Návrh aplikace pro nástěnné ovladače

Abstrakt

Aplikační poznámka se věnuje návrhu uživatelské aplikace pro malé nástěnné ovladače z produkce firmy AMiT. Dále je v ní popsán postup programování komunikace s nástěnnými ovladači ze strany řídicího systému firmy AMiT a postup tvorby obrazovek s vazbami na objekty sítě Poseidon.

Autor: Zbyněk Říha
Dokument: ap0047_cz_03.pdf

Příloha

Obsah souboru: ap0047_cz_03.zip

op70c_p1_cz_01.dsox	Příklad návrhu aplikace pro **AMR-OP70C**.
rs_p1_cz_03.dso	Příklad obsluhy **AMR-OP70C** v řídicím systému (ARION).
rs_p2_cz_01.dso	Příklad obsluhy **AMR-OP70C** v řídicím systému (MODBUS RTU).
op60_p1_cz_01.dsox	Příklad obsluhy tlačítek v **AMR-OP60**.
op70rhp_p1_cz_01.dsox	Příklad práce s rozhraním Poseidon na **AMR-OP70RHP**.

Copyright (c) 2019, AMiT®, spol. s r.o.
amitomation.cz

Obsah

Historie revizí

Verze	Datum	Autor změny	Změny
001	04. 03. 2011	Říha Z.	Nový dokument.
002	20. 04. 2012	Říha Z.	Do kapitoly 3.1.9 doplněna informace o kompatibilitě programu SAM-PROG. Do kapitoly 4.7 doplněn postup zavedení aplikace. Upravena související dokumentace. Aplikace vytvořeny v DetStudiu verze 1.7.2. Úprava obrázků a textů dle chování DetStudia verze 1.7.2.
003	25. 03. 2019	Říha Z.	Změna jména AP, změna struktury AP, přepracování dle aktuálního chování prostředí DetStudio 2.0, ovladač NOA70 nahrazen ovladačem AMR-OP70C.

Související dokumentace

1. Nápověda k části EsiDet vývojového prostředí DetStudio
   soubor: Esidet_cs.chm
2. Nápověda k části PseDet vývojového prostředí DetStudio
   soubor: Psedet_cs.chm
3. Nápověda k obrazovkám vývojového prostředí DetStudio
   soubor: Tridet_cs.chm
4. AMR-OP60 – Návod na obsluhu
   soubor: amr-op60_g_cz_xxx.pdf
5. AMR-OP70C – Návod na obsluhu
   soubor: amr-op70c_g_cz_xxx.pdf
6. AMR-OP70RHP – Návod na obsluhu
   soubor: amr-op70rhp_g_cz_xxx.pdf
7. Aplikační poznámka AP0008 – Komunikace v síti MODBUS RTU (PseDet)
   soubor: ap0008_cz_xx.pdf
8. Aplikační poznámka AP0023 – Skriptování v DetStudiu
   soubor: ap0023_cz_xx.pdf
9. Aplikační poznámka AP0025 – Komunikace v síti ARION – definice tabulkou
   soubor: ap0025_cz_xx.pdf
10. Aplikační poznámka AP0041 – Návrh grafických prvků pro ovladače řady NOA7x
    soubor: ap0041_cz_xx.pdf
11. Aplikační poznámka AP0051 – Komunikace v bezdrátové síti Poseidon
    soubor: ap0051_cz_xx.pdf
12. Aplikační poznámka AP0054 – Komunikace AMREG s řídicími systémy AMiT (ARION)
    soubor: ap0054_cz_xx.pdf
13. Aplikační poznámka AP0055 – Komunikace AMREG s řídicími systémy AMiT (MODBUS RTU)
    soubor: ap0055_cz_xx.pdf

1 Definice použitých pojmů

Registr
32bit hodnota (4 Byty). Registry jsou využívány pro výměnu dat mezi nástěnnými ovladači s dotykovým displejem a nadřazeným systémem. V registrech jsou „mapovány“ proměnné. Počet registrů se tedy vždy odvíjí od počtu přenášených proměnných. Čísla registrů v síti ARION by měla být po sobě jdoucí (např. 0, 1, 2, …).

Mapování proměnné
Určení pozice proměnné v rámci registru pro síť ARION/MODBUS RTU. Mapování může být rozdílné pro proměnné typu DInt nebo Real.

WYSIWYG
Je akronym anglické věty „What you see is what you get“, česky „co vidíš, to dostaneš“. Tato zkratka označuje způsob editace dokumentů v počítači, při kterém je verze zobrazená na obrazovce vzhledově totožná s výslednou verzí dokumentu.

NOS
Je zkratka pro Network Operating System. Jedná se o operační systém z produkce firmy AMiT pro vybrané výrobky.

2 Nástěnné ovladače

Nástěnné ovladače jsou dodávány ve formě:

- slepé ovladače bez ovládacích prvků (slouží pouze pro měření vybraných veličin),
- ovladače s tlačítky či s mechanickým točítkem,
- ovladače s dotykovým displejem.

Komunikační možnosti a možnosti měření různých veličin jsou závislé na typu ovladače. S nadřazeným systémem mohou ovladače komunikovat prostřednictvím rozhraní RS485, na kterém lze provozovat různé komunikační protokoly. Tato aplikační poznámka se zabývá komunikačními protokoly ARION a MODBUS RTU.

Ovladače jsou standardně dodávány s firmware, pomocí kterého lze nastavovat např. režim regulace místnosti, korekci požadované teploty, rychlost ventilátoru FanCoil jednotek, případně lze dát povel pro sepnutí libovolného zařízení.

Při použití ovladačů s displejem je velkou výhodou **možnost tvorby vlastního grafického návrhu a ovládacích algoritmů** ve vývojovém prostředí DetStudio. Lze v nich naprogramovat další funkce, které nesouvisí pouze s teplotou. Programátor tak může naprogramovat např. rozsvícení světel v místnosti, ovládání rolet nebo žaluzií, nastavení časových plánů apod. Výhodou je, že k ovládání všech funkcí stačí právě jen jeden ovladač a v případě, kdy je v místnosti použit např. **AMR-OP70C**, nemusí být vedle něj na stěně další ovladač/vypínač pro ovládání světel atd. Vše je dáno programovým vybavením, které do něj programátor naprogramuje.

3 Příklad návrhu aplikace pro AMR-OP70C

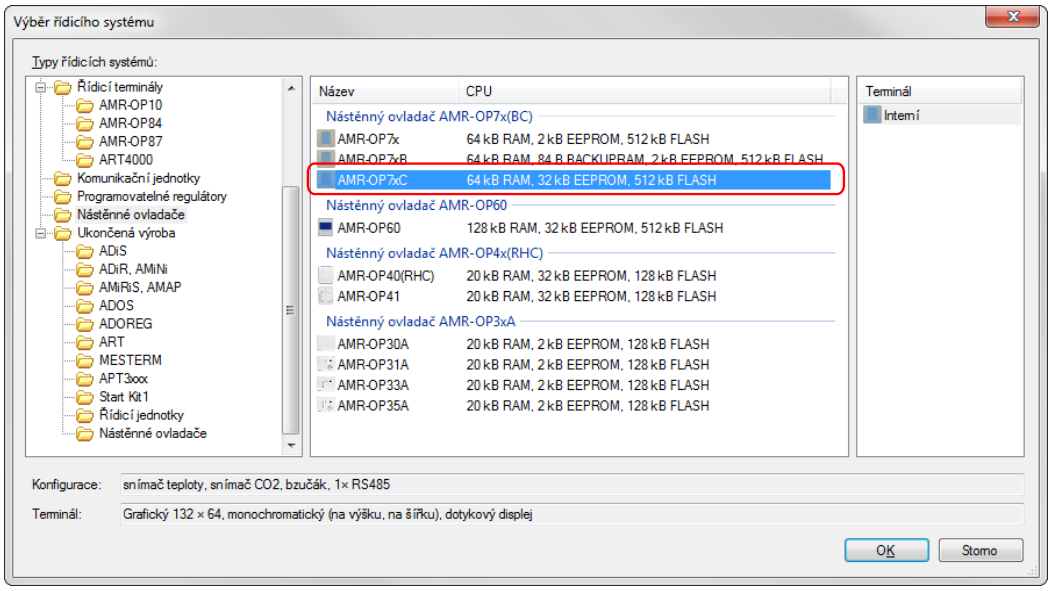
V příkladu bude řešeno:

- zobrazení teploty měřené nástěnným ovladačem,
- zobrazení koncentrace CO_2 měřené nástěnným ovladačem,
- nastavení požadovaného režimu místnosti,
- nastavení požadovaného režimu ventilátoru,
- signalizace překročení nastavené úrovně CO_2,
- poskytnutí měřených a nastavených hodnot do sítě ARION / MODBUS RTU.

Příklad je k dispozici v příloze této aplikační poznámky. Jedná se o soubor s názvem op70c_p1_cz_xx.dsox.

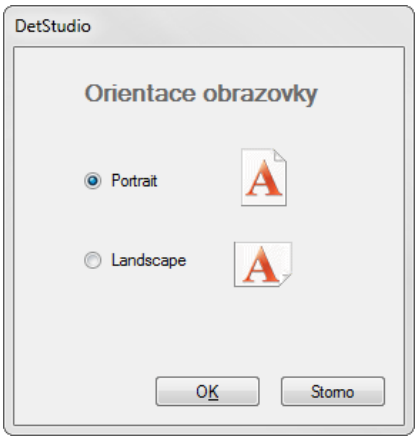
3.1 Založení projektu

Při zakládání projektu vybereme nástěnný ovladač „AMR-OP7xC“.



Obr. 1 – Volba typu nástěnného ovladače

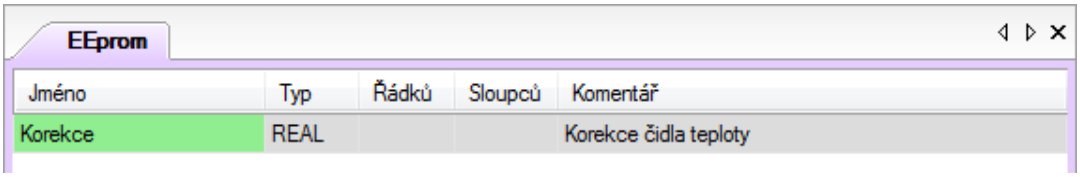
Po výběru nástěnného ovladače zvolíme orientaci „Portrait“ (na výšku).



Obr. 2 – Výběr orientace obrazovky **AMR-OP70C**

3.2 Definice vnitřních proměnných

Proměnné, které nemají po restartu ovladače ztratit svou hodnotu, je nutné umístit do paměti EEPROM. K tomu lze využít objekt **`EEPROM`**. Po dvojkliku na objekt **`EEPROM`** (v okně „Projekt“ – sekce „Objekty“) dojde k otevření záložky „EEPROM“, do které lze pomocí klávesy ***Insert*** vkládat proměnné. V ukázkové aplikaci nadefinujeme v paměti EEPROM pouze proměnnou **`Korekce`**.



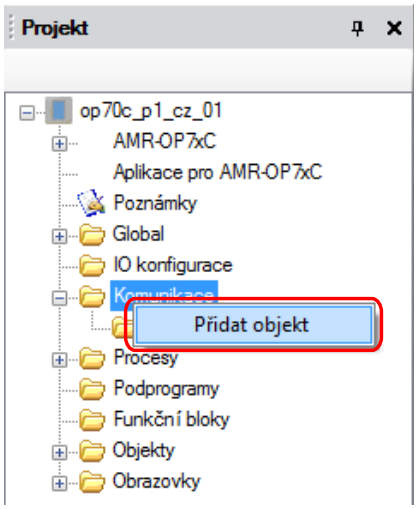
Obr. 3 – Proměnná v paměti EEPROM

3.3 Definice protokolu a proměnných určených pro komunikaci

V ukázkové aplikaci bude **AMR-OP70C** ve funkci zařízení typu slave v síti ARION nebo v síti MODBUS RTU.

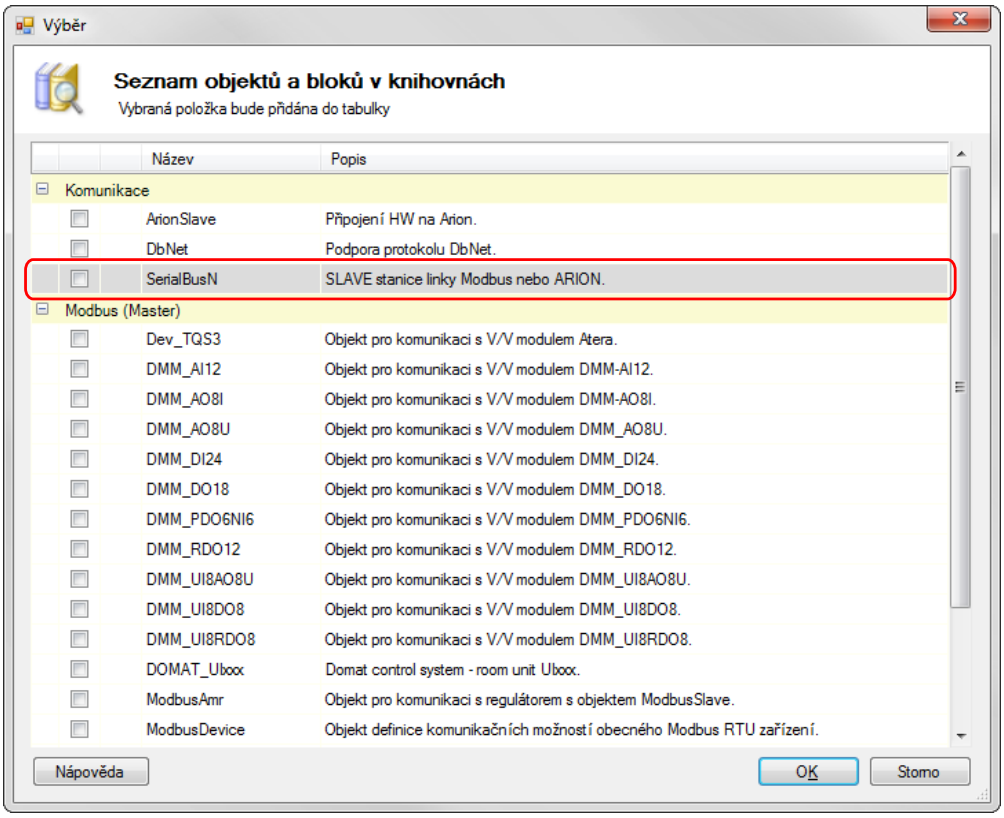
3.3.1 Definice komunikačního protokolu

Pro definici protokolů ARION a MODBUS RTU najednou lze použít komunikační objekt **`SerialBusN`**. Objekt do projektu vložíme z okna se seznamem komunikačních objektů, které lze otevřít vyvoláním kontextového menu nad složkou „Komunikace“ a výběrem položky „Přidat objekt“.



