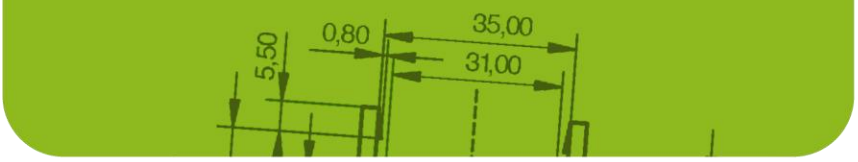
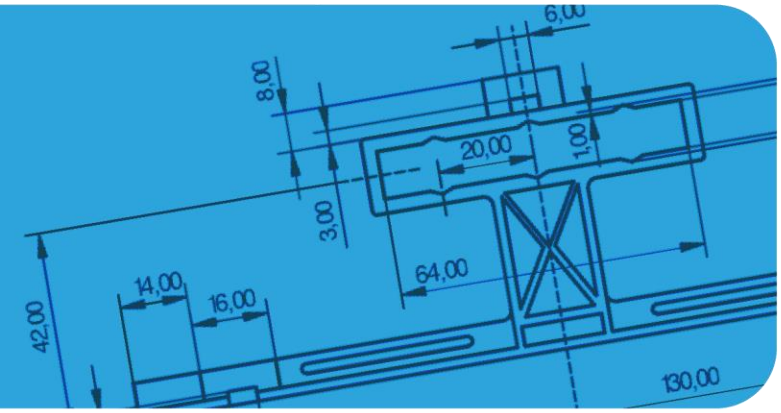
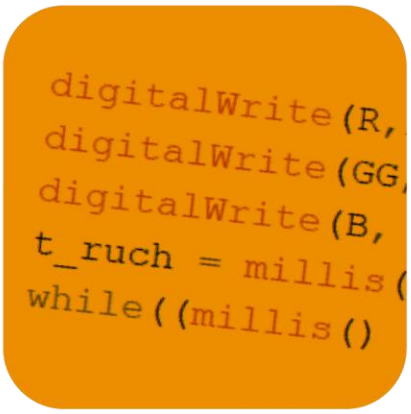






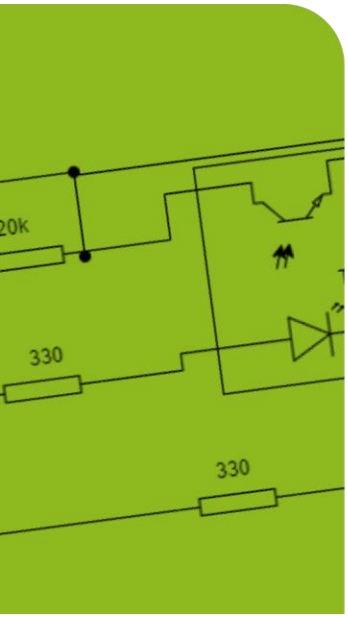
PROJEKT

Wojciech Kolarz

SYGNALIZACJA ŚWIETLNA

DIY

MODELOWANIE 3D
PROGRAMOWANIE
PODSTAWY ELEKTRONIKI



CENTRUM KISS

Materiał dodatkowy do filmu SYGNALIZACJA ŚWIETLNA.

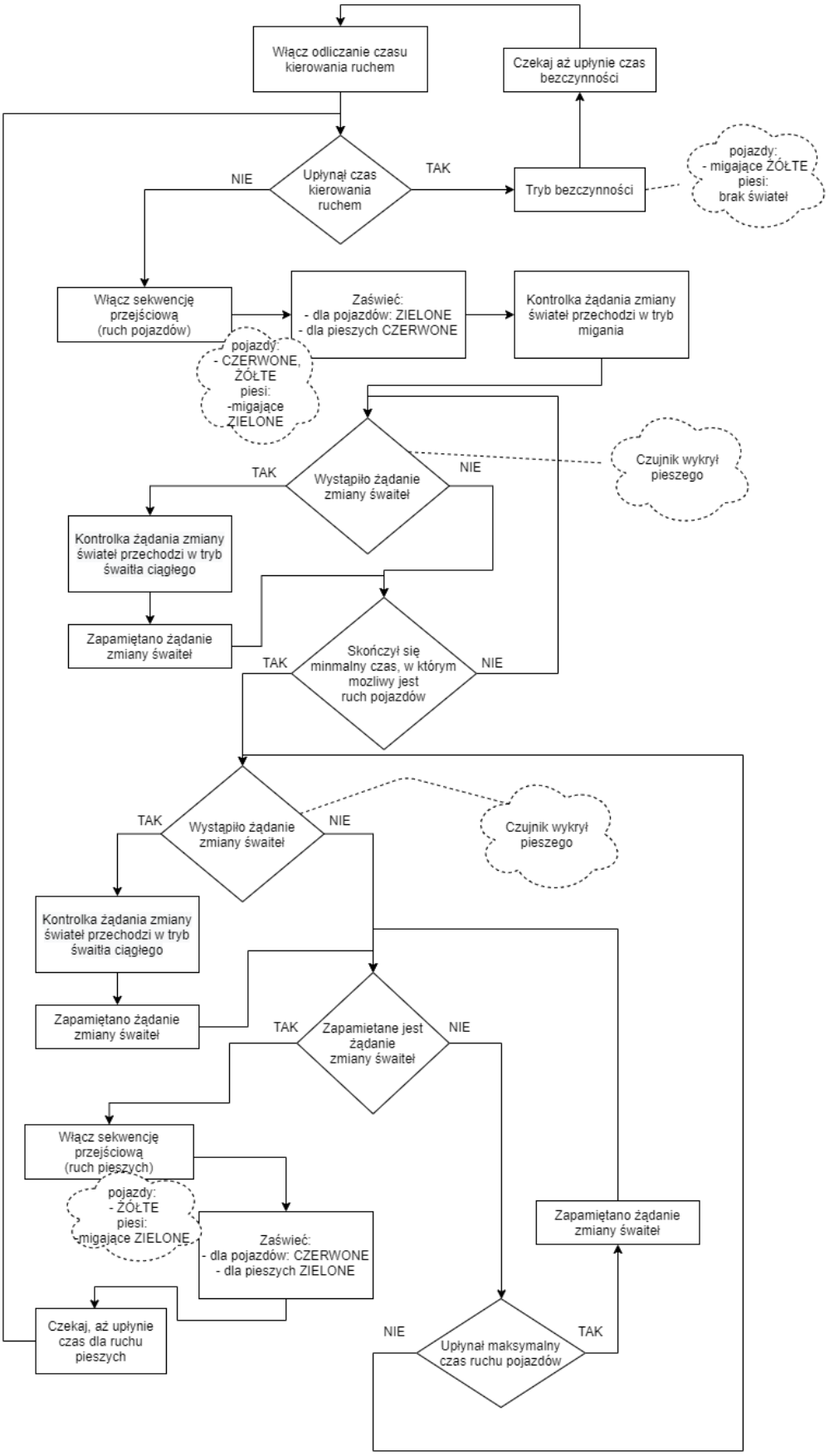
Podzespoły i materiały użyte do budowy sygnalizacji:

- Eementy drukowane: filament PLA wypełnienie dowolne (w prezentowanym modelu 20%), ustawienia typowe dla dyszy 0,4 mm, grubość warstwy 0,2 mm.
- Arduino UNO R3
- Diody LED 3mm:
  - czerwone – 4 szt.
  - zielone – 4 szt.
  - żółte – 2 szt.
  - niebieskie – 2 szt.
- Transoptor odbiciowy CNY 70 – 2 szt.
- Rezystory (ewentualnie dobrać w zależności od zastosowanych diod)
  - 220 Ω - 2 szt.
  - 330 Ω - 2 szt.
  - 470 Ω - 2 szt.
  - 20 kΩ - 1 szt.
- Farby modelarskie.
- Klej cyjanoakrylowy, klej na gorąco.
- Przewody połączeniowe – cienkie (0,2 mm) w izolacji.

