AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Automatyka i Robotyka

Informatyka w sterowaniu i zarządzaniu

Informatyka czasu rzeczywistego

Samochodowy czujnik parkowania

L.p.	**Członek**	**Numer albumu**	**Adres e-mail**
1	Artur Mzyk	400658	arturmzyk@student.agh.edu.pl
2	Kamil Pieprzycki	402037	pieprzycki@student.agh.edu.pl

Spis treści

1. Specyfikacja problemu

W obecnych czasach czujnik parkowania jest dostępny w każdym samochodzie. Pozwala on na bezpieczne parkowanie oraz zapobiega kolizjom i zderzeniom z innymi obiektami, ostrzegając kierowcę, gdy zbliża się niebezpiecznie blisko do przeszkody, takiej jak ściana bądź inny samochód.

Stworzony w ramach tego projektu czujnik parkowania składa się z następujących modułów:

- moduł ultradźwiękowych sensorów odległości,
- moduł wizualizacji.

Projekt został wykonany w symulatorze WOKWI na płytce Arduino UNO.

1.1. Moduł ultradźwiękowych sensorów odległości

Ultradźwiękowy sensor odległości jest zasilany napięciem. Jest wykorzystywany do mierzenia odległości od obiektu za pomocą fal dźwiękowych. Wysyła sygnał TRIGGER w postaci impulsów. Następnie odbiera sygnał zwrotny ECHO i mierzy jego długość.

Fale dźwiękowe mają stałą prędkość $v = 343\,\frac{m}{s}$. Droga przebyta przez falę przedstawia się zatem następującym wzorem:

$$s = t \cdot v,$$

gdzie:

$s\ [m]$ – droga przebyta przez falę,

$t\ [s]$ – zmierzony czas,

$v\ \left[\frac{m}{s}\right]$ – prędkość dźwięku.

Należy jednak pamiętać, że fala przebywa drogę równą dwukrotności odległości - z racji tego, że leci w jedną stronę, a potem wraca.

Czasem takie sensory są zamontowane tylko z tyłu bądź tylko z przodu pojazdu. W samochodach nowszej generacji obejmują one jednak niemalże cały obszar wokół pojazdu. W projekcie wykorzystano trzy sensory ulokowane z tyłu samochodu. Jest to podyktowane ograniczoną liczbą pinów na płytce.

1.2. Moduł wizualizacji

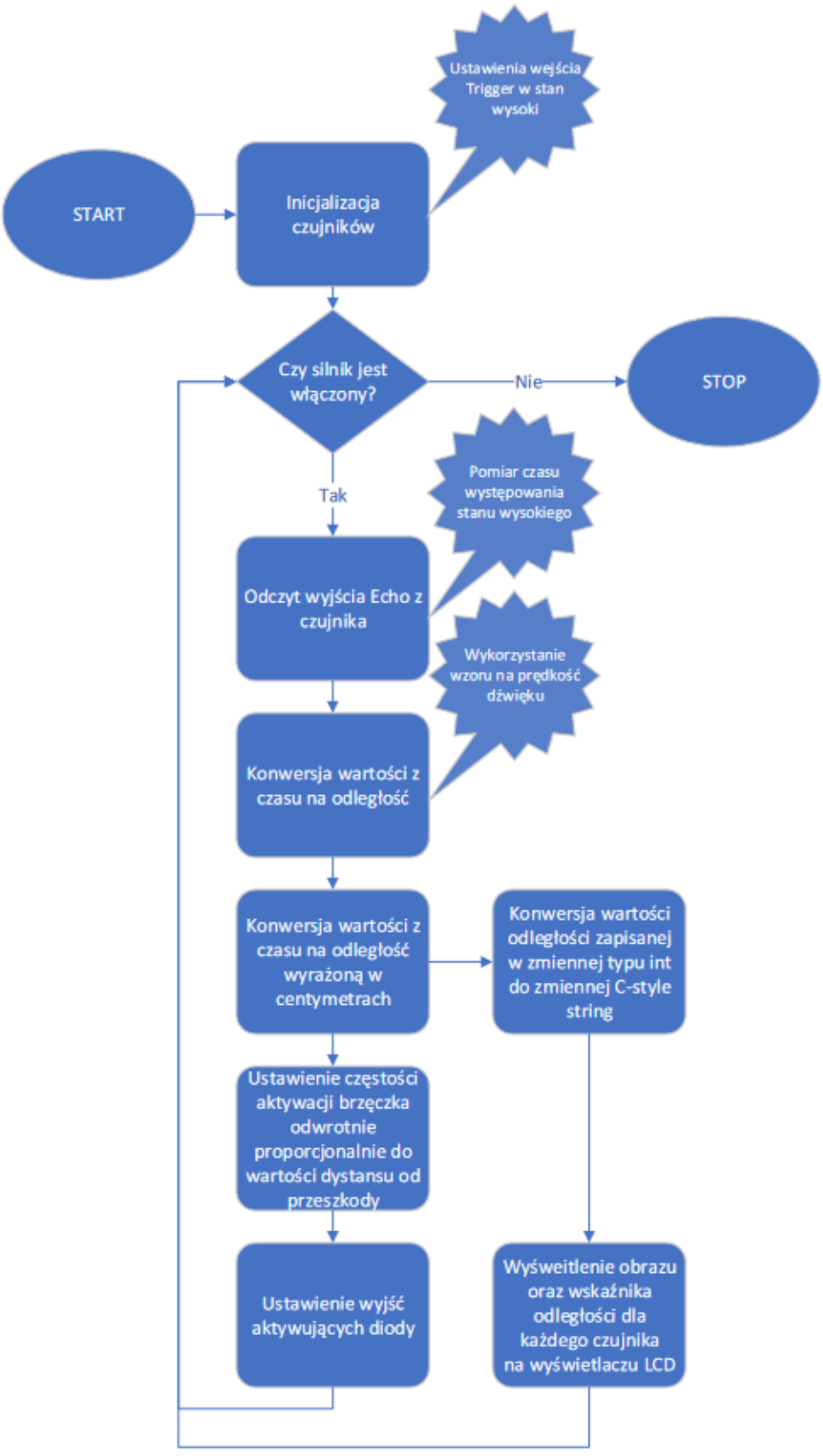
Wizualizacja realizowana jest na kilku płaszczyznach, co pozwala na lepsze oddziaływanie na zmysły kierowcy i zachowanie przez niego większej czujności oraz skrócenie czasu reakcji:

