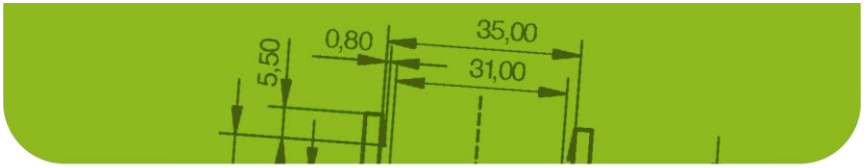
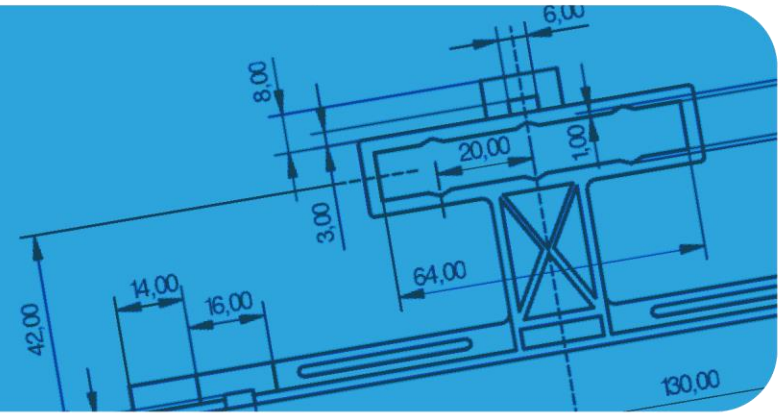
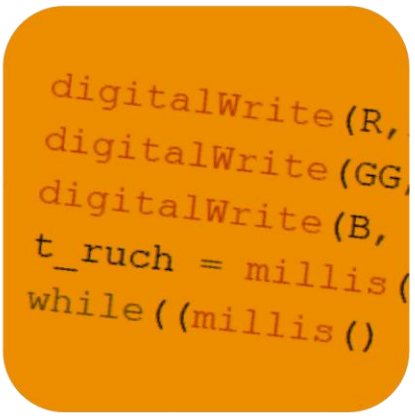






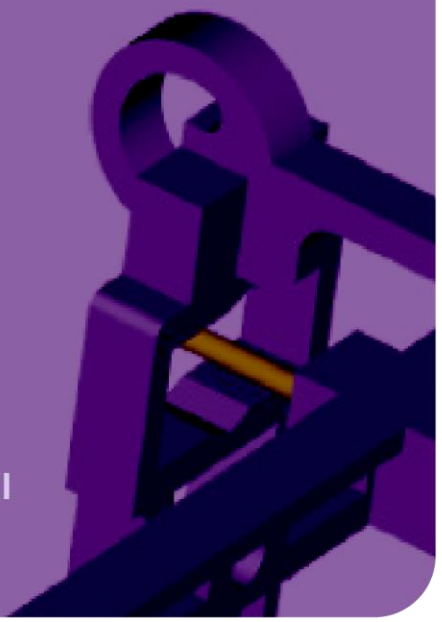
Wojciech Kolarz

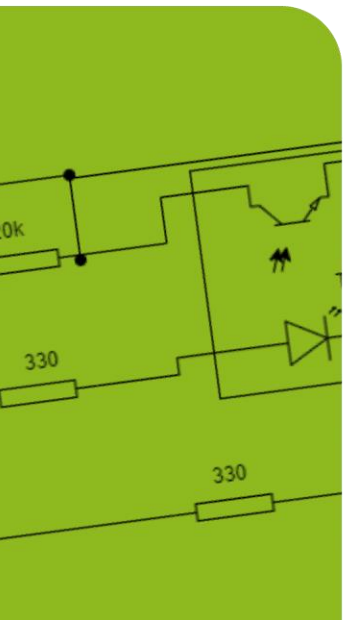
PROJEKT

NIE BÓJ SIĘ ROBOTA

DIY

MODELOWANIE 3D
PROGRAMOWANIE
PODSTAWY ELEKTRONIKI





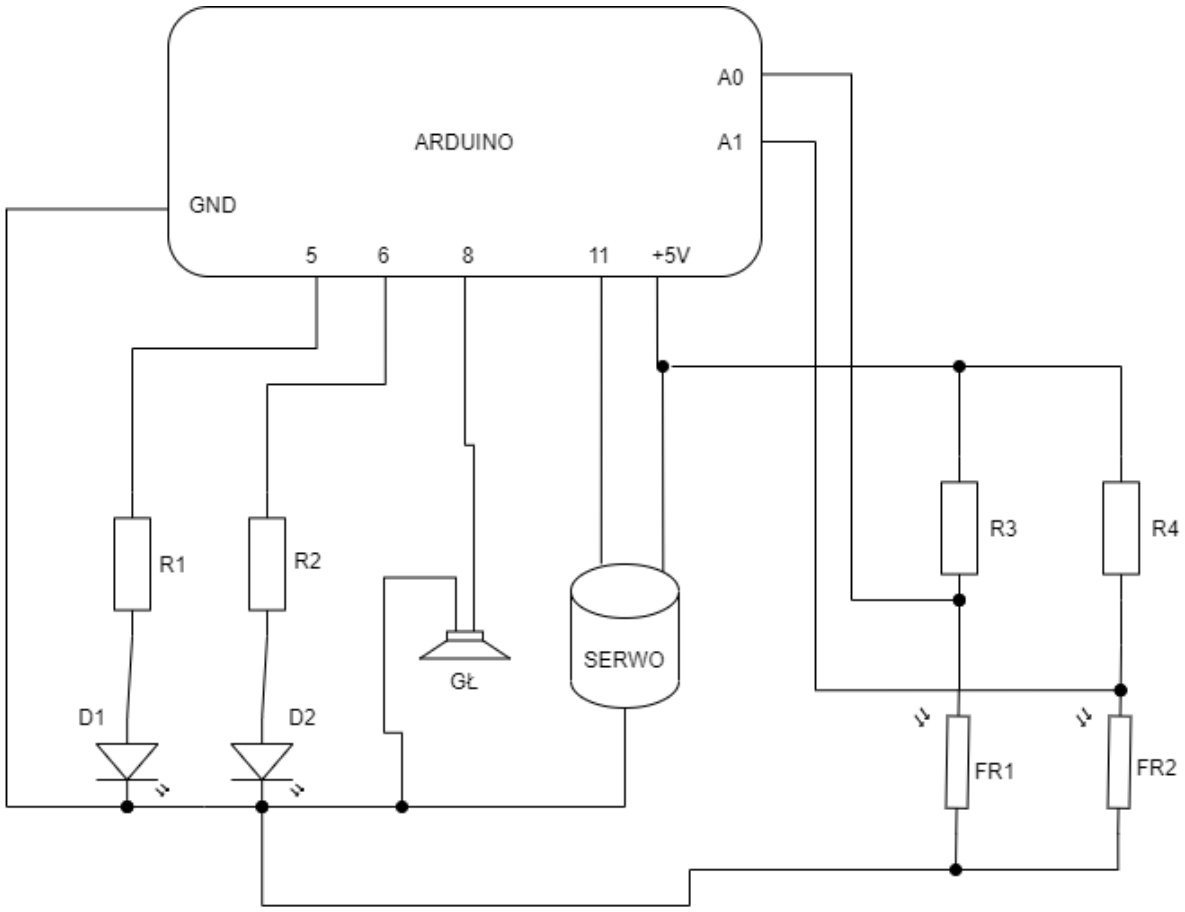
CENTRUM KISS

Materiał dodatkowy do filmu MIE BÓJ SIĘ ROBOTA.