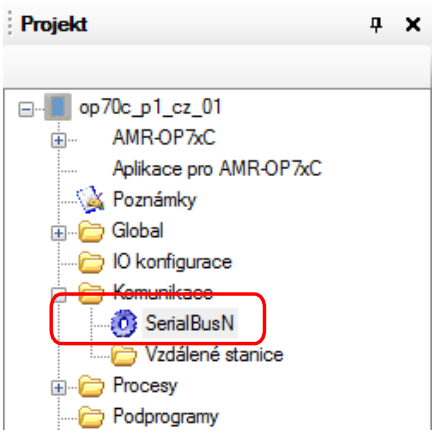
Obr. 4 – Přidání objektů do projektu

Po kliknutí na položku „Přidat objekt“ dojde k otevření okna „Seznam objektů v knihovnách“, kde vybereme objekt **`SerialBusN`** a výběr potvrdíme tlačítkem ***„OK“***.



Obr. 5 – Výběr objektu **SerialBusN**

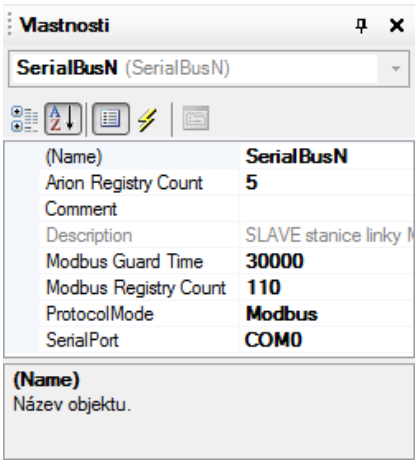
Po zavření okna s výběrem objektů se objekt **SerialBusN** zobrazí v okně „Projekt“ ve složce „Komunikace“.



Obr. 6 – Objekt **SerialBusN** ve složce Komunikace

Typ komunikačního protokolu a nastavení komunikačních parametrů v tuto chvíli nastavíme „napevno“. Později doprogramujeme možnost dynamické změny komunikačních parametrů.

Po kliknutí levým tlačítkem myši na objekt **SerialBusN** se v okně „Vlastnosti“ zobrazí nastavení komunikačních parametrů objektu. Vlastnosti objektu **SerialBusN** nastavíme dle následujícího obrázku.



Obr. 7 – Nastavení komunikačních parametrů objektu **`SerialBusN`**

Poznámka
*Komunikační protokol (vlastnost **`ProtocolMode`**) ponecháme „Modbus“ z důvodu pozdějších aktualizací SW v nástěnném ovladači (DetStudio komunikuje s nástěnnými ovladači právě prostřednictvím protokolu MODBUS RTU).*

Vlastnost **`Arion Registry Count`** nastavíme na hodnotu 5, protože budeme využívat:
- 1× registr typu DInt pro nastavení režimu místnosti a nastavení režimu ventilátoru,
- 1× registr typu DInt pro poskytnutí měřené hodnoty CO_2 nadřazenému systému,
- 1× registr typu DInt prostřednictvím kterého zadá nadřazený systém hodnotu pro upozornění na překročení meze CO_2,
- 1× registr typu Real pro poskytnutí teploty, měřené ovladačem, nadřazenému systému,
- 1× registr typu Real prostřednictvím kterého zadá nadřazený systém hodnotu žádané teploty pro zobrazení na displeji ovladače.

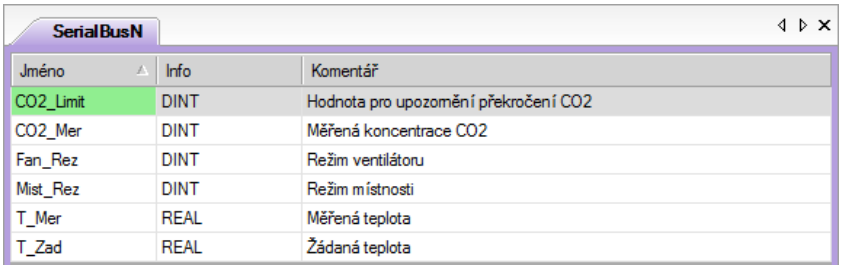
Vlastnost **`Modbus Registry Count`** nastavíme na hodnotu 110. Prvních 100 registrů (0 až 99) je vyhrazeno pro systémové účely. Jeden registr DInt či Real (v obou případech se jedná o 32 bit hodnotu) pak zabere dva MODBUS registry (16 bit hodnota).

Další komunikační parametry lze nastavit pomocí servisních obrazovek (viz návod na obsluhu k nástěnnému ovladači). V servisních obrazovkách bude nastavena adresa 1, komunikační rychlost 38400 a sudá parita pro MODBUS RTU.

3.3.2 Definice proměnných pro výměnu dat s řídicím systémem

Jak bylo uvedeno výše, budeme v ovladači (pro výměnu dat s řídicím systémem) používat pět 32bitových registrů. Význam jednotlivých registrů je popsán v předchozí kapitole.

Pro výměnu dat s řídicím systémem nadefinujeme v záložce **`SerialBusN`** (otevřeme dvojklikem levým tlačítkem myši na objekt **`SerialBusN`** v okně „Projekt“ – sekce „Komunikace“) šest proměnných (dvě proměnné budou mapovány do jednoho registru), dle následujícího obrázku.

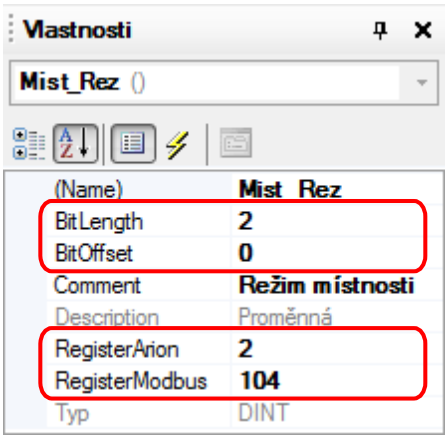


Obr. 8 – Definice proměnných v objektu **`SerialBusN`**

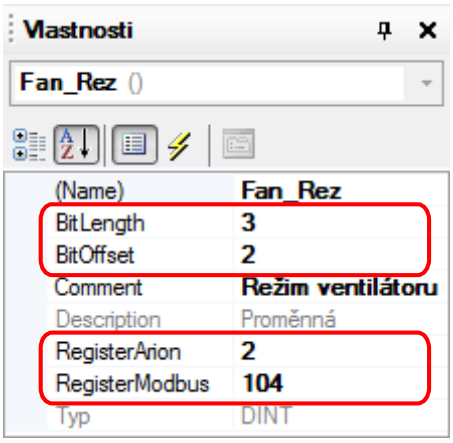
3.3.3 Mapování proměnných do registrů

Jelikož budeme z proměnné **`Mist_Rez`** (volba režimu místnosti) využívat pouze 2 bity a z proměnné **`Fan_Rez`** (volba režimu ventilátoru) pouze 3 bity, je výhodné poskytnout obě dvě proměnné do sítě ARION (MODBUS RTU) jako jeden registr, ve kterém bude využito 5 bitů. Mapování dvou a více proměnných do jednoho registru v síti ARION (MODBUS RTU) lze učinit v okně „Vlastnosti“ každé proměnné, nadefinované v objektu **`SerialBusN`**. Pro mapování slouží vlastnosti **`BitLenght`**, **`BitOffset`** a **`RegisterArion`** (**`RegisterModbus`**).

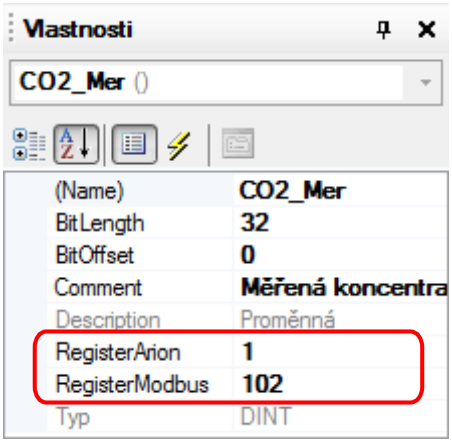
Jednotlivým proměnným upravíme vlastnosti dle následujících obrázků.



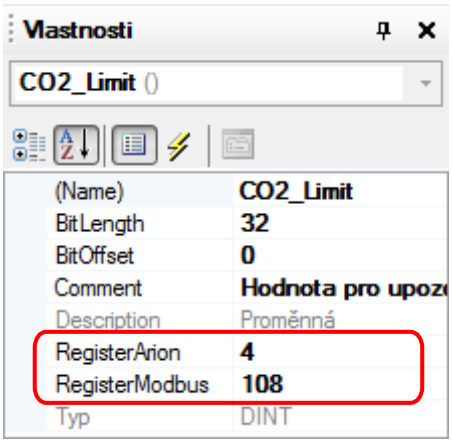
Obr. 9 – Mapování proměnné **`Mist_Rez`** do registrů



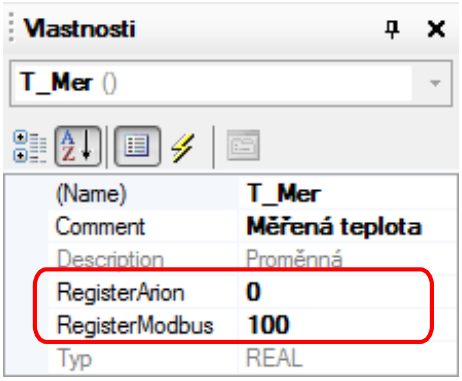
Obr. 10 – Mapování proměnné **`Fan_Rez`** do registrů



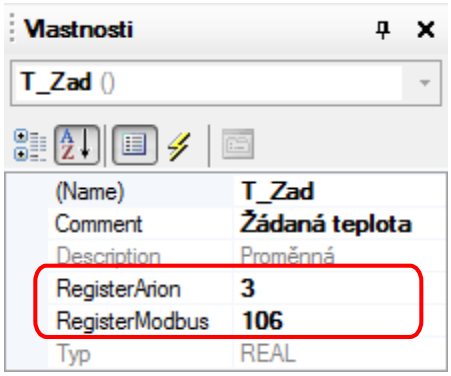
Obr. 11 – Mapování proměnné `CO2_Mer` do registrů



Obr. 12 – Mapování proměnné `CO2_Limit` do registrů



Obr. 13 – Mapování proměnné `T_Mer` do registrů



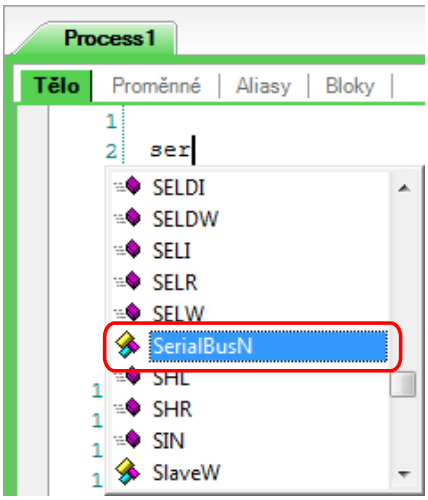
Obr. 14 – Mapování proměnné **`T_Zad`** do registrů

3.4 Vytvoření výkonné části programu AMR-OP70C

Pro vytvoření výkonné části programu, který se bude v **AMR-OP70C** periodicky neustále opakovat, použijeme proces s názvem „Process1“. V procesu naprogramujeme uložení měřené teploty (teplota bude korigována pomocí uživatelsky editovatelné korekce) a koncentrace CO_2 do proměnných, nadefinovaných v objektu **`SerialBusN`**. (viz kapitola 3.2 „Definice vnitřních proměnných“).

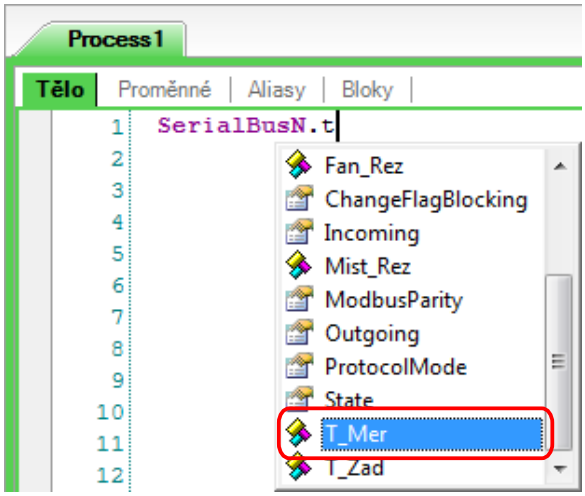
Poznámka
*Při programování lze použít tzv. intellisense nápovědu (klávesová zkratka **Ctrl+j**). Jedná se o nabídku se seznamem všech dostupných objektů, proměnných a vlastností, které lze v projektu využít. V seznamu je možné listovat pomocí šipek nahoru/dolů. Po otevření intellisense nápovědy je také možné začít psát název objektu. Tím je postupně vybrán požadovaný objekt a výběr pak postačí potvrdit klávesou **Enter**.*

Postupem uvedeným v poznámce výše vložíme do procesu objekt **`SerialBusN`**.



Obr. 15 – Použití intellisence nápovědy

Po zapsání znaku „.“ (tečka) za vloženým objektem, dojde opět k otevření intellisense nápovědy, kde bude seznam, vlastností (případně metod), které lze u daného objektu využít. Požadavek je na uložení měřené teploty do proměnné **`T_Mer`**. Ze seznamu tedy vybereme proměnnou **`T_Mer`** (obdobným způsobem jako byl do procesu umístěn objekt **`SerialBusN`**).



Obr. 16 – Výběr proměnné **T_Mer** v objektu **SerialBusN**

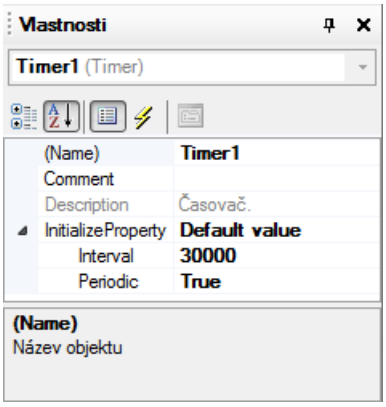
Výše uvedeným postupem naprogramujeme následující kód, který uloží měřené hodnoty do proměnných objektu **SerialBusN**.

```
SerialBusN.T_Mer = IO.DeviceTemperature + EEprom.Korekce;
SerialBusN.CO2_Mer = IO.CO2;
```

Měřenou teplotu korigujeme pomocí proměnné **Korekce** pro účely sjednocení případných rozdílů v měřených teplotách pomocí čidel od různých výrobců.

Vytvořený program bude s periodou procesu „Process1“ nastavovat bit č. 1 kanálu DI sítě ARION a bit č. 1 registru č. 8 v síti MODBUS RTU na hodnotu True (informace o zápisu do některého z registrů ze strany ovladače). Na základě bitu č. 1 bude nadřazený systém vyčítat veškerá data z ovladače. Pokud je informace o zápisu do registru ze strany ovladače nastavovaná příliš často, bude mít nadřazený systém snahu komunikovat všechna data z ovladače příliš často a může tak snadno dojít ke značnému vytížení sítě. Abychom předešli příliš častému nastavení informace o zápisu do registru ze strany ovladače, lze použít např. blok **Timer** a do registrů ukládat měřené hodnoty pouze po uplynutí času, zadaného v timeru.

Blok **Timer** vložíme do projektu jako globální a nastavíme mu vlastnosti dle následujícího obrázku.



