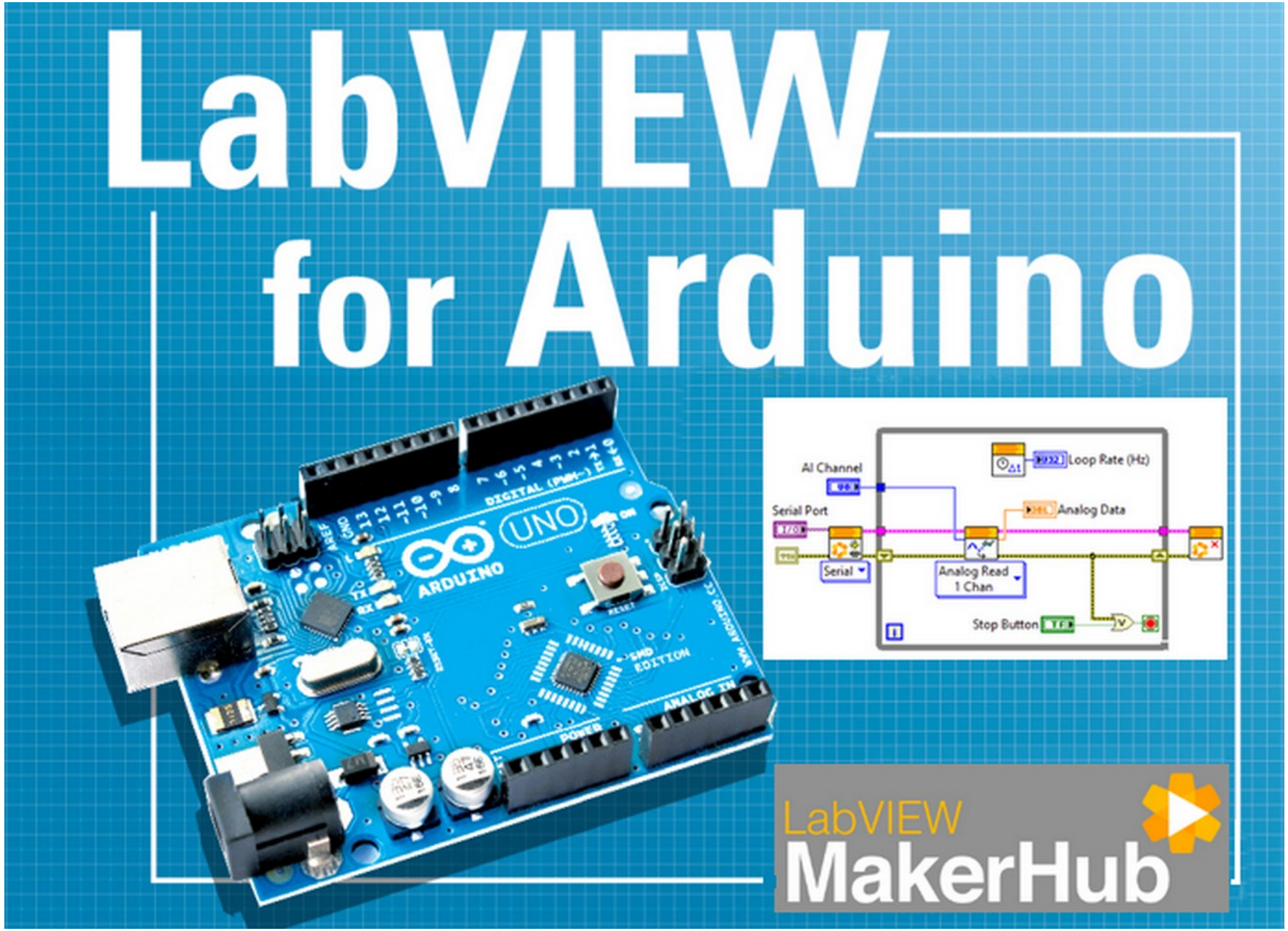
17. LabVIEW + LINX + Arduino - 7. rész



Felhasznált és ajánlott irodalom

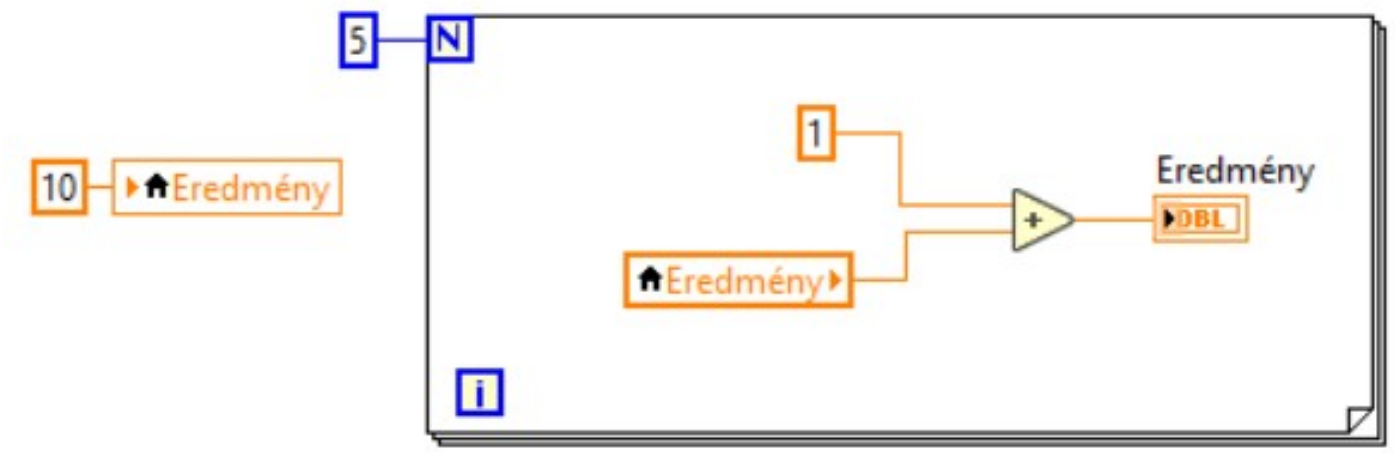
- NI: Getting Started with Arduino and LabVIEW Community Edition
- NI: LabVIEW Documentation
- Szabó Norbert: LabVIEW bevezető
- Jáger Attila: LabVIEW alapismeretek
  1. fejezet, 2. fejezet, 3. fejezet, 4. fejezet, 5. fejezet, 6. fejezet
- Friedl Gergely: LabVIEW segédlet
- Jeffrey Travis, Jim Kring: LabVIEW for Everyone (3rd Edition)

- Gremmédia: Mi az a JSON formátum?
- NI: Creating Local Variables
- NI: JSONtext download
- NI: Event-Driven Programming in LabVIEW
- OpenWeatherMap: Regisztráció, API

Lokális, globális és megosztott változók

- Eddig az előlapi objektumok blokkdiagram terminálján keresztül olvastunk vagy írtunk adatokat. Egy előlapi objektumnak azonban csak egy terminálja van a blokkdiagramon, és előfordulhat, hogy frissíteni vagy olvasni kell egy előlapi objektumot a blokkdiagram több helyén is - ehhez kellenek a változók
- **A helyi változó** (Local Variable) lehetőséget biztosít az előlapi objektumok elérésére a VI blokkvázlatának több helyéről olyan esetekben, amikor nem tud vagy nem akar vezetéket csatlakoztatni az objektum termináljához, vagy ha írni és olvasni kell
- A **globális változók** hasonlóak a lokális változókhoz, de több VI között is adhatnak át értéket
- A **megosztott változók** hasonlóak a globális változókhoz, de több helyi és hálózati alkalmazás között adhatnák át értéket. Ezenkívül további szolgáltatásokat is biztosítanak: pufferelhetik az adatokat, és segítenek elkerülni a globálisoknál előforduló szinkronizálási problémákat
- A mai előadásban csak lokális változókkal találkozunk

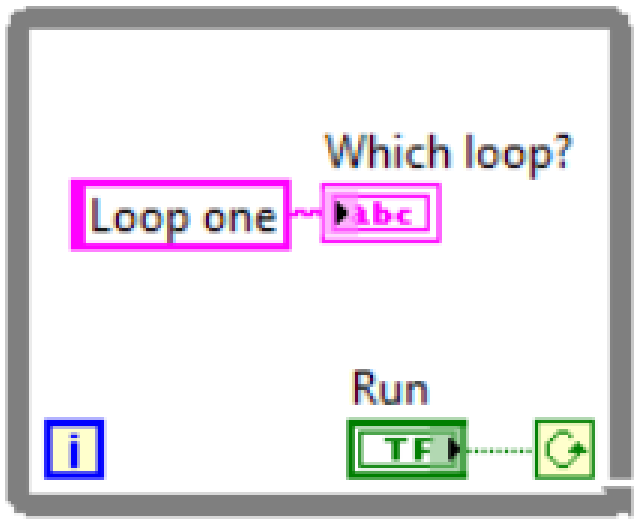
Lokális változó használata

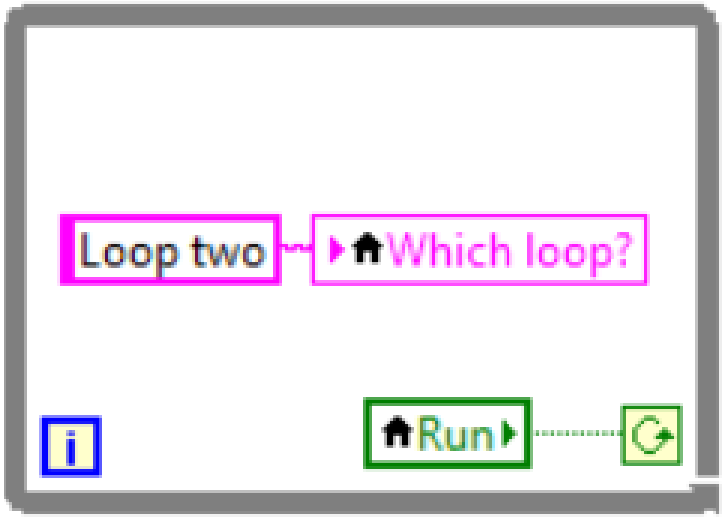


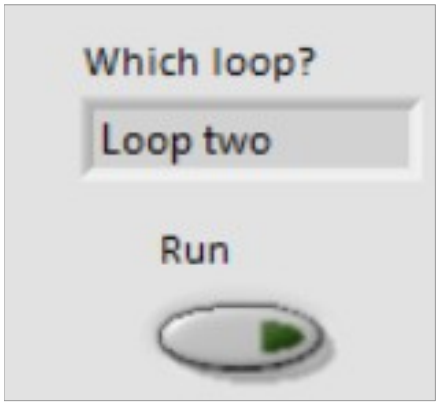
1) Hozzunk létre az előlapon (Front Panel) egy kijelzőt (**Create Indicator** a jobbklikk menüben) és legyen a neve **Eredmény!**
2) Kattintsunk jobb gombbal a kijelző elemre és válasszuk ki a felbukkanó menüben a **Create → Local Variable** menüpontot!
3) A blokkdiagramon a lokális változóhoz rendeljünk konstans értéket (pl. 10)!
4) A **Front Panel** ablakba visszatérve ismételjük meg a lokális változó létrehozását
5) A blokkdiagramon az új lokális változóra kattintsunk a jobb egérgombbal és a felbukkanó menüben válasszuk a **Change to Read** menüpontot!
6) Ezt az új példányt bemenő adatként használjuk a FOR ciklusban

Versenyfutási probléma lokális változókkal

- Egy lokális változó több helyen is kaphat értéket, illetve több helyen is felhasználhatjuk az értékét
- Az alábbi esetben ez versenyfutási helyzethez vezet, mert a két **While ciklus** párhuzamosan fut, így véletlenszerű, hogy melyik érték íródik utoljára a **Which loop?** kijelzőre







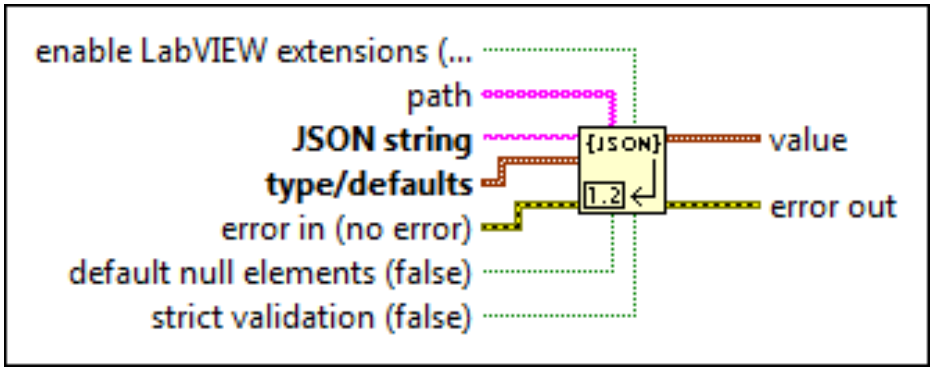
- Writing to One Front Panel Object from Two Block Diagram Locations
  www.ni.com/docs/en-US/bundle/labview/page/lvhowto/writing_to_one_front_panel.html