Założenia do projektu.

1. Ruchem pojazdów kierują dwa sygnalizatory. Każdy sygnalizator wyposażony jest w światła o trzech kolorach (czerwone, żółte, zielone).
2. Ruchem pieszych kierują dwa sygnalizatory. Każdy sygnalizator wyposażony jest w światła o dwóch kolorach (czerwone, zielone).
3. Pieszy ma możliwość przyspieszenia zmiany świateł wykorzystując czujnik zbliżeniowy. Sygnalizacja wyposażona jest w dwa takie czujniki.
4. Zmiana świateł nastąpi nie wcześniej niż po upływie minimalnego czasu, w którym umożliwiony jest ruch pojazdów.
5. Stan aktywności czujnika zbliżeniowego, oraz stan rozpoznania pieszego sygnalizowane są poprzez kontrolkę (światełko w kolorze niebieskim).
6. Po określonym czasie sygnalizacja zmienia tryb działania – tryb bezczynności (migające żółte dla ruchu pojazdów). Tryb ten również ma określony czas działania, po którym sygnalizacja zmienia tryb pracy na kierowanie ruchem.

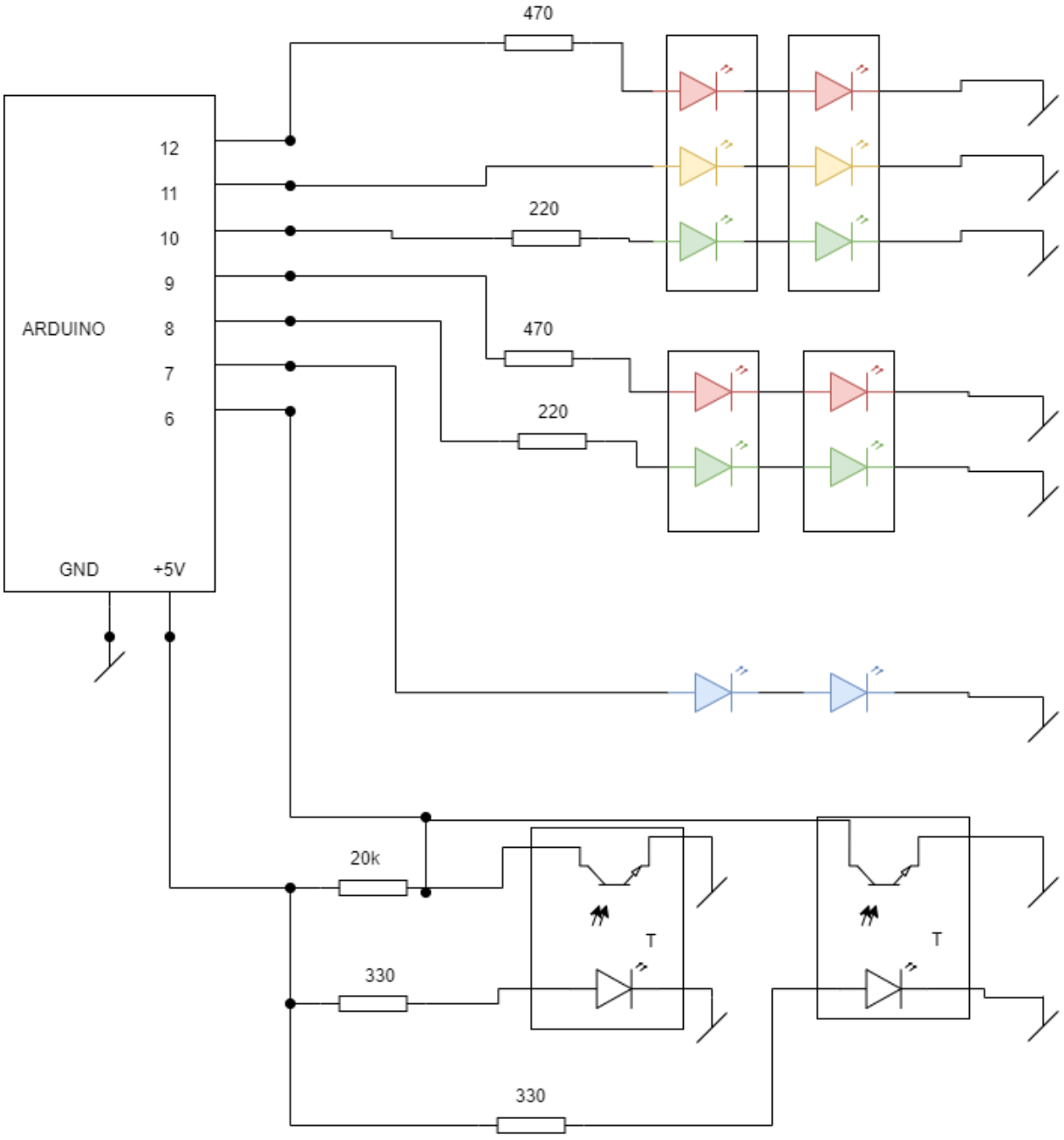
Algorytm działania sygnalizacji.



Model sygnalizacji świetlnej można wykonać konstruując tylko układ elektroniczny lub jako model przestrzenny korzystając z elementów wydrukowanych w technologii 3D.

Niezależnie od wyboru wariantu wykonania modelu dobrze będzie jeśli część elektroniczną układu najpierw zmontujemy na płytce prototypowej (stykowej). Umożliwi to przetestowanie układu, ewentualne dobranie wartości rezystorów, oraz stworzenie programu sterującego sygnalizacją.

Schemat:



Tworzenie rozwiązania problemu (programu) podzielone zostanie na mniejsze podproblemy (zgodnie z zasadą myślenia komputacyjnego).

Tworzenie programu rozpoczniemy od zdefiniowania nazw pinów i deklaracji zmiennych (deklaracje zmiennych mogą zostać zmienione w późniejszej fazie tworzenia programu).

```
#define R 12 // czerwone - pojazdy
#define Y 11 // żółte - pojazdy
#define G 10 // zielone - pojazdy
#define RR 9 // czerwone - piesi
#define GG 8 // zielone - piesi
#define B 7 // niebieska - kontrolka żądania zmiany świateł
#define T 6 // sygnał z czujnika - transoptor
unsigned long t_ruch; // czas rozpoczęcia cyklu sterowania ruchem
                      // lub czas rozpoczęcia cyklu bezczynności
unsigned long t_ruch_pojazdy; // czas rozpoczęcia ruchu pojazdów
unsigned long t_dzialania = 60000; // czas, w którym działa sterowanie ruchem
                                   // czas ustawiony na 60 sekund
unsigned long t_bezczynnosci = 15000; // czas bezczynności (miga żółte dla pojazdów)
                                      // czas ustawiony na 15 sekund
unsigned long t_min_pojazdy = 10000; // minimalny czas sygnału zielonego dla pojazdów (10 sekund)
unsigned long t_max_pojazdy = 30000; // maksymalny czas sygnału zielonego dla pojazdów (30 sekund)
```

Następnie utworzymy procedurę **setup()**, w której zdefiniujemy wejścia i wyjścia.

```
void setup()
{
// usatwienie wyjść (diody) i wejścia (transoptor)
  pinMode(R, OUTPUT);
  pinMode(Y, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(RR, OUTPUT);
  pinMode(GG, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(T, INPUT);
}
```

W działaniu sygnalizacji świetlnej można wyróżnić dwie główne funkcjonalności:

- sekwencję zaświeceń i wygaszeń świateł kończącą ruch pieszych i rozpoczynającą ruch pojazdów;
- sekwencję zaświeceń i wygaszeń świateł kończącą ruch pojazdów i rozpoczynającą ruch pieszych.

