Wydział Autoamtyki, Robotyki i Elektrotechniki
Automatyka i Robotyka sem.6

Aplikacje Mobilne i Wbudowane dla Internetu Przedmiotu

Projekt

Krzysztof Borowski
135806
Kuba Codogni
135806
Maciej Paderecki
138073

24 września 2020

1 Opis specyfikacji

Celem projektu było stworzenie programów na emulatorze RaspberryPi oraz trzech aplikacji użytkownika pozwalajace na komunikację z emulatorem. Zgodnie z założeniami projekt powinien spełniać minimalnie następujące wymogi:

- 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 umoąliwiają 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 umożliwiajż 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 umożliwią 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 Pliki wykonawcze na emualtorze RaspberryPi

Na płytcę RaspberryPi znajdują się trzy programy służące do komunikacji z aplikacjami użytkownika.

2.1 Plik dane.py

Plik dane.py jest plikiem mający w sobie pętle *whiletrue*, która co 0.1 sekundę odczytuje wartości pobierane z emulatora SenseHAT'a. Do przetworzenia danych zostały zaprojektowane trzy klasy widoczne na listingu 1. Przechowują one tylko i wyłącznie dane zapisywane do plików .json. Na listingu 2 pokazana jest funkcja zapisująca do pliku .json, funkcje zapisy do rpy.json oraz tph.json. Dodatkowo do przetworzenia zmiany pozycji joysticka utworzono serie funkcji zmieniającą wartość położenia wartości golablych x,y i z. Jedna z takich funkcji widoczna jest na listingu 3.

```
17      class EnvData_tph :
18        def __init__ (self , temp , press, humi ):
19          self.temp = temp
20          self.press = press
21          self.humi = humi
22
23      class EnvData_joy :
24        def __init__ (self , x, y, z):
25          self.x = x
26          self.y = y
27          self.z = z
28
29      class EnvData_rpy :
30        def __init__ (self , roll , pitch, yaw):
31          self.roll = roll
32          self.pitch = pitch
33          self.yaw = yaw
34
```

Listing 1: Zaprogramowane klasy

```
61      def save_joy():
62
63        with open('joy.json', 'w+') as outfile:
64          obj_data = EnvData_joy(x, y, z)
```

```
65          result = json.dumps(obj_data.__dict__)
66          outfile.write(result)
67
```

Listing 2: Funkcja zapisu pozycji joysticka do pliku joy.json

```
36      def pushed_up(event):
37        global y
38        if event.action != ACTION_RELEASED:
39        y = y + 1
40
```

Listing 3: Funkcja przetwarzająca zmiany pozycji joysticka na pozycję "górna"

Pętla while pokazana na listingu 4. przedstawia zachowanie się programu. Pętla powtarza się co 0.1 sekundy. Program odczytuje po kolei pozycję joysticka, wartości temperatury, ciśnienia, wilgotności oraz orientacji i zapisuje je po kolei do odpowiednich plików .json.

```
82      while True:
83
84        sense.stick.direction_up = pushed_up
85        sense.stick.direction_down = pushed_down
86        sense.stick.direction_left = pushed_left
87        sense.stick.direction_right = pushed_right
88        sense.stick.direction_middle = pushed_middle
89        sense.stick.direction_any = save_joy
90        save_joy()
91
92        temp = sense.get_temperature()
93        press = sense.get_pressure()
94        humi = sense.get_humidity()
95        save_tph()
96
97        orientation_degrees = sense.get_orientation_degrees()
98
99        roll=orientation_degrees["roll"]
100       pitch=orientation_degrees["pitch"]
101       yaw=orientation_degrees["yaw"]
102       save_rpy()
103
104       time.sleep(0.1)
105
```

