POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Projekt z Systemów Mikroprocesorowych

Czujnik Parkowania

Autorzy: Marek Gałeczka

Rok akademicki: 3, semestr: 5, grupa: 6:

Kierujący pracą: dr inż. Jacek Loska

Gliwice, grudzień 2022

Spis treści

1. Wstęp

Projekt, który został wybrany w sekcji składającej się z dwóch osób: Marka Gałeczki i Kamila Dziurdzii to czujnik parkowania. Początkowo tematyka projektu miała być zupełnie inna, jednak po konsultacjach z prowadzącym zajęcia zaproponowano właśnie taki temat.

1.1. Cel i zakres projektu

Projekt noszący tytuł „Czujnik Parkowania" ma na celu stworzenie podstawowego systemu mikroprocesorowego, który swoim działaniem ma przypominać czujnik parkowania. Oczywistym jest fakt, że nie działa on dokładnie jak modele, które montowane są w nowszych samochodach, aczkolwiek niektóre elementy konstrukcji i działania są podobne.

Założenia początkowe naszego projektu były dosyć proste przy takim systemie czujnika. Miał on spełniać swoje działanie, czyli miał mierzyć odległość od czujnika do najbliższego obiektu. Ponadto gdy obiekt zbliżyłby się na odległość mniejszą niż ustalona system miał zareagować. Reakcja systemu powinna być całkiem standardowa, ponieważ kiedy obiekt wejdzie w strefę reakcji, ten wydawałby sygnały dźwiękowe. Do tego założeniem było, aby zamontowana została sygnalizacja świetlna. Początkowo zakładano także wyświetlacz LCD, który miałby pokazywać odległość od czujnika do najbliższego obiektu.

Długo rozmyślano nad komponentami, które zostaną wykorzystane do stworzenia tego projektu. Potrzebne komponenty to między innymi:

- Mikroprocesor
- Diody LED
- Czujnik Ultrasoniczny
- Buzzer

Na rynku jest bardzo dużo mikroprocesorów, od „Arduino„ aż do „Raspberrry Pi”. Założeniem projektu było wykorzystanie go budżetowo, niestety żaden z tych przykładów nie jest tanim zakupem. Dlatego założyliśmy szukanie tanich opcji. W trakcie szukania i pytania innych studentów, oraz własnych znajomych o to, czy mają dostęp do jakichś mikroprocesorów zyskaliśmy dostęp do całego zestawu testowego mikroprocesora Arduino. Modelem który wykorzystano to Arduino UNO. Jeden z podstawowych mikroprocesorów, który okazał się idealny do nauki i pierwszej styczności z programowaniem takiego systemu.

Diody LED nie były ciężkim wyborem. Jedyny wybór jaki został do wybrania to kolory. Od początku założyliśmy, że będą one dwukolorowe. Zastanawialiśmy się nad połączeniami kolorów. Ostatecznie wybraliśmy kolor czerwony i zielony. Same diody zostały dołączone do zestawu z mikroprocesorem Arduino UNO. Skorzystno właśnie z tych diod.

Następne rozterki związane z wyborem komponentów przypadły na czujnik ultrasoniczny. Na rynku jest wiele czujników tego typu. Cechą, na którą najbardziej rozmyślano była odległość czułości czujnika. Popularnym na rynku modelem są czujniki z odległością czujnika (2-400 cm). Innym wyborem nad którym prowadzono był czujnik z odległością czułości(2-200 cm). Ostatecznie zdecydowano, że czujnikiem jaki zasili nasz projekt będzie Czujnik z czułością (2-200 cm). Wybór padł na ten właśnie typ komponentu dlatego, że Projekt ten miał nie być zbyt duży, a czujnik z tą czułością wpasuje się idealnie do prezentacji w warunkach laboratoryjnych. Większy zakres działania byłby zbyteczny.

Buzzer to komponent, nad którym trzeba było się dłużej zastanowić. Z pozoru jeden z najmniejszych elementów systemu mikroprocesorowego, który tworzy ten projekt. Jednakże różne typy Buzzerów różnią się od siebie. Chociażby napięciem. Ze względu na to jaki mikroprocesor zastosowano zdecydowano się na 5 Woltowy buzzer. Drugą kwestią nad którą rozmyślano było to, czy zdecydować się na buzzer z generatorem czy też bez. Wybrano buzzer bez generatora, gdyż nie był on nam potrzebny, a koszty buzzera bez niego są znacznie mniejsze.

Projekt nie do końca został wykonany z wcześniejszymi założeniami. Zdecydowano się nie dodawać wyświetlacza. Powodem było dojście do wniosku, że psuje on aspekt wizualny całego projektu, oraz można go uznać za zbyteczny. Reszta założeń została spełniona.

Twórcy projektu nie mieli za dużo styczności z programowaniem systemów mikroprocesorowych w przeszłości, dlatego z początku ważnym elementem było oglądanie tutoriali z platformy YouTube. Po zdobyciu wiedzy potrzebnej do złożenia i zaprogramowania projektu przystąpiono do czynności związanych z poskładaniem całego systemu i stworzeniem kodu, który nim operuje.

Czujnik działa poprawnie po złożeniu i zaprogramowaniu. Był on testowany w warunkach domowych, gdzie jego działaniu nic nie przeszkadzało. Nie został on sprawdzony w warunkach garażowych.

Przyszłość tego projektu jest jak najbardziej realna. Zawsze można taki system zmniejszyć, zastosować inne komponenty, które posiadają dużo mniejszy rozmiar. Być może dodać jakieś elementy, które poprawiają działanie samego systemu.

Początkowy cel nie został wykonany w 100%, aczkolwiek uznano to za właściwy tor tego projektu. Samo ustalanie jak wyglądać ma ostateczny owoc pracy, dobieranie komponentów, oraz planowanie z tygodnia na tydzień co trzeba zrobić bardzo rozwinęło sekcję.

2. Harmonogram

Program harmonogramu został stworzony tak, aby wcześniejszy brak styczności z mikroprocesorami nie wpływał na ocenę projektu. Na początku zakładał naukę, a następnie w miarę sprawne wykonanie projektu w części manualnej jak i programowej.

2.1. Harmonogram zatwierdzony

Tydzień 1: Nauka teorii.

