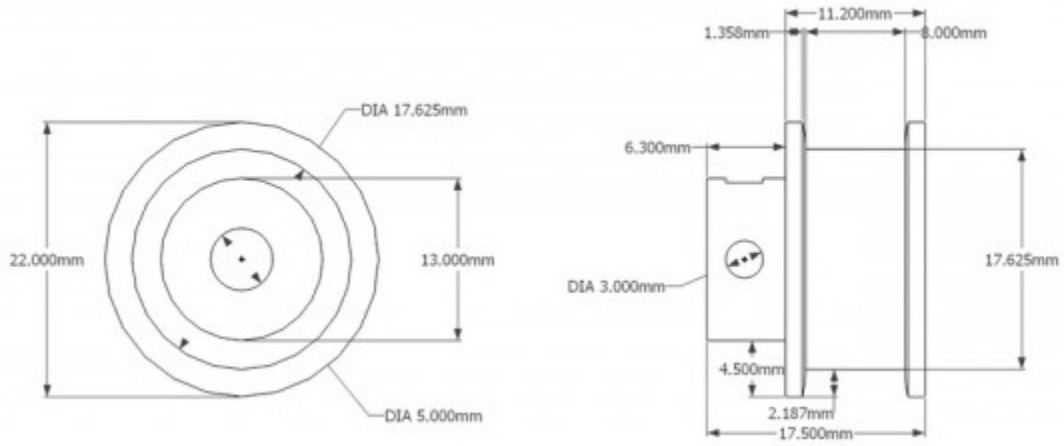

- Płyta montażowa 3x88x127 mm



- 4x rolki Solid V Wheel z łożyskami kulkowymi

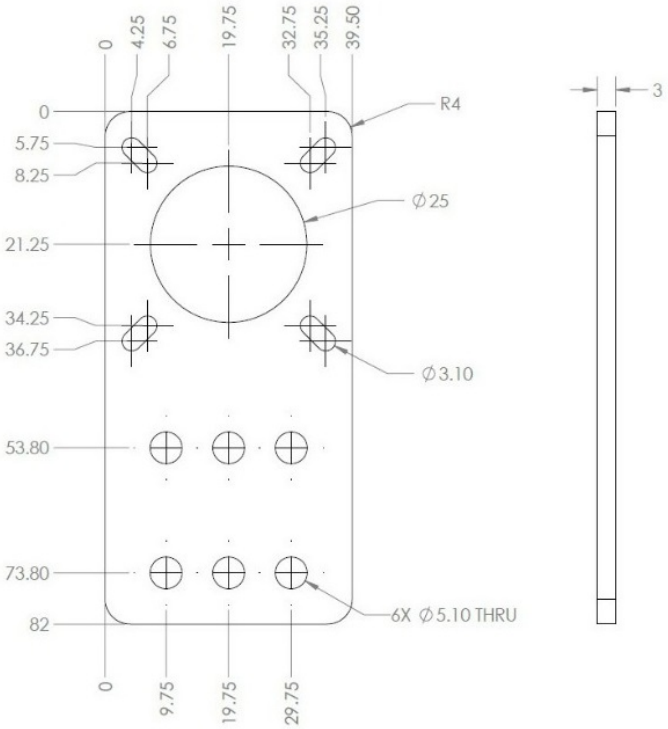



- Koło zębate GT2, 30 zębów, materiał: aluminium

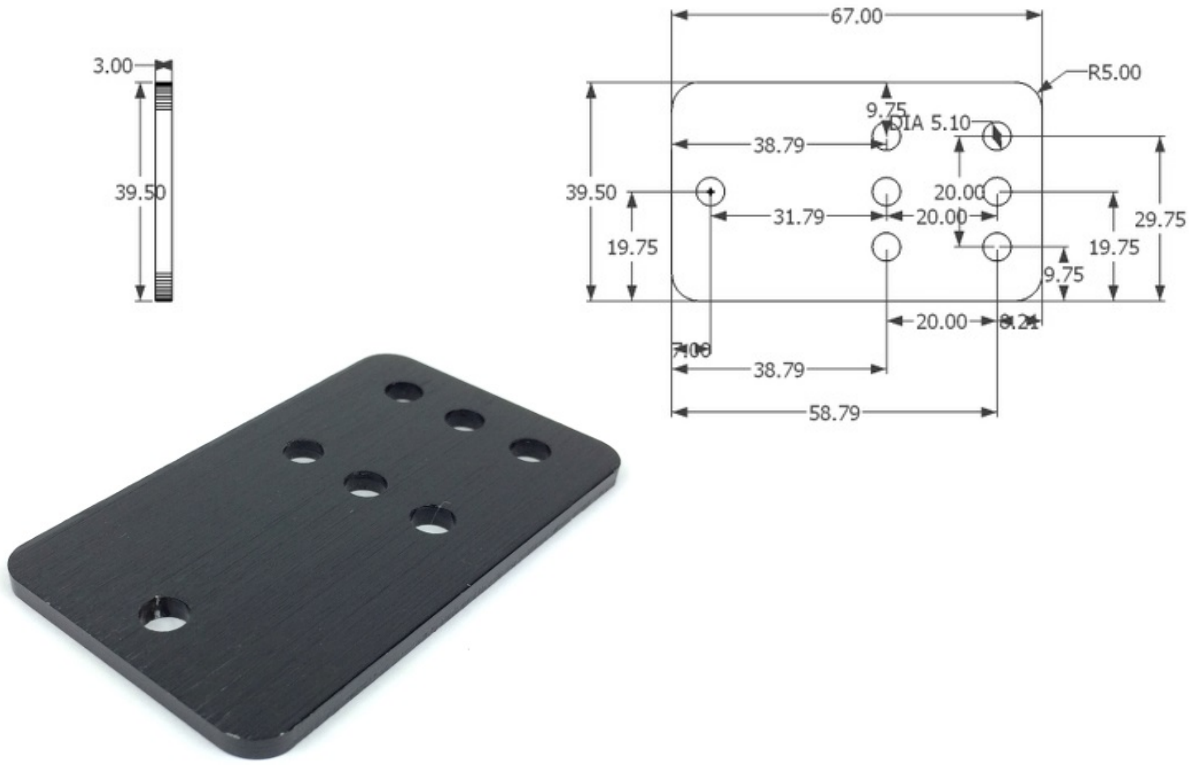



- Pasek zębaty GT2 o szerokości 6 mm, długość: 4m

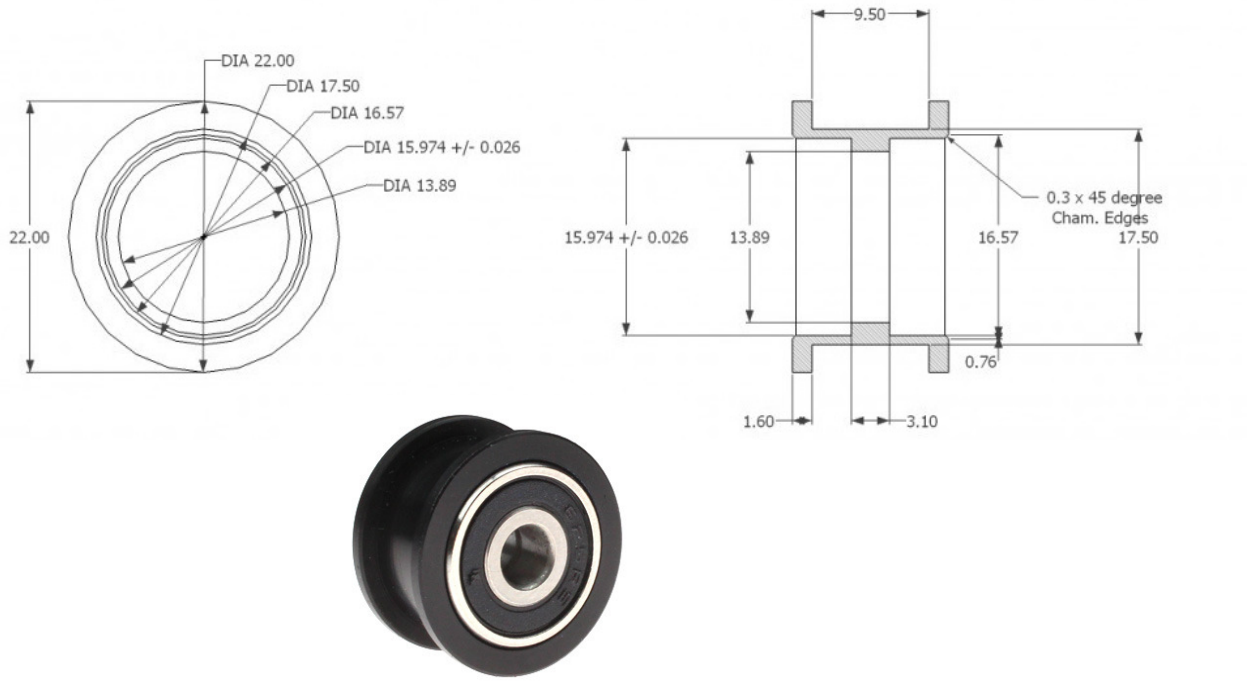
- Płyta montażowa do silnika typu NEMA17



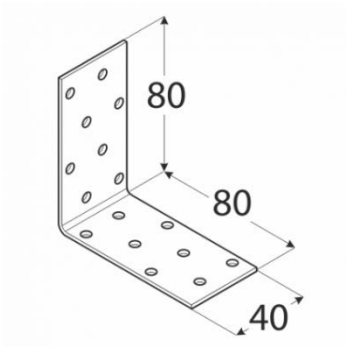

- Płyta montażowa do montażu koła pasowego



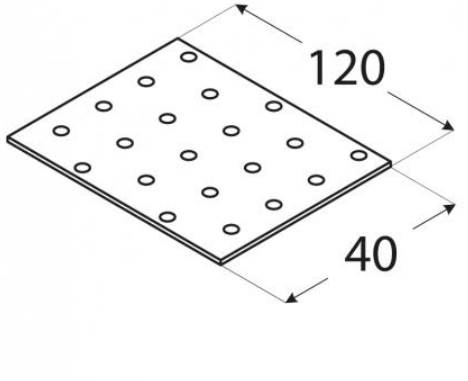

- Koło pasowe wraz z łożyskami kulkowymi



- 4 kątowniki montażowe DMX




- Płyty montażowe DMX



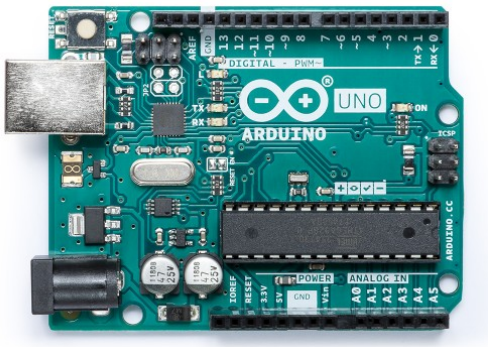

- Głowica kulowa Genesis BH-34


| Mocowanie szybkozłączki | Średnica kuli | Wysokość | Udźwig głowicy | Waga |
|---|---|---|---|---|
