POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI

INSTYTUT ROBOTYKI I INTELIGENCJI MASZYNOWEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

Zadanie zaliczeniowe
System IoT

APLIKACJE MOBILNE I WBUDOWANE DLA INTERNETU PRZEDMIOTÓW

MACIEJ PADERECKI
JAKUB CODOGNI
KRZYSZTOF BOROWSKI

PROWADZĄCY:
MGR INŻ. ADRIAN WÓJCIK

POZNAŃ 28.09.2020

1. OPIS SPECYFIKACJI

- Serwer WWW umożliwia pobieranie danych pomiarowych ze wszystkich dostępnych w układzie wbudowanym czujników oraz wysyłanie danych sterujących do wszystkich elementów wykonawczych
- Serwer WWW umożliwia przesyłanie danych pomiarowych do wszystkich trzech aplikacji klienckich oraz odbiera dane sterujące od wszystkich trzech aplikacji klienckich.
- Wszystkie trzy aplikacje klienckie umożliwiają podgląd danych pomiarowych za pomocą dynamicznie generowanego interfejsu użytkownika (np. w formie listy lub tabeli).
- Wszystkie trzy aplikacje klienckie umożliwiają podgląd danych pomiarowych za pomocą wykresu przebiegu czasowego.
- Wszystkie trzy aplikacje klienckie umożliwiają próbkowanie danych pomiarowych z okresem nie większym niż 1000 ms.
- GUI wszystkich trzech aplikacji klienckich zawiera informacje o jednostkach wielkości pomiarowych.
- Wszystkie trzy aplikacje klienckie umożliwia sterowanie pojedynczymi elementami wykonawczymi w pełnym dostępnym zakresie kontroli.
- Wszystkie trzy aplikacje klienckie umożliwiają konfiguracje komunikacji sieciowej (adres i port serwera) oraz akwizycji danych (okres próbkowania i maksymalna zapisywana liczba punktów pomiarowych).

2. IMPLEMENTACJA SYSTEMU – APLIKACJE SERWERA

Na serwerze znajdują się trzy skrypty służące do komunikacji z aplikacjami użytkownika.

2.1.1 Plik *server_script.py*

Plik server_script.py zawiera w sobie pętlę while, która co 100ms odczytuje wartości pobierane z emulatora nakładki SenseHat i zapisuje je do plików .json. Dane z nakładki przetwarzają klasy EnvData z listingu 1. Listing 2 zawiera funkcję zapisującą dane do pliku

```
1. class EnvData :
2.         def __init__ (self , roll , pitch, yaw, temp, press, humi,
   x, y, z):
3.                 self.roll = roll
4.                 self.pitch = pitch
5.                 self.yaw = yaw
6.                 self.x = x
7.                 self.y = y
8.                 self.z = z
9.                 self.temp = temp
10.                  self.press = press
11.                  self.humi = humi
```

Listing 1 Klasa zawierająca dane z nakładki Sense HAT

```
def save_data():

    with open('dataCS.json', 'w+') as outfile:
        obj_data = EnvData(roll , pitch, yaw, temp, press, humi, x, y, z)
        result = json.dumps(obj_data.__dict__)
        outfile.write(result)
```

Listing 2 Funkcja zapisująca dane do pliku .json

Na listingu 3 pokazana została pętla while. Powtarza się ona co 100ms. Program odczytuje po kolei pozycję joysticka, wartości temperatury, ciśnienia, wilgotności oraz orientacji i zapisuje je do plików .json.

```
while True:

    sense.stick.direction_up = pushed_up
    sense.stick.direction_down = pushed_down
    sense.stick.direction_left = pushed_left
    sense.stick.direction_right = pushed_right
    sense.stick.direction_middle = pushed_middle
    sense.stick.direction_any = save_data


     temp = sense.get_temperature()
     press = sense.get_pressure()
     humi = sense.get_humidity()


     orientation_degrees = sense.get_orientation_degrees()

     roll=orientation_degrees["roll"]
     pitch=orientation_degrees["pitch"]
     yaw=orientation_degrees["yaw"]

     save_arrayCS()
     save_arrayAndroid()
     save_joy()
     save_tph()
     save_rpy()
     save_data()

     time.sleep(0.1)
```

Listing 3 Pętla While

2.1.2 Plik *setled.py*

Na listingu 4. przedstawiono pełny program setled.py. Program odpowiada za zapalanie odpowiednich ledów na emulatorze senseHAT. Plik wykonuje operacje za pomocą danych zawartych w leddata.json. Plik zawiera wektor wektorów. Wewnętrzy wektor zapisany jest w sekwencji [y, x, red, green, blue].

```
1. import json
2. from sense_emu import SenseHat
3. print('IN')
4.
5. sense = SenseHat()
6.
7. filename = "leddata.json";
8.
9. if filename:
10.      with open(filename, 'r') as f:
11.          ledDisplayArray=json.load(f);
12.
13.  for led in ledDisplayArray:
14.      # schemat led: y x R G B
15.      sense.set_pixel(led[1], led[0], led[2], led[3], led[4]);
```

Listing 4 Skrypt setled.py

2.1.3 Plik *setled*.php

Listing 5. przedstawia pełny skrypt setled.php. Program otrzymuje z programów użytkownika dane metodą POST i zapisuje je do pliku .json. Następnie uruchamia on skrypt setled.py.

```
1. #!/usr/bin/php
2. <?php
3.   ini_set('display_errors',1);
4.   error_reporting(E_ALL);
5.
6.
7.   function ledIndexToTagConverter($x, $y){
8.           return "LED" .$x .$y;
9.   }
10.
11.          $ledDisplay = array();
12.          $ledDisplayDataFile = 'leddata.json';
13.
14.          $n=0;
15.
16.          for ($i=0; $i<8; $i++){
17.                  for ($j=0; $j<8; $j++){
```

```
18.                         $ledTag=ledIndexToTagConverter($i, $j);
19.                         if(isset($_POST[$ledTag])){
20.                                 $ledDisplay[$n] =
    json_decode($_POST[$ledTag]);
21.                                 $n=$n+1;
22.                         }
23.                 }
24.         }
25.
26.         $ledDisplayJson=json_encode($ledDisplay);
27.         $dataFile = fopen($ledDisplayDataFile, 'w+') or
    die("ERR1");
28.         fwrite($dataFile, $ledDisplayJson);
29.         fclose($dataFile);
30.
31.         echo "ACK1 ";
32.         $command = escapeshellcmd('sudo python setled.py');
33.         $output = shell_exec($command);
34.         echo $output;
35.         echo "ACK2 ";
36.  ?>
```

Listing 5 Skrypt setled.php

2.2 IMPLEMENTACJA SYSTEMU – APLIKACJA DESKTOPOWA

2.2.1 STRUKTURA APLIKACJI

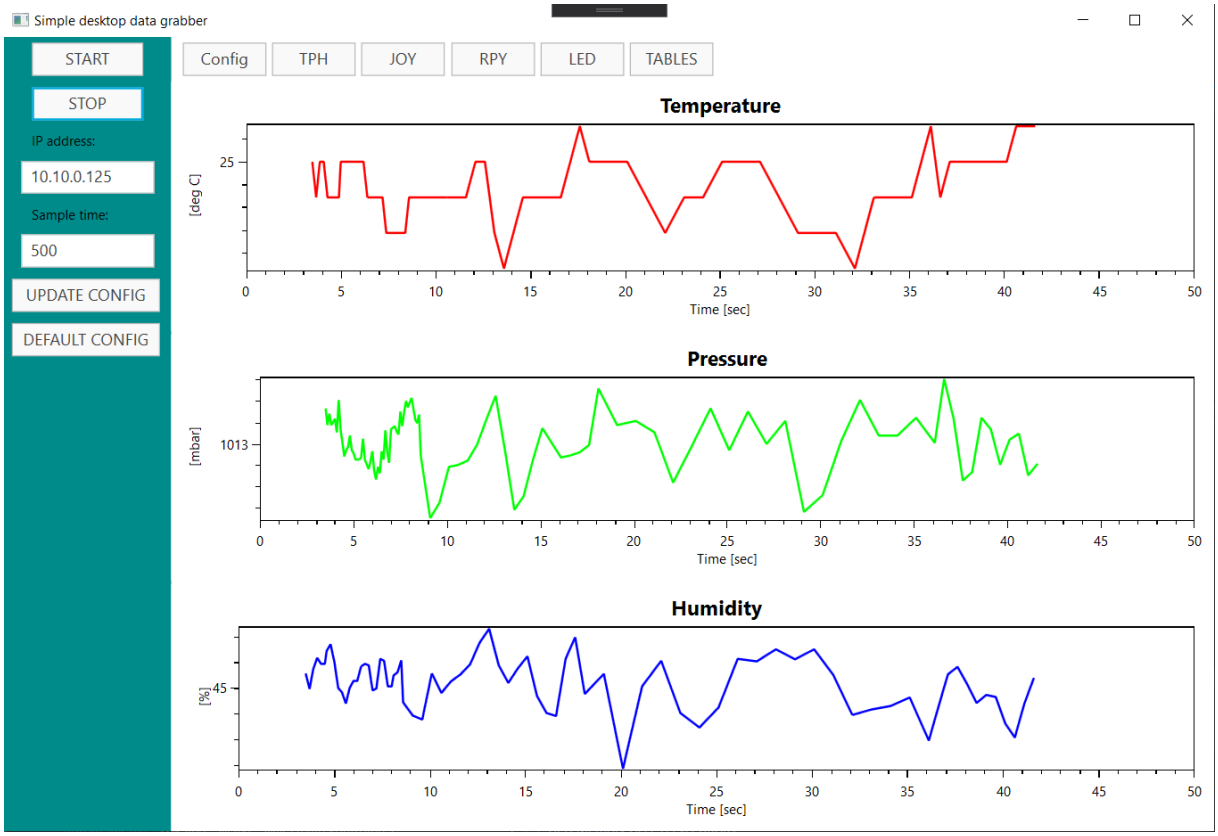
Struktura aplikacji opiera się o model MVVM. Wszystkie funkcjonalności zostały zaimplementowane w jednym oknie, a przełączanie między nimi możliwe jest dzięki użyciu atrybutu *Visibility* - włączenie jednego widoku powoduje ukrycie pozostałych. Przykład takiej akcji zawartej w pliku xaml.cs pokazany został na listingu 6.

```
1.         private void LEDBtn_Click(object sender, RoutedEventArgs
    e)
2.         {
3.
4.             this.DataPlotView_temp.Visibility =
    Visibility.Collapsed;
5.             this.DataPlotView_press.Visibility =
    Visibility.Collapsed;
6.             this.DataPlotView_humi.Visibility =
    Visibility.Collapsed;
7.             this.Data_table.Visibility = Visibility.Collapsed;
8.             this.RPYPlotView_roll.Visibility =
    Visibility.Collapsed;
9.             this.RPYPlotView_pitch.Visibility =
    Visibility.Collapsed;
10.              this.RPYPlotView_yaw.Visibility =
```

```
11.  Visibility.Collapsed;
12.              this.JOYPlotView.Visibility = Visibility.Collapsed;
13.
14.              if (this.LED1.Visibility == Visibility.Visible &&
    this.LED2.Visibility == Visibility.Visible)
15.              {
16.                  this.LED1.Visibility = Visibility.Collapsed;
17.                  this.LED2.Visibility = Visibility.Collapsed;
18.              }
19.              else
20.              {
21.                  this.LED1.Visibility = Visibility.Visible;
22.                  this.LED2.Visibility = Visibility.Visible;
23.              }
24.
25.          }
```