Tydzień 2: Tworzenie programów testowych.

Tydzień 3: Składanie projektu.

Tydzień 4: Pisanie kodu programu.

Tydzień 5: Optymalizacja kodu.

Tydzień 6: Ostatnie poprawki kodu i systemu..

2.2. Harmonogram wykonany

Tydzień 1: Nauka teorii.

Tydzień 2: Zamawianie części.

Tydzień 3: Oczekiwanie na części i nauka teorii.

Tydzień 4: Tworzenie programów testowych i składanie testowych systemów.

Tydzień 5: Składanie projektu.

Tydzień 6: Pisanie kodu programu

Tydzień 7: Optymalizacja kodu.

Tydzień 8: Ostatnie poprawki kodu i systemu.

Jak można zauważyć brak dostawy komponentów na czas i zamawianie samych elementów zajęło trochę czasu, stąd przesunięcie się wszystkich dalszych kroków w wykonaniu projektu.

3. Kosztorys

LP	Typ	Producent	Ilość	Cena [PLN]	Wartość [PLN]
1	Kable męsko-męskie	Arduino	9	0,93	0,93
2	Czujnik HC-SR04	OpenPlatform	1	7,24	8,17
3	Diody LED	Arduino	2	0,27	8,44
4	Buzzer	OEM	1	1,54	9,98
5	Arduino UNO Rev 3	Arduino	1	96,75	106,73
6	Płytka stykowa	Arduino	1	5,61	112,34

Elementów wykorzystanych zostało w sumie 6.

Godzin na projekt poświęconych zostało około 30.

4. Urządzenie wraz z aplikacją

Projekt ma na celu stworzenie czujnika, który jest systemem mikroprocesorowym złożonym z odpowiednich części, oraz zaprogramowanym tak, aby reagował na obiekty które się do niego zbliżają.

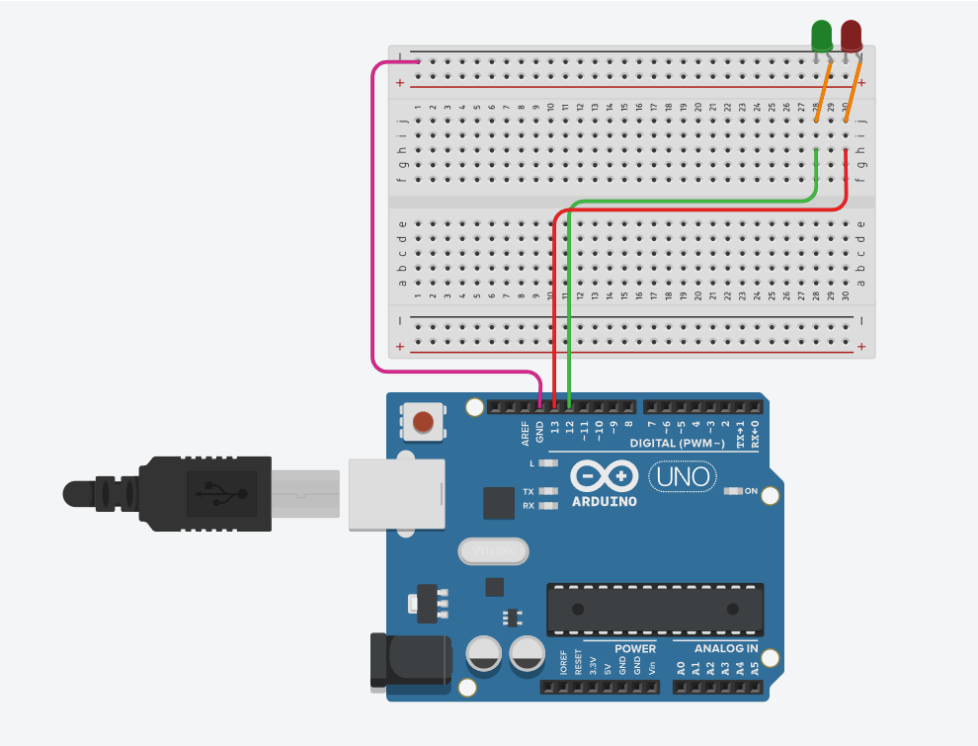
Zamysł na projekt został opisany w rozdziale dotyczącym celu projektu, oraz wstępie. Po skończonym planowaniu czas przyszedł na kupno komponentów. Część z nich została dostarczona do sekcji za pomocą znajomego za darmo. Był to mikroprocesor Arduino UNO, zestaw Kabelków, diody LED, oraz podstawka wykonana z plastiku.

Reszta systemu mikroprocesorowego składającego się na czujnik parkowania została zakupiona poprzez sklep internetowy BOTLAND. Na te elementy składają się: buzzer, oraz czujnik ultradźwiękowy.

Sekcja składająca się na tę grupę projektową nie miała wcześniej styczności z mikroprocesorami, dlatego praca z początku szła ciężko. Dugi czas czekania na otrzymanie elementów ze strony przeznaczono na oglądanie tutoriali. Bardzo pomocne były zagraniczne materiały z zakresu programowania mikroprocesorów Arduino, oraz rzetelne materiały na temat elementów zakupionych przez sekcję.

Pracę rozpoczęto od stopniowego składania wszystkich elementów w jedną spójną całość. Przede wszystkim najpierw zaczęto sprawdzać funkcjonalność komponentów. Były one pojedynczo podpinane do mikroprocesora, dorabiane do nich były krótkie programy obsługujące właśnie te komponenty. Przybrano właśnie taki tryb nauki, aby po sprawdzeniu wszystkich elementów móc łatwo i szybko złożyć je we współgrającą ze sobą całość tworząc finalny produkt.

Pierwszym elementem jaki został testowany były diody LED. Pisane właśnie dla tego komponentu zostały pierwsze programy, które odpowiednio zaświecały diody, bądź je gasiły. Testowane także były komendy dotyczące harmonogramu czasu świecenia diody.