| Arca-Swiss | 34 mm | 93 mm | 15 kg | 340 g |

- Elementy łączące: śruby M5, nakrętki, podkładki
- Śruba statywowa ¼ cala wraz z przejściówką na 3/8 cala

<u>Elementy elektroniczne:</u>

- Arduino Uno Rev3




Specyfikacja dostępna na stronie producenta: https://store.arduino.cc/usa/arduino-uno-rev3

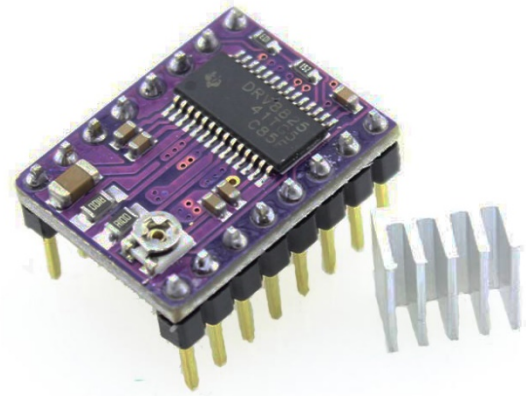
- Silnik bipolarny typu NEMA17 model 42HM48-1684

**Genaral Specification:**

| Item | Specifications |
|---|---|
| Step Angle | 1.8° |
| Temperature Rise | 80℃max |
| Ambient Temperature | -20℃~+50℃ |
| Insulation Resistance | 100 MΩ Min. ,500VDC |
| Dielectric Strength | 500VAC for 1minute |
| Shaft Radial Play | 0.02Max. (450g-load) |
| Shaft Axial Play | 0.08Max. (450g-load) |
| Max. radial force | 28N (20mm from the flange) |
| Max. axial force | 10N |