Listing 4: Pętla while programu

2.2 Plik setled.py

Na listingu 5. przedstawiono pełen program setled.py. Program odpowiada za zapalanie odpowiednich ledów na emualtorze senseHAT. Plik wykonuje operacje za pomocą danych zawartych w leddata.json. Plik zawiera wektor wektorów. Wewnętrzy wektor zapisany jest w sekwencji $[y, x, red, green, blue]$. Program ustawia zapala ledy w pętli sprawdzając każdy następujący po sobie wektor.

```
82      #!/usr/bin/python
83      import json
84      from sense_emu import SenseHat
85
86      sense = SenseHat()
87
88      filename = "leddata.json";
89
90      if filename:
91        with open(filename, 'r') as f:
92          ledDisplayArray=json.load(f);
93
94      for led in ledDisplayArray:
95        # schemat led: y x R G B
96        sense.set_pixel(led[1], led[0], led[2], led[3], led[4]);
97
```

Listing 5: Program setled.py

2.3 Plik setled.php

Na listingu 6. przedstawiono pełen program setled.php. Program odpowiada za otrzymanie z programów użytkownika metodą POST i zapisaniu ich do pliku .json. Następnie prgram wywołuje program setled.py.

```
<?php
  //ini_set('display_errors',1);
  //error_reporting(E_ALL);
  header('Access-Control-Allow-Origin: http://localhost');
  header('Content-Type: application/json; charset=utf-8');

  function ledIndexToTagConverter($x, $y){
    return "LED" .$x .$y;
    }

  $ledDisplay = array();
  $ledDisplayDataFile = 'leddata.json';

  $n=0;

  for ($i=0; $i<8; $i++){
    for ($j=0; $j<8; $j++){
      $ledTag=ledIndexToTagConverter($i, $j);
      if(isset($_POST[$ledTag])){
        $ledDisplay[$n] = json_decode($_POST[$ledTag]);
        $n=$n+1;
      }
    }
  }

  $ledDisplayJson=json_encode($ledDisplay);
  $dataFile = fopen($ledDisplayDataFile, 'w+') or die("ERR1");
  fwrite($dataFile, $ledDisplayJson);
  fclose($dataFile);

  echo "ACK1 ";
  exec("sudo ./setled.py");
  echo "ACK2 ";
?>
```

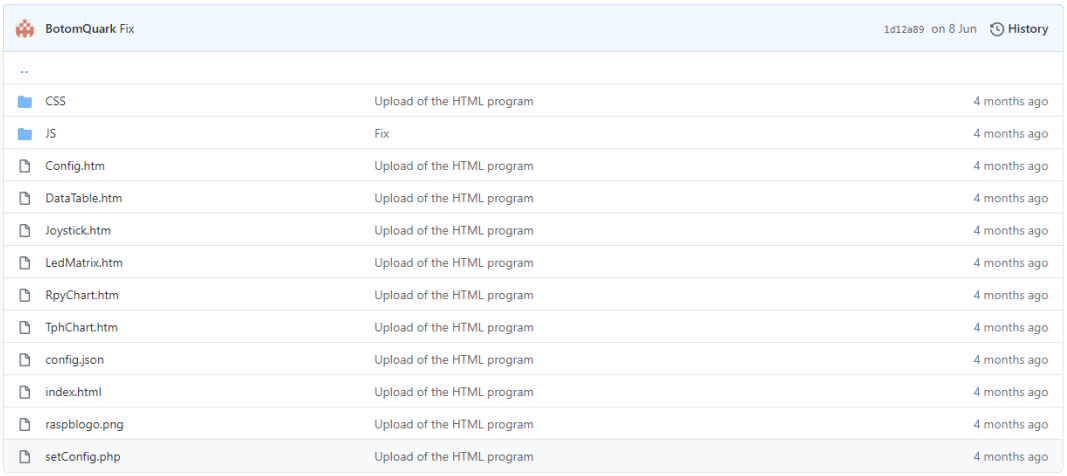
Listing 6: Program setled.php

3 Implementacja systemu - Aplikacja webowa

Link do prezentacji części aplikacji webowej znajduje się w punkcie 7. Źródła i linki. Aplikacja webowa znajduje się całkowicie na emulatorze RaspberryPi, i można uzyskać do niego dostęp za pomocą przeglądarki internetowej.
Do wykonania projektu użyto technologii AJAX oraz biblioteki Chart.js.
!!UWAGA!! Prezentacja programu została przeprowadzona na wersji v1.0 aplikacji. Aktualna wersja apliakcji to v1.1.