Dla przejrzystości programu sekwencje te realizowane będą przez procedury.

Pierwsza procedura kończąca ruch pieszych i rozpoczynająca ruch pojazdów może prezentować się następująco

```
// procedura kończąca stan sygnalizacji zezwalający na ruch pieszych
// i rozpoczynajaca stan sygnalizacji zezwalający na ruch pojazdów
void pojazdy()
{
  // zielone dla pieszych miga 5 razy w odstępach 0,8 sekundy
  for(int i = 1; i<6; i++)
  {
    digitalWrite(GG, HIGH);
    delay(800);
    digitalWrite(GG, LOW);
    delay(800);
  }
  digitalWrite(RR, HIGH); //czerwone dla pieszych
  delay(800); // po czasie 0,8 sekundy
  digitalWrite(Y, HIGH); //żółte dla pojazdów
  delay(800); // po 0,8 sekundy
  digitalWrite(R, LOW); // wygaszone czerwone dla pojazdów
  digitalWrite(Y, LOW); // wygaszone żółte dla pojazdów
  digitalWrite(G, HIGH); // zielone dla pojazdów
 }
```

Procedurę można od razu przetestować wywołując ją w głównej pętli programu.

```
30 void loop()
31 {
32     pojazdy();
33 }
```

W analogiczny sposób możemy stworzyć procedurę kończącą ruch samochodów i rozpoczynająca ruch pieszych.

```
58 // procedura kończąca stan sygnalizacji zezwalający na ruch pojazdów
59 // i rozpoczynajaca stan sygnalizacji zezwalający na ruch pieszych
60  void piesi(){
61   digitalWrite(G, LOW);
62   digitalWrite(Y,HIGH);
63   delay(800);
64   digitalWrite(Y, LOW);
65   digitalWrite(R, HIGH);
66   delay(800);
67   digitalWrite(RR,LOW);
68   digitalWrite(GG,HIGH);
69 }
```

Po umieszczeniu w pętli głównej wywołania procedury piesi(), otrzymamy prawidłowo działającą prostą sygnalizację (bez możliwości ingerencji w kolejne cykle zmian świateł).

```
29 void loop()
30 {
31     pojazdy();
32     delay(2000); //do usunięcia
33     piesi();
34     delay(2000); // zielone dla pieszych
35 }
```

Ponieważ każda z procedur (**pojazdy()** i **piesi()**) kończy się zapaleniem odpowiedniej konfiguracji świateł, wystarczy wykonywanie tych procedur rozdzielić funkcją **delay()**. W przykładzie zastosowano wstrzymanie działania programu (**delay()**) na 2 sekundy. Oczywiście czas ten nie jest adekwatny do czasu w jakim w rzeczywistości świeca się poszczególne konfiguracje świateł. W przypadku czasu, w trakcie którego umożliwiać będziemy ruch pieszych nie wystąpią żadne zdarzenia wpływające na konieczność zmiany świateł, dlatego można w tym przypadku skorzystać z funkcji **delay()** – ta część programu nie ulegnie zmianie podczas dalszej jego rozbudowy. Natomiast zgodnie z wcześniejszymi założeniami, podczas ruchu pojazdów może nastąpić żądanie zmiany świateł, dlatego funkcja **delay()** znajdująca się po wywołaniu procedury **pojazdy()** zostanie zastąpiona ciągiem instrukcji, zanim to jednak nastąpi, doprogramu wprowadzimy sekwencję poleceń realizującą funkcjonowanie sygnalizacji w trybie kierowania ruchem oraz w trybie bezczynności.

W trakcie kierowania ruchem pojazdów i pieszych, powinien być sprawdzany stan czujnika wykrywającego żądanie zmiany świateł. Z tego powodu jedną z funkcji **delay()** należy zastąpić pętlą **while()**. Instrukcje zawarte w tej pętli będą wykonywany do momentu, aż upłynie czas kierowania ruchem (po tym czasie sygnalizacja przejdzie w tryb bezczynności).

Przed instrukcją pętli **while()**, w zmiennej **t-ruch** zapamiętany zostanie aktualny czas, a wnętrze pętli będzie wykonywane tak długo, dopóki aktualne czas pomniejszony o czas zapamiętany nie przekroczy wartości zmiennej **t_dzialania**. W analogiczny sposób zapisane zostaną instrukcje realizowane w czasie bezczynności.

```
void loop()
{
    // stan działania
    t_ruch = millis(); // odczyt bieżącego czasu
    while((millis() - t_ruch) < t_dzialania)
    {
      pojazdy();
      delay(2000); // do usunięcia
      piesi();
      delay(2000); // zielone dla pieszych
    }
    // stan bezczynności
    // wygaszenie wszystkich zielonych i czerwonych
    digitalWrite(R,LOW);
    digitalWrite(GG,LOW);
    t_ruch = millis(); // odczyt bieżącego czasu
    while((millis() - t_ruch) < t_bezczynnosci)
      bezczynnosc();
}
```

W drugiej pętli pojawiła się procedura **bezczynność()**, która ma postać:

```
// procedura stanu bezczynności - migające żółte
void bezczynnosc()
{
  delay(800);
  digitalWrite(Y, LOW);
  delay(800);
  digitalWrite(Y,HIGH);
}
```

Teraz możemy się zająć zastąpieniem funkcji **delay()** znajdującej się po wywołaniu procedury **pojazdy()**. W trakcie, gdy sygnalizacja umożliwiać będzie ruch pojazdów, jak już wspomniano, konieczne będzie odczytywanie stanu czujnika stwierdzającego wystąpienie żądania zmiany świateł. Ponadto czas, gdy sygnalizacja umożliwiać będzie ruch pojazdów, należy podzielić na dwie części – zgodnie z założeniami: czas, w którym może zostać wykryte żądanie zmiany świateł, ale światła nie zostaną zmienione, oraz czas, w którym może nastąpić wykrycie żądania zmiany świateł – światła zostaną zmienione niezależnie od tego, czy żądanie zmiany świateł nastąpiło wcześniej, czy w danym momencie.

Wspomniane funkcjonalności sygnalizacji zostaną zrealizowane poprzez dodanie dwóch pętli **while()** wykonujących się do momentu przekroczenia ustalonych wcześniej czasów – analogicznie jak pisano wcześniej. Pętle te będą znajdowały się wewnątrz pętli „odliczającej” czas działania sygnalizacji w trybie kierowania ruchem.

```
29 void loop()
30 {
31     // stan działania
32     t_ruch = millis(); // odczyt bieżącego czasu
33     while((millis() - t_ruch) < t_dzialania)
34     {
35       pojazdy();
36       t_ruch_pojazdy = millis();
37       while((millis() - t_ruch_pojazdy) < t_min_pojazdy) // odliczanie minimalnego czasu
38                                                          // dla pojazdów
39       {
40
41       }
42       while((millis() - t_ruch_pojazdy) < t_max_pojazdy) // odliczanie maksymalnego czasu
43                                                          // dla pojazdów
44       {
45
46       }
47       piesi();
48       delay(2000); // zielone dla pieszych
49     }
```