- **Dźwiękowa.** Obecny w układzie brzęczek (ang. *buzzer*) generuje sygnały dźwiękowe, których częstotliwość nasila się wraz ze zbliżaniem się do obiektu. Częstotliwość generowania sygnałów jest odwrotnie proporcjonalna do zmierzonej odległości. Gdy obiekt jest na tyle daleko od czujnika, że nie stwarza zagrożenia, to nie jest generowany żaden sygnał, aby nie wprowadzać kierowcy w zakłopotanie.

- **Świetlna.** Diody tworzą tak zwany pasek postępu (ang. *progress* bar). Określa on, jak blisko obiektu znajduje się pojazd. Zdeterminowany jest przez dwie wartości graniczne LED_WARNING i LED_TH, wyrażone w centymetrach. Gdy odległość zmierzona przez któryś z sensorów jest mniejsza od wartości LED_TH, to zapala się czerwona dioda. W przeciwnym razie jeśli odległość zmierzona przez któryś z sensorów jest mniejsza od wartości LED_WARNING, to zapala się żółta dioda. W pozostałych przypadkach świeci dioda zielona, co symbolizuje bezpieczną odległość. Najlepszym rozwiązaniem byłoby zastosowanie osobnego paska postępu dla każdego sensora, jednak niezbyt pozwala na to ograniczona liczba pinów. W przypadku brzęczka nie byłoby to możliwe, gdyż ciężko byłoby człowiekowi rozróżnić kilka sygnałów dźwiękowych generowanych równocześnie.
- **Obrazkowa.** Na ekranie wyświetlacza ukazuje się obraz symbolizujący samochód i poszczególne sensory. Pod każdym z sensorów widnieje odległość przez niego wykryta, wyrażona w centymetrach.

2. Schemat blokowy oprogramowania

Na *Rys. 1.* zamieszczony został schemat blokowy oprogramowania.



Rys. 1. Schemat blokowy oprogramowania

3. Kod programu

3.1. Wykorzystane biblioteki

Wykorzystano jedynie bibliotekę *Adafruit SSD1306*, która pozwala na programowanie wyświetlacza LED SDD1306.

3.2. Definicje stałych i obiektów

Zdefiniowano zbiór stałych, które:

- określają przeznaczenie poszczególnych pinów,
- określają liczbę sensorów ultradźwiękowych,
- definiują prędkość dźwięku,
- określają liczbę pulsów sygnału odbieranego przez sensor ultradźwiękowy,
- określają częstotliwość i czas trwania sygnału brzęczka,
- definiują progowe wartości odległości.

