POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

INSTYTUT AUTOMATYKI, ROBOTYKI I INŻYNIERII

INFORMATYCZNEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

SPRAWOZDANIE

PROJEKT KOŃCOWY

MATEUSZ WĄTŁY

APLIKACJE MOBILNE I WBUDOWANE DLA INTERNETU PRZEDMIOTÓW

PROJEKT

PROWADZĄCY:
MGR INŻ. ADRIAN WÓJCIK

POZNAŃ 14.01.2020

DANE SZCZEGÓŁOWE

Rok studiów: INŻ. III

Rok akademicki: 2019/2020

Termin zajęć: wtorek g. 11:45

Data wykonania: 2020/01/14

Temat zajęć: Projekt końcowy

Prowadzący: Adrian Wójcik

Skład grupy (nazwisko, imię, nr indeksu):

1. Wątły Mateusz 131390

OCENA (*WYPEŁNIA PROWADZĄCY*)

Spełnienie wymogów redakcyjnych: ……

……

Wykonanie, udokumentowanie oraz opis wykonanych zadań: ……

……

Zastosowanie prawidłowego warsztatu programistycznego: ……

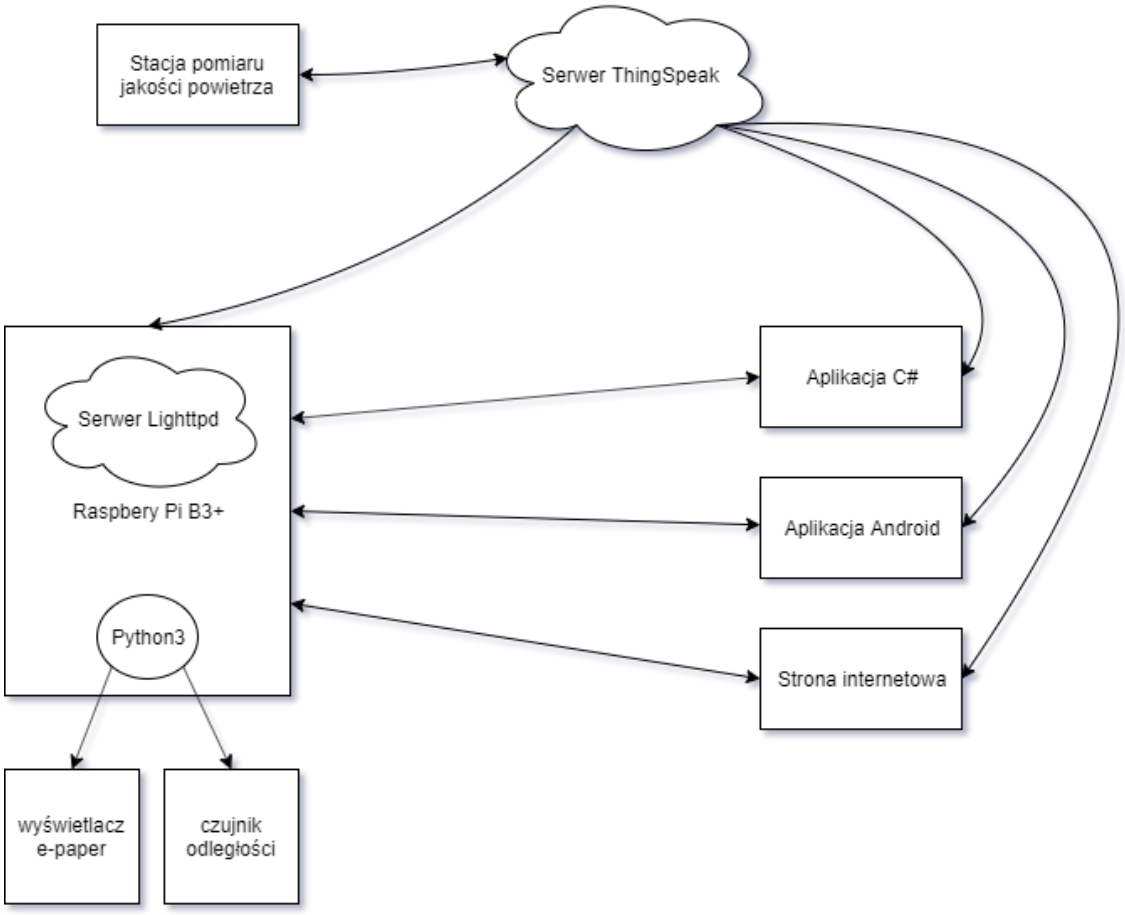
……

Spis treści

1 WSTĘP

W poniższym sprawozdaniu przedstawiono przebieg projektu z przedmiotu Aplikacje Mobilne.

Ogólna struktura stworzonego systemu została przedstawiona na poniższej ilustracji.

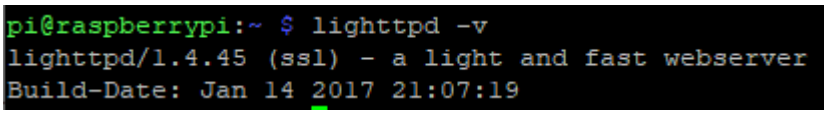


2 RASPBERRY PI

Jako komputera który pełni rolę serwera użyto Raspberry Pi B 3+ z zainstalowanym systemem Raspbian opartym na Debianie. Program PuTTY posłużył do komunikacji z konsolą Raspberry, natomiast pliki przesyłano z użyciem WinSCP.

