Instytut Informatyki
Wydział Informatyki i Telekomunikacji, Politechnika Poznańska

Laboratorium z przedmiotu Systemy wbudowane (SW)

Karta projektu – zadanie 7

Nazwa projektu:

Alkomat – baza danych z czytnikiem kart + Web Server

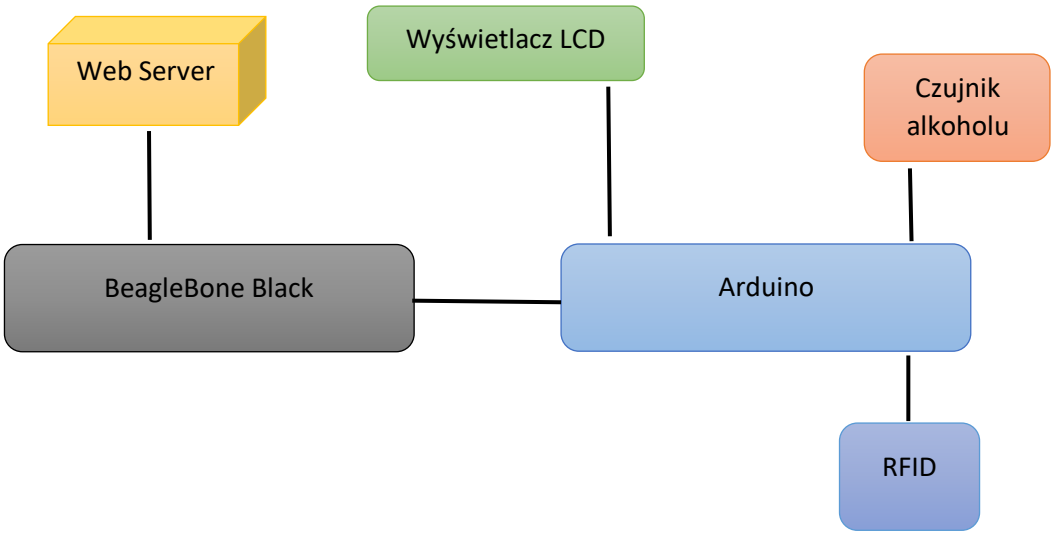
Prowadzący:	*Autorzy (tylko nr indeksu):*	*Grupa dziekańska:*	*I4.1*
Ariel Antonowicz	145333 145248	***Ocena:***	

Cel projektu:

Celem projektu jest zaprojektowanie układu, którego zadaniem będzie mierzenie alkoholu w wydychanym powietrzu pracowników firmy. Dodatkowo wyniki będą wyświetlane na wyświetlaczu LCD oraz zapisywane w bazie danych. Aby wykonać pomiar, będzie trzeba przyłożyć kartę, która będzie identyfikatorem badania. W bazie danych będą przechowywane: id badania, data badania, wynik, decyzja (czy można prowadzić samochód).

Dodatkowo będzie opcja administratora, który po przyłożeniu swojej karty będzie miał możliwość wykonania pomiaru lub akcji na bazie danych. Admin będzie miał dostęp do jej zawartości oraz będzie mógł wyczyścić bazę danych. Wszystkie akcje będą odbywały się poprzez interfejs na serwerze webowym postawionym na BeagleBone Black.

Schemat:



Wykorzystana platforma sprzętowa, czujniki pomiarowe, elementy wykonawcze:

- Arduino [Program Arduino IDE + C]
- BeagleBone Black z bazą danych [Program PyCharm + Python + SQL]
- Czytnik RFID
- Wyświetlacz LCD
- Czujnik alkoholu MQ-135
- Web serwer na BeagleBone