Kod wdrażający tę część został przedstawiony na *Rys. 2.*

```
// Pins
#define ECHO_PIN1 2
#define TRIG_PIN1 3
#define ECHO_PIN2 4
#define TRIG_PIN2 5
#define ECHO_PIN3 6
#define TRIG_PIN3 7
#define BUZZ_PIN 8
#define RED_LED_PIN 9
#define YELLOW_LED_PIN 10
#define GREEN_LED_PIN 11

// Number of sensors
#define N_SENSORS 3

// Speed of sound [m / s]
#define SOUND_SPEED 343

// Number of pulses for one signal
#define N_PULSES 10

// Buzzer's sound signal frequency [Hz], duration [ms]
and distance threshold [cm]
#define BUZZ_FREQ 262
#define BUZZ_DUR 200
#define BUZZ_DIST 100

// LED display reset pin
#define OLED_RESET 4

// LED diodes threshold [cm]
#define LED_WARNING 100
#define LED_TH 15
```

Rys. 2. Zbiór stałych

Skonfigurowano wyświetlacz LED SSD1306, co zostało ukazane na *Rys. 3.*

```
// Screen setup
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_DEV_0 |
                          U8G_I2C_OPT_NO_ACK | U8G_I2C_OPT_FAST);
const unsigned char bitmap_car[] PROGMEM = {
        0x00, ..., 0x00
};
const unsigned char bitmap_unit_cm [] PROGMEM = {
        0xf0, ..., 0xc0
};

const unsigned char bitmap_sound_on [] PROGMEM = {
        0x03, ..., 0x0f
};
```

Rys. 3. Ustawienia wyświetlacza

Stworzono tablicę obiektów sensorów i zmienne pomocnicze dla brzęczka, co zostało ukazane na *Rys. 4.*

```
// Buffer to store digits to be converted to CString
char buffer[10];

// Structure for the ultrasonic sensor data
struct Sensor {
  int echo_pin; // ECHO pin
  int trig_pin; // TRIG pin
  float dist; // measured distance [cm]
  int label_xpos; // x position on the screen
  int label_ypos;  // y position on the screen
  int label_width;
};

// Sensors table
struct Sensor sensors[N_SENSORS];

// Buzzer variables
bool buzzerOn = false;
unsigned long previousBuzzTime = 0;
unsigned long buzzInterval = 0;

// OLED display object connected to I2C
Adafruit_SSD1306 display(OLED_RESET);
```

Rys. 4. Stworzenie tablicy obiektów sensorów i zmiennych pomocniczych dla brzęczka

3.3. Konfiguracja płytki

Na *Rys. 5.* przedstawiono konfigurację płytki. Przypisano odpowiednie piny do pinów sensora i ustawiono je odpowiednio jako wejścia bądź wejścia. Ponadto otwarto komunikację z wyświetlaczem za pomocą interfejsu I^2C.

```
void setup() {
  u8g.setFont(u8g_font_tpssb);
  u8g.setColorIndex(1);

  // Initialize display by providing the display type and its I2C address
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  // Assign pins to sensors
  sensors[0].echo_pin = ECHO_PIN1;
  sensors[0].trig_pin = TRIG_PIN1;
  sensors[1].echo_pin = ECHO_PIN2;
  sensors[1].trig_pin = TRIG_PIN2;
  sensors[2].echo_pin = ECHO_PIN3;
  sensors[2].trig_pin = TRIG_PIN3;

  // Choose inputs and outputs
  pinMode(sensors[0].trig_pin, OUTPUT);
  pinMode(sensors[0].echo_pin, INPUT);
  pinMode(sensors[1].trig_pin, OUTPUT);
  pinMode(sensors[1].echo_pin, INPUT);
  pinMode(sensors[2].trig_pin, OUTPUT);
  pinMode(sensors[2].echo_pin, INPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(YELLOW_LED_PIN, OUTPUT);
  pinMode(RED_LED_PIN, OUTPUT);

  sensors[0].label_xpos = 100;
  sensors[0].label_ypos = 60;
  sensors[1].label_xpos = 55;
  sensors[1].label_ypos = 60;
  sensors[2].label_xpos = 10;
  sensors[2].label_ypos = 60;

  // Initialize serial port
  Serial.begin(9600);
}
```

Rys. 5. Konfiguracja płytki Arduino UNO

3.4. Odczyt odległości

Odczytu odległości dokonuje się w następujący sposób:

1. Wysłanie sygnału TRIGGER (na 10 sekund).
2. Odczyt pulsu.
3. Wykorzystanie wzoru na drogę i prędkości dźwięku.

Kod wdrażający to postępowanie został przedstawiony na *Rys. 6.*

```
void loop() {

    /*
    OTHER FUNCTIONALITY CODE
    */

    // For each sensor
    for (int i = 0; i < N_SENSORS; i++) {

        // Send the trigger signal
        digitalWrite(sensors[i].trig_pin, HIGH);
        delayMicroseconds(10);
        digitalWrite(sensors[i].trig_pin, LOW);

        // Measure the time for the pulse on a defined pin
        // float pulse_duration = 0.000001 * pulseIn(sensors[i].echo_pin,
                                                     HIGH) / N_PULSES;
        // sensors[i].dist = pulse_duration * SOUND_SPEED / 2.0;
        sensors[i].dist = pulseIn(sensors[i].echo_pin, HIGH);
        sensors[i].dist = round(sensors[i].dist * 0.0343/2.0);
        itoa(sensors[i].dist, buffer, 10);

        /*
        OTHER FUNCTIONALITY CODE
        */

    }
}
```