2.1 SERWER LIGHTTPD

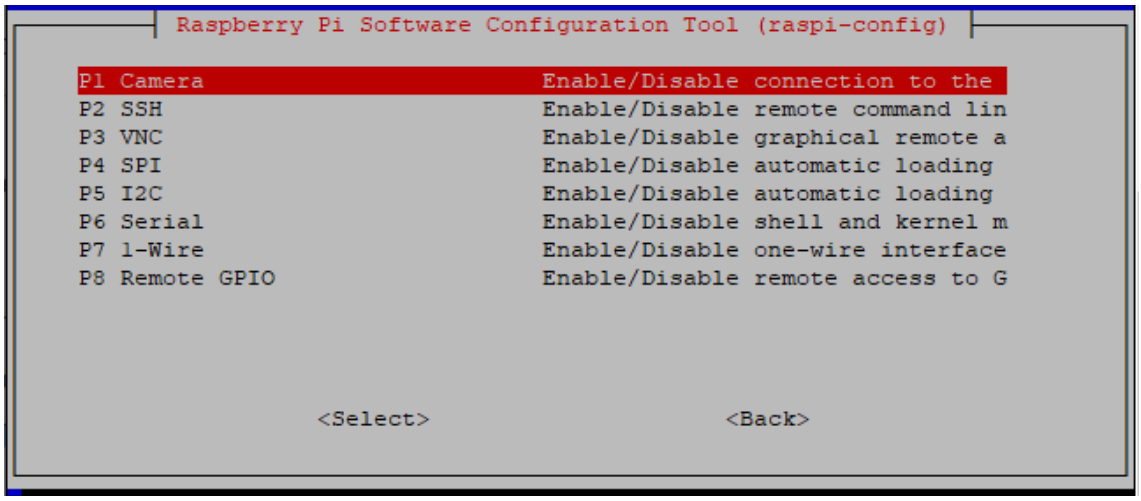
Zainstalowano serwer WWW – Lighttpd [1] wraz z modułami do obsługi skryptów PHP (Rys. 1).



Rys. 1 Wersja zainstalowanego serwera

2.2 SKRYPTY PYTHON – OBSŁUGA URZĄDZEŃ

Do Raspberry podłączono wyświetlacz e-paper [2] i ultradźwiękowy czujnik odległości [3]. Aby możliwe było korzystanie z nich, konieczne było aktywowanie odpowiednich interfejsów na Raspberry (Rys. 2).



Rys. 2 Konfiguracja interfejsów Raspberry

Skrypty obsługujące urządzenia zostały napisane w języku Python3. Skorzystano ze zmodyfikowanej klasy dostarczonej przez producenta obsługującej wyświetlacz e-paper, natomiast dla czujnika odległości stworzono funkcję zwracającą odległość w centymetrach.

Wywołując skrypty możliwe jest podawanie argumentów (Listing 1). Dzięki temu modyfikowane są parametry wybranych operacji.

```
1.  text = ""
2.  if len(sys.argv) >= 2:
3.      text = sys.argv[1]
```

Listing 1 Argumenty w Pythonie

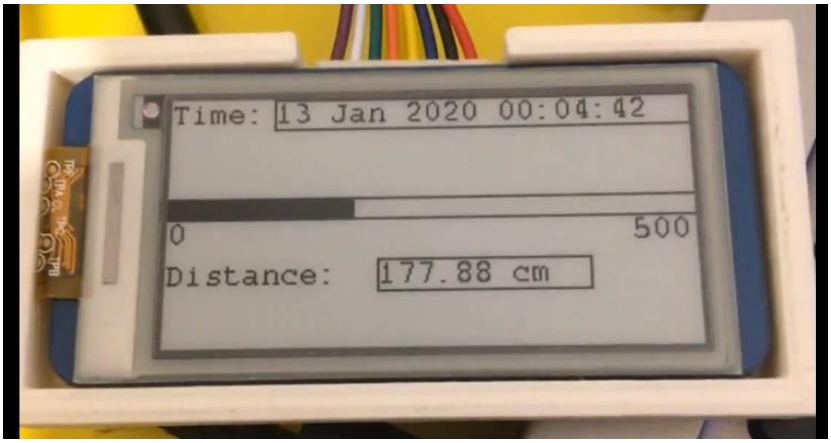
Poniżej przedstawiono skrypty które stworzono w celu zrealizowania projektu wraz z krótkim opisem działania:

- clear.py:

Skrypt po wywołaniu czyści wyświetlacz e-paper, pozostawiając go białym.

- distance.py:

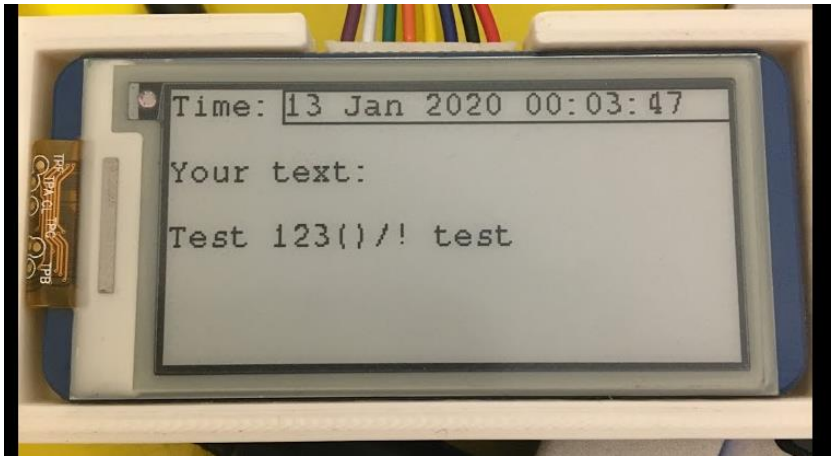
Przeprowadzany jest pomiar odległości z użyciem czujnika ultradźwiękowego. Rezultat pomiaru zwracany jest jako odległość w centymetrach (float z dokładnością do dwóch miejsc po przecinku). Wynik wyświetlany jest także na wyświetlaczu e-paper w formie tekstowej i graficznie jako oś (Rys. 3). Podczas wywoływania skryptu możliwe jest podanie argumentu określającego maksymalną mierzoną odległość – wartość ta będzie maksimum osi.