Listing 6 Przykład zmiany parametru Visibility

Interfejs użytkownika aplikacji desktopowej pokazany jest na Rysunku 1.



Rysunek 1 Interfejs użytkownika aplikacji

2.2.2 WYKRESY TPH

Wykresy temperatury, ciśnienia i wilgotności zostały wykonane w oparciu o bibliotekę OxyPlot. Na listingu 7 przedstawiono inicjalizację jednego z wykresów. Na listingu 8 znajduje się funkcja aktualizująca wykres. Rysunek 2. pokazuje interfejs i działanie wykresów tph.

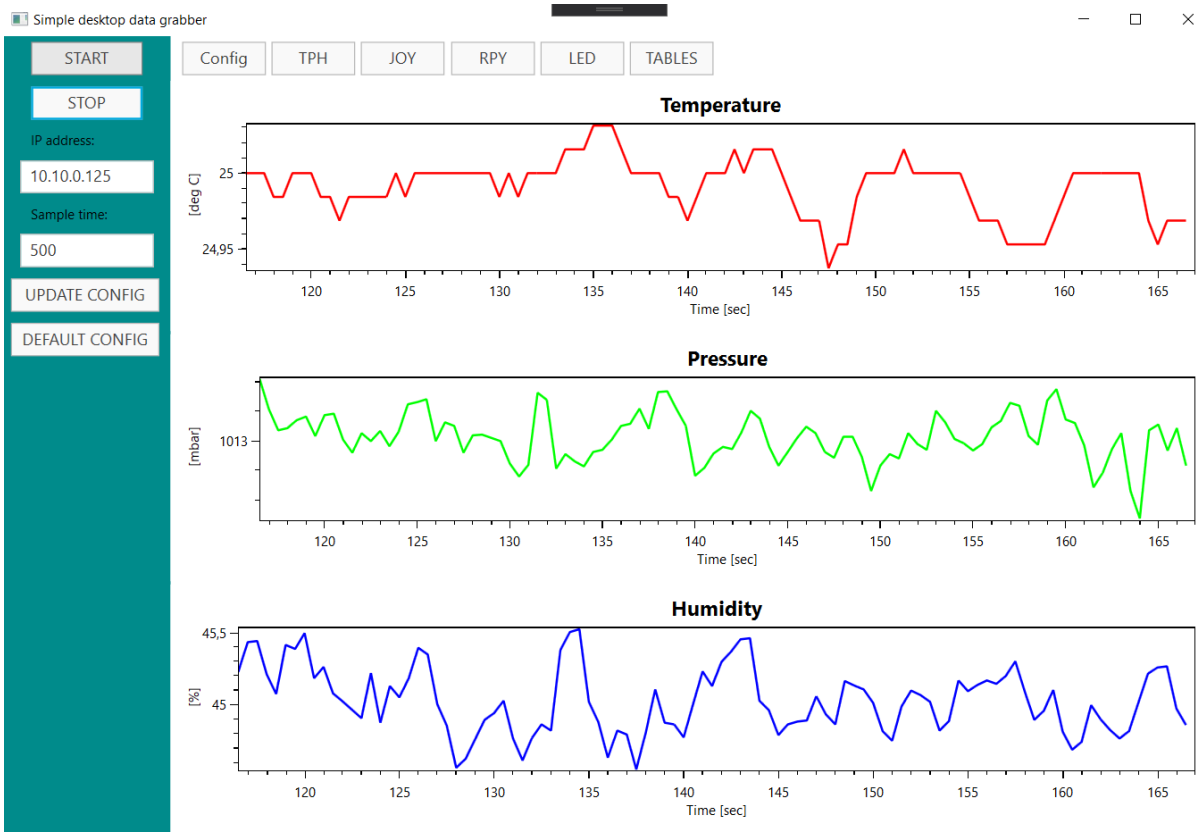
```
1. DataPlotModel = new PlotModel { Title = "Temperature" };
2.
3.             DataPlotModel.Axes.Add(new LinearAxis()
4.             {
5.                 Position = AxisPosition.Bottom,
6.                 Minimum = 0,
7.                 Maximum = config.XAxisMax,
8.                 Key = "Horizontal",
9.                 Unit = "sec",
10.                  Title = "Time"
11.             });
12.             DataPlotModel.Axes.Add(new LinearAxis()
13.             {
14.                 Position = AxisPosition.Left,
15.                 Key = "Vertical",
16.                 Unit = "deg C",
17.             });
18.
19.             DataPlotModel.Series.Add(new LineSeries() {Color =
    OxyColor.Parse("255,0,0") });
```

Listing 7 Tworzenie wykresu temperatury

```
1.         private void UpdatePlot(double t, double d)
2.         {
3.             LineSeries lineSeries = DataPlotModel.Series[0] as
    LineSeries;
4.
5.             lineSeries.Points.Add(new DataPoint(t, d));
6.
7.             if (lineSeries.Points.Count > config.MaxSampleNumber)
8.                 lineSeries.Points.RemoveAt(0);
9.
10.             if (t >= config.XAxisMax)
11.             {
12.                 DataPlotModel.Axes[0].Minimum = (t -
    config.XAxisMax);
13.                 DataPlotModel.Axes[0].Maximum = t +
    config.SampleTime / 1000.0; ;
```

```
14.                 }
15.
16.                 DataPlotModel.InvalidatePlot(true);
17.             }
```

Listing 8 Funkcja aktualizująca wykres



Rysunek 2 Interfejs użytkownika i działanie wykresów tph

2.2.3 WYKRESY RPY

Wykresy orientacji roll, pitch, yaw zostały wykonane analogicznie do wykresów tph z punktu poprzedniego. Na listingu 9 przedstawiono inicjalizację jednego z wykresów. Na listingu 10 znajduje się funkcja aktualizująca wykres. Rysunek 3. pokazuje interfejs i działanie wykresów rpy.

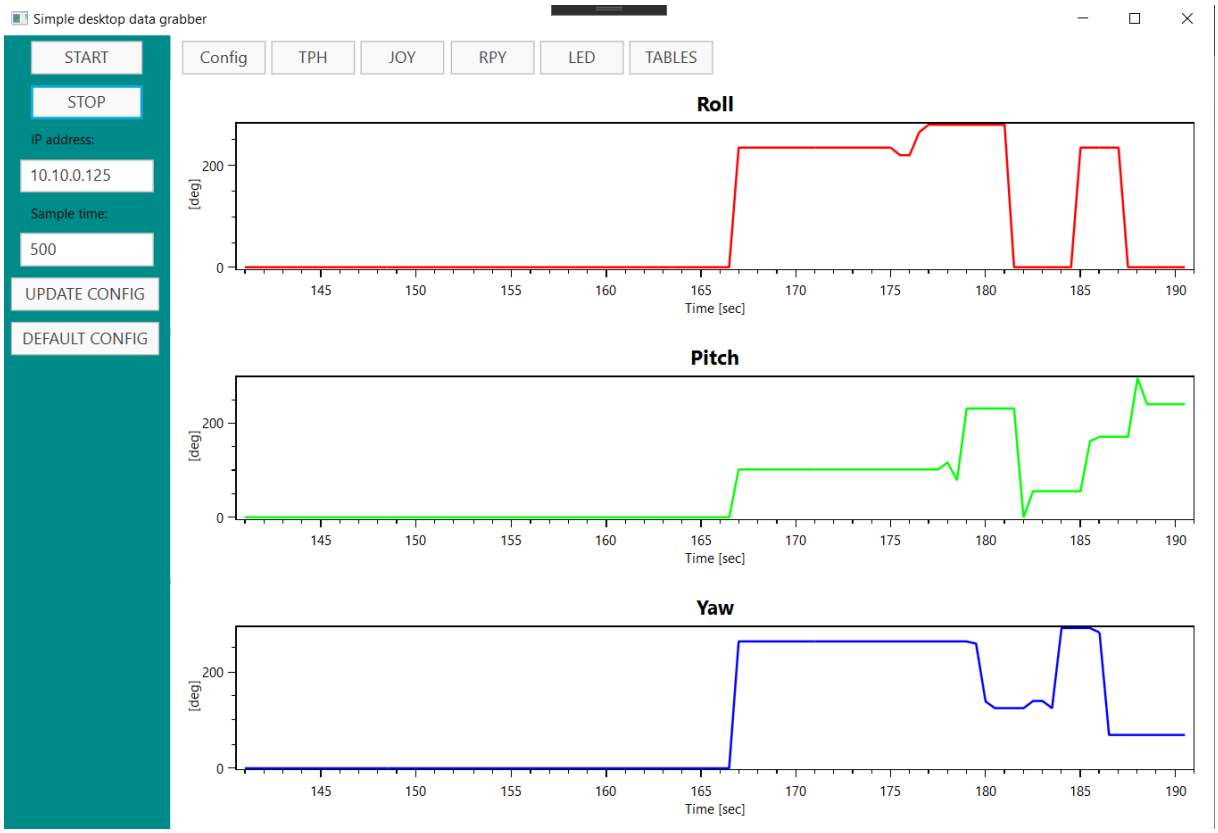
```
1. RPYPlotModel = new PlotModel { Title = "Roll" };
2.
3.             RPYPlotModel.Axes.Add(new LinearAxis()
4.             {
5.                 Position = AxisPosition.Bottom,
6.                 Minimum = 0,
7.                 Maximum = config.XAxisMax,
8.                 Key = "Horizontal",
9.                 Unit = "sec",
10.                  Title = "Time"
```

```
11.             });
12.             RPYPlotModel.Axes.Add(new LinearAxis()
13.             {
14.                 Position = AxisPosition.Left,
15.                 Key = "Vertical",
16.                 Unit = "deg"
17.             });
18.             RPYPlotModel.Series.Add(new LineSeries() {Color =
    OxyColor.Parse("255,0,0")});
```