Obr. 17 – Vlastnosti globálního bloku **Timer1**

V periodickém procesu budeme kontrolovat, zda došlo k tiknutí timeru či nikoliv. Pokud ano, provedeme zápis měřených hodnot do registrů.

```
Timer1();  // obsluha časovače pro periodické čtení
if Timer1.Out then  // Timer1 tiknul
     SerialBusN.T_Mer = IO.DeviceTemperature + EEprom.Korekce;
     SerialBusN.CO2_Mer = IO.CO2;
endif;
```

Blok **Timer1** je nutné spustit pomocí jeho vlastnosti **Action**. Tuto vlastnost nastavíme na hodnotu True již v „ProcessInit“ následovně.

```
Timer1.Action=1;
```

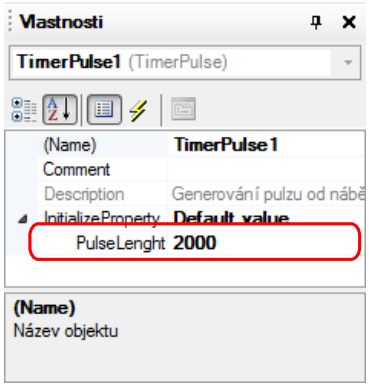
Signalizace překročení meze CO_2 naprogramujeme pomocí bloku **TimerPulse** (blok postačí vložit jako lokální). Do jeho vstupní vlastnosti **Input** dosadíme výraz:

```
IO.CO2 > SerialBusN.CO2Limit
```

Vlastnost **Out** bude zapisovat přímo do **IO.Buzzer** (aktivace zvukové signalizace). Výsledný kód tedy bude vypadat následovně.

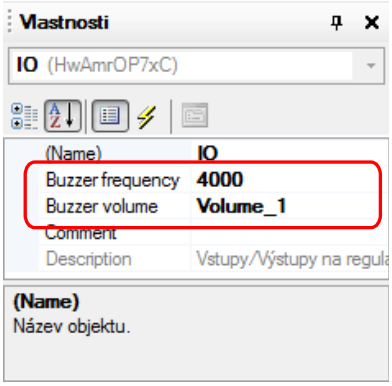
```
TimerPulse1(Input = IO.CO2 > SerialBusN.CO2Limit, Out => IO.Buzzer);
```

Dobu trvání zvukového signálu nastavíme v okně vlastností, bloku **TimerPulse1** (prostřednictvím vlastnosti **PulseLength**) dle následujícího obrázku.



Obr. 18 – Vlastnosti lokálního bloku **TimerPulse1**

Úroveň hlasitosti bzučáku a jeho frekvenci nastavíme v okně vlastností, které se zobrazí po kliknutí na text **IO.Buzzer** přímo v procesu strukturovaného textu.



Obr. 19 – Vlastnosti bzučáku

3.5 Tvorba obrazovek AMR-OP70C

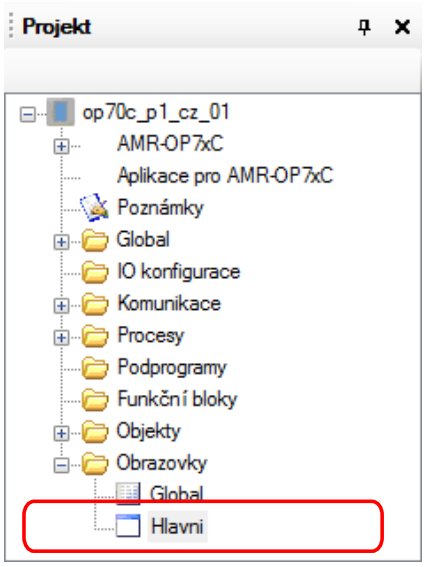
Tvorba obrazovek probíhá v **AMR-OP70C** stejně jako tvorba obrazovek u běžných stanic s dotykovým displejem. **Velkou výhodou** (stejně jako u ostatních stanic s dotykovým displejem z produkce firmy AMiT) **je možnost volby, zda bude mít ovladač orientaci displeje na výšku (Portrait) či na šířku (LandScape).**

Orientaci lze zvolit při založení projektu (viz kapitola 3.1 „Založení projektu“), nebo kdykoliv v průběhu tvorby projektu (v sekci „Obrazovky“ okna „Parametry projektu“, který lze zobrazit pomocí menu „Projekt/Nastavení“).

Na obrazovku lze vkládat jak standardní textové prvky, nabízené vývojovým prostředím DetStudio, tak prvky pro práci s obrázky či grafikou. Více informací o práci s obrázky na ovladačích **AMR-OP7x** lze nalézt v aplikační poznámce AP0041 „Návrh grafických prvků pro ovladače řady NOA7x“.

3.5.1 Nastavení režimu vytápění místnosti a ventilátoru

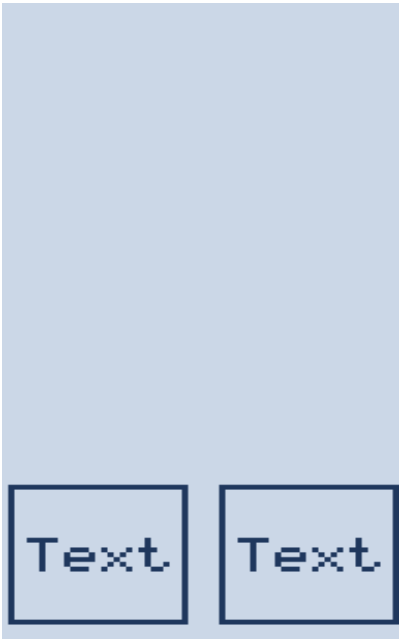
Pro možnost uživatelské změny režimu přejmenujeme (v záložce „Obrazovky) automaticky vytvořenou obrazovku „Screen1“ na „Hlavni“.



Obr. 20 – Zobrazení názvu obrazovky v okně „Projekt“

Požadovaný režim vytápění místnosti a ventilátoru lze nastavovat např. pomocí prvku **`CaseButton`** jehož popis lze nalézt v nápovědě k vývojovému prostředí DetStudio.

Na obrazovku umístíme dva prvky **`CaseButton`** (pro každý režim jeden) a pomocí osmi záchytných bodů (kolem prvků se zobrazí v případě, kdy je vložíme na obrazovku nebo v případě, kdy na prvek jedenkrát klikneme levým tlačítkem myši) upravíme jejich velikost do čtvercové podoby. Pozici prvků na obrazovce pak upravíme dle následujícího obrázku.

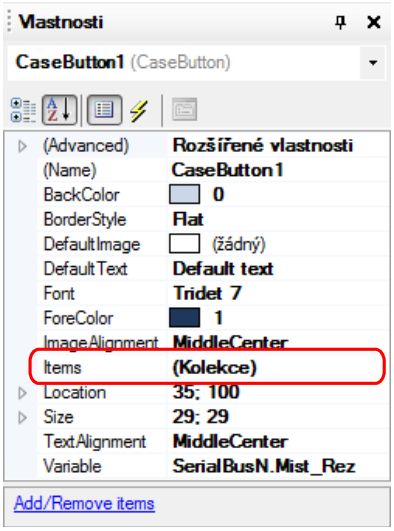


Obr. 21 – Umístění prvků **CaseButton** na obrazovce „Hlavni“

Pravý prvek **CaseButton** bude sloužit pro nastavení požadovaného režimu vytápění, levý prvek **CaseButton** bude sloužit pro nastavení požadovaného režimu ventilátoru.

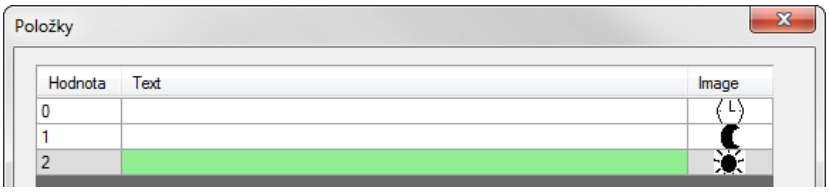
Kliknutím levým tlačítkem myši na pravý prvek **CaseButton** zobrazíme v okně „Vlastnosti“ seznam jeho vlastností. Prvku je nutné přiřadit proměnnou, jejíž hodnota se bude měnit v závislosti na počtu doteků na prvek. Klikneme na vlastnost **Variable** a ve volném poli se zobrazí tlačítko ..., pomocí kterého otevřeme okno „Výběr proměnné“. V okně bude mimo jiné zobrazen i seznam proměnných. Jelikož budeme pravý prvek **CaseButton** využívat pro nastavení režimu místnosti, vybereme z otevřeného okna proměnnou **Mist_Rez** objektu **SerialBusN**.

Po potvrzení výběru požadované proměnné nadefinujeme vzhled prvku pomocí vlastnosti **Items** v okně „Vlastnosti“.



Obr. 22 – Nastavení vzhledu prvku **CaseButton**

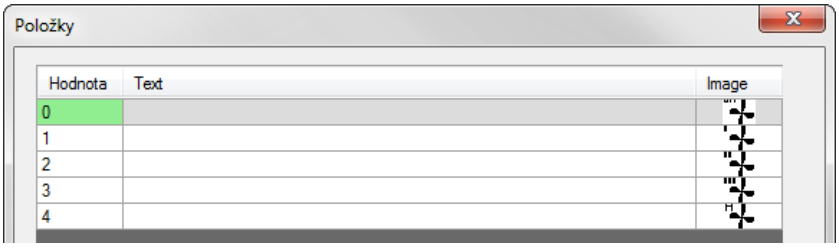
Kliknutím na tlačítko ... u výše zmíněné položky otevřeme okno „Položky“, které nastavíme dle následujícího obrázku.



Obr. 23 – Přiřazení obrázků jednotlivým hodnotám proměnné **`Mist_Rez`**

Editaci jednotlivých buněk v položkách okna lze provést pomocí klávesy **F2**.

Obdobně budeme postupovat s nastavením levého prvku **`CaseButton`**. Zde však v okně „Vlastnosti“ dosadíme za parametr **`Variable`** proměnnou **`Fan_Rez`** a jeho položky nastavíme dle následujícího obrázku.

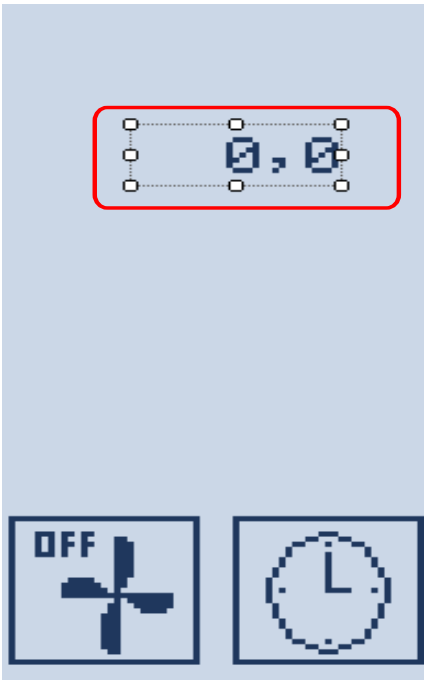


Obr. 24 – Přiřazení obrázků jednotlivým hodnotám proměnné **`Fan_Rez`**

Ikony použité v nastavení prvků **`CaseButton`** jsou součástí instalace DetStudia a lze je nalézt v adresáři „C:\Users\<user_name>\Documents\DetStudio\Icons\NOAx7x\“ tak, jak popisuje aplikační poznámka „AP0041 – Návrh grafických prvků pro ovladače řady NOA7x“.

3.5.2 Zobrazení teploty měřené čidlem AMR-OP70C

Teplotu, měřenou čidlem (integrovaným v **AMR-OP70C**), získáváme ve výkonné části programu a ukládáme ji do proměnné **`SerialBusN.T_Mer`** (viz kapitola 3.4 „Vytvoření výkonné části programu AMR-OP70C“). Pro zobrazení hodnoty, proměnné **`T_Mer`**, použijeme prvek **`NumericView`**. Na obrazovku jej umístíme tažením z okna „Toolbox“ (sekce „Basic“) na požadovanou pozici (viz následující obrázek).

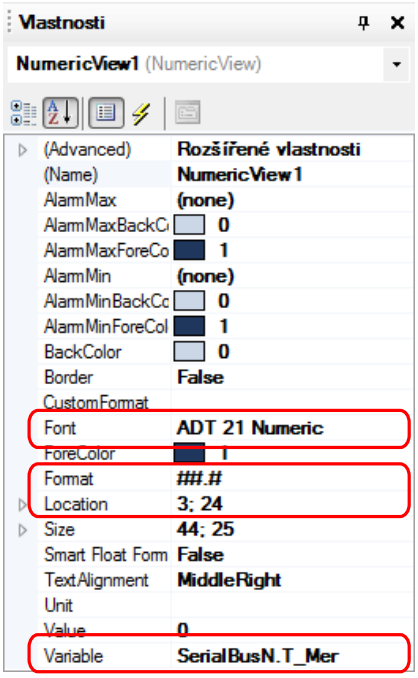


Obr. 25 – Umístění prvku **NumericView** na obrazovku

Dvojklikem levým tlačítkem myši na prvek **NumericView** otevřeme okno „Výběr proměnné“ ze kterého vybereme proměnnou **SerialBusN.T_Mer**.

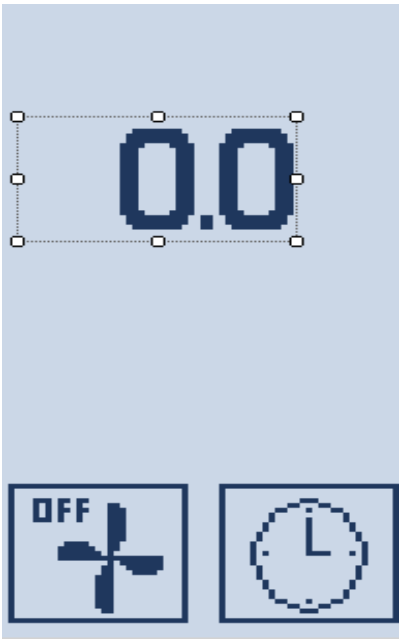
Neprovádíme přímo vazbu na **IO.DeviceTemperature**, protože měřenou hodnotu navíc korigujeme pomocí proměnné **EEprom.Korekce**. Stejná (korigovaná) hodnota je zasílána také nadřazenému systému.

Po přiřazení proměnné nastavíme prvku **NumericView** také jeho vzhled. Pomocí vlastnosti **Font** (výběr fontu pro zobrazení) vybereme „ADT 21“ a ve vlastnosti **Format** (formát zobrazení) ponecháme výchozí řetězec ##.# (zobrazení proměnné v rozsahu -9.9 až 99.9). Po úpravě vlastnosti Location (pozice prvku na obrazovce) na hodnotu „3;24“ bude okno „Vlastnosti“ prvku **NumericView** vypadat dle následujícího obrázku.



Obr. 26 – Nastavení vlastností prvku **`NumericView`**

Po nastavení vlastností prvku **`NumericView`** dle obrázku výše bude obrazovka „Hlavni“ vypadat následovně.