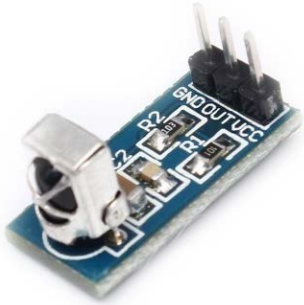
| Model No. | Step Angle | Motor Length | Current /Phase | Resistance /Phase | Inductance /Phase | Holding Torque | # of Leads | Detent Torque | Rotor Inertia | Motor Length |
|---|---|---|---|---|---|---|---|---|---|---|
| 42HM48-1684 | 0.9 | 48 | 1.68 | 1.65 | 4.1 | 4.4 | 4 | 250 | 68 | 0.35 |

- Sterownik silnika DRV8825



| Napięcie zasilania | Prąd ciągły na cewkę | Maksymalny chwilowy prąd na cewkę | Napięcie zasilania części logicznej | Rozdzielczość |
|---|---|---|---|---|
| 8,2 V – 45 V | 1,5 A | 2,2 A | 2,5 V – 5,25 V | 1, ½, ¼, 1/8, 1/16, 1/32 |

- Moduł odbiornika IR HX1838 zasilany napięciem 5V, kompatybilny z Arduino



- Pilot IR działający w standardzie NEC o nośnej 38kHz, 21 przycisków

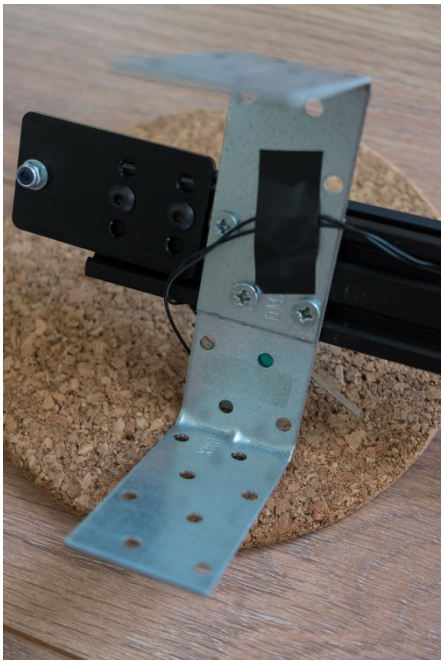
- 2x przełącznik krańcowy monostabilny typu ON-(ON) MSW-02-38

| Obciążalność styków | Konfiguracja syków | Dźwignia | Żywotność | Temperatura Pracy |
|---|---|---|---|---|
| 10A, 250VAC | ON-(ON); SPDT 3P | L = 38 mm | 100 000 cykli | -25°C do +85°C |

- Koszyk na 8 połączonych szeregowo baterii AA 1,5 V (w sumie 12 V)
- 8 baterii AA 1,5 V
- Kondensator 100 µF
- Płytka stykowa 170 pól

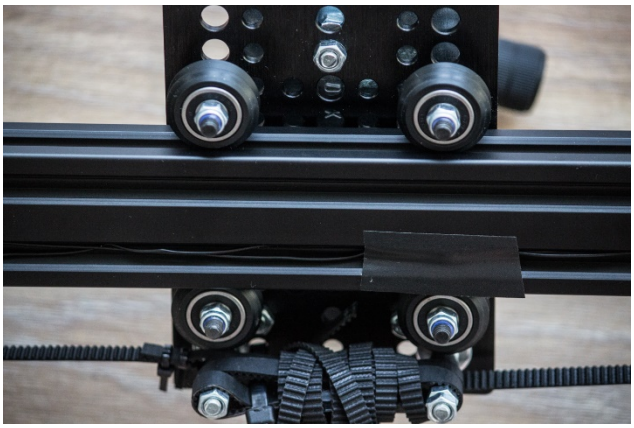
**2. Montaż**

1. Nogi slidera



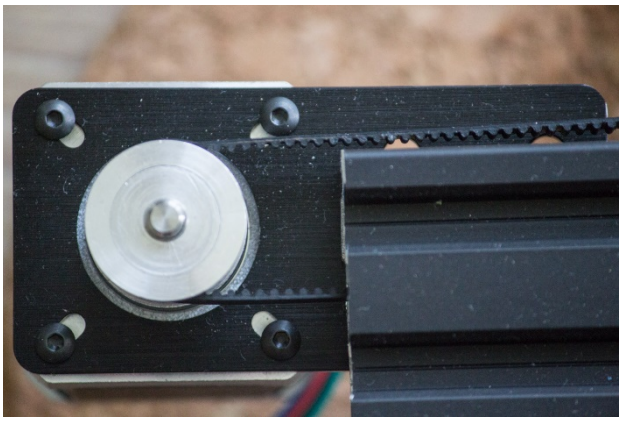
Dwa kątowniki zostały nałożone jeden na drugi i przykręcone bezpośrednio 4 śrubami do zeszlifowanych nakrętek w szynie.

2. Płyta montażowa główna

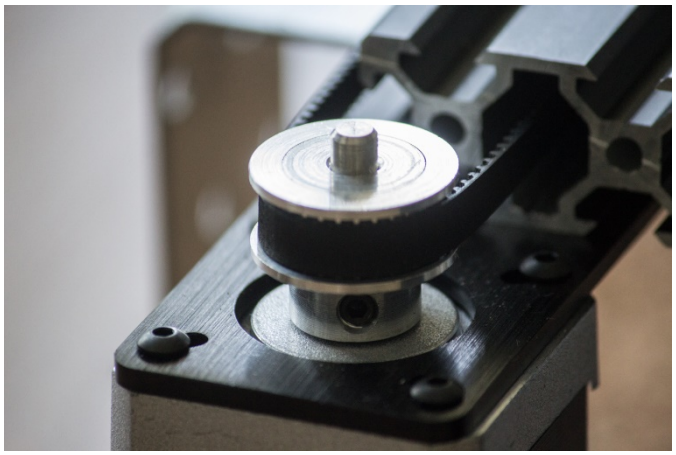


Do płyty montażowej zostały przykręcone 4 rolki (zawierające łożyska kulkowe o małym oporze ruchu). Następnie wsunięto płytę montażową wraz z rolkami na szynę. Mocowanie 2 rolek umożliwia kasowanie luzów wózka na szynie, dzięki zastosowaniu tulei mimośrodowych regulowanych kluczem francuskim.

