Skriptovatelný LED panel

Uživatelská dokumentace pro varianty

Ethernet (HTTP GET API);

Ethernet (HTTP klient);

Ethernet (MODBUS/TCP Slave)

EGMedical, s.r.o. Filipínského 1534/55, 615 00, Brno CZ

www.egmenergo.cz

2023

Obsah

Verze příručky ze dne 17. 7. 2019

Grafická úprava ze dne 5. 10. 2023, 16 stran.

Zpracoval Bc. Antonín Hruškovský

Aleš Jílek; Martin Sekáč

1. Předmluva

Vážený zákazníku,

děkujeme Vám za to, že jste si zvolil produkty firmy EGMedical, s.r.o.

Produkty naší firmy jsou výrobky vycházející z mnoha let zkušeností s vývojem a výrobou elektronických zařízení ze širokého spektra oborů elektronických systémů, hlasových aplikací, průmyslového řízení, robotiky, automatizace, telekomunikací i sdělovací techniky. Tento návod Vám pomůže při instalaci, správném používání a údržbě výrobku. Jsme si jisti, že Vám bude produkt od EGMedical bezproblémově sloužit.

Předtím než naše produkty opustí brány vývojových laboratoří prochází plným testem funkčnosti a kvality. Budete-li mít i přesto nějaký problém s naším zbožím, rádi Vám pomůžeme jej vyřešit.

EGMedical poskytuje záruku na všechny své výrobky, ta se však vztahuje pouze na výrobky používané v souladu s návodem a bezpečnostními pokyny. Zásah a opravy do výrobků smí provádět pouze pověřený technik EGMedical, pokud není vysloveně uvedeno jinak. Upozorňujeme, že změny v nastavení výrobku nebo zásahy do hardwaru systému mohou podstatně ovlivnit jeho fungování a životnost.

Návod k použití byl sepsán na základě našich poznatků a zkušeností. Mějte prosím na zřeteli, že naše výrobky jsou neustále vyvíjeny a zlepšovány, proto se můžete v budoucnu setkat s modifikacemi, které v tomto manuálu nejsou popsány.

Za tým EGMedical Ing. Ivo Strašil, vedoucí vývoje.

2. Webové rozhraní

Pro ovládání a nastavování displeje slouží jednoduché webové rozhraní. Ve webovém rozhraní lze nastavit požadovanou informaci k zobrazení, systémové údaje a nastavení sítě. Dále v textu budou podrobněji rozebrána jednotlivá nastavení.

Pro přístup do webového rozhraní je zapotřebí znát IP adresu zařízení a heslo. Tyto údaje jsou ve výchozím stavu nastaveny následovně:

- IP adresa: ***192.168.1.206***
- Heslo: ***test***

2.1. Zobrazení informace na displeji

Informace zobrazovaná na displeji je zadávána pomocí LUA skriptu. LUA skripty se píší přímo ve webovém rozhraní, kde je k tomuto účelu vytvořen editor, který zadané skripty zpracovává. Editor navíc umí zvýrazňovat syntaxi. Skripty je možné vytvářet, editovat a mazat.

Před tím, než se dostaneme na stránku s formulářem pro psaní LUA skriptů, klikneme na tlačítko *Display* a zadáme název nového LUA skriptu. Zvolíme, zda půjde o *Pokročilý* skript, který umožňuje vytváření složitějších skriptů a definování vlastních funkcí nebo jednoduchý, ve kterém lze používat jen definovanou sadu uživatelských funkcí.

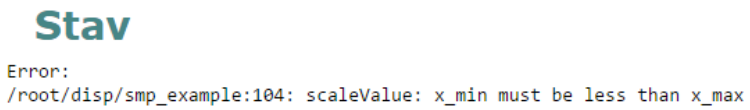
Vytvořením jednoduchého skriptu (nezatržená volba *Pokročilý*) se v seznamu skriptů objeví nová položka, budeme automaticky přesměrováni na stránku s LUA editorem a tělo skriptu bude předvyplněno ukázkovým kódem.

Uživatel si zobrazovanou informaci upravuje editací skriptu s využitím dostupných funkcí, které má k dispozici spolu s popisem na webové stránce. Výpis funkcí pro jednoduchý skript s popisem je navíc uveden i v dokumentaci v kapitole 3.1.

