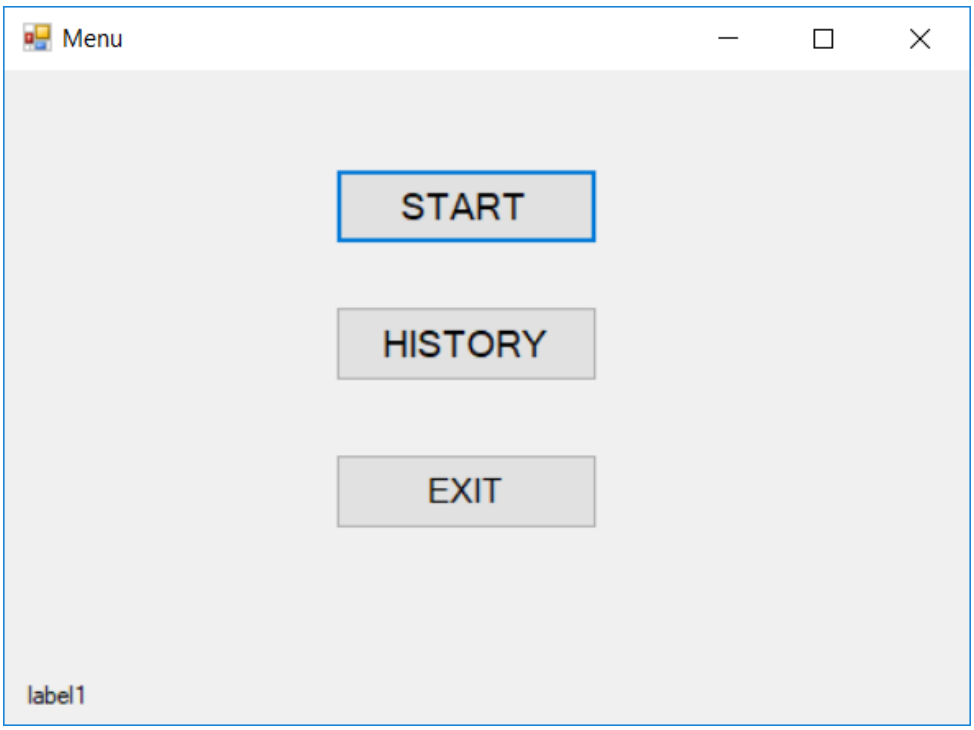
Wykonawcy: Maciej Korcz, Michał Ochniowski, Paweł Górski
Grupa: 33INF-SSI-SP/A
Moduł: Komunikacji przez bezprzewodowy interfejs komunikacyjny Bluetooth z częścią pomiarowo-sterującą systemu

1. Wykorzystane technologie:
   Symulator wykonywany jest przy użyciu języka C# oraz platformy .NET (Visual Studio 2019). Do komunikacji między członkami zespołu wykorzystywane są m. in. Discord oraz TeamSpeak 3. Cele organizacyjne obsługiwane są natomiast przez serwisy GitHub oraz ZenHub.

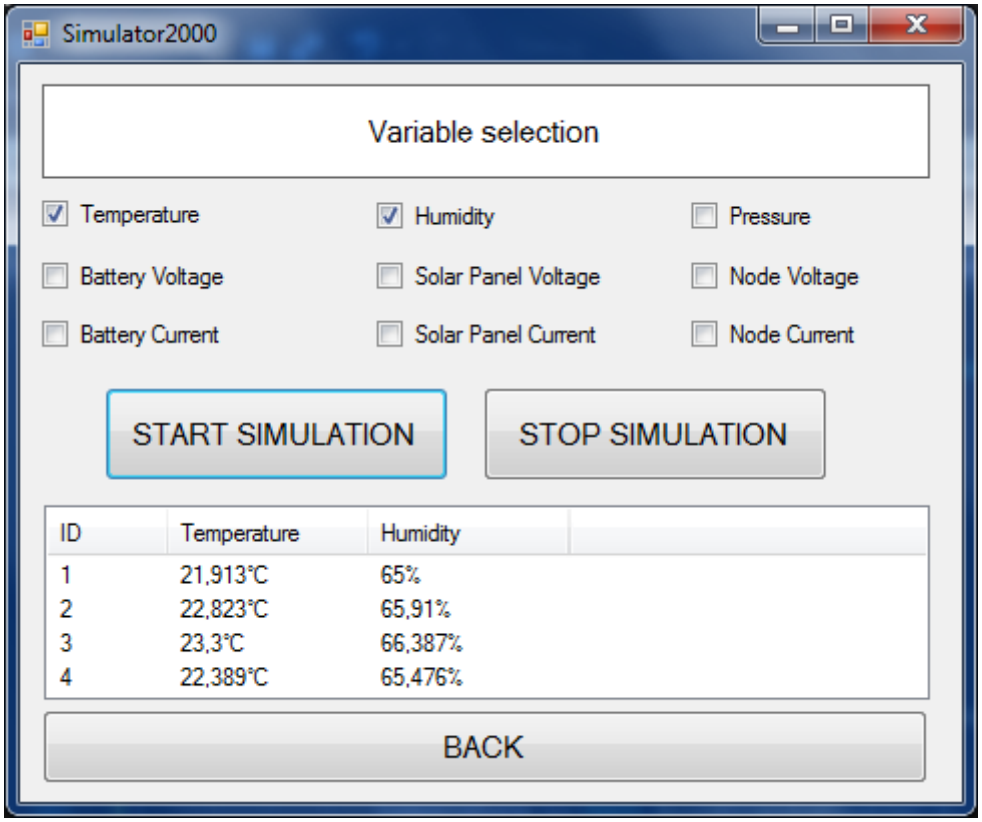
2. Opis postępu prac.
   Zostały wprowadzone drobne zmiany i poprawki w działaniu symulatora.