Podzespoły i materiały użyte do budowy robota:

- Elementy drukowane: filament PLA, wypełnienie dowolne (w prezentowanym modelu 20%), ustawienia typowe dla dyszy 0,4 mm, grubość warstwy 0,2 mm.
- Arduino UNO R3 – 1 szt.
- serwomechanizm modelarski SG-90 - micro – 180 – 1 szt.
- typowe diody LED 5mm (w prezentowanym przykładzie niebieskie) – 2 szt.
- fotorezystor – 2 szt.
- rezystor 22 - 100 kΩ - 2 szt.
- rezystor 220 – 470 Ω - 2 szt.
- buzzer (bez generatora, 23 mm) – 1 szt.
- opcjonalnie koszyczek na baterie (4 paluszki AA lub AAA może być z wyłącznikiem) – 1 szt.

Schemat robota.



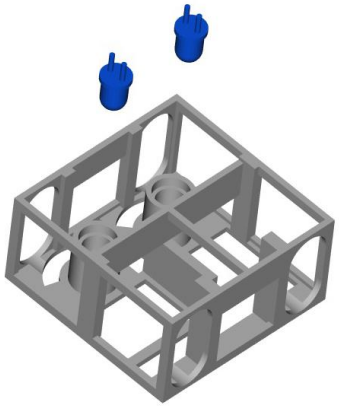
Przed ostatecznym montażem układ można skonfigurować na płytce testowej. Rezystory R1 i R2 należy dobrać w zależności od rodzaju zastosowanych diod LED. Również rezystory R3 i R4 należy dobrać w zależności od zastosowanych fotorezystorów.

W prezentowanym rozwiązaniu, przy zastosowaniu typowych niebieskich diod LED rezystory R1 i R2 mają wartość 220 Ω. Jeśli zastosowane fotorezystory będą miały parametry: rezystancja jasna 10-30 kΩ, rezystancja ciemna 2 MΩ, R3, R4 mogę mieć wartości 47 – 68 kΩ.

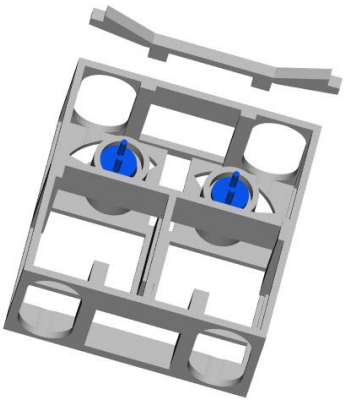
Konstrukcja robota.

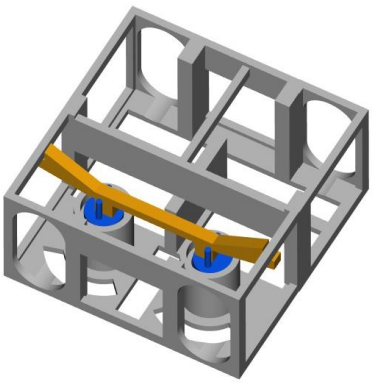
Całość została zaprojektowana tak, aby poszczególne części robota można było złożyć bez użycia kleju i połączeń śrubowych (jedyny wyjątek stanowi montowanie serwomechanizmu).

Montaż robota należy rozpocząć od złożenia jego głowy. Zegnij końcówki diod LED o 90°. Katody diod (końcówki podłączane do bieguna ujemnego) zlutuj wspólnie z jednym przewodem, który potem dołączysz do pinu GND Arduino. Następnie diody wsuń w odpowiednie otwory w przedniej części głowy (na rysunku nie pokazano przewodów).

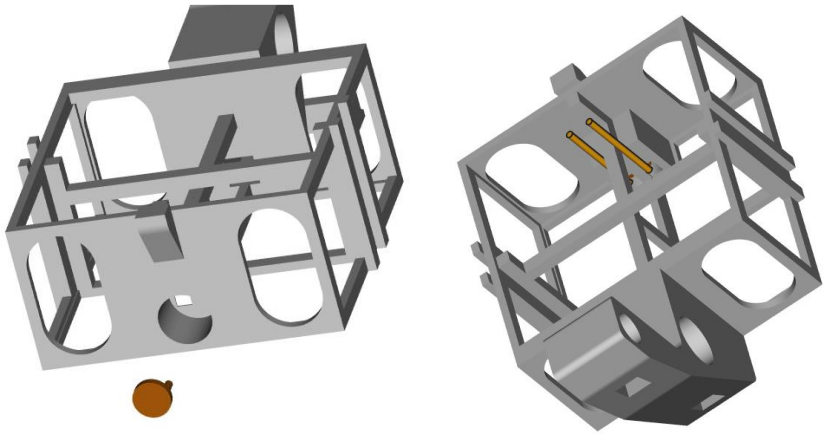


Aby diody nie wypadały z otworów zablokuj je wstawiając górne mocowanie diod. Wstawiany element należy lekko wygiąć, tak aby jego końce „wskoczyły” w boki przedniej części głowy.

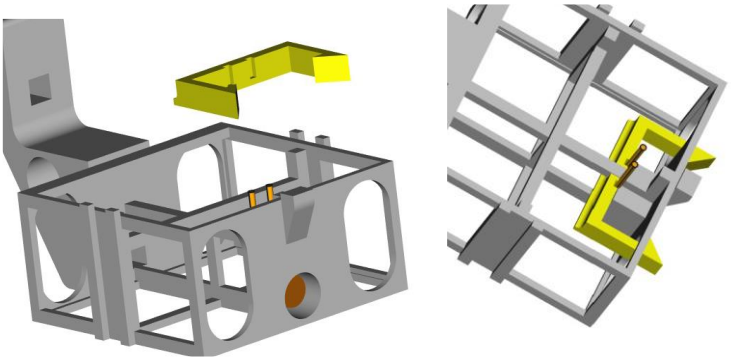




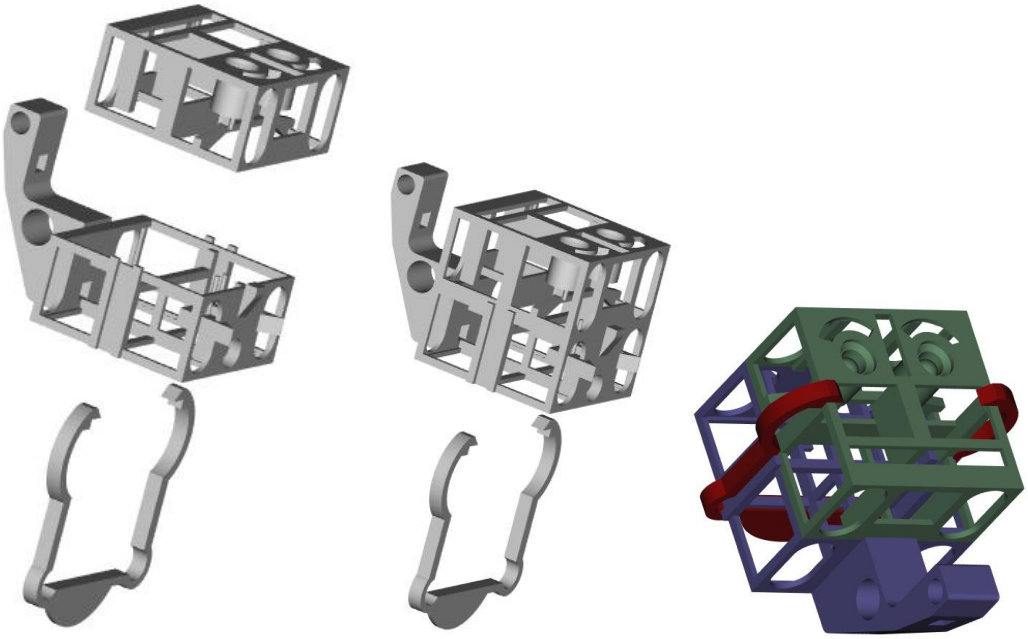
Przylutuj przewody do fotoopornika, który będzie pełnił rolę górnego czujnika (czujnika umieszczonego w głowie). Następnie przeciągnij przewody przez otwór w drugiej połowie głowy. Wsuń fotoopornik do gniazda w drugiej połowie głowy. Wygnij odpowiednio końcówki fotoopornika tak, aby fotoopornik nie mógł wysunąć się z otworu.