Obr. 27 – Obrazovka „Hlavni“ se zobrazením teploty měřené integrovaným čidlem

3.5.3 Zobrazení statických textů

Uživatel musí být informován o jednotkách, ve kterých jsou mu veličiny zobrazeny. Z důvodu omezeného prostoru na obrazovce mu zobrazíme jednotky v jiné velikosti. Je tedy nutné použít pro jejich zobrazení samostatný statický text. Statický text je možné vytvořit pomocí prvku **`Label`** ze sekce „Basic“ okna „Toolbox“. Prvek umístíme na požadovanou pozici a po dvojkliku levým tlačítkem myši na prvek mu zadáme text „°C“. Obrazovka „Hlavni“ nyní bude vypadat následovně.

Obr. 28 – Obrazovka „Hlavni“ se statickým textem „°C“

Dalším požadavkem je zobrazení žádané teploty, kterou zasílá nadřazený systém. Pro zobrazení hodnoty opět využijeme prvek **`NumericView`**. Jednotky však tentokrát zadáme přímo do vlastnosti **`Unit`** v okně vlastností. Obrazovka „Hlavni“ pak bude vypadat následovně.

Obr. 29 – Obrazovka hlavní se všemi požadovanými údaji

Výše uvedený příklad pro **AMR-OP70C** je součástí přílohy této aplikační poznámky. Jedná se o soubor op70c_p1_cz_xx.dsox.

3.6 Generace aplikace pro AMR-OP70C

Pro generaci je nutné mít na PC nainstalován balíček „DetStudioTools.exe“, který není součástí instalace DetStudia. Balíček je dostupný (po registraci) na webu amitomation.cz.

Výsledkem generace je soubor s koncovkou *.bin, který má stejný název, jako je název projektu DetStudia.

3.7 Zavedení aplikace do AMR-OP70C

Zavedení aplikace do **AMR-OP70C** se provádí prostřednictvím rozhraní RS485, prostřednictvím protokolu MODBUS RTU. Na straně PC lze využít převodník USB<->RS485 (např. **SB485s**, který nabízí firma AMiT). Postup zavedení aplikace je stejný jako u jiných stanic typu AMR bez ethernetového rozhraní a je popsán v nápovědě k části EsiDet vývojového prostředí DetStudio.

4 Příklad obsluhy AMR-OP70C ve stanicích s NOS

Pro komunikaci **AMR-OP70C** s řídicím systémem lze využít komunikační protokol ARION nebo MODBUS RTU. V našem případě bude provedena ukázka s využitím obou komunikačních protokolů.

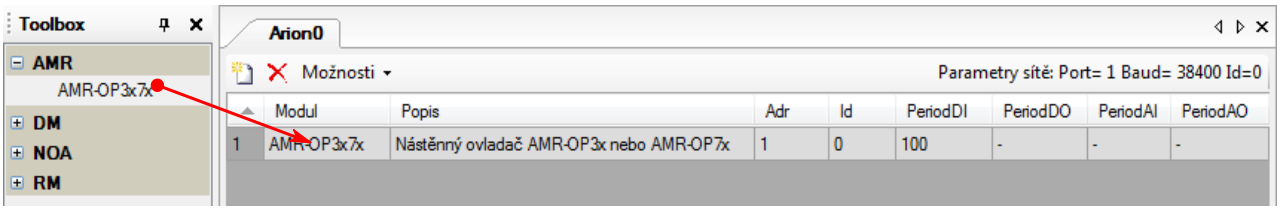
4.1 Využití protokolu ARION

Následující příklad je k dispozici v příloze této aplikační poznámky. Jedná se o soubor s názvem rs_p1_cz_xx.dso.

4.1.1 Definice AMR-OP70C v DetStudiu

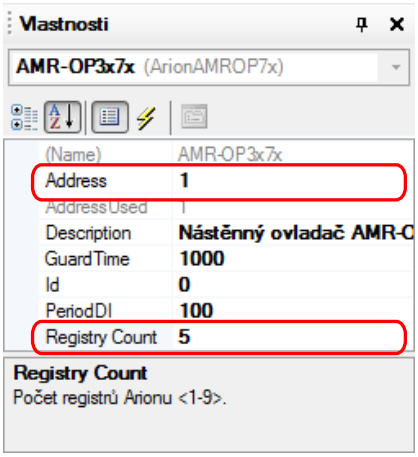
Definici **AMR-OP70C** provedeme v řídicím systému firmy AMiT pomocí tabulky ARION (viz „AP0025 – Komunikace v síti ARION – definice tabulkou“). Tabulku zobrazíme dvojklikem na položku „Arion0“ (lze nalézt ve složce „Komunikace/Arion“ okna projektu).

V případě, že je v **AMR-OP70C** zavedena aplikace s objektem **`SerialBusN`**, nadefinujeme **AMR-OP70C** v aplikaci pro řídicí systém přetažením modulu **`AMR-OP3x7x`** do tabulky (viz následující obrázek).



Obr. 30 – Definice **AMR-OP70C** v řídicím systému

Po umístění modulu **`AMR-OP3x7x`** do tabulky nadefinujeme v okně vlastností modulu počet registrů, který je na **AMR-OP70C** k dispozici. Tento se musí shodovat s počtem registrů, nastavených v návrhu aplikace pro **AMR-OP70C** (viz kapitola 3.3.1 „Definice komunikačního protokolu“). Jelikož jsme v aplikaci pro **AMR-OP70C** nadefinovali 5 registrů, bude okno vlastností vypadat následovně.



Obr. 31 – Nastavení adresy a počtu registrů, naprogramovaných v **AMR-OP70C**

Mimo nadefinované registry poskytuje **AMR-OP70C** do sítě ARION také kanál digitálních vstupů a digitálních výstupů.

Z kanálu digitálních vstupů jsou využity první tři bity, jejichž význam je následující:
- DI.0 – došlo k restartu **AMR-OP70C**,
- DI.1 – v **AMR-OP70C** došlo k zápisu do libovolné proměnné namapované do registru v objektu **`SerialBusN`**,
- DI.2 – došlo k výpadku komunikace **AMR-OP70C** s řídicím systémem.

Čtení kanálu digitálních vstupů se provádí s periodou zadanou v okně vlastností modulu **`AMR-OP3x7x`** při jeho definici v tabulce s definicí sítě ARION.

Kanál digitálních výstupů slouží pro vynulování příznaků, které poskytuje kanál digitálních vstupů. Jakmile tedy řídicí systém zaregistruje nastavení některého z digitálních vstupů na hodnotu True, měl by provést odpovídající kroky a následně daný vstup pomocí kanálu digitálních výstupů vynulovat. Význam jednotlivých bitů kanálu je následující:
- DO.0 – nulování příznaku restartu,
- DO.1 – nulování příznaku změny proměnné objektu **`SerialBusN`** v **AMR-OP70C**,
- DO.2 – nulování příznaku výpadku komunikace.

4.1.2 Způsob komunikace s AMR-OP70C

Pro komunikaci s **AMR-OP70C** lze využít dvou způsobů výměny informací s řídicím systémem.
- Zjednodušený způsob
- Doporučený způsob

V obou dvou případech funguje komunikace tak, že se jedním požadavkem provede jak zápis, tak čtení požadovaných dat.

Zjednodušený způsob
Pro komunikaci s **AMR-OP70C** jsou využity moduly **`ARI_RegIn`**/**`ARI_RegOut`**, které se umístí do periodického procesu. Pomocí speciálního modulu **`ARI_Trig`** pak bude zadáván požadavek na čtení/zápis dat.

Doporučený způsob
Pro komunikaci s **AMR-OP70C** je využit kanál digitálních vstupů. Na základě stavu jeho signálů se bude zadávat požadavek na načtení dat z **AMR-OP70C**. Periodicky se tedy zjišťuje pouze stav vstupu DI.1 (viz předchozí kapitola). Na základě nastavení tohoto vstupu je provedeno načtení hodnot z **AMR-OP70C** pomocí modulu **`ARI_RegIn`** a pomocí modulu **`ARI_Trig`**.

Zápis je proveden na základě změny hodnoty na straně řídicího systému pomocí modulu **`ARI_RegOut`** opět společně s modulem **`ARI_Trig`**.

Poznámka
*Registry nelze z **AMR-OP70C** vyčítat jednotlivě. Při použití modulu **`ARI_Trig`** dojde vždy k vykomunikování všech registrů, které jsou v **AMR-OP70C** nadefinovány. Vlastní čtení registrů z bufferu řídicího systému a jejich následné uložení do proměnné v řídicím systému již lze v případě potřeby provést i po jednotlivých registrech, použitím více modulů **`ARI_RegIn`**.*

4.1.3 Vytvoření programu pro komunikaci s AMR-OP70C

Pro komunikaci řídicího systému s **AMR-OP70C** využijeme doporučený způsob (viz předchozí kapitola).

Do periodického procesu vložíme modul **`ARI_State`**, kterým budeme zjišťovat, zda probíhá komunikace s ovladačem. Dále použijeme modul **`ARI_DigIn`**, pomocí kterého budeme zjišťovat, zda došlo k zápisu do některé proměnné mapované v objektu **`SerialBusN`** do registru, nadefinovaného v ovladači. Na základě příznaku o zápisu do proměnné v objektu **`SerialBusN`** provedeme načtení registrů.

Zápis požadované hodnoty provedeme, až zaznamenáme její změnu a to pouze v případě, že není nastaven požadavek na čtení (při čtení se provádí i zápis).

```
// stav AMR-OP70C a stav přenosu registrů
ARI_State 1, OP_stav, 4, OP_R_pren

// zjištění změny v AMR-OP70C
ARI_DigIn 1, 0, priznaky, 0x00000000

// zápis žádané teploty (pouze v případě, že se změní její hodnota)
Let @SetPointTmp = (SetPoints[0,0] > (SetPoint_Old[0,0] + 0.1)) or (SetPoints[0,0] <
(SetPoint_Old[0,0] - 0.1))  // kontrola, zda došlo ke změně
Let @LimitCO2 = (SetPoints[1,0] > (SetPoint_Old[1,0] + 0.1)) or (SetPoints[1,0] <
(SetPoint_Old[1,0] - 0.1))  // kontrola, zda došlo ke změně
Let @OP_Zap_Reg = @SetPointTmp or @LimitCO2  // nastavení příznaku, že (ne)došlo ke
                                             // změně

// příznak o změně hodnot z ovladače, nebo došlo ke změně žádané hodnoty ze systému
Let @OP_RW = (příznaky > 0) or @OP_Zap_Reg
If @OP_RW  // na základě příznaku se spustí komunikace
     // komunikace registrů, kterou v dalších řádcích spustíme, probíhá tak, že se na
     // jeden požadavek provede zápis i čtení. Proto se před samotným vyvoláním
     // komunikace provede vložení dat do zápisového bufferu
     ARI_RegOut 1, 3, 2, SetPoints[0,0], SetPointType[0,0], 5
     // pokyn ke komunikaci registrů AMR-OP70C
     ARI_Trig 1, 4
     // potvrzení přijetí informace a nulování příznaku o změně, restartu či výpadku
     Let Priznak_nul = 7
     ARI_DigOut 1, 0, 3, Priznak_nul, 0x00000000
     // pokyn k vynulování příznaku
     ARI_Trig 1, 3
     If @OP_Zap_Reg  // pokud je požadavek na zápis
          Let SetPoint_Old[0,0] = SetPoints[0,0]  // uloží se zapsaná hodnota
          Let SetPoint_Old[1,0] = SetPoints[1,0]  // uloží se zapsaná hodnota
     EndIf
EndIf

// čtení z bufferu se aktivuje, až se vykomunikují data
Let @OP_Cti_Reg = OP_stav.0 and not(OP_R_pren.0)

If @OP_Cti_Reg
     // načtení měřené teploty
     ARI_RegIn 1, 0, 1, T_Mist_Mer, NONE[0,0], 5

     // načtení měřené koncentrace CO2
     ARI_RegIn 1, 1, 1, CO2, NONE[0,0], 3

     // načtení režimu místnosti a ventilátoru
     ARI_RegIn 1, 1, 1, OP_Rez, NONE[0,0], 4
EndIf
```

Poznámka

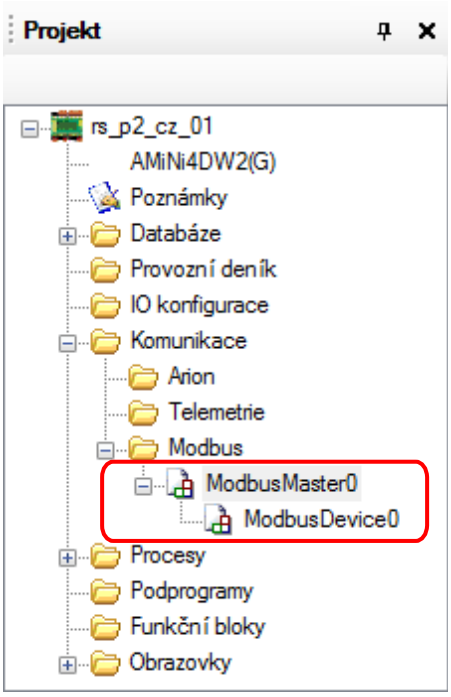
*Použitím modulu **ARI_Trig** dojde k vyvolání příkazu pro čtení registrů i pro jejich zápis současně. Pokud programátor využívá jednu proměnnou pro čtení i pro zápis a pomocí modulu **ARI_RegOut** do této proměnné zapíše. Dojde po vyvolání modulu **ARI_Trig**, k zápisu této proměnné do příslušného registru v **AMR-OP70C**. Pokud tedy uživatel změnil hodnotu takového registru a v řídicím systému byla hodnota shodného registru změněna modulem **ARI_RegOut** dříve, než stihl řídicí systém změnu ze strany uživatele zaznamenat, dojde použitím modulu **ARI_Trig** k přepsání uživatelem zadané hodnoty v **AMR-OP70C** hodnotou, která byla zadána modulem **ARI_RegOut** v řídicím systému. Pro zjednodušení řešení takovéto problematiky je doporučeno používat zvlášť registry pro čtení a zvlášť registry pro zápis.*

4.2 Využití protokolu MODBUS RTU

Následující příklad je k dispozici v příloze této aplikační poznámky. Jedná se o soubor s názvem rs_p2_cz_xx.dso.

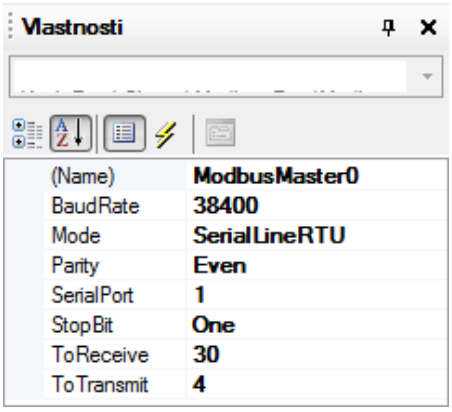
4.2.1 Definice AMR-OP70C v DetStudiu

Definici **AMR-OP70C** provedeme v řídicím systému firmy AMiT pomocí tabulky MODBUS (viz „AP0008 – Komunikace v síti MODBUS RTU (PseDet)“). Pomocí kontextového menu (vyvolaného nad složkou „Modbus“) v okně projektu vložíme do složky „Modbus“ definici master komunikace v síti MODBUS RTU. Pomocí kontextového menu vyvolaného nad vloženým objektem „ModbusMaster0“ pak vložíme definici zařízení (ModbusDevice), se kterým má řídicí systém v rámci sítě MODBUS RTU komunikovat.



