AGH

Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie

Złożone systemy cyfrowe
2019/2020

Rastrowy projektor laserowy
Dokumentacja techniczna

Mateusz Wieczorek

January 22, 2020

Contents

0.1 Wstępny plan projektu.

0.1.1 Cel projektu.

Celem niniejszego projektu jest pogłębienie wiedzy i praktycznego doświadczenia w zakresie projektowania, konstruowania, budowania i programowania złożonego systemu cyfrowego. Ostatecznie celem projektu będzie w miarę możliwości otrzymanie działającego urządzenia, który będzie pełnił przedstawione później funkcje.

Zaprojektowane przeze mnie urządzenie powinno pełnić rolę projektora, który będzie w stanie wyświetlić obraz o wystarczająco dobrej rozdzielczośći, który zostanie podany na wejście VGA. Obraz będzie wyświetlany po przez bardzo szybkie skanowanie wiązką laserową w osi pionowej i poziomej, wiązka laserowa będzie tworzyć poziome linie. W ten sposób uzyskamy obraz rastrowy (nie wektorowy jak w przypadku standardowych projektorów laserowych).

0.1.2 Założenia projektu.

Głównymi założeniami są:

a) dostępność materiałów konstrukcyjnych w możliwie najniższych cenach:

- programowalny układ cyfrowy służący do zarządzania pracą urządzenia
- lasery na prąd elektryczny służące jako źródło światła
- silniki elektryczne służące jako układ odchylający wiązkę laserową w poziomie i pionie
- materiał na walce, które zostaną zamontowane na osiach silników, oraz małe zwierciadełka
- ewentualne soczewki pozwalające na korekcje wiązki laserowej
- zwierciadła dichroiczne
- czujniki obrotów
- wszelakie okablowanie
- części na obudowę urządzenia, komponentów oraz szkielet

b) dostępność wolnego czasu

c) brak większych trudności w zakresie praw fizyki uniemożliwiających konstrukcję urządzenia

d) dostępność narzędzi do zbudowania urządzenia

0.1.3 Zastosowany układ cyfrowy

Do sterowania pracą całego urządzenia użyję układu ATmega

0.1.4 Kolejne etapy rozwoju projektu.

1. Urządzenie powinno rysować z jak największą częstotliwością poziomą linię.

   Prosty układ powinien wyświetlać poziomą linię z jak największą częstotliwością. Na tym etapie będzie wymagane odpowiednie podłączenie lasera do układu cyfrowego, który będzie źródłem światła, oraz silnika elektrycznej charakteryzującego się wysokimi obrotami. Laser powinno się dać włączać i wyłączać programowo, tak samo powinno się dać sterować i kontrolować obroty silnika elektrycznego. Na osi silnika powinien być zamontowany płaski walec, na którego powierzchni bocznej będą przyklejone symetrycznie zwierciadełka o wymiarach 1cm x 1cm. Ten komponent będzie odpowiedzialny za odchylanie wiązki w kierunku poziomym. Ilość rysowanych linii w ciągu sekundy będzie miała znaczenie w kwestii pionowej rozdzielczości oraz częśtotliwości odżwierzania obrazu.

2. Urządzenie powinno rysować z jak największą częstotliwośćią jak najwięcej poziomych linii.

   Do układu powinien zostać domontowany dodatkowy silnik elektryczny charakteryzujący się niskimi obrotami, na osi którego będzie zamontowany podobny jak wcześniej walec z przyklejonymi zwierciadełkami na jego powierzchni bocznej. Ten komponent będzie odpowiedzialny za odchylenie odchylonej wcześniej wiązki laserowej w kierunku pionowym.

3. Wszystkie komponenty urządzenia powinny zostać odpowiednio zsynchronizowane.

   Należy odpowiednio zsynchronizować prędkości obrotowe obydwu silników elektrycznych według następujących relacji:

   N_{EH} - prędkość obrotowa silnika elektrycznego służącego do odchylania wiązki w kierunku poziomym $[obr./min.]$

   N_{EV} - prędkość obrotowa silnika elektrycznego służącego do odchylania wiązki w kierunku pionowym $[obr./min.]$

   M_{EH} - ilość zwierciadełek przyklejonych na powierchni bocznej walca służącego do odchylania wiązki w kierunku poziomym $[j.]$

   M_{EV} - ilość zwierciadełek przyklejonych na powierchni bocznej walca służącego do odchylania wiązki w kierunku pionowym $[j.]$

   W_V - rozdzielczość wyświetlanego obrazu w poziomie $[j.]$

   H_V - rozdzielczość wyświetlanego obrazu w pionie $[j.]$

   F_V - częstotliwość wyświetlania obrazu $[Hz]$

   $H_V * F_V = N_{EH} * M_{EH}/60$

   $F_V = N_{EV} * M_{EV}$

4. Urządzenie powinno umożliwiać kontrolę nad wyświetlanym obrazem.

   Należy zsynchronizować włączanie i wyłączanie lasera razem z prędkościami obrotowymi silników w taki sposób, żeby móc zapalać lub wygaszać odpowiedni piksel wyświetlany na ekranie. Aby zarządzać stanem piksela o współrzędnych x i y w danej klatce, należy kontrolować napięcie przyłożone na laser w przedziale czasowym $t = [(1/F_V) * (y * W_V + x)/(W_V * H_V)), (1/F_V) * (y * W_V + x + 1)/(W_V * H_V)]$

5. Urządzenie powinno odpowiednio analizować sygnał podany na wejściu VGA oraz skalować odbierany obraz.

   Układ cyfrowy powinien prawidłowo interpretować sygnał podawany na wejściu VGA, oraz skalować przesyłany obraz do rozdzielczości natywnej urządzenia za pomocą zwykłego algorytmu nearest-neighbour.

6. Urządzenie powinno być w stanie wyświetlić obraz podawany na wejściu VGA w formacie monochromatycznym.

   Urządzenie powinno sterować stanem wszystkich pikseli na podstawie odbieranego sygnału na wejściu VGA. Dany piksel powinien się zapalić wtedy i tylko wtedy, gdy chociaż jedna składowa transmitowanego piksela będzie nie mniejsza niż 50% intensywności.

7. Urządzenie powinno być w stanie wyświetlić obraz podawany na wejsciu VGA w formacie odcieni szarości.

   Urządzenie powinno sterować stanem wszystkich pikseli na podstawie odbieranego sygnału na wejściu VGA. Dany piksel powinien mieć intensywność równą kolorowi RGB transmitowanego piksela rzutowanego na odcienie szarości.

8. Urządzenie powinno być w stanie wyświetlić obraz podawany na wejściu VGA w formacie RGB.

   Urządzenie powinno sterować stanem wszystkich pikseli na podstawie odbieranego sygnału na wejściu VGA. Aby osiągnąć model RGB, należy do całego układu domontować lasery o kolorze niebieskim i zielonym, odpowiednio kontrolować napięcie na nich, oraz zwierciadła dichroiczne, które będą scalać wiązki różnych kolorów w pojedynczą wiązkę. Napięcie dla poszczególnych laserów powinno być wprost proporcjonalne do wartości odpowiednich składowych transmitowanego piksela.