Listing 9 Tworzenie wykresu orientacji Roll

```
1.         private void UpdatePlot4(double t, double d)
2.         {
3.             LineSeries lineSeries = RPYPlotModel.Series[0] as
    LineSeries;
4.
5.             lineSeries.Points.Add(new DataPoint(t, d));
6.
7.             if (lineSeries.Points.Count > config.MaxSampleNumber)
8.                 lineSeries.Points.RemoveAt(0);
9.
10.              if (t >= config.XAxisMax)
11.              {
12.                  RPYPlotModel.Axes[0].Minimum = (t -
    config.XAxisMax);
13.                  RPYPlotModel.Axes[0].Maximum = t +
    config.SampleTime / 1000.0; ;
14.              }
15.
16.              RPYPlotModel.InvalidatePlot(true);
17.          }
```

Listing 10 Funkcja aktualizująca wykres



Rysunek 3 Interfejs użytkownika i działanie wykresów tph

2.2.4 WYKRES JOYSTICK

Wykres reprezentujący joystick został wykonany analogicznie do wykresów rpy z punktu poprzedniego. Różni się on jednak sposobem aktualizacji wykresu z uwagi na tylko jeden punkt reprezentujący dane. Na listingu 11 przedstawiono inicjalizację wykresu. Na listingu 12 znajduje się funkcja aktualizująca wykres. Rysunek 4. pokazuje interfejs i działanie wykresu joysticka.

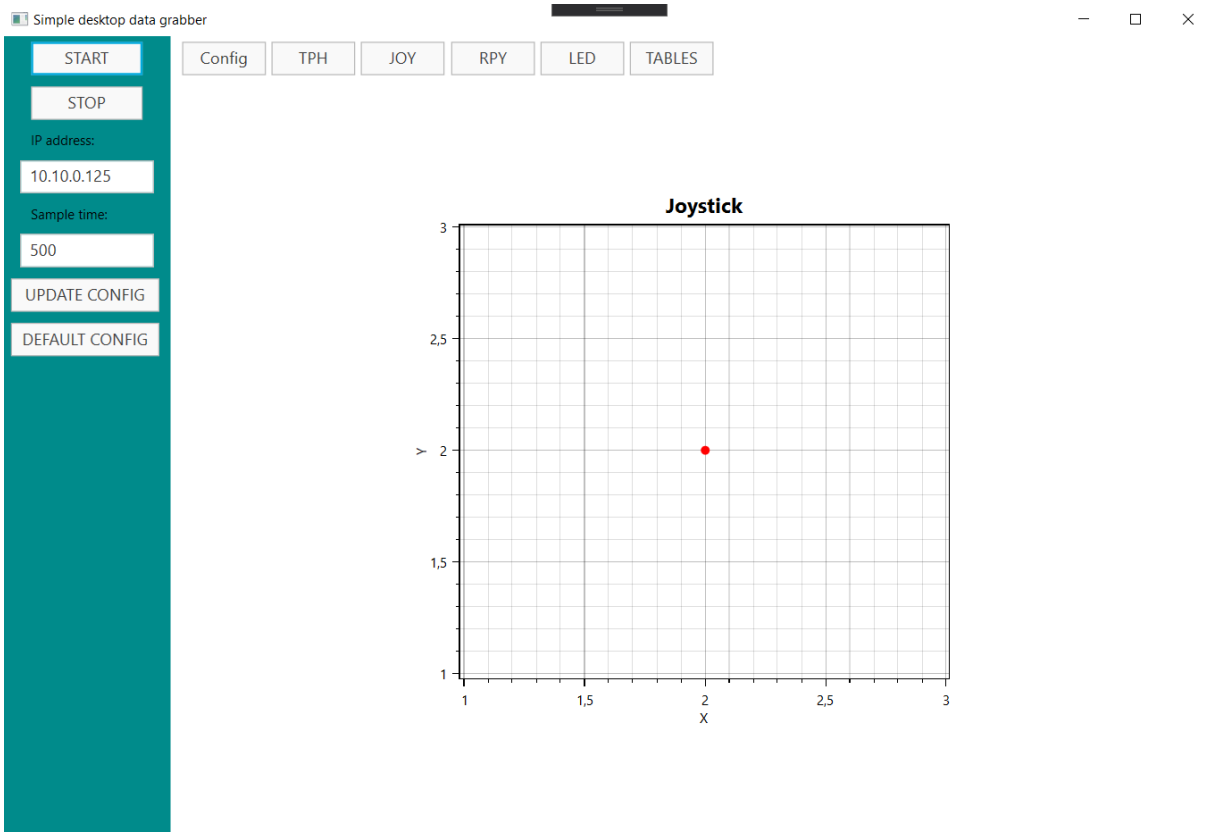
```
1.  JOYPlotModel = new PlotModel { Title = "Joystick" };
2.
3.              JOYPlotModel.Axes.Add(new LinearAxis()
4.              {
5.                  Position = AxisPosition.Bottom,
6.                  MajorGridlineStyle = OxyPlot.LineStyle.Solid,
7.                  MinorGridlineStyle = OxyPlot.LineStyle.Solid,
8.                  Key = "Horizontal",
9.                  Title = "X"
10.               });
11.               JOYPlotModel.Axes.Add(new LinearAxis()
12.               {
13.                   Position = AxisPosition.Left,
14.                   MajorGridlineStyle = OxyPlot.LineStyle.Solid,
15.                   MinorGridlineStyle = OxyPlot.LineStyle.Solid,
16.                   Key = "Vertical",
```

```
17.                     Title = "Y"
18.                 });
19.  JOYPlotModel.Series.Add(new LineSeries() { Color =
   OxyColor.Parse("#FFFF0000"), MarkerType =
   OxyPlot.MarkerType.Circle,MarkerSize = 4});
```

Listing 11 Tworzenie wykresu orientacji Joystick

```
1.         private void UpdatePlotJoy(double horizontal, double
   vertical)
2.         {
3.             LineSeries lineSeries = JOYPlotModel.Series[0] as
   LineSeries;
4.
5.             lineSeries.Points.Add(new DataPoint(horizontal,
   vertical));
6.
7.             if (lineSeries.Points.Count > 1)
8.             lineSeries.Points.RemoveAt(0);
9.
10.
11.              JOYPlotModel.InvalidatePlot(true);
12.         }
```

Listing 12 Funkcja aktualizująca wykres



Rysunek 4 Interfejs użytkownika i działanie wykresu joystick

2.2.5 STEROWANIE LED

Do sterowania matrycą LED służą dwie funkcje przedstawione na listingu 13. Pierwsza z nich „zapala” diody, a druga je „gasi”. Obie korzystają z zapytania POST co widać na listingu 14.

```
1.    private async void Send_Led_Data()
2.         {
3.             string R_s = R.ToString();
4.             string G_s = G.ToString();
5.             string B_s = B.ToString();
6.             await Server.POSTwithClient_send_led(X, Y, R_s, G_s,
   B_s);
7.         }
8.
9.         private async void Send_Led_Clear()
10.          {
11.
12.              int iteration = 0;
13.              string[] led_val_string = new string[64];
14.              string[] x_val_string = new string[64];
15.              string[] y_val_string = new string[64];
16.              string[] rgb_clear_val_string = new string[64];
17.
18.              for (int i = 0; i < 8; i++)
19.              {
20.                  for (int u = 0; u < 8; u++)
21.                  {
22.                      led_val_string[iteration] = i.ToString() +
   u.ToString();
23.                      x_val_string[iteration] = i.ToString();
24.                      y_val_string[iteration] = u.ToString();
25.                      rgb_clear_val_string[iteration] = "0";
26.                      iteration++;
27.                  }
28.              }
29.              await
   Server.POSTwithClient_send_led_clear(led_val_string, x_val_string,
   y_val_string, rgb_clear_val_string, rgb_clear_val_string,
   rgb_clear_val_string);
```