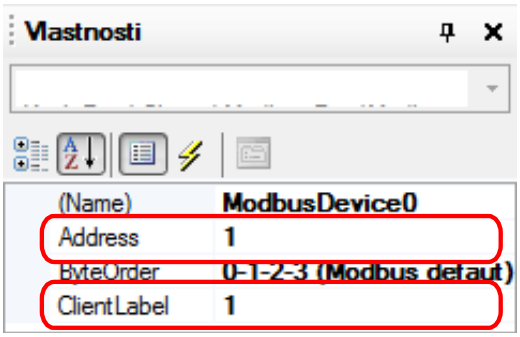
Obr. 32 – Definice **AMR-OP70C** v řídicím systému

Objektu „ModbusMaster0“ nadefinujeme v okně vlastností komunikační parametry, které musí být shodné s komunikačními parametry, nastavenými v **AMR-OP70C**. Ve vlastnosti „SerialPort“ nastavíme rozhraní řídicího systému, prostřednictvím kterého se bude komunikovat protokolem MODBUS RTU. Při požadavku na komunikaci prostřednictvím RS485 je nutné za parametr zadat hodnotu 1.



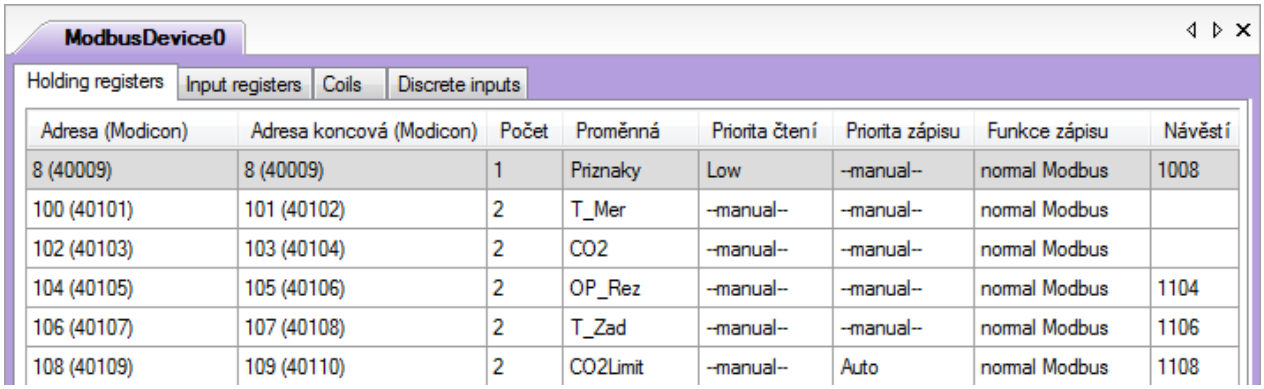
Obr. 33 – Nastavení vlastností objektu „ModbusMaster0“

Objektu „ModbusDevice0“ nastavíme v okně vlastností stejnou adresu, jako je adresa **AMR-OP70C** v síti MODBUS RTU. Vlastnost `ClientLabel` nastavíme na hodnotu rozdílnou od „-1“ pro účely pozdějšího vyhodnocování komunikace s **AMR-OP70C**.



Obr. 34 – Nastavení parametrů objektu „ModbusDevice0“

Dvojklikem na objekt „ModbusDevice0“ otevřeme záložku, kde nadefinujeme registry pro komunikaci z/do **AMR-OP70C** tak, jak byly nadefinovány v kapitole 3.3.2 „Definice proměnných pro výměnu dat s řídicím systémem“. Pro komunikaci řídicího systému s **AMR-OP70C** využijeme doporučený způsob komunikace (viz kapitola 4.2.2 „Způsob komunikace s AMR-OP70C“). Pro doporučený způsob komunikace je nutné nadefinovat také komunikaci systémového registru č. 8.



Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěstí
8 (40009)	8 (40009)	1	Priznaky	Low	--manual--	normal Modbus	1008
100 (40101)	101 (40102)	2	T_Mer	--manual--	--manual--	normal Modbus	
102 (40103)	103 (40104)	2	CO2	--manual--	--manual--	normal Modbus	
104 (40105)	105 (40106)	2	OP_Rez	--manual--	--manual--	normal Modbus	1104
106 (40107)	107 (40108)	2	T_Zad	--manual--	--manual--	normal Modbus	1106
108 (40109)	109 (40110)	2	CO2Limit	--manual--	Auto	normal Modbus	1108

Obr. 35 – Definice registrů v objektu „ModbusDevice0“

Mimo systémový registr a limit pro signalizaci překročení úrovně CO_2 se bude veškerá komunikace spouštět manuálně. U vybraných registrů zadáme libovolnou hodnotu do sloupečku „Návěstí“ (např. dle obrázku výše). Odpovídající hodnoty pak použijeme v modulu pro spuštění komunikace či vyhodnocení stavu komunikace.

Poznámka
Pokud se má do ovladače zapsat jednorázové nastavení (např. z displeje nadřazeného systému), lze s výhodou použít prioritu zápisu „Auto“. Příkladem může být zápis limitu pro signalizaci překročení úrovně CO_2 v příkladu. Priorita zápisu „Auto“ však není vhodná pro registry s namapovanou proměnnou, do které se zapisuje v periodickém procesu. Při nastavené prioritě zápisu „Auto“, dojde k vyvolání požadavku na zápis do periferie s každým zápisem do namapované proměnné (i když se její hodnota nezmění). Zbytečně tak naroste komunikace v síti. Příkladem může být zápis žádané teploty v místnosti. Proměnná s žádanou teplotou v místnosti je ve většině případů výstupní hodnota modulu ***`DayPlan(2)`****. Jelikož je modul* ***`DayPlan(2)`*** *zpracováván v periodickém procesu, nastaví se požadavek na zápis hodnoty s periodou procesu, kde je modul umístěn. Proto je v příkladu u definice zápisu žádané teploty v místnosti použito návěstí s prioritou zápisu „--manual--“.*

Mimo nadefinované registry poskytuje objekt **`SerialBusN`** do sítě MODBUS RTU také skupinu systémových holding registrů.

Ze systémových registrů lze využít registr č. 8 (SystemStatus), u kterého jsou využity první tři bity. Jejich význam je následující:
- SystemStatus.0 – došlo k restartu ovladače,
- SystemStatus.1 – v ovladači došlo k zápisu do libovolné proměnné namapované do registru v objektu **`SerialBusN`**,
- SystemStatus.2 – došlo k výpadku komunikace ovladače s řídicím systémem.

Nulování výše uvedených příznaků se provede zápisem hodnoty True do stejného bitu registru „SystemStatus“. Jakmile tedy řídicí systém zaregistruje nastavení některého z digitálních vstupů na hodnotu True, měl by provést odpovídající kroky a následně daný příznak vynulovat zápisem hodnoty True do stejného bitu.

4.2.2 Způsob komunikace s AMR-OP70C

Pro komunikaci s **AMR-OP70C** lze využít dvou způsobů výměny informací s řídicím systémem.
- Zjednodušený způsob
- Doporučený způsob

Zjednodušený způsob
Požadované hodnoty se z **AMR-OP70C** čtou s periodou, nastavenou v definici odpovídajícího registru či registrů. Zápis požadovaných hodnot do **AMR-OP70C** se provádí při každém zápisu do proměnné, která je na straně řídicího systému mapována do sítě MODBUS RTU.

Uvedený způsob komunikace je sice velmi jednoduchý pro programování řídicího systému, avšak zbytečně vytěžuje komunikační síť dotazy na nastavené či měřené hodnoty i v případě, kdy se na straně ovladače nic nezměnilo. Využití zjednodušeného způsobu komunikace nedoporučujeme pro případ rozsáhlejší sítě MODBUS RTU.

Doporučený způsob
Pro komunikaci s **AMR-OP70C** je využit registr č. 8 (SystemStatus). Na základě stavu jeho bitů se bude zadávat požadavek na načtení dat z **AMR-OP70C**. Periodicky se tedy zjišťuje pouze stav registru č. 8. Vlastní komunikace se pak spustí až na základě nastavení bitu č. 1. Na základě nastavení tohoto bitu bude provedeno načtení hodnot z **AMR-OP70C.**

V této aplikační poznámce bude dále řešen doporučený způsob komunikace.

4.2.3 Vytvoření programu pro komunikaci s AMR-OP70C

Do periodického procesu vložíme modul **MdbmReqSt**, kterým budeme zjišťovat, zda probíhá komunikace systémového registru č. 8. Pomocí systémového registru č. 8 budeme zjišťovat, zda došlo k zápisu do některé proměnné, mapované v objektu **SerialBusN** do registru nadefinovaného v ovladači. Na základě příznaku o zápisu do proměnné v objektu **SerialBusN** provedeme načtení registrů.

Zápis požadované hodnoty provedeme, až zaznamenáme její změnu.

```
// zjištění stavu komunikace
MdbmReqSt 1, 1008, OP_R_pren, NONE

If not (OP_R_pren.0)  // pokud aktuálně neprobíhá komunikace
     If Priznaky > 0  // pokud je změna ze strany ovladače
          MdbmReqSt 1, 1104, OP_Cteni, NONE  // stav komunikace posledního registru
          If not(OP_Cteni.0)  // pokud se poslední registr nekomunikuje
               // pokud byl poslední registr vykomunikován úspěšně a je acknowledge
               If OP_Cteni.1 and @Acknowledge
                    Let Priznaky = 7  // vynuluje se systémový registr
                    MdbmWrite 1, 1008, NONE
                    Let @Acknowledge = false // shodí se příznak o změnách z ovladače
               Else
                    // čtení měřené teploty, koncentrace CO2 a režimu
                    Let @Acknowledge = true // nastaví se příznak o změnách z ovladače
                    MdbmMark 1, 3, 100, 6, MarkRes
               EndIf
          EndIf
     EndIf

EndIf

// kontrola, zda došlo ke změně žádané teploty ze strany systému
Let @SetPointTmp = (T_Zad > (T_Zad_Old + 0.1)) or (T_Zad < (T_Zad_Old - 0.1))

// pokud změna ze systému nebo restart ovladače a není nahozen příznak acknowledge
If (@SetPointTmp or bool(Priznaky & 5))
     MdbmWrite 1, 1106, NONE
     Let T_Zad_Old = T_Zad
EndIf
```

5 Rozšíření funkcí aplikace pro AMR-OP70C

5.1 Nastavení komunikačních parametrů AMR-OP70C z displeje

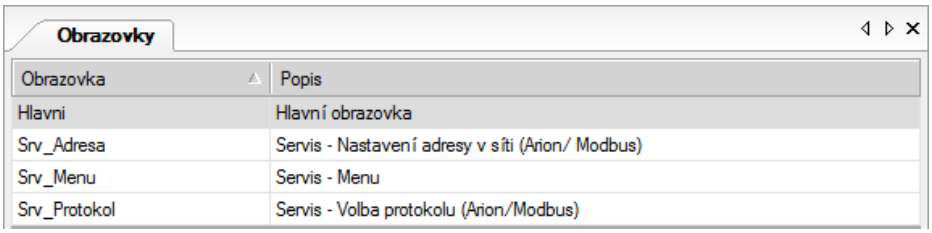
Nastavení komunikačních parametrů **AMR-OP70C** lze (pomocí parametrů objektu **`SerialBusN`**) učinit také přímo z obrazovek.

Příklad je k dispozici v příloze této aplikační poznámky. Jedná se o soubor op70c_p1_cz_xx.dsox.

Pozor
Komunikační parametry jsou umístěny v paměti EEPROM, která má omezené množství zápisů!

V případě vytvářené aplikace je požadavek na možnost dynamické změny komunikačního protokolu i na možnost dynamické změny komunikačních parametrů.

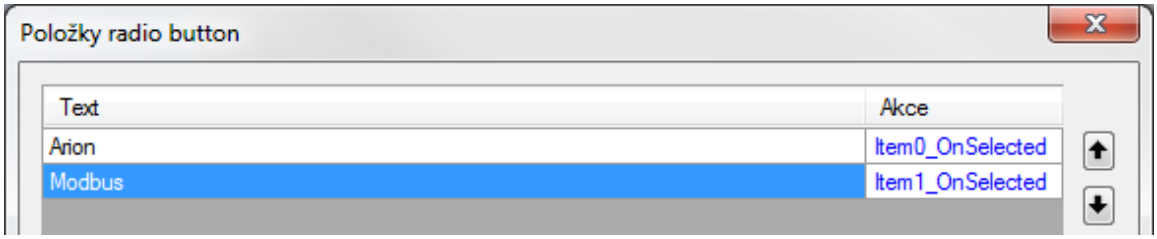
Pro možnosti nastavení veškerých komunikačních parametrů nadefinujeme další 3 obrazovky dle následujícího obrázku.



Obr. 36 – Seznam obrazovek pro nastavení komunikačních parametrů

5.1.1 Obrazovka pro volbu typu protokolu

Pro volbu typu protokolu, prostřednictvím kterého si bude **AMR-OP70C** vyměňovat data s nadřazeným systémem (např. jakýkoliv řídicí systém z produkce firmy AMiT), použijeme obrazovku „Srv_Protokol“. Volbu komunikačního protokolu na obrazovce naprogramujeme pomocí prvku **`RadioButton`**, který na obrazovku umístíme tažením z okna „Toolbox“ (sekce „General“). Dvojklikem na umístěný prvek **`RadioButton`** otevřeme okno s definicí jednotlivých položek prvku. Nastavení provedeme dle následujícího obrázku.



Obr. 37 – Nastavení položek prvku **`RadioButton`**

Po potvrzení nastavení tlačítkem ***„OK“*** dojde k otevření skriptovací části obrazovky, ve které budou vloženy události **`RadioButton1_Item0_OnSelected`** a **`RadioButton1_Item1_OnSelected`**. V kódu vytvářené aplikace je však nepoužijeme. Proto automaticky nadefinované události smažeme a vrátíme se zpět do návrhové části obrazovky.

Pro potvrzení nastaveného protokolu a návrat zpět na obrazovku s prvkem **`MenuScreen`** použijeme prvek **`Button`**, který je k dispozici v sekci „TouchScreen“ okna „Toolbox“. Prvek umístíme na obrazovku a dvojklikem na prvek otevřeme skriptovací část obrazovky s automaticky vytvořenou událostí **`Button1_OnButtonDown`**. Prostřednictvím vytvořené události naprogramujeme uložení požadovaného typu komunikačního protokolu a návrat na obrazovku s prvkem **`MenuScreen`**.