3.1 Organizacja kodu

W głównym folderze aplikacji webowej znajdują się wszystkie pliki html/htm. Wszystkie pliki CSS znajdują się w folderze CSS, oraz wszystkie pliki JavaScript znajdują się w folderze JS. Dodatkowo w głównym folderze znajduje się plik setConfig.php używany do ustawienia konfiguracji. Na rysunku 1. przedstawiono główny folder aplikacji webowej na github'ie.
Dodatkowo na listingu 7. przedstawiono ciało pliku index.html. Szablon wyglądu zawarta jest w pliku menu.css. Poprzez naciśnięcie przycisku po lewej strone interfejsu wywoływany jest odpowiedni plik .htm i wyświetlany jest po prawej stronie ekranu w iframe. Na rysunku 2 przedstawiono interfejs urzytkownika.

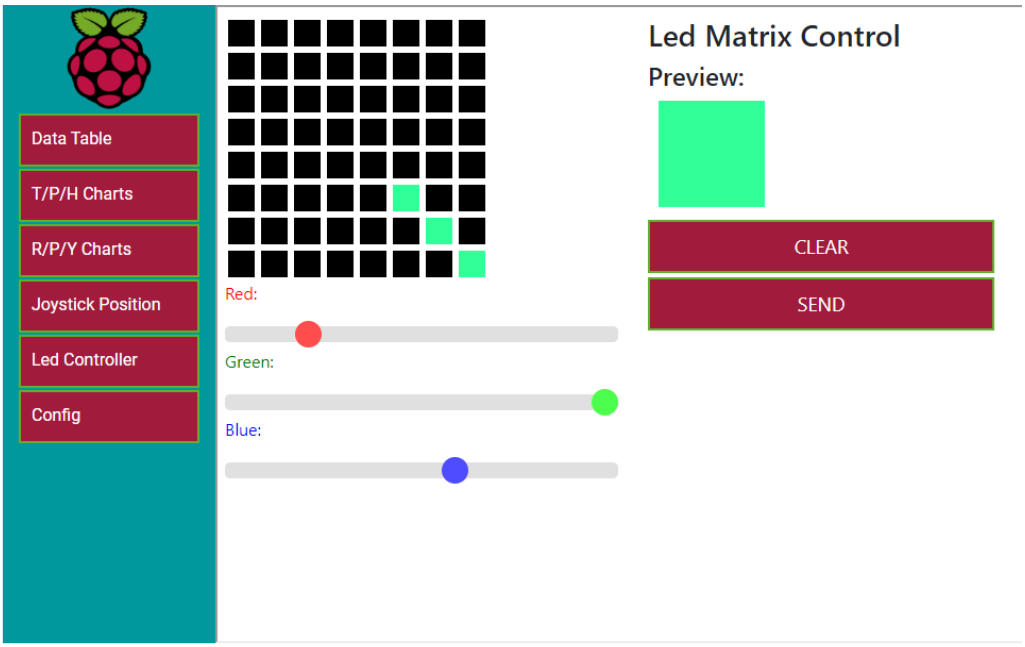


Rysunek 1: Folder główny w którym znajduje się index.html

```
27          <body>
28            <div id="container">
29              <div id="sidebar">
30                <div id="logo">
31                  <img src="raspblogo.png">
32                </div>
33                <a href="DataTable.htm" target="pageSelectFrame"><div class="menuOption"> Data Table
    </div></a>
34                <a href="TphChart.htm" target="pageSelectFrame"><div class="menuOption"> T/P/H Charts
     </div></a>
35                <a href="RpyChart.htm" target="pageSelectFrame"><div class="menuOption"> R/P/Y Charts
     </div></a>
36                <a href="Joystick.htm" target="pageSelectFrame"><div class="menuOption"> Joystick
    Position </div></a>
37                <a href="LedMatrix.htm" target="pageSelectFrame"><div class="menuOption"> Led
    Controller </div></a>
38                <a href="Config.htm" target="pageSelectFrame"><div class="menuOption"> Config </div><
    /a>
39              </div>
40              <div id="content">
41                <iframe  height="100%" width="100%" name="pageSelectFrame"></iframe>
42                </div>