Rys. 6. Odczyt odległości z sensora ultradźwiękowego

3.5. Oprogramowanie wyświetlacza

Na *Rys. 6.* został przedstawiony kod wyświetlacza tekstowego. Wyświetla on w centymetrach odległości odczytane z sensorów ultradźwiękowych.

```
void loop() {

    /*
    OTHER FUNCTIONALITY CODE
    */

    // For each sensor
    for (int i = 0; i < N_SENSORS; i++) {

        /*
        OTHER FUNCTIONALITY CODE
        */

        // Display the measured distance
        u8g.firstPage();
        do {

            // Car indicator
            u8g.drawBitmapP(0, 0, 128/8, 64, bitmap_car);

            // cm unit indicator
            u8g.drawBitmapP(0, 0, 24/8, 10, bitmap_unit_cm);

            // Sound on indicator
            u8g.drawBitmapP(104, 0, 24/8, 10, bitmap_sound_on);

            // For all the sensors
            for (int i=0; i<N_SENSORS; i++) {

                // Convert integer to C-style string
                itoa(sensors[i].dist, buffer, 10);

                // Draw the distance label
                u8g.drawStr(sensors[i].label_xpos , sensors[i].label_ypos ,
                            buffer);
            }
        }
        while (u8g.nextPage());


    /*
    OTHER FUNCTIONALITY CODE
    */

    }
}
```

Rys. 7. Kod wyświetlacza

3.6. Oprogramowanie brzęczka

Na *Rys. 7.* został przedstawiony kod generujący sygnał brzęczka. Wykorzystano funkcję *tone()*, która przyjmuje jako parametry częstotliwość i czas trwania sygnału dźwiękowego.

```
void loop() {

    /*
    OTHER FUNCTIONALITY CODE
    */

    // Update the buzzer state based on the closest distance
    float closestDistance = 1000.0; // Initialize with a large value

    for (int i = 0; i < N_SENSORS; i++) {
        if (sensors[i].dist < closestDistance) {
            closestDistance = sensors[i].dist;
        }
    }

    // Calculate the buzz interval inversely proportional to the
       closest distance
    buzzInterval = map((int)(closestDistance * 100), 0,
                       (int)(BUZZ_DIST * 200), 0, 2000);

    // Check if it is to turn on the buzzer
    buzzerOn = closestDistance < BUZZ_DIST || closestDistance < 5;

    // Play the buzzer sound if it's time for a new beep
    unsigned long currentMillis = millis();
        if(buzzerOn && currentMillis - previousBuzzTime >= buzzInterval) {
        previousBuzzTime = currentMillis;
        tone(BUZZ_PIN, BUZZ_FREQ, BUZZ_DUR);
    } else {
        noTone(BUZZ_PIN);
    }

    /*
    OTHER FUNCTIONALITY CODE
    */
}
```

Rys. 8. Kod generujący sygnał dźwiękowy

3.7. Oprogramowanie diod

Na *Rys. 8.* Został przedstawiony kod zapalający odpowiednie diody w zależności od odległości. Wykonano to w sposób oddziałujący na ludzkie zmysły, czyli czerwona dioda LED oznacza bardzo małą odległość od przeszkody.

```
void loop() {

    // Check which diode to turn on
    int diode_on = 0;

    /* OTHER FUNCTIONALITY CODE */

    // Check which diode to turn on
    if (sensors[i].dist < LED_WARNING && diode_on < 2) {
        diode_on = 1;
    }

    if (sensors[i].dist < LED_TH) {
        diode_on = 2;
    }

    /* OTHER FUNCTIONALITY CODE */

    // Set the LED diodes based on the closest distance
    switch (diode_on) {
        case 0:
            digitalWrite(GREEN_LED_PIN, HIGH);
            digitalWrite(YELLOW_LED_PIN, LOW);
            digitalWrite(RED_LED_PIN, LOW);
            break;

        case 1:
            digitalWrite(GREEN_LED_PIN, LOW);
            digitalWrite(YELLOW_LED_PIN, HIGH);
            digitalWrite(RED_LED_PIN, LOW);
            break;

        case 2:
            digitalWrite(GREEN_LED_PIN, LOW);
            digitalWrite(YELLOW_LED_PIN, LOW);
            digitalWrite(RED_LED_PIN, HIGH);
            break;

        default:
            digitalWrite(GREEN_LED_PIN, LOW);
            digitalWrite(YELLOW_LED_PIN, LOW);
            digitalWrite(RED_LED_PIN, LOW);
            break;
    }
}
```

Rys. 9. Kod zapalający diody LED

3.8. Wnioski dotyczące oprogramowania

Wszystkie funkcjonalności wizualizacyjne są wykonywane cyklicznie – z wykorzystaniem funkcji *loop()*. Iteracje wywoływane są co 200 ms.