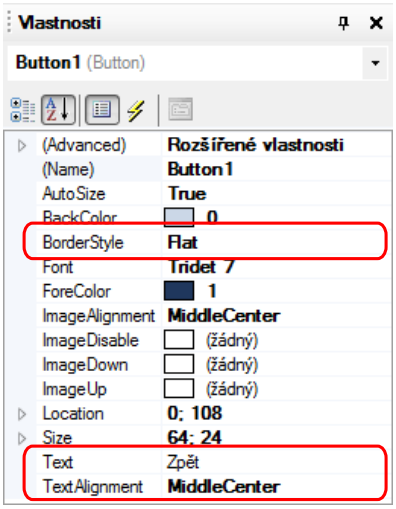
V závislosti na vybrané položce prvku **RadioButton** bude v této události uložena hodnota 0 nebo 1 do vlastnosti **SerialBusN.ProtocolMode**. Hodnota 0 odpovídá v objektu **SerialBusN** komunikačnímu protokolu ARION, hodnota 1 odpovídá v objektu **SerialBusN** protokolu MODBUS RTU. Zároveň s uložením požadovaného typu protokolu naprogramujeme do stejné události také odchod z obrazovky. Výsledný skript tedy bude vypadat následovně.

```
event Button1_OnButtonDown()
     SerialBusN.ProtocolMode = RadioButton1.SelectedIndex;
     Srv_Menu.Show();
end;
```

Poznámka

*Zápis do vlastnosti **SerialBusN.ProtocolMode** neprovedeme přímo v událostech **ItemX_OnSelected** prvku **RadioButton**, protože by při častém přepínání mezi jednotlivými typy komunikačního protokolu zbytečně docházelo k zápisu do paměti EEPROM. Z toho důvodu je výhodnější provést zápis do vlastnosti **SerialBusN.ProtocolMode** až při odchodu z obrazovky, kdy uživatel nastavil požadovaný typ protokolu.*

Po kliknutí na prvek **Button** upravíme jeho vzhled (v okně vlastností) dle následujícího obrázku.



Obr. 38 – Nastavení vlastností prvku **Button**

V dalším kroku umístíme do horní části obrazovky prvek **Label**, který nalezneme v okně „Toolbox“ v sekci „Basic“. Provedeme dvojklik na něj a do editačního pole napíšeme text „Protokol“.

Výsledná obrazovka pro nastavení volby komunikačního protokolu pak bude vypadat následovně.

Obr. 39 – Obrazovka s nastavením volby komunikačního protokolu

Závěrem doplníme do události „OnOpen“ obrazovky skript, který zajistí, aby se po otevření této obrazovky zobrazil aktuálně nastavený komunikační protokol. Kamkoliv do události **`OnOpen`** obrazovky tedy doplníme následující skript:

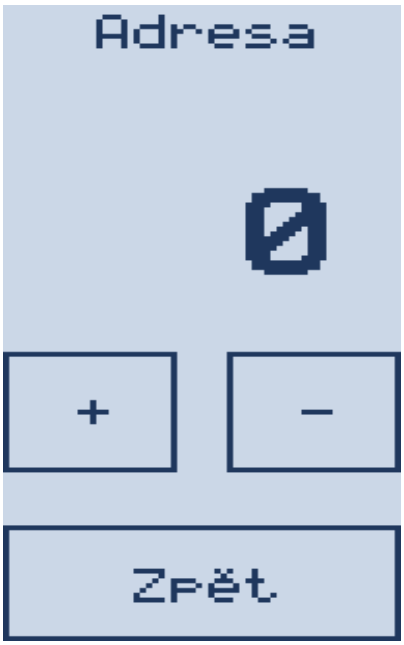
```
RadioButton1.SelectedIndex = SerialBusN.ProtocolMode;
RadioButton1.Refresh();
```

5.1.2 Obrazovka pro nastavení adresy AMR-OP70C

Pro nastavení adresy, se kterou bude **AMR-OP70C** připojen do vybrané sítě, použijeme obrazovku „Srv_Adresa“. Rozmezí použitelných adres budeme nastavovat v závislosti na vybraném komunikačním protokolu.

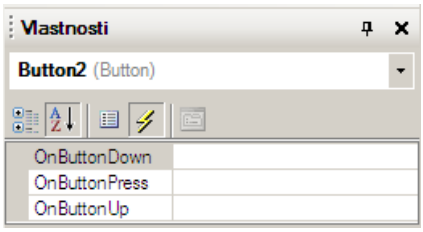
Ještě před vlastním programováním nastavení adresy **AMR-OP70C** zkopírujeme (použitím klávesové zkratky **Ctrl+c**) z obrazovky „Srv_Protokol“ tlačítko s textem „Zpět“ a **`Label`** s textem „Protokol“ a vložíme je (použitím klávesové zkratky **Ctrl+v**) na obrazovku „Srv_Adresa“ (na stejnou pozici). Text „Protokol“ v prvku **`Label`** změníme na „Adresa“.

Vlastní nastavení adresy pak budeme provádět držením jednoho ze dvou tlačítek, které na obrazovku umístíme ze sekce „TouchScreen“ okna „Toolbox“. Tlačítka budeme využívat pro inkrementaci/dekrementaci adresy. Na obrazovku dále z okna „Toolbox“ (sekce „Basic“) umístíme prvek **`NumericView`** (bez vazby na proměnnou). Uspořádání prvků a nastavení jejich velikostí provedeme dle následujícího obrázku.



Obr. 40 – Obrazovka s nastavením adresy

Zvolenou adresu, stejně jako v obrazovce „Srv_Protokol“ nebudeme ukládat do vlastnosti **`SerialBusN.Address`** ihned po její inkrementaci/dekrementaci, ale až při odchodu z obrazovky. Pro obsluhu stisku tlačítek **„+“** a **„-“** využijeme jejich události **`OnButtonPress`**. K výběru událostí, jednotlivých prvků na obrazovkách, se lze dostat vždy stiskem tlačítka ⚡ v okně „Vlastnosti“ zvoleného prvku. Pokud jsou v okně „Vlastnosti“ zobrazeny vlastnosti požadovaného prvku **`Button`**, zobrazí se po kliknutí na tlačítko ⚡ následující seznam událostí.



Obr. 41 – Události prvku Button

Po kliknutí levým tlačítkem myši do prázdného pole vedle události **`OnButtonPress`** se v prázdném poli zobrazí tlačítko …. Klikneme na tlačítko …, čímž dojde k otevření skriptovací části obrazovky a k vytvoření odpovídající události (**`OnButtonPress`**). V této události naprogramujeme pomocí skriptu inkrementaci adresy **AMR-OP70C** v závislosti na zvoleném komunikačním protokolu. Výsledný kód obsluhy stisku a držení tlačítka by pak vypadal následovně.

```
event Button2_OnButtonPress()  // obsluha tlačítka „+“
     If SerialBusN.ProtocolMode.0 then  // je zvolen komunikační protokol MODBUS
          If NumericView1.Value < 247 then
               NumericView1.Value = NumericView1.Value + 1;
          Else
               NumericView1.Value = 1;
          EndIf;
     Else  // je zvolen komunikační protokol ARION
          If NumericView1.Value < 63 then
               NumericView1.Value = NumericView1.Value + 1;
          Else
```

```
                NumericView1.Value = 1;
            EndIf;
        EndIf;
        NumericView1.Refresh();
end;
```

Obdobně pak naprogramujeme obsluhu události **OnButtonPress** tlačítka pro dekrementaci adresy **AMR-OP70C**. Kód obsluhy tlačítka pro dekrementaci bude vypadat následovně.

```
event Button3_OnButtonPress()  // obsluha tlačítka „-"
     If SerialBusN.ProtocolMode.0 then  // je zvolen komunikační protokol MODBUS
          If NumericView1.Value > 1 then
               NumericView1.Value = NumericView1.Value - 1;
          Else
               NumericView1.Value = 247;
          EndIf;
     Else  // je zvolen komunikační protokol ARION
          If NumericView1.Value > 1 then
               NumericView1.Value = NumericView1.Value - 1;
          Else
               NumericView1.Value = 63;
          EndIf;
     EndIf;
     NumericView1.Refresh();
end;
```

V dalším kroku naprogramujeme obsluhu vloženého tlačítka s textem „Zpět“. U tlačítka „Zpět“ využijeme událost **OnButtonDown**, ve které nejprve uložíme nastavenou hodnotu adresy do vlastnosti **Address** objektu **SerialBusN** a poté vložíme kód pro odchod na obrazovku „Srv_Menu“. Výsledný kód bude vypadat následovně.

```
event Button1_OnButtonDown()  // obsluha tlačítka „Zpět"
     SerialBusN.Address = NumericView1.Value;
     Srv_Menu.Show();
end;
```

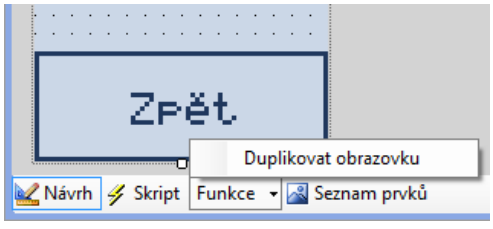
Závěrem je nutné doplnit do události **OnOpen** obrazovky skript, který zajistí, aby se po otevření této obrazovky zobrazila aktuálně nastavená adresa **AMR-OP70C**. Kamkoliv do události **OnOpen** obrazovky tedy doplníme následující skript.

```
NumericView1.Value = SerialBusN.Address;
NumericView1.Refresh();
```

5.1.3 Obrazovka pro nastavení komunikační rychlosti

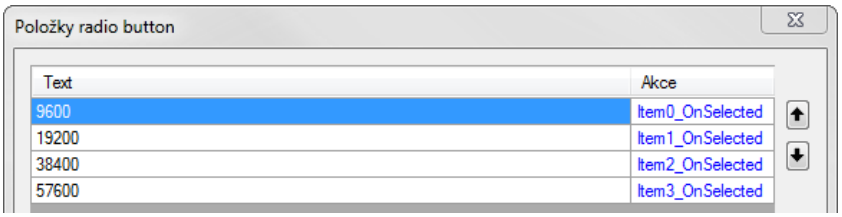
Pro nastavení rychlosti, s jakou bude **AMR-OP70C** v síti komunikovat, bude sloužit obrazovka „Srv_Rychlost“, kterou nadefinujeme duplikací již vytvořené obrazovky “Srv_protokol“.

Duplikaci obrazovky „Srv_Protokol“ provedeme pomocí tlačítka **Funkce/Duplikovat obrazovku** z nástrojové lišty obrazovky.



Obr. 42 – Funkce duplikace obrazovky

Po provedení duplikace upravíme prvek **Label** tak, aby zobrazoval text „Rychlost“. Položky prvku **RadioButton** upravíme (po dvojkliku na prvek) dle následujícího obrázku.



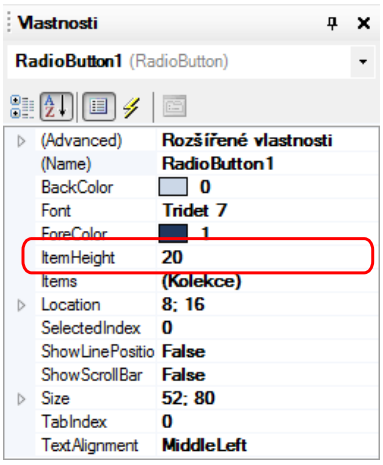
Obr. 43 – Nastavení položek prvku **RadioButton**

Po potvrzení nastavení tlačítkem *„**OK**“* dojde k otevření skriptovací části obrazovky, ve které budou vloženy události **RadioButton1_Item0_OnSelected**, **RadioButton1_Item1_OnSelected**, **RadioButton1_Item2_OnSelected** a **RadioButton1_Item3_OnSelected**. V kódu vytvářené aplikace je však nepoužijeme. Proto automaticky nadefinované události smažeme a vrátíme se zpět do návrhové části obrazovky.

Skript **Button1_OnButtonDown** prvku **Button** upravíme následovně.

```
event Button1_OnButtonDown()
     If  RadioButton1.SelectedIndex == 0 then
          SerialBusN.BaudRate = 9600;
     Else
          If  RadioButton1.SelectedIndex == 1 then
               SerialBusN.BaudRate = 19200;
          Else
               If  RadioButton1.SelectedIndex == 2 then
                    SerialBusN.BaudRate = 38400;
               Else
                    SerialBusN.BaudRate = 57600;
               EndIf;
          EndIf;
     EndIf;
     Srv_Menu.Show();
end;
```

Vlastnosti prvku **RadioButton** upravíme dle následujícího obrázku.



Obr. 44 – Nastavení vlastností prvku **RadioButton**

Výsledná obrazovka pro nastavení komunikační rychlosti pak bude vypadat následovně.

Obr. 45 – Obrazovka pro nastavení komunikační rychlosti

Skript v události **OnOpen** upravíme následovně.

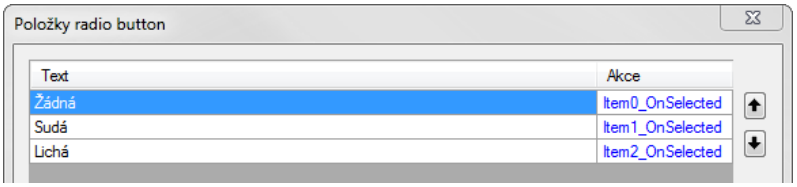
```
If SerialBusN.BaudRate == 9600 then
     RadioButton1.SelectedIndex = 0;
Else
     If SerialBusN.BaudRate == 19200 then
          RadioButton1.SelectedIndex = 1;
     Else
          If SerialBusN.BaudRate == 38400 then
               RadioButton1.SelectedIndex = 2;
          Else
               RadioButton1.SelectedIndex = 3;
          EndIf;
     EndIf;
EndIf;
```

5.1.4 Obrazovka pro nastavení parity

Pro nastavení parity, s jakou bude **AMR-OP70C** komunikovat v síti MODBUS RTU (v síti ARION nelze paritu měnit) bude sloužit obrazovka „Srv_Parita“, kterou vytvoříme duplikací již vytvořené obrazovky “Srv_protokol“.

Duplikaci obrazovky „Srv_Protokol“ provedeme stejně jako v předchozí kapitole pomocí tlačítka **Funkce/Duplikovat obrazovku** z nástrojové lišty obrazovky.

Po provedení duplikace upravíme prvek **Label** tak, aby zobrazoval text „Parita“. Položky prvku **RadioButton** upravíme (po dvojkliku na prvek) dle následujícího obrázku.



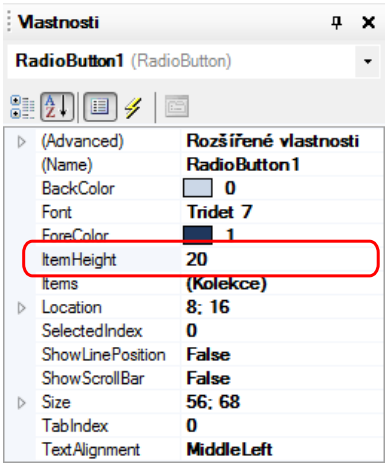
Obr. 46 – Nastavení položek prvku **RadioButton**

Potvrzením nadefinovaných položek tlačítkem *„**OK**“* se otevře skriptovací část obrazovky, ve které budou vloženy události **RadioButton1_Item0_OnSelected**, **RadioButton1_Item1_OnSelected** a **RadioButton1_Item2_OnSelected**. V kódu vytvářené aplikace je však nepoužijeme. Proto automaticky nadefinované události smažeme a vrátíme se zpět do návrhové části obrazovky.

Skript **Button1_OnButtonDown** prvku **Button** upravíme následovně.

```
event Button1_OnButtonDown()
     SerialBusN.ModbusParity = RadioButton1.SelectedIndex;
     Srv_Menu.Show();
end;
```

Vlastnosti prvku **RadioButton** upravíme dle následujícího obrázku.