Rys. 3 Efekt działania skryptu distance.py

- send_text_epaper.py:

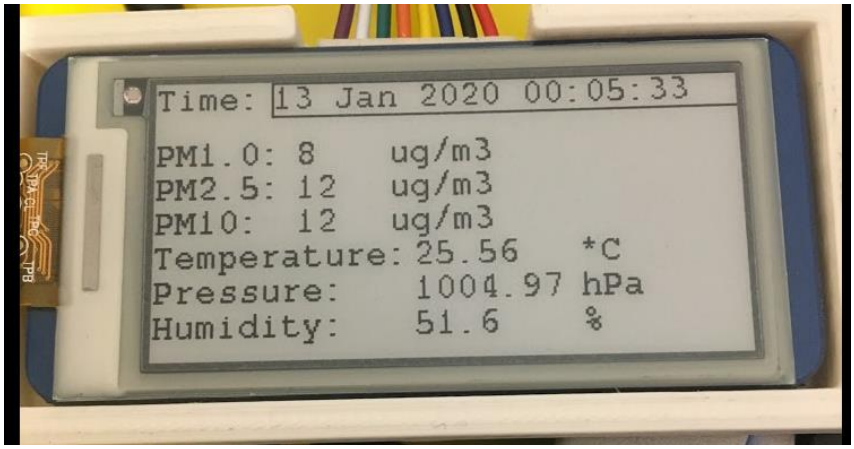
Skrypt który wyświetla tekst wysłany do niego w argumencie na wyświetlaczu e-paper. Obsługa ograniczona jest do jednej linii i szerokości wyświetlacza (Rys. 4).



Rys. 4 Efekt działania skryptu send_text_epaper.py

- main.py:

Skrypt pobiera z zewnętrznego serwera dane pomiarowe w formacie JSON, przetwarza je i wyświetla na wyświetlaczu (Rys. 5).



Rys. 5 Efekt działania skryptu main.py

2.3 POŁĄCZENIE Z ZEWNĘTRZNYM SERWEREM THINGSPEAK

W skrypcie main.py użyto danych mierzonych przez układ zbudowany w celach pracy dyplomowej. Są to dane dotyczące parametrów i zanieczyszczeń powietrza. Aby nie przerabiać samego urządzenia skorzystano z API zewnętrznego serwera ThingSpeak na którym dane te są przechowywane [4]. Metodą GET otrzymywany jest najnowszy pomiar (Listing 2) (Listing 3).

```
def get_thingspeak_data():
    url = requests.get('https://api.thingspeak.com/channels/775759/feeds.json?results=1')
    j = url.json()
    return j
```

Listing 2 Odczyt danych z serwera ThingSpeak

```
{
    "channel": {
        "id": 775759,
        "name": "AirQualityStation",
        "description": "Indoor test",
        "latitude": "0.0",
        "longitude": "0.0",
        "field1": "Temperature",
        "field2": "Pressure",
        "field3": "Humidity",
        "field4": "PM1.0",
        "field5": "PM2.5",
        "field6": "PM10",
        "field7": "NUM_OF_PAR_0_3_UM_IN_0_1_L_OF_AIR",
        "field8": "NUM_OF_PAR_0_5_UM_IN_0_1_L_OF_AIR",
        "created_at": "2019-05-07T12:47:10Z",
        "updated_at": "2019-10-25T22:39:25Z",
        "last_entry_id": 10582
    },
    "feeds": [
        {
            "created_at": "2020-01-12T16:16:56Z",
            "entry_id": 10582,
            "field1": "25.31",
            "field2": "1005.96",
            "field3": "60.6",
            "field4": "22",
            "field5": "33",
            "field6": "40",
            "field7": "3798",
            "field8": "1138"
        }
```

```
    ]
}
```

Listing 3 Odpowiedź serwera ThingSpeak

Następnie dane te są parsowane z formatu JSON do obiektów (Listing 4).

```
data = get_thingspeak_data()
time.sleep(1)

channel_id = data["channel"]["id"]
for i in range(8):
    fields_names[i] = data["channel"][fields[i]]
for i in range(8):
    fields_feeds[i] = data["feeds"][0][fields[i]]
created_at = data["feeds"][0]["created_at"]

draw.rectangle((60, 25, 100, 40), fill=255,)
draw.text((60, 25), str(fields_feeds[fields_names.index("PM1.0")]), font=font15, fill=0)
```

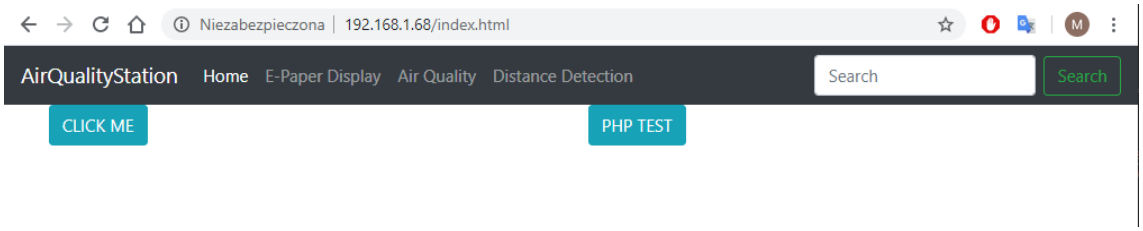
Listing 4 Przykład użycia i wyświetlenia danych

3 STRONA INTERNETOWA

3.1 HTML

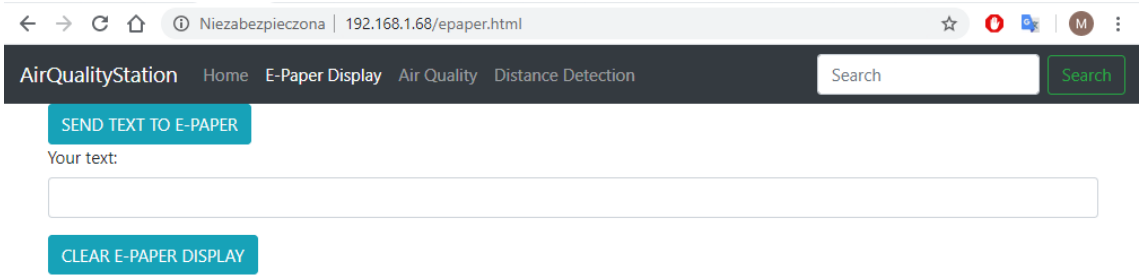
Na wersję przeglądarkową składają się cztery pliki html.