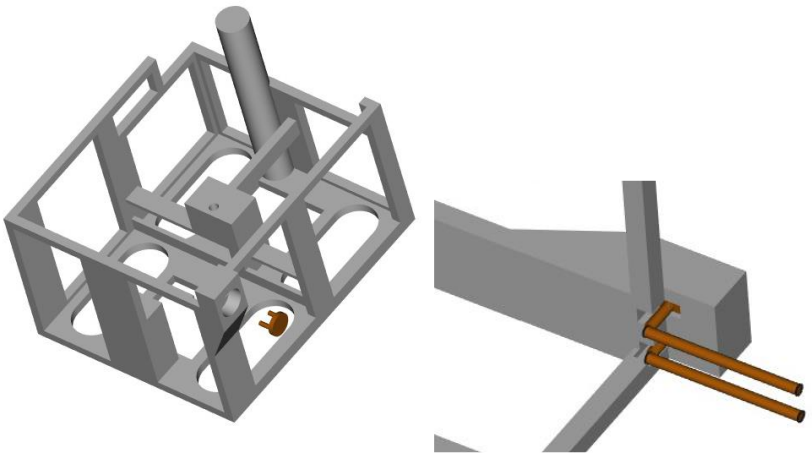
Zabezpiecz fotoopornik przez wysunięciem zakładając mocowanie czujnika górnego. Mocowanie delikatnie wygnij i wsuń w tylną cześć głowy.



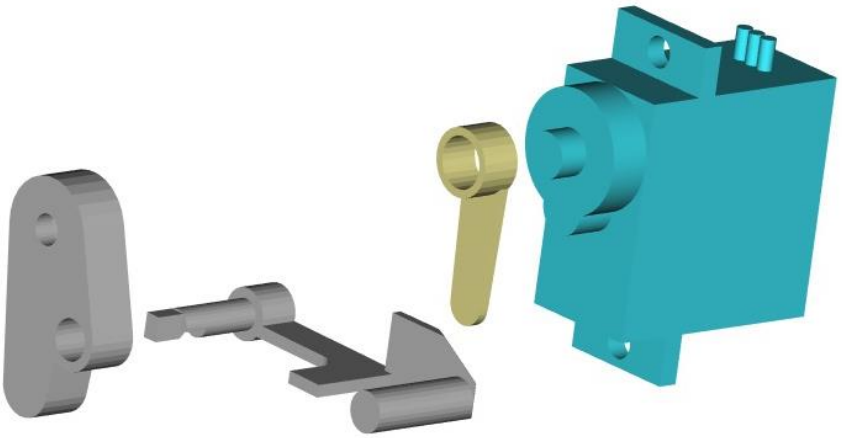
Przeciągnij przewody przez otwory i złóż obydwie części głowy spinając je zaczepem:



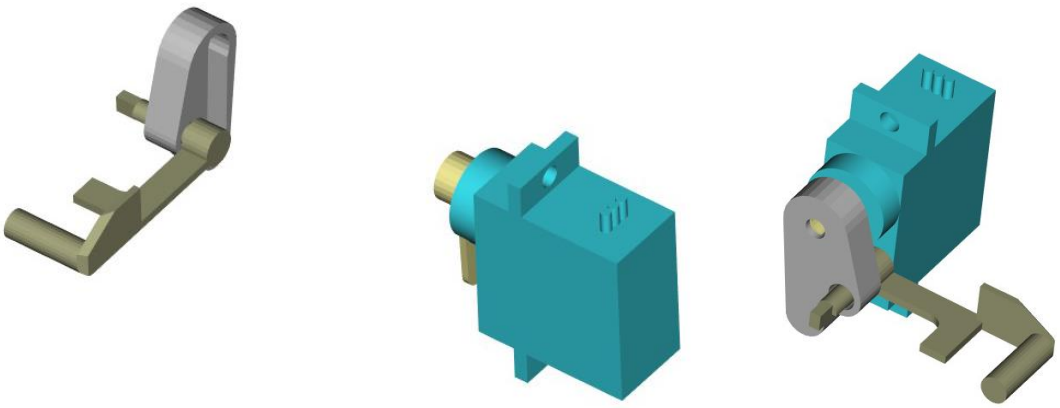
Do fotoopornika przylutuj przewody, przewlecz je przez otwór w lewej części tułowia. Do otworu wsuń fotoopornik, a jego końcówki wygnij tak, aby uniemożliwić wypadnięcie.



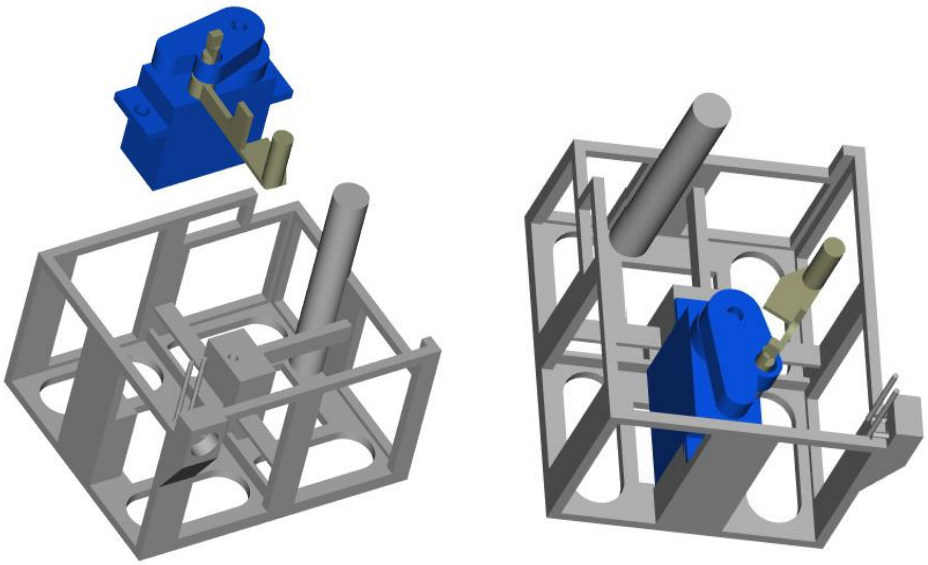
Montowanie orczyka i wodzika do serwomechanizmu rozpocznij od Przygotowania orczyka (jednoramienny) oraz wkrętów do montowania orczyka (orczyk i wkręt powinny znajdować się w komplecie z serwomechanizmem). Podczas montażu oś serwomechanizmu ustaw w skrajnym lewym położeniu. Orczyk należy umieścić na osi serwomechanizmu w taki sposób, aby jego oś symetrii przebiegała równo z osią symetrii serwa, orczyk powinien być ustawiony jak na rysunku:



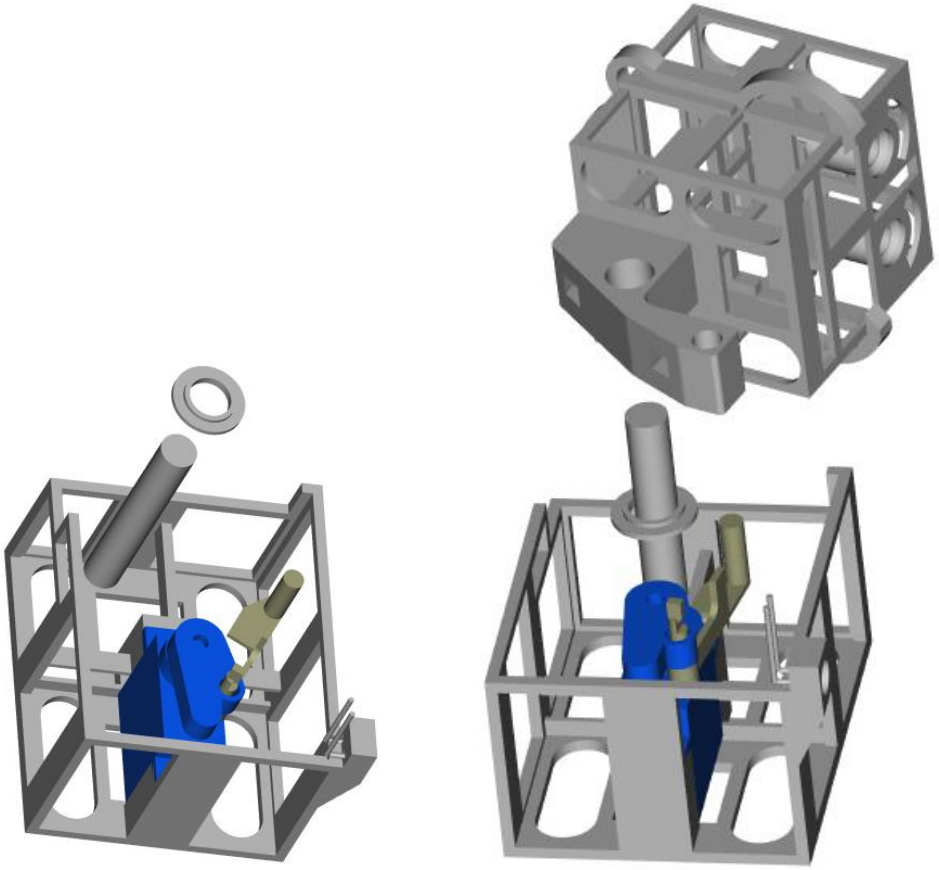
Na orczyk należy nałożyć mocowanie wodzika wraz z wsuniętym w otwór wodzikiem. Mocowanie wodzika wraz z orczykiem należy przykręcić do osi serwa.

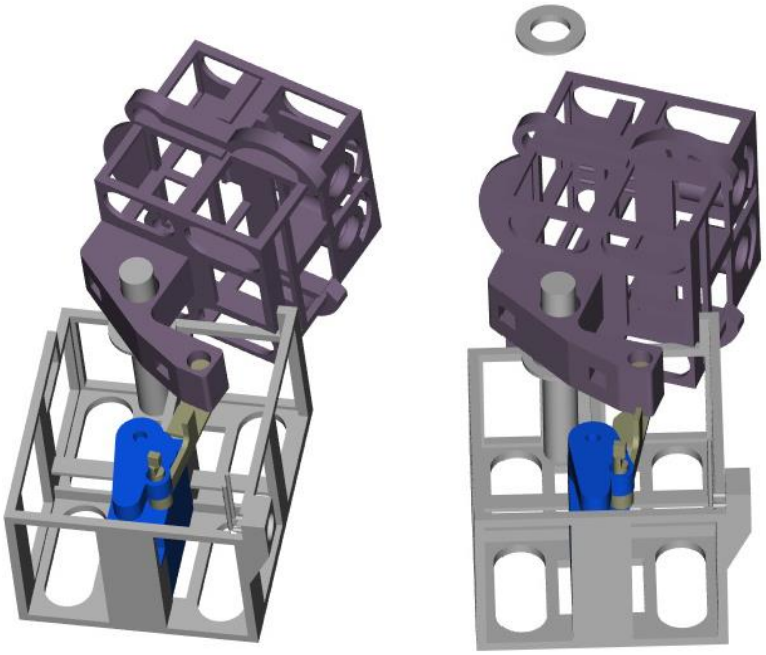


Serwo należy przykręcić do korpusu.

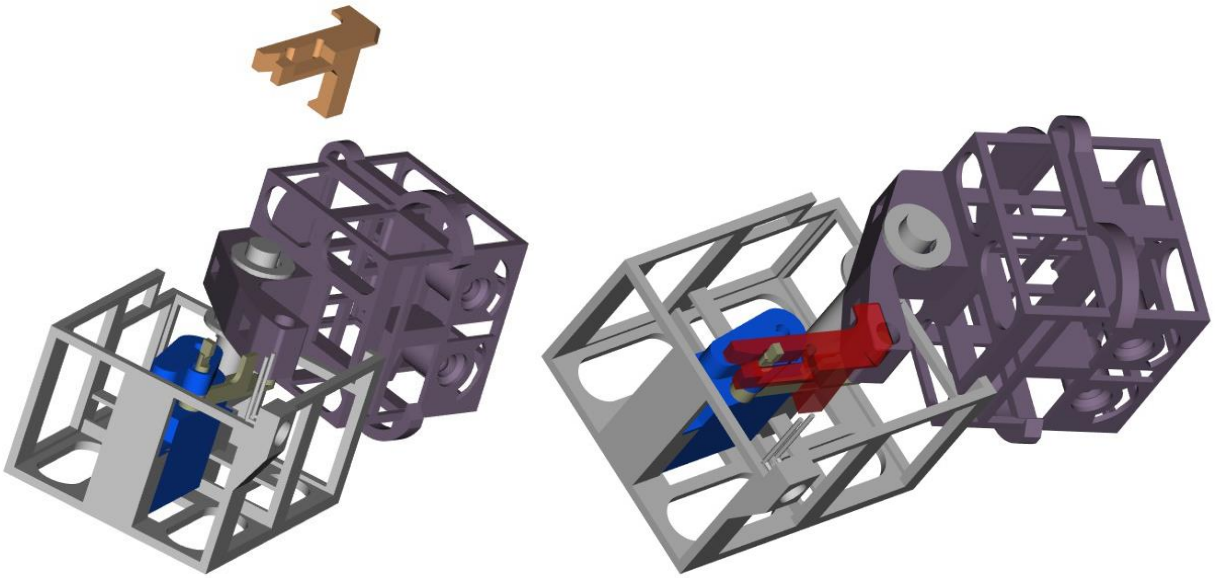


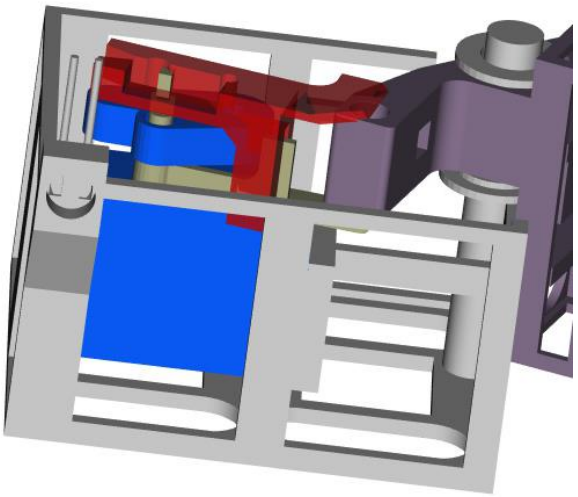
Głowę robota osadź na osi znajdującej się w lewej części tułowia. Przed założeniem głowy umieść podkładkę. Głowę połącz z wodzikiem poprzez umieszczenie odpowiedniego otworu w bolcu wodzika.



