Politechnika Świętokrzyska Wydział Elektrotechniki, Automatyki i Informatyki	
Technologie loT rozproszone sieci sensoryczne	
Laboratorium 3	Adrian Dorota 3ID15B

1. Migająca dioda za pomocą Arduino.

W tym ćwiczeniu na podstawie schematu zaprezentowanego w instrukcji laboratoryjnej(2.2.2.5 Blinking an LED using an Arduino), należało przeprowadzić symulację migającej co sekundę diody. Dioda LED w zależnośći od koloru świecenia charakteryzuje się zróżnicowanym napięciem przewodzenia:

- Czerwona
  - 2.2V.
- Żółta
  - 2.3V
- Zielona
  - 3.7V
- Niebieska
  - 4.0V
- Biała
  - 3.6V

Zaś maksymalny prąd wynosi w zależności od modelu, od 20 do 30mA.

Arduino może nadać na pin I/O przy stanie wysokim prąd o wartości :

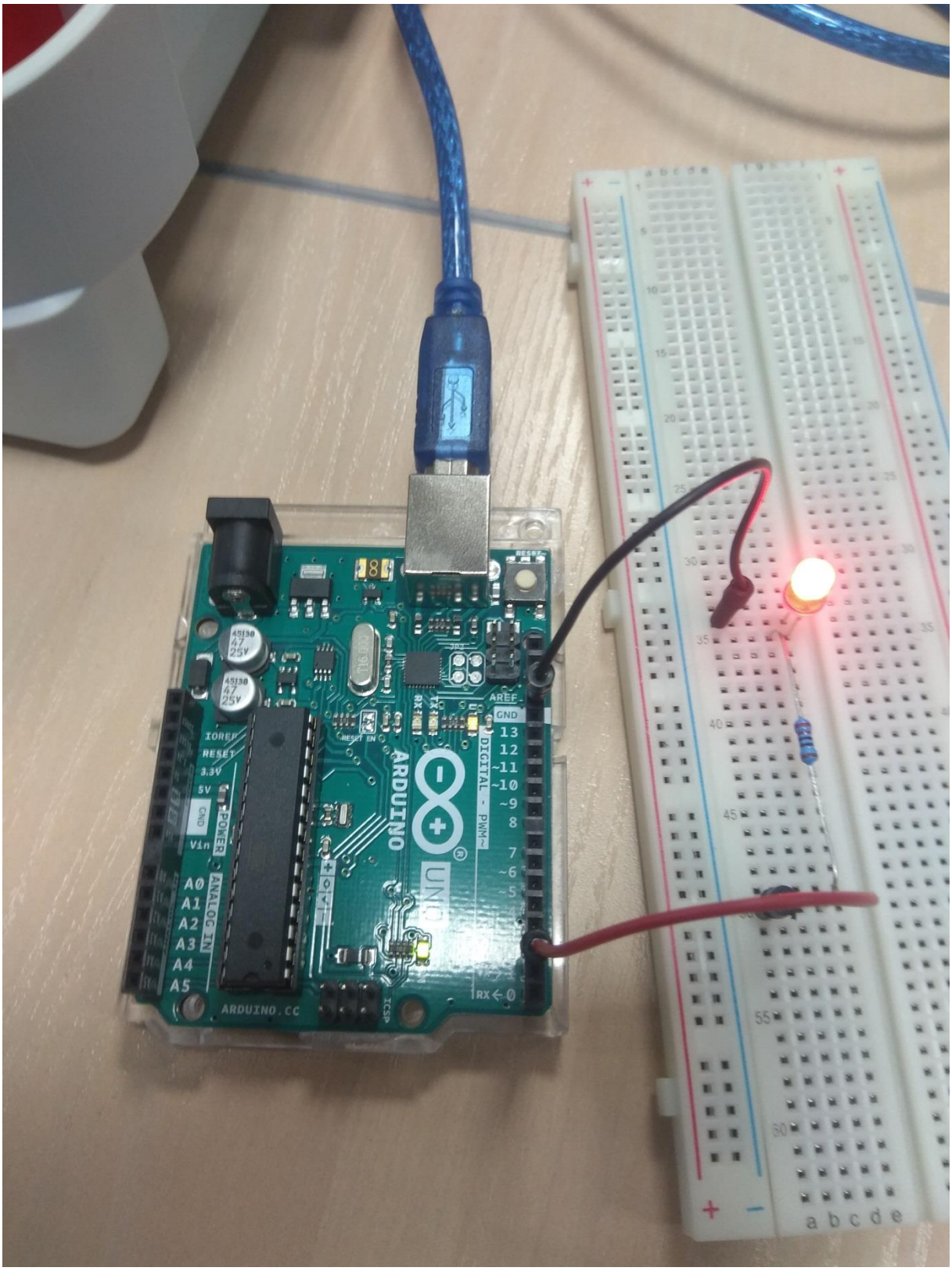
- Napięcie 5V
- Natężenie : 20mA

Są to wartości przewyższające te zawarte w specyfikacji diody LED. Podłączenie jej bezpośrednio pod port mikrokontrolera równałoby się uszkodzeniem danej diody, dlatego użyty został rezystor 220Ω by uzyskać niższy prąd który miałby przechodzić przez diodę:

$R = \frac{U}{I}$ $\quad$ $R = \frac{U_{zas} - U_{diody}}{I_{diody}}$

$I_{max} = \frac{(U_{zas} - U_{diody})}{R}$ $\quad$ $I_{max} = \frac{(5V - 2.2V)}{220\Omega} \approx 0{,}013\text{A} = 13mA$

$U = R * I$ $\quad$ $U = 220\Omega * 0{,}013\text{A} = 2.86V$


ARDUINO
UNO
ARDUINO.CC
POWER
ANALOG IN
DIGITAL (PWM~)
a b c d e

2. Buzzer

Kolejne ćwiczenie wykonano na podstawie instrukcji laboratoryjnej 2.2.4.3 (Alternate Lab – Buzzer). Celem zadania jest przetestowania działania brzęczyka(buzzer) przy zastosowaniu potencjometru do regulacji głośności dźwięku melodii.