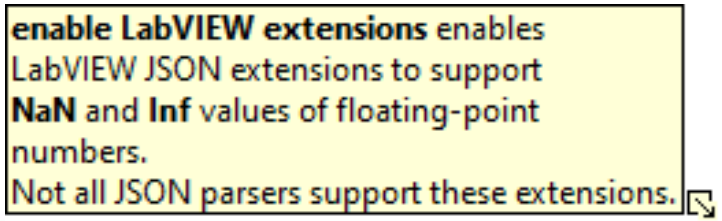
A JSON formátum

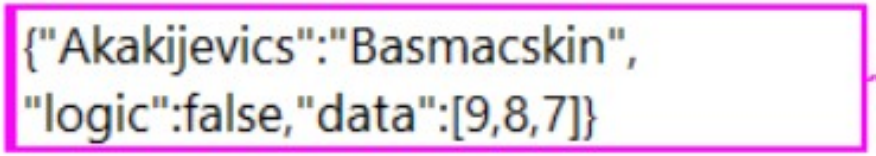
- **JSON** (JavaScript Object Notation) szöveg alapú formátum ami C, C++, C#, Java, JavaScript, Perl, Python és más nyelveken könnyen kezelhető
- A **JSON objektum** kulcs-érték párokból áll, és {} zárójelek veszik körül
  - **Példa:** `{ "name":"John","age":31,"city":"New York" }`
- Az „érték” lehet string, szám, logikai érték, de akár tömb is
  - **Példa:** `{"name":"John","languages":["C++", "Python","Java"]}`
- A **JSON tömb** elemeit [] jelek veszik körül, és **JSON objektumok** alkotják, melyeket vesszővel kell elválasztani
  - **Példa:** `[{"name":"John","age":31},{"name":"Bill",”age”:45}]`
- A **JSON egymásbaágyazott** (vagy hálózott) **objektum** némelyik kulcsának értéke egy másik JSON objektum
  - Példa: `{"firstName": "John","lastName": "Doe","gender": "male", "hobbies": {first":"biking","second":"climbing",”third":"TV" }}`

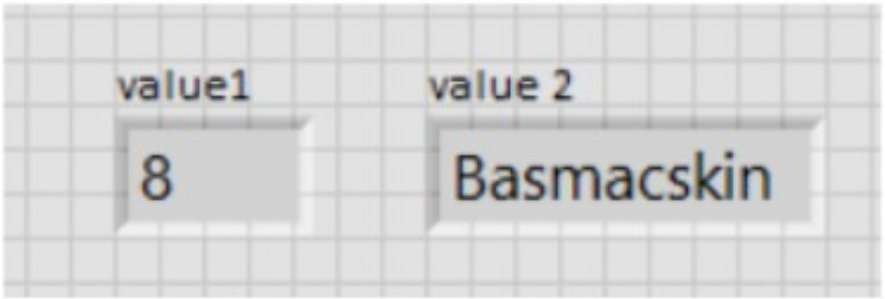
Az *Unflatten from JSON* függvény használata

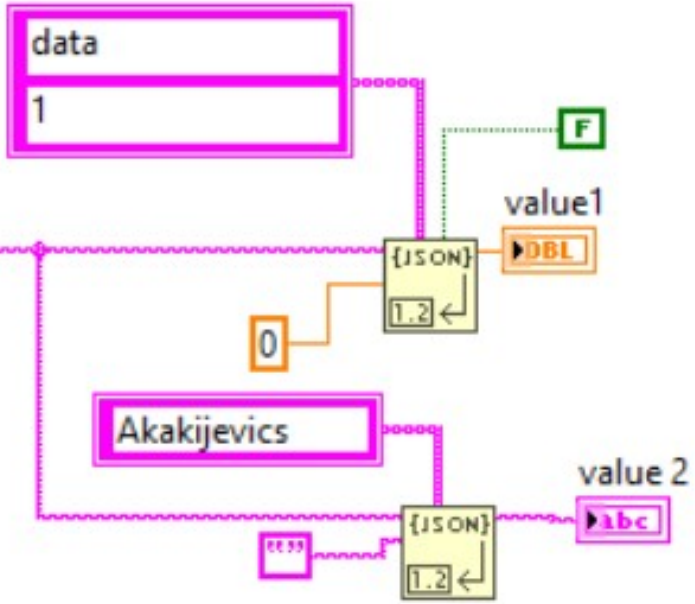
- **JSON** szövegek visszaalakításához legegyszerűbb a beépített ***„Unflatten from JSON”*** függvény használata (***String → Flatten/Unflatten String*** paletta)
- A kiválasztott elem(eket) többféle módon is azonosíthatjuk (path, név, sorszám)
  Bővebben: Unflatten From JSON Function
- Itt a „data” címkéjű tömb 2. elemét, és az „Akakijevics” cimkéjű stringet adtuk meg





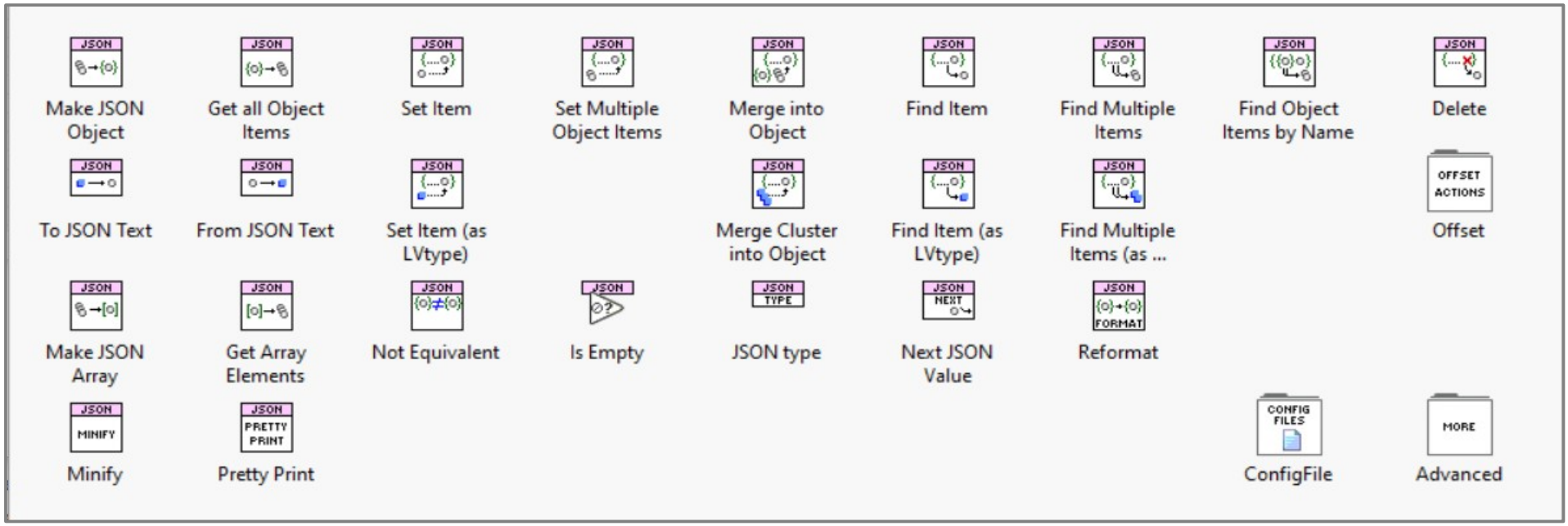






A JSONText bővítmény

- A **JSONText** egy **LabVIEW** szoftverbővítmény (Add-On), ami egy könyvtárat biztosít a **JSON** nyílt szabványú fájl- és adatcseréhez
- Ez a bővítmény a **JSON Path** jelölési forma segítségével teszi hozzáférhetővé az al-elemeket. A **JSONText**-et használhatjuk az al-elemek **LabVIEW** adattípusokká vagy azokból történő konvertálására is
- A **JSONText** telepítéshez először az **NI Package Manager** segítségével telepítenünk kell a **JKI VI Package Manager** (VIPM) programot, amely segít a **LabVIEW** bővítmények felfedezésében és telepítésében



JSON mintafájlok

- A **JSON** fájlok beolvasásához és a programok kipróbálásához mellékeltük az alábbi fájlokat
- A **Debrecen.json** állomány egy aktuális időjárási adat lekérésének eredménye az **OpenWeatherMap** szerverről

simple.json

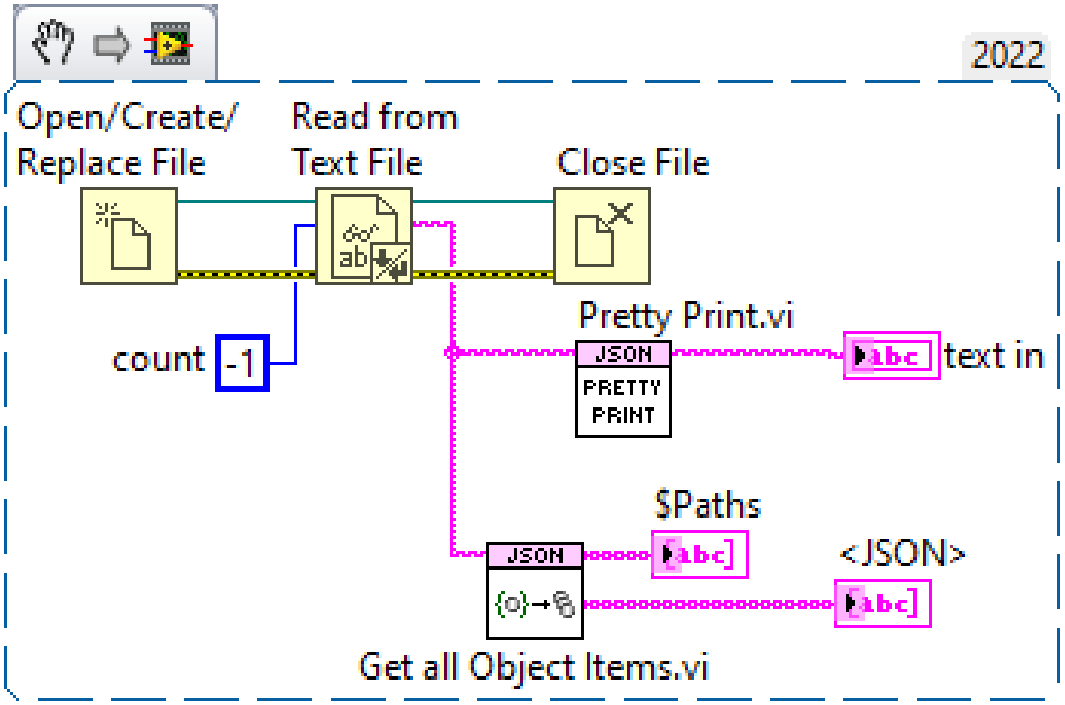
```
{
    "firstName": "John",
    "lastName": "Doe",
    "email": "john.doe@example.com",
    "age": 45,
    "weight": 67,
    "admin": true
}
```

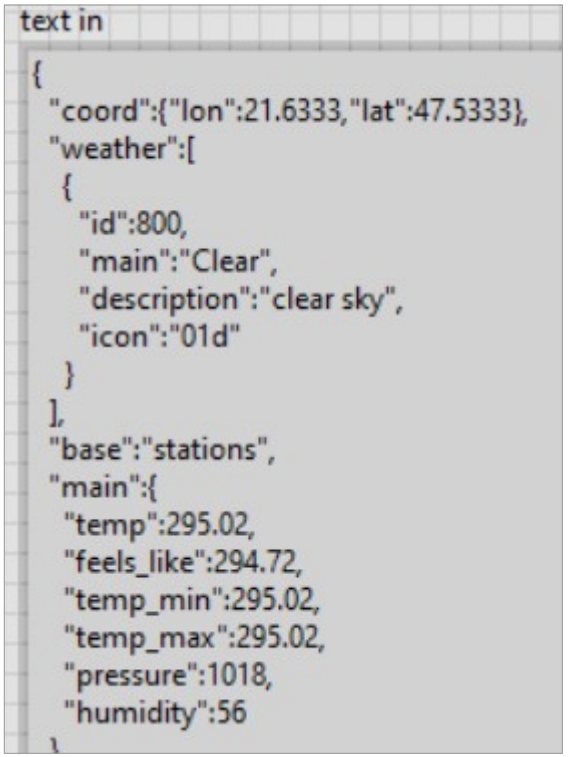
Debrecen.json


```
{
  "coord":{"lon":21.6333,"lat":47.5333},
  "weather":[
    {
      "id":800,
      "main":"Clear",
      "description":"clear sky",
      "icon":"01d"
    }
  ],
  "base":"stations",
  "main":{
    "temp":295.02,
    "feels_like":294.72,
    "temp_min":295.02,
    "temp_max":295.02,
    "pressure":1018,
    "humidity":56
  },
  "visibility":10000,
  "wind":{"speed":4.63,"deg":40},
  "clouds":{"all":0},
  "dt":1685089162,
  "sys":{
    "type":1,
    "id":6665,
    "country":"HU",
    "sunrise":1685069115,
    "sunset":1685124960
  },
  "timezone":7200,
  "id":721472,
  "name":"Debrecen",
  "cod":200
}
```