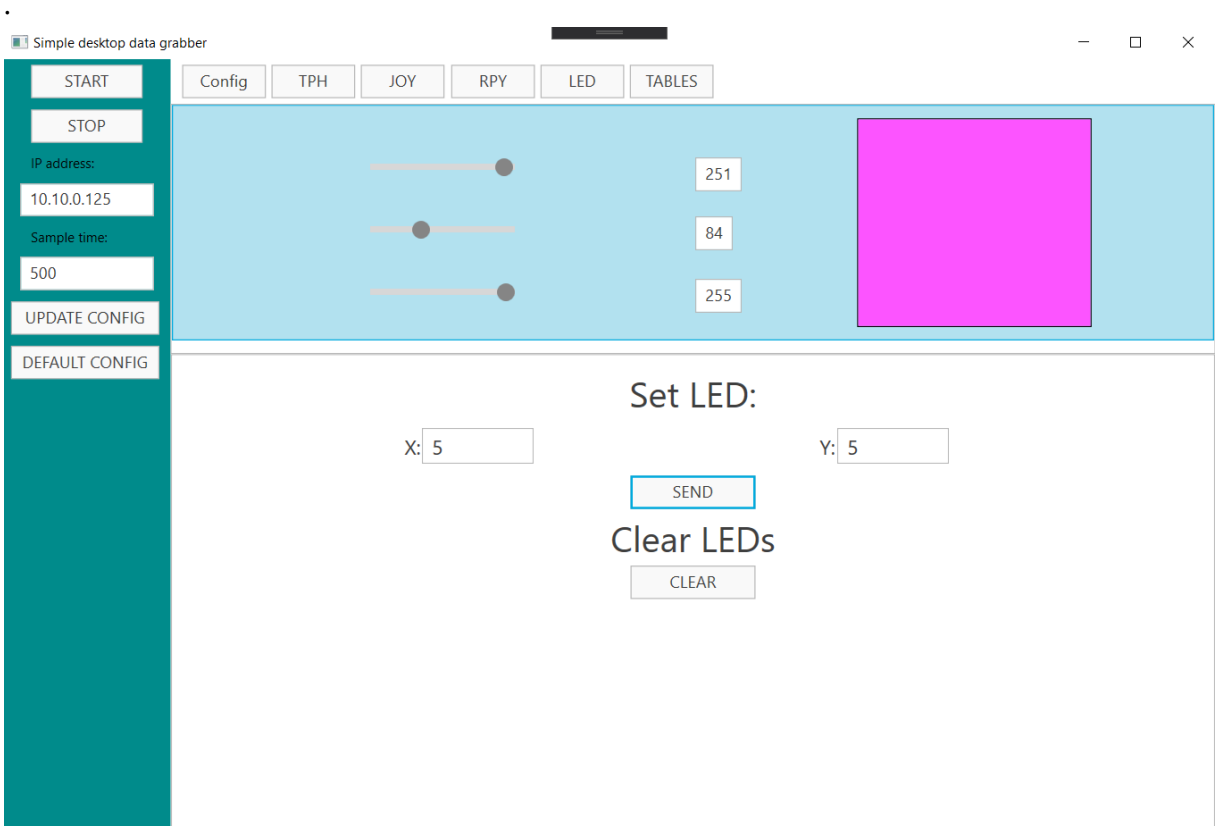
Listing 13 Funkcje włączające i wyłączające diody LED

```
1. public async Task<string> POSTwithClient_send_led(string x, string
   y, string r, string g, string b)
2.         {
3.             string responseText_send = null;
4.             string url = "http://" + ip +
   "/projekt/setled.php?LED00=[" + x + "," + y + "," + r + "," + g +
   "," + b + "]";
```

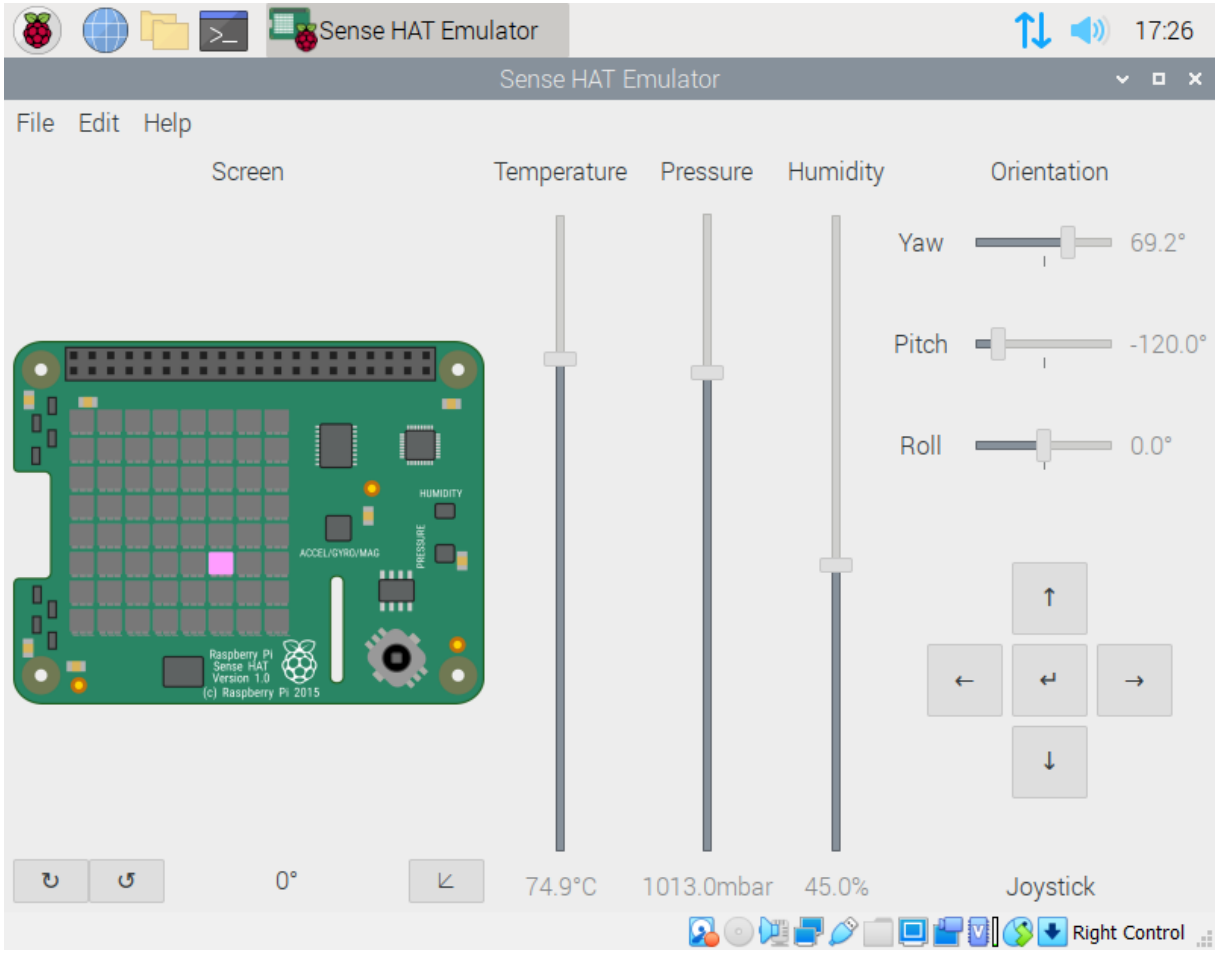
```
5.              try
6.              {
7.                  using (HttpClient client = new HttpClient())
8.                  {
9.                      // POST request data
10.                     var requestDataCollection = new
    List<KeyValuePair<string, string>>();
11.                     requestDataCollection.Add(new
    KeyValuePair<string, string>("filename", "data"));
12.
13.                     var requestData = new
    FormUrlEncodedContent(requestDataCollection);
14.                     // Sent POST request
15.                     var result = await client.PostAsync(url,
    requestData);
16.                     // Read response content
17.                     responseText_send = await
    result.Content.ReadAsStringAsync();
18.
19.                 }
20.             }
21.             catch (Exception e)
22.             {
23.                 Debug.WriteLine("NETWORK ERROR");
24.                 Debug.WriteLine(e);
25.             }
26.             return responseText_send;
27.         }
28.         public async Task<string>
    POSTwithClient_send_led_clear(string[] led, string[] x, string[]
    y, string[] r, string[] g, string[] b)
29.         {
30.             string responseText_send = null;
31.
32.             string url = "http://" + ip +
    "/projekt/setled.php?LED" + led[0] + "=[" + x[0] + "," + y[0] +
    "," + r[0] + "," + g[0] + "," + b[0] + "]";
33.
34.             for (int i = 1; i < 64; i++)
35.             {
36.                 url = url + "&LED" + led[i] + "=[" + x[i] + "," +
    y[i] + "," + r[i] + "," + g[i] + "," + b[i] + "]";
37.             }
38.             try
39.             {
40.                 using (HttpClient client = new HttpClient())
41.                 {
42.                     // POST request data
```

```
43.                     var requestDataCollection = new
   List<KeyValuePair<string, string>>();
44.                     requestDataCollection.Add(new
   KeyValuePair<string, string>("filename", "data"));
45.
46.                     var requestData = new
   FormUrlEncodedContent(requestDataCollection);
47.                     // Sent POST request
48.                     var result = await client.PostAsync(url,
   requestData);
49.                     // Read response content
50.                     responseText_send = await
   result.Content.ReadAsStringAsync();
51.
52.                 }
53.             }
54.             catch (Exception e)
55.             {
56.                 Debug.WriteLine("NETWORK ERROR");
57.                 Debug.WriteLine(e);
58.             }
59.             return responseText_send;
60.         }
```

Listing 14 Zapytania POST obsługujące komunikacje między klientem i serwem



Rysunek 5 Interfejs użytkownika i działanie sekcji LED



Rysunek 6 Efekt działania aplikacji klienta

2.2.6 TABLICA DYNAMICZNA

Wyświetlanie dynamiczne danych w tablicy realizowane jest za pomocą funkcji wywoływanej okresowo, która pobiera dane w postaci Json_Array. Później pętla for ustawia wartości poszczególnych komórek (listing 15) w tabeli za pomocą klasy TablesViewModel.(listing 16)