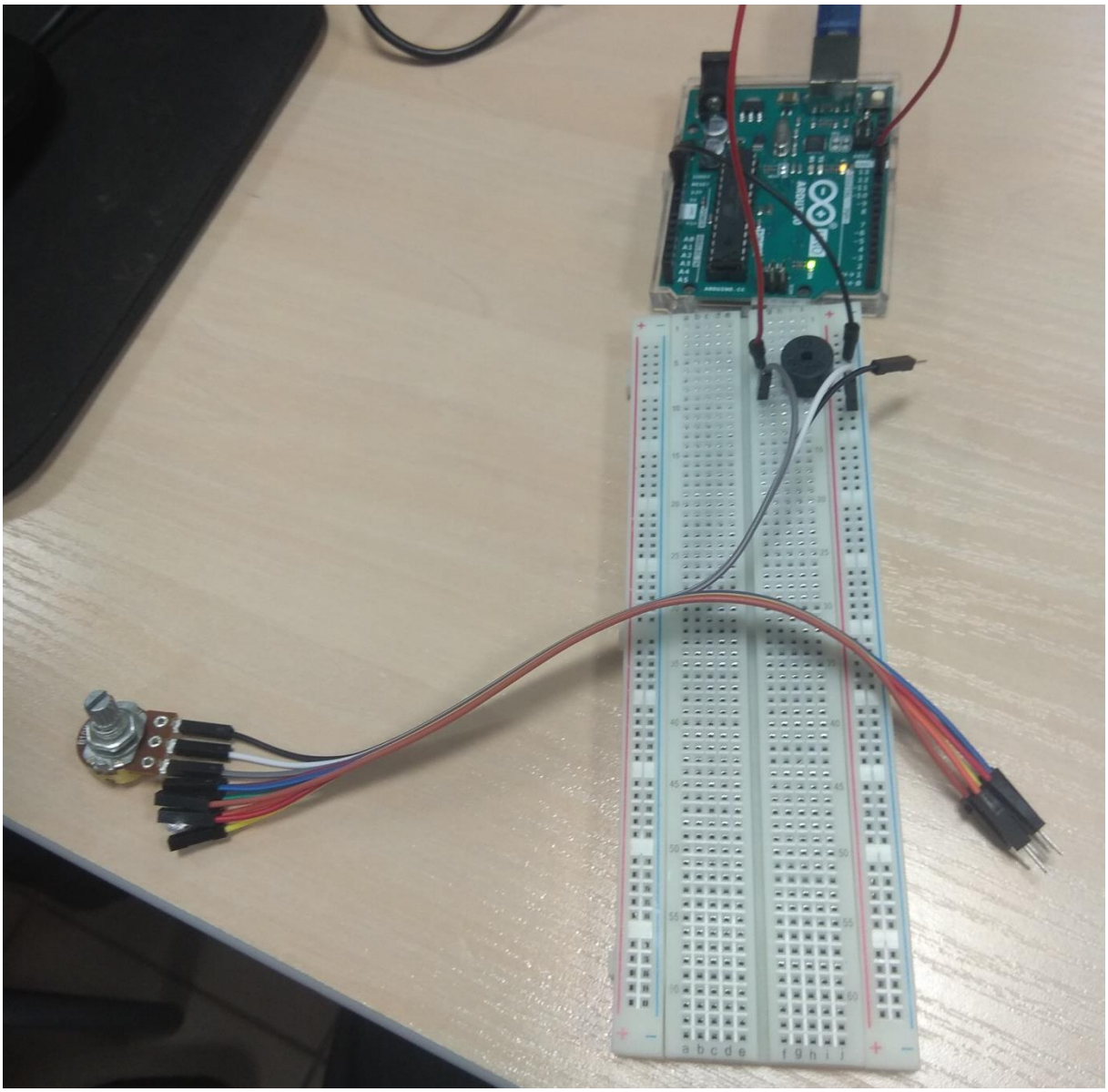
A. Kod programu dostarczony z odnośnika zamieszczonego w instrukcji:

```
/*
SparkFun Inventor's Kit
Circuit 2A - Buzzer

Play notes using a buzzer connected to pin 10

This sketch was written by SparkFun Electronics, with lots of help from the Arduino community.
This code is completely free for any use.

View circuit diagram and instructions at: https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40
Download drawings and code at: https://github.com/sparkfun/SIK-Guide-Code
*/

int speakerPin = 10;            //the pin that buzzer is connected to

void setup()
{
  pinMode(speakerPin, OUTPUT);   //set the output pin for the speaker
}

void loop()
{

  play('g', 2);     //ha
  play('g', 1);     //ppy
  play('a', 4);     //birth
  play('g', 4);     //day
  play('C', 4);     //to
  play('b', 4);     //you

  play(' ', 2);     //pause for 2 beats

  play('g', 2);     //ha
  play('g', 1);     //ppy
  play('a', 4);     //birth
  play('g', 4);     //day
  play('D', 4);     //to
  play('C', 4);     //you

  play(' ', 2);     //pause for 2 beats

  play('g', 2);     //ha
  play('g', 1);     //ppy
  play('G', 4);     //birth
```

```
  play('E', 4);      //day
  play('C', 4);      //dear
  play('b', 4);      //your
  play('a', 6);      //name

  play(' ', 2);      //pause for 2 beats

  play('F', 2);      //ha
  play('F', 1);      //ppy
  play('E', 4);      //birth
  play('C', 4);      //day
  play('D', 4);      //to
  play('C', 6);      //you

  while(true){}      //get stuck in this loop forever so that the song only plays once
}


void play( char note, int beats)