Schemat nr 1: Schemat Testowy mikroprocesora z dwoma diodami LED

```
void setup() {

  pinMode(13,OUTPUT); // RED LED PIN
  pinMode(12,OUTPUT); // GREEN LED PIN
```

Rys. 1: „Fragment kodu testowego”

```
digitalWrite(13,HIGH);
digitalWrite(12,LOW);
```

Rys. 2: „Fragment kodu testowego”

Na rysunkach powyżej można zauważyć fragmenty kodu napisanego na potrzebę nauki programowania i obsługi diod przez mikroprocesor. Czerwona dioda została przypisana do

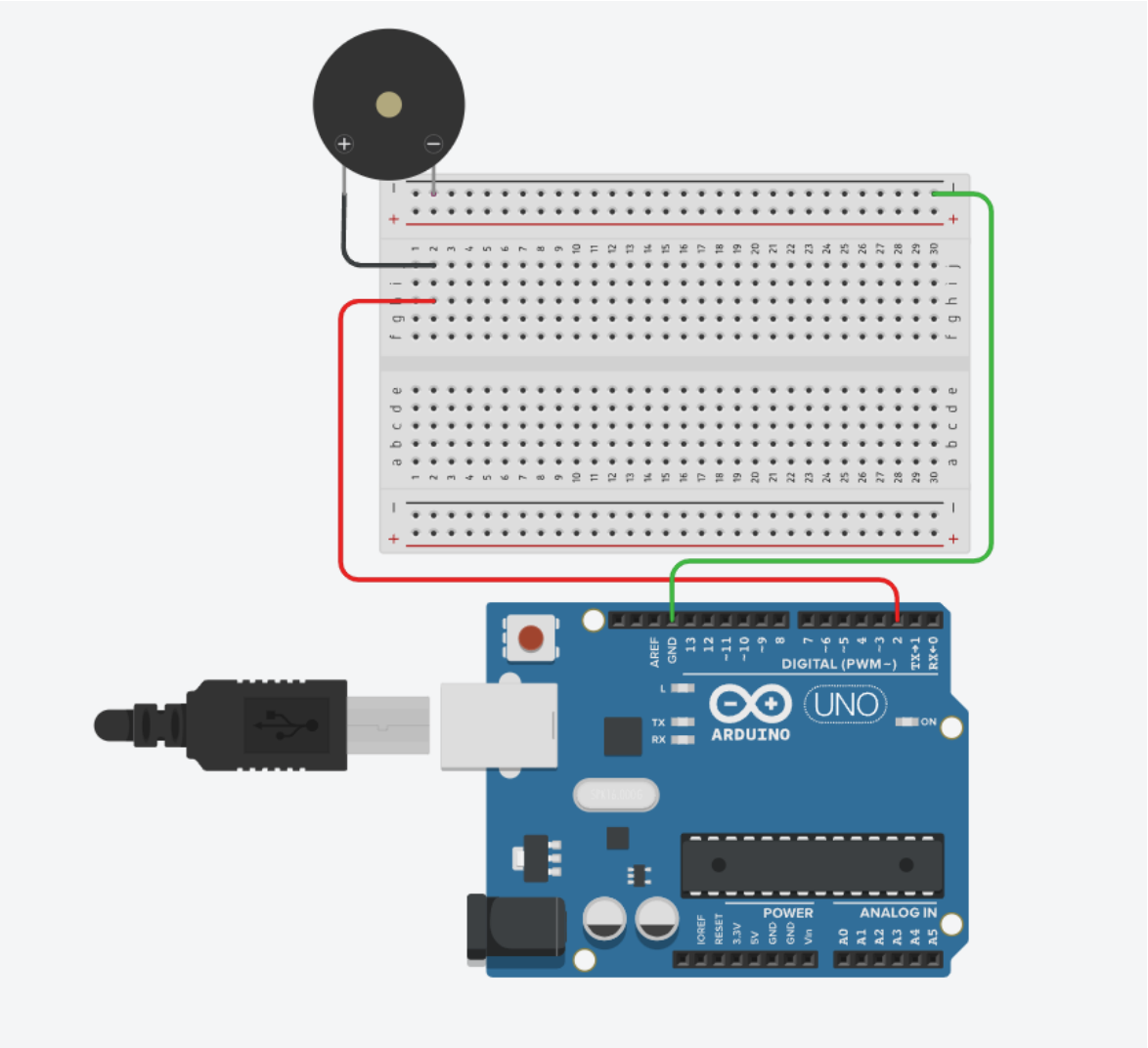
wyjścia z portu numerr 13, natomiast zielona do portu z wyjściem numer 12. Następnie napisany został program, który odpowiednio zaświecał jedną diodę, a drugą gasił. Ponadto po określonym czasie wykonywał tę czynność odwrotnie, co dawało efekt świecenia naprzemiennie. Raz zielona dioda, a raz czerwona.

```
delay(200);
```

Rys. 3: „Fragment kodu testowego”

To właśnie za pomocą komendy delay() udało się zaprogramować mikroprocesor tak, żeby czekał on 200 milisekund, aby dopiero po ustalonym czasie wykonać następną komendę.

Po owocnej nauce w działaniu i programowaniu diod przyszedł czas na naukę obsługi buzzera. Z tym komponentem nie było łatwo, ponieważ bardzo długo nie udawało się odnaleźć oczekiwanego efektu dźwiękowego. Zaczynając od postaw montażu, buzzer został zaaplikowany na płytkę stykową, którą później odpowiednio połączono z mikroprocesorem. Port jaki ustalono, że będzie przysługiwał buzzerowi miał numer 2.