Główna strona – index.html, znajdują się na niej dwa przyciski służące do testowania komunikacji i pasek nawigacyjny, umożliwiający przełączenie się na inne strony (Rys. 6).



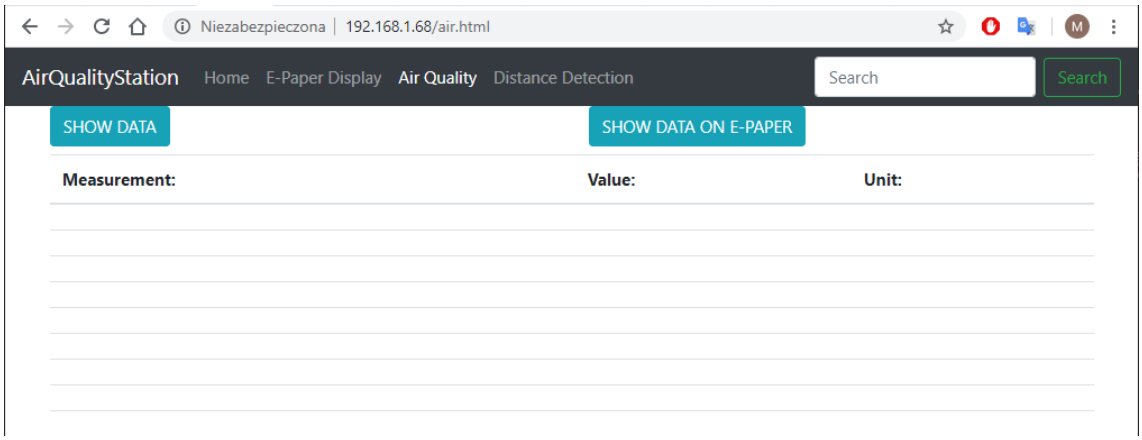
Rys. 6 Strona główna

Po przejściu do zakładki epaper.html możliwe jest wykonanie czyszczenia ekranu na Raspberry, lub wysłanie i wyświetlenie linijki swojego tekstu (Rys. 7).

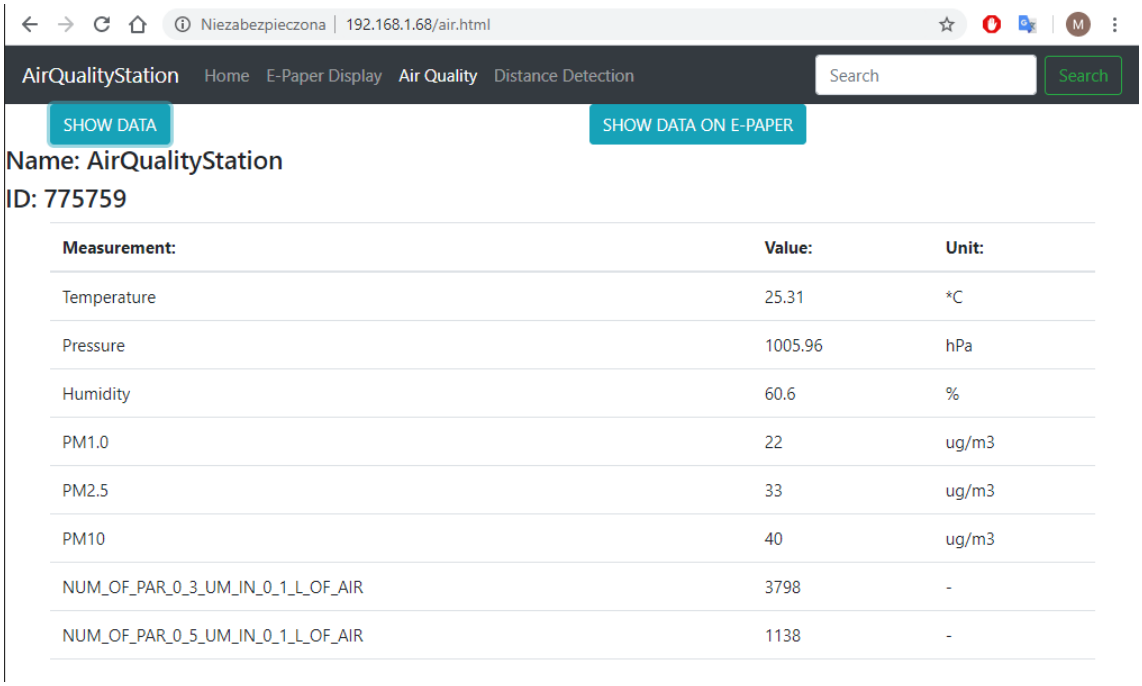


Rys. 7 Strona E-paper Display

Zakładka Air Quality umożliwia odczytanie danych pomiarowych ze stacji pomiarowej poprzez serwer ThingSpeak (Rys. 9), a także wyświetlanie tych danych na wyświetlaczu e-paper (Rys. 8).

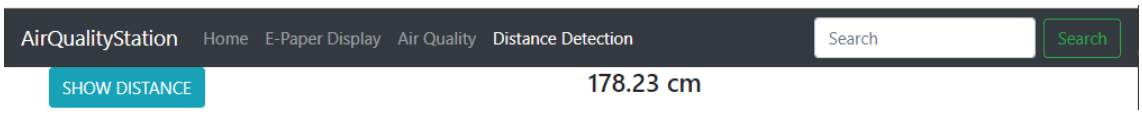


Rys. 8 Strona Air Quality



Rys. 9 Efekt działania przycisku SHOW DATA

Po naciśnięciu zakładki Distance Detection ukazuje się strona pozwalająca zmierzyć odległość z użyciem czujnika ultradźwiękowego (Rys. 10). Wynik jest wyświetlany zarówno w przeglądarce jak i na wyświetlaczu.