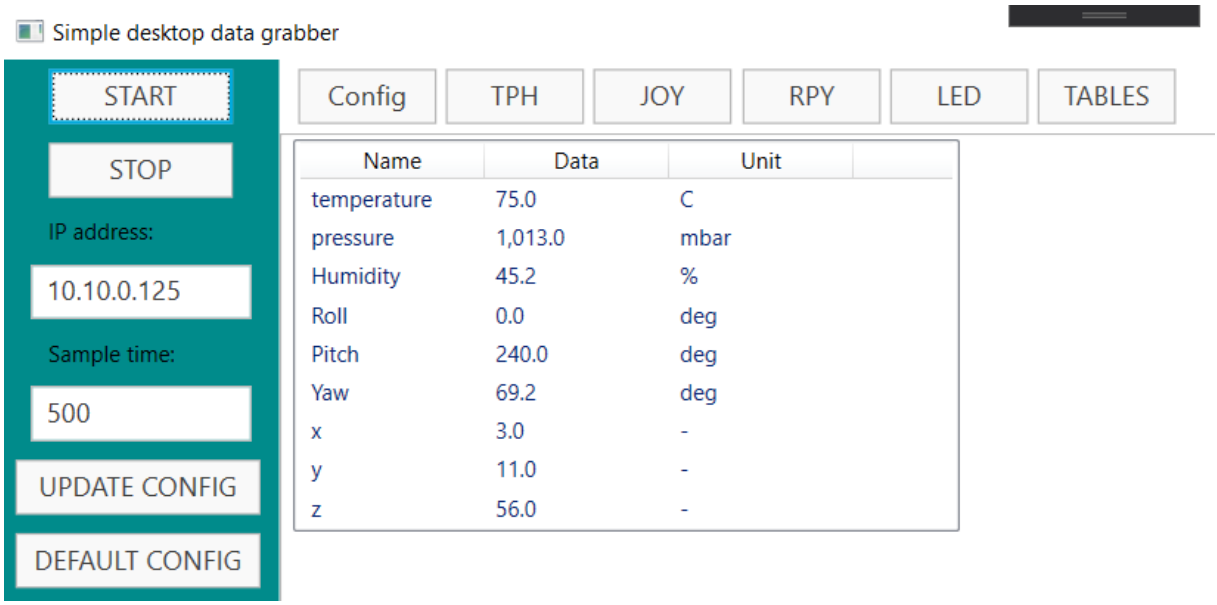
```
1.    public ObservableCollection<TablesViewModel> Data_tables { get; set; }
2.
3.
4.  try
5.    {
6.
7.    App.Current.Dispatcher.Invoke((System.Action)delegate
8.    {
9.        if (responseText_Array != null)
10.         {
11.           JArray DataJsonArray = JArray.Parse(responseText_Array);
12.

    var Data_Tables_List =DataJsonArray.ToObject<List<TablesModel>>();

13.   Data_tables.Clear();
14.
15.       if (Data_tables.Count < Data_Tables_List.Count)
16.       {
17.           foreach (var d in Data_Tables_List)
18.           {
19.            Data_tables.Add(new TablesViewModel(d));
20.            }
21.       }
22.        }
23.   });
```

Listing 15 Pętla ustawiająca wartości komórek

```
1.           public class TablesViewModel : INotifyPropertyChanged
2.     {
3.         private string name_data;
4.
5.         public string Name
6.         {
7.             get
8.             {
9.                 return name_data;
10.            }
11.            set
12.            {
13.                if (name_data != value)
14.                {
15.                    name_data = value;
16.                    OnPropertyChanged("Name");
17.                }
18.            }
19.        }
20.
21.        private double value_data;
22.        public string Value
23.        {
24.            get
25.            {
26.                return value_data.ToString("N1",
   CultureInfo.InvariantCulture);
27.            }
28.            set
29.            {
30.                if (Double.TryParse(value, NumberStyles.Any,
   CultureInfo.InvariantCulture, out double tmp) && value_data !=
   tmp)
31.                {
32.                    value_data = tmp;
33.                    OnPropertyChanged("Value");
34.                }
35.            }
36.        }
```

```
1.
2.
3.          private string unit_data;
4.          public string Unit
5.          {
6.              get
7.              {
8.                  return unit_data;
9.              }
10.               set
11.               {
12.                   if (unit_data != value)
13.                   {
14.                       unit_data = value;
15.                       OnPropertyChanged("Unit");
16.                   }
17.               }
18.           }
19.
20.
21.           public TablesViewModel(TablesModel model)
22.           {
23.               name_data = model.Name;
24.               OnPropertyChanged("Name");
25.
26.               value_data = model.Value;
27.               OnPropertyChanged("Value");
28.
29.               unit_data = model.Unit;
30.               OnPropertyChanged("Unit");
31.
32.           }
```

Listing 16 Klasa TablesViewModel



Rysunek 7 Interfejs użytkownika sekcji TABLES

2.3 IMPLEMENTACJA SYSTEMU – APLIKACJA ANDROID

2.3.1 STRUKTURA APLIKACJI

Struktura aplikacji opiera się o model widoków oraz aktywności. Wszystkie funkcjonalności zostały zaimplementowane w pliku MainActivity, a przełączanie między nimi możliwe jest dzięki użyciu obiektów Ident, które pozwalają przekazać dane oraz wyświetlić dany widok. Przykład takiej akcji zawartej w pliku został zaprezentowany na listingu 17.

```
1. private void configurebutton_joystick() {
2.         Intent openDataIntent = new Intent(this,
   JoystickActivity.class);
3.         Bundle configBundle = new Bundle();
4.         configBundle.putInt(DATA.CONFIG_SAMPLE_TIME, sampleTime);
5.         configBundle.putString(DATA.CONFIG_IP_ADDRESS,
   ipAddress);//to
6.         configBundle.putInt(DATA.CONFIG_SAMPLE_QUANTITY,
   sampleQuantity);
7.         openDataIntent.putExtras(configBundle);
8.         startActivity(openDataIntent);
9.     }
```

Listing 17 Przełączanie między widokami

2.3.2 WYKRESY TPH

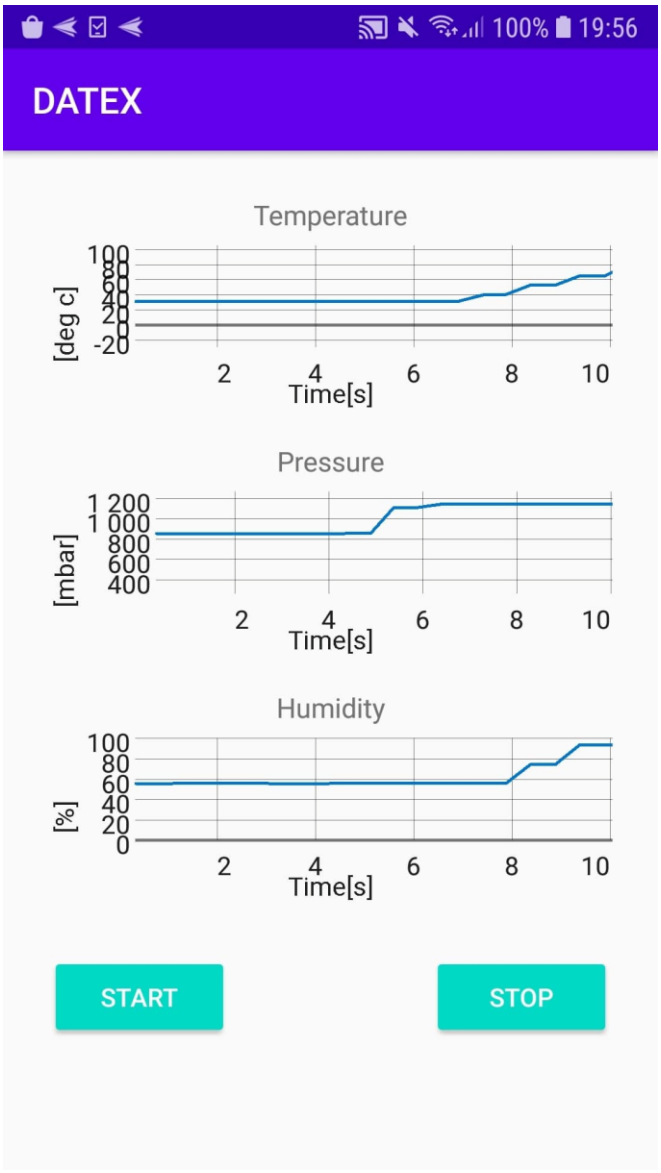
Wykresy temperatury, ciśnienia i wilgotności zostały wykonane w oparciu o bibliotekę GraphView. Na listingu 17 przedstawiono inicjalizację jednego z wykresów. Na listingu 18 ustawiona zostaje startowa konfiuracja, a na 19 została przedstawioan aktualizacj wykresu. Rysunek 8. pokazuje interfejs i działanie wykresów tph.

```
1.      public class ChartsActivity extends AppCompatActivity {
2.
3.      int sampleTime =DATA.DEFAULT_SAMPLE_TIME;
4.      String ipAddress =DATA.DEFAULT_IP_ADDRESS;
5.      int sampleQuantity =DATA.DEFAULT_SAMPLE_QUANTITY;
6.
7.
8.      /* Graph1 */
9.      private GraphView dataGraph1;
10.       private LineGraphSeries<DataPoint> dataSeries1;
11.       private final int dataGraph1MaxDataPointsNumber = 1000;
12.       private final double dataGraph1MaxX = 10.0d;
13.       private final double dataGraph1MinX =  0.0d;
14.       private final double dataGraph1MaxY =  105.0d;
15.       private final double dataGraph1MinY = -30.0d;
16.   ...
```