Při vytváření pokročilého skriptu (zatržení volby *Pokročilý*) se v seznamu skriptů objeví nová položka, budeme automaticky přesměrováni na stránku s LUA editorem a tělo skriptu bude předvyplněno ukázkovým kódem. Uživatel má opět k dispozici funkce, které jsou popsány na webové stránce a v dokumentaci v kapitole 3.2.

Po vytvoření skriptu klikneme na tlačítko *Uložit* umístěné v pravé horní části obrazovky pod nadpisem *Script editor*, vedle názvu našeho skriptu. Přepneme se zpět na obrazovku s tabulkou všech skriptů, kliknutím na tlačítko *Display* v uživatelském menu, vybereme nově vytvořený skript a klikneme na *Uložit*.

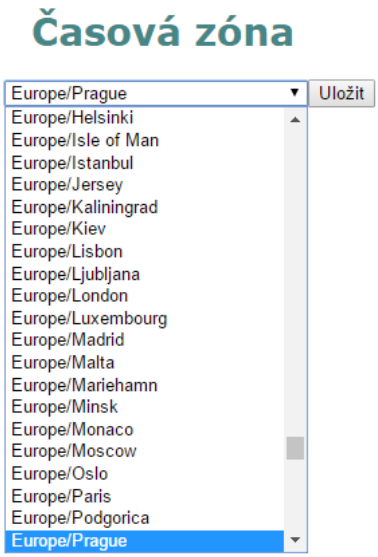
Pokud je kód validní, dojde k okamžitému spuštění a vykreslení na displeji. Pokud je v kódu chyba, jsme o této skutečnosti informování prostřednictvím chybové hlášky umístěné pod tabulkou se skripty. Příklad možné chyby je uveden na následujícím obrázku.



Obr. 2.1: Příklad chybové hlášky

2.2. Systémové nastavení

Do systémového nastavení se dostaneme kliknutím na položku *Systém* v uživatelském menu. První položkou nastavení je *Časová zóna*. Pokud je zařízení připojeno k internetu, je datum a čas nastaven automaticky pomocí protokolu NTP pro vybranou časovou zónu.



Obr. 2.2: Volba časové zóny

Aby se projevila změna časové zóny, je potřeba zařízení restartovat. To provedeme kliknutím na tlačítko *Restartovat zařízení* v sekci *Reset*.

Obr. 2.3: Tlačítko pro restart zařízení

V případě, že chceme čas nastavit ručně, je pro tuto možnost k dispozici formulář pro ruční nastavení. V ručním nastavení času lze nastavit datum ve formátu `DD.MM.YYYY` a čas ve formátu `hh:mm`. Čas nastavený pomocí NTP má však vyšší prioritu a má tedy přednost před manuálním nastavením.

Obr. 2.4: Formulář pro ruční nastavení data a času

Pro přístup do webového rozhraní je potřeba přihlášení heslem. Heslo lze nastavit či změnit v sekci *Přihlášení*. Při zadávání nového hesla se používá dvojí kontrola z důvodu vyvarování se nechtěných překlepů.

Obr. 2.5: Formulář pro změnu hesla

2.3. Nastavení sítě

Do nastavení sítě se dostaneme kliknutím na tlačítko *Síť*. V horní části obrazovky je zobrazena informace o síťovém rozhraní, přidělené IP adrese, masce, atd. Jedná se o výpis linuxového příkazu `ifconfig eth0`.

Pod tímto výpisem se nachází formulář pro nastavení síťového rozhraní. Pomocí formuláře lze nastavit vlastní IP adresu, síťovou masku, výchozí bránu a adresu DNS serveru. Výchozí nastavení je zobrazeno na obrázku 2.6.

Obr. 2.6: Výchozí nastavení síťového rozhraní

Změna IP adresy, masky a výchozí brány se projeví okamžitě. Změna DNS serveru však vyžaduje restart zařízení, aby se projevila.

3. Seznam funkcí

3.1. Jednoduchý skript

V tabulce je uveden výčet všech podporovaných funkcí v jednoduchém skriptu spolu s popisem jednotlivých funkcí.

