

Autorzy	
Kostiukov Oleksii	231972
Uladzimir Lipski	238961
Vikatr Hasiul	231862

---

Prowadzący	Dr inż. Marek Woda
Termin	Środa, godzina: 13:15

## Zastosowanie systemów wbudowanych

### *Raspberry Pi - Telegram Bot i web-aplikacja dla śledzenia klimatu otoczenia*

# 1 Cel projektu

Celem danego projektu jest wykorzystanie platformy *Raspberry* do realizacji *Telegram bot*'u oraz punktu pomiarowego.

**Telegram bot** jest aplikacją wykorzystującą interfejs aplikacji *Telegram* w celu komunikacji z wybranymi użytkownikami (którzy posiadają możliwość komunikacji ze stworzonym *bot'em*).

**Punkt pomiarowy.** Platforma *Raspberry* umożliwia podłączenie licznych czujników, dane z których można gromadzić na urządzeniu lub wysyłać do serwerów zdalnych. W danym projekcie zostaną podłączone czujniki:

1. temperatury,
2. wilgotności,
3. światła

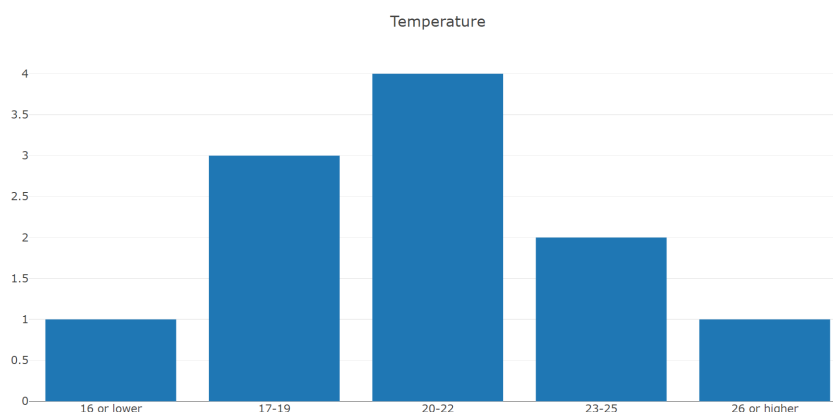
dane z których będą przechowywane na urządzeniu w celu przetwarzania i wyświetlania na stronie **WEB** w postaci interaktywnego wykresu.

Dodatkowo dany zbiór danych zostanie wykorzystany przez *Telegram bot* w celu powiadomienia użytkownika o aktualnych danych.

## 2 Implementacja asynchronicznego wyświetlania wyników pomiarowych

### 2.1 Cel aplikacji internetowej

Celem aplikacji internetowej jest wizualizacja danych, pobranych za pomocą czujników podłączonych do platformy Raspberry Pi. Dane są wyświetlane za pomocą histogramów. Wykresy pokazują jak często występuje każdy z zadanych zakresów temperatury, natężenia światła i wilgotności. Wykresy w przeglądarce muszą się odświeżać asynchronicznie po zmianie pliku z danymi.



Rysunek 1: Histogram temperatury

## 2.2 Wybór platformy web-serwera

Aplikacja internetowa została stworzona za pomocą *frameworku Flask*. *Flask - framework* do tworzenia aplikacji internetowych w języku *Python*, głównie do minimalistycznych aplikacji, które celowo zapewniają tylko podstawowe funkcje.

## 2.3 Generowanie wykresów w przeglądarce

Dla utworzenia wykresów w przeglądarce używana jest biblioteka *Plotly*, napisana w języku *JavaScript*. Biblioteka pozwala na utworzenie różnorodnych wykresów, z których dla danej aplikacji jest typem *bar*, odpowiadającym histogramowi. Dane dla wykresów są przekazywane od serwera do przeglądarki jako odpowiedź na żądanie *HTTP*.

Żądanie jest obsługiwane w języku *JavaScript* za pomocą ciągu funkcji, które zostały omówione w dalszej części dokumentacji.