Listing 18

```
1.      @Override
2.      protected void onCreate(Bundle savedInstanceState) {
3.          super.onCreate(savedInstanceState);
4.          setContentView(R.layout.activity_charts);
5.
6.          // get the Intent that started this Activity
7.          Intent intent = getIntent();
8.
9.          // get the Bundle that stores the data of this Activity
10.           Bundle configBundle = intent.getExtras();
11.           ipAddress =
    configBundle.getString(DATA.CONFIG_IP_ADDRESS,
    DATA.DEFAULT_IP_ADDRESS);
12.
13.           sampleTime = configBundle.getInt(DATA.CONFIG_SAMPLE_TIME,
    DATA.DEFAULT_SAMPLE_TIME);
14.
15.           sampleQuantity =
    configBundle.getInt(DATA.CONFIG_SAMPLE_QUANTITY,
    DATA.DEFAULT_SAMPLE_QUANTITY);
16.
17.           /* BEGIN initialize GraphView1 */
18.           // https://github.com/jjoe64/GraphView/wiki
19.           dataGraph1 = (GraphView)findViewById(R.id.dataGraph1);
20.           dataSeries1 = new LineGraphSeries<>(new DataPoint[]{});
21.           dataGraph1.addSeries(dataSeries1);
22.           dataGraph1.getViewport().setXAxisBoundsManual(true);
23.           dataGraph1.getViewport().setMinX(dataGraph1MinX);
24.           dataGraph1.getViewport().setMaxX(dataGraph1MaxX);
25.           dataGraph1.getViewport().setYAxisBoundsManual(true);
26.           dataGraph1.getViewport().setMinY(dataGraph1MinY);
27.           dataGraph1.getViewport().setMaxY(dataGraph1MaxY);
28.           /* END initialize GraphView */
```

Listing 19


100% 19:56
DATEX
Temperature
[deg c]
100
80
60
40
20
0
-20
2
4
6
8
10
Time[s]
Pressure
[mbar]
1 200
1 000
800
600
400
2
4
6
8
10
Time[s]
Humidity
[%]
100
80
60
40
20
0
2
4
6
8
10
Time[s]
START
STOP

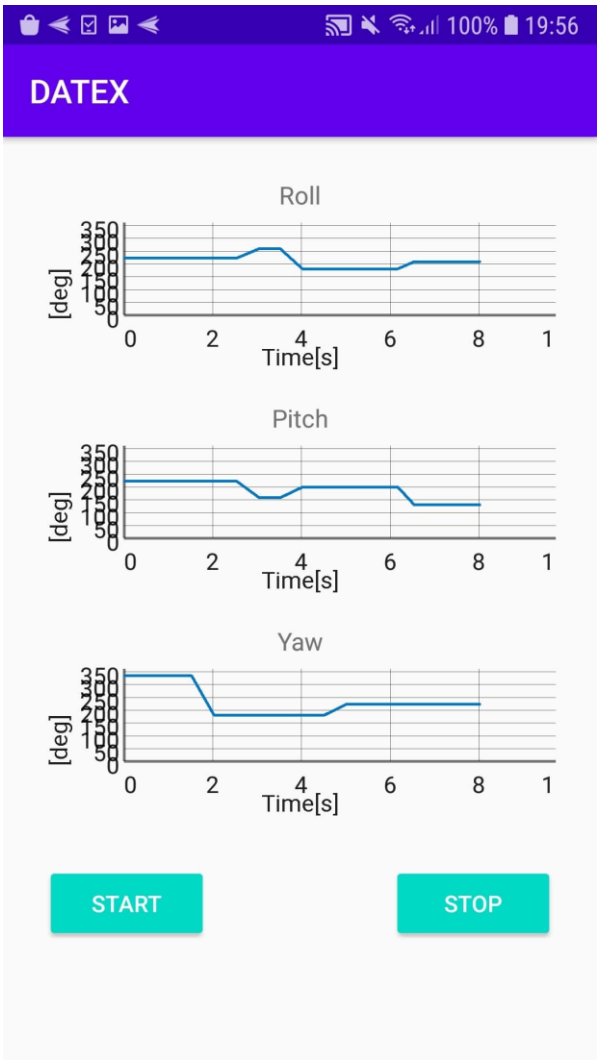
Rysunek 8

```
1.      private void responseHandling(String response)
2.     {
3.         if(requestTimer != null) {
4.             // get time stamp with SystemClock
5.             long requestTimerCurrentTime =
   SystemClock.uptimeMillis(); // current time
6.             requestTimerTimeStamp +=
   getValidTimeStampIncrease(requestTimerCurrentTime);
7.
8.             // get raw data from JSON response
9.             double temp = getRawDataFromResponse(response,"temp");
10.              double press =
   getRawDataFromResponse(response,"press");
11.              double humi =
   getRawDataFromResponse(response,"humi");
12.              drawcharts(temp,press,humi);
13.
14.
15.
16.              // remember previous time stamp
17.              requestTimerPreviousTime = requestTimerCurrentTime;
18.          }
19.      }
```

Listing 20

2.3.3 WYKRESY RPY

Wykresy orientacji roll, pitch, yaw zostały wykonane analogicznie do wykresów tph z punktu poprzedniego.. Rysunek 9. pokazuje interfejs i działanie wykresów rpy.



Rysunek 9

2.3.4 WYKRES JOYSTICK

Wykres reprezentujący joystick został wykonany analogicznie do wykresów rpy z punktu poprzedniego. Różni się on jednak sposobem aktualizacji wykresu z uwagi na tylko jeden punkt reprezentujący dane. Na listingu 20 przedstawiono inicjalizację wykresu. Na listingu 21 znajduje się funkcja aktualizująca wykres. Rysunek 10. pokazuje interfejs i działanie wykresu joysticka.

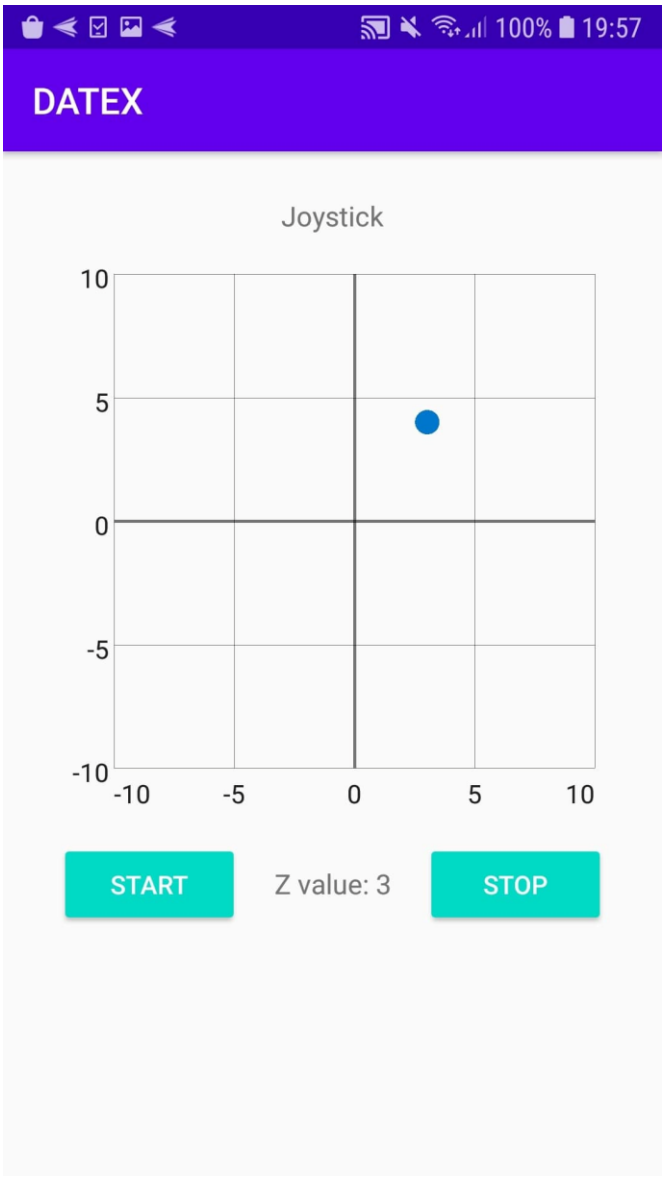
```
1. public class JoystickActivity extends AppCompatActivity {
2.
3.     int sampleTime =DATA.DEFAULT_SAMPLE_TIME;
4.     String ipAddress =DATA.DEFAULT_IP_ADDRESS;
5.     int sampleQuantity =DATA.DEFAULT_SAMPLE_QUANTITY;
6.     TextView z_val;
7.
8.     /* Graph1 */
9.     private GraphView dataGraph_joystick;
10.      private PointsGraphSeries<DataPoint> dataSeriesx;
11.      private final int dataGraph1MaxDataPointsNumber = 1000;
12.      private final double dataGraph1MaxX = 10.0d;
13.      private final double dataGraph1MinX =  -10.0d;
14.      private final double dataGraph1MaxY =  10.0d;
15.      private final double dataGraph1MinY = -10.0d;
16.
```

Listing 21

```
1. private void drawcharts(int x, int y)
2.     {
3.         // update plot series
4.         double timeStamp = requestTimerTimeStamp / 1000.0; //
   [sec]
5.
6.         boolean scrollGraph1 = (timeStamp > dataGraph1MaxX);
7.         dataSeriesx.resetData(generateData(x,y));
8.         dataSeriesx.appendData(new DataPoint(x, y), scrollGraph1,
   sampleQuantity);
9.
10.          // refresh chart
11.          dataGraph_joystick.onDataChanged(true, true);
12.
13.      }
```

```
14.  private void responseHandling(String response)
15.      {
16.          if(requestTimer != null) {
17.              // get time stamp with SystemClock
18.              long requestTimerCurrentTime =
    SystemClock.uptimeMillis(); // current time
19.              requestTimerTimeStamp +=
    getValidTimeStampIncrease(requestTimerCurrentTime);
20.
21.              // get raw data from JSON response
22.              int x = (int) getRawDataFromResponse(response,"x");
23.              int y = (int) getRawDataFromResponse(response,"y");
24.              int z = (int) getRawDataFromResponse(response,"z");
25.
26.
27.              z_val = findViewById(R.id.z_value);
28.              //String z_string = int.toString(z);
29.              String z_string = String.valueOf(z);
30.              String z_text = "Z value: ";
31.              String full_z  =z_text + z_string;
32.              z_val.setText(full_z);
33.
34.
35.              drawcharts(x,y);
36.
37.              // remember previous time stamp
38.              requestTimerPreviousTime = requestTimerCurrentTime;
39.          }
40.      }
41.      /**
42.       * @brief Swaps old Datapoints for new ones.
43.       */
44.      private DataPoint[] generateData(int x, int y) {
45.          int count = 1;
46.          DataPoint[] values = new DataPoint[1];
47.          for (int i=0; i<count; i++) {
48.              int horizontal = x;
49.              int vertical = y;
50.              DataPoint v = new DataPoint(horizontal, vertical);
51.              values[i] = v;
52.          }
53.          return values;
54.      }
```