3. Płyty montażowe na krańcach szyny



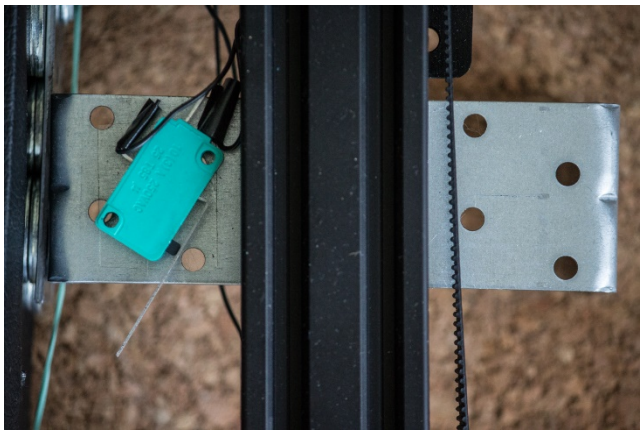
Płyty montażowe na krańcach szyny zostały przykręcone za pomocą dwóch śrub dokręconych do blaszek schowanych w szynie. Na jednej został zamocowany silnik, na drugiej koło pasowe na których rozciągnięty jest pasek zębaty. W przypadku poluzowania paska należy jedną płytek montażowych lekko odkręcić i przesunąć w odpowiednim kierunku. Pasek zębaty został przymocowany do wózka wraz z pewnym zapasem, który w razie przetarcia lub przerwania można wykorzystać do naprawy, następnie został poprowadzony z jednej strony na zewnątrz do wózka, a z drugiej strony jest schowany w środku szyny.

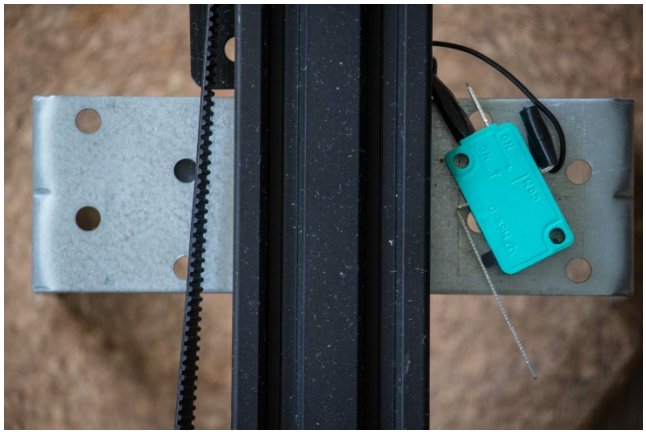


4. Silnik

Do silnika przymocowano koło zębate, na które nałożony jest pasek zębaty. Sam silnik jest przymocowany 4 śrubami do płyty montażowej. Zasilanie silnika jest odprowadzone do pudełka z elektroniką, które znajduje się tuż obok.