Schemat nr 2: Schemat Testowy mikroprocesora z buzzerem.

```
tone(2,2400);
```

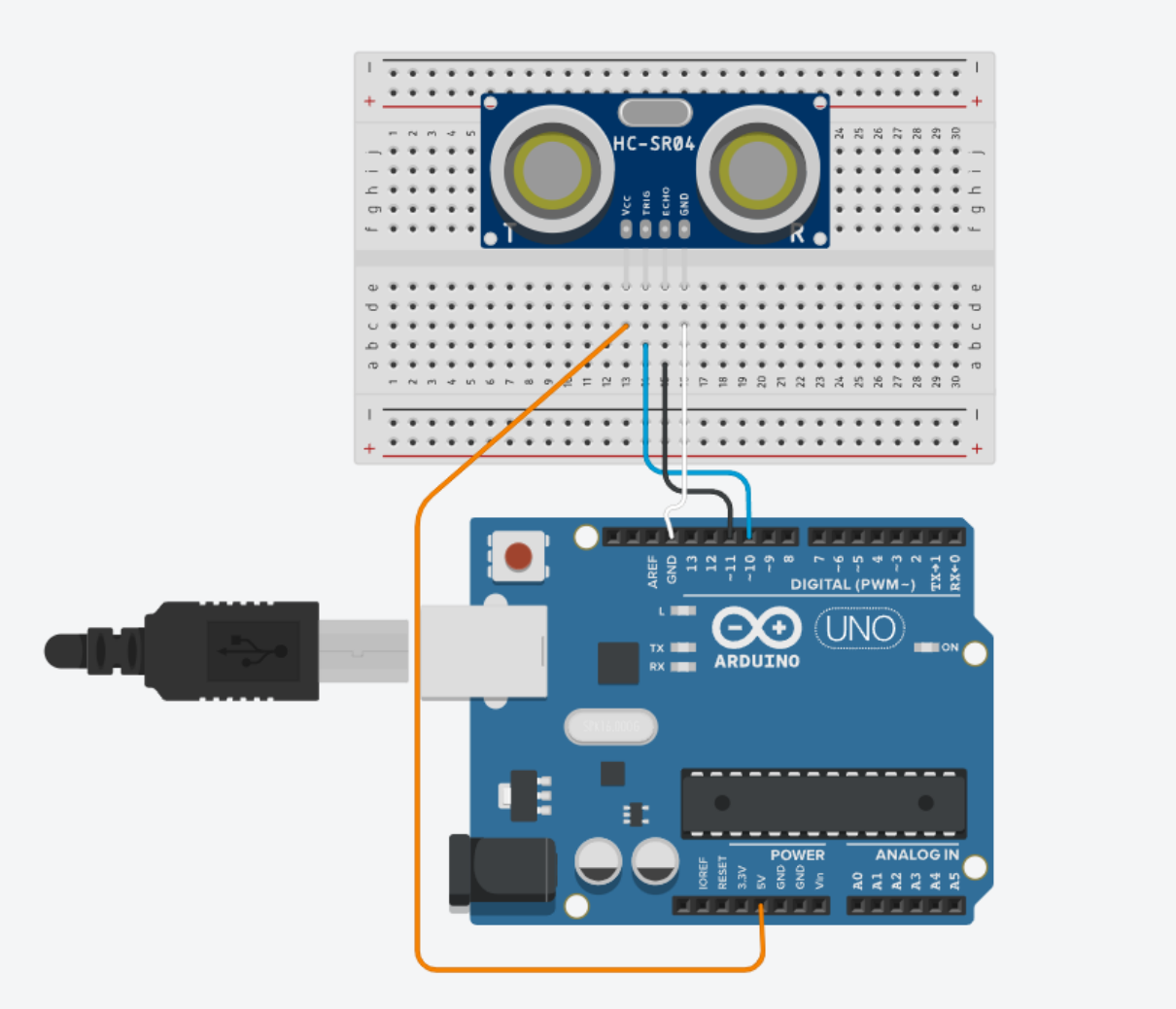
Rys. 4: „Fragment kodu testowego”

```
noTone(2);
```

Rys. 5: „Fragment kodu testowego”

Pierwszą z komend jaką zastosowano i testowano była komenda tone(). Jej pierwszy argument odnosi się do numeru pinu w jaki wpięty został element. Drugi argument odnosi się natomiast do sygnału jaki chcemy uzyskać. Liczba 2400, która widnieje jako jeden z argumentów tej funkcji odnosi się do tego, że mikroprocesor nakazuje buzzerowi wygenerować sygnał o częstotliwości 2400 Hz. Dzięki manipulacji tej wartości można uzyskać różne dźwięki. W projekcie zdecydowano, że najbliższą częstotliwością, która realnie oddaje odgłosy w aucie jest właśnie częstotliwość 2400 Hz, dlatego ustalono, że to właśnie ta będzie częstotliwością docelową.

Następnym elementem, który wymagał testowania był ultradźwiękowy czujnik odległości. To właśnie to urządzenie jest najbardziej istotnym elementem tego projektu. Trzeba było je dobrze poznać, dlatego idealnym pomysłem okazało się testowanie go w praktyce. Jednak żeby do tego doszło najpierw trzeba było zamontować je i podpiąć do mikroprocesora. Kiedy ten etap został już wykonany można było przejść do programowania tego urządzenia. To właśnie ten komponent zabrał najwięcej czasu do zrozumienia i zanalizowania. Bardzo przydatne były fora internetowe na których można było zrozumieć jak to urządzenie działa, oraz jak je zaprogramować. Element ten wysyła sygnał ultradźwiękowy w jednym kierunku, a następnie nasłuchuje kiedy wróci do niego echo. To właśnie na tej podstawie jest się w stanie obliczyć jak daleko stoi przedmiot od którego odbił się sygnał ultradźwiękowy.



Schemat nr 3: Schemat Testowy mikroprocesora ultradźwiękowym czujnikiem odległości.

Powyżej można zobaczyć schemat, który został zrealizowany i to właśnie na nim rozpoczęto naukę obsługi tego elementu. Kod tego programu jest najbardziej skomplikowanym fragmentem całości w dalszych etapach. Poradzono sobie z tym tak, że stworzono zmienne globalne: duration, oraz distance.

```
long duration;
float distance;
```

Rys. 6: „Fragment Programu testowego”

Echo urządzenia zostało przypisane do pinu numer 11, a trigger do pinu numer 10. Są to wyjścia czujnika odległości. Echo odpowiada za nasłuchiwanie, czy sygnał odbity od przeszkody wraca w stronę urządzenia, a trigger odpowiada za wysyłanie sygnału. Odległość można mierzyć na podstawie zmierzonego przez urządzenie czasu. A oto fragment kody obsługującego ten czujnik:

```
digitalWrite(10,LOW);
delayMicroseconds(2);

digitalWrite(10,HIGH);
delayMicroseconds(10);
digitalWrite(10,LOW);

duration = pulseIn(11,HIGH);
distance = duration*0.034/2; // 0.034 [cm/mikrosec]
```

Rys. 7: „Fragment programu testowego”