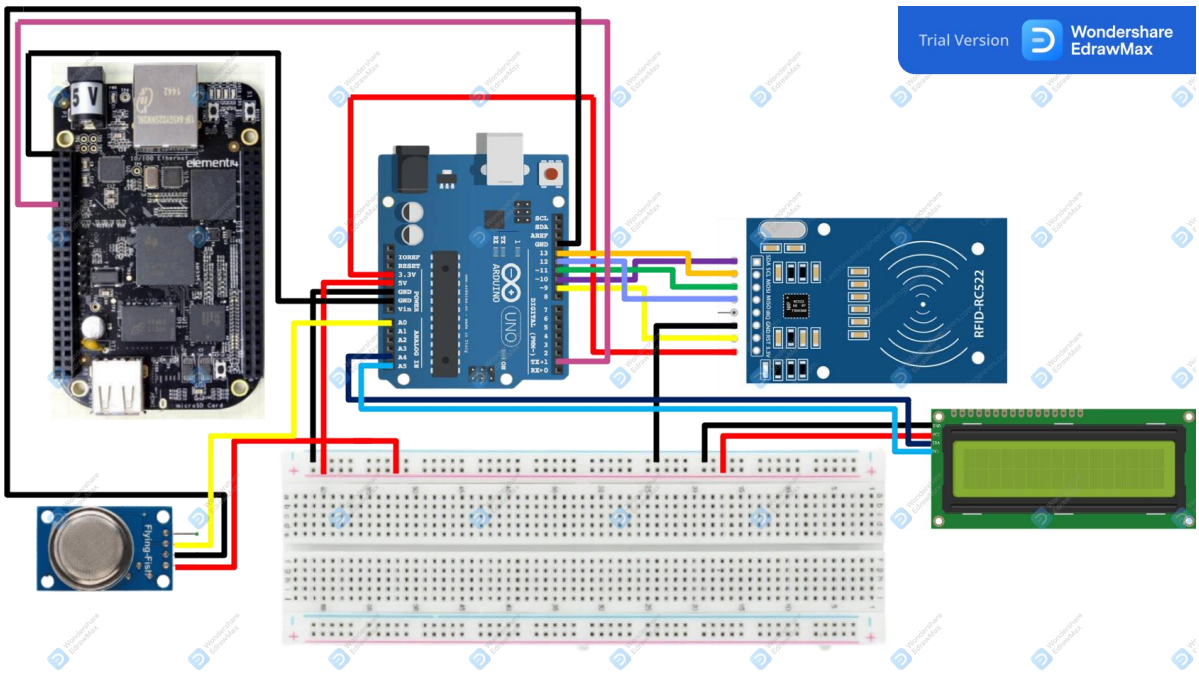
Poznań 2022

1. Zakres projekt

Opis celu oraz zakresu projektu został umieszczony w karcie projektu. Szczególnie temat części ‘fizycznej’ został w niej w niej wyczerpany. Kilka słów należy jednak dodać do części softwarowej.

Dostęp do bazy danych będzie odbywał się za pomocą interfejsu na serwerze webowym postawionym na BeagleBone Black za pomocą technologii FLASK. Na serwerze tym użytkownik będzie miał dostęp do przeglądu bazy danych, dodatkowo serwer będzie posiadał opcje modyfikacji, czyszczenia dostępne tylko dla admina po podaniu loginu i hasła.

2. Schemat



Rys.1 Schemat połączeniowy

3. Projekt a realizacja

Celem projektu było stworzenie alkomatu dla firm, zapisującego wyniki w bazie danych do której dostęp zapewniał serwer postawiony na BeagleBone Black. W tym celu skorzystaliśmy z technologii FLASK. Strona zapewnia swobodny i wygodny dostęp do przeglądu bazy danych oraz zapewnia dodatkowe opcje takie jak czyszczenia bazy danych, przegląd wyników, w których pracownik był/nie był pod wpływem alkoholu. Opcje te są możliwe jedynie po wprowadzeniu loginu i hasła ADMINA. Jest tutaj zdecydowanie pole do rozwinięcia, dodanie kilku opcji takich jak przegląd pomiarów od określonego pracownika. Jako czujnik alkoholu użyliśmy MQ-135. Sensor ten jest dość tanią opcją czujnika alkoholu. Początkowo w założeniach projektu chcieliśmy zmierzyć dokładną ilość alkoholu w wydychanym powietrzu.