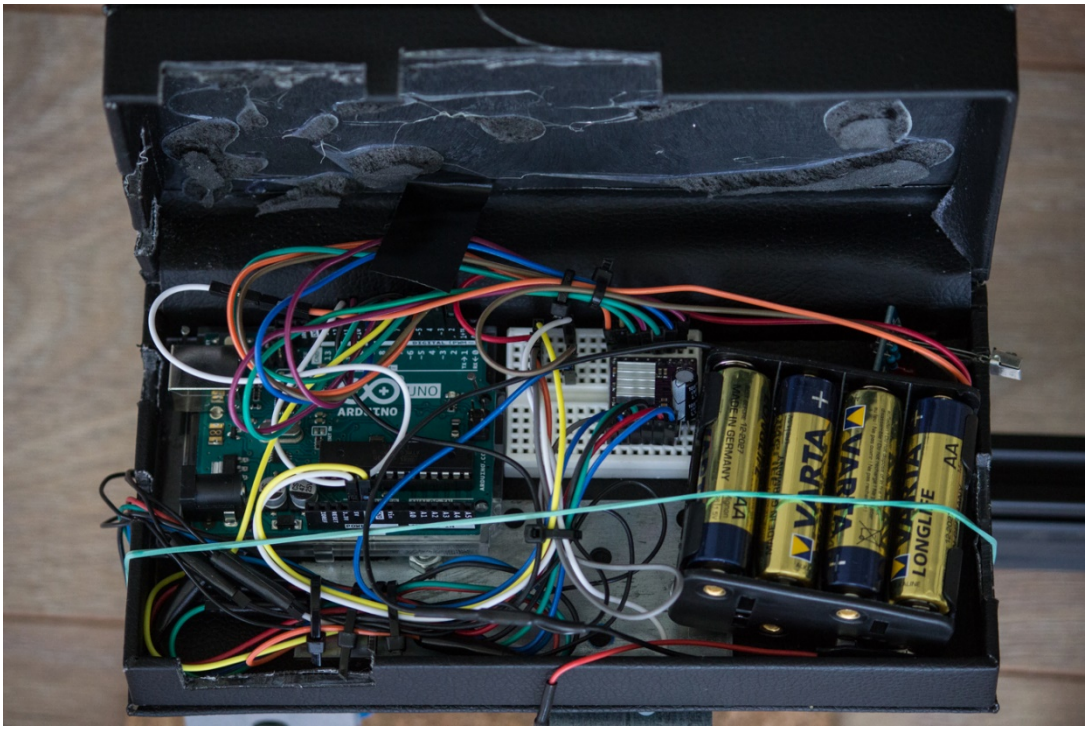

5. Krańcówki



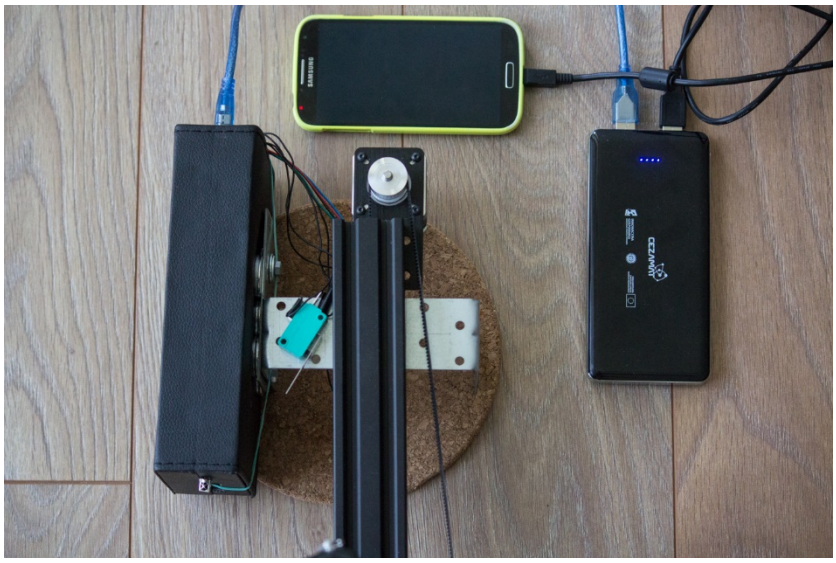


Krańcówki zostały połączone przewodem z mikrokontrolerem i przylutowane do styków przełączników oraz zabezpieczone taśmą izolacyjną. Krańcówki zostały przyklejone klejem do kątowników na jednym i drugim końcu szyny. Przewód dochodzący do pudełka na elektronikę jest schowany pod szyną i został zabezpieczony taśmą izolacyjną.

6. Pudełko na elektronikę

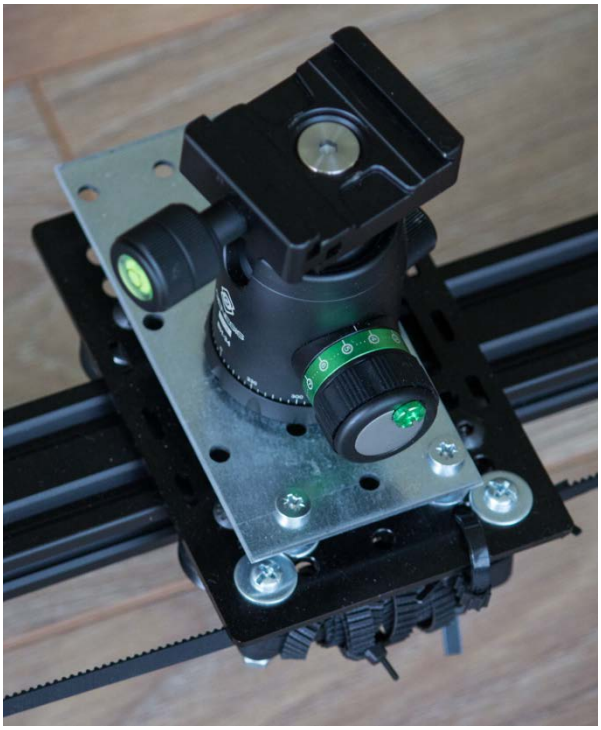



Całą elektronikę udało się upakować do niewielkiego pudełka. Poszczególne elementy zostały przyklejone bezpośrednio do pudełka, istnieje możliwość łatwego wyjęcia koszyka na baterie w celu wymiany baterii. Zasilanie Arduino zostało zrealizowane niezależnie od zasilania silnika. Przez kabel USB typu A Arduino jest podłączone do powerbanku, co zapewnia mobilną pracę całego urządzenia.



7. Głowica na płytę montażową

Do płyty montażowej głównej została przykręcona oddzielna płyta montażowa DMX, która po rozwierceniu pozwoliła na zamontowanie głowicy. Zamontowanie bezpośrednio do płyty montażowej głównej nie było możliwe, ponieważ między szyną, a płytą nie było wystarczającej ilości miejsca na mocowanie.



<u>Problemy i rozwiązania, które wystąpiły podczas montażu:</u>

- Prowadzenia paska zębatego - konieczne było zamontowanie paska w formie zamkniętej pętli, więc należało znaleźć sposób na jego lokalizację, aby jego ruch przebiegał płynnie i nie zakłócał ruchu wózka. Dzięki charakterystycznej budowie szyny udało się poprowadzić pasek w jej wnętrzu, dzięki czemu nie jest on narażony na przetarcia i uszkodzenia.
- Z uwagi na oszczędność czasu i materiałów potrzebnych na wydrukowanie dedykowanej obudowy na drukarce 3D do elementów elektronicznych, zdecydowano się na wykorzystanie używanego otwieralnego pojemnika po sprzęcie optycznym, kierunek jego otwarcia jest podyktowany potrzebą uniknięcia ryzyka kolizji pojemnika z podłożem lub wózkiem. W pudełku wykonano otwory, które mają na celu zarówno umożliwienie podłączenia zasilania modułu Arduino bez otwierania obudowy, jak również przepływ powietrza, niezbędny do odprowadzenia ciepła z radiatora sterownika.
- Niektóre elementy nie były perfekcyjnie dopasowane (zbyt grube nakrętki, aby zmieścić się w rowku szyny, brak otworów w wybranych miejscach płytek montażowych), więc konieczne było wyszlifowanie niektórych elementów i nawiercenie otworów w płytkach montażowych oraz obudowie na elektronikę.
- Do zamontowania głowicy z gwintem 3/8 cala na płytce montażowej z małymi otworami konieczne było zastosowanie podkładki oraz specjalnej przejściówki z gwintu ¼ cala na 3/8 cala, dzięki temu rozwiązaniu możliwe okazało się wykorzystanie uniwersalnej głowicy firmy Genesis, która umożliwia obrót aparatu o 360 stopni oraz jego pochylenie od -90 do +90 stopni.

**3. Programowanie**

Wszystkie wstępne założenia programistyczne udało się zrealizować (zakładana funkcjonalność programu była taka jak opisana poniżej kodu programu). Podczas testów programu nie zauważono żadnych odstępstw od założeń ani błędnego działania. Zdarza się, że użytkownik może stwierdzić błędne działanie programu, jednak może być to spowodowane nieprawidłową obsługą pilota IR, który ze względu na moc nadajnika i charakterystykę tej technologii wymaga skierowania bezpośrednio w stronę odbiornika umiejscowionego na bocznej krawędzi obudowy podzespołów elektronicznych.