JSON_fileread.vi

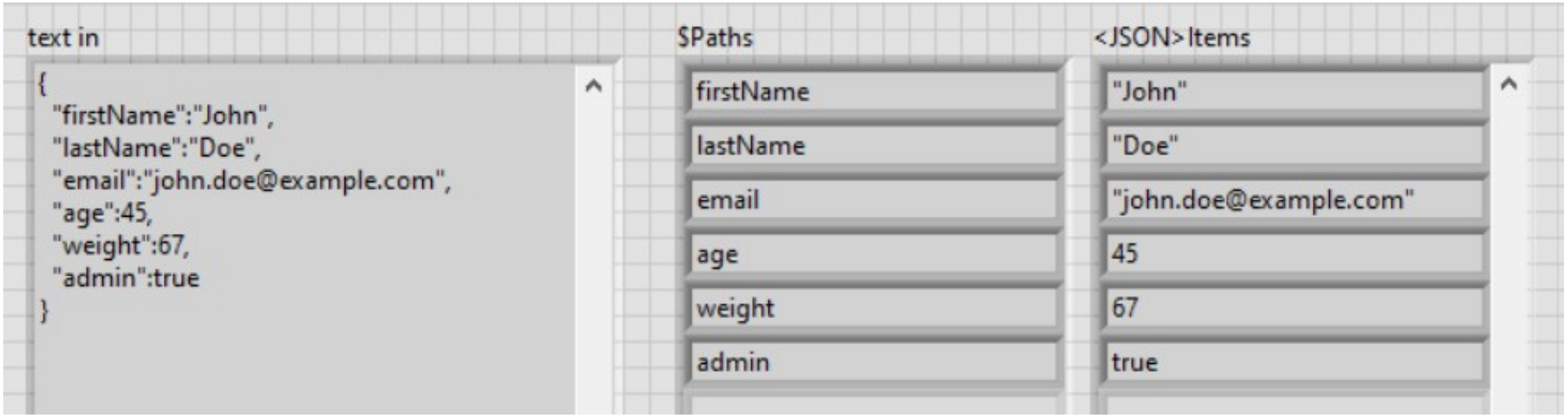
- Keressünk vagy készítsünk egy **JSON** fájlt és olvassuk be, s próbáljuk meg értelmezni a **JSONText** függvények segítségével!
- **File I/O** palettán találhatók az **Open, Close** és **Read from Text** funkciók
- A fájlt teljes egészében beolvassuk (count = –1)
- Az **Addons/JSONText** palettáról a **Pretty Print** és **Get all Object items** függvényeket használjuk, s az eredményül kapott szöveget, vagy string tömböt megjelenítjük





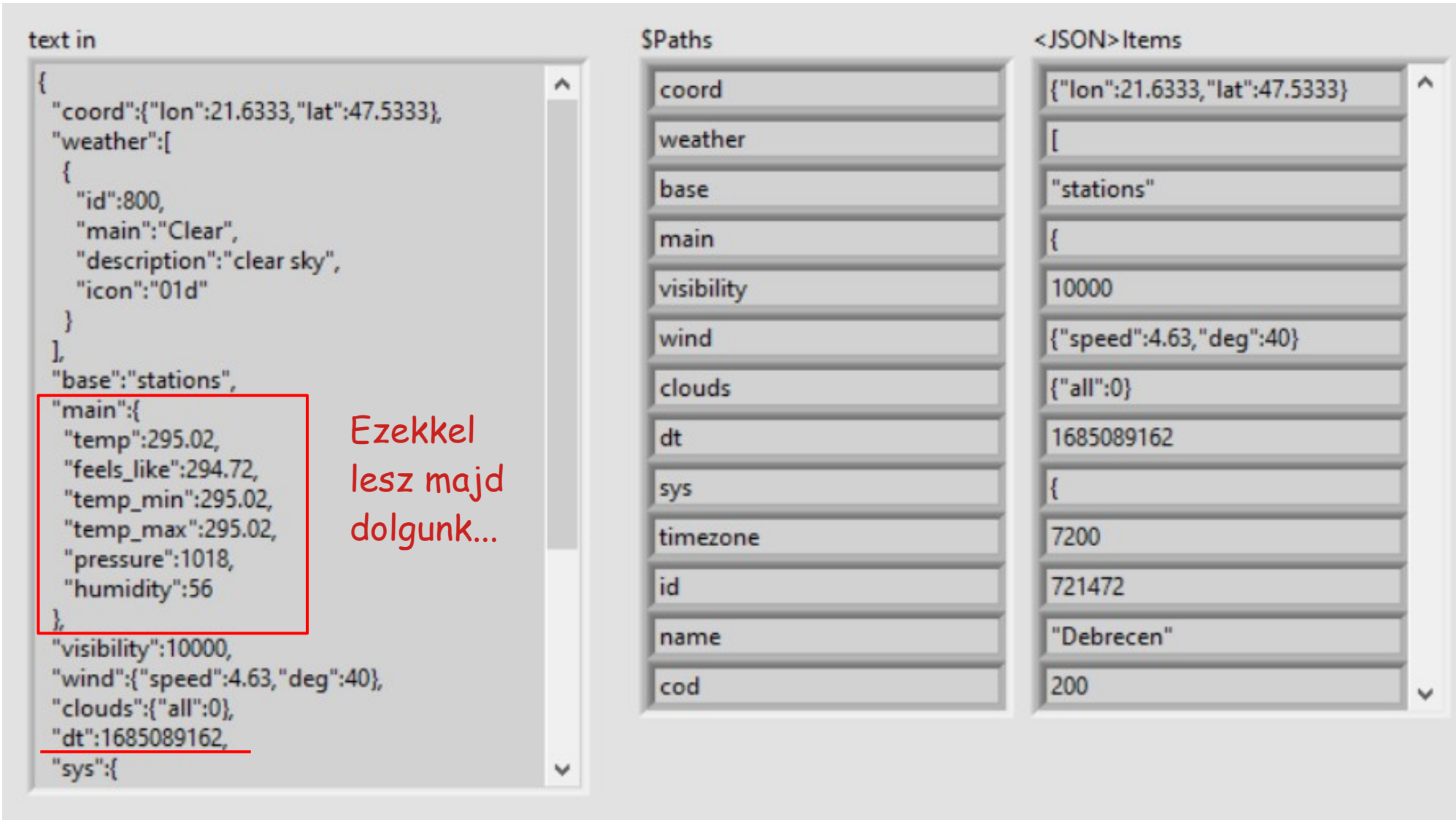
JSON_fileread.vi

- Az alábbi példában a mellékelt **simple.json** nevű állományt olvastuk be
- Mivel a fájl csupa **kulcs – érték** párból áll, így a **$Paths** és a **<JSON> Items** tömbök azonos sorszámú elemei az összetartozó kulcsot illetve értéket tartalmazzák
- Amint látható, a *string* típusú értékeket időzőjelben kapjuk meg, úgy, ahogy az eredeti **JSON** szövegben is szerepeltek



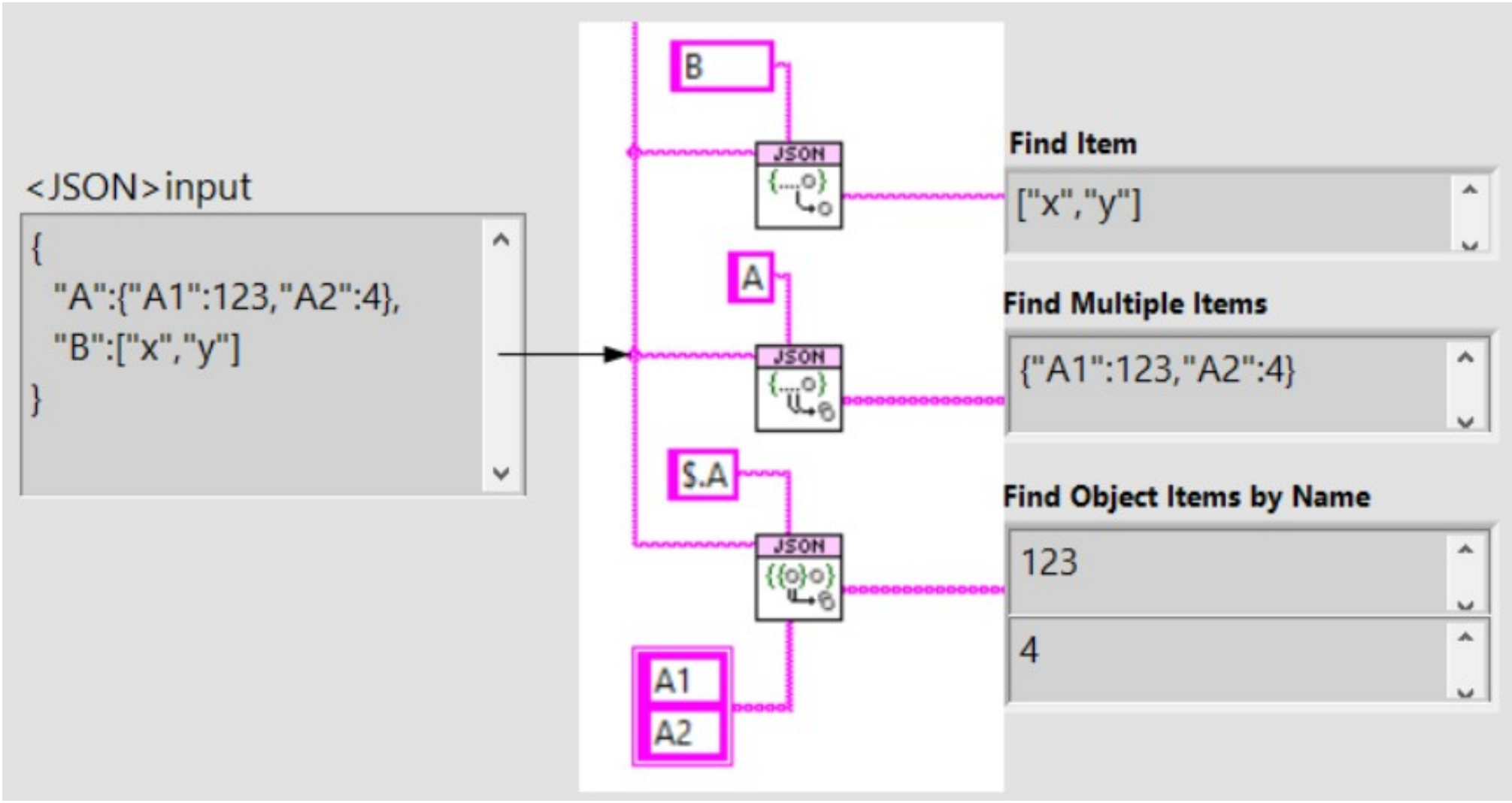
JSON_fileread.vi

- Itt egy **OpenWeatherMap** lekérés eredményét olvastuk be (Debrecen.json)
- Az összetett objektumoknak, sajnos, itt csak a kezdő zárójele jelenik meg…



JSONtext_demo1.vi

- Nézzünk néhány további függvényt a JSONtext palettáról!
  A felső paraméter egy **név** (kulcs), vagy **Path** lehet (ld, köv. oldal)



JSON Path jelölés a JSONtext-ben

- A **Path** egy elemet jelöl ki a behálózott elemek fáján és $ jellel kezdődik
  Minden elemkijelölés ***.name*** (objektum), vagy ***[n]*** (tömbindex) alakú

Path Elements:

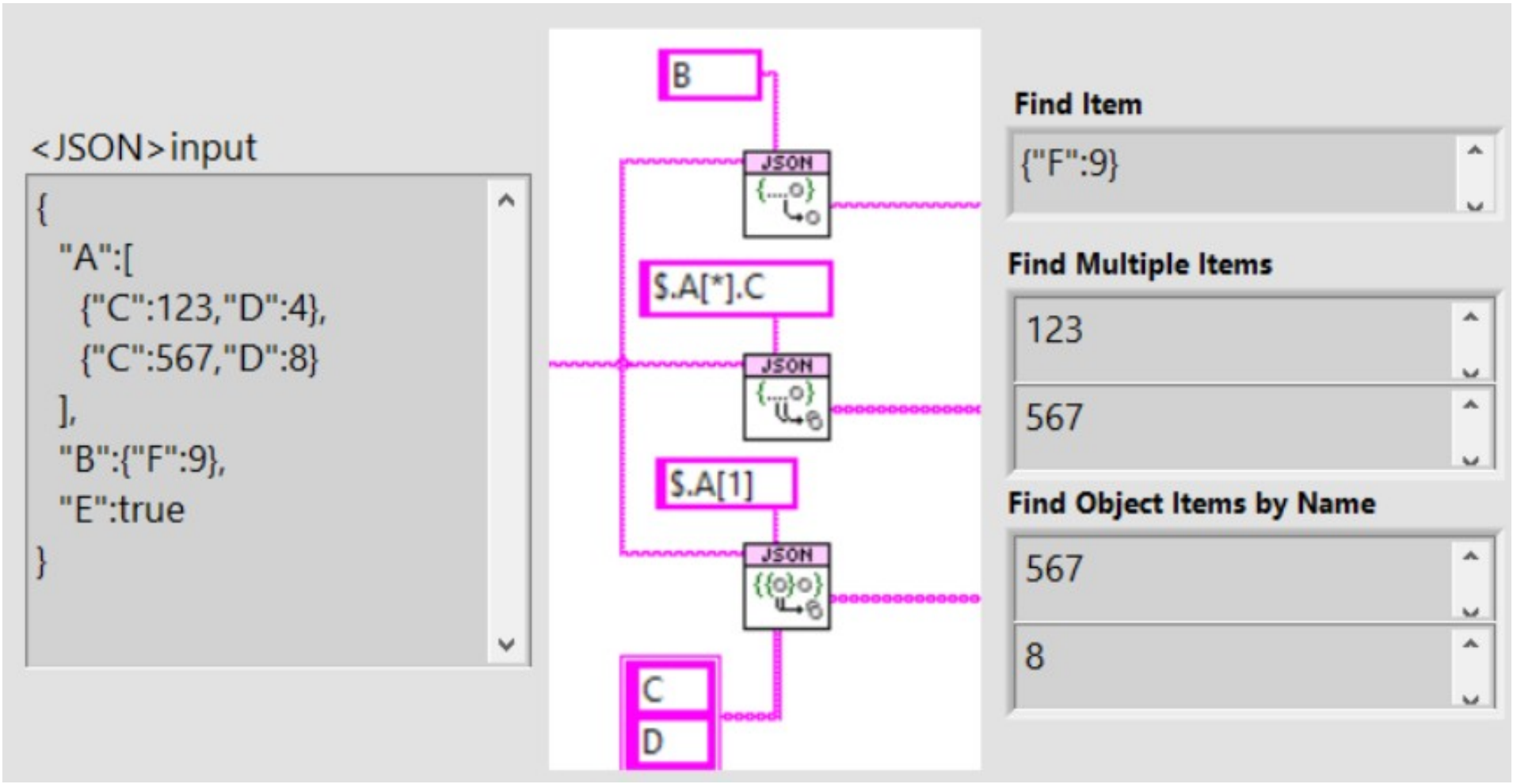
$	Root JSON Value
.name	Object item "name"
.[name1,name2,..]	Object items
.*	All Object items
[n]	Array index n (negative n counts back from end; -1 == last element)
[n1,n2,n3,...]	Multiple Array indexes
[*]	All Array elements
[#]	For appending to the Array, indicates element past the end
[#0]	For inserting to the Array at front

- Példa JSON: **{"A":[{"C":123,"D":4},{"C":567,"D":8}],"B":{"F":9},"E":true}**
- At path $.A[*].C are 123 and 567,
- At path $.B (or just B) is {"F":9}
- At path $.[B,E] are {"F":9} and true
- At path $.* are [{"C":123,"D":4},{"C":567,"D":8}], {"F":9}, and true

At path $.A[1].C is 567
At path $.B.F is 9

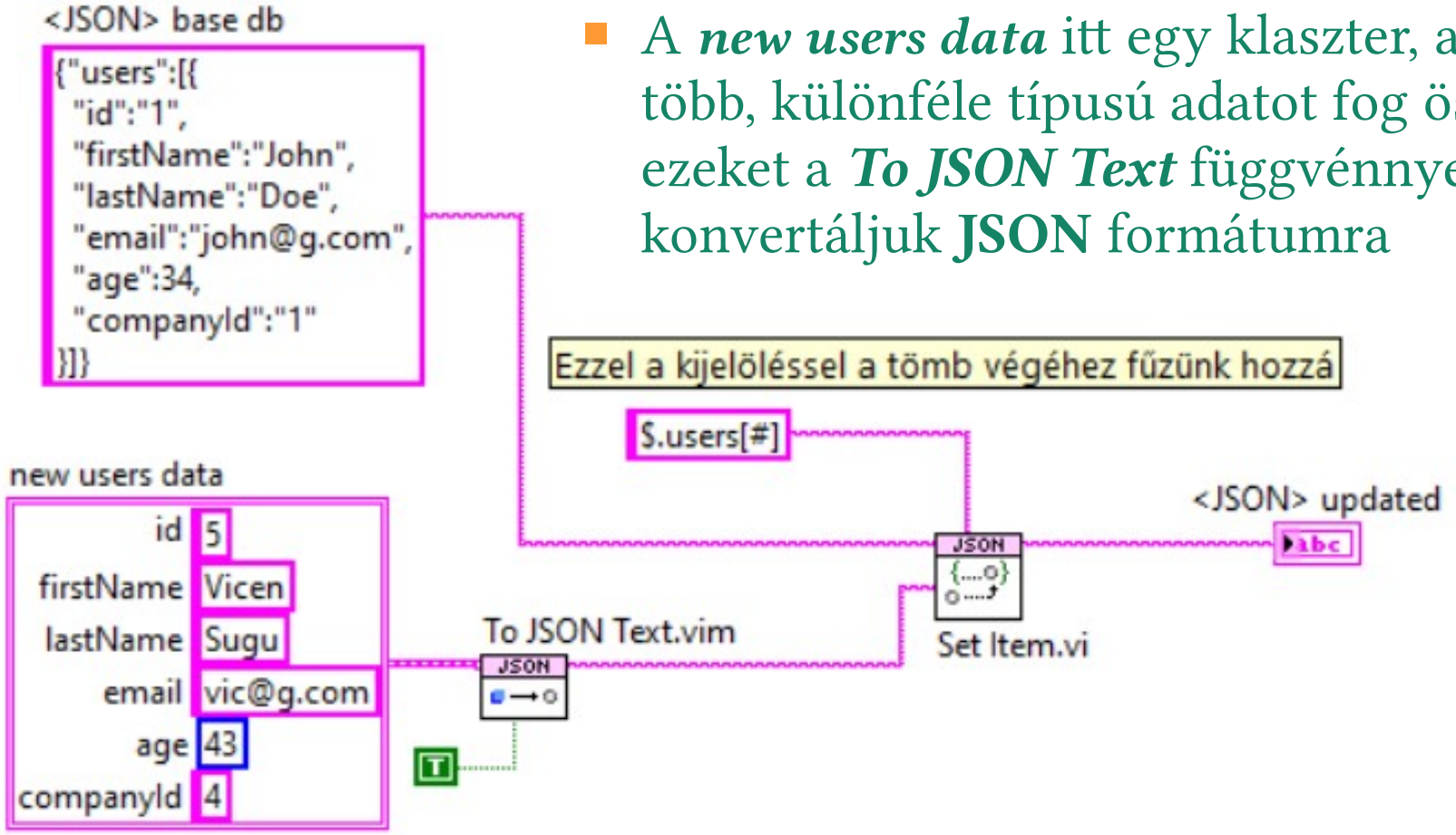
JSONtext_demo2.vi

- Nézzünk néhány példát a **JSON Path** megadására, amelyek a baloldali **JSON** szövegből a jobboldali elemeket állítják elő
- Vegyük észre, hogy $.A[1] a ***második*** tömbelemet jelöli ki!



Add data to JSON Mod ver2.vi

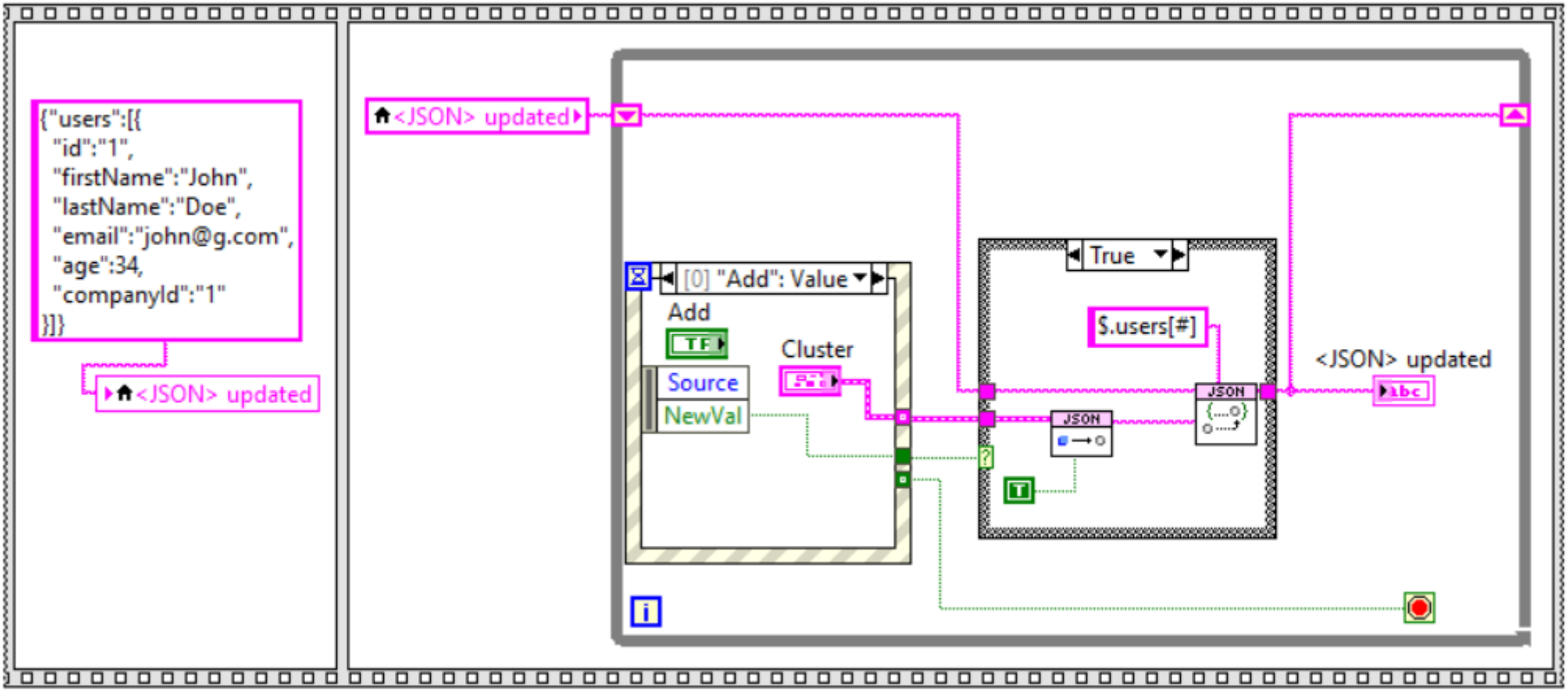
- Egy fórumon kérdezte valaki, hogy hogyan fűzhetünk hozzá egy újabb rekordot a kiindulási **JSON** szöveghez. A legfrappánsab megoldás ez volt: a `$.users[#]` Path megadásakor a **Set Item** függvény nem elemet helyettesít, hanem a végéhez hozzáfűz (Append)



  - A ***new users data*** itt egy klaszter, ami több, különféle típusú adatot fog össze, s ezeket a ***To JSON Text*** függvénnyel konvertáljuk **JSON** formátumra