Rys. 10 Strona Distance Detection

3.2 CSS

Użyto biblioteki Bootstrap zapewniającej gotowe narzędzia do tworzenia interfejsu graficznego [5].

3.3 JS

Aby obsłużyć akcje wykonywane na stronach html, np. naciśnięcie przycisku, napisano skrypt „skrypt.js”. Zawiera on funkcje dla każdej obsługiwanej akcji. Funkcje te wywołują odpowiednie skrypty php z zadanymi argumentami, oraz jeśli trzeba przetwarzają odebrane dane z formatu JSON, tak aby wyświetlić je na stronie.

Interfejs użyty do połączenia z serwerem to fetch API (Listing 5) [6].

```
async function sendToEpaper(){
  const button_send_epaper = document.getElementById("b_send_epaper");
  const text_field = document.getElementById("text_field");
  button_send_epaper.innerHTML = "Sending...";
  console.log(text_field.value);
  const php_url = "send_text_epaper.php?text='" + text_field.value + "'";
  const php_script = await fetch(php_url);
  const php_log = await php_script.text();
  console.log(php_log);
  button_send_epaper.innerHTML = "SEND TEXT TO E-PAPER";
}
```

Listing 5 Obsługa naciśnięcia przycisku wysyłającego tekst na serwer

W skrypcie obsłużono także parsowanie danych pomiarowych pobranych z serwera Thingspeak (Listing 6) i umieszczenie ich w odpowiednich miejscach na stronie html (Listing 7).

```
async function showData(){
  const button_show_data = document.getElementById("b_show_data");
  const data = await fetch("https://api.thingspeak.com/channels/775759/feeds
.json?results=1");
  const data_log = await data.text();
  // console.log(data_log);

  var data_json = JSON.parse(data_log);
  console.log(data_json);
```

Listing 6 Pobieranie danych z Thingspeak i parsowanie

```
const name = document.getElementById("station_name");
const id = document.getElementById("station_id");
name.innerHTML = "Name: " + data_json.channel.name
id.innerHTML = "ID: " + data_json.channel.id
```

Listing 7 Umieszczenie danych w odpowiednich komórkach

3.4 PHP

Każdemu skryptowi Pythonowemu przypisany jest dedykowany skrypt PHP służący do jego wykonania (Listing 8). Skrypty PHP przekazują także argumenty do skryptów Pythona. Jeśli skrypt Pythonowy zwraca lub wypisuje jakąś wartość, jest ona także wypisywana przez skrypt PHP.

```
<?php
    if(isset($_GET["text"]))
    {
        $text = $_GET["text"];
    }
    $command = "sudo /usr/bin/python3 /var/www/html/send_text_epaper.py ".$t
ext;
    echo shell_exec($command);
?>
```

Listing 8 Przykładowy skrypt PHP, służący do wywoływania skryptu wypisującego wysłany w argumencie tekst na wyświetlaczu

Podczas używania skryptów PHP należało ustawić odpowiednie uprawnienia grup, tak aby użytkownik www-data mógł wykonywać polecenia w danym katalogu.

4 APLIKACJA ANDROID

Aplikację mobilną na system Android podzielono na cztery aktywności. Do każdego widoku aktywności wprowadzono przycisk HOME umożliwiający powrót na ekran główny aplikacji z użyciem Intent (Listing 9).

```
1.  public void goToMain(View view){
2.      Intent myIntent = new Intent(DistanceActivity.this, MainActivity.class);

3.      DistanceActivity.this.startActivity(myIntent);
4.  }
```

Listing 9 Fragment kodu powrotu do głównego ekranu apliakcji

Aby aplikacja posiadała możliwość połączenia się z siecią należało ustawić odpowiednie zezwolenia w pliku AndroidManifest.xml (Rys. 11).

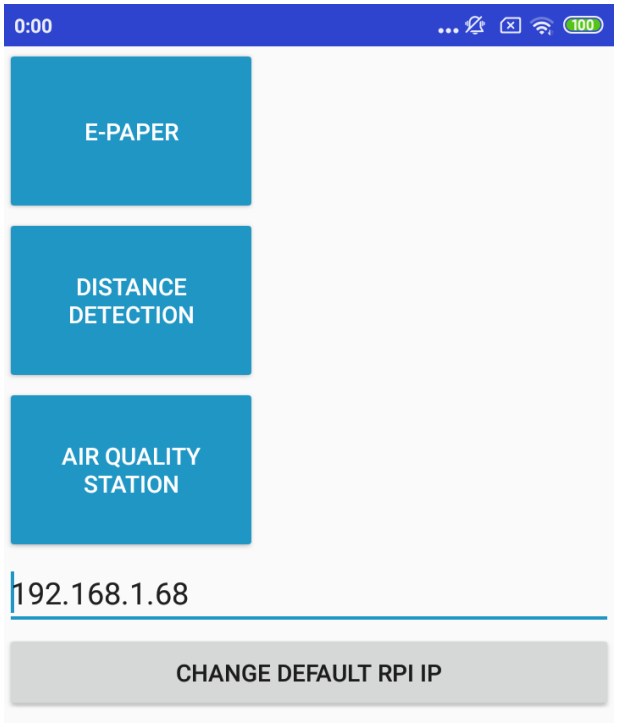
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.airqualitystation_iot">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Rys. 11 Zezwolenia na połączenie z siecią

Aplikację tworzono i testowano jedynie na telefonie Redmi 4X (5.00’’).

4.1 AKTYWNOŚĆ MAINACTIVITY

Główna aktywność umożliwia przejście do innych aktywności, a także zmianę adresu IP serwera z którym następuje połączenie (Rys. 12).