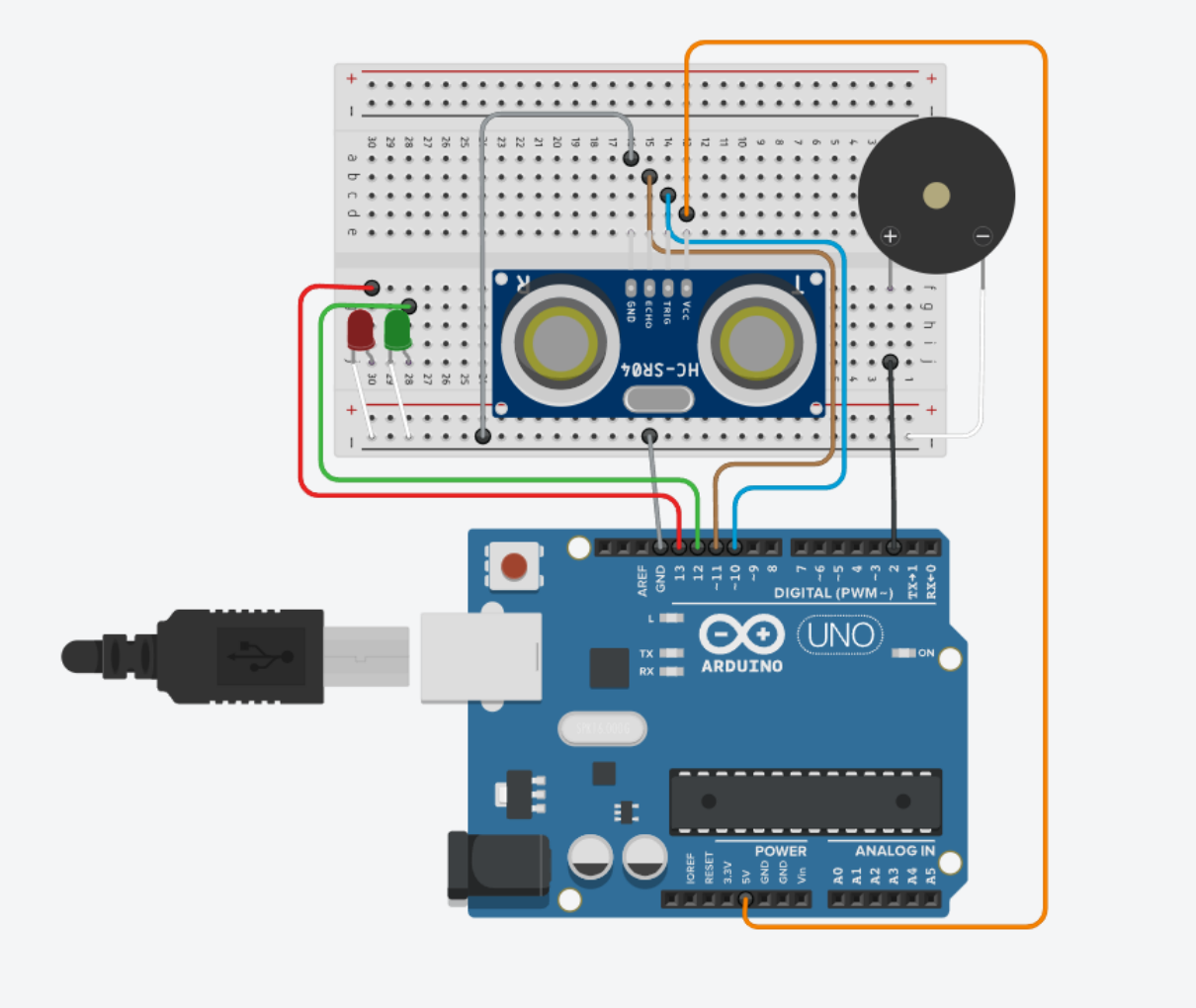
Fragment ten przedstawia kod, który umożliwia przez prosty wzór nadanie zmiennej distance wartości, która odzwierciedla odległość czujnika od obiektu. Następnie wywołuje się konsole w programie arduino i prostą komendą można zobaczyć wynik naszego działania.

```
Serial.print(distance);
Serial.println(" [cm]");
```

Rys. 8: „Fragment programu testowego”

To właśnie ten kawałek kodu pozwala na odczytanie wyników w konsoli.

Kiedy wiedza została już utrwalona i zapisana w postaci testowych programów dla każdego z elementów tego systemu mikroprocesorowego przyszła pora na złożenie całości. Wraz z zestawem „Arduino experimentation kit”, który został dostarczony przez znajomego można było wykorzystać sam mikroprocesor, oraz płytkę stykową. Do tego można było tam odnaleźć także płytkę wykonaną z plastiku, która spaja dwa poprzednie elementy w całość. Jest to stabilne połączenie, które nie tylko zapewnia lepsze aspekty wizualne projektu, ale też pomaga przy prace, okablowanie nie rozjeżdża się.