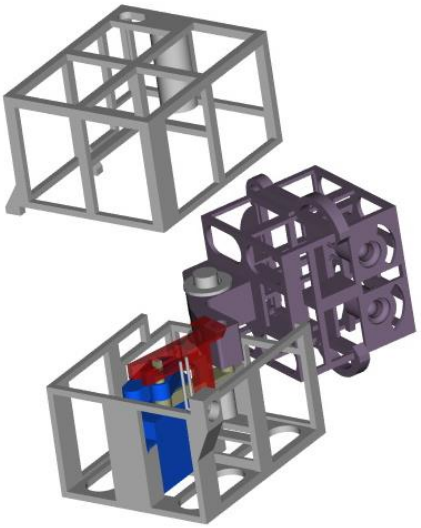


Załóż drugą część wodzika – element ten należy (lekko wyginając) wsunąć tak, aby z lewej strony zablokował się na pierwszej części wodzika, a z prawej strony wsunął się w wyżłobienie osi współpracującej z elementem przymocowanym do orczyka serwa.

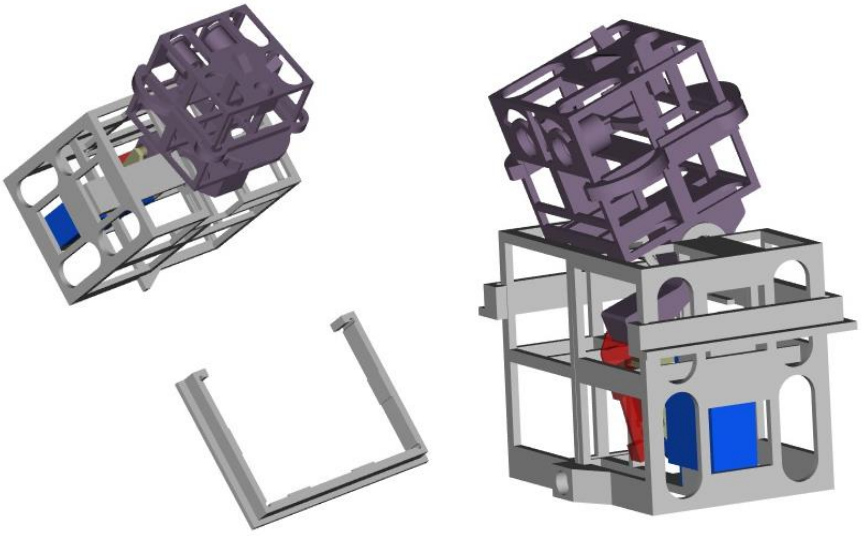




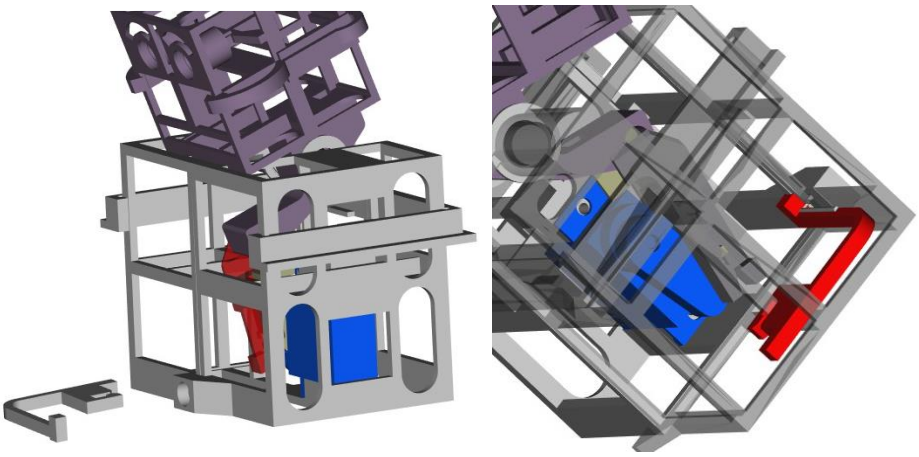
Przeciągnij przewody, tak aby wychodziły w dolnej części tułowia. Załóż prawą cześć tułowia.