43                <div id="footer">
44                  v1.1 Laboratoria Aplikacji Mobilnych i Wbudowanych dla Internetu przedmiotu
45                </div>
46              </div>
```

```
47          </body>
48
```

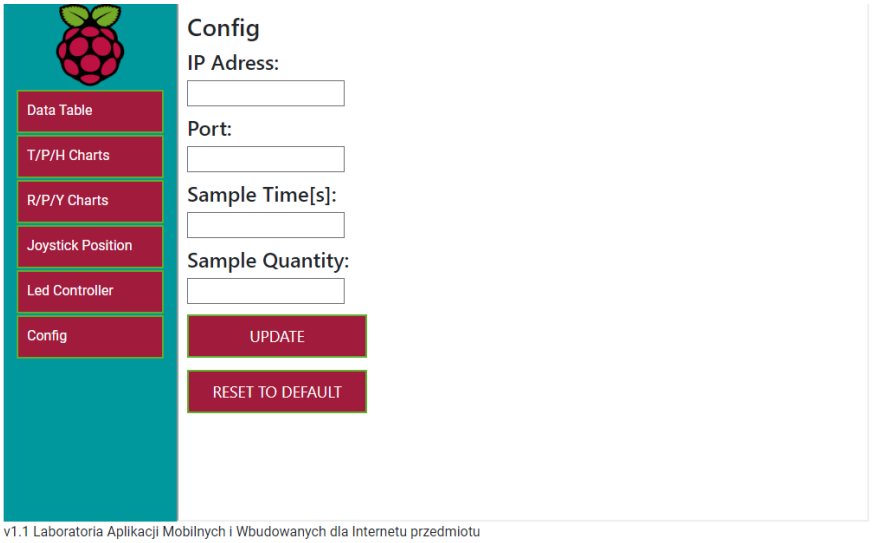
Listing 7: Kod Body plikeu index.html



Rysunek 2: Interfejs użytkownika

3.2 Kod konfiguracji

W celu zmiany konfiguracji (adresu IP, portu, czasu próbkowania i ilości próbek) należy nacisnać na przycisk "Config" w menu. Po naciśnięciu tego przycisku pojawi się ekran widoczny na rysunku 3. W celu wysłania zawołania do pliku



Rysunek 3: Interfejs użytkownika okna konfiguracji

setConfig.php ustawiający dane konfiguracyjne w pliku config.php program JS używa technologi AJAX. Taka sama sytuacja występuje przy resecie, z różnicą wysłania pustej wiadomości metodą GET.

```
58          function updateConfig(){
59            //Format for POST method: ip=...&&port=...&&sampleTime=...&&sampleQuantity=...
60            var dataText=getConfigDataForPostRequest();
61            console.log("Updating Config");
62
63            $.ajax(baseUrl+"setConfig.php", {
64              type: "POST",
65              data:dataText,
66              dataType: "text",
67              crossDomain: true,
68              beforeSend: function(x) {
69                console.log("AJAX POST REQUEST: BEGIN SENDING");
70              },
71              success: function(result) {
72                console.log("AJAX POST REQUEST: SUCCESFULL CODE: " + result);
73              },
74              error: function(XMLHttpRequest, textStatus, errorThrown) {
75                console.log("AJAX POST REQUEST: FAILURE");
76                console.log("STATUS: " + textStatus);
77                console.log("ERROR: " + errorThrown);
78              },
79              cache: false
80            });
81
82            getConfigData();
83          }
84
```