Wewnątrz pętli należy umieścić instrukcję warunkową **if()**, która, zależności od stanu czujnika ustawiać będzie zmienną **zmien_swiatla** na **true** (konieczne będzie zadeklarowanie tej zmiennej jako zmiennej typu **boolean**). Ponadto działanie drugiej, wewnętrzne pętli **while()** zostanie przerwane gdy zmienna **zmien_swiatla** będzie miała wartość **true**. Po wykonaniu procedury ustawiającej stan sygnalizacji umożliwiający ruch pieszych zmienna **zmien_swiatla** zostanie ustawiona na **false**.

```
17 boolean zmien_swiatla = false; // zmienna przyjmie wartość true jeśli nastąpi
18                                // żądanie zmiany świateł
```

```
31 void loop()
32 {
33     // stan działania
34     t_ruch = millis(); // odczyt bieżącego czasu
35     while((millis() - t_ruch) < t_dzialania)
36     {
37       pojazdy();
38       t_ruch_pojazdy = millis();
39       while((millis() - t_ruch_pojazdy) < t_min_pojazdy) // odliczanie minimalnego czasu
40                                                          // dla pojazdów
41       {
42           if(not(digitalRead(T))) zmien_swiatla = true;
43       }
44       while((millis() - t_ruch_pojazdy) < t_max_pojazdy) // odliczanie maksymalnego czasu
45                                                          // dla pojazdów
46       {
47           if(not(digitalRead(T))) zmien_swiatla = true;
48           if(zmien_swiatla) break;
49       }
50       piesi();
51       zmien_swiatla = false;
52       delay(2000); // zielone dla pieszych
53     }
```

Pozostaje jeszcze wzbogacenie programu o sekwencję poleceń sterujących kontrolką. W tym celu wprowadzone zostaną dwie nowe zmienne:

```
19 boolean niebieski = false;  //jeśli true - kontrolka świeci, false - wygaszona
20 unsigned long t_niebieski=200;  //czas świecenia/wygaszenia kontrolki
```

Utworzona zostanie procedura sterująca kontrolką:

```
109 // procedura sterująca kontrolką
110 void kontrolka()
111 {
112    if (zmien_swiatla)
113       niebieski = true;
114    else
115    {
116       if((millis()-t_niebieski) > 500)
117       {
118         niebieski = not(niebieski);
119         t_niebieski = millis();
120       }
121    }
122    digitalWrite(B, niebieski);
123 }
```

Część główna programu zostanie wzbogacona o instrukcje:

```
35    // stan działania
36    t_ruch = millis(); // odczyt bieżącego czasu
37    while((millis() - t_ruch) < t_dzialania)
38    {
39      pojazdy();
40      t_ruch_pojazdy = millis();
41      while((millis() - t_ruch_pojazdy) < t_min_pojazdy) // odliczanie minimalnego czasu
42                                                         // dla pojazdów
43      {
44          if(not(digitalRead(T))) zmien_swiatla = true;
45          kontrolka();
46      }
47      while((millis() - t_ruch_pojazdy) < t_max_pojazdy) // odliczanie maksymalnego czasu
48                                                         // dla pojazdów
49      {
50          if(not(digitalRead(T))) zmien_swiatla = true;
51          kontrolka();
52          if(zmien_swiatla) break;
53      }
54      piesi();
55      zmien_swiatla = false;
56      delay(2000); // zielone dla pieszych
57    }
```

Przed częścią programu realizującą stan bezczynności trzeba dodać instrukcje:

```
58    // stan bezczynności
59    // wygaszenie wszystkich zielonych i czerwonych
60    digitalWrite(R,LOW);
61    digitalWrite(GG,LOW);
62    digitalWrite(B, LOW);
63    t_ruch = millis(); // odczyt bieżącego czasu
64    while((millis() - t_ruch) < t_bezczynnosci)
65      bezczynnosc();
```

W programie na sztywno założono czas działania sygnalizacji w trybie ruchu pieszych, częstotliwości działania migającego światła zielonego, żółtego, kontrolki itp.

Na koniec warto nieco zmodyfikować program: wspomniane powyżej czasy można zastąpić zmiennymi i w momencie ich inicjalizacji podać żądane wartości, jak również można program poddać innym kosmetycznym zmianom.

Do pobrania – poszczególne programy tworzone zgodnie z powyższym opisem.

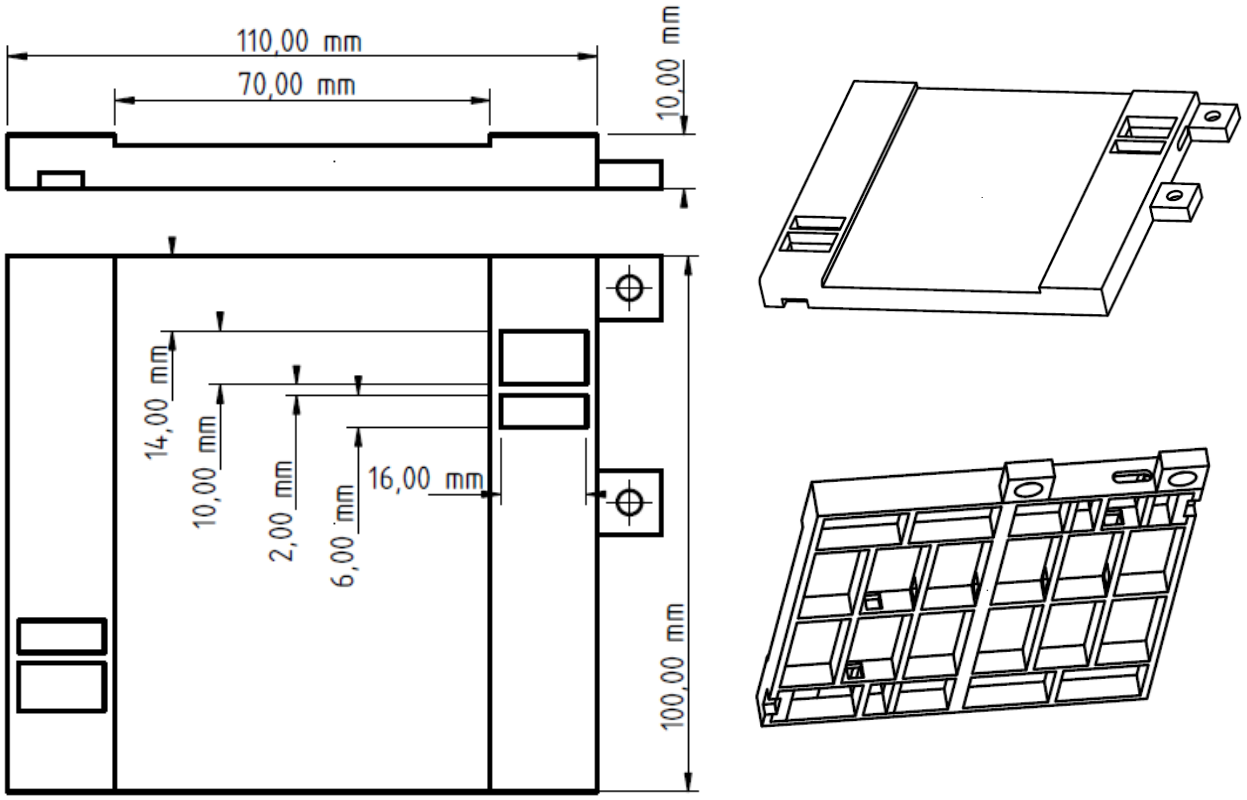
Konstrukcja – druk 3D.

Jeśli model sygnalizacji świetlnej, zbudowanej na płytce prototypowej działa poprawnie, można pokusić się zbudowanie kompletnego modelu, z elementów stworzonych w technologii 3D.

Przedstawiony model i proponowane pliki **.stl** stanowią propozycję, każdy może samodzielnie zaprojektować elementy konstrukcyjne tworząc własną postać sygnalizacji.

W przedstawionej poniżej konstrukcji zastosowano diody LED o średnicy 3mm, całość została wydrukowana z użyciem typowego filamentu PLA w kolorze białym, a następnie pomalowana akrylowymi farbami modelarskimi. Na rysunkach podano najistotniejsze wymiary.