Poznań 2022

Z uwagi jednak na specyfikacje czujnika wyniki pomiarów traktowane są zero jedynkowo – ktoś jest lub nie jest pod wpływem alkoholu. Zmiana czujnika na lepszy model mogłaby zdecydowanie zwiększyć jakość pomiarów.

W stosunku do początkowego zarysu projektu jak i karty dokonaliśmy kilka innych korekt. Admin strony musi podać jedynie login i hasło - bez konieczności okazania karty. Jest to bardziej naturalne i wygodne. W bazie danych zamiast daty pomiaru widnieje przyjęta przez nas nazwa pracownika - przypisana do danej karty. W bazie danych zamiast stwierdzenia pijany/nie pijany zapisujemy, czy pracownik jest zdolny czy niezdolny do pracy. W przypadku kiedy pomiar okazał się błędny:

- użytkownik nie chciał oszukać alkomat nie dmuchając
- użytkownik nie dmuchał w alkomat

W decyzji znajdującej się w bazie danych widnieje informacja o błędzie.

4. Najważniejsze fragmenty kodu z komentarzami

Arduino:

W naszym projekcie Arduino odpowiedzialne jest za wykonanie pomiaru oraz przesłanie go do BeagleBoneBlack. Dodatkowo Arduino obsługuje wyświetlacz LCD, na którym wyświetla komunikaty dla użytkownika.

```
#include <Wire.h>   // standardowa biblioteka Arduino
#include <Servo.h>
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>  // dolaczenie pobranej biblioteki I2C dla LCD

#define MQpin 0
#define SS_PIN 10
#define RST_PIN 9

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Ustawienie adresu ukladu na 0x27
float sensorValue;  //variable to store sensor value
float suma = 0;
float testSensorValue;
MFRC522 mfrc522(SS_PIN, RST_PIN);

// Karty RFID
const byte brelok[] = {0xCC, 0xD4, 0x2D, 0x3B};
const byte biala_karta[] = {0x99, 0x09, 0x9B, 0x75};
const byte legitka_T[] = {0x4F, 0xD3, 0xF9, 0x7D};
const byte legitka_S[] = {0xAF, 0x2B, 0xC4, 0x77};
const byte LM_admin[] = {0xFE, 0x7D, 0xE8, 0xFB};


void setup()
{
 Serial.begin(9600); // sets the serial port to 9600
 SPI.begin();
 mfrc522.PCD_Init();
 lcd.init();
 lcd.backlight();
 lcd.setCursor(0,0);
 lcd.print("MG135 Warming up");
 delay(2500);

}
```