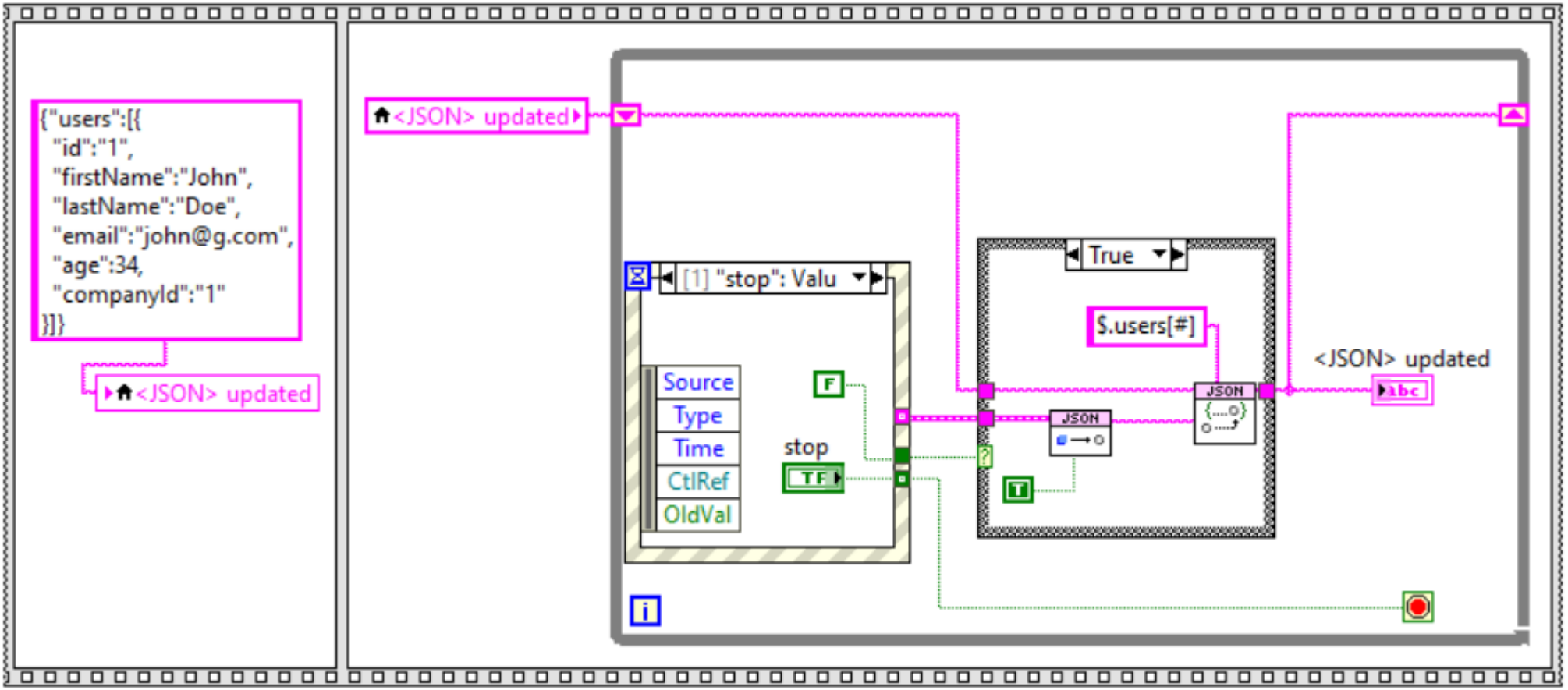
Add data to JSON manually.vi

- Az előző programot kis kiegészítéssel átalakíthatjuk kézi adatbeíráshoz (a korábbi ***new users data*** konstans helyett előlapi beírókat készítünk. Az legtöbb bonyodalmat a rekord hozzáadását indító gomb kezelése okozta, végül eseménykezelővel oldottuk meg, amit még egy IF feltételviszgálattal is meg kellett toldani (csak lenyomáskor kell aktiválni, felengedéskor nem)



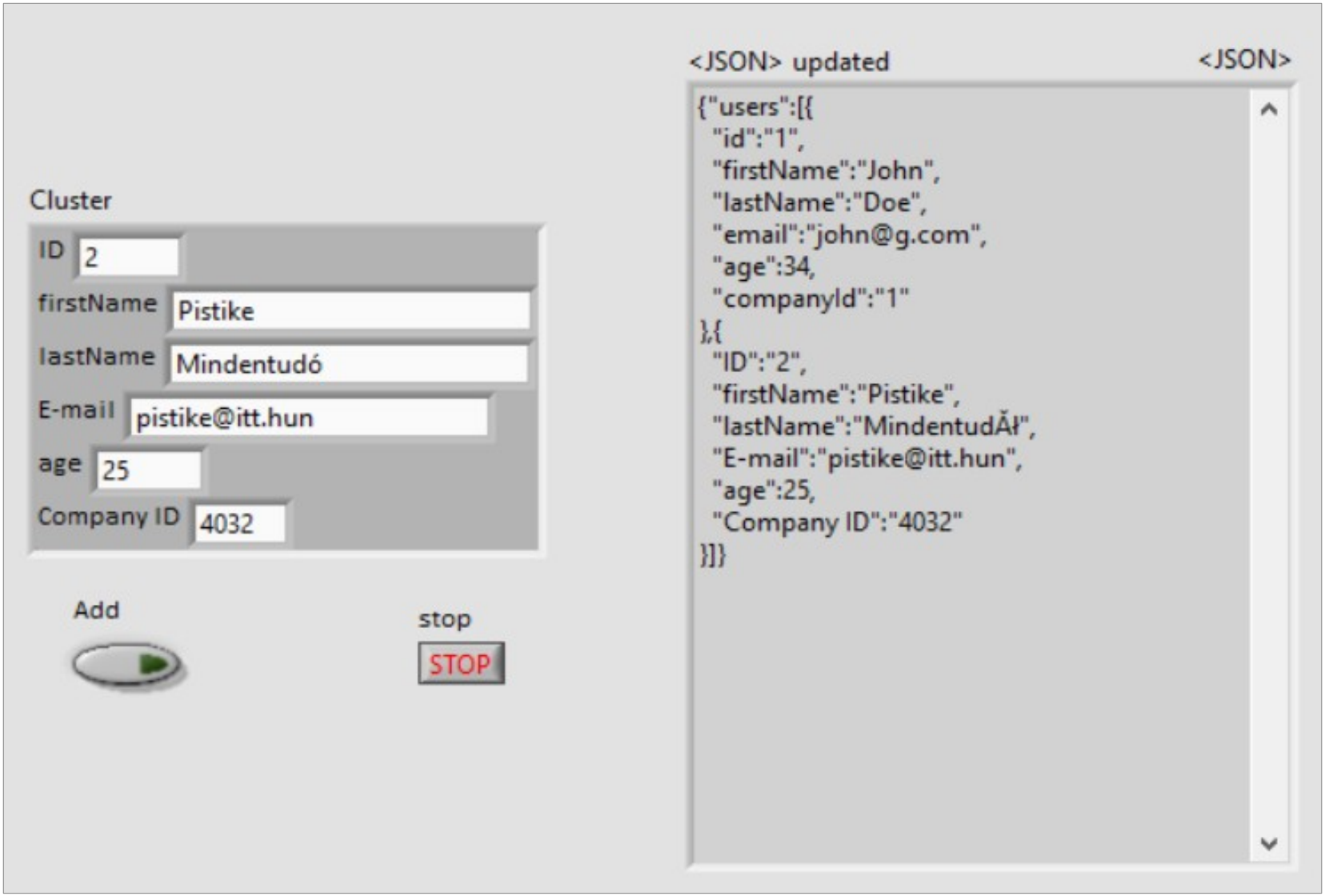
Add data to JSON manually.vi

- Ügyeljünk rá, hogy eseménykezelés használata esetén a **While ciklus** leállítását is az eseménykezelővel kell megoldani, tehát a **STOP** gombnak is fel kell venni egy új esetet
- Az **If** feltételvizsgálathoz ekkor *false*, a leállítóhoz *true* értéket küldünk



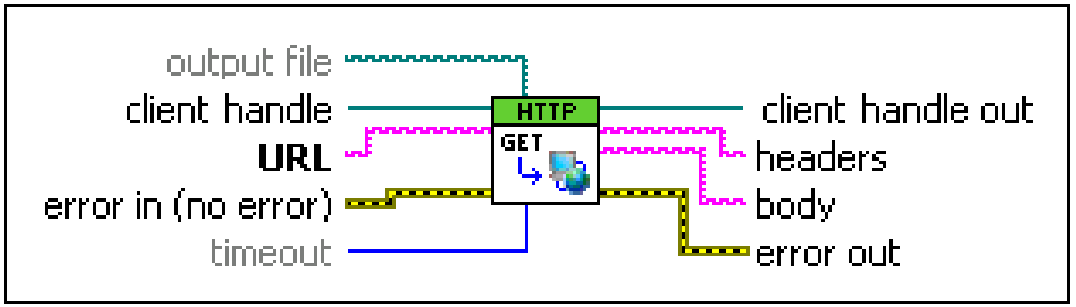
Add data to JSON manually.vi

- A program futási eredménye az ábrán látható. Az ékezetes karakterek kezelése még „hagy némi kívánnivalót maga után”...

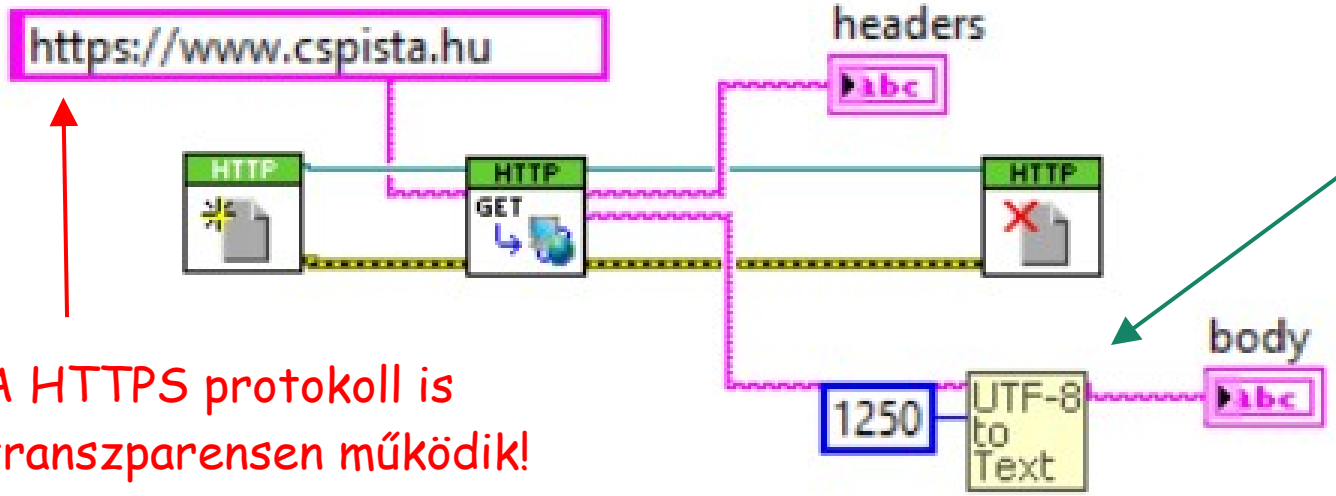


http_get_request.vi

- A **Data Communications » Protocols » HTTP Client** palettán találjuk a **HTTP kliens** kezeléséhez szükséges függvényeket
- A legegyszerűbb lekéréshez az alábbi függvényeket használjuk:
  - Open Handle
  - GET
  - Close Handle



- A *Client Handle* kezelése elsősorban a felhasználó azonosító és a sütik megőrzése szempontjából fontos, egyedi lekéréseknél elhagyható

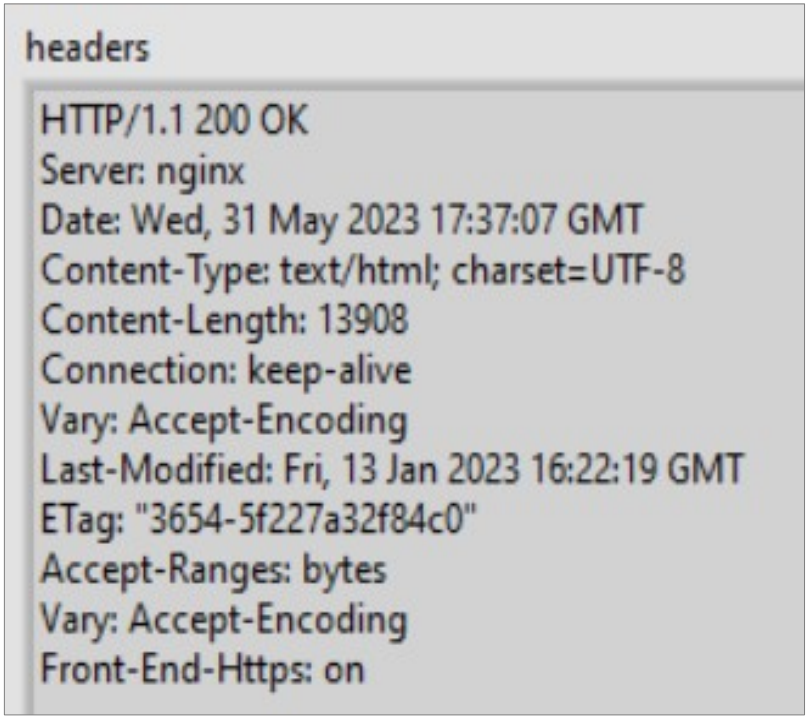


A HTTPS protokoll is transzparensen működik!