DispRed()	Nastavení červené barvy.
Text vypsaný za touto funkcí bude mít červenou barvu	
DispGreen()	Nastavení zelené barvy.
Text vypsaný za touto funkcí bude mít zelenou barvu	
DispYellow()	Nastavení žluté barvy.
Text vypsaný za touto funkcí bude mít žlutou barvu	
DispNextLine()	Nastavení pozice na začátek spodního řádku.
Text vypsaný za touto funkcí vypsán od začátku spodního řádku	
DispLinePosition(l)	Nastavení řádku.
Funkce má jeden vstupní parametr: l: číslo řádku, 0 - horní řádek, 1 - spodní řádek	
DispAdd(t)	Výpis uživatelského textu
Funkce má jeden vstupní parametr: t: uživatelský text, který má být vypsán na displej	

DispSetX(x)	Nastavení pozice X na řádku.
Funkce má jeden vstupní parametr: x: pozice na řádku. Hodnota je omezená rozlišením displeje	
DispAddReg(reg)	Výpis obsahu registru na displej.
Funkce má jeden vstupní parametr: reg: číslo registru. Hodnota je omezená <0, 99>. Zadání čísla mimo povolený rozsah způsobí vypsání 0.	
DispAddReg32s(reg, endian)	Výpis obsahu registru jako 32-bitového znaménkového čísla na displej.
Funkce má dva vstupní parametry: reg: číslo registru. Hodnota je omezená <0, 98>. Zadání čísla mimo povolený rozsah způsobí vypsání 0. endian: výběr endianity. True - BigEndian, False - LittleEndian	
DispAddReg32u(reg, endian)	Výpis obsahu registru jako 32-bitového bezznaménkového čísla na displej.
Funkce má dva vstupní parametry: reg: číslo registru. Hodnota je omezená <0, 98>. Zadání čísla mimo povolený rozsah způsobí vypsání 0. endian: výběr endianity. True - BigEndian, False - LittleEndian	
DispTime()	Výpis času na displej.
Funkce nemá žádný vstupní parametr. Funkce vypisuje aktuální čas v hodinách a minutách na displej (hh:mm)	
DispDate()	Výpis data na displej.
Funkce nemá žádný vstupní parametr. Funkce vypisuje aktuální datum na displej (DD.MM.YYYY)	
DispAddBit(reg, tt, tf)	Výpis požadované zprávy na displej v závislosti na hodnotě bitového registru.
Funkce má tři vstupní parametry: reg: číslo registru. Hodnota je omezená <0, 99>. Zadání čísla mimo povolený rozsah způsobí vypsání zprávy uložené v parametru tf. tt: zpráva, která bude vypsána, jestliže registr obsahuje hodnotu True tf: zpráva, která bude vypsána, jestliže registr obsahuje hodnotu False	
getBit(i)	Funkce vrací hodnotu konkrétního bitového registru jako Boolean.
Vstupní parametr je číslo registru <0, 99> Zadání adresy registru mimo uvedený rozsah způsobí vrácení False Příklad: bit = getBit(1) -- Vyčtení hodnoty bitu	
getRegister(i[, b])	Funkce vrací hodnotu konkrétního Modbus registru.

První, povinný parametr je číslo registru <0, 99> Zadání adresy registru mimo uvedený rozsah způsobí vrácení 0 Druhý, nepovinný parametr je Boolean: true: vrací číslo jako znaménkové false: vrací číslo jako neznaménkové (výchozí hodnota) Příklad: nez = getRegister(1) -- Vyčtení hodnoty registru 1 zna = getRegister(1, true) -- Vyčtení hodnoty registru 1	
getReg32u(i[, e])	Funkce vrací hodnotu zadaného a následujícího registru jako 32-bitové číslo bez znaménka.
První, povinný parametr je číslo registru <0, 98> Zadání adresy registru mimo uvedený rozsah způsobí vrácení 0 Druhý, nepovinný parametr je Boolean: true: vrací hodnotu v BigEndian formátu false: vrací hodnotu v LittleEndian formátu (výchozí hodnota) Příklad: nezn32le = getReg32u(1) -- Vyčtení spojené hodnoty registrů 1 a 2 nezn32be = getReg32u(1, true) -- Vyčtení spojené hodnoty registrů 1 a 2	
getReg32s(i[, e])	Funkce vrací hodnotu zadaného a následujícího registru jako 32-bitové číslo se znaménkem.
První, povinný parametr je číslo registru <0, 98> Zadání adresy registru mimo uvedený rozsah způsobí vrácení 0 Druhý, nepovinný parametr je Boolean: true: vrací hodnotu v BigEndian formátu false: vrací hodnotu v LittleEndian formátu (výchozí hodnota) Příklad: zn32le = getReg32s(1) -- Vyčtení spojené hodnoty registrů 1 a 2 zn32be = getReg32s(1, true) -- Vyčtení spojené hodnoty registrů 1 a 2	
scaleValue(v, o1, o2, n1, n2)	Funkce vrací přeškálovanou hodnotu ze současných mezí do nových mezí.
Funkce má pět vstupních parametrů: v: číselná hodnota, kterou chceme škálovat o1: číselná hodnota, dolní mez původního rozsahu o2: číselná hodnota, horní mez původního rozsahu n1: číselná hodnota, dolní mez nového rozsahu n2: číselná hodnota, horní mez nového rozsahu Příklad: hodnota = scaleValue(5, 0, 10, 0, 100) -- Převede hodnotu 5 z intervalu <0,10> do intervalu <0,100> (50)	
getDec(v, d)	Funkce vrací hodnotu zadaného desetinného čísla s požadovaným počtem desetinných míst.
Funkce má dva vstupní parametry: v: původní číselná hodnota d: počet desetinných míst, lze i přidávat Příklad: hodnota = getDec(666.456332, 3) -- Vrací hodnotu na 3 desetinná místa	