Rys. 12 Ekran startowy

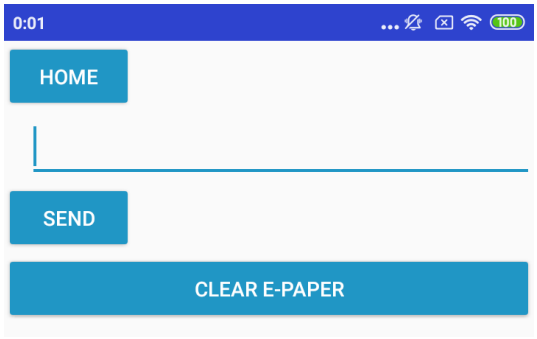
Aby zmiana adresu IP była dostępna dla wszystkich innych aktywności w oddzielnym pliku stworzono klasę Globals (Listing 10 Kod klasy Globals). Przechowuje ona zmienną zawierającą IP, umożliwia jej pobranie i zmianę dzięki metodom setIP() i getIP(). Przykład użycia to linijki 5, 6 i 7 w (Listing 11).

```
1.  public class Globals{
2.      private static Globals instance;
3.
4.      // Global variable
5.      private String data = "192.168.1.68";
6.
7.      // Restrict the constructor from being instantiated
8.      private Globals(){}
9.
10.     public void setIP(String d){
11.         this.data=d;
12.     }
13.     public String getIP(){
14.         return this.data;
15.     }
16.
17.     public static synchronized Globals getInstance(){
18.         if(instance==null){
19.             instance=new Globals();
20.         }
21.         return instance;
22.     }
23. }
```

Listing 10 Kod klasy Globals

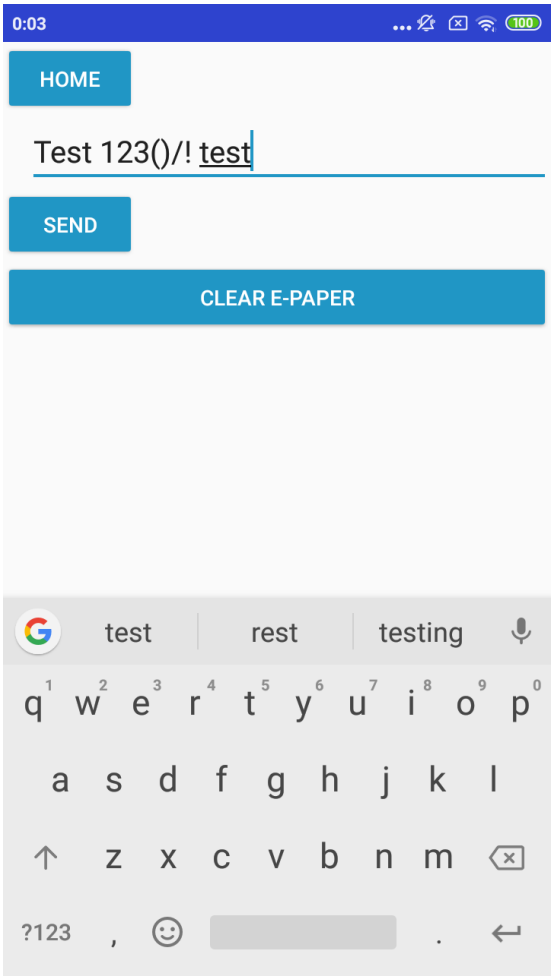
4.2 AKTYWNOŚĆ EPAPERACTIVITY

Po naciśnięciu przycisku E-PAPER na ekranie głównym, użytkownik zostaje przeniesiony do aktywności obsługującej wysyłanie własnego tekstu i czyszczenie wyświetlacza e-paper (Rys. 13).



Rys. 13 Ekran aktywności E-Paper

Wprowadzony w pole tekstowe tekst (Rys. 14) jest przesyłany do skryptu PHP, który z kolei wywołuje skrypt Pythonowy wyświetlający ten tekst na wyświetlaczu (Listing 11).



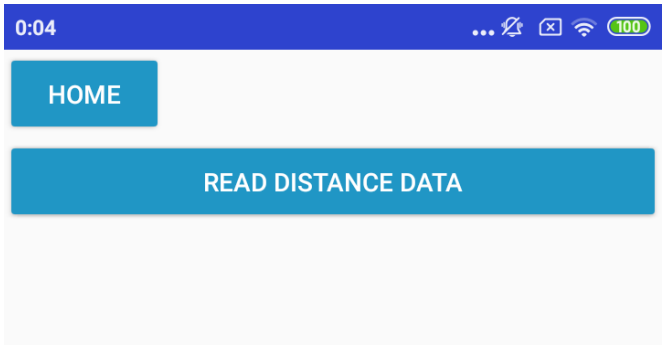
Rys. 14 Wprowadzanie tekstu do wysłania na wyświetlacz podłączony do Raspberry

```
1.  public void sendToEpaper(View view){
2.      final EditText editText = (EditText) findViewById(R.id.editText);
3.      RequestQueue queue = Volley.newRequestQueue(this);
4.
5.      Globals g = Globals.getInstance();
6.      String rpi_ip = g.getIP();
7.      String url ="http://"+rpi_ip+"/send_text_epaper.php?text='" + editText.g
    etText() + "'";
8.
9.      StringRequest stringRequest = new StringRequest(Request.Method.POST, url
    , new Response.Listener<String>() {
10.         @Override
11.         public void onResponse(String response) {
12.         }
13.     }, new Response.ErrorListener() {
14.         @Override
15.         public void onErrorResponse(VolleyError error) {
16.         }
17.     });
18.     stringRequest.setRetryPolicy(new DefaultRetryPolicy(100000, DefaultRetry
    Policy.DEFAULT_MAX_RETRIES, DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
19.     queue.add(stringRequest);
20. }
```

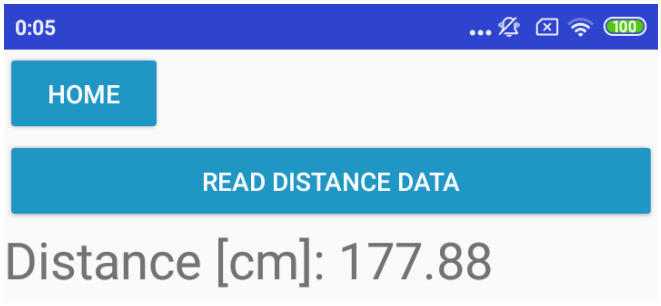
Listing 11 Wysyłanie tesktu do Raspberry

4.3 AKTYWNOŚĆ DISTANCEACTIVITY

Aktywność DistanceActivity zawiera jedynie przycisk pozwalający odczytać z czujnika odległości pojedynczy pomiar zwracany w centymetrach (Rys. 15 i Rys. 16).