Windows esetén az UTF-8 kódolású szöveget át kell konvertálni ANSI 1250 (közép-európai) kódtáblára

http_get_request.vi

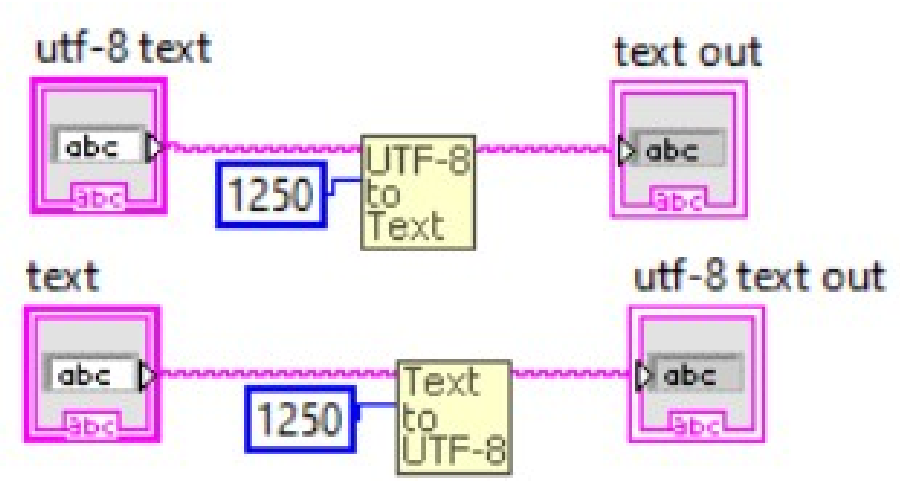
- A lekérés eredményeként megkapjuk az üzenet fejlécét és törzsét
- Sikeres lekérés esetén a **200 OK** kódot kapjuk vissza
- Az **ETag** (Entity Tag) a tartalom azonosítója. A böngésző újabb lekérésnél az **ETag** visszaküldésével jelezheti, hogy csak akkor kér új letöltést, ha megváltozott a tartalom (aminek már más lesz az **ETag** azonosítója)
  A **304**-es kód jelzi, ha a tartalom nem változott

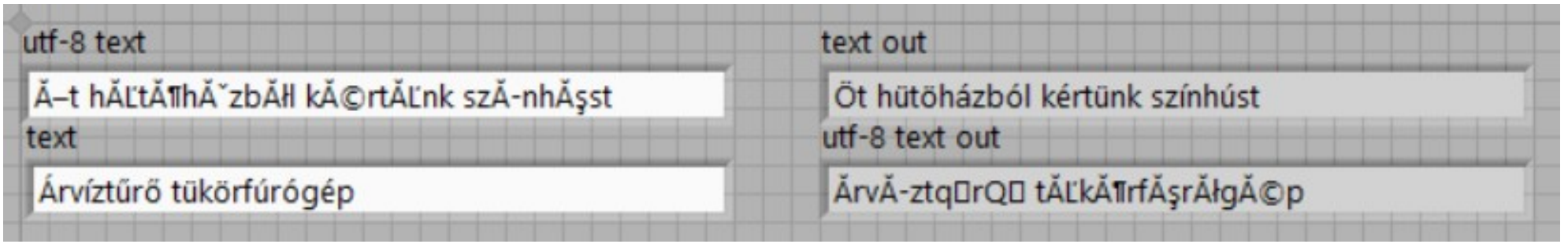



```
<html><head><meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
<title>Hobbielektronika csoport</title>
<link rel="stylesheet" href="assets/css/main.css" />
<link rel="stylesheet" href="./assets/css/fontawesome-all.min.css" />
</head>
<body class="is-preload">
<div id="wrapper"> <div id="main"> <div class="inner">
<header id="header">
<a href="index.html" class="logo"><strong>Hobbieletronika csoport</strong>
Debreceni Megtestesülés Plébánia, 4032 Debrecen, Borbíró tér 9.</a>
</header>

<h1>Hobbielektronika </h1>
<p>Szeretetel várunk minden érdeklődőt a Hobbielektronika csoport
foglalkozásaira a  Debreceni Megtestesülés Plébánia  közösségi termébe
szeptembertől júniusig, csütörtök délutánonként 17 órára!</p>
<p>Ingyenes tanfolyamokat és ismeretterjesztő előadásokat kínálunk.
Foglalkozásainkon írásvetítővel segített előadásokat tartunk, illetve egyszerű
kapcsolásokat állítunk össze és próbálunk ki (hozott anyagból).

<a href="2022.html" class="button big">Bővebb info</a>
. . . . .
```

UTF8 LV80.vi

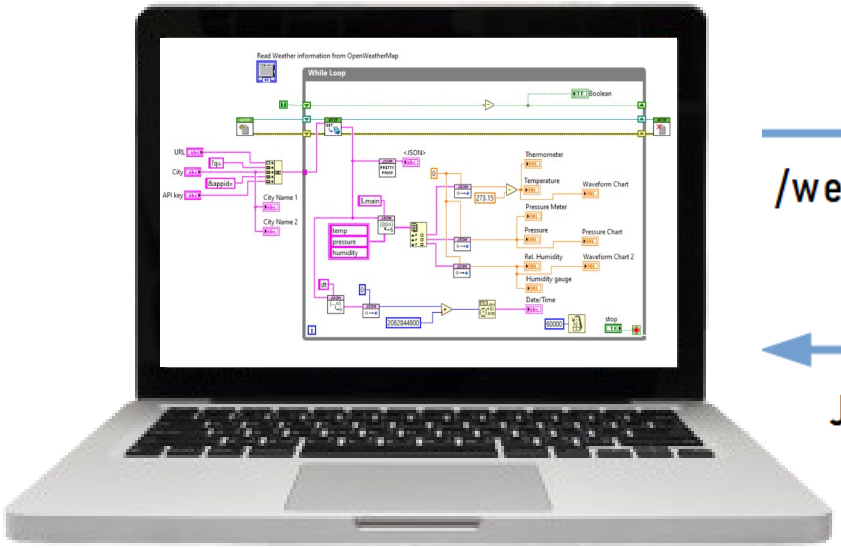
- **UTF-8 to Text** és **Text to UTF-8** konverzióra csak egy 2009-ből származó fórumbejegyzést találtam. Az abban közzétett VI (UTF8 LV80.vi) tartalmazza a konverziós függvényeket, amelyek többé-kevésbé ellátják a feladatot (az **ő** és **ű** betűknél lehetnek problémák). Ezeket át kell másolnunk a saját alkalmazásainkba, ha kellenek…
- Ugyanezt az **UTF-8 to Text** függvényt használtuk az előző programban is. Ott az 1250-es kódtábla megadásakor az *ő, ű* is jól konvertálódott

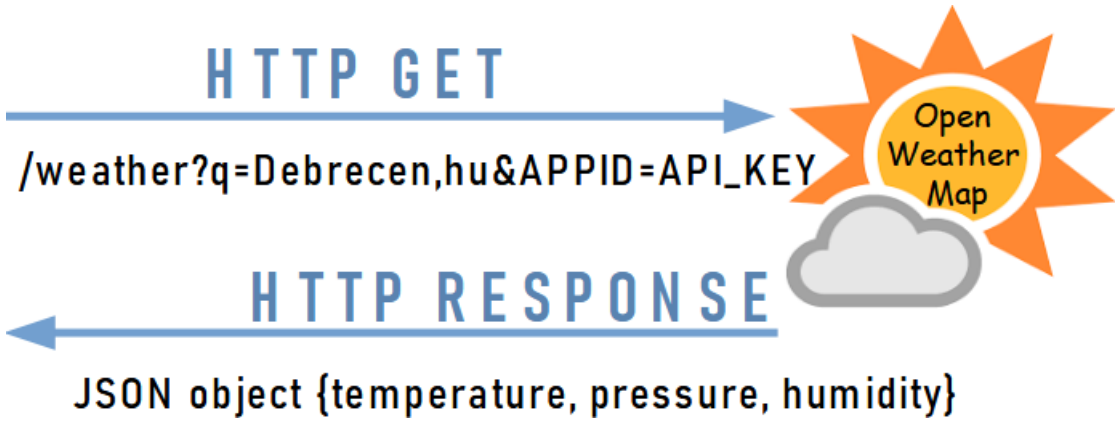




Időjárási adatok lekérése (OpenWeatherMap.org)

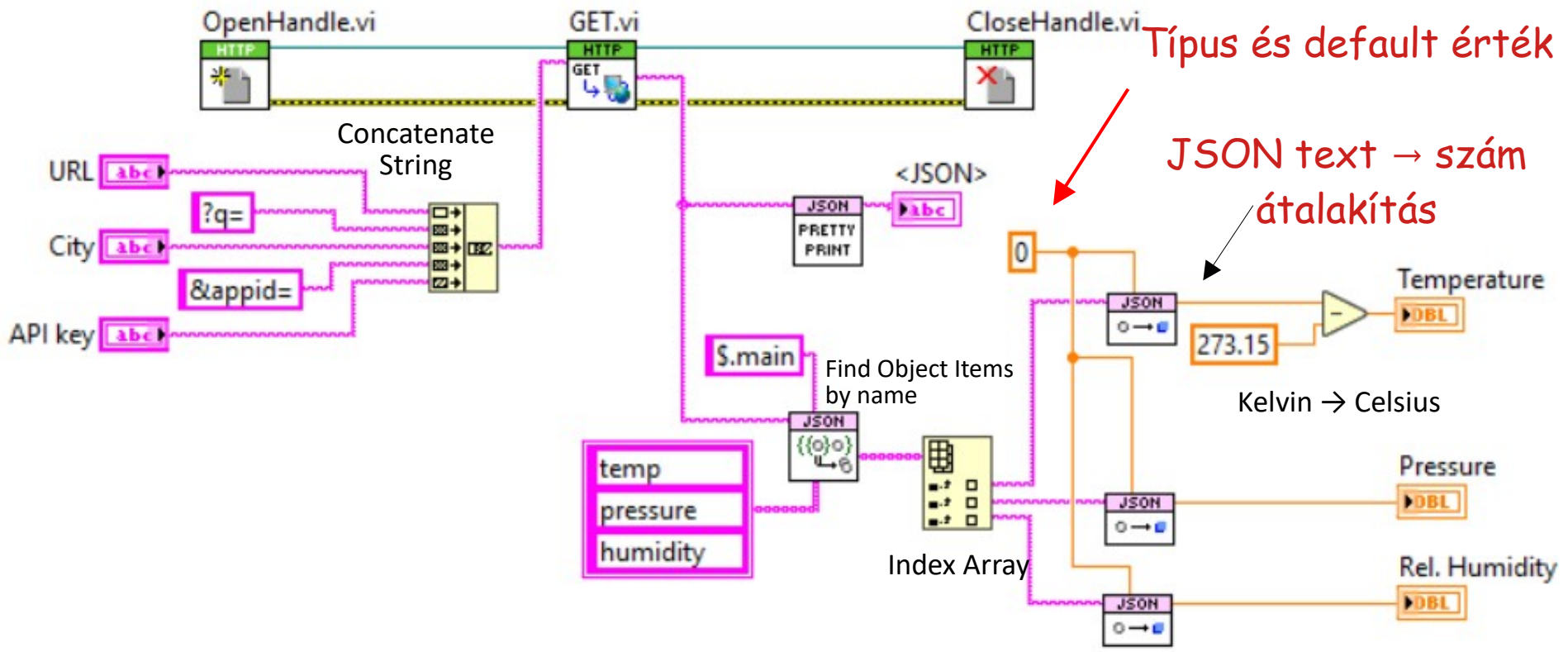
- Az időjárási adatokat HTTP kliensként tölthetjük le az api.openweathermap.org webszerverről
- Regisztráció (Sign in/Create Account) után az ingyenes szolgáltatásokat vehetjük igénybe (lásd Pricing, Free oszlop)
- Első bejelentkezéskor szerezzük meg és jegyezzük fel az API kulcsot!
- Mi most csak az **aktuális adatok** lekérdezésével foglalkozunk, például:
  `http://api.openweathermap.org/data/2.5/weather?q=Debrecen&appid=********`
- A választ alapértelmezetten JSON formátumban kapjuk meg