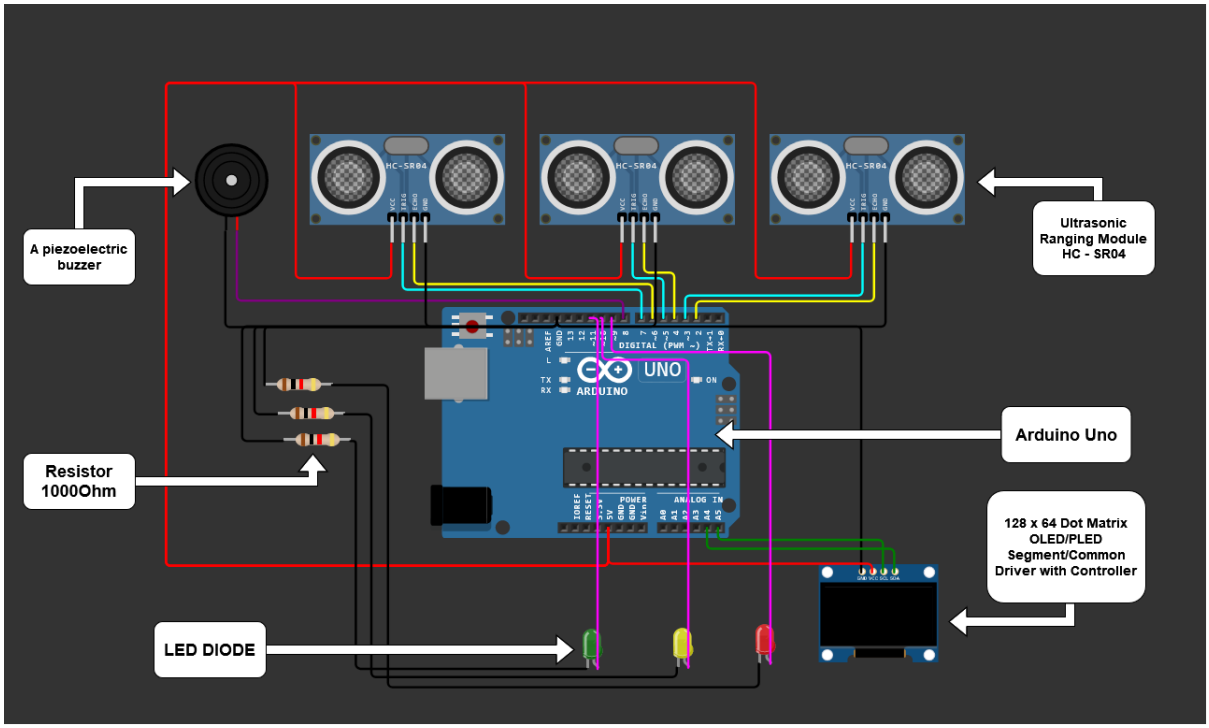
4. Zbiór definiujący połączenia na płytce