{
  int numNotes = 14;  // number of notes in our note and frequency array (there are 15 values, but arrays start at 0)

  //Note: these notes are C major (there are no sharps or flats)

  //this array is used to look up the notes
  char notes[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D', 'E', 'F', 'G', 'A', 'B', ' '};
  //this array matches frequencies with each letter (e.g. the 4th note is 'f', the 4th frequency is 175)
  int frequencies[] = {131, 147, 165, 175, 196, 220, 247, 262, 294, 330, 349, 392, 440, 494, 0};

  int currentFrequency = 0;    //the frequency that we find when we look up a frequency in the arrays
  int beatLength = 150;   //the length of one beat (changing this will speed up or slow down the tempo of the song)

  //look up the frequency that corresponds to the note
  for (int i = 0; i < numNotes; i++)  // check each value in notes from 0 to 14
  {
    if (notes[i] == note)          // does the letter passed to the play function match the letter in the array?
    {
      currentFrequency = frequencies[i];   // Yes! Set the current frequency to match that note
    }
  }

  //play the frequency that matched our letter for the number of beats passed to the play function
  tone(speakerPin, currentFrequency, beats * beatLength);
  delay(beats* beatLength);  //wait for the length of the tone so that it has time to play
  delay(50);              //a little delay between the notes makes the song sound more natural

}

/* CHART OF FREQUENCIES FOR NOTES IN C MAJOR
Note     Frequency (Hz)
c       131
d       147
e       165
f       175
g       196
```