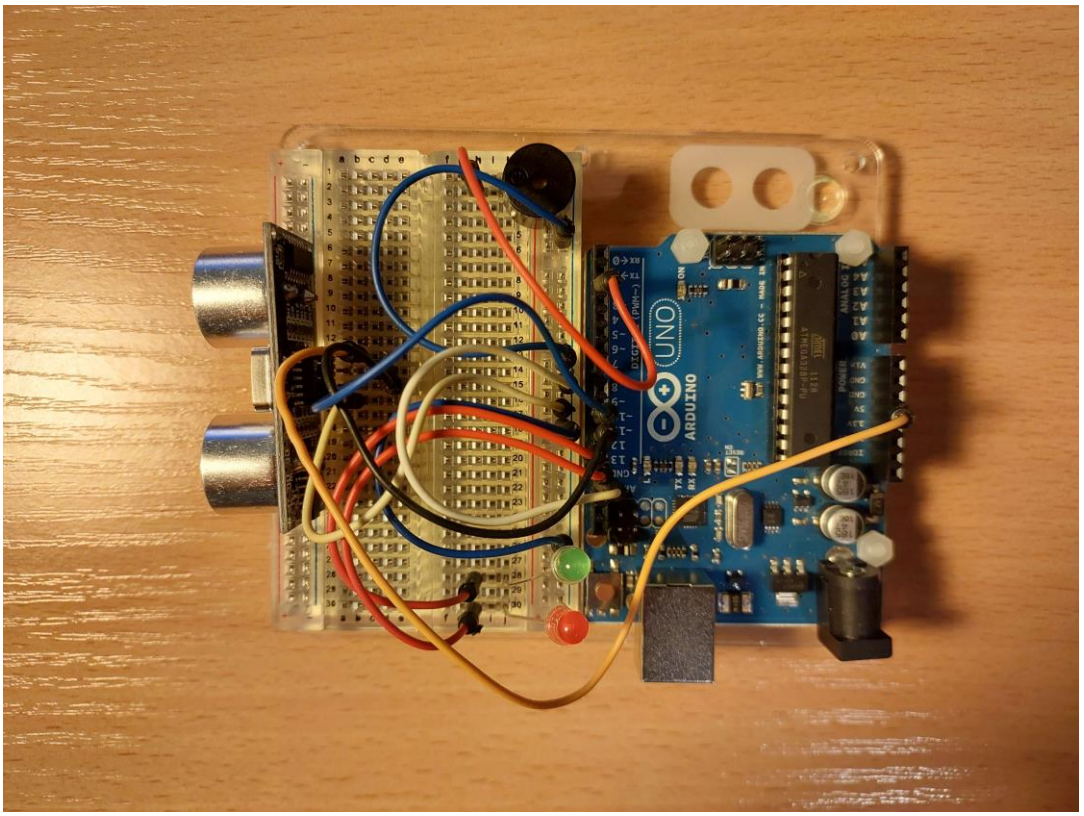
Schemat nr 4: Schemat końcowy projektu pod tytułem: „Czujnik Parkowania”.

Na samym początku zamontowane zostały diody LED. Wybrane kolory to czerwień i zieleń. Na Schemacie nr 4 można zobaczyć jak zostały podpięte. Obie katody zarówno jednego jak i drugiego LED'a zostały podpięte do płytki stykowej w pasmo, które jest uziemione. Natomiast Anody obu LED'ów zostały podpięte do odpowiednich pinów. Są to piny numer 12 dla diody o zabarwieniu zielonym, oraz 13 dla diody o zabarwieniu czerwonym.

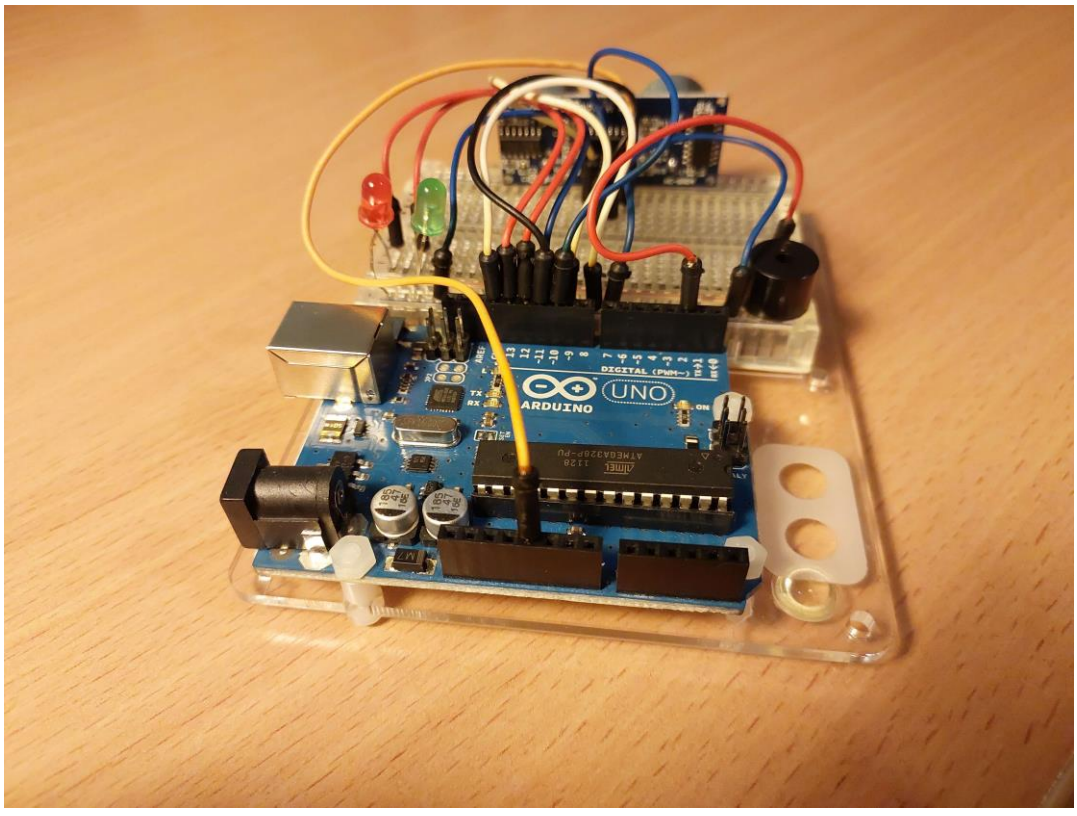
Następnym krokiem jaki obrano było zamontowanie czujnika ultradźwiękowego. To właśnie ten element zakładany był jako następny, ponieważ dzięki diodom LED można upewnić się, że cały projekt działa należycie, oraz efektu wizualne można obserwować niemal od samego początku.

Zasilanie ultradźwiękowego czujnika odległości wpięte zostało w gniazdo 5V na Arduino UNO. Wybór padł na ten pin, a nie ten, który zapewnia 3.3V, ponieważ takie były zalecenia producenta naszego modelu czujnika. Kabel prowadzący od wejścia TRIG, które oznacza trigger, czyli aktywację czujnika podpięta została pod pin z numerem 10. Na schemacie jest to kabel zaznaczony kolorem niebieskim. Kabel który łączy ECHO z pinem, oznaczony kolorem brązowym został poprowadzony do pinu o numerze 11. Natomiast GND, które jest skrótem od anglojęzycznego słowa: „ground", które oznacza uziemienie, został poprowadzony do listewki, która właśnie do uziemienia na Arduino UNO jest poprowadzona. Zostało to oznaczone kablem o kolorze szarym.