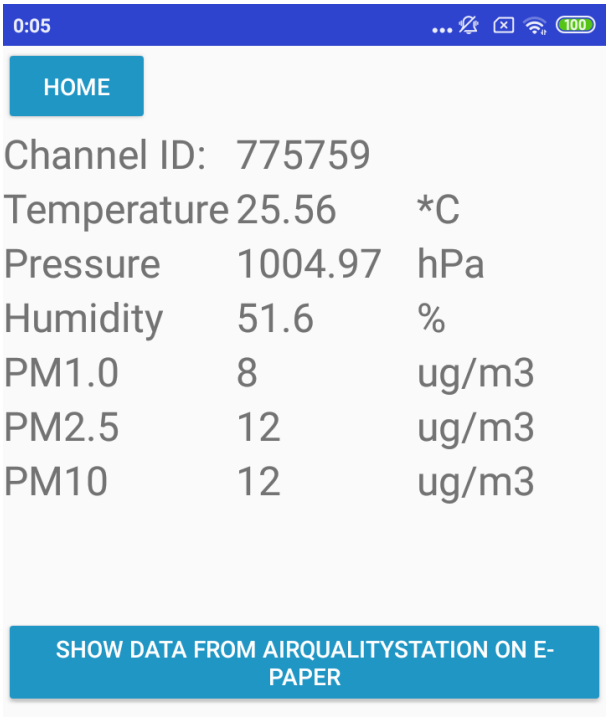
Rys. 15 Ekran aktywności DistanceActivity



Rys. 16 Zwrócona wartość odległości

4.4 AKTYWNOŚĆ STATIONACTIVITY

Aktywność ta po uruchomieniu wyświetla dane pobrane z serwera ThingSpeak, a także umożliwia wyświetlenie ich na wyświetlaczu podłączonym do serwera Raspberry (Rys. 17).



Rys. 17 Ekran aktywnośći StationActivity

Aby dane pobierane były bez ingerencji użytkownika zastosowano Timer pobierający je co pięć sekund gdy aktywność jest wyświetlana (Listing 12). Do metody onCreate() dodano start timera (Listing 14), natomiast po użyciu przycisku HOME timer jest zatrzymywany, aby nie działał w tle kiedy nie jest to potrzebne. Timer wywołuje metodę getThingSpeakFields(), która odbiera i przetwarza dane z formatu JSON na tekst wyświetlany w odpowiednich polach tabeli (Listing 13).

```
1.  private Timer timer;
2.  private TimerTask timerTask = new TimerTask() {
3.      @Override
4.      public void run() {
5.          getThingSpeakFields(getCurrentFocus());
6.      }
7.  };
8.
9.  public void timerStart() {
10.     if(timer != null) {
11.         return;
12.     }
13.     timer = new Timer();
14.     timer.scheduleAtFixedRate(timerTask, 0, 5000);
15. }
16.
17. public void timerStop() {
18.     timer.cancel();
19.     timer = null;
20. }
```

Listing 12 Timer wywołujący metodę getThingSpeakFields() co 5 sekund

```
1.  RequestQueue queue = Volley.newRequestQueue(this);
2.  String url = "https://api.thingspeak.com/channels/775759/feeds.json?results=
    1";
3.  StringRequest stringRequest = new StringRequest(Request.Method.GET, url, new
     Response.Listener<String>() {
4.      @Override
5.      public void onResponse(String response) {
6.          //textView.setText("Response:" + response);
7.          try {
8.              JSONObject reader = new JSONObject(response);
9.
10.             JSONArray feeds = reader.getJSONArray("feeds");
11.             JSONObject feed = feeds.getJSONObject(0);
12.             long entry_id = feed.getInt("entry_id");
13.             String entry_created_at = feed.getString("created_at");
14.             String field1 = feed.getString("field1");
15.             String field2 = feed.getString("field2");
```

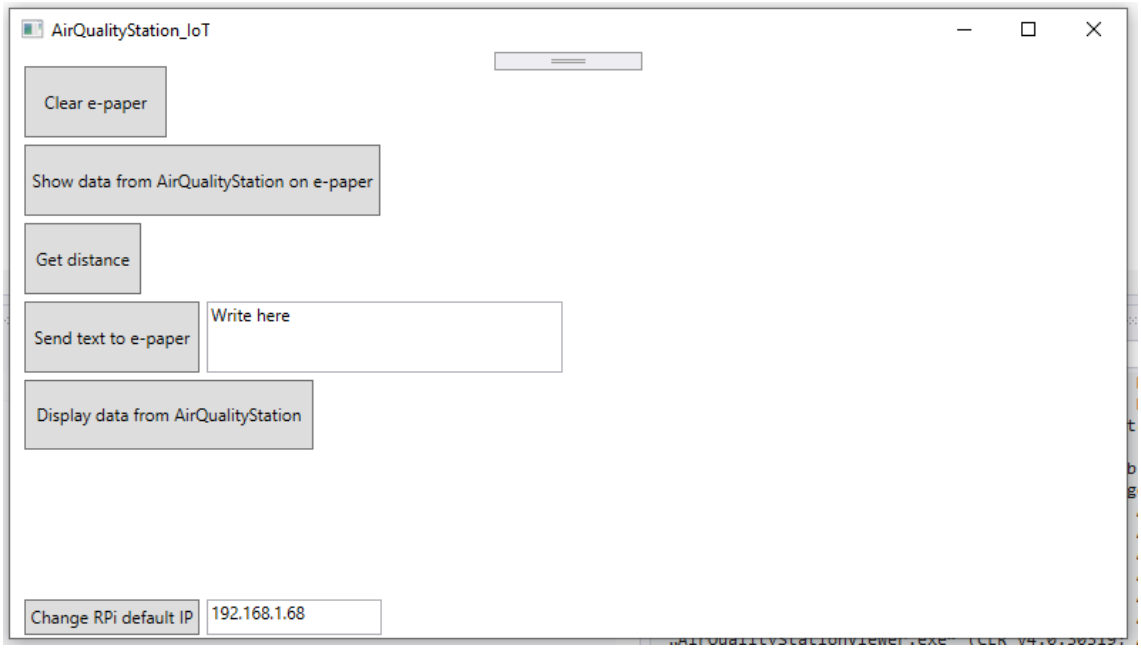
Listing 13 Fragment kodu metody getThingSpeakFields() pobierający odpowiedź serwera i przetwarzający kilka z jej pól danych

```
1.  @Override
2.  protected void onCreate(Bundle savedInstanceState) {
3.      super.onCreate(savedInstanceState);
4.      setContentView(R.layout.activity_station);
5.      timerStart();
6.  }
```