4.1. Plik JSON

```
"connections": [
    ["ultrasonic1:ECHO", "uno:2", "yellow", ["v50", "h33"]],
    ["ultrasonic1:TRIG", "uno:3", "Cyan", ["v45", "h33"]],
    ["ultrasonic2:ECHO", "uno:4", "yellow", ["v25", "h33"]],
    ["ultrasonic2:TRIG", "uno:5", "Cyan", ["v30", "h33"]],
    ["ultrasonic3:ECHO", "uno:6", "yellow", ["v35", "h33"]],
    ["ultrasonic3:TRIG", "uno:7", "Cyan", ["v40", "h33"]],
    ["buzzer:2", "uno:8", "Purple", ["v60", "h33"]],
    ["ultrasonic1:GND", "uno:GND.1", "black", ["v72", "h33"]],
    ["ultrasonic2:GND", "uno:GND.1", "black", ["v72", "h33"]],
    ["ultrasonic3:GND", "uno:GND.1", "black", ["v72", "h33"]],
    ["buzzer:1", "uno:GND.1", "black", ["v83", "h33"]],
    ["ultrasonic1:VCC", "uno:5V", "red", ["v30", "h-83", "v-170", "h-513",
                                          "v420", "h385"]],
    ["ultrasonic2:VCC", "uno:5V", "red", ["v30", "h-83", "v-170", "h-313",
                                          "v420", "h385"]],
    ["ultrasonic3:VCC", "uno:5V", "red", ["v30", "h-83", "v-170", "h-113",
                                          "v420", "h385"]],
    ["ssd1306:GND", "uno:GND.1", "black", ["v0"]],
    ["ssd1306:VCC", "uno:5V", "red", ["v0"]],
    ["ssd1306:SDA", "uno:A4", "green", ["v0"]],
    ["ssd1306:SCL", "uno:A5", "green", ["v0"]],
    ["led3:C", "r3:2", "green", ["v0"]],
    ["r3:1", "uno:GND.1", "green", ["h-52.6", "v-224.89", "h303.75"]],
    ["r2:1", "uno:GND.1", "green", ["v-119.68", "h243.78"]],
    ["r1:1", "uno:GND.1", "green", ["v0"]],
    ["r1:2", "led1:C", "green", ["v0" ]],
    ["r2:2", "led2:C", "green", ["v0" ]],
    ["led1:A", "uno:9", "green", ["v0" ]],
    ["led2:A", "uno:10", "green", ["v0" ]],
    ["led3:A", "uno:11", "green", ["v0" ]]
]
```

Rys. 10. Połączenia na płytce w pliku JSON

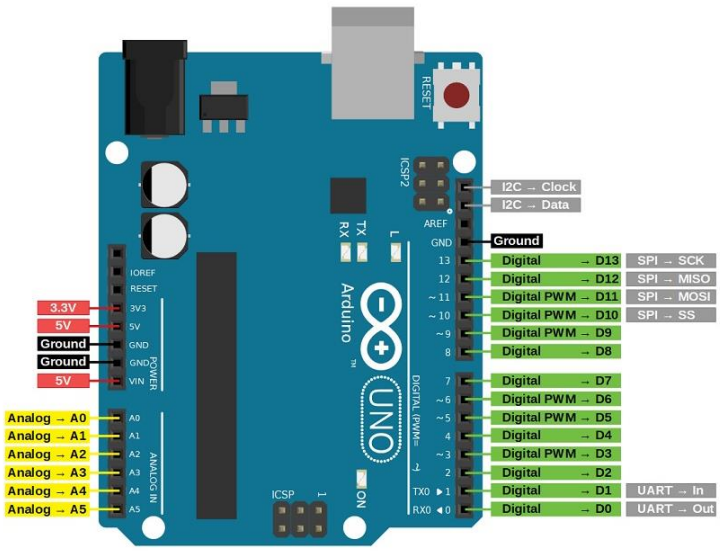
4.2. Schemat połączeń



4.3. Zbiór dokumentacji wykorzystanych podzespołów

4.3.1. **Arduino uno** – jest to platforma prototypowania oparta na mikrokontrolerze ATmega328P. Jest wyposażona w 14 cyfrowych pinów wejścia/wyjścia, w tym 6 z nich obsługuje modulację szerokości impulsu (PWM). Płytka oferuje także 6 wejść analogowych oraz interfejsy komunikacyjne, takie jak UART, SPI i I2C, umożliwiające łączność z innymi urządzeniami elektronicznymi.

Schemat pinów został przedstawiony na ilustracji



4.3.2. **HC – SR04 -** czujnik ultradźwiękowy, który wykorzystuje fale dźwiękowe o częstotliwości ultradźwiękowej do pomiaru odległości. Generuje krótki sygnał ultradźwiękowy i odbiera jego echa od przeszkody, a następnie oblicza odległość na podstawie czasu, jaki upłynął między wysłaniem sygnału a odbiorem echa.

Specyfikacja	Wartość
Zakres pomiaru	2 – 400 [cm]
Dokładność pomiaru	3 [mm]
Napięcie zasilania	5 – 12 [V]
Interfejs komunikacyjny	TRIGGER, ECHO
Obszar wiązki ultradzwiękowej	Kąt rozsyłu ~15 stopni

4.3.3. **SDD1306 -** wyświetlacz OLED (organiczne diody elektroluminescencyjne), który emituje światło samodzielnie, co umożliwia wysoki kontrast i niskie zużycie energii.

Specyfikacja	Wartość
Rozdzielczość	128x64 piksele
Interfejs komunikacyjny	I2C (TWI) lub SPI
Obsługiwane funkcje	Wyświetlanie tekstu, rysowanie figur, wyświetlanie obrazów bitmapowych
Napięcie zasilania	3 V - 5 V

4.4. Opis połączeń:

Obwód składa się z płytki Arduino Uno, buzzera, wyświetlacza SSD1306, trzech czujników ultradźwiękowych (ultrasonic1, ultrasonic2, ultrasonic3), trzech rezystorów (r1, r2, r3) oraz trzech diod LED (led1, led2, led3).