0.2 Właściwa realizacja projektu

0.2.1 Zakupione i wykorzystane przedmioty w projekcie

a) 100g zwierciadełek prostych o zbliżonych wymiarach 1cm x 1cm (większość ze względu na wady produkcyjne nie nadawała się do użycia).

W projekcie posłużyły ostatecznie do zbudowania układu optycznego, dzięki któremu można było umieścić źródło światła w dowolnym miejscu (w przyszłości jak najbliżej radiatora).

b) Silnik szczotkowy na prąd stały Redox o maksymalnej prędkośći obrotowej 20000 obr./min., poborze prądu przy starcie 7,6 A, podczas pracy 1,8 A, napięciu znamionowym 7,2 V - 1sztuka.

W projekcie został wykorzystany jako napęd wirujących luster odchylających puszczaną na nie wiązkę laserową w kierunku poziomym (potrzebna jest możliwie jak największa prędkość oraz moment obrotowy, dlatego został użyty taki silnik).

c) Dioda laserowa czerwona 5mW, 650nm, na napięcie 5V, moduł Iduino ST1172 - 1 sztuka.

W projekcie została użyta jako źródło mocnego, skupionego światła, dzięki czemu można było swobodnie manipulować wiązką i nie potrzebne było stosowanie soczewek.

d) Płytka stykowa, 830 otworów, 126 wierszy, 4 magistrale zasilające - 1 sztuka.

   W projekcie posłużyła jako płytka prototypowa (podstawa) dla elektroniki projektowane systemu.

e) Mikrokontroler AVR - ATMega8A-PU DIP - 1 sztuka.

   W projekcie został użyty jako główny procesor systemu, który za pomocą programatora, taśmy i odpowiednio podpiętych przewodów można było łatwo programować.

f) Programator AVR zgodny z USBasp ISP + taśma IDC - po 1 sztuce.

   W projekcie ten zestaw posłużył jako interfejs między komputerem (w którym można było napisać kod źródłowy, na podstawie którego został wygenerowany program dla mikrokontrolera) a mikrokontrolerem (który sterował systemem).

g) Zestaw przewodów połączeniowych 20cm, m-m, m-ż, ż-ż, każdy rodzaj po 40 sztuk.

   W projekcie posłużyły jako "luźne" połączenia pomiędzy odpowiednimi komponentami urządzenia. Kilka złączonych razem w jeden przewód stanowiło długi kabel.

h) Zestaw 140 przewodów do płytek stykowych.

   W projekcie posłużyły do łączenia wierszy płytki stykowej w celu zbudowania odpowiedniego układu elektronicznego.

i) Zestaw diod LED 5mm - 4 sztuki.

   W projekcie posłużyły jako kontrolki - interfejs wyjściowy dla użytkownika.

j) Zestawy rezystorów 1/4W: 10 kOhm, 1 kOhm, 100 Ohm, 1 Ohm.

   W projekcie zostały użyte jako rezystory redukujące prąd w poszczególnych obwodach.

k) Nakrętka chromowana do baterii wannowej 3/4 - 1 sztuka.

   W projekcie została użyta jako wirujące lustra (gdyż jest to prawdopodobnie symetryczny element, o przekroju poprzecznych zbliżonym do sześciokąta foremnego, posiadający ściane boczne w formie "zwierciadeł płaskich".

l) Odbiornik i nadajnik podczerwieni LiteOn 940nm - 1 para.

   W projekcie para tych elementów posłużyła do testowania fazy obrotu nakrętki chromowanej, a przez to i także do zliczania prędkości obrotowej silnika.

m) Tack Switch z nasadką - 5 sztuk.

   W projekcie zostały użyte jako przyciski - interfejs wejściowy dla użytkownika.

n) Tranzystor bipolarny NPN BD911 100V/15A - 1 sztuka.

   W projekcie został użyty jako "klucz" obwodu silnika pracującego przy wysokim prądzie. Dzięki niemu możliwa była zmiana prędkości obrotowej za pomocą pośrednio podłączonego na bramkę sygnału PWM z mikrokontrolera.

o) Tranzystor bipolarny NPN BC639 80V/1A - 1 sztuka.

   W projekcie został wykorzystany w układzie Darlingtona, którego zadaniem było sterowanie obrotami silnika. Był on potrzebny, gdyż szyna zasilająca z portu USB 5V była za słaba aby pełnić tą funkcję prawidłowo.

p) dioda zaporowa o dużym maksymalnym prądzie i napięciu.

   W projekcie została wykorzystana jako zabezpieczenie całego układu przed zniszczeniem z powodu odłączenia zasilania z pracującego silnika, który w tej sytuacji pracował jak cewka.

q) Tranzystor unipolarny typu nMOSFET, o małym maksymalnym prądzie i napięciu - 3 sztuki.

   W projekcie posłużyły jako elektronika dla mniejszych prądów, bezpośrednio na płytce stykowej.

r) Nakrętka metalowa - 1 sztuka.

   W projekcie posłużyła do utrzymywania nakrętki chromowanej oraz możliwie jak najlepszego jej wyśrodkowania.

s) Listewki drewniane, grube tekturki, brzeszczot, klej do drewna.

   W projekcie zostały wykorzystane do zbudowania szkieletu konstrukcji, do którego zostały zamontowane poszczególne komponenty.

0.2.2 Wydobyte z niedziałających urządzeń przedmioty wykorzystane w projekcie.

a) Wentylator 8cm x 8cm (wydobyty z niesprawnego zasilacza komputerowego)

   W projekcie posłużył jako wentylator chłodzący zamontowany radiator, do którego przykręcony był tranzystor sterujący silnikiem (pracując przy wysokim prądzie bardzo szybko się nagrzewał).

b) Prosty radiator - gruba alumiowa płytka (wydobyty z niesprawnego zasilacza komputerowego)

   W projekcie posłużył jako radiator tranzystora sterującego pracą silnika.

c) Oraz inne mniejsze rzeczy, które zostały wykorzystane do konstrukcji.

0.2.3 Inne wykorzystane urządzenia.

Zasilacz komputerowy.

W projekcie posłużył jako stabilne źródło zasilania elementów nie będących częścią elektroniki cyfrowej.