Poniższy kod został opatrzony komentarzami, aby wyjaśnić działanie zastosowanych funkcji:

```
#include "PinChangeInterrupt.h"
#include "IRLremote.h"

// korzystanie z wybranych bibliotek

#define pinIR 11
#define pinLed LED_BUILTIN
#define kranL 5
#define kranP 6
#define ms1 10
#define ms2 9
#define ms3 8
#define stepPin 3
#define dirPin 4
#define sleep 2
#define res 7

// Przypisanie jednoznacznych nazw do poszczególnych wejść/wyjść w Arduino

CNec IRLremote;
int ruch = 1;
int ruch1 = 0;

// zmienne globalne odpowiadające za pamięć aktualnie oraz poprzednio
// wciśniętego przycisku na pilocie

void ruchy()
{
  if (IRLremote.available())
  {
    auto data = IRLremote.read();
    if (data.address != 65535)
    {
      ruch1 = ruch;
      ruch = data.command;
    }
  }
}

// Funkcja wczytująca kody z pilota do zmiennych globalnych
// Adres 65535 jest zwracany przy zbyt szybkim wciskaniu danego przycisku
// (lub jego przytrzymaniu)
```

```
// Aby zapobiec wykonaniu niechcianej funkcji wciśnięcia takie są ignorowane

void petla(int pin, int dela)
{
  while (digitalRead(pin) == LOW && ruch != 67 && ruch != 9 && ruch != 1)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(dela);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(dela);
    ruchy();
    if (ruch == 67 || ruch == 9)
    {
      ruch1 = ruch;
      ruch = 1;
      break;
    }
  }
}


// Funkcja realizująca ruch silnika (silnik krokowy pracuje przy szybkich
// skokach napięcia na cewkach). Funkcja ta posiada argumenty pin
// (pod tym argumentem kryje się wywołana w zależności od kierunku ruchu
// krańcówka, jeśli ruch jest w lewo to sygnał z lewej krańcówki sygnalizuje
// konieczność zatrzymania lub zmiany kierunku ruchu wózka). Przerwanie pętli
// następuje po wciśnięciu odpowiednich przycisków (pauza -> 67 lub stan
// uśpienia -> 9) lub osiągnięciu przerwania sygnału z krańcówki) oraz dela,
// który odpowiada za opóźnienie pomiędzy poszczególnymi krokami silnika,
// czyli za jego prędkość obrotową

void tryb(uint8_t m1, uint8_t m2, uint8_t m3, int del)
{
  digitalWrite(ms1, m1);
  digitalWrite(ms2, m2);
  digitalWrite(ms3, m3);
  if (ruch1 == 7 || ruch == 68)
  {
    digitalWrite(dirPin, LOW);
    petla(kranL, del);
    ruch1 = ruch;
    ruch = 1;
  }
  if (ruch1 == 21 || ruch == 64)
  {
    digitalWrite(dirPin, HIGH);
    petla(kranP, del);
    ruch1 = ruch;
    ruch = 1;
  }
  if (ruch1 == 69)
  {
    while (ruch != 67 && ruch != 9 && ruch != 1)
    {
      digitalWrite(dirPin, LOW);
      petla(kranL, del);
      digitalWrite(dirPin, HIGH);
```

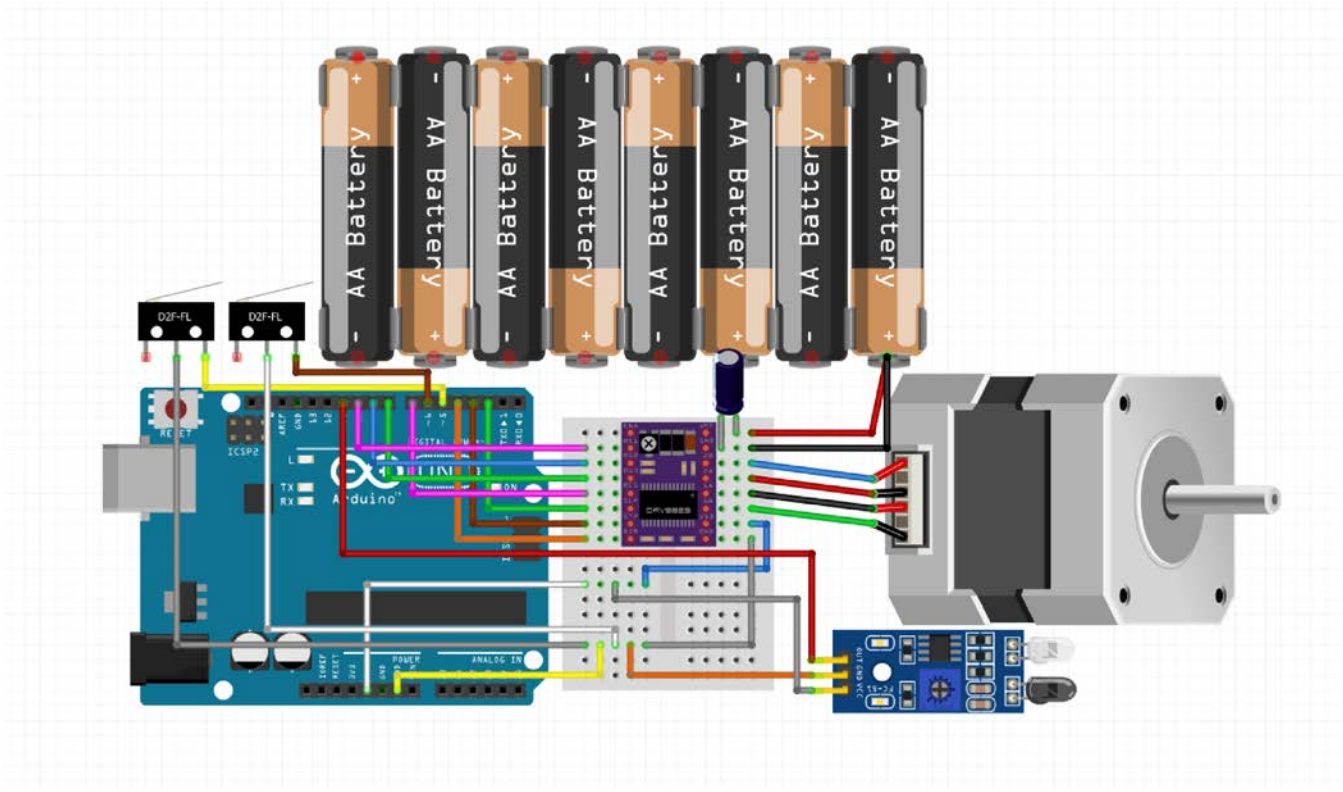
```
      petla(kranP, del);
    }
    ruch1 = ruch;
    ruch = 1;
  }
  if (ruch1 == 71)
  {
    while (ruch != 67 && ruch != 9 && ruch != 1)
    {
      digitalWrite(dirPin, HIGH);
      petla(kranP, del);
      digitalWrite(dirPin, LOW);
      petla(kranL, del);
    }
    ruch1 = ruch;
    ruch = 1;
  }
  if (ruch == 9)
    digitalWrite(sleep, LOW);

}

// Główna funkcja program, jej wywołanie musi zawierać argumenty takie jak:
// ms1, ms2, ms3 - różne kombinacje 0 i 1 na tych wyjściach powodują
// ustawienie różnych rozdzielczości kroku silnika (przykładowo przy
// wartościach 0, 0, 0 silnik na obrót o 360 st. potrzebuje 200 kroków),
// maksymalna rozdzielczość, którą sterownik może osiągnąć to 1/32 kroku
// czyli na cały obrót przypada 200*32=6400 kroków. Im większa rozdzielczość
// tym bardziej płynny ruch silnika (mniej kroków na ten sam odcinek przesuwu
// wózka). W programie regulacja prędkości jest zależna od wartości ms1, ms2,
// ms3 (rozdzielczości) oraz od opóźnienia czasowego pomiędzy krokami -
// parametr del.
// Warunki w tej funkcji sprawdzają, który tryb ruchu został wybrany (czy
// ruch od krawędzi do krawędzi, czy przejazd do krawędzi i zatrzymanie).
// Tutaj również określony jest kierunek ruchu silnika - zależny od parametru
// dirPin.

void setup()
{
  pinMode(pinLed, OUTPUT);
  pinMode(ms1, OUTPUT);
  pinMode(ms2, OUTPUT);
  pinMode(ms3, OUTPUT);
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
  pinMode(sleep, OUTPUT);
  pinMode(res, OUTPUT);
  digitalWrite(res, HIGH);
  pinMode(kranL,  INPUT_PULLUP);
  pinMode(kranP,  INPUT_PULLUP);
  if (!IRLremote.begin(pinIR))
    Serial.println(F("You did not choose a valid pin."));
}

// W tej funkcji deklarowane jest, które piny w Arduino są wyjściami,
// wejściami lub przyciskami (dla krańcówek parametr - INPUT_PULLUP). Również
// tutaj znajduje się polecenie inicjalizujące odczyt danych z pilota.
```