Obr. 47 – Nastavení vlastností prvku **RadioButton**

Výsledná obrazovka pro nastavení komunikační rychlosti pak bude vypadat následovně.

Obr. 48 – Obrazovka pro nastavení parity

Paritu je však možné volit pouze pro komunikaci prostřednictvím protokolu MODBUS RTU. Na obrazovku tedy dále naprogramujeme zobrazení/skrytí informace s upozorněním na možnost nastavení parity pouze v případě použití protokolu MODBUS RTU. Jelikož bude upozornění na více řádků, využijeme prvek **MultilineLabel**. Po jeho umístění na obrazovku mu nastavíme text "Pouze pro Modbus." Do události **OnOpen** obrazovky „Srv_Parita" pak doplníme následující kód.

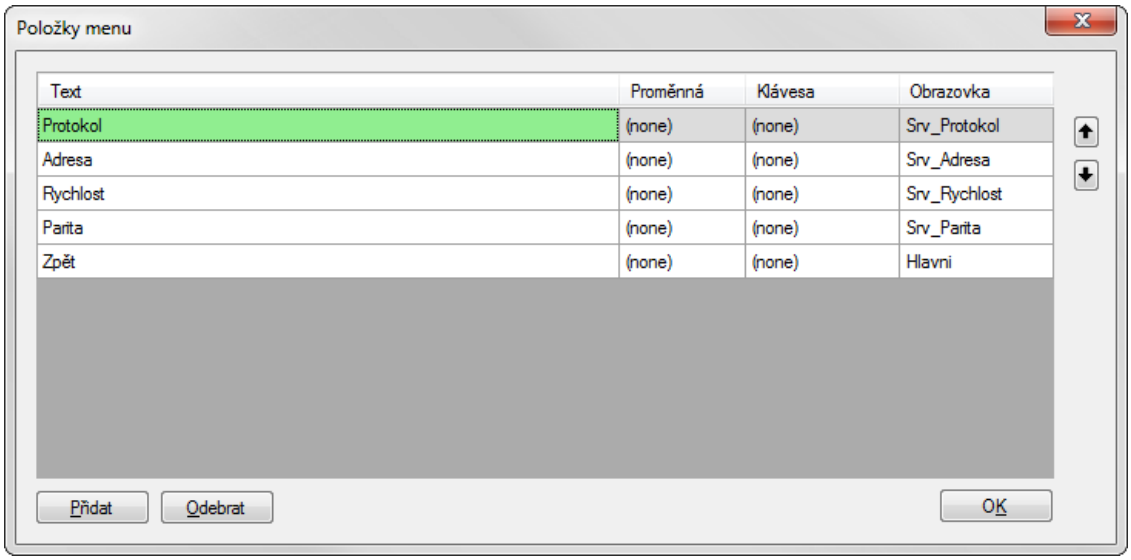
```
If SerialBusN.ProtocolMode.0 then  // pokud je nastaven protokol MODBUS
     MultilineLabel1.Visible = false;  // skrytí textu
     RadioButton1.Visible = true;  // zobrazení volby parity
     RadioButton1.Enabled = true;  // umožnění volby parity
     RadioButton1.SelectedIndex = SerialBusN.ModbusParity;  // zobrazení aktuálně
nast. parity
Else  // pokud je nastaven ARION
     MultilineLabel1.Visible = true;  // zobrazení textu
     RadioButton1.Visible = false;  // skrytí volby parity
     RadioButton1.Enabled = false;  // znemožnění volbu parity
EndIf;
Srv_Parita.Refresh();
```

Tím je zajištěno, že v případě, kdy je vybrána komunikace prostřednictvím protokolu ARION, bude na obrazovce zobrazen text dle následujícího obrázku. V opačném případě bude zobrazena možnost výběru parity (prvek **RadioButton** bude zobrazovat aktuálně nastavenou paritu).

Obr. 49 – Obrazovka s nastavením parity v případě vybraného protokolu ARION

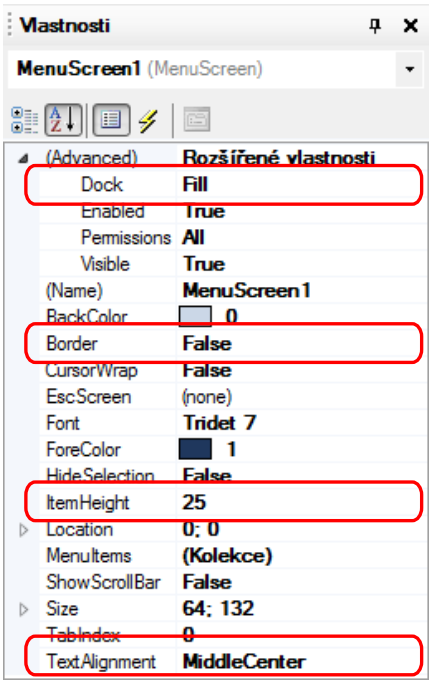
5.1.5 Obrazovka se servisním menu

Pro přechod na obrazovky s jednotlivými nastaveními bude sloužit vytvořená obrazovka „Srv_Menu“. Otevřeme ji a z okna „Toolbox“ (sekce „General“) do ní umístíme prvek **`MenuScreen`**. Po jeho umístění na obrazovku na něj dvakrát klikneme a v otevřeném okně nastavíme jednotlivé položky dle následujícího obrázku.



Obr. 50 – Nastavení položek prvku **`MenuScreen`**

Výsledný vzhled prvku **`MenuScreen`** upravíme v okně „Vlastnosti“ dle následujícího obrázku.



Obr. 51 – Nastavení vlastností prvku **`Menu`**

Po změně dle obrázku výše bude obrazovka „Srv_Menu“ vypadat následovně.

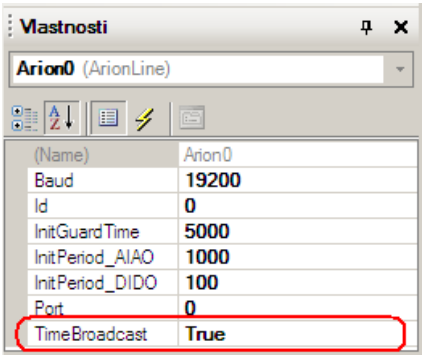
Obr. 52 – Obrazovka se servisním menu

5.2 Zobrazení času řídicího systému na AMR-OP70C

Pokud budeme požadovat zobrazení času řídicího systému na **AMR-OP70C**, je nutné toto naprogramovat jak na straně **AMR-OP70C** tak na straně řídicího systému.

5.2.1 Úprava aplikace pro řídicí systém (ARION).

V řídicím systému upravíme globální nastavení sítě ARION (pomocí okna „Vlastnosti“). V okně vlastností sítě Arion0 nastavíme vlastnosti **`TimeBroadcast`** hodnotu True. Tím dosáhneme rozesílání rámců s údajem o čase řídicího systému všem zařízením v síti ARION.



Obr. 53 – Nastavení rozesílání informace o čase

5.2.2 Úprava aplikace pro řídicí systém (MODBUS RTU)

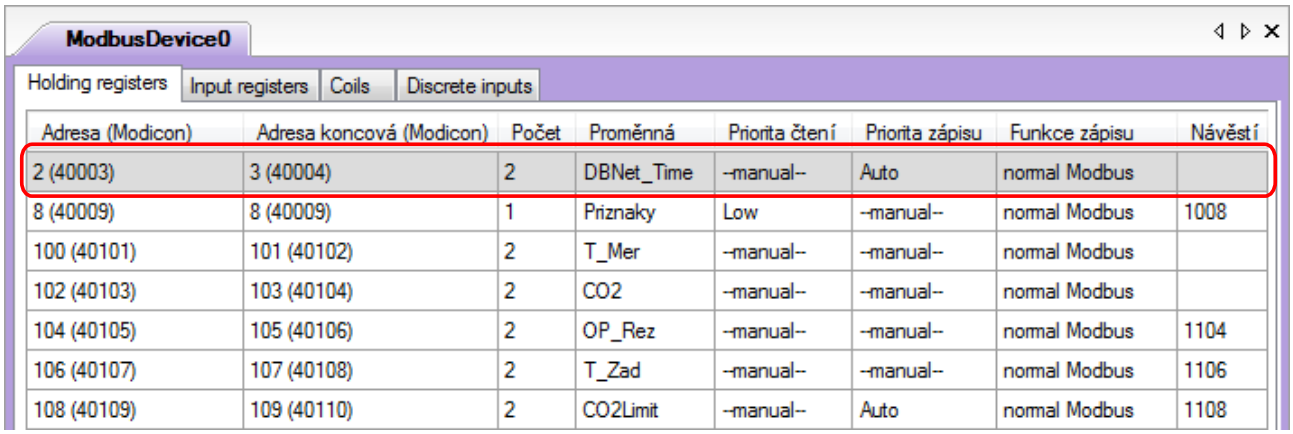
V řídicím systému je nutné načíst hodnotu času řídicího systému a nadefinovat zápis do systémových registrů 2 a 3 (registry Time) v ovladači.

Čas řídicího systému získáme pomocí modulu **`GetTime`**. Pro zaslání času do ovladače využijeme speciální (tzv. DB-Net) formát. Pro získání času tedy nadefinujeme proměnnou typu Long a použijeme ji v parametru **`Time`**, modulu **`GetTime`**.

Modul umístíme do procesu s periodou, odpovídající požadované periodě zápisu času do ovladače. V případě ukázkového projektu rs_p2_cz_xx.dso se jedná o „Proc01“ s periodou 5000 ms.

```
GetTime DBNet_Time, NONE, NONE
```

Zápis hodnoty proměnné **`DBNet_Time`** provedeme v tabulce „ModbusDevice0“. Jelikož není požadavek na čtení registru do řídicího systému, bude priorita čtení nastavena na „--manual--“. Prioritu zápisu nastavíme na „Auto“.



Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěstí
2 (40003)	3 (40004)	2	DBNet_Time	--manual--	Auto	normal Modbus	
8 (40009)	8 (40009)	1	Priznaky	Low	--manual--	normal Modbus	1008
100 (40101)	101 (40102)	2	T_Mer	--manual--	--manual--	normal Modbus	
102 (40103)	103 (40104)	2	CO2	--manual--	--manual--	normal Modbus	
104 (40105)	105 (40106)	2	OP_Rez	--manual--	--manual--	normal Modbus	1104
106 (40107)	107 (40108)	2	T_Zad	--manual--	--manual--	normal Modbus	1106
108 (40109)	109 (40110)	2	CO2Limit	--manual--	Auto	normal Modbus	1108

Obr. 54 – Definice zápisu do systémových registrů 2 a 3 (Time)

Poznámka
Výše uvedený postup zápisu času bude funkční pouze pro případ komunikace s jedním ovladačem. Při zvolené prioritě zápisu „Auto“ dojde po zapsání času do ovladače ke shození příznaku o požadavku na zápis. Pokud by tedy byl stejný řádek také v dalších tabulkách (jiných „ModbusDeviceX“), nikdy by v nich zápis nebyl proveden. První tabulka provede shození příznaku o požadavku na zápis a k dalším tabulkám se již požadavek nedostane. Řešením je použít prioritu zápisu „--manual--“ v kombinaci s modulem ***`MdbmWrite`****.*

5.2.3 Úprava aplikace pro AMR-OP70C

Na straně **AMR-OP70C** plně postačí umístit na obrazovku prvek **`DateTimeView`**, kterému nastavíme požadovaný formát zobrazení. V našem případě prvek umístíme na obrazovku „Hlavni“. Dvojklikem na prvek jej navážeme na vlastnost **`NowLong`** objektu **`DateTime`**. Ve výsledku pak bude obrazovka vypadat následovně.

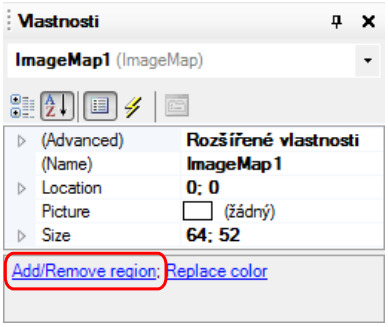
Obr. 55 – Obrazovka se zobrazeným časem

5.3 Přechod mezi obrazovkami

V případech kdy potřebujeme provést přechod mezi hlavní obrazovkou a např. obrazovkou s uživatelským nastavením korekce nebo přechod mezi hlavní obrazovkou a servisní obrazovkou s nastavením komunikačních parametrů, lze toto realizovat např. prostřednictvím prvku **`ImageMap`**. Přechod budeme realizovat tak, že pokud se uživatel pouze dotkne sekce, která je označena na níže uvedeném obrázku, zobrazí se např. obrazovka s nastavením korekce měřené teploty. Pokud bude dotek trvat cca 10 s, zobrazí se servisní menu.

Obr. 56 – Sekce pro přechod mezi obrazovkami

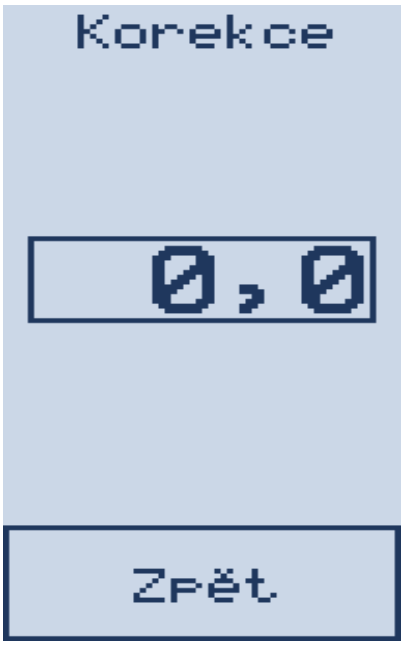
Do požadované sekce tedy umístíme prvek **`ImageMap`**, kterému upravíme velikost tak, aby ji měl stejnou jako vybraná sekce. Pomocí položky „Add/Remove region“ (v okně vlastností prvku) nadefinujeme jeden region, který zabere celou plochu prvku.



Obr. 57 – Vlastnosti prvku **`ImageMap`**

Pro přechod mezi obrazovkami v prvku **`ImageMap`** využijeme události **`OnRegionUP0`** (pro přechod na zadání korekce) a **`OnRegionPress0`** (pro přechod do servisního menu).

Vytvoříme novou obrazovku s názvem „Korekce“, na které budeme zadávat korekci měřené teploty (pomocí prvku **`NumericEdit`** na ní budeme provádět editaci proměnné **`Temp_SensorCorrection`** umístěné v paměti EEPROM). Vzhled obrazovky bude dle následujícího obrázku.