Rys. 1: Menu główne

W menu głównym (Rys. 1) został dodany przycisk „HISTORY”, po którego kliknięciu wyświetlane będzie nowe okno, w którym będzie można przejrzeć historię zapisanych pomiarów.

W oknie symulacji (Rys.2) pomiarów został nieznacznie zmieniony interfejs. Dodany został także przycisk powrotu, który pozwala na powrót do menu głównego.



Rys. 2: Symulacja pomiarów

Po wciśnięciu przycisku „START SIMULATION” wyniki pomiarów aktualizowane są co pięć sekund przy pomocy nowych wartości - te ukazują się w kontrolce „ListView”. Symulowane są jedynie wielkości pomiarowe, przy których zaznaczone są checkboxy. Po wciśnięciu przycisku „STOP SIMULATION” symulacja pomiarów zostaje przerwana, a wyniki przestają pojawiać się na liście. Po ponownym wciśnięciu przycisku „START SIMULATION” lista jest czyszczona oraz pojawiają się na niej nowe wartości.

```
1 odwołanie
private void buttonStart_Click(object sender, EventArgs e)
{
    Random random = new Random();
    listViewResult.Columns.Clear();
    listViewResult.Items.Clear();
    listViewResult.Columns.Add("ID");
    ListViewItem item = new ListViewItem(id.ToString());

    buttonStart.Enabled = false;
    foreach (Control c in Controls)
    {
        if(c is CheckBox)
        {
            c.Enabled = false;
        }
    }

    if (checkTemp.Checked)
    {
        listViewResult.Columns.Add("Temperature", 100);
        temperature = Math.Round(randomDouble(-30, 50), 3);
        item.SubItems.Add(temperature.ToString() + " °C");
    }

```

Rys. 3: Aktualizacja stanu kontrolek oraz tworzenie kolumn

Każde włączenie symulacji powoduje utworzenie nowych kolumn z wybranymi przez użytkownika wartościami pomiarowymi. Blokowany jest także stan poszczególnych kontrolek w celu uniemożliwienia użytkownikowi doboru nowych wielkości podczas trwania symulacji. Po zakończeniu procesu dane w tabelach nie są usuwane aż do momentu rozpoczęcia nowej symulacji, co w przyszłości wykorzystane zostanie w celu ich zapisu oraz przesłania do bazy danych.

```
1 odwołanie
129     private void t_Tick(object sender, EventArgs e)
130     {
131         id++;
132         ListViewItem item = new ListViewItem(id.ToString());
133
134         if (checkTemp.Checked)
135         {
136             double temporary = temperature + Math.Round(randomDouble(-1, 1), 3);
137             while (temporary < -30 || temporary > 50)
138             {
139                 temporary = temperature + Math.Round(randomDouble(-1, 1), 3);
140             }
141             temperature = temporary;
142             item.SubItems.Add(temperature.ToString() + " °C");
143         }
144
145         if (checkWilg.Checked)
146         {
147             double temporary = humidity + Math.Round(randomDouble(-1, 1), 3);
148             while (temporary < 0 || temporary > 100)
149             {
150                 temporary = humidity + Math.Round(randomDouble(-1, 1), 3);
151             }
152             humidity = temporary;
153             item.SubItems.Add(humidity.ToString() + "%");
154         }
```

Rys. 4: Symulacja pomiarów (Timer)

Aktualna wartość poszczególnych wielkości pomiarowych zrealizowana została przy użyciu klasy Timer (Rys. 4). W momencie, gdy kolejna wylosowana liczba spowodowałaby wykroczenie poza przyjęty zakres, losowana jest nowa wartość.