3.2. Pokročilý skript

Následuje výpis podporovaných funkcí pro pokročilý skript. V pokročilém skriptu **musí** být uvedena funkce `update_text(counter)`, která zajišťuje výpis na displej.

update_text(counter)	Aktualizace textu na displeji

Funkce se volá automaticky s každým přetečením čítače nastaveném funkcí setPeriod(t). Má jeden parametr:
counter: číslo, které se zvyšuje o jedna po každém spuštění.
 	
Příklad:
`function update_text(counter)`
`  local text`
 	
`  text = "\\x056P "`
`  text = text .. getDec(0.456332 + counter, 3)`
 	
`  return text`	

Následující tabulka obsahuje výpis a popis podporovaných funkcí.

setPeriod(t)	Nastavení periody update textu.
Vstupní parametr je celé číslo. Číslo je z rozsahu 1 až 50, což odpovídá 100 ms až 5000 ms. Zadání čísla mimo uvedený rozsah způsobí chybu. Po vypršení timeoutu se volá funkce update_text(counter) Příklad: `setPeriod(10) -- Perioda nastavena na 1 s`	
getRegister(i[, b])	Funkce vrací hodnotu konkrétního Modbus registru.

První, povinný parametr je číslo registru <0, 99>
Zadání adresy registru mimo uvedený rozsah způsobí vrácení 0
Druhý, nepovinný parametr je Boolean:
true: vrací číslo jako znaménkové
false: vrací číslo jako neznaménkové (výchozí hodnota)
Příklad:
`nez = getRegister(1) -- Vyčtení hodnoty registru 1`
`zna = getRegister(1, true) -- Vyčtení hodnoty registru 1`	

getBit(i)	Funkce vrací hodnotu konkrétního bitového registru jako Boolean.
Vstupní parametr je číslo registru <0, 99> Zadání adresy registru mimo uvedený rozsah způsobí vrácení False Příklad: `bit = getBit(1) -- Vyčtení hodnoty bitu`	
getReg32u(i[, e])	Funkce vrací hodnotu zadaného a následujícího registru jako 32-bitové číslo bez znaménka.
První, povinný parametr je číslo registru <0, 98> Zadání adresy registru mimo uvedený rozsah způsobí vrácení 0 Druhý, nepovinný parametr je Boolean:	