Rys.2 Definicja stałych i setup programu

Kod wykonujący pomiar składa się z kilku instrukcji warunkowych, w których sprawdzamy, czy przyłożona karta znajduje się wśród stałych. Następnie program wykonuje testowy pomiar, a później prosi użytkownika o dmuchnięcie. Po zakończeniu drugiego pomiaru obliczana jest różnica między wynikami i na jej podstawie podejmowana jest decyzja.

```
//Czy admin
if (mfrc522.uid.uidByte[0] == LM_admin[0] &&
    mfrc522.uid.uidByte[1] == LM_admin[1] &&
    mfrc522.uid.uidByte[2] == LM_admin[2] &&
    mfrc522.uid.uidByte[3] == LM_admin[3] )
    {
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Przygotowanie");
        lcd.setCursor(0,1);
        lcd.print("alkomatu");
         for(int i = 0; i<30; i++) {
          suma += analogRead(MQpin);
          delay(150);
          //Serial.println(suma);
        }

        testSensorValue = suma/30;
        suma = 0;


        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Dmuchnij za 3s");
        delay(1000);
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Za 2s");
        delay(1000);
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Za 1s");
        delay(1000);
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Dmuchaj");

        for(int i = 0; i<30; i++) {
          suma += analogRead(MQpin);
          delay(250);
          //Serial.println(suma);
        }
        sensorValue = suma/30;
        sensorValue -= testSensorValue;
        delay(5000);

        //Serial.print("Sensor Value: ");
        Serial.print(mfrc522.uid.uidByte[0]);
        Serial.print(mfrc522.uid.uidByte[1]);
        Serial.print(mfrc522.uid.uidByte[2]);
        Serial.println(mfrc522.uid.uidByte[3]);

        Serial.println(sensorValue);

        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Wynik:");
        lcd.setCursor(0,1);
        lcd.print(sensorValue);
        lcd.setCursor(0,0);
        delay(5000);
        lcd.clear();

        if (sensorValue < 12 && sensorValue > -12){
          lcd.print("Decyzja");
          lcd.setCursor(0,1);
          lcd.print("Brak pomiaru");
          delay(5000);
          lcd.clear();

        }else if(sensorValue < 90){
          lcd.print("Decyzja");
          lcd.setCursor(0,1);
          lcd.print("Zdolny do pracy!");
          delay(5000);
          lcd.clear();
          }else{
          lcd.print("Decyzja");
          lcd.setCursor(0,1);
          lcd.print("Pijany!");
          delay(5000);
          lcd.clear();
            }
}
```

Rys.3 Instrukcja warunkowa dla karty admin

Obsługa bazy danych BBB:

Drugim urządzeniem, które wykorzystaliśmy w projekcie jest BeagleBoneBlack. Jego zadaniem jest obsługa bazy danych, w której przechowujemy wyniki pomiarów i dane użytkowników, których te pomiary dotyczą. Beagle tworzy tabele „wyniki” (jeśli ona nie istnieje) i po otrzymaniu wyników z Arduino za pomocą portu szeregowego zapisuje odpowiednie dane do tabeli

```
if __name__ == '__main__':
    db = sqlite3.connect('Baza_Danych.db')
    cursor = db.cursor()
    cursor.execute(
        'CREATE TABLE IF NOT EXISTS wyniki(id INTEGER PRIMARY KEY, pomiar
REAL, karta VARCHAR , text VARCHAR);')
    db.commit()
    UART.setup("UART4")
    ser = serial.Serial(port="/dev/ttyO4", baudrate=9600)
    ser.close()
    ser.open()
    if ser.isOpen():
        print("Serial is open!")
    karta = ser.readline()
    pomiar = float(ser.readline())

    if pomiar > 200:
        decyzja = "Niezdolny do pracy"
    else:
        decyzja = "Zdolny do pracy"

    if int(karta) == 254125232251:
        prac = "Admin"
    elif int(karta) == 2042124559:
        prac = "Wozny"
    elif int(karta) == 1539155117:
        prac = "Sekretarz"
    elif int(karta) == 79211249125:
        prac = "Kurier Tobiasz"
    elif int(karta) == 17543196119:
        prac = "Kurier Sebastian"
    else:
        prac = "Blad autoryzacji"

    cursor.execute('INSERT INTO wyniki(pomiar, karta, text) '
                   'VALUES(?,?,?)', (pomiar, str(prac), str(decyzja)))
    db.commit()
    ser.close()
```

Rys.4 Kod obsługujący bazę danych na BeagleBoneBlack

Aplikacja BBB:

Nasza aplikacja webowa została napisana w języku Python przy wykorzystaniu frameworka Flask. Jest ona uruchamiana na porcie 1234 BeagleBoneBlack. Aplikacja składa się z kilku ekranów. Najważniejszymi z nich są:

Strona główna „O projekcie”.

```
@app.route('/')
@app.route('/home')
def home():
    return render_template('projekt.html')
```