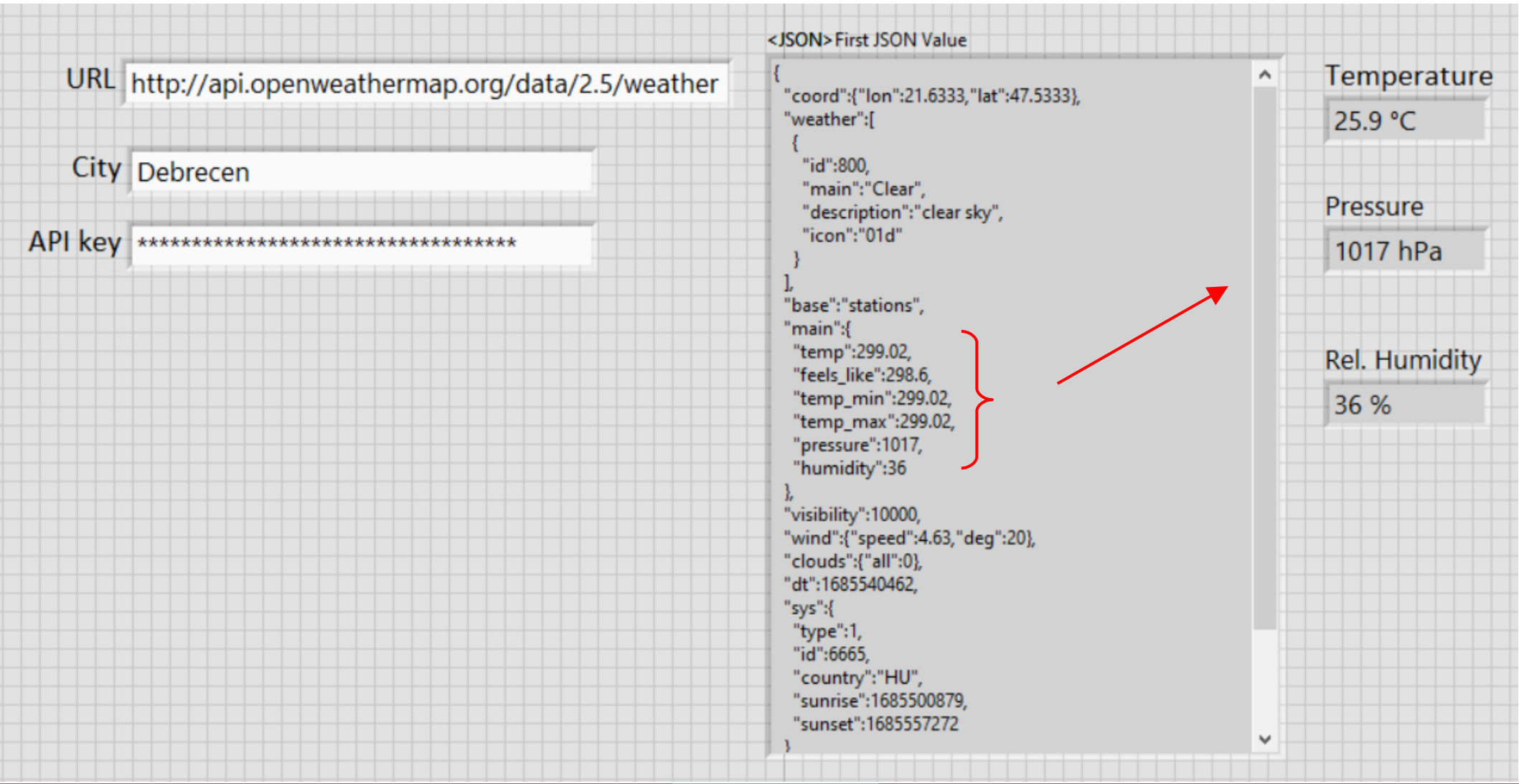
OpenWeatherMap_demo.vi

- A **Data Communications » Protocols » HTTP Client** palettán találjuk a **HTTP kliens** kezeléséhez szükséges függvényeket
- A lekérésből a **main** objektum kimazsolázására az **Addons » JSONtext** palettáról a **Find Object Items by Name** függvényt használjuk, a kapott tömböt azonban még numerikus értékekké kell konvertálni a **From JSON Text** függvénnyel, a hőmérsékletet pedig át kell váltani Kelvin skáláról Celsiusra



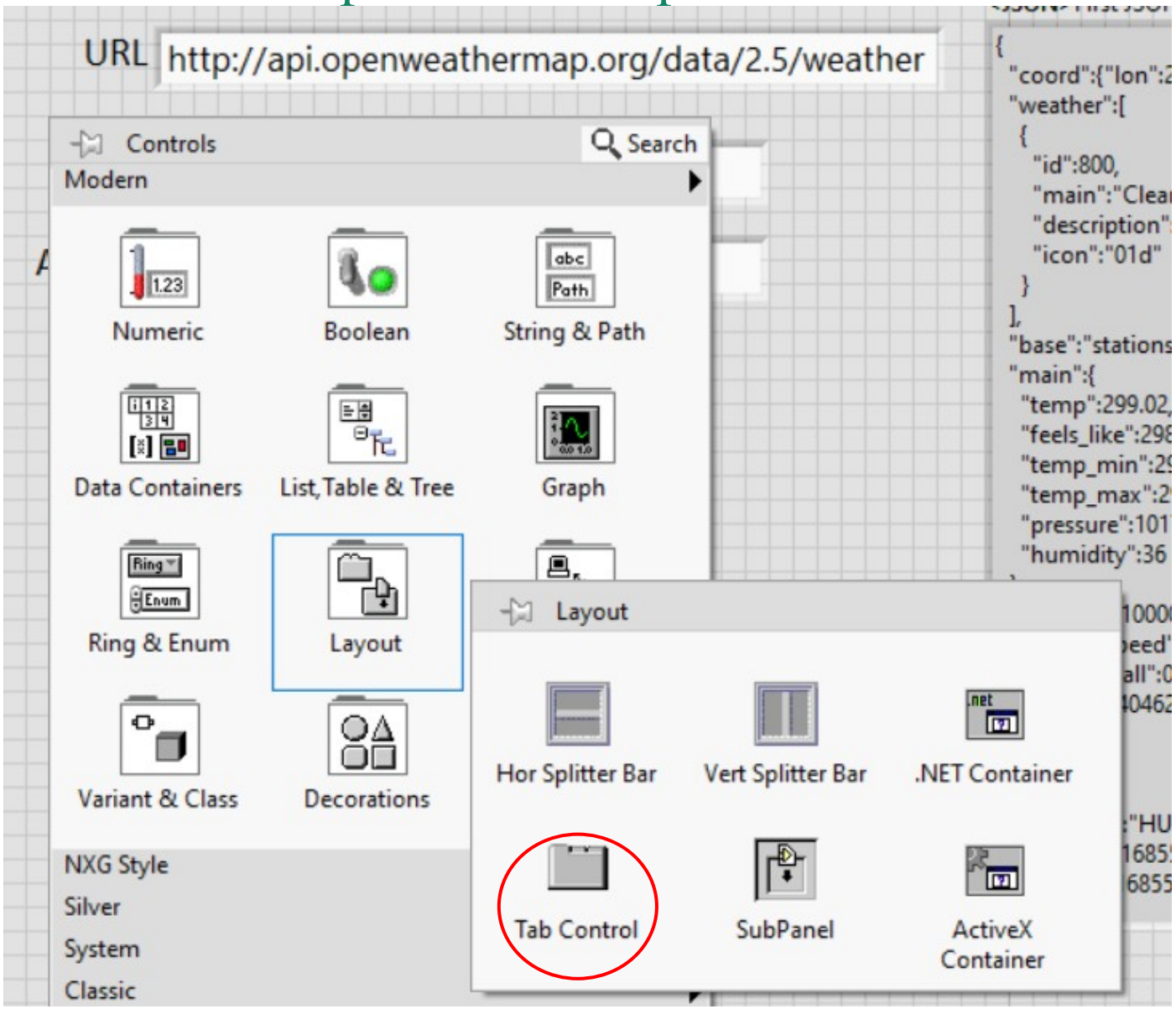
OpenWeatherMap_demo.vi

- Egy lekérés eredménye látható az alábbi ábrán. Természetesen a csillagok helyére be kell írnunk a saját API kulcsunkat!