0.2.4 Schemat obwodów elektrycznych w ostatecznej wersji urządzenia.

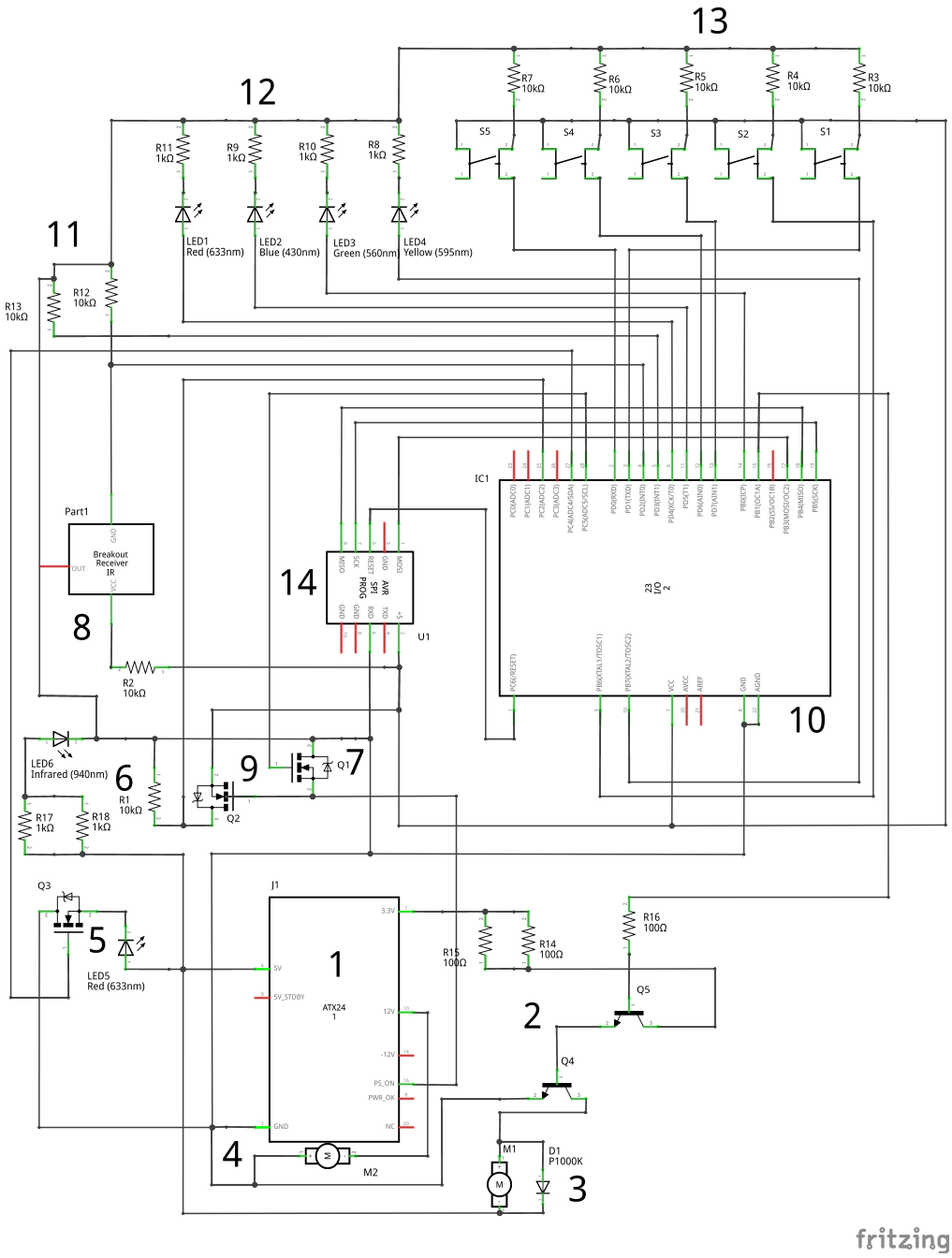


Figure 1: Kompletny schemat obwodów elektrycznych w urządzeniu.

0.2.5 Opis poszczególnych grup zaznaczonych na schemacie.

1. Zasilacz komputerowy.

   Zasilacz komputerowy po podłączeniu zasilania jest uruchomiony tylko wtedy, gdy pin PS_ON jest zwarty z masą tego zasilacza. Takie zwarcie powoduje otwarcie się tranzystora nMOSFET oznaczonego grupą 7. Z tego zasilacza zostały wykorzystane linie 3,3 V, 5V oraz 12V.

2. Układ Darlingtona służący do sterowania pracą silnika wirujących luster.

   Na bazę tranzystora NPN Q5 podawany jest sygnał PWM z ATMegi, który w zależności od wypełnienia powoduje szybszą oraz wolniejszą pracę silnika. Przy podawaniu logicznej jedynki tranzystor ten jest otwierany, a obwód 3,3V - rezystory - kolektor i emiter tanzystora Q5 - baza i emiter tranzystora Q4 jest zamykany, w związku z czym tranzystor NPN Q4 (przystosowany do dużych mocy) zaczyna przewodzić prąd. Skutkiem tego jest zamknięcie obwodu 5V - silnik - kolektor i emiter tranzystora Q4, a więc i uruchomienie silnika oznaczonego grupą 3.

   Do tego zadania nie można było użyć jedynie tranzystora Q4, gdyż linia 5V z portu USB była zbyt słaba aby otworzyć ten tranzystor (również gwałtownie spadało napięcie w całym obwodzie)

3. Silnik - napęd wirujących luster odchylających wiązkę lasera w poziomym kierunku.

   Ten silnik na prąd stały, wysokich obrotów został wykorzystany jak napęd modułu wirujących luster. Ze względu na swoją specyfikację, podczas pracy pobiera duży prąd, a więc i w chwili wyłączania mógłby zniszczyć układ. Z tego powodu została zastosowana dioda zaporowa, która po odłączeniu silnika (który zaczyna pracować jak cewka) zwiera jego piny.

4. Wentylator.

   Wentylator dla tego urządzenia zaczął być konieczny po zaimplementowaniu sterowania silnikiem. Tranzystor zamykający obwód silnika, pod wpływem wysokiego prądu zaczął się gwałtownie przegrzewać, przez co było potrzebne przykręcenie go do radiatora. Jednak to nie wystarczało, gdyż wtedy cały radiator zaczął się nagrzewać do wysokich temperatur, zatem trzeba było zapewnić przepływ powietrza wokół radiatora.

5. Obwód lasera - źródła światła w projektorze.

   W skład obwodu wchodzi tranzystor nMOSFET małych mocy, który umieszony w obwodzie w kolejności za diodą laserową, działał jak klucz, przez co można było wyłączać i włączać laser. Pomimo napięcia 5V z zasilacza przykładanego na pin anodę diody laserowej, nie potrzebny był tranzystor dużych mocy, gdyż maksymalny pobór prądu wynosił

około 25 mA. Na bramkę tranzystora jest podawany sygnał z pinu PC4 mikrokontrolera.

6. Obwód nadajnika podczerwieni.

   Jest to prosty obwód z koniecznie dołożonymi rezystorami zmniejszającymi prąd przepływający przez diodę IR. Po uruchomieniu zasilacza nadajnik zaczyna wysyłać promieniowanie podczerwone. Zasilanie z linii 5V zasilacza. Nadajnik wraz z odbiornikiem służą do wyzwalania przerwań układowych w mikrokontrolerze, dzięki czemu urządzenie wie, w której fazie obrotu znajdują się wirujące lustra.