1. Połączenia z płytką Arduino Uno:
    - Pin ECHO czujnika ultradźwiękowego ultrasonic1 jest połączony z pinem cyfrowym 2 (uno:2).
    - Pin TRIG czujnika ultradźwiękowego ultrasonic1 jest połączony z pinem cyfrowym 3 (uno:3).
    - Pin ECHO czujnika ultradźwiękowego ultrasonic2 jest połączony z pinem cyfrowym 4 (uno:4).
    - Pin TRIG czujnika ultradźwiękowego ultrasonic2 jest połączony z pinem cyfrowym 5 (uno:5).
    - Pin ECHO czujnika ultradźwiękowego ultrasonic3 jest połączony z pinem cyfrowym 6 (uno:6).

- Pin TRIG czujnika ultradźwiękowego ultrasonic3 jest połączony z pinem cyfrowym 7 (uno:7).
- Pozytywny biegun (VCC) buzzera jest połączony z pinem cyfrowym 8 (uno:8).
- Wszystkie piny GND czujników ultradźwiękowych są połączone z masą Arduino (uno:GND.1).
- Ujemny biegun (GND) buzzera jest połączony z masą Arduino (uno:GND.1).

2. Połączenia zasilania i masy do płytki Arduino Uno:
   - Wszystkie piny VCC czujników ultradźwiękowych są połączone z pinem 5V Arduino (uno:5V).
   - Pin GND wyświetlacza SSD1306 jest połączony z masą Arduino (uno:GND.1).
   - Pin VCC wyświetlacza SSD1306 jest połączony z pinem 5V Arduino (uno:5V).
3. Połączenia do wyświetlacza SSD1306:
   - Pin SDA wyświetlacza SSD1306 jest połączony z pinem A4 Arduino (uno:A4).
   - Pin SCL wyświetlacza SSD1306 jest połączony z pinem A5 Arduino (uno:A5).
4. Połączenia między rezystorami a diodami LED:
   - Dioda LED led1 jest połączona z rezystorem r1. Pin 2 rezystora jest połączony z katodą diody LED (led1:C).
   - Dioda LED led2 jest połączona z rezystorem r2. Pin 2 rezystora jest połączony z katodą diody LED (led2:C).
   - Dioda LED led3 jest połączona z rezystorem r3

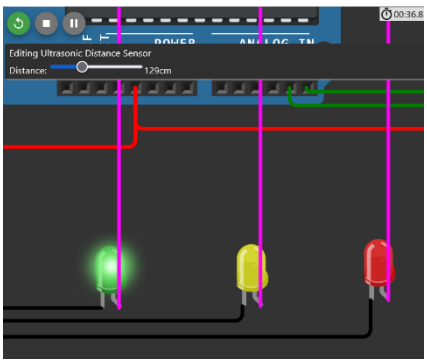
5. Przeprowadzone testy oprogramowania wraz z zrzutami ekranu

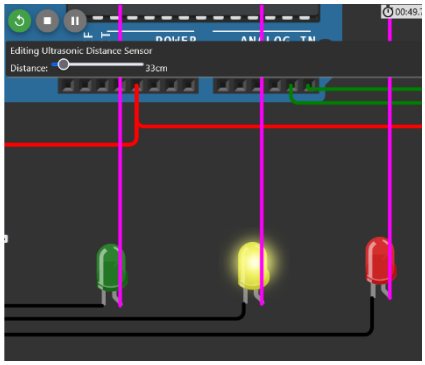
5.1. Diody

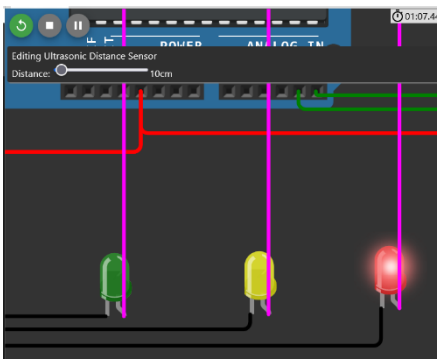
Przeprowadzono test dla wartości znajdujących się w przedziałach aktywacji diód:

- Zakres bezpieczeństwa: $(100; 400)$
- Zakres ostrzeżenia $(15; 100\rangle$
- Zakres niebezpieczeństwa $(2; 15\rangle$