true: vrací hodnotu v BigEndian formátu false: vrací hodnotu v LittleEndian formátu (výchozí hodnota) Příklad: nezn32le = getReg32u(1) -- Vyčtení spojené hodnoty registrů 1 a 2 nezn32be = getReg32u(1, true) -- Vyčtení spojené hodnoty registrů 1 a 2	
getReg32s(i[, e])	Funkce vrací hodnotu zadaného a následujícího registru jako 32-bitové číslo se znaménkem.
První, povinný parametr je číslo registru <0, 98> Zadání adresy registru mimo uvedený rozsah způsobí vrácení 0 Druhý, nepovinný parametr je Boolean: true: vrací hodnotu v BigEndian formátu false: vrací hodnotu v LittleEndian formátu (výchozí hodnota) Příklad: zn32le = getReg32s(1) -- Vyčtení spojené hodnoty registrů 1 a 2 zn32be = getReg32s(1, true) -- Vyčtení spojené hodnoty registrů 1 a 2	
getTime()	Funkce vrací tři hodnoty v pořadí hodiny, minuty, sekundy.
Funkce nemá žádný vstupní parametr Příklad: h, m, s = getTime() -- Vyčtení hodin, minut a sekund	
getDate()	Funkce vrací čtyři hodnoty v pořadí den, měsíc, rok, den v týdnu.
Funkce nemá žádný vstupní parametr. Dny v týdnu jsou v pořadí: 0 - Neděle 1 - Pondělí 2 - Úterý 3 - Středa 4 - Čtvrtek 5 - Pátek 6 - Sobota Příklad: d, m, r, w = getDate() -- Vyčte den, měsíc, rok a den v týdnu	
scaleValue(v, o1, o2, n1, n2)	Funkce vrací přeškálovanou hodnotu ze současných mezí do nových mezí.
Funkce má pět vstupních parametrů: v: číselná hodnota, kterou chceme škálovat o1: číselná hodnota, dolní mez původního rozsahu o2: číselná hodnota, horní mez původního rozsahu n1: číselná hodnota, dolní mez nového rozsahu n2: číselná hodnota, horní mez nového rozsahu Příklad: hodnota = scaleValue(5, 0, 10, 0, 100) -- Převede hodnotu 5 z intervalu <0,10> do intervalu <0,100> (50)	
getDec(v, d)	Funkce vrací hodnotu zadaného desetinného čísla s

požadovaným počtem desetinných míst.
Funkce má dva vstupní parametry: v: původní číselná hodnota d: počet desetinných míst, lze i přidávat Příklad: hodnota = getDec(666.456332, 3) -- Vrací hodnotu na 3 desetinná místa

3.3. Vzdálené zadávání textů

Zařízení umožňuje přímé zadání textu bez využití skriptu. Tohoto lze docílit pomocí HTTP GET požadavku. Dle kódování lze využít jednu z těchto variant:

http://192.168.1.206/script/text1250?test%20text - kódování textu Win1250

- http://192.168.1.206/script/text1250 – vrací naposledy manuálně zaslaný text

http://192.168.1.206/script/textUTF8?test%20text - kódování textu UTF8

- http://192.168.1.206/script/textUTF8 – vrací naposledy manuálně zaslaný text

Obě tyto varianty umožňují změnu jasu. Změny jasu lze docílit přidáním znaku ASCII 0x7F za kterým následuje hodnota jasu 0-9 (0 = minimální jas; 9 = maximální jas). Tato kombinace znaků lze použít pouze na začátku textu. Příklady použití:

http://192.168.1.206/script/text1250?%7F4test%20text - Win1250 ; úroveň jasu 4

http://192.168.1.206/script/textUTF8?%7F9test%20text - UTF8 ; úroveň jasu 9 - max

3.4. Escape sekvence

Tyto sekvence se používají pro změnu vlastností textů. Následující sekvence jsou specifické pro tento displej. Lze je použít pro Pokročilý skript a pro Vzdálené zadávání textů.

Pozor! Při využití ve skriptu je nutné zadat dvojitě znak **"\"** např. **"\\g"**

\xNNN	Nastavení X souřadnice textu.
Za znakem x (namísto NNN) musí vždy následovat 3 čísla. Hodnota je omezená rozlišením displeje. Lze použít velké X které zdvojí zadané souřadnice v případě tlustého textu. Příklad: \x010TEXT -- Nastavení kurzoru osy X na souřadnici 10 \x-010TEXT -- Nastavení kurzoru osy X na souřadnici -10	
\r	Nastaví text na červenou barvu.
Následující zobrazení bude červenou barvou.	
\g	Nastaví text na zelenou barvu.
Následující zobrazení bude zelenou barvou.	
\a	Nastaví text na žlutou barvu.
Následující zobrazení bude žlutou barvou.	
\0 \1 ...	Pozicování textu: rozsah 0 až 7, dle panelu.

Následující zobrazení bude na specifikovaném řádku, ale souřadnice X zůstane zachována.	
\i	Velikost textu: normální text.
Následující zobrazení bude přes jeden řádek, ale souřadnice X zůstane zachována.	
\b	Velikost textu: vysoký text.
Následující zobrazení bude přes dva řádky, ale souřadnice X zůstane zachována.	
\h	Velikost textu: velmi vysoký text.
Následující zobrazení bude přes tři řádky, ale souřadnice X zůstane zachována.	
\m	Velikost textu: maximální text.
Následující zobrazení bude přes čtyři řádky, ale souřadnice X zůstane zachována.	
\d	Styl textu: tučný text.
Následující zobrazení bude tučným písmem.	
\n	Styl textu: normální text.
Následující zobrazení bude normálním písmem.	