Listing 8: Kod config.js

Przy wywołaniu każdego pliku .htm wywoływana jest funkcja widoczna na listingu 9. Dodano linijkkę $cashe : false$ w celu zapobiegnięcia błędnego odczytu w pliku config.json spowodowany zapisem danych do cache. Każdy osobny plik JS posiada osobną funkcję $function\ updateConfig(jsonObject)$ aktualizującą domyślne ustawienia konfiguracyjne.

```
58        function getConfigData() {
59        $.ajax(baseUrl+"config.json", {
60          type: 'GET',
61          dataType: 'text',
62          crossDomain: true,
63          success: function(responseTEXT, status, xhr) {
64            var responseJSON = JSON.parse(responseTEXT);
65            debugHelp1=responseJSON;
66            console.log("Ajax success");
67            updateInputValues(responseJSON);
68          },
69          error: function (ajaxContext) {
70            console.log("Ajax error");
71          },
72          cache: false
73          });
74        }
75
```

Listing 9: Kod config.js

3.3 Strona tabeli danych

Okno tabeli danych pozwala na wybranie jakie dane mają zostać wyświetlane:

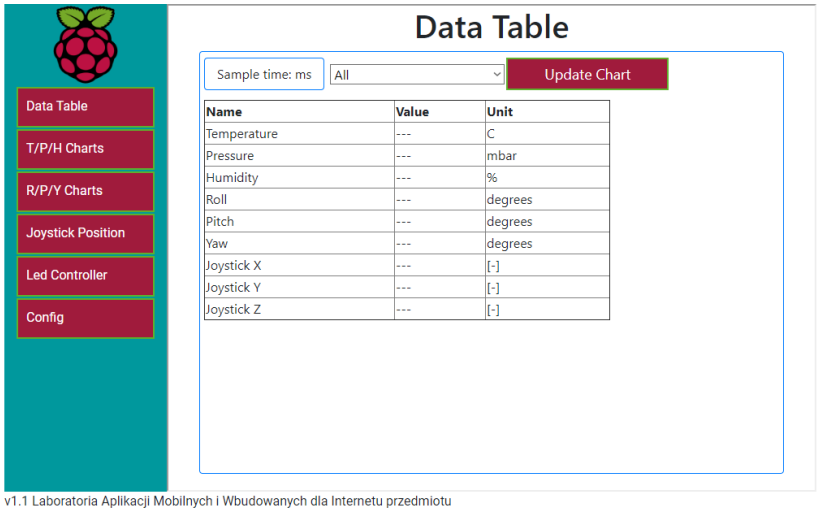
- ALL
- TPH measurement
- RPY measurement
- Joystick position
- TPH and RPY measurement