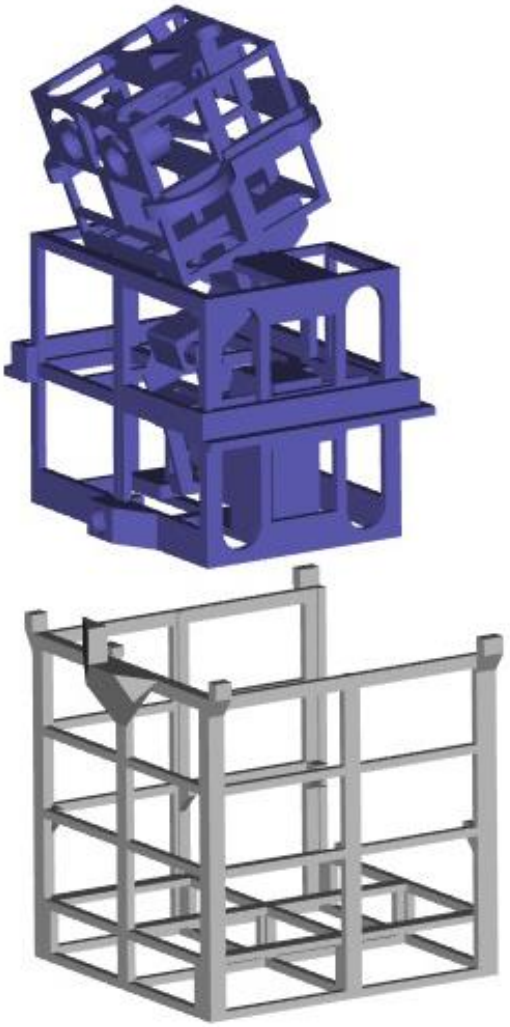
Tułów zepnij klamrą



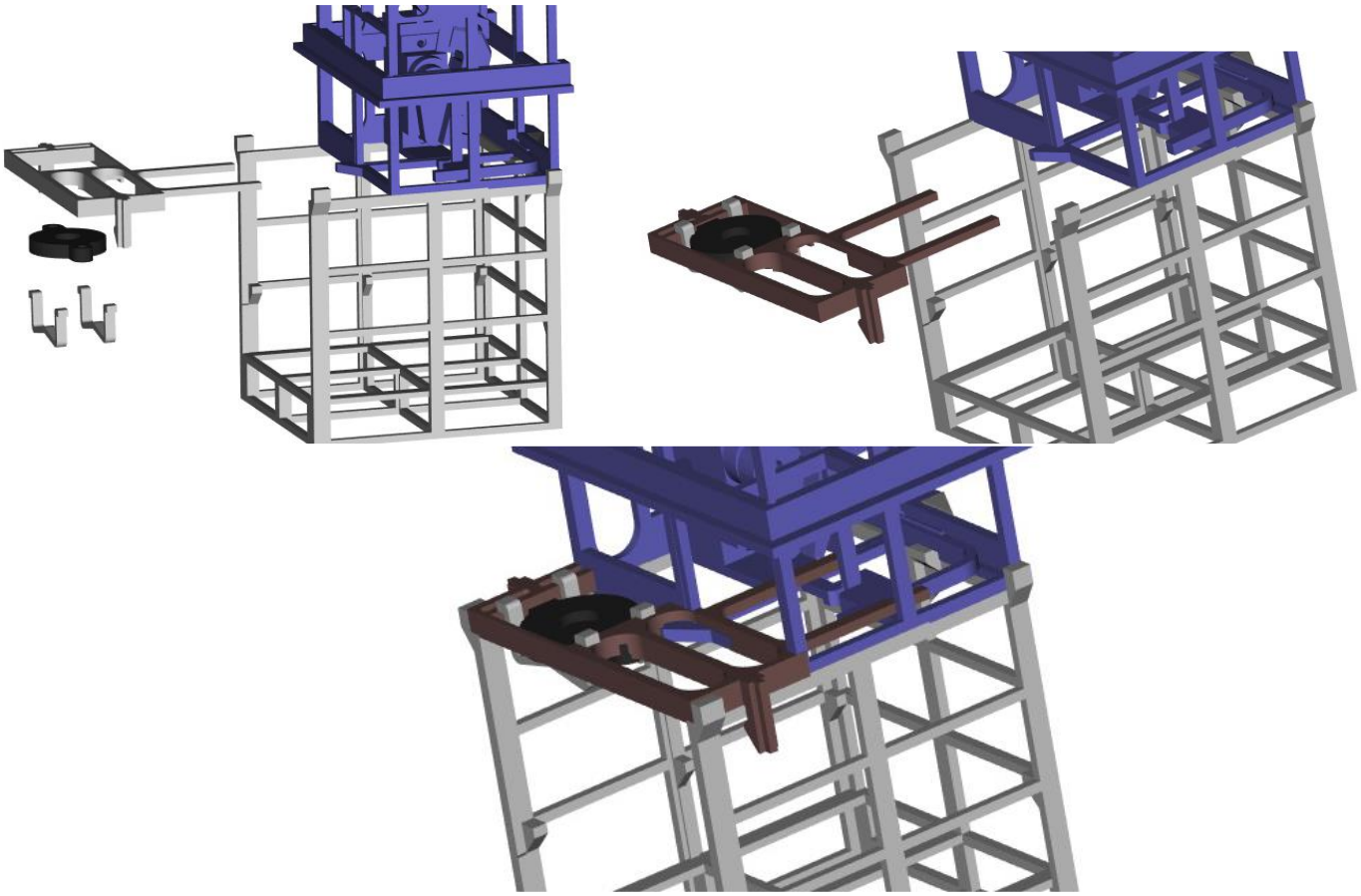
Od tyłu tułowia wsuń element zabezpieczający. Jego ułożenie powinno zabezpieczyć wypadanie fotorezystora.



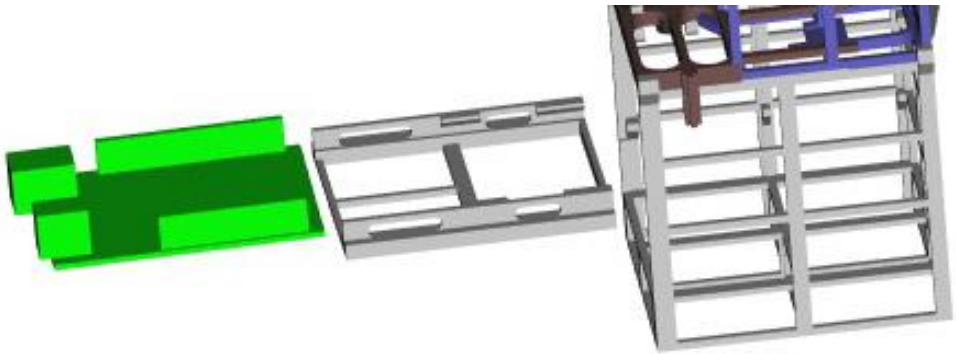
Złożoną głowę i górną część tułowia nasadź na dolną część tułowia.

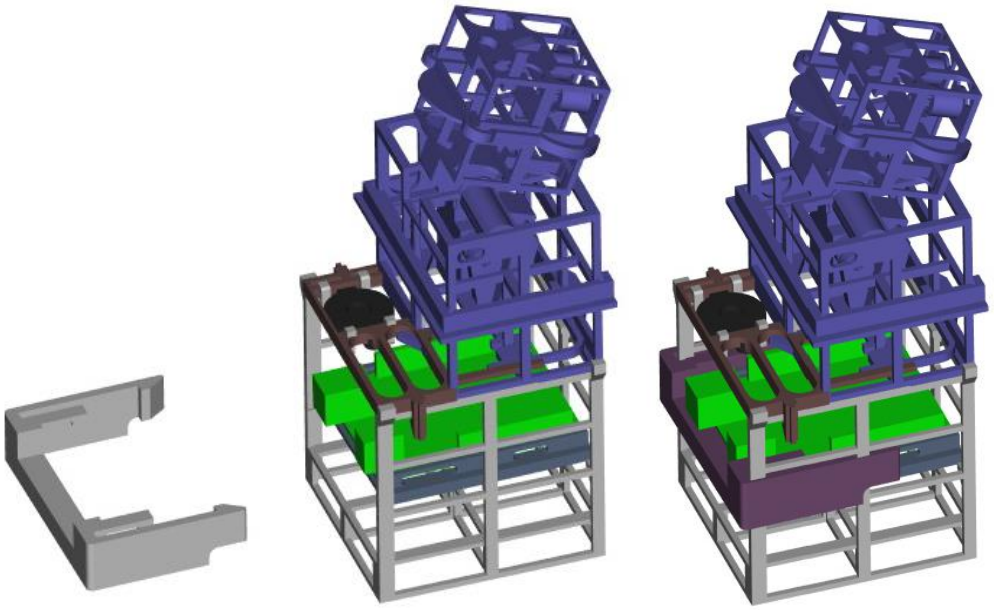


Zamontuj buzzer. Część z buzzerem wsuń tak, aby jej wystające listwy znalazły się nad dolnymi krawędziami konstrukcji górnej części tułowia, a zatrzaski wpięły się w górne krawędzie dolnej części tułowia.



Wsuń Arduino w uchwyt. Jeśli rezystory montowałeś na płytce drukowanej, połącz przewody z płytką, a płytkę połącz z Arduino. Jeśli rezystory montowałeś bezpośrednio na przewodach – połącz przewody z Arduino. Uchwyt z zamontowanym Arduino wsuń do dolnej części tułowia. Załóż klamrę.





Jeżeli będziesz korzystał z zasilania bateryjnego, na samym dole umieść koszyczek na baterie.

Program.

Interakcje robota.

1. Pogłaskany raz po głowie – reakcja 1.
2. Pogłaskany drugi razy po głowie – reakcja 2.
3. Pogłaskany trzeci i każdy następny – reakcja 3.
4. Kolejne głaskanie po głowie występuje wtedy, gdy górny czujnik zostaje przesłonięty w dwu- trze- wielokrotnie w krótkich odstępach czasu.
5. Pogłaskany po brzuszku – reakcja 4.
6. Jeśli upłynie ustalony czas, a czujniki nie wykryły zmiany natężenia światła – reakcja 5.
7. Jeśli czujnik górny i dolny wykryły stałe zmniejszenie natężenia światła - robot zasypia.
8. Jeśli czujniki górny i dolny wykryły stały wzrost natężenia światła - robot budzi się.
9. Gdy robot śpi to nie reaguje na bodźce zewnętrzne (za wyjątkiem stałej zmiany natężenia światła zewnętrznego.

Zanim rozpoczniemy tworzenie szczegółowego algorytmu działania robota należy sprawdzić jakie wartości odczytywane z czujników w zależności od natężenia światła. W tym celu należy stworzyć prosty program, który za pomocą portu szeregowego będzie wysyłał do komputera wartości adekwatne do napięć pojawiających się na dzielniku napięć (rezystor/fotorezystor). Wartości te odczytamy za pomocą monitora portu szeregowego.