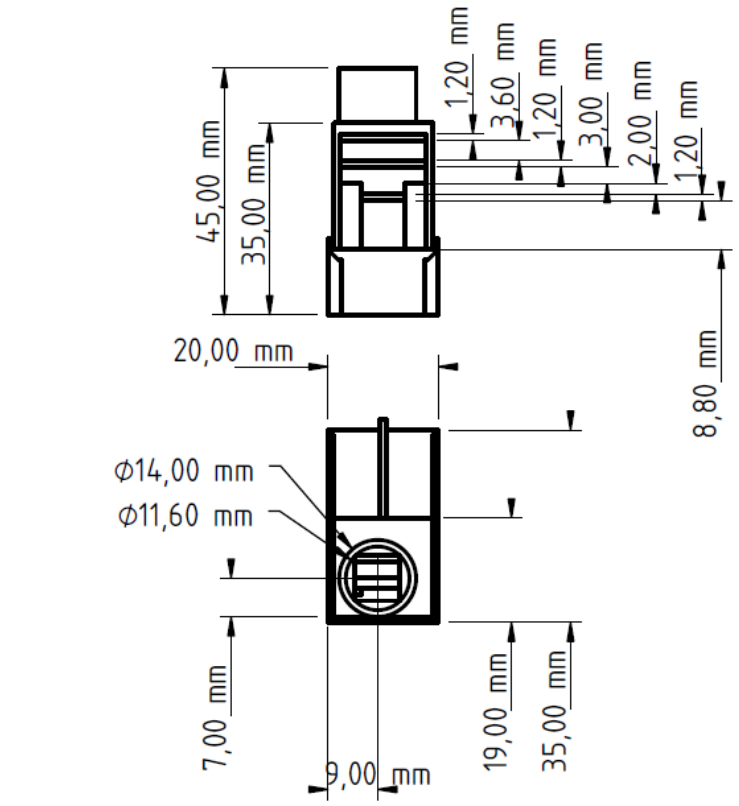
Model przejścia dla pieszych z sygnalizacją świetlną składa się z podstawki:

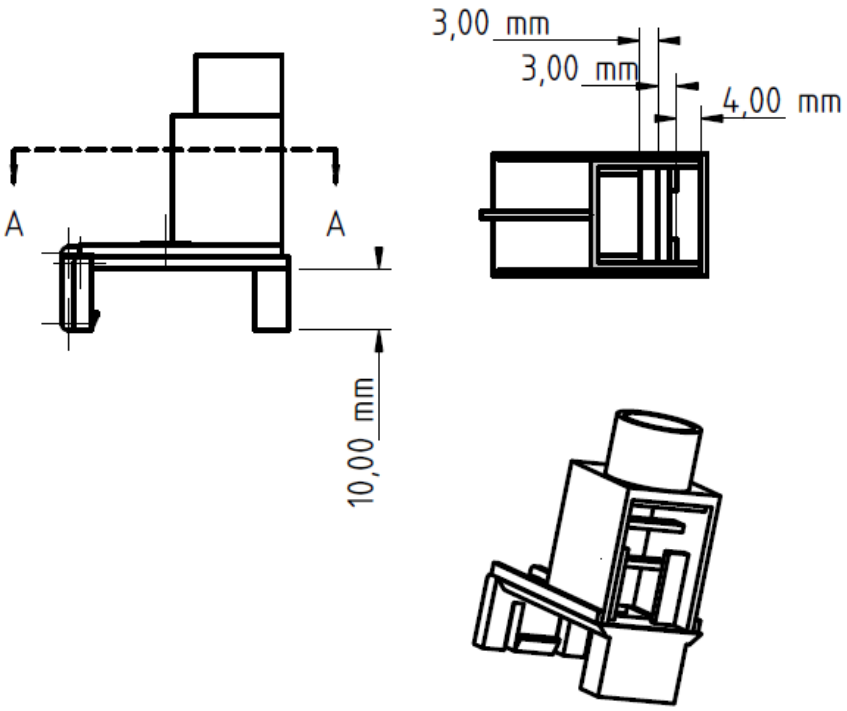


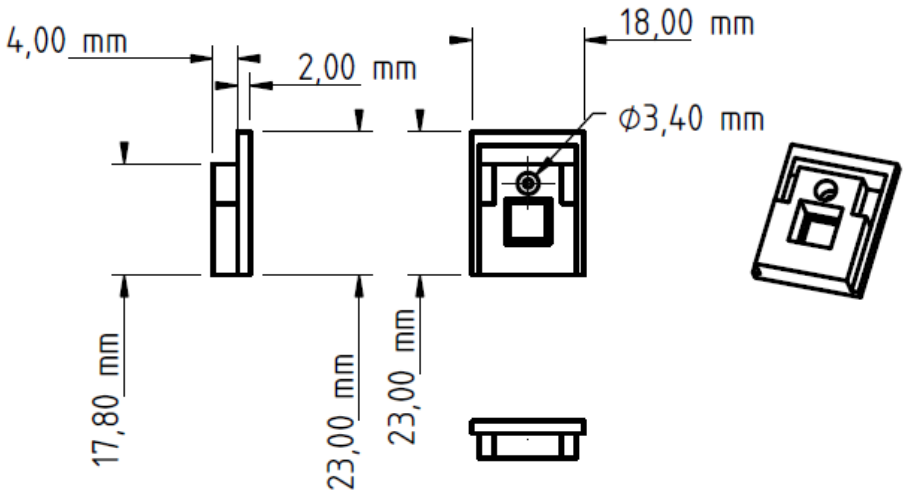
Występy z otworami znajdujące się po lewej stronie służą do zamocowania płytki z niezbędnymi elementami elektronicznymi (rezystorami), oraz złączem umożliwiającym podpięcie całości do Arduino. Otwory i wgłębienia służą do zamocowania kolejnych elementów konstrukcji oraz do przeprowadzenia przewodów.

Podstawka może być wykonana tak jak powyżej lub jako lustrzane odbicie, w zależności układu modelu.