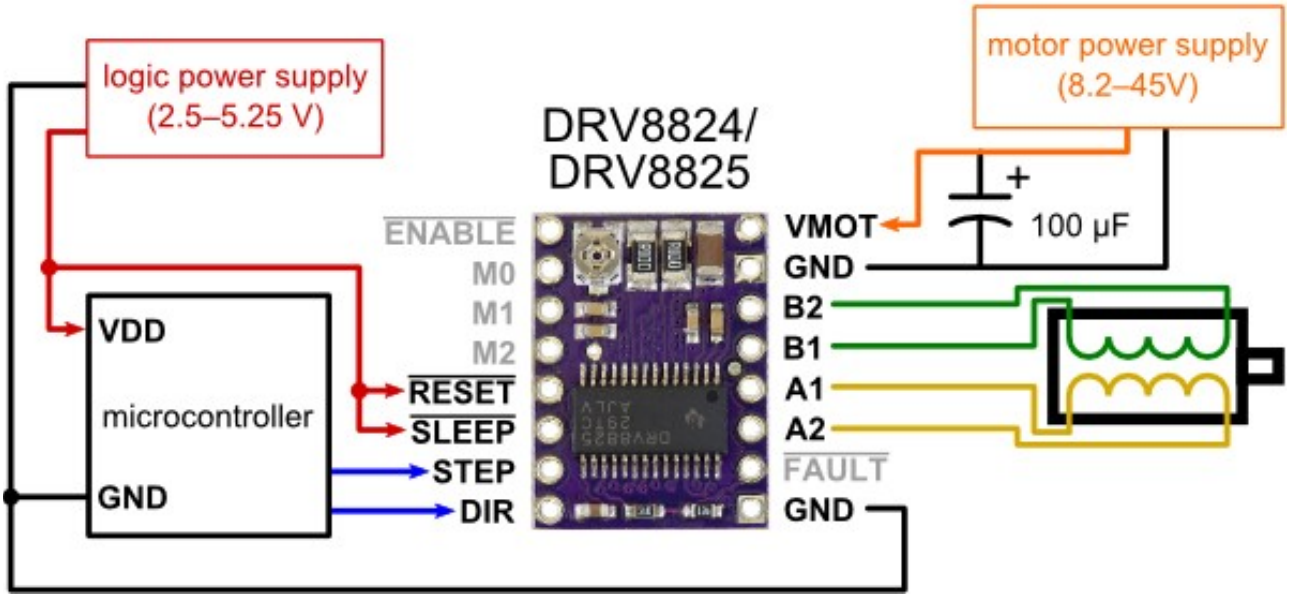
```
void loop()
{
  digitalWrite(stepPin, LOW);
  ruchy();
  if (ruch == 9 || ruch == 1)
    digitalWrite(sleep, LOW);
  else digitalWrite(sleep, HIGH);
  if (ruch == 12 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(LOW, LOW, LOW, 1000);
  if (ruch == 24 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(HIGH, LOW, LOW, 1000);
  if (ruch == 94 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(LOW, HIGH, LOW, 1000);
  if (ruch == 8 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 == 21))
tryb(HIGH, HIGH, LOW, 1000);
  if (ruch == 28 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(LOW, LOW, HIGH, 1000);
  if (ruch == 90 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(HIGH, LOW, HIGH, 1000);
  if (ruch == 66 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(HIGH, LOW, HIGH, 2000);
  if (ruch == 82 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(HIGH, LOW, HIGH, 4000);
  if (ruch == 74 && (ruch1 == 69 || ruch1 == 71 || ruch1 == 7 || ruch1 ==
21)) tryb(HIGH, LOW, HIGH, 8000);
  if (ruch == 68) tryb(LOW, LOW, LOW, 500);
  if (ruch == 64) tryb(LOW, LOW, LOW, 500);
}

// W tej pętli wywoływane są omówione wcześniej funkcje zależnie od wybranych
// przycisków (regulacja kierunkiem ruchu, trybem działania i prędkością), w
// wywołaniach podawane są takie informacje jak rozdzielczość i opóźnienie
```