## 2.4 Tworzenie wykresu

Listing 1: Funkcja do tworzenia histogramu za pomocą biblioteki *Plotly.js*

```
1 function drawChart(chart_divId, chart_title) {
2     var trace = {
3         type: 'bar',
4         x: chartRanges[chart_title],
5         y: chartData[chart_title],
6     };
7
8     var data = [ trace ];
9
10    chart_title = chart_title.charAt(0).toUpperCase()+chart_title.slice(1);
11    var layout = {
12        title: chart_title,
13        font: {size: 18}
14    };
15
16    Plotly.react(chart_divId, data, layout);
17 }
```

Dane z serwera są przekazywane w postaci tablicy. Dodatkowym zadaniem skryptu w języku *JavaScript* jest kategoryzowanie danych dla różnych zakresów, zanim wykres będzie stworzony.

Wartości dla osi *x* są nazwami zakresów, do których może należeć wartość przekazana z serwera. Dane zakresy są zdefiniowane statycznie i zadeklarowane na samym początku skryptu.

Listing 2: Zdefiniowany zakresy w skrypcie *JavaScript*

```
1 chartRanges = {
2     "temperature" :
3         ["16 or lower", "17-19", "20-22", "23-25", "26 or higher"],
4     "light":
5         ["349 or lower", "350-450", "451-550", "551-650", "651 or higher"],
6     "humidity":
7         ["0-20", "21-40", "41-60", "61-80", "81-100"]
8 }
```

## 2.5 Asynchroniczne wczytywanie danych z pliku

Dane, pobrane z czujników za pomocą platformy *Raspberry Pi*, są przechowywane w pliku z rozszerzeniem `.csv`. Dla asynchronicznego wczytywania danych z pliku została użyta moduł `watchdog`. Dany moduł został użyty do monitorowania wybranych plików. Wybór plików został zdefiniowany za pomocą wyrażeń regularnych. Jeżeli dowolny plik o zadanym rozszerzeniu zostaje zmieniony, to do serwera jest wysyłany sygnał. Do obsługi różnego rodzaju sygnałów jest używany moduł `flask.socketio`.

Listing 3: Połączenie gniazdka na serwerze z gniazdkiem w przeglądarce

```
1 @socketio.on('connect')
2 def test_connect():
3     global thread
4     if thread is None:
5         thread = socketio.start_background_task(target=background_thread)
```

Dla każdego sygnału mogą być zdefiniowane osobne funkcje. W danej implementacji sygnał zostaje wysłany jeżeli monitorowany plik został zmieniony, to serwer wysyła sygnał do przeglądarki za pomocą funkcji `on_modified`, która dodatkowo wczytuje dane z pliku, żeby przekazać do przeglądarki.

Listing 4: Definicja wątku, monitorującego zmiany w plikach o zadanych rozszerzeniu

```
1 def background_thread():
2     global observer
3     event_handler = CsvWatcher()
4     observer = Observer()
5     observer.schedule(event_handler, './', recursive=True)
6     observer.start()
```

Listing 5: Definicja funkcji wysyłającej sygnał i dane

```
1 def on_modified(self, event):
2     global csvFileName
3     data = readfile(csvFileName)
4     socketio.emit('modified', {'data': data})
```

## 2.6 Asynchroniczne odświeżanie wykresów w przeglądarce

Asynchroniczne odświeżanie wykresów umożliwia połączenie przeglądarki i serwera za pomocą gniazdek. Celem gniazdek jest asynchroniczna obsługa sygnałów, wysyłanych z serwera.

Do obsługi sygnału w języku JavaScript zostały zdefiniowane następujące funkcje:

Listing 6: Połączenie gniazdek w przeglądarce z serwerem i odebranie danych

```
1 var socket = io.connect('http://' + document.domain + ':' + location.port);
2
3 socket.on('modified', function(data) {
4     var newData = data['data'];
5     start(newData)
6 });
```

## 2.7 Dostęp do aplikacji w sieci prywatnej

Aplikacja jest uruchamiana po połączeniu *Raspberry Pi* z siecią. IP urządzenia zostanie wysłany do zdefiniowanego Telegram Bot'u.