7. Tranzystor służący jako klucz do sterowania pracą zasilacza.

   Jest to tranzystor nMOSFET małych mocy, który po otwarciu zwiera pin PS_ON z pinem masy w zasilaczu, przez co zasilacz się uruchamia. Na bramkę tego tranzystora podawana jest wartość logiczna z pinu PC5 mikrokontrolera.

8. Obwód odbiornika podczerwieni.

   Jest to prosty obwód z rezystorem ograniczającym prąd oraz fototranzystorem, który po odebraniu promieniowania podczerwonego się otwiera i zaczyna przewodzić prąd. Obwód zasilany jest linią 5V z portu USB. Po otwarciu tranzystora prąd trafia na pin przerwania układowego 0 mikrokontrolera.

9. Tranzystor służący do testowania obecności zasilania doprowadzanego do zasilacza.

   Gdy zasilacza jest włączony przełącznikiem, ale nie pracuje, na pinie PS_ON pojawia się napięcie 5,5 V. Właśnie to napięcie służy do sprawdzania, czy zasilacz jest gotowy do pracy. Podawane na bramkę tranzystora nMOSFET otwiera go, przez co napięcie 5V z linii z portu USB trafia na pin PC2 mikrokontrolera.

10. Główny mikrokontroler urządzenia - ATMega8A-PU.

    Pełni on rolę procesora urządzenia. Jego wykorzystanie i działanie, wraz z kodem źródłowym zostanie przedstawione w dalszych sekcjach.

11. Grupa rezystorów dla pinów przerwań układowych mikrokontrolera.

    Zastosowanie tych tranzystorów wymusza przepływ prądu do pinu mikrokontrolera w przypadku logicznej jedynki, natomiast w przypadku logicznego zera = braku prądu łączy z masą.

12. Grupa czterech diod LED - kontrolek. Interfejs wyjściowy dla użytkownika.

Jest to grupa czterech podobnych obwodów elektrycznych. Każdy obwód zaczyna się od określonego pinu (odpowiadającego konkretnej kontrolce), diody LED i rezystora ograniczającego prąd w obwodzie.

13. Grupa pięciu przycisków typu Tact Switch. Interfejs wejściowy dla użytkownika.

    Jest to grupa pięciu podobnych do siebie obwodów elektrycznych. W skład każdego obwodu wchodzą: linia 5V z portu USB z jednej strony, masa poprzedzona rezystorem ograniczającym prąd w przypadku naciśnięcia przycisku oraz wyjście do konkretnego pinu mikrokontrolera z drugiej strony. Po naciśnięciu przycisku zwierają sie obie strony, a z powodu obecności rezystora prąd trafia na pin.

14. Programator AVR.

    Jest to główny element służący do zasilania elektroniki cyfrowej urządzenia oraz do programowania mikrokontrolera za pomocą linii MISO, MOSI i SCK.