Problemy i rozwiązania, które wystąpiły podczas pisania programu:

- Poniższy kod w wersji pierwotnej zajmował o wiele więcej linijek, jednak dzięki stworzeniu dedykowanych 3 funkcji, a następnie jedynie wywoływaniu ich wraz z odpowiednimi argumentami udało się uprościć i skrócić kod.
- Podczas programowania największym problemem okazało się wykorzystanie bibliotek odpowiadających za przetwarzanie sygnału IR z pilota. Najbardziej popularna biblioteka – IRremote nie spełniła oczekiwań, ponieważ kody odczytywane po naciskaniu jednego przycisku na pilocie były za każdym razem inne. Jednak udało się znaleźć inną bibliotekę obsługującą sygnał IR – IRLremote (autor: Nico Hood), która bezbłędnie i jednoznacznie rozpoznawała poszczególne przyciski.
- Zastosowane podzespoły umożliwiają praktycznie nieograniczone możliwości wykorzystania takiego slidera w zależności od zaprogramowanego kodu, niewielkie zmiany mogłyby umożliwić np. wykonywanie pewnych sekwencji ruchów, dowolne prędkości (w zakresie pracy silnika), płynne poruszanie się (ruszania i hamowania).

**4. Schemat układu elektronicznego**




Silnik bipolarny (4 przewody po 2 na cewkę) podłączamy do sterownika kolejno do portów 2B, 2A, 1A, 1B. Zasilanie silnika 12 V jest realizowane przez 8 baterii AA (1,5V) połączonych szeregowo do wejść VMT (plus) i GND (masa) z dołączonym równolegle kondensatorem (100uF) w celu wyeliminowania skoków napięcia. Pin DIR w sterowniku odpowiada za kierunek pracy silnika i sterowany jest wyjściem cyfrowym nr 4. Szybkie zmiany stanów na pinie STP (wyjście nr 3 na Arduino) powodują płynną pracę silnika

krokowego. Pin SLP jest zrealizowany połączeniem do wyjścia nr 2 w Arduino, niski stan powoduje uśpienie sterownika silnika, co pozwala oszczędzić energię. Pin RES jest w stanie wysokim (z założenia kodu oprogramowania Arduino), ponieważ stan niski powodowałby resetowanie ustawień prędkości silnika do wartości początkowych (piny MS1, MS2, MS3).

| MODE0 | MODE1 | MODE2 | Rozdzielczość |
|---|---|---|---|
| Low | Low | Low | Pełny krok |
| **High** | Low | Low | 1/2 kroku |
| Low | **High** | Low | 1/4 kroku |
| **High** | **High** | Low | 1/8 kroku |
| Low | Low | **High** | 1/16 kroku |
| **High** | Low | **High** | 1/32 kroku |
| Low | **High** | **High** | 1/32 kroku |
| **High** | **High** | **High** | 1/32 kroku |

Kombinacja pinów MS1, MS2, MS3 odpowiada za różną rozdzielczość kroku silnika.
Moduł odbioru fal IR HX1838 podłączony jest do zasilania 5V z Arduino, masy oraz do wejścia cyfrowego nr 11. Krańcówki MSW-02-38 (typu ON-(ON)) podłączone są do wejść cyfrowych nr 5 i 6 oraz masy. Arduino jest zasilane wejściem USB typu A za pomocą powerbanku lub laptopa/komputera, co zapewnia mobilność układu.

<u>Problemy i rozwiązania, które wystąpiły podczas realizacji układu elektronicznego:</u>

- Aby ograniczyć wpływ niechcianych skoków napięcia lub niestabilności źródła zasilania należało zastosować dodatkowy kondensator elektrolityczny o pojemności 100 µF pomiędzy plusem a minusem źródła zasilania
- Aby nie zniszczyć płytki sterownika i silnika, który w danych technicznych podane ma maksymalne natężenie prądu na cewkę należało za pomocą potencjometru umieszczonego na sterowniku dobrać odpowiednie napięcie referencyjne za pomocą multimetru sprawdzając różne wartości przy różnych położeniach potencjometru

**5. Opis działania slidera**


Slider posiada 9 prędkości (1 - najszybciej, 9 - najwolniej). Zmiana regulacji prędkości jest realizowana przez odpowiednio przyciski od 1 do 9 na pilocie. Po wybraniu prędkości należy określić kierunek ruchu slidera.

Ruch slidera w lewą stronę: CH– oraz –.

Ruch slidera w prawą stronę: CH+ oraz +.

Różnica w działaniu przycisków CH–, CH+ oraz –, + polega na tym, że przyciski CH– lub CH+ wywołują tryb ruchu od krawędzi do krawędzi - wózek zacznie się poruszać i po dojechaniu do końca (zetknięciu z krańcówką) automatycznie zacznie poruszać się w przeciwną stronę. Natomiast używając przycisków – i + wózek po dojechaniu do krańca zatrzyma się.

Przykładowo, aby zmusić slider do ruchu w lewo z maksymalną prędkością, tak aby po dojechaniu do krańcówki zmienił kierunek i zaczął poruszać się w prawo, należy nacisnąć CH–, a następnie 1.

Przyciski << oraz >> służą do jak najszybszego ustawienia wózka slidera na jednym z końców (lewym lub odpowiednio prawym). Przycisk PLAY/PAUZA służy do zatrzymania slidera w dowolnym momencie jego ruchu. Przycisk EQ służy do przejścia sterownika silnika w stan uśpienia (należy to robić, aby oszczędzać energię).

**6. Ograniczenia**

Maksymalna siła osiowa silnika to 10N. Zakładamy, że slider przystosowany jest do małych aparatów do 1kg. Testów praktycznych z granicznym obciążeniem, jak również z ustawieniem slidera innym niż poziome, nie przeprowadzaliśmy ze względu na obawy przed uszkodzeniem urządzenia. Wszelkie ograniczenia są zależne od mocy silnika oraz od naciągu paska zębatego, jak również od wydajności sterownika (i jego chłodzenia) i źródła zasilania. Naciąg paska został dobrany do optymalnej pracy silnika na podstawie testów praktycznych z większym i mniejszym naciągiem, przy poziomym ustawieniu slidera.

**7. Podsumowanie**

Wszystkie wstępne założenia projektu udało nam się zrealizować. Slider uważamy za funkcjonalny i gotowy do realizacji przejazdów kamerą średniej wielkości oraz do wykonywania timelapsów aparatem fotograficznym.