Tabela jest tworzona dynamiczne na podstawie wybranej opcji, która zapisywana jest w zniemmej *key*. Tabela tworzona jest poprzez stworzenie tablicy, do której dodane są obiekty, dodatkowo każda odczytywana zmienna dostaje index potrzebny do zmiany wartości w tablicy przy odczycie. Następnie tworzony jest obiekt HTML za pomocą komendy *document.createElement*("*TABLE*"), i dynamicznie tworzony jest element wyświetlany na interfejsie użytkownika.

```
58          /**
59          * @brief Code generating dynamic table, the size depends solely on the chosen key
60          */
61          function generateTable() {
62            //Build an array containing Customer records.
63            var data = new Array();
64            //Header
65            data.push(["Name", "Value", "Unit"]);
66
67            switch(key){
68              case "all":{
69                data.push(["Temperature", "---", 'C']);   tempIndex=1;
70                data.push(["Pressure", "---", 'mbar']);   pressIndex=2;
71                data.push(["Humidity", "---", '%']);    humiIndex=3;
72                data.push(["Roll", "---", 'degrees']);    rollIndex=4;
73                data.push(["Pitch", "---", 'degrees']);   pitchIndex=5;
74                data.push(["Yaw", "---", 'degrees']);   yawIndex=6;
75                data.push(["Joystick X", "---", '[-]']);  xIndex=7;
76                data.push(["Joystick Y", "---", '[-]']);  yIndex=8;
77                data.push(["Joystick Z", "---", '[-]']);  zIndex=9;
78              break;
79              }
80              case "tph":{
81                data.push(["Temperature", "---", 'C']);   tempIndex=1;
82                data.push(["Pressure", "---", 'mbar']);   pressIndex=2;
83                data.push(["Humidity", "---", '%']);    humiIndex=3;
84              break;
85              }
86              case "rpy":{
87                data.push(["Roll", "---", 'degrees']);    rollIndex=1;
88                data.push(["Pitch", "---", 'degrees']);   pitchIndex=2;
89                data.push(["Yaw", "---", 'degrees']);   yawIndex=3;
90              break;
91              }
92              case "joy":{
93                data.push(["Joystick X", "---", '[-]']);  xIndex=1;
94                data.push(["Joystick Y", "---", '[-]']);  yIndex=2;
95                data.push(["Joystick Z", "---", '[-]']);  zIndex=3;
96              break;
97              }
98              case "tph+rpy":{
99                data.push(["Temperature", "---", 'C']);   tempIndex=1;
100               data.push(["Pressure", "---", 'mbar']);   pressIndex=2;
101               data.push(["Humidity", "---", '%']);    humiIndex=3;
102               data.push(["Roll", "---", 'degrees']);    rollIndex=4;
103               data.push(["Pitch", "---", 'degrees']);   pitchIndex=5;
104               data.push(["Yaw", "---", 'degrees']);   yawIndex=6;
105             break;
106             }
107             default:{}
108           }
109
```

```
110           //Create a HTML Table element.
111           var table = document.createElement("TABLE");
112           table.border = "1";
113
114           //Get the count of columns.
115           var columnCount = data[0].length;
116
117           //Add the header row.
118           var row = table.insertRow(-1);
119           for (var i = 0; i < columnCount; i++) {
120             var headerCell = document.createElement("TH");
121             headerCell.innerHTML = data[0][i];
122             row.appendChild(headerCell);
123           }
124
125           //Add the data rows.
126           for (var i = 1; i < data.length; i++) {
127             row = table.insertRow(-1);
128             for (var j = 0; j < columnCount; j++) {
129               var cell = row.insertCell(-1);
130               cell.innerHTML = data[i][j];
131             }
132           }
133
134           debugHelp=table;
135
136           table.id="table";
137           var divTable = document.getElementById("divTable");
138           divTable.innerHTML = "";
139           divTable.appendChild(table);
140
141         }
142
```