Rys.5 Kod dla strony projektu

Strona zawierająca raport z bazy danych „Raport”. Wykorzystując polecenie SQL wyświetlamy zawartość tabeli przechowującej wyniki pomiarów.

```
@app.route('/raport', methods=["GET", "POST"])
def raport():
    if request.method == 'GET':
        db_r, cursor_r = cursor()

        cursor_r.execute('SELECT * FROM wyniki')
        all_rows = cursor_r.fetchall()
        return render_template('raport.html', items=all_rows)
    else:
        db_r, cursor_r = cursor()

        cursor_r.execute('DELETE FROM wyniki WHERE id > 0')

        db_r.commit()

        return render_template('raport.html')
```

Rys.6 Kod dla strony z raportem

Panel admina umożliwiający dodatkową analizę i wyczyszczenie bazy danych, do której dostęp jest poprzedzony ekranem logowania „Modyfikacje”. Przy wykorzystaniu poleceń SQL można wyświetlić raport zawierający pomiary, które wykazały, że pracownik był/nie był pod wpływem alkoholu.

```python
@app.route('/login', methods=["GET", "POST"])
def login():
    if request.method == 'GET':
        return render_template('login.html')
    else:
        login = request.form.get('login')
        haslo = request.form.get('hasło')
        trzezwi = request.form.get('trzezwi')
        nietrzezwi = request.form.get('nietrzezwi')
        if trzezwi == 'trzezwi':
            db_r, cursor_r = cursor()

            cursor_r.execute('SELECT * FROM wyniki where text = "Zdolny do pracy" ')
            all_rows = cursor_r.fetchall()
            return render_template('trzezwi.html',items=all_rows)
        elif nietrzezwi == 'nietrzezwi':
            db_r, cursor_r = cursor()

            cursor_r.execute('SELECT * FROM wyniki where text = "Niezdolny do pracy" ')
            all_rows = cursor_r.fetchall()
            return render_template('nietrzezwi.html', items=all_rows)
        elif login == 'admin' and haslo == "admin":
            return render_template('modyfikacje.html')
        else:
            flash("Błędne dane logowania! Proszę sprawdzić dane logowania i spróbować ponownie.")
            return render_template('login.html')
```

Rys.7 Kod dla strony z logowaniem

Każda z stron, która przeprowadza akcje na bazie danych wykorzystuje kursor. W tym celu stworzyliśmy prostą funkcję, która zwraca nam kursor umożliwiający wykonywanie działań na bazie danych. Dodatkowo funkcja tworzy tabele przechowującą wyniki, w przypadku gdy taka nie znajduje się w bazie.

```python
def cursor():
    db = sqlite3.connect('Baza_Danych.db')
    db.row_factory = sqlite3.Row
    cursor = db.cursor()
    cursor.execute(
        'CREATE TABLE IF NOT EXISTS wyniki(id INTEGER PRIMARY KEY, pomiar REAL, karta VARCHAR , text VARCHAR);')
    db.commit()

    return db, cursor
```