3.5. Nouzový režim

Zařízení lze přepnout do nouzového režimu pomocí tlačítka uvnitř. Tlačítko na modulu Linkit označené „W_1F_1“ je třeba podržet alespoň 2 vteřiny. Poté dojde k aktivaci nouzového režimu.

Nouzový režim dočasně nastaví síť na „IP 192.168.1.206 maska 255.255.255.0“ a ve webovém rozhraní nebude vyžadováno přihlášení.

Tento režim je aktivní až do restartu zařízení. Restart lze provést ve webovém rozhraní.

4. Příklad LUA skriptů

Display je vybaveno ukázkovými skripty pro pro pohyb textu. Skripty jsou napsané tak aby šlo upravit zobrazení.

4.1. Pohyblivý text přes celý panel

- pro změnu textu je třeba upravit hodnotu „text“ a „sc0“

```
-- příklad pokročilého skriptu
function position(num)
        local ret
        if num < 0 then
                num = -num
                ret = "\\x-"
        else
                ret = "\\x"
```

```
        end
        if num < 10 then
                return ret .. "00" ..num
        elseif num < 100 then
                return ret .. "0" ..num
        elseif num > 999 then
                num = 999
        end
        return ret .. num
end
-- tato funkce je volána pravidelně a musí vracet text ke zobrazení
function update_text(counter)
        scp = scp - 2 -- tady lze přidat rychlost, 2 = počet bodů
        if scp < sc0 then
                scp = sc1
        end
        return position(scp) .. text
end
-- toto je konec kam až text dojede (liší se dle délky textu a efektu)
sc0 = -256
-- toto je začátek odkud text přijede (rozlišení panelu)
sc1 = 96
-- toto je text který pojede
text = "\\b\\aScrolující text pokus bla bla The Sting atd atd .."
-- počáteční pozice
scp = sc1
-- rychlost animování
setPeriod(1)
```

4.2. Nehybný a pohyblivý text současně

- pro změnu pohyblivého textu je třeba upravit hodnotu „text“ a „sc0“
- pro měnu statického textu je třeba upravit hodnotu „txt0“

```
-- příklad pokročilého skriptu
function position(num)
        local ret
        if num < 0 then
                num = -num
                ret = "\\x-"
        else
                ret = "\\x"
        end
        if num < 10 then
                return ret .. "00" ..num
        elseif num < 100 then
                return ret .. "0" ..num
        elseif num > 999 then
                num = 999
        end
        return ret .. num
end
-- tato funkce je volána pravidelně a musí vracet text ke zobrazení
function update_text(counter)
        scp = scp - 2 -- tady lze přidat rychlost, 2 = počet bodů
        if scp < sc0 then
                scp = sc1
        end
        return txt0 .. "\\l" .. position(scp) .. text
end
-- toto je konec kam až text dojede (liší se dle délky textu a efektu)
sc0 = -128
```

```
-- toto je začátek odkud text přijede (rozlišení panelu)
sc1 = 96
-- toto je text který nepojede
txt0 = "\\r\\x017Statický text"
-- toto je text který pojede
text = "\\gScrollující text na druhém řádku ..."
-- počáteční pozice
scp = sc1
-- rychlost animování
setPeriod(1)
```

4.3. Složitější animace, dva texty přecházející v jeden.

- hodnota „scl“ musí být přesně délka textu v pixelech
- hodnota „text“ je text ke zobrazení
- hodnota „wait0“ je doba zobrazení dvou textů po střetu
- hodnota „wait1“ je doba zobrazení jednoho velkého textu
  - po uplynutí této doby dojde k opakování animace

```
-- příklad pokročilého skriptu
function position(num)
        local ret
        if num < 0 then
                num = -num
                ret = "\\x-"
        else
                ret = "\\x"
        end
        if num < 10 then
                return ret .. "00" ..num
        elseif num < 100 then
                return ret .. "0" ..num
        elseif num > 999 then
                num = 999
        end
        return ret .. num
end

function update_text_scroll(counter)
        scp = scp + 4 -- tady lze přidat rychlost, 4 = počet bodů
        if scp >= sct then
                scp = sct
                tw = tw + 1
                if tw >= wait0 then
                        tw = 0
                        update_text = update_text_big
                end
        end
        return "\\g" .. position(scp-scl) .. text .. "\\l\\r" .. position(96-scp) .. text
end

function update_text_big(counter)
        tw = tw + 1
        if tw >= wait1 then
                tw = 0
                scp = -16 -- tady lze změnit posun = doba bez zobrazení
                update_text = update_text_scroll
        end
        return "\\a\\b" .. position(96-scp) .. text
end
```

```
-- délka textu
scl = 26
-- konec animace (střed dle textu)
sct = 48 + (scl/2)
-- text ke zobrazení
text = "pokus"
-- čas čekání po střetu
wait0 = 2
-- čas čekání po zvětšení
wait1 = 30
-- proměnné procesu
scp = -10
tw = 0
-- rychlost animace
setPeriod(1)
-- tato funkce je volána pravidelně a musí vracet text ke zobrazení
update_text = update_text_scroll
```