Przykładowy program:

```
#define D1 5
#define D2 6
void setup()
{
  Serial.begin(9600);
  pinMode(D1,OUTPUT);
  pinMode(D2,OUTPUT);
  digitalWrite(D1,HIGH);
  digitalWrite(D2,HIGH);
}
 void loop()
{
  delay(500);
  Serial.print("Czujnik górny: ");
  Serial.print(analogRead(A0));
  Serial.print(" - Czujnik dolny: ");
  Serial.println(analogRead(A1));
}
```

Poniżej wyniki mojego testu. Testy przeprowadzone zostały przy włączonych diodach LED (świecące oczy), aby sprawdzić, czy świecenie diod nie zakłóci działania czujników.

W moim przypadku dane oscylowały wokół wartości:

Natężenie światła	**Czujnik górny**	**Czujnik dolny**
Bardzo dobrze oświetlone pomieszczenie	100-200	100-200
Słabo oświetlone pomieszczenie	500 - 800	400 – 600
Ciemne pomieszczenie	800-850	800 – 850
Bardzo ciemne pomieszczenie	>1000	>1000

Praktycznie w każdym pomieszczeniu (za wyjątkiem bardzo ciemnego, przysłonięcie któregokolwiek z czujników powodowało wzrost odczytanej wartości przynajmniej o 70-100. Zaświecenie lub wygaszenie diod LED nie ma praktycznie wpływu na działanie czujników.

Przystępując do tworzenia algorytmu a następnie programu sterującego robotem, nasz tok postępowania wykonamy zgodnie z zasadami myślenia komputacyjnego. W związku z powyższym konieczny będzie podział zadania na mniejsze części.

Proponuję rozpocząć od stworzenia procedur realizujących pobudkę oraz zasypianie robota.

Zanim rozpoczniemy tworzenie poszczególnych elementów algorytmu/programu przyjmiemy założenia:

Robot jest obudzony – w trybie wykrywania bodźców zewnętrznych – głowa ustawiona jest prosto, diody LED (oczy) delikatnie świecą.

Robot śpi – głowa opuszczona do przodu, diody LED (oczy) wygaszone.

Sekwencja poleceń realizująca pobudkę robota będzie zawarta w procedurze:

```
27 void pobudka()
28 {
29     for(glowa_pozycja = 0; glowa_pozycja <=glowa_norm; glowa_pozycja=glowa_pozycja+5)
30   {
31     pwm_d1 = map(glowa_pozycja, 0, glowa_norm, 0, pwm_norm);
32     pwm_d2 = pwm_d1;
33     ton = map(glowa_pozycja, 0, glowa_norm, 100, 1200); // częstotliwość dźwieku będzie stopniowo rosła
34     serwomechanizm.write(glowa_pozycja); //głowa będzie się unosiła stopniowo
35     analogWrite(D1, pwm_d1); // diody będą się stopniowo rozjaśniać
36     analogWrite(D2, pwm_d2);
37     tone(GL, ton);
38     delay(50);
39   }
40   noTone(GL);
41 }
```

W pętli ***for*** będzie zwiększana wartość kąta ustawienia orczyka serwomechanizmu, a tym samym położenia głowy robota.

Ponieważ wraz ze zmianą kąta ustawienia serwomechanizmu (położenia głowy) będzie narastała częstotliwość dźwięku oraz jasność świecenia diod LED, wraz ze zmiana wartości zmiennej ***glowa_pozycja*** zmieniać będą się zmienne ***ton, pwm_d1.*** i ***pwm_d2***. W przedstawionym rozwiązaniu przyjęto, że zmianie pozycji głowy (***glowa_pozycja*** przyjmować będzie wartości od 0 do ***glowa_norm***), towarzyszyć będzie zmiana częstotliwości generowanego dźwięku od 100Hz do 1200 Hz. W celu obliczenia częstotliwości adekwatnej do danego kąta nachylenia głowy użyta została funkcja ***map()***.

Funkcja działa w taki sposób, że (linia kodu nr 33) zmiennej ***ton*** przypisuje wartości od 100 do 1200 proporcjonalnie do zmiany wartości zmiennej ***glowa_pozycja*** (od 0 do wartości zapisanej w zmiennej ***glowa_norm***). W analogiczny sposób zmieniane są wartości ***pwm_d1*** i ***pwm_d2***.

Zasypianie robota będzie realizowała procedura:

```
43 void zasypia()
44 {
45   for(glowa_pozycja = glowa_norm; glowa_pozycja >=0; glowa_pozycja=glowa_pozycja-10)
46   {
47     pwm_d1 = map(glowa_pozycja, 0, glowa_norm, 0, pwm_norm);
48     pwm_d2 = pwm_d1;
49     serwomechanizm.write(glowa_pozycja); //głowa będzie stopniowo opadała
50     analogWrite(D1, pwm_d1); // diody będą się stopniowo wygaszać
51     analogWrite(D2, pwm_d2);
52       //sekwencja generowanych dźwieków
53       tone(GL, 100);
54       delay(200);
55       noTone(GL);
56       delay(200);
57       tone(GL, 80);
58       delay(200);
59       noTone(GL);
60       delay(200);
61       tone(GL, 60);
62       delay(200);
63       noTone(GL);
64       delay(200);
65       tone(GL, 40);
66       delay(200);
67       noTone(GL);
68     }
69 }
```

Pierwszą wersję programu (robot budzi się lub zasypia) znajdziesz w pliku ***Robocik_program_2***.

Teraz wzbogacimy nasz program tak, aby robot budził się i zasypiał odpowiednio gdy światło otoczenia będzie miało duże lub małe natężenie. Jednak robot nie może reagować na chwilowe zmiany natężenia światła, jego pobudka lub zasypianie powinny następować wtedy, gdy zmiana natężenia światła będzie trwała. W tym celu wprowadzimy zmienną ***czas_dzien_noc***. Jeżeli nastąpi zmiana natężenia światła zarejestrowana przez obydwa czujniki i zmiana ta będzie nieprzerwanie trwała dłużej niż czas określony przez wartość zmiennej ***czas_dzien_noc*** to robot obudzi się lub zaśnie.

```
31  void loop()
32 {
33     pobudka();
34     czas_1 = 0;
35     while(dzien)
36     {
37        swiatlo_gora = analogRead(CZ_G);
38        swiatlo_dol = analogRead(CZ_D);
39        if ((swiatlo_gora > dzien_noc) and (swiatlo_dol > dzien_noc))
40        {
41          if (czas_1 == 0) czas_1 = millis();
42          if ((millis() - czas_1) > czas_dzien_noc) dzien = false;
43        }
44        else czas_1 = 0;
45     }
46     zasypia();
47     czas_1 = 0;
```

Załóżmy, że po włączeniu zasilania robot obudzi się – pierwsza instrukcja pętli głównej to wywołanie odpowiedniej procedury. Zmienne ***czas_1*** będzie potrzebna po to, aby stwierdzić, czy zmiana natężenia światła jest trwała. Sprawdzanie natężenia światła zewnętrznego będzie odbywało się w pętli ***while()***, której wnętrze będzie wykonywane jeśli zmienna logiczna ***dzien*** będzie miała wartość ***true*** (wartość tej zmiennej można wcześniej, np. w procedurze ***setup()*** ustawić na ***true***). Wartości odczytane z wejść analogowych (czujniki) zapisane zostaną w dwóch zmiennych (odpowiednio dla czujnika górnego i dolnego).