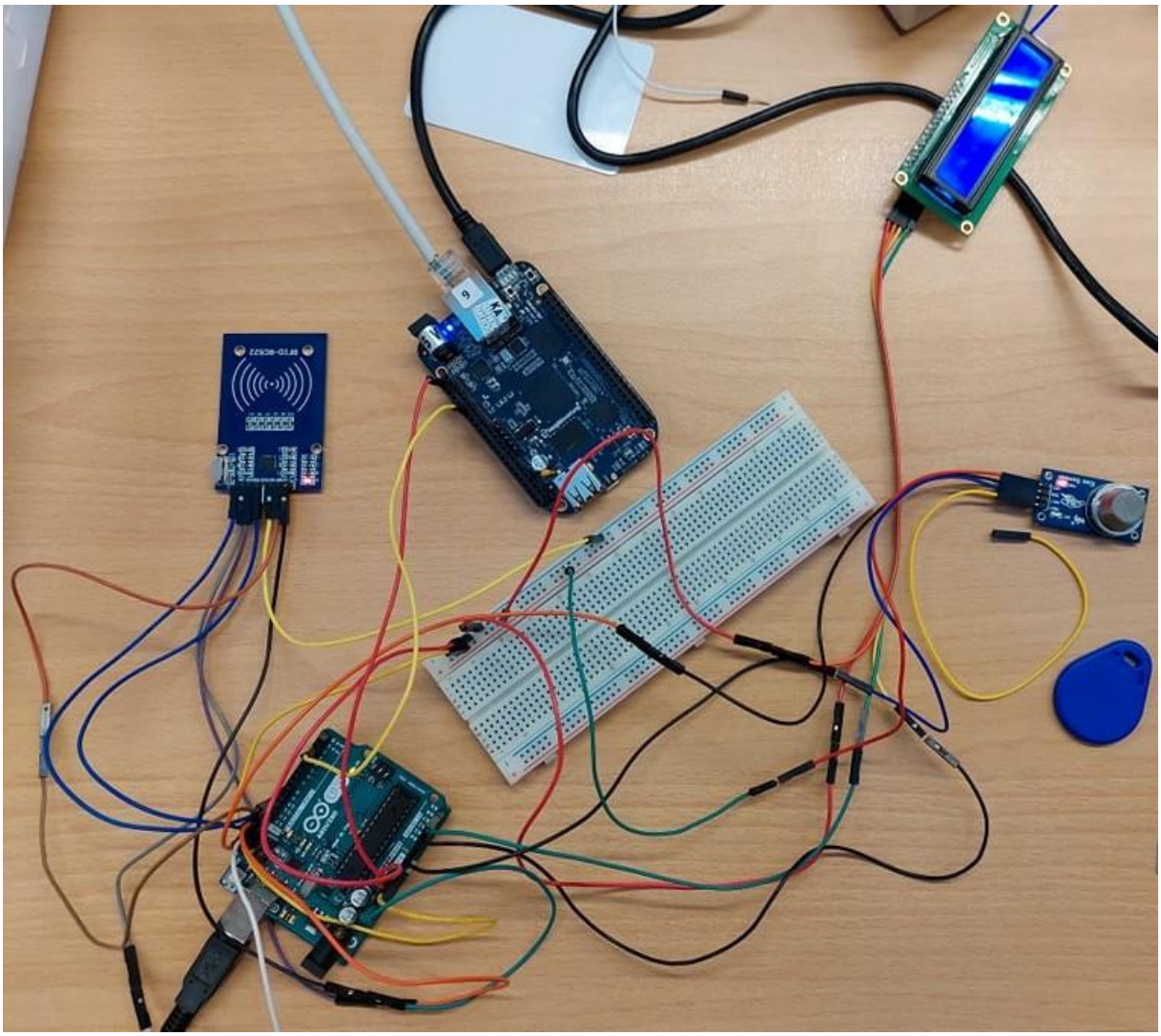
Rys.8 Funkcja zwracająca kursor

5. Zdjęcie połączeń

Opis:

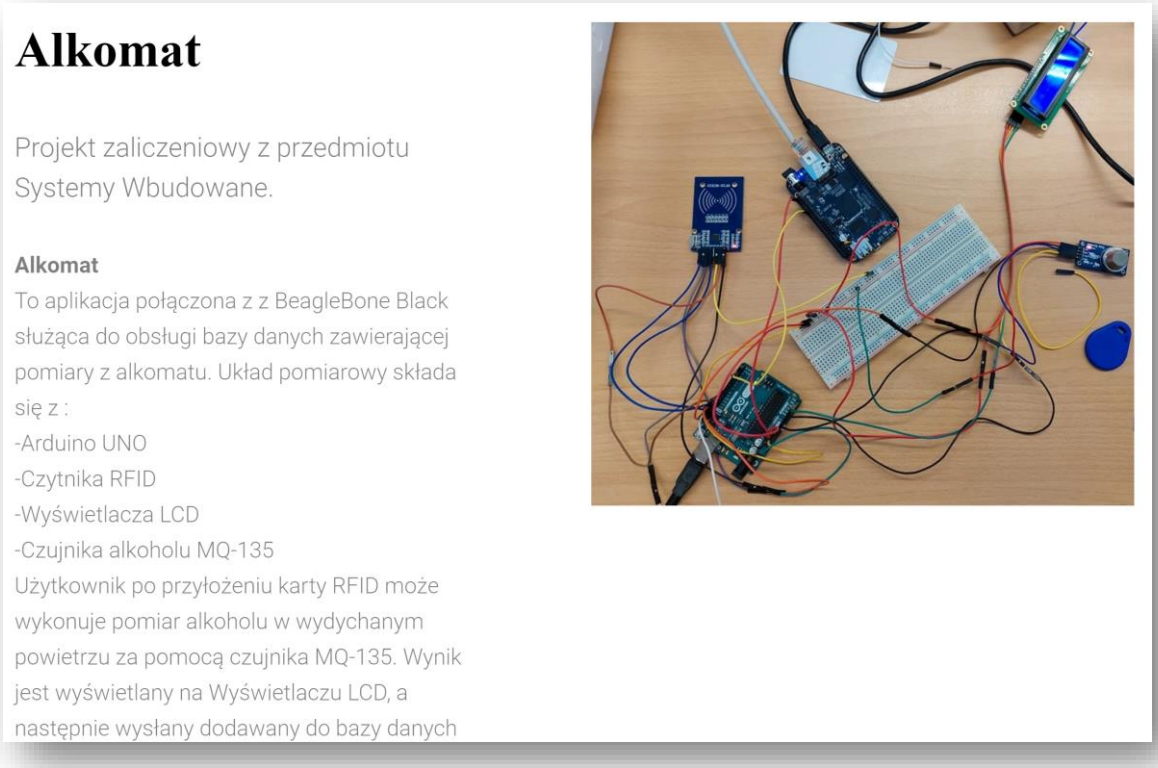
Do realizacji projektu wykorzystaliśmy sprzęt znany nam z zajęć laboratoryjnych. Dzięki komunikacji Arduino oraz BeagleBoneBlack wykonaliśmy system sprawdzający ilość alkoholu w wydychanym przez nas powietrzu.

Zdjęcie połączeń:



Rys.9 Schemat połączeń

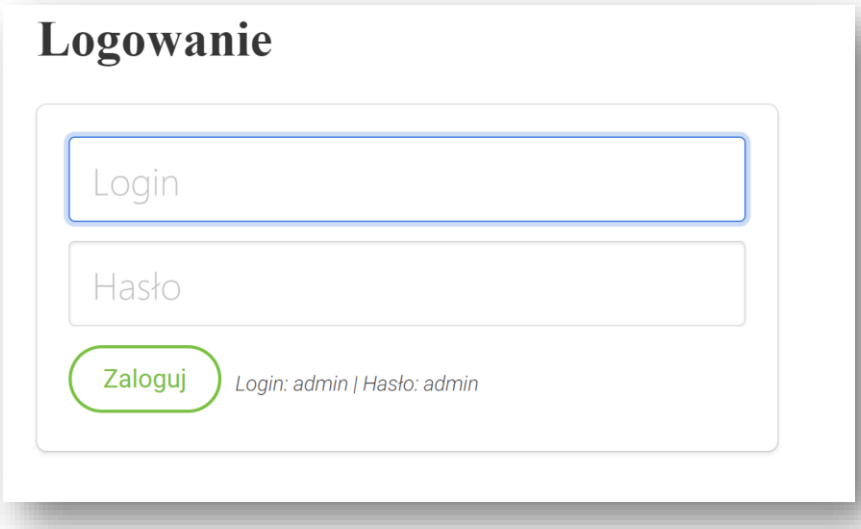
6. Zrzuty z ekranu aplikacji



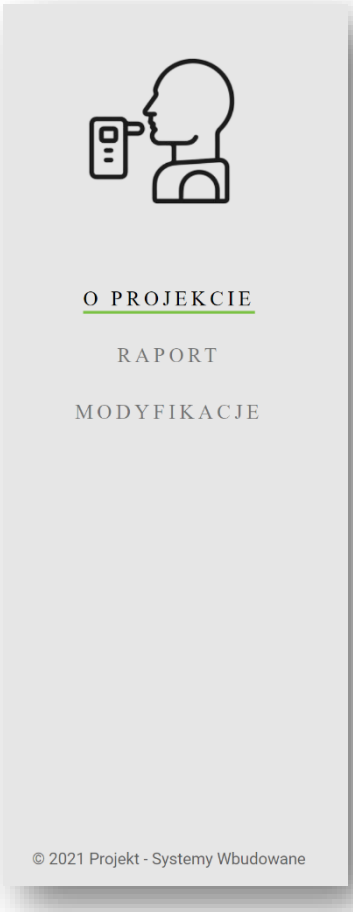
Rys.10 Strona główna

Rys.11 Ekran raportu

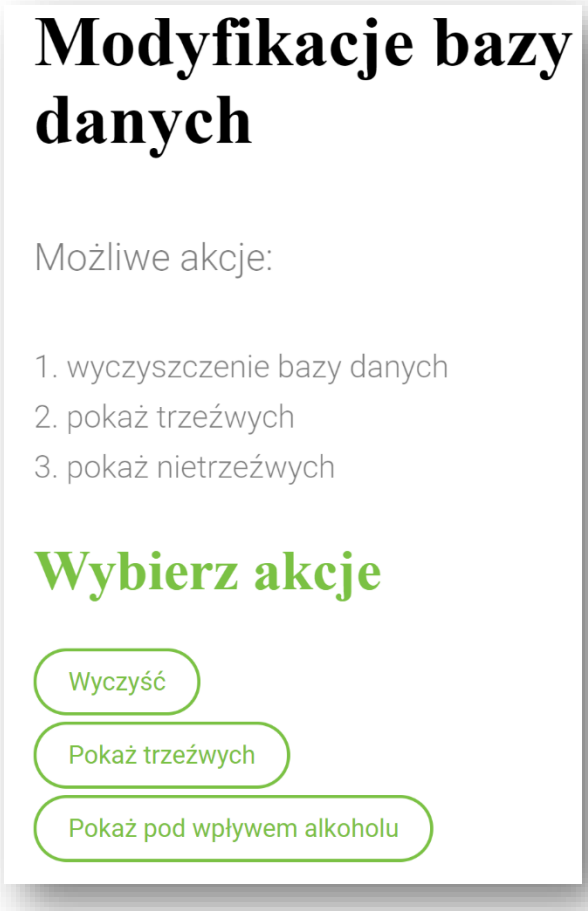
Poznań 2022



Rys.12 Ekran logowania administratora



Rys.13 Pasek nawigacyjny



Rys.14 Ekran modyfikacji

Poznań 2022

7. Kilka słów podsumowania

Podsumowując nasz projekt, udało nam się uzyskać podstawową funkcjonalność projektu. Zbudowaliśmy alkomat połączony z BeagleBoneBlack, który umożliwia nam swobodny dostęp do bazy danych pomiarów. Alkomat ten znalazłby swoje zastosowanie w tym momencie, ale z pewnością jest on dobrym wstępem do budowy pełnoprawnego alkomatu, mierzącego dokładną ilość alkoholu w wydychanym powietrzu. Dużym plusem projektu byłe zadecydowanie praca z BeagleBoneBlack oraz postawienie na nim serwera przy wykorzystaniu frameworka Flask. Wiedza oraz umiejętności nabyte podczas wykonywania projektu „Alkomat" niewątpliwie zbliżyły nas do bycia prawdziwymi inżynierami przemysłu 4.0 i z pewnością okażą się bardzo przydatne w przyszłości.