0.2.6 Kod źródłowy mikrokontrolera w ostatecznej wersji.

```
1  /*
2   * MyFirstProject.c
3   *
4   * Created: 2019-11-30 14:51:10
5   * Author : t530
6   *
7
8  #ifndef F_CPU         // if F_CPU was not defined in Project ->
       Properties
9  #define F_CPU 1000000UL     // define it now as 1 MHz unsigned long
10 #endif
11
12 #include <avr/io.h>       // this is always included in AVR programs
13 /*
14  * RasterLaserProjector.c
15  *
16  * Created: 2019-12-12 09:51:29
17  * Author : t530
18  */
19
20 #ifndef F_CPU         // if F_CPU was not defined in Project ->
       Properties
21 #define F_CPU 8000000UL     // define it now as 8 MHz unsigned long
22 #endif
23
24 #include <avr/io.h>
25 #include <avr/interrupt.h>
26 #include <util/delay.h>
27
28
29 // zdefiniowana funkcja s  u   c a  do synchronizacji ustawie
30 void _NOP(){
```

```
31   __asm__ __volatile__ ("nop");
32 }
33
34 // grupa funkcji s u   cych do zapalania (1) i zgaszania (0)
       poszczeg lnych diod LED
35 void setYellowLEDState(char newState){
36   if(newState==0)
37     PORTB &= ~(1<<PB7);
38   else
39     PORTB |= (1<<PB7);
40 }
41
42 void setGreenLEDState(char newState){
43   if(newState==0)
44     PORTB &= ~(1<<PB0);
45   else
46     PORTB |= (1<<PB0);
47 }
48
49 void setBlueLEDState(char newState){
50   if(newState==0)
51     PORTD &= ~(1<<PD5);
52   else
53     PORTD |= (1<<PD5);
54 }
55
56 void setRedLEDState(char newState){
57   if(newState==0)
58     PORTD &= ~(1<<PD4);
59   else
60     PORTD |= (1<<PD4);
61 }
62
63 // funkcja zwracaj ca 1 w przypadku gotowo ci zasilacza do pracy,
64 // 0 w przeciwnym wypadku
65 char getPowerSupplyStatus(){
66   char cData = PINC;
67   if(cData & (1<<PC2))
68     return 1;
69   return 0;
70 }
71
72 // zmienna globalna s u   ca do pomini cia kolejnych odczyt w
       naci ni cia
73 // przycisk w w przypadku jego niemo liwego do pomini cia
       przytrzymania
74 char buttonReady;
75
76 // funckja zwracaj ca 1, gdy przycisk od przekazanym id zosta
       wci ni ty ,
77 // 0 w przeciwnym wypadku
78 char get1ButtonState(char buttonId){
79   char bData = PINB;
80   char dData = PIND;
81   switch(buttonId){
82     case 1:
83       if(dData & (1<<PD1)){
84         if(buttonReady & (1<<buttonId)){
85           buttonReady &= ~(1<<buttonId);
```

```
86            return 1;
87          }
88        }
89        else
90          buttonReady |= 1<<buttonId;
91        break;
92      case 2:
93        if(bData & (1<<PB6)){
94          if(buttonReady & (1<<buttonId)){
95            buttonReady &= ~(1<<buttonId);
96            return 1;
97          }
98        }
99        else
100         buttonReady |= 1<<buttonId;
101       break;
102     case 3:
103       if(dData & (1<<PD7)){
104         if(buttonReady & (1<<buttonId)){
105           buttonReady &= ~(1<<buttonId);
106           return 1;
107         }
108       }
109       else
110         buttonReady |= 1<<buttonId;
111       break;
112     case 4:
113       if(dData & (1<<PD6)){
114         if(buttonReady & (1<<buttonId)){
115           buttonReady &= ~(1<<buttonId);
116           return 1;
117         }
118       }
119       else
120         buttonReady |= 1<<buttonId;
121       break;
122     case 5:
123       if(dData & (1<<PD0)){
124         if(buttonReady & (1<<buttonId)){
125           buttonReady &= ~(1<<buttonId);
126           return 1;
127         }
128       }
129       else
130         buttonReady |= 1<<buttonId;
131       break;
132   }
133
134   return 0;
135 }
136
137 // zmienna globalna przechowuj ca flagi dotycz ce pracy urz dzenia
138 char systemFlags;
139 #define SYSTEM_READY 0x01 // zasilacz gotowy do pracy
140 #define SYSTEM_ON 0x02 // urz dzenie pracuje
141 #define SYSTEM_INITIATED 0x04 // urz dzenie zako czy o
        konfiguracj
142 #define SYSTEM_SHUTDOWN_ON 0x40 // urz dzenie jest w trakcie
        ko czenia pracy
```

```
143 #define SYSTEM_TO_RESET 0x80 // urz dzenie jest w stanie do
        zresetowania
144
145 // zmienna globalna przechowuj ca flagi b   d  w
146 char systemErrorFlags;
147 #define POWER_SUPPLY_ERROR 0x01 // zasilacz nie jest gotowy do pracy
148
149 long timeToReset;
150
151 // funkcja ustawiaj ca flagi systemowe
152 void setSystemFlags(char flagsMask){
153   systemFlags |= flagsMask;
154 }
155
156 // funkcja kasuj ca flagi systemowe
157 void unsetSystemFlags(char flagsMask){
158   systemFlags &= ~flagsMask;
159 }
160
161 // funkcja zwracaj ca postawienie danej flagi
162 char getSystemFlag(char flag){
163   if(systemFlags & flag)
164     return 1;
165   return 0;
166 }
167
168 // funkcja ustawiaj ca flagi b   d  w
169 void setSystemErrorFlags(char flagsMask){
170   systemErrorFlags |= flagsMask;
171 }
172
173 // funckja kasuj ca flagi b   d  w
174 void unsetSystemErrorFlags(char flagsMask){
175   systemErrorFlags &= ~flagsMask;
176 }
177
178
179 // funckja zwracaj ca flagi b   d  w
180 char getSystemErrorFlag(char flag){
181   if(systemErrorFlags & flag)
182     return 1;
183   return 0;
184 }
185
186 // funkcja zwracaj ca 1, gdy system jest w stanie w a ciwej pracy
187 char isSystemWorking(){
188   return (char)(getSystemFlag(SYSTEM_INITIATED) & !getSystemFlag(
        SYSTEM_SHUTDOWN_ON));
189 }
190
191 // funkcja w   czaj ca i wy   czaj ca zasilacz
192 void setPowerOnState(char on){
193   if(on == 0)
194     PORTC &= ~(1<<PC5);
195   else
196     PORTC |= (1<<PC5);
197 }
198
199 // funkcja w   czaj ca i wy   czaj ca laser
```

```
200 void setLaserState(char on){
201   if(on == 0)
202     PORTC &= ~(1<<PC4);
203   else
204     PORTC |= (1<<PC4);
205 }
206
207 // funkcja testuj ca, czy ma nast pi  resetowanie urz dzenia,
208 // oraz w razie konieczno ci go resetuje
209 void processShutDown(){
210   if(getSystemFlag(SYSTEM_SHUTDOWN_ON))
211     if(timeToReset > 0)
212       timeToReset--;
213   if(timeToReset == 0)
214     setSystemFlags(SYSTEM_TO_RESET);
215 }
216
217 // w kt rym miejscu ma zosta  narystowana linia
218 char drawNow = 0;
219
220 // licznik przerwa  uk adowych INT0
221 long int0Count;
222 ISR(INT0_vect)
223 {
224   setBlueLEDState(1);
225   drawNow = (drawNow + 1) % 6;
226   {
227     // rysowanie kresek
228     int i = 10;
229     while(i--){
230       if(i - 1 == drawNow)
231         setLaserState(1);
232       _delay_us(50);
233       setLaserState(0);
234       _delay_us(50);
235     }
236   }
237   int0Count++;
238   setBlueLEDState(0);
239 }
240
241 ISR(INT1_vect)
242 {
243   setGreenLEDState(1);
244   _delay_ms(1);
245   setGreenLEDState(0);
246   _delay_ms(1);
247 }
248
249
250 // funkcja kontroluj ca zasilanie
251 void controlPower(){
252   setPowerOnState(getSystemFlag(SYSTEM_ON));
253 }
254
255
256 // funkcja kontroluj ca prac  lasera
257 char laserOn;
258 char laserUpState;
```

```
259 void controlLaser(){
260   if(isSystemWorking()){
261     int i = 1000;
262     while(i--){
263       if(laserOn)
264       laserUpState = 1 - laserUpState;
265       else
266       laserUpState = 1;
267       setLaserState(laserUpState);
268       _delay_us(10);
269     }
270   }
271   else
272     setLaserState(0);
273 }
274
275 // funkcja ustawiaj ca uk adowy PWM, do kt rego podpi ty jest
      modu
276 // sterowania silnikiem
277 void setXAxisMotorSpeed(int speed){
278   if(speed >= 0 && speed < 1024)
279     OCR1A = speed;
280 }
281
282 // funkcja testuj ca dzia anie zasilania w uk adzie
283 int timeToNextAdvancedPowerSupplyCheck;
284 void checkPowerSupplyStatus(){
285   if(getSystemFlag(SYSTEM_ON) && timeToNextAdvancedPowerSupplyCheck ==
       0){
286     setPowerOnState(0);