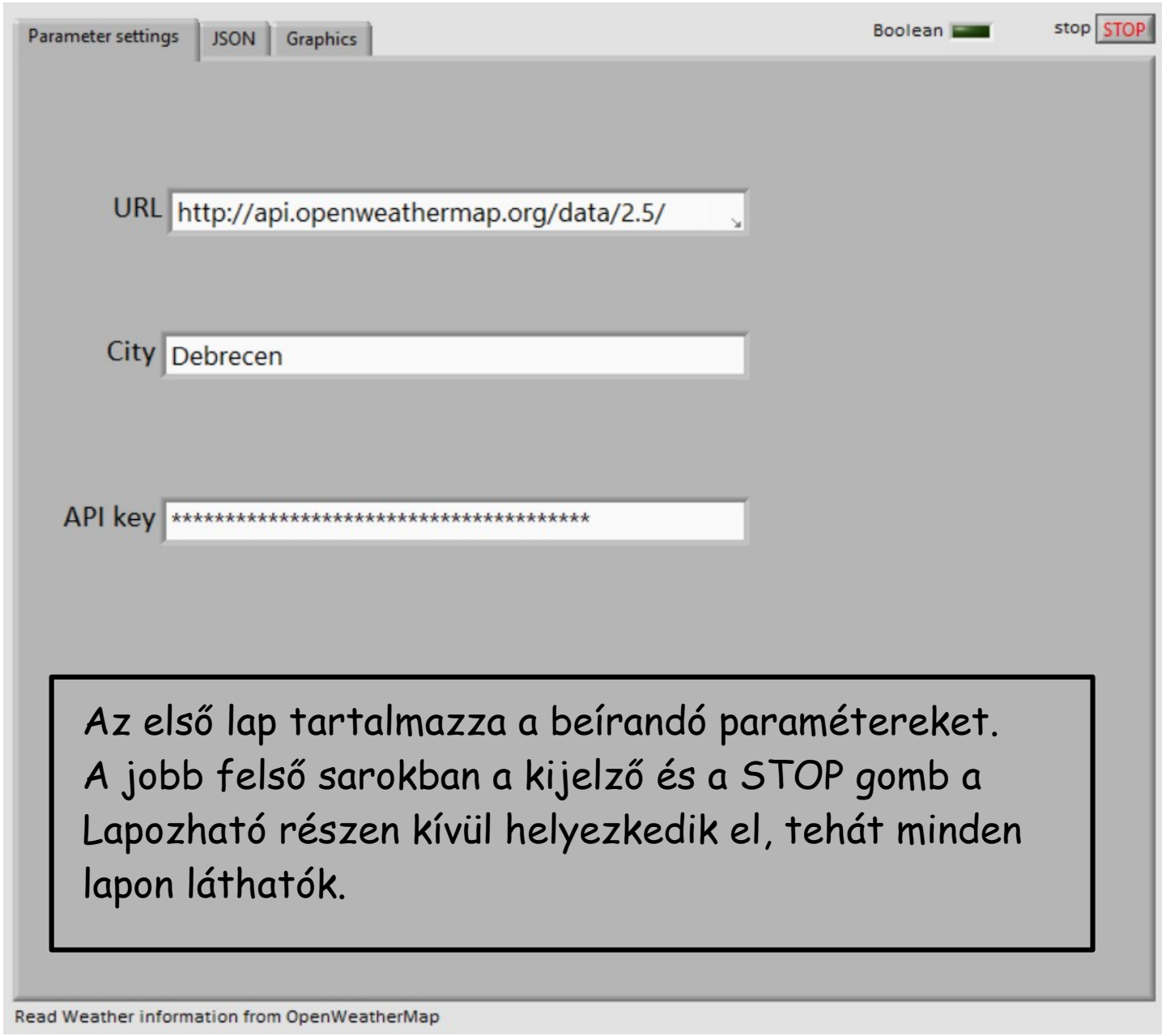


Lapozható (tabbed) előlap

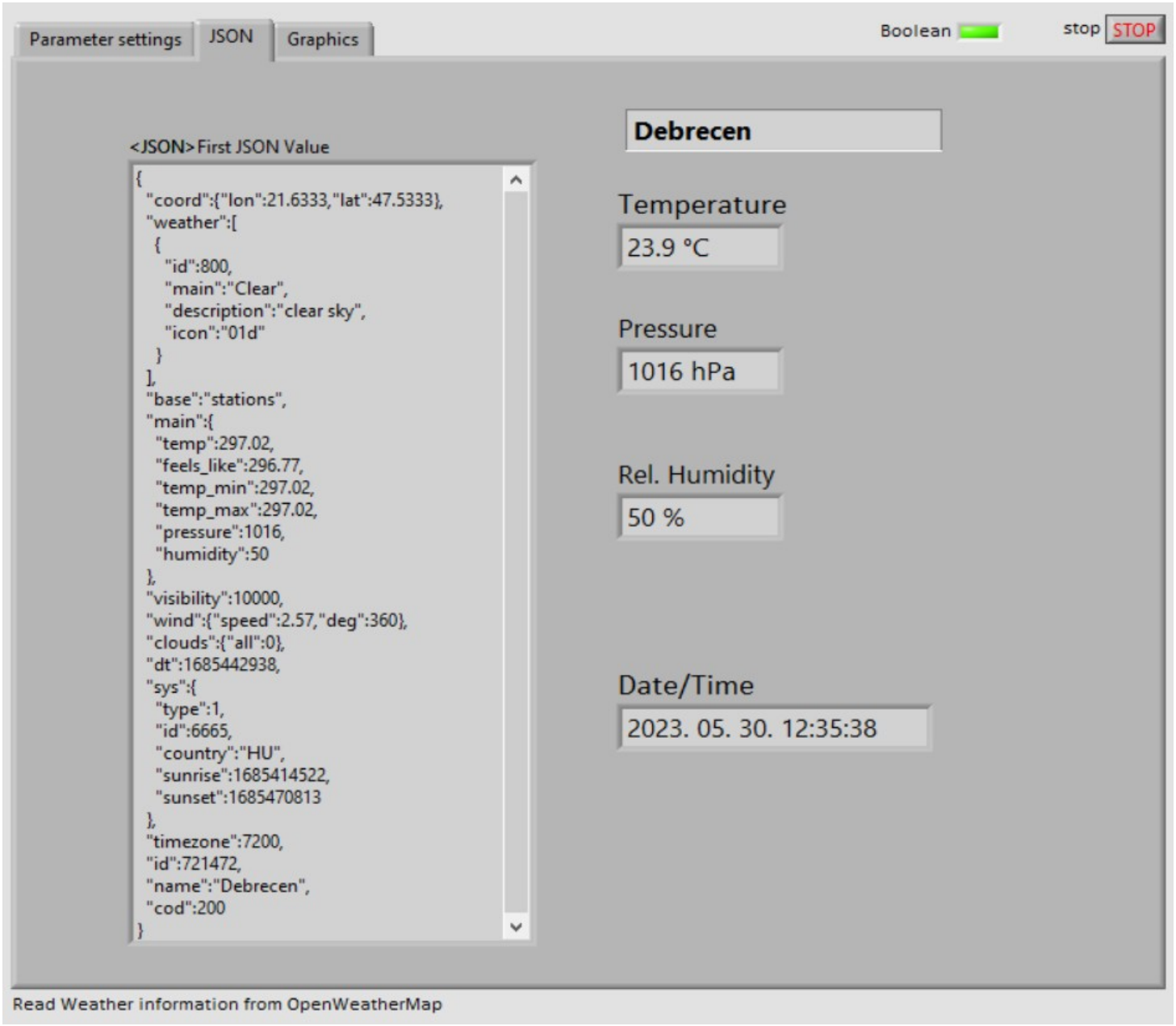
- A **Layout » Tab Control** elem lapozható előlapot kínál
  - Lapokkal bővíthető
  - Ráhúzzuk az elemeket
  - Átméretezhető lapok
  - A **Tab Control** ikon a blokkdiagramban is megjelenik, de nem kell foglalkozni vele



GET_OpenWeatherMap_data.vi



GET_OpenWeatherMap_data.vi

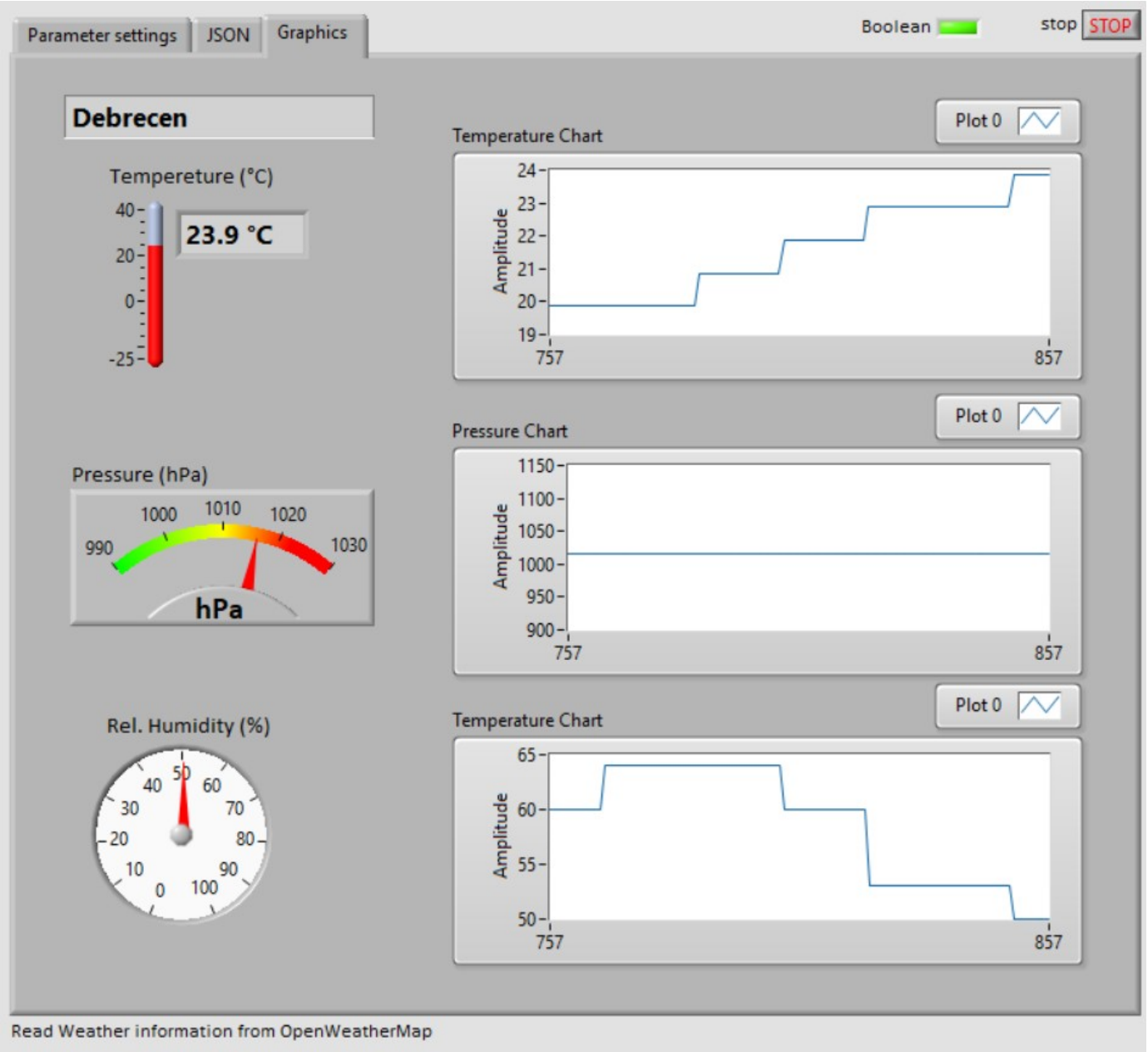


A második lap mutatja a válaszul kapott JSON szöveget, a város nevét, valamint a kimazsolázott adatokat.

A Date/Time időbélyeg az adatközlés időpontja (amikor a mérés történt)

A LED minden lekérésnél állapotot vált.

GET_OpenWeatherMap_data.vi



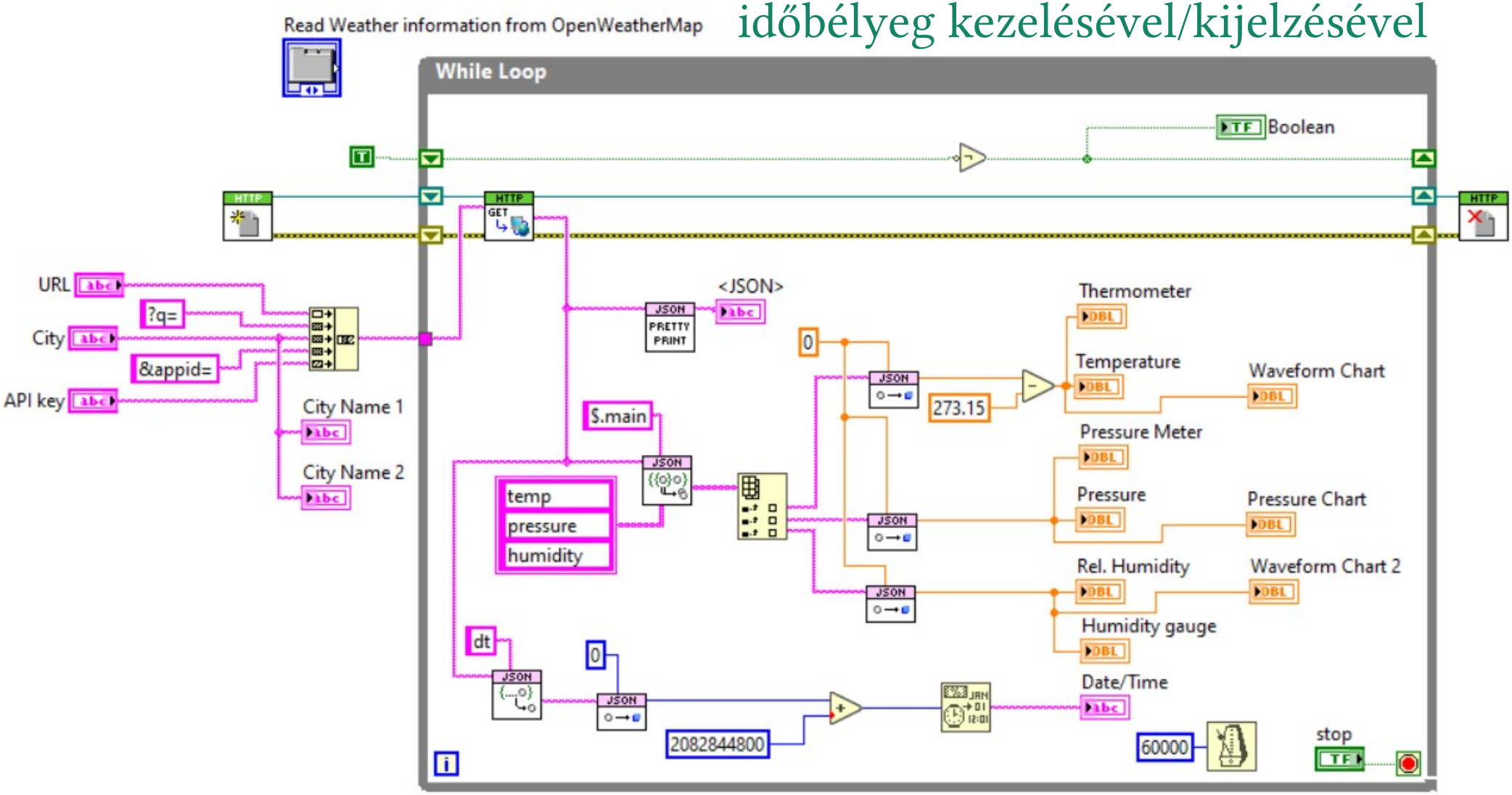
A harmadik lapon grafikus kijezőkkel jelenítjük meg az előző oldalon látott adatokat

A mérési adatoktról egy-egy grafikon az időbeli függést is mutatja (a legutóbbi száz lekérés adatait látjuk)

A LED minden lekérésnél állapotot vált.

GET_OpenWeatherMap_data.vi

- A blokkdiagram hasonló az **OpenWeatherMap_demo.vi** programéhoz, csak kiegészítettük néhány további kijelzéssel és az időbélyeg kezelésével/kijelzésével



Időszámításunk kezdete...

- Az előző lap alján eldugva egy időbélyeg kiolvasás és olvasható formába történő konvertálás is láthatunk
- A **Find Item** és a **From JSON Text** használata már ismerős számunkra. A bonyodalom ott kezdődik, hogy az **OpenWeatherMap** és a **LabVIEW** időszámításának kezdete (***epoch time***) nem esik egybe:
  - LabVIEW: 1904. január 1.
  - UNIX Epoch: 1970. január 1.
- A különbség *2 082 844 800* másodperc, ennyivel kell korrigálni
- A **Format Date/Time String** VI a **Timing** palettán található, ez konvertálja olvashatóvá az időbélyeget. Mi az alapértelmezett formátumot használtuk, de lehetőség van a formátum megadására