Listing 14 Do metody onCreate() dodano start timera

5 APLIKACJA C#

Aplikacja posiada prosty interfejs który zawiera wszystkie jej funkcje (Rys. 18).

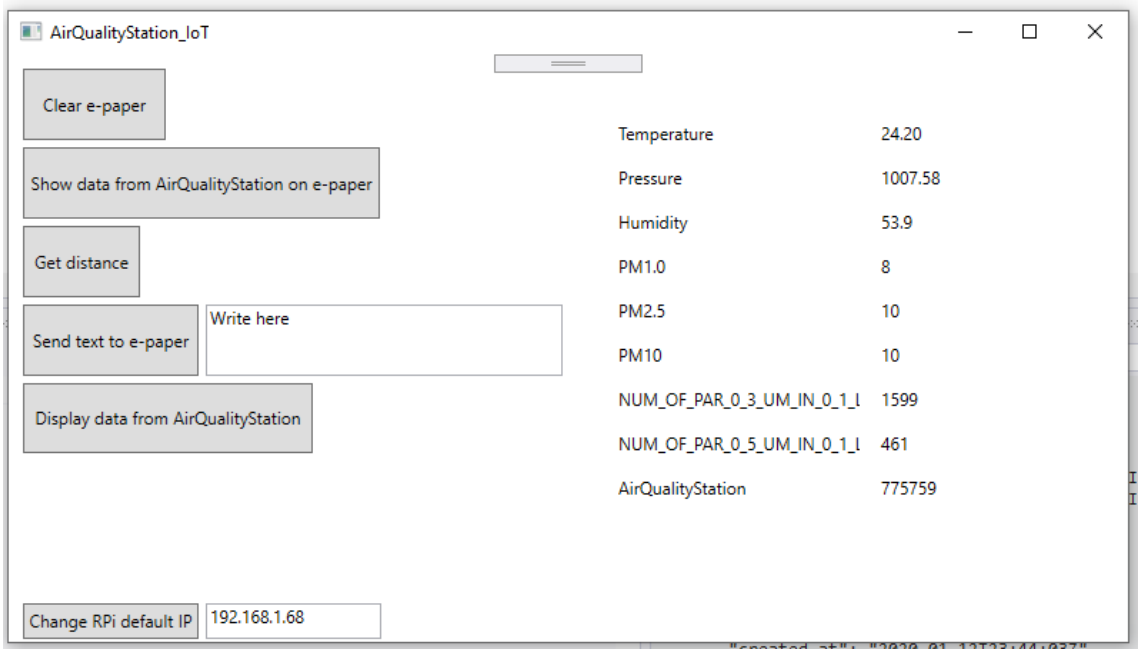


Rys. 18 Interfejs aplikacji C#

Funkcje działają analogicznie do aplikacji na system Android. Możliwa jest zmiana adresu serwera dzięki użyciu zmiennej globalnej zawierającej IP (Listing 15). Po naciśnięciu przycisku „Display data from AirQualityStation” aplikacja wyświetla przetworzone dane pomiarowe (Rys. 19) (Listing 16).

```
1.  private string rpi_ip = "192.168.1.68";
2.  public MainWindow()
3.  {
4.      InitializeComponent();
5.      textbox_ip.Text = rpi_ip;
6.  }
```

Listing 15 Zmienna globalna przechowująca IP serwera



Rys. 19 Wyświetlenie danych z serwera ThingSpeak

```
1.  private void Button_show_airqualitystation_Click(object sender, RoutedEventA
    rgs e)
2.          {
3.              string sURL = "https://api.thingspeak.com/channels/775759/feeds.j
    son?results=1";
4.              var json = new WebClient().DownloadString(sURL);
5.              dynamic stuff = JsonConvert.DeserializeObject(json);
6.              l1.Content = stuff.channel.field1;
7.              l2.Content = stuff.channel.field2;
```

Listing 16 Odbieranie i deserializacja danych z serwera ThingSpeak

LITERATURA

[1] „https://websiteforstudents.com/setup-lighttpd-web-server-with-php-supports-on-ubuntu-servers/?fbclid=IwAR39mmekrA-dtbb6xCxOnG3keShPS_AMd7FkssUdjD8YUwRoAJZlbWFRmrk,” [Online].

[2] „https://www.waveshare.com/wiki/2.13inch_e-Paper_HAT,” [Online].

[3] „https://mikrokontroler.pl/2017/05/24/projekt-czujnik-odleglosci-hc-sr04-i-arduino/,” [Online].

[4] „https://thingspeak.com/channels/775759,” [Online].

[5] „https://getbootstrap.com/,” [Online].

[6] „https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch,” [Online].

[7] D. Łuczak, „Szablon sprawozdania [online]. 2014. s.l.: s.n. Pobrano: http://zsep.cie.put.poznan.pl/.,” [Online].