Listing 22


100% 19:57
DATEX
Joystick
10
5
0
-5
-10
-10
-5
0
5
10
START
Z value: 3
STOP

Rysunek 10

2.3.5 MATRYCA LED

Do sterowania matrycą LED służy funkcj przedstawione na listingu 23, którazapisuje do pamięci naciśniety guzik przez użytkownika. Na listingu 24 przedstawiona została obługa pasków wyboru koloru oraz konwersji danych do postać Int. Listing 25 przedstawia metody

```
1.  /**
2.       * @brief Handles the logic for pressing any button on
    led_matrix layout
3.       * @param v View from which the code takes the ID of pressed
    button.
4.       */
5.      public void ledBtn_onClick(View v) {
6.          switch (v.getId()){
7.              case R.id.ledBtnClear:{
8.                  setAllLedColoursToDefault();
9.                  sendLedClearRequest();
10.                   break;
11.               }
12.               case R.id.ledBtnSend:{
13.                   sendLedChangeRequest();
14.                   break;
15.               }
16.               default:{/*Do nothing*/}
17.           }
18.       }
19.
20.       /**
21.        * @brief Handles the logic for pressing any view on
    led_matrix layout
22.        * @param v View from which the code takes the ID of pressed
    View
23.        */
24.       public void View_onClick(View v) {
25.           v.setBackgroundColor(currentColour);
26.           String tag=(String)v.getTag();
27.           Vector index=ledTagToIndexConverter(tag);
28.           int _x=(int)index.get(0);
29.           int _y=(int)index.get(1);
30.
31.           ledColours[_x][_y][0]=red;
32.           ledColours[_x][_y][1]=green;
33.           ledColours[_x][_y][2]=blue;
34.       }
```

Listing 23

komunikacji z skryptem serwerowym oraz przesyłanie danych. Funkcje korzystają z zapytania POST. Rysunek 11. pokazuje interfejs i działanie LEDów.

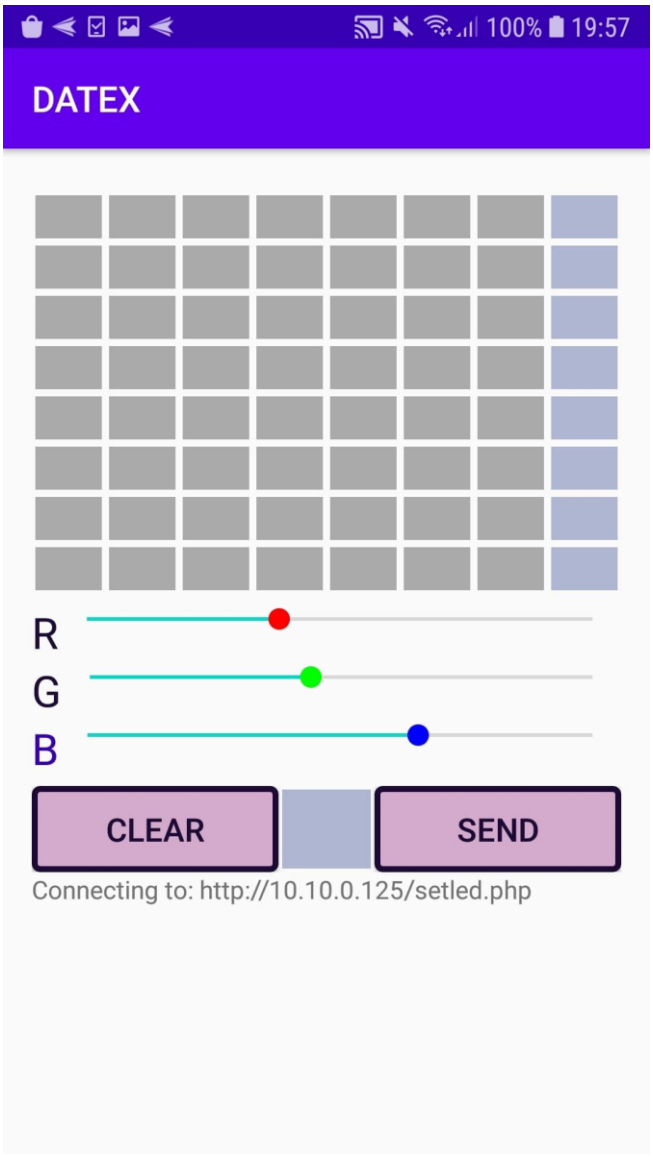
```
1. /**
2.      * @brief updates colour selected by colourSelect to the
   changedValue
3.      * @param colourSelect 'R', 'G' or 'B' key
4.      * @param changedValue changed value to be saved in global
   variable
5.      * @retval ARGB intiger value of a colour
6.      */
7.     private int seekBarUpdate(char colourSelect, int
   changedValue){
8.         switch (colourSelect){
9.             case 'R': red=changedValue; break;
10.              case 'G': green=changedValue; break;
11.              case 'B': blue=changedValue; break;
12.              default: break;
13.          }
14.          alpha=(red+green+blue)/3;
15.          return argbToIntConverter(alpha, red, green, blue);
16.     }
17.
18.     /**
19.      * @brief converts 4 ARGB varaibles into one int variable
20.      * @param _alpha transparency of a colour
21.      * @param _red red element of a colour
22.      * @param _green green element of a colour
23.      * @param _blue blue element of a colour
24.      * @retval int value of a colour
25.      */
26.     private int argbToIntConverter(int _alpha, int _red, int
   _green, int _blue){
27.          return (_alpha & 0xff) << 24 | (_red & 0xff) << 16
   |(_green & 0xff) << 8 | (_blue & 0xff);
28.     }
```

Listing 24

```
1.   /**
2.       * @brief sends request to change data on physical device or
   emulator
3.       */
4.     private void sendLedChangeRequest(){
5.         String url = getURL(ipAddress);
6.         StringRequest postRequest = new
   StringRequest(Request.Method.POST, url,
7.                 new Response.Listener<String>() {
8.                     @Override
9.                     public void onResponse(String response) {
10.                         Log.d("Response", response);
11.                     }
12.                 },
13.                 new Response.ErrorListener(){
14.                     @Override
15.                     public void onErrorResponse(VolleyError
   error){
16.                         String msg = error.getMessage();
17.                         if( msg != null) {
18.                             Log.d("Error.Response", msg);
19.                         } else {
20.                             Log.d("Error.Response", "UNKNOWN");
21.                             // error type specific code
22.                         }
23.                     }
24.                 }
25.         ){
26.             @Override
27.             protected Map<String, String> getParams(){
28.                 return getLedDisplayParams();
29.             }
30.         };
31.         postRequest.setRetryPolicy(new DefaultRetryPolicy(5000,
   0, DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
32.         queue.add(postRequest);
33.     }
```

```
34.      /**
35.       * @brief sends request to clear pixel colours on physical
   device or emulator
36.       */
37.      private void sendLedClearRequest(){
38.          String url = getURL(ipAddress);
39.          StringRequest postRequest = new
   StringRequest(Request.Method.POST, url,
40.                  new Response.Listener<String>() {
41.                      @Override
42.                      public void onResponse(String response) {
43.                          Log.d("Response", response);
44.                      }
45.                  },
46.                  new Response.ErrorListener(){
47.                      @Override
48.                      public void onErrorResponse(VolleyError
   error){
49.                          String msg = error.getMessage();
50.                          if( msg != null) {
51.                              Log.d("Error.Response", msg);
52.                          } else {
53.                              // error type specific code
54.                          }
55.                      }
56.                  }
57.          ){
58.              @Override
59.              protected Map<String, String> getParams(){
60.                  return paramClear;
61.              }
62.          };
63.          postRequest.setRetryPolicy(new DefaultRetryPolicy(5000,
   0, DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
64.          queue.add(postRequest);
65.      }
```

Listing 25



Ryunek 11

2.3.6 TABLICE DYNAMICZNE

Wyświetlanie dynamiczne danych w tablicy realizowane jest za pomocą funkcji wywoływanej okresowo, która pobiera dane w postaci Json_Array listing 26. Później pętla for ustawia wartości poszczególnych TextView listing 27. Rysunek 12. pokazuje interfejs i działanie tablic.

```
1. private String getRawDataFromResponse(String response, Integer
   item,String name) {
2.         JSONArray jObject;
3.        JSONObject x = null;
4.         String out = null;
5.
6.         // Create generic JSON object form string
7.         try {
8.             jObject = new JSONArray(response);
9.         } catch (JSONException e) {
10.             e.printStackTrace();
11.             return out;
12.          }
13.
14.          // Read chart data form JSON object
15.          try {
16.              x = jObject.getJSONObject(item);
17.              out = x.get(name).toString();
18.          } catch (JSONException e) {
19.              e.printStackTrace();
20.          }
21.
22.          return out;
23.      }
```