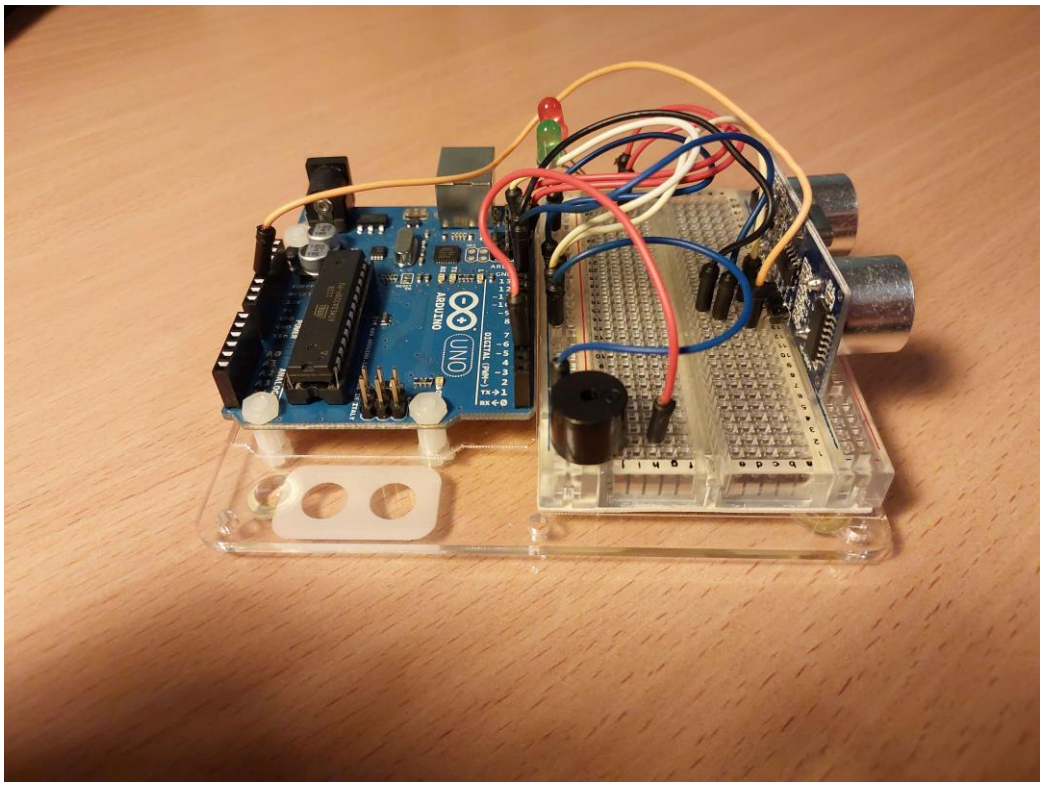
Niestety w rzeczywistości nie udało się osiągnąć tak pięknego układu jak na schematach, aczkolwiek wynik manualnych starań sekcji nie jest niezadowalający. Wszystko wygląda w miarę schludnie, niestety płytka stykowa nie wygląda dokładnie tak jak na schematach, dlatego nie można było podłączyć kabli, które prowadzą do uziemienia na płytce mikroprocesora do jednej listewki. Poradzono sobie z tym inaczej prowadząc je do więszej ilości listewek, natomiast jedna z nich łączy je wszystkie. Ostatecznie tak jak na schematach występuje tylko jedno połączenie między płytką stykową, a uziemieniem na mikroprocesorze.



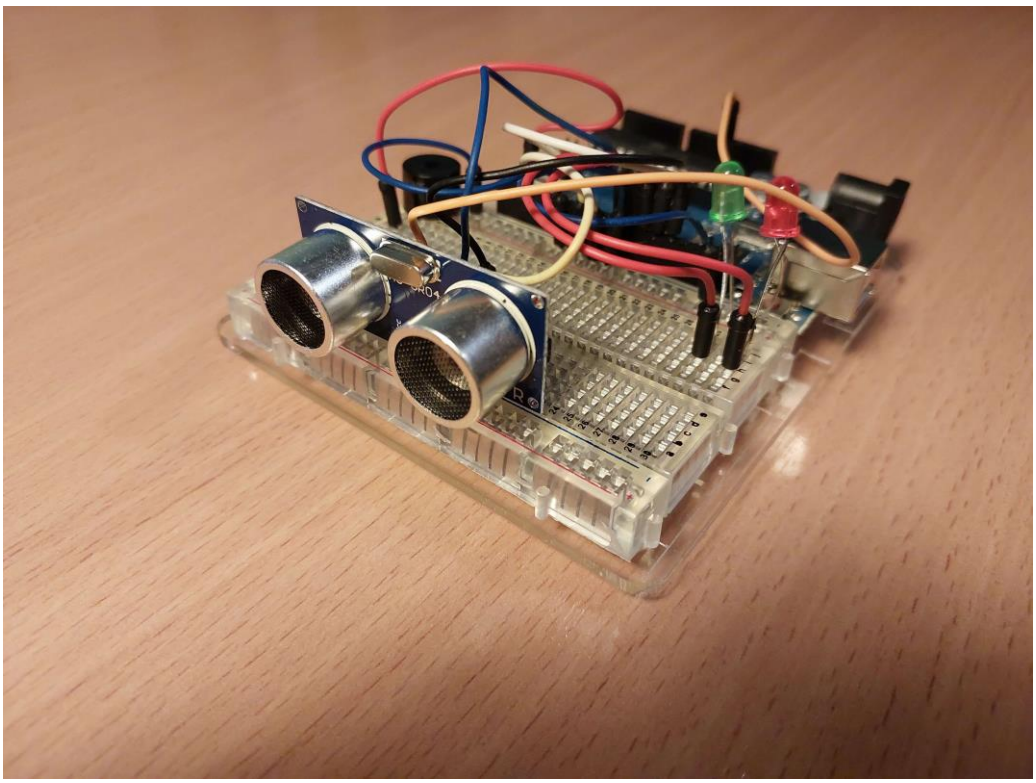
Zdjęcie nr 1: Zdjęcie przedstawiające całkowicie zbudowany projekt od góry.



Zdjęcie nr 2: Zdjęcie przedstawiające całkowicie zbudowany projekt od strony mikroprocesora.



Zdjęcie nr 3: Zdjęcie przedstawiające całkowicie zbudowany projekt od boku.