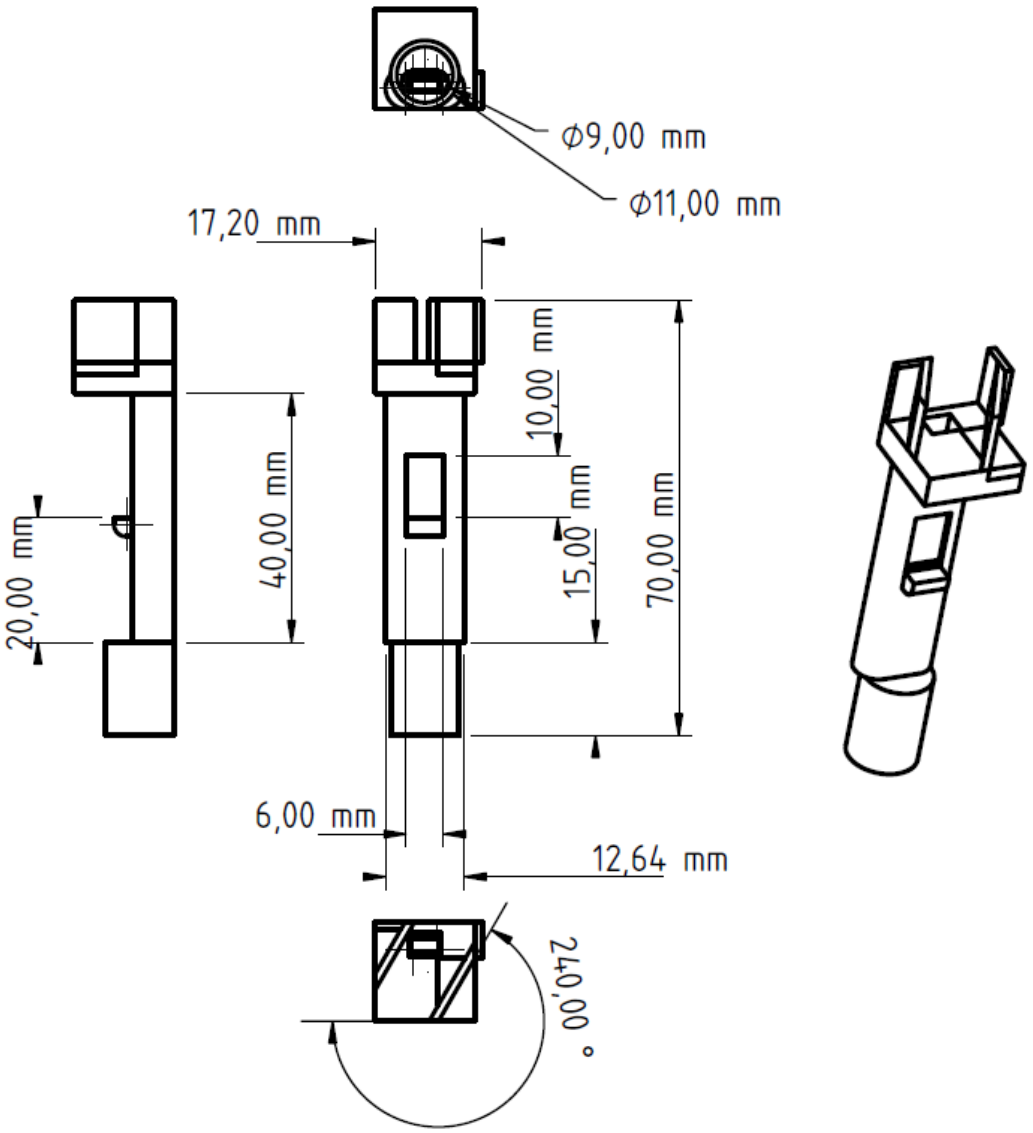
Na podstawce zainstalowane są podstawy słupków sygnalizacji. Elementy te są najbardziej skomplikowanymi częściami modelu. W podstawach słupków zainstalowane zostaną czujniki i diody sygnalizacyjne. Podstawy słupków składają się z dwóch części – części podstawowej i nakładki.


1,20 mm
3,60 mm
1,20 mm
3,00 mm
2,00 mm
1,20 mm
45,00 mm
35,00 mm
8,80 mm
20,00 mm
Φ14,00 mm
Φ11,60 mm
7,00 mm
9,00 mm
19,00 mm
35,00 mm

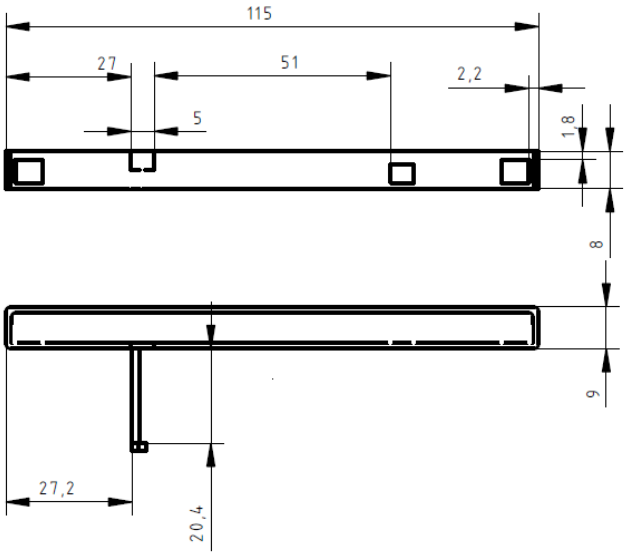

3,00 mm
3,00 mm
4,00 mm
A
A
10,00 mm

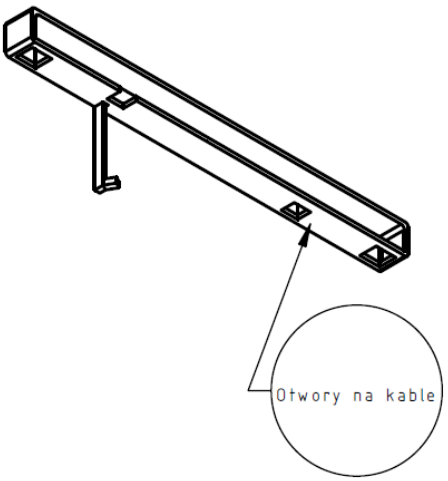


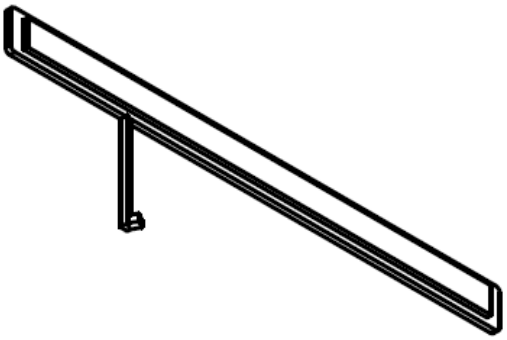
Na podstawach słupków zainstalowane są słupki.

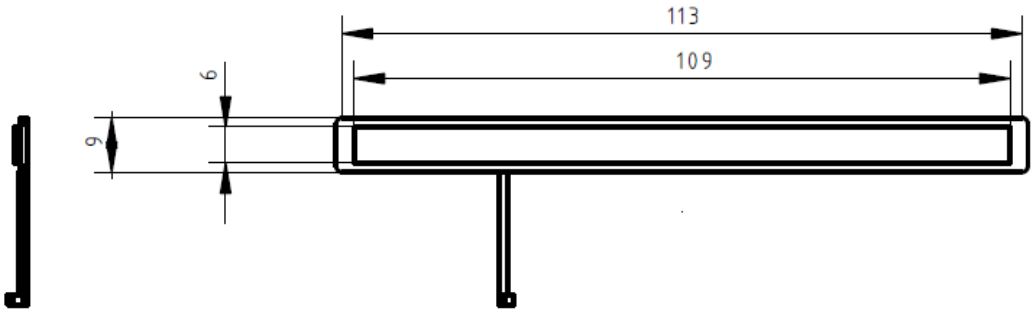


Słukpki zwieńczone są poprzeczką, składającą się z dwóch cześci.