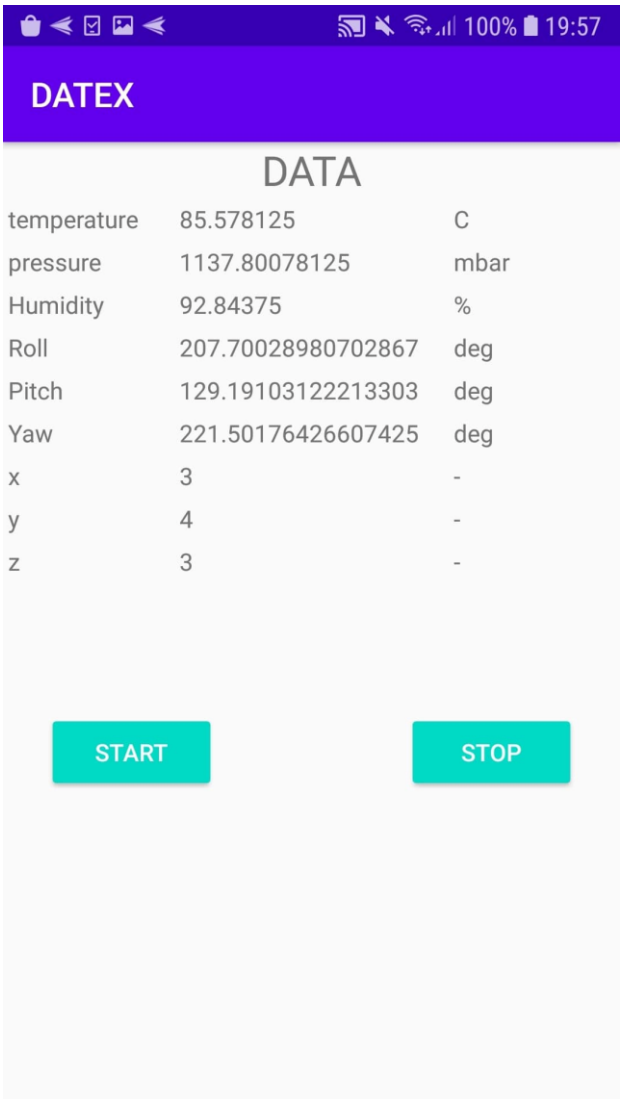
Listing 26

```
1. private void responseHandling(String response) throws
   JSONException {
2.         if(requestTimer != null) {
3.             // get time stamp with SystemClock
4.             long requestTimerCurrentTime =
   SystemClock.uptimeMillis(); // current time
5.             requestTimerTimeStamp +=
   getValidTimeStampIncrease(requestTimerCurrentTime);
6.             double timeStamp = requestTimerTimeStamp / 1000.0; //
   [sec]
7.
8.             // get raw data from JSON response
9.             String data1 = "error";
10.              String data2 = "error";
11.              String data3 = "error";
12.
13.
14.              Vector<String>buffor  = new Vector<String>();
15.              // set ip text in box
16.
17.
18.              int i = 0;
19.
20.
21.              while (true) {
22.                  data1 = getRawDataFromResponse(response, i,
   "name");
23.                  if (data1 == null) {
24.                      break;
25.                  }
26.                  data2 = getRawDataFromResponse(response, i + 1,
   "value");
27.                  if (data2 == null) {
28.                      break;
29.                  }
30.                  data3 = getRawDataFromResponse(response, i + 2,
   "unit");
31.                  if (data3 == null) {
32.                      break;
33.                  }
34.
35.                  buffor.add(data1);
36.                  buffor.add(data2);
37.                  buffor.add(data3);
38.
39.                  i = i + 3;
40.              }
```

```
41.  int buffer_iteration = 0;
42.              int id = 1;
43.              //clearng textview
44.   while (buffer_iteration< clearing_buffor_size) {
45.
46.                  name_view =
    (TextView)findViewById(getResources().getIdentifier("name"+ id,
    "id", getPackageName()));
47.                  name_view.setText(null);
48.
49.
50.                  value_view= (TextView)
    findViewById(getResources().getIdentifier("value"+id, "id",
    getPackageName()));
51.                  value_view.setText(null);
52.
53.
54.                  unit_view = (TextView)
    findViewById(getResources().getIdentifier("unit"+id, "id",
    getPackageName()));
55.                  unit_view.setText(null);
56.                  id++;
57.                  buffer_iteration = buffer_iteration + 3;
58.              }
59.
60.              id=1;
61.              buffer_iteration = 0;
62.              //set data to textview
63.  while (buffer_iteration < buffor.size()) {
64.
65.                  String name = buffor.get(buffer_iteration);
66.                  String value = buffor.get(buffer_iteration + 1);
67.                  String unit = buffor.get(buffer_iteration + 2);
68.                  name_view =
    (TextView)findViewById(getResources().getIdentifier("name"+ id,
    "id", getPackageName()));
69.                  name_view.setText(name);
70.
71.
72.                  value_view= (TextView)
    findViewById(getResources().getIdentifier("value"+id, "id",
    getPackageName()));
73.                  value_view.setText(value);
74.
75.
76.                  unit_view = (TextView)
    findViewById(getResources().getIdentifier("unit"+id, "id",
    getPackageName()));
77.                  unit_view.setText(unit);
78.                  id++;
79.                  buffer_iteration = buffer_iteration + 3;
80.              }
```

```
81.  clearing_buffor_size=buffor.size();
82.              buffor.clear();
83.
84.
85.              // remember previous time stamp
86.              requestTimerPreviousTime = requestTimerCurrentTime;
87.          }
88.      }
```

Listing 27



Rysunek 12

3. WYNIKI TESTÓW I INTEGRACJI SYSTEMU

Testy działania wszystkich aplikacji zostały przeprowadzone ręcznie. Testowane było zapisywanie danych, komunikacja serwer-klient, interakcja z UI oraz działanie całego systemu.

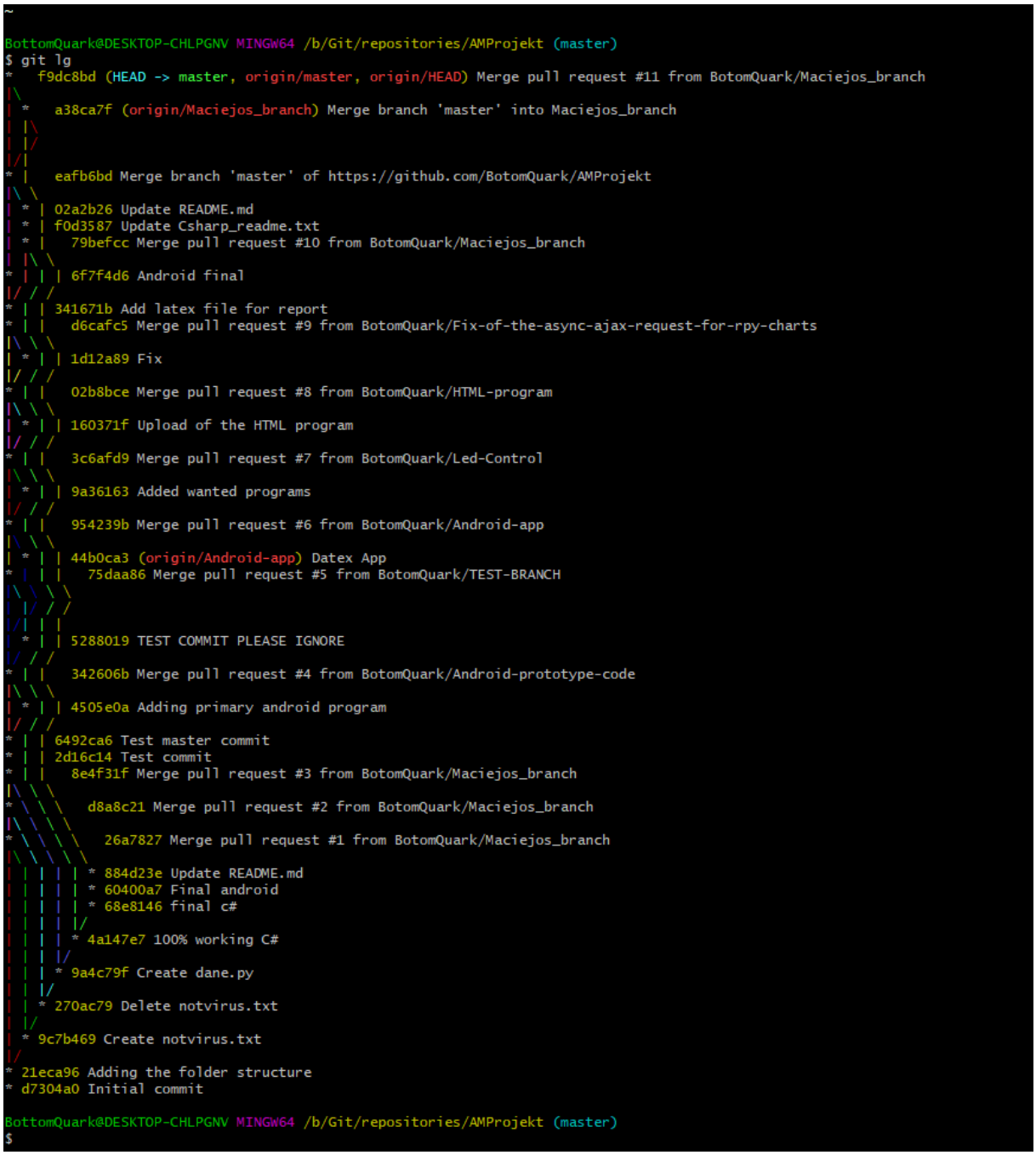
W procesie testowania wyeliminowane zostało wiele błędów takich jak:

- Błąd z wczytywaniem pamięci w aplikacji webowej, który powodował crashe aplikacji
- Problem z inicjalizacją skryptu python przez skrypt php – brak „sudo” przed ścieżką
- Problem z odczytem danych zapisanych w pliku .json przy konstruowaniu tablicy dynamicznej – należało zmienić strukturę danych na: [{"name": "temperature", "value": 85.453125, "unit": "C"},itd.]

Nie udało się wyeliminować błędu zmuszającego użytkownika aplikacji desktopowej do kliknięcia okien wyświetlających wartości kolorów RGB aby wartości te zostały przekazane dalej. Obecnie wiemy, że zostałoby to wykonane inaczej – TextBoxy byłyby bezpośrednio zbindowane do wartości double, a nie pośrednio przez slider.

4. Wnioski i podsumowanie

Spełnione zostały wszystkie wymogi podstawowe zakładane na początku oraz jeden wymóg dodatkowy – system kontroli wersji github.



Rysunek 13 Historia użytego repozytorium

Zrealizowany projekt nauczył nas konfiguracji serwera linux, protokoły komunikacji, format json oraz zapoznał z nowym IDE (android studio). Dla części grupy, również niektóre języki były nowością.