Wartość odpowiadająca progowi pomiędzy dniem i nocą zapisana będzie w zmiennej ***dzień_noc*** (wartość tę można ustawić wcześniej np. podczas deklarowania tej zmiennej). Jeżeli obydwa czujniki zarejestrują niskie natężenie światła (wartości odczytane z wejść analogowych przekroczą próg) wtedy:

- jeżeli jest to pierwsze wykrycie takiej sytuacji (zmienna ***czas_1 = 0***), do zmiennej ***czas_1*** zostanie zapisany aktualny czas;

- jeżeli jest to kolejne wykrycie takiej sytuacji, to nastąpi sprawdzenie czy czas jaki minął od pierwszego wykrycia niskiego natężenia światła jest większy niż czas założony do zinterpretowania sytuacji jako trwałej zmiany natężenia światła – jeżeli zmiana natężenia światła uznana zostanie jako trwała nastąpi ustawienie zmiennej ***dzien*** na wartość ***false*** – co zakończy działanie pętli ***while()***.

Gdyby jednak występowały chwilowe przysłonięcia czujników (chwilowe zmniejszenia natężenia światła), to aby robot nie zidentyfikował tego jako noc, za każdym razem gdy natężenie światła będzie wyższe niż założona wartość progowa nastąpi wyzerowanie zmiennej ***czas_1***.

Po wyjściu z pętli uruchomiona zostanie procedura ***zasypianie()*** i wyzerowana zostanie zmienna ***czas_1***.

Dalsze część programu to analogiczna pętla ***while()***, która wykonywana będzie tym razem dopóki zmienna dzień nie przyjmie wartości ***true***.

```
46      zasypia();
47      czas_1 = 0;
48      while(not(dzien))
49      {
50         swiatlo_gora = analogRead(CZ_G);
51         swiatlo_dol = analogRead(CZ_D);
52         if ((swiatlo_gora < dzien_noc) and (swiatlo_dol < dzien_noc))
53         {
54           if (czas_1 == 0) czas_1 = millis();
55           if ((millis() - czas_1) > czas_dzien_noc) dzien = true;
56         }
57         else czas_1 = 0;
58      }
59  }
```

Cały program – w pliku ***Robocik_program_3***.

Kolejny problem do rozwiązania to reakcje robota na przysłonięcie czujników. Aby robot rozpoznał, że czujnik został przysłonięty konieczne jest zidentyfikowanie natężenia światła, które na stałe towarzyszy robotowi.

Załóżmy, że robot co jakiś czas sprawdzać będzie natężenie światła zewnętrznego – nazwałem to światło tła – natężenie to zapamiętane będzie w zmiennych ***swiatlo_gora_tlo*** i ***swiatlo_dol_tlo***. W stworzonej wcześniej pętli ***while()*** odczytywane będzie aktualne natężenie światła, a następnie po porównaniu z natężeniem tła, robot wykona odpowiednią reakcję (jeśli oczywiście natężenie światła stwierdzone w danym momencie będzie mniejsze od natężenia tła o założoną wartość – założona wartość będzie przechowywana w zmiennej ***dswiatlo***).

W tym celu należy stworzyć procedurę ustalającą światło otoczenia (światło tła):

```
125 void swiatlo_otoczenia()
126 {
127   swiatlo_gora_tlo = 0;
128   swiatlo_dol_tlo = 0;
129   for(int i = 1; i<=10; i++)
130   {
131     swiatlo_gora_tlo = swiatlo_gora_tlo + analogRead(CZ_G);
132     swiatlo_dol_tlo = swiatlo_dol_tlo + analogRead(CZ_D);
133     delay(20);
134   }
135   swiatlo_gora_tlo = swiatlo_gora_tlo/10;
136   swiatlo_dol_tlo = swiatlo_dol_tlo/10;
137 }
```

W procedurze ***swiatlo_otoczenia()*** nastąpi dziesięciokrotne odczytanie czujników górnego i dolnego, a następnie jako światło tła zostanie przyjęta wartość średnia (dla każdego z czujników z osobna).

Procedura ta zostanie po raz pierwszy wywołana w części ***setup()***, a następnie co jakiś czas (ustalony poprzez wartość zmiennej ***dczas***) będzie wywoływana w pętli ***while()***.

Pętla ***while()*** zostanie wzbogacona o instrukcje (linia 46, oraz 51-55):

```
42  void loop()
43 {
44     pobudka();
45     czas_1 = 0;
46     czas_0 = millis();
47     while(dzien)
48     {
49        swiatlo_gora = analogRead(CZ_G);
50        swiatlo_dol = analogRead(CZ_D);
51        if((millis() - czas_0) > dczas)
52        {
53         swiatlo_otoczenia();
54         czas_0 = millis();
55        }
56        if ((swiatlo_gora > dzien_noc) and (swiatlo_dol > dzien_noc))
57        {
58          if (czas_1 == 0) czas_1 = millis();
59          if ((millis() - czas_1) > czas_dzien_noc) dzien = false;
60        }
61        else czas_1 = 0;
```

Po rozpoznaniu przysłonięcia czujnika górnego, powinna nastąpić reakcja robota. Konieczne będzie stworzenie odpowiedniej procedury, np.:

```
135 void reakcja_1()
136 {
137   tone(GL,300);
138   delay(300);
139   noTone(GL);
140   delay(100);
141   tone(GL,600);
142   delay(300);
143   noTone(GL);
144   delay(100);
145   tone(GL,300);
146   delay(300);
147   noTone(GL);
148   delay(100);
149   tone(GL,1000);
150   delay(300);
151   noTone(GL);
152  }
```

Następnie do pętli ***while()*** należy dodać instrukcję:

```
62        if ((swiatlo_gora - swiatlo_gora_tlo) > dswiatlo) reakcja_1();
```

Całość w pliku – ***Robocik_program_4***.

Aby robot realizował wszystkie zaplanowane funkcjonalności pozostaje stworzenie odpowiednich procedur realizujących pozostałe reakcje. Ponadto program należy wzbogacić o kilka instrukcji warunkowych m. innymi w celu realizacji reakcji na bodźce (przysłonięcie czujnika górnego) występujące bezpośrednio po sobie.

Kompletny program znajdziesz w pliku ***Robocik_program_5***, istotne zmiany w stosunku do poprzednich wersji to:

- w pętli ***while()*** „kręcącej się" gdy jest dzień:

  - instrukcja wybierająca różne reakcje w zależności od kolejności przysłaniania górnego czujnika – bodźce zewnętrzne i reakcje na te bodźce zliczane są w określonym przedziale czasowym;
  - realizacja reakcji na brak bodźców w określonym przedziale czasowym

- w drugiej pętli ***while()*** - inny czas wykrywania dnia;

- dodatkowe procedury realizujące różne reakcje;

- kosmetyczne zmiany w programie.

UWAGI:

1. Przed ostatecznym złożeniem sprawdź pasowanie elementów, w zależności od jakości wydruku może być konieczne delikatne spiłowanie niektórych krawędzi.
2. Jeśli zdecydujesz się na umieszczenie rezystorów na płytce drukowanej, w płytce oprócz pinów łączących ją z Arduino umieść również gniazda lub piny do podłączenia przewodów. Przed podłączeniem przewodów do płytki konieczne będzie ich przeciągnięcie przez tułów robota. Zaplanuj wykonanie płytki drukowanej tak, aby po zamontowaniu na Arduino swobodnie zmieściła się w dostępnej przestrzeni.
3. Przed ostatecznym złożeniem robota sprawdź długość przewodów. Długość przewodów dobierz tak, aby możliwe było ich swobodne przeciągnięcie przez tułów i podpięcie do Arduino (płytki drukowanej).