Listing 10: Kod datatable.js



Rysunek 4: Interfejs użytkownika

3.4 Strona wykresu tph

Wykresy temperatury, ciśnienia i wilgotności zostały wykonane analogicznie jak wykresy położenia kątowego. Do wykonania wykresów użyto biblioteki zewnętrznej Chart.js. Na listingu 11 przedstawiono inicjalizacje wykresów. Dla uproszczenia pokazano tylko inicjalizacje pierwszego wykresu, wszystkie wykresy są typu ”line”. Na listingu 12 przedstawiono funkcję aktualizacji wykresu.

```
128         /**
129         * @brief Chart initialization
```

```
130         */
131         function chartInit()
132         {
133           //array wich consecutive integers: <0, maxSamplesNumber-1>
134           xData=[...Array(maxSamplesNumber.keys()]
135           //scaling all values times the sample time
136           xData.forEach(function(p,i) {this[i]=(this[i]*sampleTimeSec).toFixed(4);}, xData);
137
138           //last value of 'xdata'
139           lastTimeStam = +xData[xData.length=-];
140
141           // empty array
142           tyData = [];
143
144           // get chart context from 'canvas' element
145           tChartContext = $("#tChart")[0].getContext('2d');
146
147           tChart = new Chart(tChartContext, {
148             // The type of chart: linear plot
149             type: 'line',
150
151             // Dataset: 'xdata' as labels, 'ydata' as dataset.data
152             data: {
153               labels: xData,
154               datasets: [{
155               fill: false,
156               label: 'Temperature Chart',
157               backgroundColor: 'rgb(0, 0, 255)',
158               borderColor: 'rgb(0, 0, 255)',
159               data: tyData,
160               lineTension: 0
161               }]
162             },
163
164             // Configuration options
165             options: {
166               legend: {
167                 display: false
168               },
169               responsive: true,
170               maintainAspectRatio: false,
171               animation: false,
172               scales: {
173               yAxes: [{
174                 scaleLabel: {
175                 display: true,
176                 labelString: 'Temperature[C]'
177               },
178               ticks: {
179                 suggestedMin: -30,
180                 suggestedMax: 105
181               }
182               }],
183               xAxes: [{
184                 scaleLabel: {
185                   display: true,
186                 labelString: 'Time [s]'
187               }
188               }]
189             }
190             }
191           });
192
193
194           tyData = tChart.data.datasets[0].data;
195           xData = tChart.data.labels;
196         }
197
```

Listing 11: Kod tph_charts.js inicjalizacja wykresów

```
71          /**
```

```
72          * @brief Add new value to next data point.
73          * @param y New y-axis value
74          */
75          function addData(t, p, h){
76            if(tyData.length > maxSamplesNumber)
77            {
78              xData.splice(0,1);
79
80              tyData.splice(0,1);
81              pyData.splice(0,1);
82              hyData.splice(0,1);
83
84              lastTimeStamp += sampleTimeSec;
85
86              xData.push(lastTimeStamp.toFixed(4));
87            }
88
89            tyData.push(t);
90            pyData.push(p);
91            hyData.push(h);
92
93            tChart.update();
94            pChart.update();
95            hChart.update();
96
97
98          }
99
```

Listing 12: Kod tph_charts.js inicjalizacja wykresów

3.5 Strona strowania LED

Do sterowania diodami LED program używa kilku funkcji. Po naciśnięciu przycisku SEND program pobiera informację o aktualnie zapalonych diodach LED za pomocą kodu na listingu 13. oraz wysyła zapytanie za pomocą funkcji AXAJ. W przypadku naciśnięcia przycisku CLEAR program najpierw usuwa kolor każdego elementu tablicy na interfejsie użytownika, a następnie tworzy tablicę zawierającą każdy element w wartościami kolorów ustawione na 0. Taka komenda gasi wszystkie ledy na emulatorze.

```
124       /**
125        * @brief Creates string of data needed for POST request
126        * @return stringData String containing the data in "LEDxy=[x,y,r,g,b]&&..." format or null if
       no cells are colored
127        */
128        function getStringData(){
129          var stringData='';
130          var element;
131          var color;
132          var JSONArrayElement;
133          for(var i=0; i<8; i++){
134            for(var j=0; j<8; j++){
135              element = document.getElementById("led"+i+j);
136              color=element.style.backgroundColor;
137              if(color.length!=0){
138                stringData=stringData+"LED"+i+j+'=';
139                JSONArrayElement=getJSONArrayElementInString(i, j, color);
140
141                stringData=stringData.concat(JSONArrayElement, '&&');
142              }
143            }
144          }
145          if(stringData.length>1){
146            stringData=stringData.substring(0,stringData.length-2);
147            return stringData;
148          }
149          else return null;
150        }
```

```