Zdjęcie nr 4: Zdjęcie przedstawiające całkowicie zbudowany projekt od strony płytki stykowej.

Kiedy projekt został już całkowicie złożony można było przystąpić do napisania programu obsługującego ten układ. Posiadając całą wiedzę, którą nabyto w trakcie testów programowanie było całkiem sprawne. Oczywistym było, że trzeba zaimplementować zmienne globalne do obsługi kodu związanym z częścią działania czujnika ultradźwiękowego. Zmienne globalne to duration i distance :

```
long duration;
float distance;
```

Rys. 8: „Fragment kodu projektu”

Następnym korkiem było uzupełnienie funkcji setup(), w której zaimplementowane i opisane zostały wszystkie piny, które obsługuje czujnik.

```
void setup() {

  pinMode(13,OUTPUT); // RED LED PIN
  pinMode(12,OUTPUT); // GREEN LED PIN

  pinMode(11,INPUT); // ECHO PIN
  pinMode(10,OUTPUT); // TRIGGER PIN

  Serial.begin(9600);
}
```

Rys. 9: „Fragment kodu projektu”

Kolejnym etapem pisania kodu było stworzenie funkcji zapętlającej nasz kod, tak aby procesor przez cały czas wysyłał sygnał aktywujący na czujnik ultradźwiękowy, w wyniku czego odległość sprawdzana będzie co jakiś okres czasu.

```
void loop() {

  digitalWrite(10,LOW);
  delayMicroseconds(2);

  digitalWrite(10,HIGH);
  delayMicroseconds(10);
  digitalWrite(10,LOW);
```

Rys. 10: „Fragment kodu projektu”

Na powyższym rysunku można zobaczyć, jak na samym początku każdego rozpoczęcia pętli czujnik ultradźwiękowy jest „resetowany” tak, aby jego wynik był jak najbardziej dokładny. Napięcie ustalane jest na LOW, co oznacza że w praktyce pin nie wysyła sygnału do urządzenia. To właśnie 2 mikrosekundy później sygnał jest wysyłany do aktywatora (TRIGGER PIN), który wysyła ultradźwięk przez 10 mikrosekund. Kolejnym etapem jest wyłączenie wysyłania sygnału.

```
duration = pulseIn(11,HIGH);
distance = duration*0.034/2; // 0.034 [cm/mikrosec]
```

Rys. 11: „Fragment kodu projektu”

Ten fragment kodu pokazuje, jak dwie zmienne globalne współdziałają miedzy sobą. Komenda pulseIn() to wbudowana funkcja, która mierzy czas napięcia na pinie. W tym przypadku mierzy ona czas napięcia, kiedy pin jest w stanie wysokim. To właśnie ten czas pomnożony przez stałą 0.034/2, która jest prędkością jaką dźwięk porusza się przez 1 cm. To działanie matematyczne pozwala otrzymać wynik, którym jest dystans między czujnikiem a przedmiotem.

```
Serial.print("Distance: ");
if(distance >= 50 || distance <= 4){
  Serial.println("Out of range");
  }
 else {
  Serial.print(distance);
  Serial.println(" [cm]");
  delay(300);
  }

  if(distance <= 17){
  digitalWrite(13,HIGH);
  digitalWrite(12,LOW);
  tone(2,2400);
  delay(200);
  digitalWrite(13,LOW);
  digitalWrite(12,HIGH);
  delay(200);
  noTone(2);
  digitalWrite(12,LOW);
 }
```

Rys. 12: „Fragment kodu projektu”

Ten fragment projektu dotyczy reakcji całego układu w przypadku. Kiedy przedmiot znajduje się w odległości większej od 50 cm, bądź mniejszej od 4 cm konsola programu pozwalającego programować mikroprocesor Arduino pokazuje, że nie ma żadnego obiektu w pobliżu. Jeśli ten warunek nie jest spełniony, czyli obiekt znajduje się w odległości do 50 cm, bądź jest oddalony o przynajmniej 4 cm od czujnika konsola pokazuje w jakiej odległości ten przedmiot się znajduje.

Szczególnym przypadkiem działania tego programu i jego głównym założeniem jest co działanie w przypadku, kiedy przedmiot zbliży się do strefy zagrożenia, która jest określona na 17 cm od czujnika. Wtedy cały system reaguje i zaczyna manipulować napięciem na pinach LED'ów, oraz buzzera. Diody LED świecą naprzemiennie światłem. Raz zielona a raz czerwona. Natomiast buzzer wydaje krótkie dźwięki o częstotliwości 2400 Hz.

5. Podsumowanie

W projekcie prawie wszystkie założenia zostały wykonane. Jedynym elementem, który został usunięty w trakcie rozmyślania nad projektem jest wyświetlacz, który byłby po prostu zbędny. Cała reszta projektu całkowicie odzwierciedla pierwotny plan. Nabyta wiedza z pewnością będzie przydatna w przyszłości. Program działa tak jak należy, pokazuje na konsoli odległość od czujnika do obiektu, poprawnie działa w przypadku zbliżenia się obiektu do czujnika na odległość zagrożenia, która oznacza bardzo bliską odległość obiektu od czujnika.

Literatura

1. https://www.youtube.com/watch?v=XZlP9BASbH4 – 10.11.22 r.
2. https://www.youtube.com/watch?v=tjj_a9PRKqM – 10.11.22 r.
3. https://www.youtube.com/watch?v=IKD5Jhyg_cc&t=243s – 10.11.22 r.
4. https://www.googleadservices.com/pagead/aclk?sa=L&ai=DChcSEwi479m0o9v8AhUJcBgKHYYsCIIYABARGgJsZQ&ohost=www.google.com&cid=CAESa-D2IaKC5NPotiMkZNGOjmSIOtwf6ucfHAbYA_zZRD5HkYoQZ8jPxorKj9MyX8UhApYP1ix4QHHEvmH-zxTYbLuvRTF1pdpB1e_y4XX17tF6DXxbFAYxAZX0S-2juxJnrDlmFyPKC-f9RFzJ&sig=AOD64_11D3HIw0YvEQRxmLthcbC44U5YKg&q&adurl&ved=2ahUKEwjQ89O0o9v8AhWHgSoKHdyJBtUQ0Qx6BAgIEAE – 01.01.23r.