287     _delay_us(10);
288   }
289   if(!getSystemFlag(SYSTEM_ON) || timeToNextAdvancedPowerSupplyCheck
      == 0){
290     if(getPowerSupplyStatus()){
291       setSystemFlags(SYSTEM_READY);
292       unsetSystemErrorFlags(POWER_SUPPLY_ERROR);
293     }
294     else{
295       unsetSystemFlags(0xFF);
296       setSystemErrorFlags(POWER_SUPPLY_ERROR);
297     }
298   }
299   if(getSystemFlag(SYSTEM_ON)){
300     if(timeToNextAdvancedPowerSupplyCheck == 0){
301       setPowerOnState(1);
302       _delay_us(10);
303       timeToNextAdvancedPowerSupplyCheck = 8000;
304     }
305     timeToNextAdvancedPowerSupplyCheck--;
306   }
307 }
308
309 // funkcja kontroluj ca prac  silnika (wy  cza go w odpowiednim
      momencie)
310 void controlXAxisMotor(){
311   if(isSystemWorking()){
312     //setXAxisMotorSpeed(512);
313   }
```

```
314   else
315     setXAxisMotorSpeed(0);
316 }
317
318 // funkcja dostosowuj ca obroty silnika do zadanych, za pomoc
319 // uk adu nadajnika i odbiornika podczerwieni
320 // (tak na prawd  to jest RPS/6 obrot w na sekund )
321 void adjustXAxisMotor(long targetRPS){
322   if(targetRPS < 10)
323     return;
324   long nextSpeedDelta = 320;
325   long curSpeed = nextSpeedDelta;
326   setXAxisMotorSpeed(curSpeed + 383);
327   _delay_ms(2000);
328   long curRPS = 0;
329   char measureFailure = 0;
330   nextSpeedDelta /= 2;
331   while(nextSpeedDelta){
332     long startInt0Count = int0Count;
333     _delay_ms(1000);
334     long endInt0Count = int0Count;
335     long delta = endInt0Count - startInt0Count;
336     if(delta < 10){
337       measureFailure++;
338       if(measureFailure == 5)
339         break;
340       continue;
341     }
342     else{
343       measureFailure = 0;
344       curRPS = delta;
345       if(curRPS > targetRPS){
346         curSpeed -= nextSpeedDelta;
347       }
348       else{
349         curSpeed += nextSpeedDelta;
350       }
351       setXAxisMotorSpeed(curSpeed + 383);
352       _delay_ms(2000);
353     }
354     nextSpeedDelta /= 2;
355   }
356   setBlueLEDState(1);
357   _delay_ms(2000);
358   setBlueLEDState(0);
359   int0Count = curRPS;
360 }
361
362 // funkcja obs ugi przycisku zasilania (przycisk 1)
363 long powerSwitchButtonSecondPress;
364 void handlePowerSwitchButton(){
365   if(getSystemFlag(SYSTEM_READY) && get1ButtonState(1))
366   {
367     if(!getSystemFlag(SYSTEM_ON)){
368       setSystemFlags(SYSTEM_ON|SYSTEM_INITIATED);
369       return;
370     }
371     if(getSystemFlag(SYSTEM_INITIATED)){
372       if(!powerSwitchButtonSecondPress)
```

```
373         powerSwitchButtonSecondPress = 5000;
374       else
375         setSystemFlags(SYSTEM_SHUTDOWN_ON);
376     }
377   }
378   if(powerSwitchButtonSecondPress > 0)
379     powerSwitchButtonSecondPress--;
380 }
381
382 // funcja obs ugi przycisku zmiany trybu pracy lasera (przycisk 2)
383 void handleLaserOnButton(){
384   if(getSystemFlag(SYSTEM_ON) && get1ButtonState(2))
385   {
386     laserOn = 1 - laserOn;
387     setBlueLEDState(laserOn);
388   }
389 }
390
391 // funkcja obs ugi przycisku pobieraj cego kolejne bity
392 // warto ci INT0Count
393 char int0CountNextBit = 0;
394 void handleGetInt0CountButton1(){
395   if(get1ButtonState(3)){
396     int copy = int0Count;
397     if(int0CountNextBit == 16)
398       int0CountNextBit = 0;
399     char temp = int0CountNextBit + 1;
400     char val = 0;
401     while(temp--){
402       val = copy % 2;
403       copy = copy / 2;
404     }
405     setBlueLEDState(1);
406     _delay_ms(10);
407     setBlueLEDState(0);
408     _delay_ms(100);
409     setBlueLEDState(val);
410     _delay_ms(10);
411     setBlueLEDState(0);
412     int0CountNextBit++;
413   }
414
415 }
416
417 // funkcja obs ugi przycisku s u   cego do resetowania odczytu
      bit w
418 // warto ci INT0Count
419 void handleGetInt0CountButton2(){
420   if(get1ButtonState(4)){
421     int0CountNextBit = 0;
422   }
423 }
424
425 // funkcja aktulizuj ca stan kontrolki od b   d  w urz dzenia
426 void updateErrorLEDState(){
427   setRedLEDState(systemErrorFlags);
428 }
429
430 // funkcja aktulizuj ca stan kontrolki informuj cej o gotowo ci
```

```
431 // zasilacza do pracy
432 void updatePowerSupplyLEDState(){
433   setYellowLEDState(getSystemFlag(SYSTEM_READY));
434 }
435
436 // funkcja aktulizuj ca stan kontrolki informuj cej o pracy
      urz dzenia
437 unsigned int systemOnLEDBlinkPoints;
438 char systemOnLEDstate;
439 void updateSystemOnLEDState(){
440   if(getSystemFlag(SYSTEM_ON)){
441     if(getSystemFlag(SYSTEM_INITIATED) & !getSystemFlag(
      SYSTEM_SHUTDOWN_ON)){
442       setGreenLEDState(1);
443     }
444     else{
445       if(systemOnLEDBlinkPoints == 0){
446         systemOnLEDBlinkPoints = 8000;
447         systemOnLEDstate = 1 - systemOnLEDstate;
448         setGreenLEDState(systemOnLEDstate);
449       }
450       systemOnLEDBlinkPoints--;
451     }
452   }
453   else{
454     setGreenLEDState(0);
455     systemOnLEDBlinkPoints = 0;
456     systemOnLEDstate = 0;
457   }
458 }
459
460 // funkcja aktulizuj ca wszystkie kontrolki
461 void updateLEDsState(){
462   updatePowerSupplyLEDState();
463   updateSystemOnLEDState();
464   updateErrorLEDState();
465 }
466
467 // funkcja inicjalizacji urz dzenia
468 void initSystem(){
469   setYellowLEDState(1);
470   setGreenLEDState(1);
471   setBlueLEDState(1);
472   setRedLEDState(1);
473   unsetSystemErrorFlags(0xFF);
474   unsetSystemFlags(0xFF);
475   timeToReset = 65000;
476   laserOn = 0;
477   laserUpState = 0;
478   controlLaser();
479   controlPower();
480   buttonReady = 0;
481   timeToNextAdvancedPowerSupplyCheck = 0;
482   systemOnLEDBlinkPoints = 0;
483   systemOnLEDstate = 0;
484   powerSwitchButtonSecondPress = 0;
485   int0Count = 0;
486   _delay_ms(2000);
487   setYellowLEDState(0);
```

```
488   setGreenLEDState(0);
489   setBlueLEDState(0);
490   setRedLEDState(0);
491 }
492 
493 int main(void)
494 {
495   // konfiguracja pin w przycisk w oraz diod LED
496   DDRB = (1<<PB7)|(1<<PB0)|(1<<PB1)|(1<<PB2);
497   DDRC = (1<<PC4)|(1<<PC5);
498   DDRD = (1<<PD5)|(1<<PD4);
499 
500   // konfiguracja pin w od przerwa  uk adowych
501   cli();
502   MCUCR = (MCUCR & 0b11110000) | 0b1010;
503   GICR = (GICR & 0b00111111) | (1<<INT0)|(1<<INT1);
504   //PORTD = (1<<PD2)|(1<<PD3);
505   sei();
506 
507   // konfiguracja sprz towego PWM
508   OCR1A = 0;
509   OCR1B = 0;
510 
511   TCCR1A |= (1 << COM1A1) | (1 << COM1B1);
512   // set non-inverting mode
513   TCCR1A |= (1 << WGM11) | (1 << WGM10);
514   // set 10bit phase corrected PWM Mode
515   TCCR1B |= (1 << CS20)|(1 << CS21);
516   // set prescaler to 8 and starts PWM
517 
518   //PORTB = (1<<PB6);
519   //PORTC = (1<<PC2);
520   //PORTD = (1<<PD1)|(1<<PD7)|(1<<PD6)|(1<<PD0);
521   // synchronizowanie uk adu
522   _NOP();
523 
524     // g   wna  p tla
525     while (1)
526     {
527       // inicjalizacja urz dzenia
528     initSystem();
529     // p tla  dzia ania  urz dzenia  mi dzy  jego resetami
530     while(!getSystemFlag(SYSTEM_TO_RESET)){
531       checkPowerSupplyStatus();
532       controlPower();
533       controlLaser();
534       controlXAxisMotor();
535       handlePowerSwitchButton();
536       //handleLaserOnButton();
537       // kod eksperymentalny
538       if(get1ButtonState(2)){
539         int0Count = 0;
540       }
541       handleGetInt0CountButton1();
542       handleGetInt0CountButton2();
543       if(get1ButtonState(5)){
544         /*setXAxisMotorSpeed(1023);
545         _delay_ms(100);
546         for(int i = 400; i < 1023; i++){
```