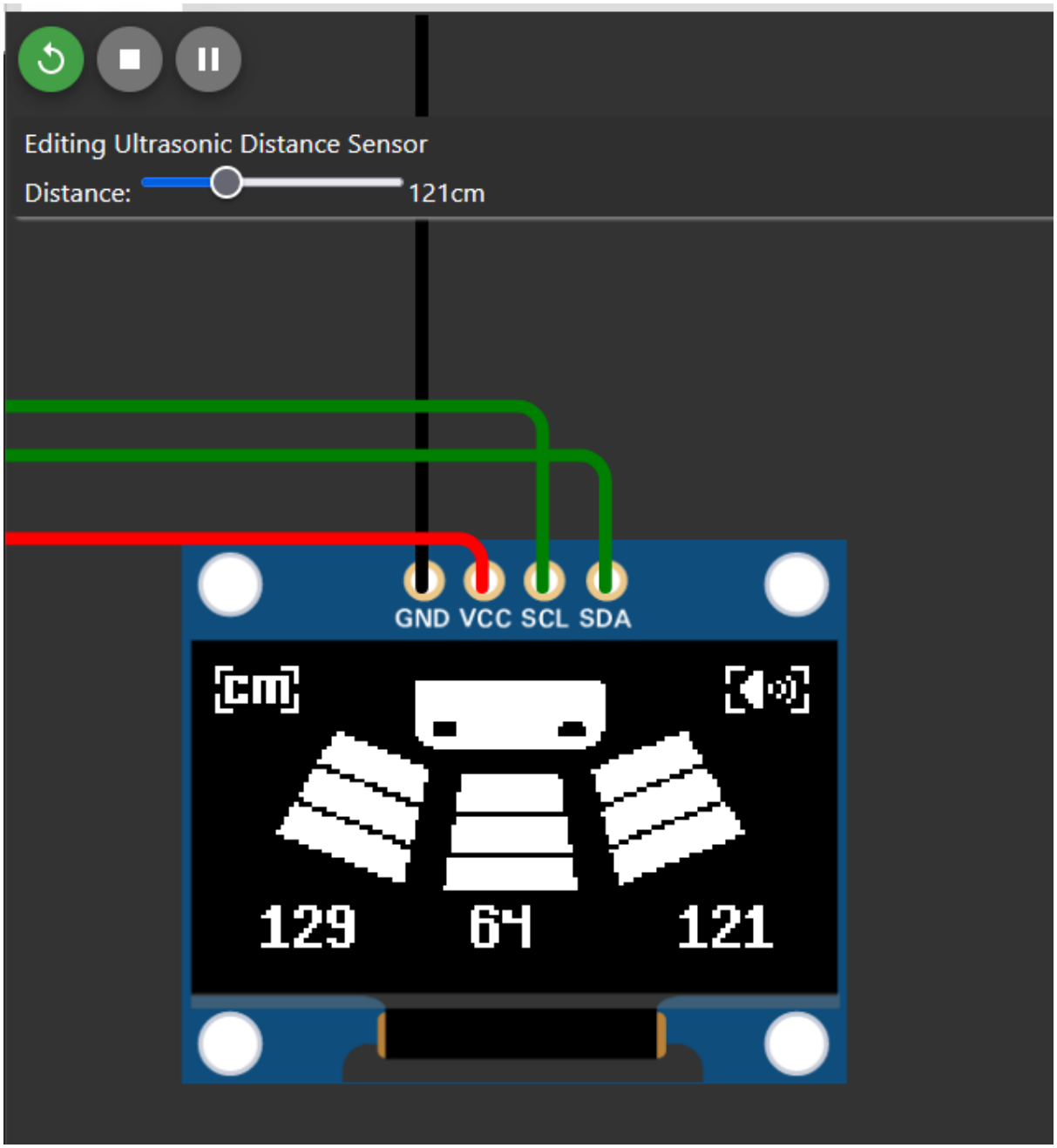
Poniżej przedstawiono zrzuty ekranu dla wartości wybranych z przedziałów







5.2. Wyświetlacz LED



6. Propozycje rozwoju projektu

6.1. Zwiększenie liczby sensorów

Pierwszym, co rzuca się na myśl w kontekście rozwoju projektu, jest poszerzenie liczby sensorów w celu zwiększenia dokładności i objęcia szerszego obszaru wokół pojazdu. Oprócz tego, można by było zwiększyć również liczbę diod i stworzyć osobne paski postępu dla każdego z sensorów.

Ponadto, można by spróbować przemodelować wizualizację na wyświetlaczu, zmieniając obraz wraz ze zbliżaniem się do obiektu.

7. Wykorzystane narzędzia

Dokumentacja sensora ultradźwiękowego:
https://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf

Symulator WOKWI:
https://wokwi.com/

Edytor Overleaf:
https://www.overleaf.com