4.4. Modbus – zobrazení procenta výroby

- Příklad zobrazuje na prvním řádku počet požadovaných a počet vyrobených kusů a na druhém řádku se střída stav výroby v procentech a aktuální datum
- modbus registr č. 0 - počet vyrobených kusů
- modbus registr č. 1 - počet požadovaných kusů
- příklad obsahuje možnost přepínání obrazu

4.4.1. Jednoduchý skript

```
DispAdd("Pož. ")
DispAddReg(1)
DispAdd(" ks")
DispSetX(64)
DispAdd("Vyr. ")
DispAddReg(0)
DispAdd(" ks")

DispNextLine()
-- každých pět vteřin zobrazovat různý text
if (counter % 10) < 5 then
    DispGreen()
    DispDate()
else
    if getRegister(1) == 0 then
            -- požadovaný počet kusů není nastaven, nelze přepočítat na procenta
            DispAdd("= N/A")
    else
            DispAdd("= ")
            -- přepočet na procenta, s omezením na jedno desetinné místo
            DispAdd(getDec(scaleValue(getRegister(0), 0, getRegister(1), 0, 100), 1))
            DispAdd("%")
    end
    DispGreen()
end

DispSetX(96)
DispTime()
```

4.4.2. Pokročilý skript

```
function timeToText()
    local h, m = getTime()
    if m < 10 then
            m = "0" .. m
    end
    if h < 10 then
            h = "0" .. h
    end
    return h .. ":" .. m
end

function dateToText()
    local d, m, y = getDate()
    return d .. "." .. m .. "." .. y
end

function update_text(counter)
    -- modbus registr č. 1 - počet požadovaných kusů
    line0 = "Pož. " .. getRegister(1) .. " ks"

    -- modbus registr č. 0 - počet vyrobených kusů
    line1 = "\\x064Vyr. " .. getRegister(0) .. " ks"

    -- každých pět vteřin zobrazovat různý text
    if (counter % 10) < 5 then
            extra = "\\g" .. dateToText()
    else
            if getRegister(1) == 0 then
              -- požadovaný počet kusů není nastaven, nelze přepočítat na procenta
                    extra = "= N/A"
            else
                    -- přepočet na procenta, s omezením na jedno desetinné místo
                    perc = getDec(scaleValue(getRegister(0), 0, getRegister(1), 0, 100), 1)
                    extra = "= " .. perc .. "%"
            end
            extra = extra .. "\\g"
    end

    time = "\\x096" .. timeToText()

    return line0 .. line1 .. "\\x000\\1" .. extra .. time
end

-- rychlost animování
setPeriod(10)
```

Skripty je možné ve webovém rozhraní kopírovat jako text. Pří úpravě zdrojového skriptu označit vše CTRL+A, zkopírovat do schránky CTRL+C a poté vložit do jiného, např. nově vytvořeného (pozor na možnost „pokročilý“) pomocí CTRL+A a poté CTRL+V.

5. Kontaktní informace

V případě jakýchkoli problémů, připomínek nebo pokud máte nějaké pochvaly, rádi vás vyzýváme, abyste se na nás obrátili prostřednictvím níže uvedené kontaktní adresy. Vaše zpětná vazba a komunikace jsou pro nás důležité a pomáhají nám neustále zlepšovat naše služby. Děkujeme vám za vaši podporu.

EGMedical, s.r.o.

Filipínského 1534/55
615 00 Brno
Česká republika

tel.: +420 537 014 211
email: vyvojari@egmenergo.cz
web: www.egmenergo.cz