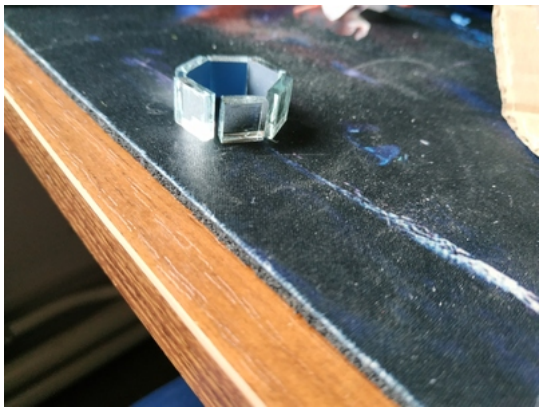
```
547             setXAxisMotorSpeed(i);
548             _delay_ms(100);
549             if(get1ButtonState(5)){
550               int0Count = i;
551               break;
552             }
553           }*/
554           adjustXAxisMotor(50);
555         }
556         processShutDown();
557         updateLEDsState();
558         _delay_us(1);
559       }
560       }
561 }
```

0.2.7 Budowanie poszczególnych modułów urządzenia.

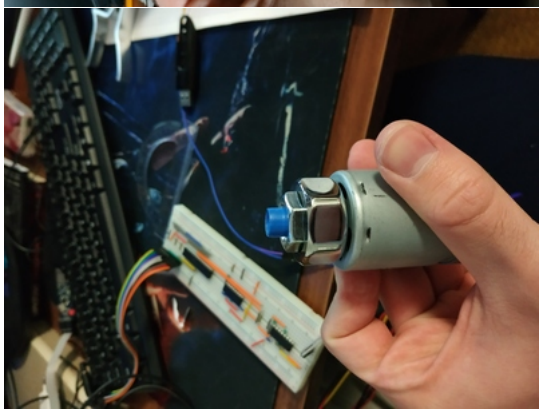
1. Moduł wirujących luster.

   Był to i jest nadal najtrudniejszy moduł do zaprojektowania i zbudowania. Problemem, który ciągnął się za mną przez cały czas była niemożliwość idealnego osadzenia wirującego komponentu na wale silnika, przez co przy większych obrotach pojawiało się bicie. Pomimo, że wirujący komponent nie wyskakiwał z osi, to powodował on dosyć spore wibracje, które były głośne.

   a) Na początku był pomysł z zakupieniem lekkiej nakrętki i zwierciadełek płaskich. Po dokonaniu zakupów, przykleiłem zwierciadełka do ścian bocznych nakrętki. Po stestowaniu modułu okazało się, że skończona różna ilość kleju użytego do przyklejnia każdego zwierciadełka powodowała odchylanie wiązki pod różnymi kątami, co było nie do zaakceptowania. Efekt został przedstawiony na filmiku 2.



b) Następnym pomysłem, który ostatecznie zaakceptowałem, było zakupienie nakrętki chromowanej do baterii wannowej, i za pomocą różnych symetrycznych przedmiotów osadzenie jej na wale silnika. Takie rozwiązanie ze względu na to, że proces produkcji takich nakrętek jest bardziej staranny niż ręczne klejenie, umożliwiło odpowiednie osadzenie nakrętki, co pozwoliło na naświetlenie jednej linii za pomocą każdej ze ścian nakrętki. Pomimo owalnych brzegów pomiędzy ścianami bocznymi, część pozostała zwierciadłem płaskim.



Jednak z powodu, że ta nakrętka nie była na szytwno osadzona na wale, to wraz prędkością obrotową silnika odchylała się pod minimalnie różnym kątem, co przedstawione zostało na filmiku 1.

2. Moduł napędu wirujących luster.

   Był to jeden z większych problemów podczas budowy układu. Z powodu zakupionego silnika wysokich obrotów, pobierał on bardzo duży prąd podczas pracy, a w szczególności podczas startu. Dlatego potrzebne było zastosowanie specjalnego tranzystora przystosowanego na duże prądy.

   a) Pierwszym podejściem, po przestraszeniu się działania tranzystorów bipolarnych i ich przewodzenia prądu pomiędzy pinami, zakupiłem tranzystory typu nMOSFET. Jednak z powodu, że pracuje on na wysokich prądach, potrzebował on specjalnego napięcia progowego, aby otworzyć ten tranzystor. Linia 5V z portu USB była tak słaba, że przy obciążeniu przez elektronikę cyfrową oraz inne komponenty, nie wystarczała na jakiekolwiek otwarcie tego tranzystora. Nad tym problemem spędziłem dużo czasu, próbując różnych podejść, czasami również śmiesznych.

   b) Po konsultacjach z prowadzącym okazało się, że ten problem może rozwiązać tranzystor bipolarny, gdyż nie wymaga on odpowiedniego napięcia baza-emiter względem kolektor-emiter. Po zastosowaniu zakupionego na początku tranzystora NPN dużych prądów, problem się znacznie rozwiązał, jednak nadal słaba linia 5V z portu USB nie wystarczała na sterowanie tranzystorem, a w dodatku gwałtownie spadało napięcie na całym obwodzie. Dlatego podświadomie wykorzystałem układ Darlingtona, wykorzystując przy tym tanzystor NPN małych prądów, który było otwierany na podstaiwe sygnału PWM z mikrokontroleta, a to powodowało podanie prądu z linii 3,3 V z zasilacza na bazę właściwego tranzystora. Dzięki temu układowi udało się w 100% sterować obrotami silnika.