```

Listing 13: Kod ledmatrix.js pobieranie danych z tablicy i tworzenie zmiennej typu string do przesłania metodą POST

```
/**
* @brief Generates the string code in POST form, and then sends POST request to setled.php
*/
function clearColors(){
  var ledId;
  var element;
  var postRequestText = "";
  for(var i=0; i<8; i++){
    for(var j=0; j<8; j++){
      ledId="led"+i+j;
      element = document.getElementById(ledId);

      element.style.backgroundColor='rgb(0, 0, 0)';

      postRequestText = postRequestText + 'LED'+i+j+'=['+i+','+j+',0,0,0]&&';
    }
  }
  postRequestText=postRequestText.substring(0,postRequestText.length-2);
  debugHelpVariable=postRequestText;
  $.ajax(url, {
    type: "POST",
    data: postRequestText,
    dataType: "text",
    crossDomain: true,
    beforeSend: function(x) {
      console.log("AJAX POST REQUEST: BEGIN SENDING");
    },
    success: function(result) {
      console.log("AJAX POST REQUEST: SUCCESFULL CODE: " + result);
    },
    error: function(XMLHttpRequest, textStatus, errorThrown) {
      console.log("AJAX POST REQUEST: FAILURE");
      console.log("STATUS: " + textStatus);
      console.log("ERROR: " + errorThrown);
    },
    cache: false
  });

}

```

Listing 14: Kod ledmatrix.js wysłanie rozkazu zgaszenia wszystkich diod LED

3.6 Strona wykresu położenia joysticka

W ten sam sposób co wykresy temperatury, ciśnienia, wilgotności i położenia kątowego wykres położenia joysticka został sporządzony za pomocą biblioteki Chart.js. W odróżnieniu do pozostałych wykresów ten wykres jest typu "bubble". Na listingu 15 przedstawiono inicjalisację wykresu, a na listungu 16 aktualizację wykresu.

```
  /**
  * @brief Chart initialization
  */
  function chartInit()
  {

    // empty array
    jData = [{x:0,y:0,r:10}];

    // get chart context from 'canvas' element
    jChartContext = $("#jChart")[0].getContext('2d');

    jChart = new Chart(jChartContext, {
      // The type of chart: bubble plot
      type: 'bubble',

```

```
            // Dataset: 'xdata' as labels, 'ydata' as dataset.data
            data: {
              datasets: [{
                label: 'Joystick Position chart',
                backgroundColor: "rgba(0,0,0,0.2)",
                borderColor: "#000",
                data:jData
              }]
            },

            // Configuration options
            options: {
              legend: {
                display: false
              },
              responsive: true,
              maintainAspectRatio: false,
              animation: false,
              scales: {
                yAxes: [{
                  scaleLabel: {
                    display: true
                  },
                  ticks: {
                    min: -10,
                    max: 10
                  }
                }],
                xAxes: [{
                  scaleLabel: {
                    display: true
                  },
                  ticks: {
                    min: -10,
                    max: 10
                  }
                }]
              }
            }
          });


          jData = jChart.data.datasets[0].data;
        }

```

Listing 15: Kod joystick.js inicjalizacja wykresu

```
        /**
        * @brief Add new value to next data point.
        * @param y New y-axis value
        */
        function updateData(_x, _y, _z){

          jData=[{x:_x, y:_y, z:10}];
          jChart.data={
            datasets: [{
              label: 'Joystick Position chart',
              backgroundColor: "rgba(0,0,0,0.2)",
              borderColor: "#000",
              data:jData
            }]
          }
          jChart.update();

          $("#zLevel").text(_z.toString());
        }

```

Listing 16: Kod joystick.js aktualizacja wykresu

4 Implementacja systemu - Aplikacja desktopowa

5 Implementacja systemu - Aplikacja mobilna

6 Wyniki testów i integracji systemu

7 Wnioski i podsumowanie

8 Źródła i linki

1. Github [online 2020]: https://github.com/BotomQuark/AMProjekt/tree/master/RaspberryPi/http
2. Prezentacja części webowej v1.0 [online 2020]: https://www.youtube.com/watch?v=eH1l82eWoxg