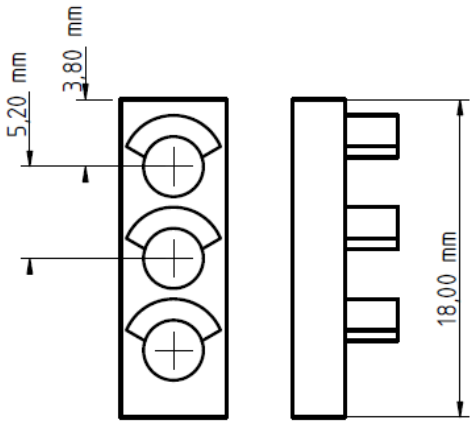


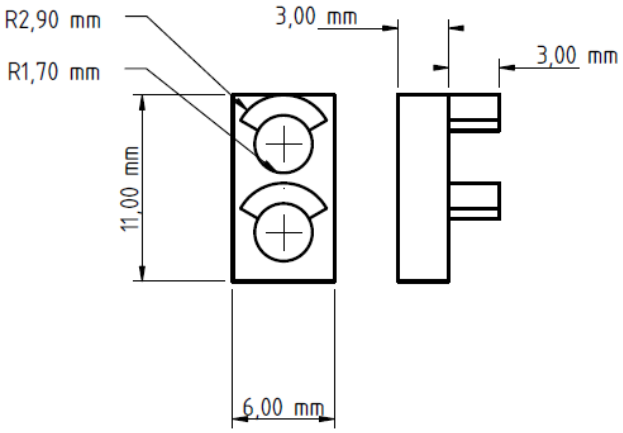


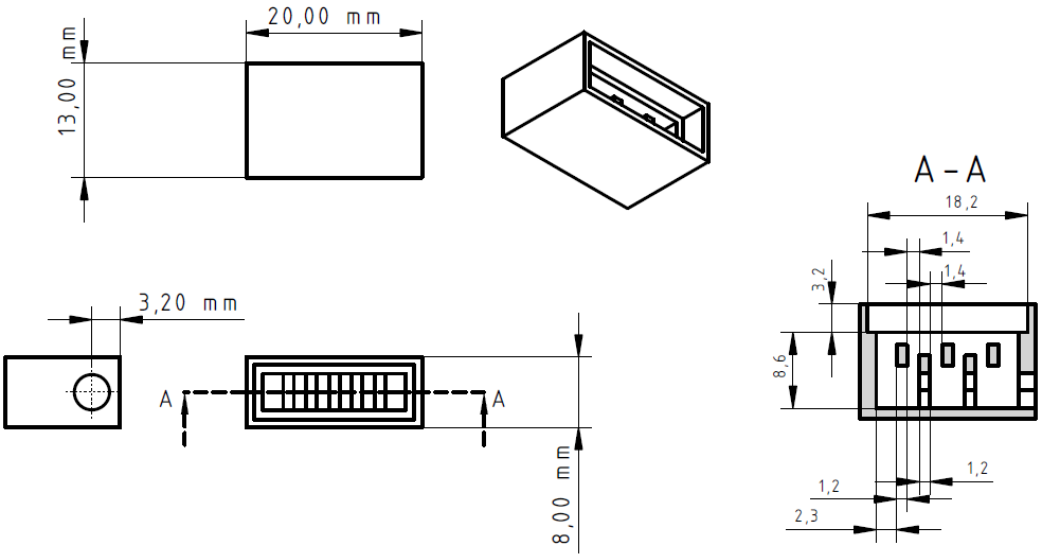


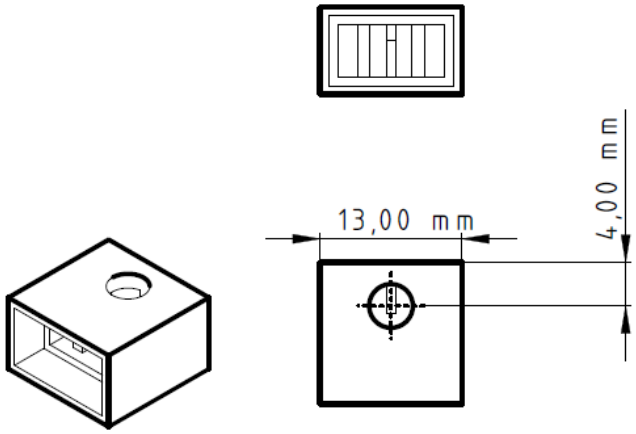


Semafory duży (dla pojazdów) i mały (dla pieszych) składają się z dwóch części:

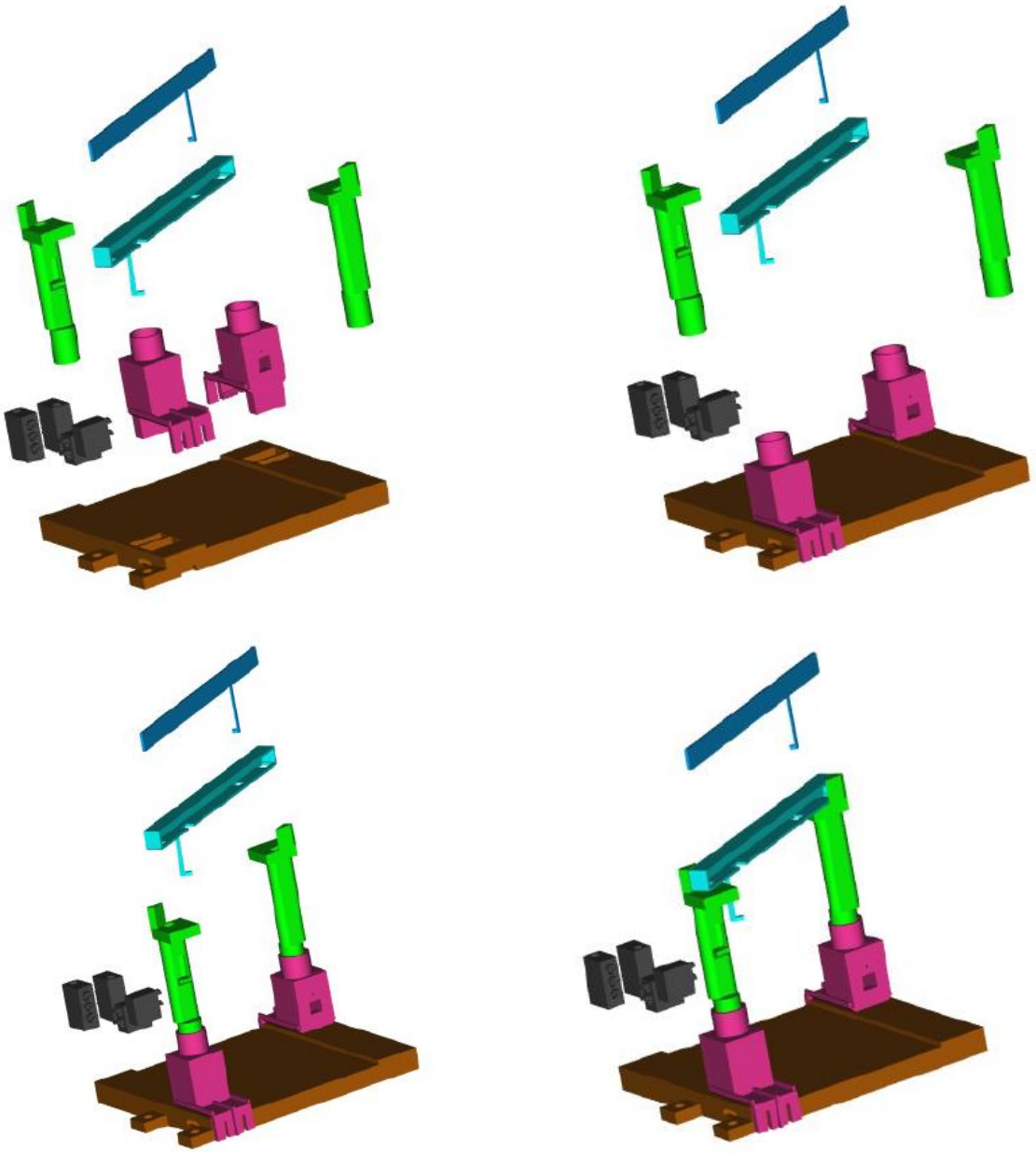


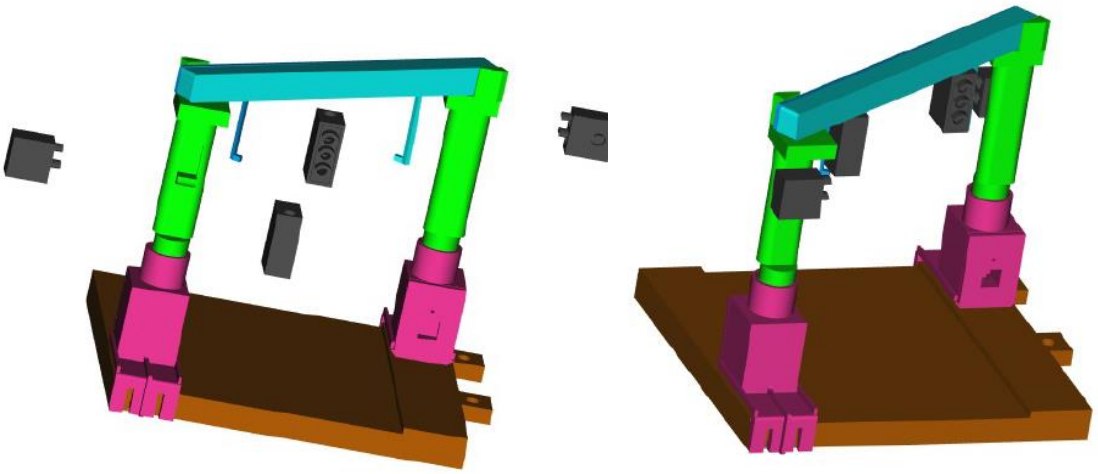




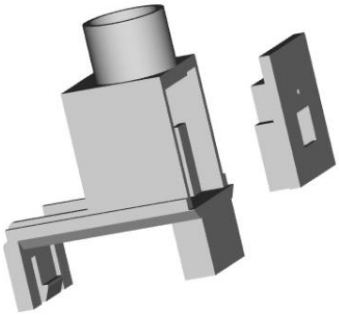


Całość należy zmontować zgodnie z rysunkami poniżej:





Sposób złożenia podstawy słupków:

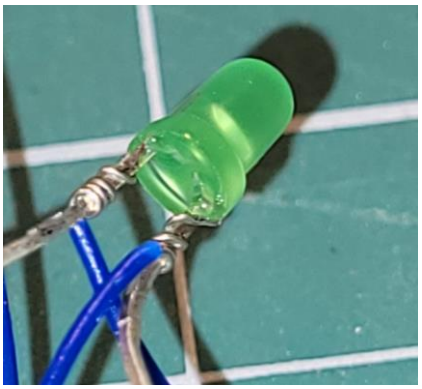


UWAGI: Po wydrukowaniu wszystkich elementów sprawdź ich spasowanie i ewentualnie spiłuj delikatnie elementy, których spasowanie okazało się zbyt ciasne. Dobrze zaplanuj kolejność wykonywania czynności montażowych.

Montaż należy rozpocząć od przygotowania przewodów. Przewody z sygnalizatorów będą przeprowadzone poprzez odpowiednie otwory w elementach składowych. W prezentowanym rozwiązaniu przewody z poszczególnych elementów zostały wyprowadzone na zewnątrz i na uniwersalnej płytce miedzianej zlutowane wraz z rezystorami zgodnie z zamieszczonym schematem. Sposób zlutowania przewodów oraz montaż płytki pozostawiam do indywidualnego rozwiązania.

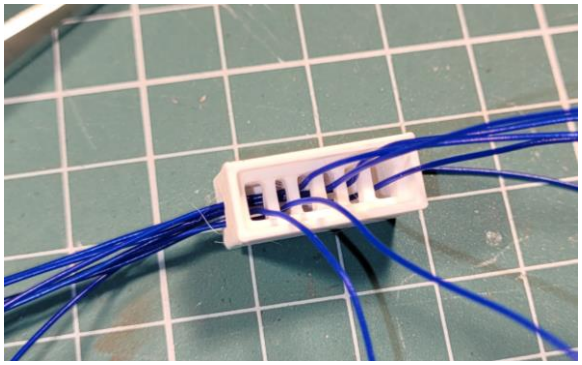
W zrealizowanym projekcie użyty został cienkie przewód montażowy (kynar o średnicy 0,2 mm) – taki jaki stosuje się do połączeń owijanych. Przekroje elementów są tak dobrane, że z powodzeniem zmieści się w nich komplet takich przewodów.

Dla każdej diody LED i transoptora należy przyciąć około 30 cm przewodu (po jednym na każde wyprowadzenie). W pierwszej kolejności należy przylutować przewody do wyprowadzeń diod LED i transoptora. Przed lutowaniem należy przyciąć końcówki w diodach LED na około 4-5mm. Połączenie lutowane należy wykonać starannie, tak aby pomiędzy wyprowadzeniami pozostała wolna przestrzeń (około 2mm).



Najlepiej jest owinąć przewód dookoła wyprowadzenia (tak jakby miało to być połączenie owijane), następnie na połączenie należy nanieść bardzo małą ilość cyny.

Przewody należy odpowiednio przeciągnąć przez tylna część sygnalizatorów tak, aby każdy przewód znajdował się w odpowiedniej „przegródce”.



Przylutowane przewody należy stopniowo przewlekać przez obudowę sygnalizatorów lub podstawę słupków.

Należy zwrócić szczególną uwagę na odpowiednie ułożenie kolejności diod, tak aby po zamocowaniu sygnalizatorów do pozostałych części konstrukcji ułożenie kolorów było odpowiednie. Szczególnie należy uważać przy montażu diod do sygnalizatorów dla pieszych. Pomimo, że sygnalizatory są po różnych stronach przejścia to kolejność diod względem otworu bocznego (przez który wyprowadzone zostaną kable) jest taka sama.

Część sygnalizatora maskującego diody, przed osadzeniem dobrze jest od wewnątrz pomalować czarną farbą – zminimalizuje to „przebicie" światła do sąsiednich diod (uzyskamy lepszy efekt wizualny). Jeśli zdecydujesz się całość sygnalizacji pomalować – części maskujące sygnalizatorów pomaluj przed ich osadzeniem. Podobnie postępuj z nakładką stanowiąca cześć podstawy słupka.

Ponieważ łatwo jest się pomylić, jakie przewody odpowiadają konkretnym wyprowadzeniom, po zamontowaniu elementów do sygnalizatorów i podstaw słupków sprawdź miernikiem przewody, wykonaj test działania transoptora. Podczas sprawdzania przewodów oznacz ich końcówki (np., przylepiając kawałek papierowej taśmy samoprzylepnej, ja używam malarskiej taśmy maskującej), opisz każdy z przewodów. Aby poszczególne elementy osadzić trwale użyj żelowego kleju cyjan akrylowego (najlepiej w żelu, o czasie wiązania 20-30 sekund).

Następnie przeciągnij przewody górnych sygnalizatorów przez otwory w poprzeczce wieńczącej słupki sygnalizatorów, sklej w całość sygnalizatory, poprzeczkę i pokrywę poprzeczki.

Przeciągnij przewody sygnalizatorów dla pieszych przez otwory w słupkach, przyklej sygnalizatory do słupków.

Przeciągnij przewody górnych sygnalizatorów (wychodzące ze sklejonej w całość poprzeczki) przez słupki, jeszcze nie przyklejaj poprzeczki do słupków. Dla ułatwienia dalszego montażu możesz przymocować tymczasowo poprzeczkę do słupków za pomocą taśmy klejącej. Przeciągając przewody przez słupki pozostaw w słupkach mały zapas przewodów – nie naciągaj przewodów.

Przewody z kontrolki i transoptora przeciągnij przez podstawkę, zamocuj do podstawki podstawy słupków, jeszcze ich ostatecznie nie sklejaj z podstawką (mają zatrzask, który umożliwi tymczasowe mocowanie).

Przewlecz przez podstawy słupków przewody wychodzące ze słupków. Należy tu zachować ostrożność i przewlekać każdy przewód z osobna (nie próbuj przewlekać całej wiązki przewodów).

Jeśli wszystkie przewody zostały przewleczone (ich końce znajdują się w spodniej części podstawki) – tymczasowe mocowania elementów zastąp mocowaniem na klej.

Na koniec przeciągnij wszystkie przewody na zewnątrz podstawki (od spodu w podstawce wykonane są odpowiednie otwory).

Zlutuj całość zgodnie ze schematem i ewentualnie pomaluj model sygnalizacji.