a 220
b 247
C 262
D 294
E 330
F 349
G 392
A 440
B 494
**/*

B. Połączony obwód



Funkcja *play(char note, int beats)* odpowiada za wydawanie dźwięków przez brzęczyk.

Za parametry pobiera:

- *char note* – notacja dźwięku(nuta),
- *int beats* – długość dźwięku

```
void play( char note, int beats)
{
 int numNotes = 14;
```

rozmiar tablic

```
 char notes[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D', 'E', 'F', 'G', 'A', 'B', ' '};
 int frequencies[] = {131, 147, 165, 175, 196, 220, 247, 262, 294, 330, 349, 392, 440, 494, 0};
```

char notes[] I int frequencies[] – w tych tablicach przchowywane są parami noty i częstotliwości. Przyk. Nota o indeksie 5 odpowiada częstotliwości o indeksie 5,

```
 int currentFrequency = 0;   //the frequency that we find when we look up a frequency in the arrays
```

sterowanie długością danego dźwięku, więcej = szybciej

```
 int beatLength = 150;   //the length of one beat (changing this will speed up or slow down the tempo of the song)
```

w pętli porównywane są wartości *char notes[i]* z parametrem *note*, *jeśli się zgadzają to zapisywana jest wartość częstotliwości o danym indeksie.*

```
 for (int i = 0; i < numNotes; i++)  // check each value in notes from 0 to 14
 {
   if (notes[i] == note)          // does the letter passed to the play function match the letter in the array?
   {
     currentFrequency = frequencies[i];   // Yes! Set the current frequency to match that note
   }
 }
```

Graj dźwięk o wybranej częstotliwości I danej długośći

```
   tone(speakerPin, currentFrequency, beats * beatLength);
```

odczekaj długość aktualnego dźwięku I trochę dłużej by się nie mieszało

```
 delay(beats* beatLength);   //wait for the length of the tone so that it has time to play
 delay(50);              //a little delay between the notes makes the song sound more natural

}
```

Sekwencja zagra tylko raz z powodu niekończącej się pętli w głównej funkcji programu, konieczne wciśniecie przycisku RESET na mikrokontrolerze by odtworzyć ponownie. Aby zagrać inny utwór należy wywoływać funkcję play() z odpowiednimi parametrami częstotliwości i długości.