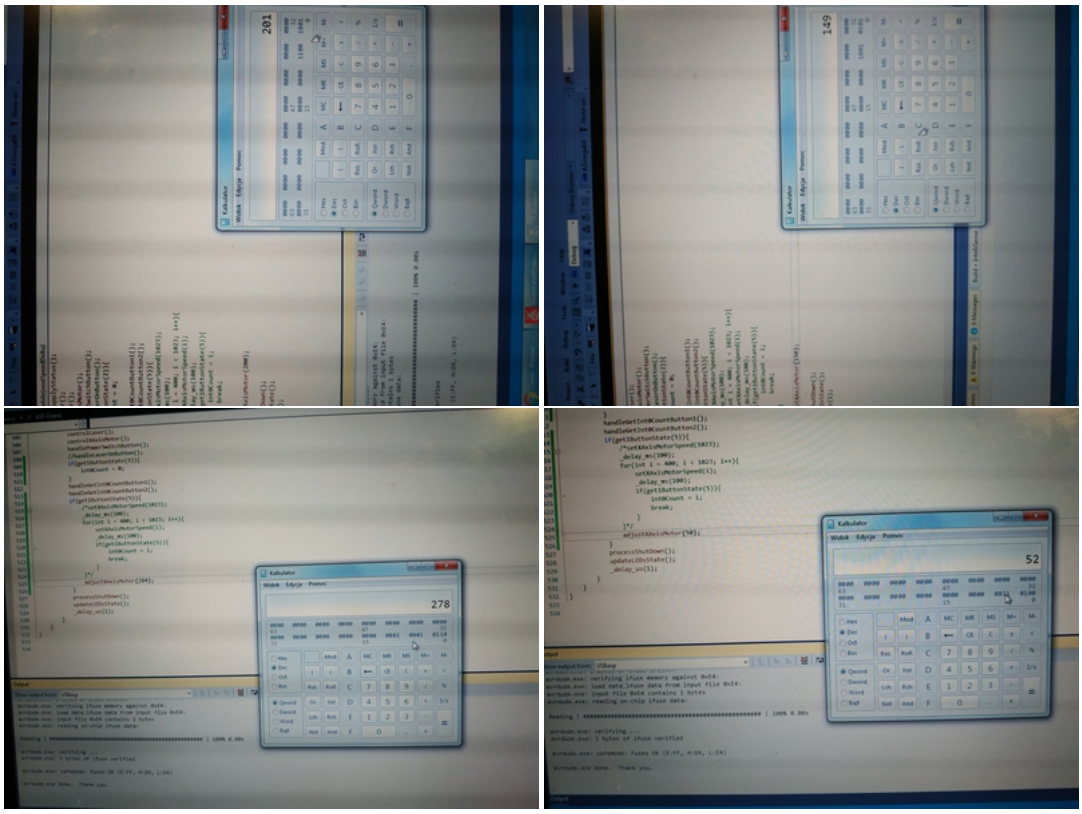
3. Moduł nadajnika i odbiornika podczerwieni.

   Z tym modułem nie było żadnych problemów. Od samego początku spełniał zaplanowane przeze mnie zadanie. Jednak przed wymyśleniem zastosowania takiego układu do sprawdzania fazy obrotu nakrętki minęło dużo czasu. Właściwe podpięcie nadajnika i odbiornika do płyty głównej oraz przede wszystkim właściwe wzajemne ustawienie tych komponentów względem siebie i wirującej nakrętki pozwoliło na bardzo szybki, dokładny i sprawny test fazy obrotu nakrętki (takich faz było 6, gdyż każda ze ścian odbijała w pewnym momencie promieniowanie podczerwone z nadajnika do odbiornika).

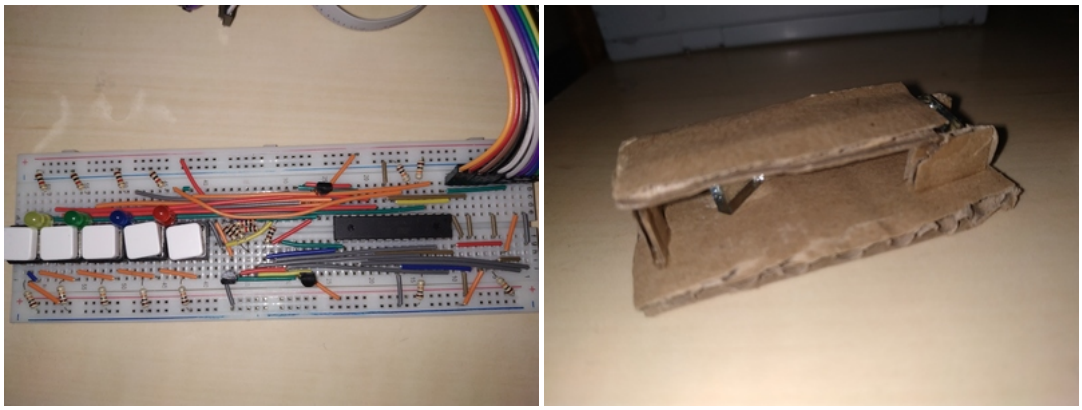
   Po odebraniu promieniowania podczerwonego, wywoływała się funkcja obsługi przerwania systemowego INT0.

   Dzięki odpowiednio napisanej funkcji przedstawionej w sekcji wyżej, udało się uzyskać odpowiednią, zadaną prędkość obrotową silnika. Udokumentowane zostało do na zdjęciach oraz filmiku 4. Dokładna wartość prędkości obrotowej została odczytana przy pomocy przycisków 3 i 4, co również zostało przedstawione na filmiku. Po wciśnięciu przycisku 5, uruchamia się algorytm połówkowego znajdowania PWM, który przy odpowiednim zasilaniu spowoduje, że silnik będzie się obracał z zadaną prędkością. Należy odpowiednio umieścić względem siebie nadajnik

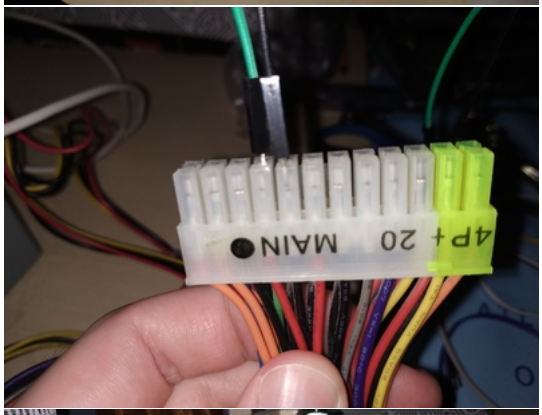
i odbiornik podczerwieni, aby lekko świeciła się niebieskia kontrolka. Świeci się ona, gdy wywoływane jest przerwanie układowe. Po jej mignięciu kończy się działanie algorytmu, laży wtedy odstawić moduł, aby zachować ostatnią uzyskaną prędkość obrotową.



4. Ogólna konstrukcja urządzenia.




SATA


MAIN
20
4P+