Obr. 58 – Obrazovka pro zadání korekce

Jak bylo uvedeno výše, pro přechod na tuto obrazovku použijeme událost **OnRegionUp0** regionu „Region0“ prvku **ImageMap**, který jsme umístili na obrazovku „Hlavní“. Událost bude vypadat následovně.

```
event ImageMap1_OnRegionUp0()
     Korekce.Show();
end;
```

Po přidržení regionu prvku na dobu cca 10 s budeme požadovat přechod na obrazovku se servisním menu („Srv_Menu“). Pro tuto funkčnost využijeme událost **OnRegionPress0** prvku **ImageMap** (stále se pracuje se stejným prvkem **ImageMap**, pouze je využita jeho druhá událost). Zároveň musíme nadefinovat proměnnou typu Integer (např. s názvem **Time**) pomocí které se bude zjišťovat počet, kolikrát byla vyvolána událost **OnRegionPress0**. Při stisku a držení region0 prvku **ImageMap** na obrazovce, se událost **OnRegionPress0** vyvolává s periodou cca 200 ms. Na obrazovku se servisním menu budeme přecházet až poté, co region0 prvku **ImageMap** budeme držet 10 s. Při držení region0 tedy budeme inkrementovat hodnotu proměnné **Time** tak dlouho, dokud nedosáhne hodnoty 50 (50 × 200 ms = 10 s). Jakmile proměnná **Time** dosáhne hodnoty 50, vynulujeme její hodnotu a přejdeme na obrazovku se servisním menu. Výsledný kód události **OnRegionPress0** pak bude vypadat následovně.

```
event ImageMap1_OnRegionPress0()
     Time = Time + 1;
     If Time > 50 then  // čekáme 10 sekund, než se držení projeví
          Time = 0;
          Srv_Menu.Show();
     EndIf;
end;
```

Nulování hodnoty proměnné **Time** je nutné doprogramovat i do události **OnRegionUp0** . Toto je pro případ, kdy uživatel tlačítko pustí předčasně, ještě před otevřením obrazovky se servisním menu.

```
event ImageMap1_OnRegionUp0()
     Korekce.Show();
     Time = 0;
end;
```

6 Dodatek A

6.1 Využití samostatných komunikačních objektů

Pokud není požadavek na přepínání komunikace mezi protokolem ARION a MODBUS RTU, je možné využít pro komunikaci danými protokoly také samostatné komunikační objekty:

- **ArionSlave,**
- **ModbusSlave**.

V obou případech se programování více či méně liší.

6.1.1 Objekt ArionSlave

Objekt **ArionSlave** se výrazně liší od objektu **SerialBusN** způsobem poskytnutí dat do sítě ARION. Objekt **SerialBusN** mapuje od sítě ARION tzv. registry (typu DInt či Real), do kterých lze přiřazovat hodnoty různých proměnných či vlastností objektů. Počet registrů, poskytnutých do sítě ARION, je v objektu **SerialBusN** omezen na 9.

Objekt **ArionSlave** poskytuje do sítě ARION:

- 24 signálů typu DI,
- 24 signálů typu DO,
- 24 signálů typu AI,
- 24 signálů typu AO.

Na straně řídicího systému je pak nutné zařízení s objektem **ArionSlave** nadefinovat v tabulce „Arion0“ (viz aplikační poznámka AP0025 – Komunikace v síti ARION (definice tabulkou)) pomocí modulu **ArionDevice** (v Toolboxu se nachází v sekci „DM“). Signály typu DI či DO se pak zpracovávají pomocí modulů **ARI_DigIn** či **ARI_DigOut**.

Analogové hodnoty jsou přenášeny jako celá 14 bitová čísla. V závislosti na znaménku lze přenést hodnotu v rozsahu 0 až 16383 (unipolar) nebo v rozsahu -8192 až 8191 (bipolar). Informaci o tom, zda je přenášena hodnota unipolární či bipolární je nutné nastavit řídicímu systému prostřednictvím tabulky „Arion0“ v definici modulu **ArionDevice**, pomocí vlastností **ModeAI** a **ModeAO**. Data se pak v celočíselné podobě zpracovávají pomocí modulů **ARI_NumAI** nebo **ARI_NumAO**.

Z výše uvedeného je zřejmé, že při požadavku na přenos proměnné typu Real z/do řídicího systému je nutné proměnnou přetypovat na celočíselnou hodnotu. Teprve poté je možné ji přenést prostřednictvím kanálu AI či AO.

Při požadavku na přenos hodnoty z analogového vstupu do řídicího systému nebo na přenos hodnoty ze strany řídicího systému na analogový výstup je nutné použít v zařízení **AMR-xxx** objekt **ArionIO**. Pokud se na straně **AMR-xxx** použije objekt **ArionIO**, zpracovávají se se data poskytnutá objektem **ArionIO** na straně řídicího systému pomocí modulů **ARI_AnIn** nebo **ARI_AnOut**.

Objekt **ArionSlave** lze s výhodou využít v případech, kdy je požadavek na přenos analogové hodnoty, která se přímo předává na analogové výstupy nebo je přímo čtena z analogového vstupu. V ostatních případech (přenos proměnných či vlastností objektů typu Real) je výhodnější použít objekt **SerialBusN**.

Více informací včetně příkladů lze nalézt v aplikační poznámce AP0054 – Komunikace AMREG s řídicími systémy AMiT (ARION).

6.1.2 Objekt ModbusSlave

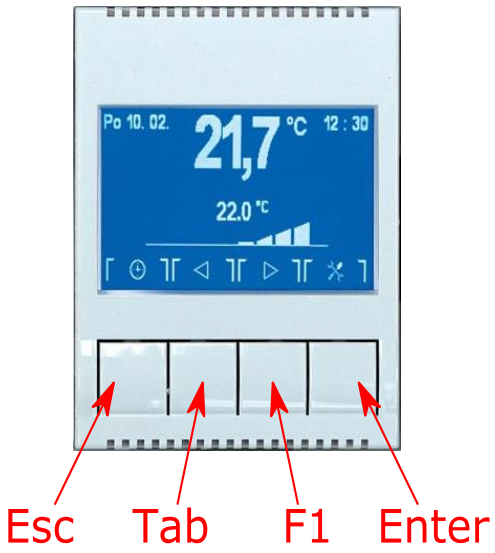
Objekt ModbusSlave lze použít podobným způsobem jako objekt **`SerialBusN`**. Na rozdíl od objektu SerialBusN umožňuje objekt ModbusSlave mapovat do sítě MODBUS RTU data nejen v podobě holding registrů, ale také v podobě input registrů, coilů či digitálních vstupů. Od DetStudia verze 2.1 probíhá mapování tak, že přímo v tabulce ModbusSlave lze vytvořit referenci vybraného MODBUS objektu na požadovanou vlastnost či proměnnou. Nebo vytvořit samostatný registr, který bude přímo používán v kódu aplikace.

Více informací včetně příkladů lze nalézt v aplikační poznámce AP0055 – Komunikace AMREG s řídicími systémy AMiT (MODBUS RTU).

7 Dodatek B

7.1 Obsluha tlačítek na AMR-OP60

Nástěnný ovladač umožňuje ovládání pomocí sady 4 tlačítek. Tlačítka jsou pro obrazovkové prvky mapována jako klávesy ***Esc***, ***Tab***, ***F1*** a ***Enter*** dle následujícího obrázku.



Obr. 59 – Význam tlačítek na **AMR-OP60**

Na obrazovkách lze tlačítka obsluhovat několika způsoby:

1. Využití prvků **`Keyxxx`** (výhodné pro změny režimů se signalizací zvoleného režimu).
2. Využitím prvku **`OP60kbd`** s předdefinovanými klávesami (vhodné pro zadávání požadovaných hodnot prostřednictvím prvků **`NumericEdit`** či **`NumericUpDn`**).
3. Využitím prvku **`OP60kbd`** s uživatelsky definovanými klávesami (vhodné pro použití u prvků se speciálním způsobem ovládání např. **`RadioButton`**).

Program s výše uvedenými způsoby obsluhy tlačítek je součástí této aplikační poznámky. Jedná se o soubor op60_p1_cz_xx.dsox.

7.1.1 Využití prvků Keyxxx

Využití prvků **`Keyxxx`** je výhodné zejména v případech, kdy je potřeba měnit režim zařízení a zároveň na obrazovce zobrazovat režim aktuálně nastavený. Do stejné skupiny patří také prvek **`KeyScreen`**, který lze využít pro přechod mezi obrazovkami.

V rámci příkladu op60_p1_cz_xx.dsox je nadefinována obrazovka „Hlavni“, na které jsou využity čtyři prvky **`Key`** (K_1, K_2, K_3 a K_4). Prvek s názvem „K_1“ slouží pro přepínání režimu místnosti, prvky „K_2“ a „K_3“ slouží pro korekci žádané teploty v místnosti a prvek „K_4“ slouží pro přepínání do uživatelského a servisního nastavení.

Obsluha prvku Key („K_1“)

Pomocí prvku „K_1“ bude možné měnit požadovaný režim místnosti (přepínáním režimů časový plán, komfort, útlum). Aktuálně nastavený režim bude zobrazen pomocí prvku **`CaseImage`** nad tlačítkem.

CaseImage („CI_RMode“)

Prvku **CaseImage** je nutné nadefinovat, pomocí vlastnosti **Images**, obrázky, odpovídající jednotlivým režimům. Za vlastnost **Variable** prvku **CaseImage** se dosadí proměnná **SerialBusN.Mist_Rez**.

Key („K_1“)

Obsluhu prvku je nutné řešit pomocí události „OnKeyDown“ ve skriptu.

```
event K_1_OnKeyDown()  // změna režimu místnosti
     if SerialBusN.Mist_Rez < 2 then
          SerialBusN.Mist_Rez = SerialBusN.Mist_Rez + 1;
     else
          SerialBusN.Mist_Rez = 0;
     endif;
     CI_RMode.Refresh();
end;
```

Obsluha prvků Key („K_2“ a „K_3“)

Pomocí prvků „K_2“ a „K_3“ bude možné měnit hodnotu (v rozsahu -100 až 100) korekce žádané teploty v místnosti. Aktuálně nastavená hodnota bude zobrazena pomocí prvku **CaseImage** nad tlačítky.

CaseImage („CI_Korr“)

Prvku **CaseImage** je nutné nadefinovat, pomocí vlastnosti **Images**, obrázky, odpovídající jednotlivým úrovním korekce. Za vlastnost **Variable** prvku **CaseImage** se dosadí proměnná **SerialBusN.Korekce**.

Key („K_2“ a „K_3“)

Obsluhu prvků je nutné řešit pomocí události „OnKeyPress“ ve skriptu. Zobrazení ikon nad tlačítky lze vyřešit pomocí prvku **Image**.

```
event K_2_OnKeyPress()  // korekce žádané teploty (snížení žádané teploty)
     if SerialBusN.Mist_Rez == 0 then  // změna korekce jen při režimu Časový plán
          if SerialBusN.Korekce > -100 then  // pokud je korekce větší než -100
               SerialBusN.Korekce = SerialBusN.Korekce - 20;  // odečte se hodnota
          else
               SerialBusN.Korekce = -100;
          endif;
          CI_Korr.Refresh();
     endif;
end;

event K_3_OnKeyPress()  // korekce žádané teploty (zvýšení žádané teploty)
     if SerialBusN.Mist_Rez == 0 then  // změna korekce jen při režimu Časový plán
          if SerialBusN.Korekce < 100 then  // pokud je hodnota menší než 100
               SerialBusN.Korekce = SerialBusN.Korekce + 20;  // přičte se hodnota
          else
               SerialBusN.Korekce = 100;
          endif;
          CI_Korr.Refresh();
     endif;
end;
```

Obsluha prvku Key („K_4“)

Pomocí prvku „K_4“ bude možné:

- zobrazit obrazovku s uživatelským nastavením korekce čidla (po krátkém stisku tlačítka),
- zobrazit servisní obrazovku s volbou komunikačního protokolu (po dlouhém stisku).

Zobrazení ikon nad tlačítkem lze vyřešit pomocí prvku **Image**.

Jelikož je požadavek na dvojí funkcionalitu tlačítka (krátký stisk/dlouhý stisk), nelze pro přechod mezi obrazovkami využít prvek **KeyScreen**. Požadovanou funkci je nutné naprogramovat pomocí prvku Key a jeho událostí „OnKeyPress“ (dlouhý stisk) a „OnKeyUp“ (krátký stisk). Událost „OnKeyDown“ není možné využít, protože se volá vždy před událostí „OnKeyPress“.

V projektu je nutné nadefinovat obrazovku pro uživatelské nastavení (např. s názvem „U_Korekce“) a obrazovku pro volbu komunikačního protokolu (např. s názvem „S_Protokol“).

Skript pro obsluhu obou událostí pak bude vypadat následovně.

```
// zobrazení uživatelského nastavení
event K_4_OnKeyUp()
     Time = 0;  // nulování čítače pro dlouhý stisk
     U_Korekce.Show();
end;

// zobrazení systémového nastavení
event K_4_OnKeyPress()
     Time = Time + 1;  // s každou událostí se inkrementuje hodnota (čítač)
     if Time > 50 then  // pokud čítač dosáhl požadované hodnoty
          Time = 0;  // čítač se vynuluje
          S_Protokol.Show();  // pokyn pro zobrazení systémového nastavení
     endif;
end;
```

7.1.2 Využití OP60kbd s předdefinovanými klávesami

Použití prvku **OP60kbd** s předdefinovanými klávesami lze v ukázkové aplikaci nalézt na obrazovce „U_Korekce“. Z výchozího nastavení prvku byl z vlastností **ImageButton2** a **ImageButton3** odstraněn obrázek, protože funkce kláves „Tab“ a „F1“ nejsou na obrazovce využity. Bližší popis funkce předdefinovaných kláves prvku **OP60kbd** lze nalézt v nápovědě obrazovek vývojového prostředí DetStudio.

Pro návrat na předchozí obrazovku se použije událost „OnButton1KeyDown“ prvku **OP60kbd**. Bude vypadat následovně.

```
event OP60kbd1_OnButton1KeyDown()
     Hlavni.Show();
end;
```

7.1.3 Využití OP60kbd s uživatelsky definovanými klávesami

Použití prvku **OP60kbd** s uživatelsky definovanými klávesami lze v ukázkové aplikaci nalézt na obrazovce „S_Protokol“. Na obrazovce je použit také prvek **RadioButton**. pomocí kterého je možné vybrat komunikační protokol (ARION/MODBUS) nástěnného ovladače. Prvek **RadioButton** se však ovládá pomocí kláves, které nejsou nabízeny prvkem **OP60kbd** ve výchozím nastavení. Klávesy, které nelze využít pro práci s prvkem **Radiobutton** je tedy nutné změnit na klávesy, které jsou vyžadovány pro ovládání prvku **RadioButton** (viz nápověda obrazovek vývojového prostředí DetStudio). Změnu kláves lze učinit pomocí vlastností **KeyCodeButtonX**. V rámci projektu je tedy nutné změnit klávesu „Tab“ na klávesu „Up“ a klávesu „F1“ na klávesu „Down“. Ve vlastnostech **ImageButton2** a **ImageButton3** se změní ikony kláves.

Při otevření obrazovky musí prvek **RadioButton** zobrazit aktuální nastavení. Pro naprogramování lze využít událost „OnOpen“ obrazovky „S_Protokol“.

```
event S_Protokol_OnOpen()
     S_Protokol.FocusFirstControl();
     RadioButton1.SelectedIndex = SerialBusN.ProtocolMode;  // načtení nastavení
     RadioButton1.Refresh();
end;
```

Pro odchod z obrazovky bez uložení se využije první tlačítko („Esc“). Obsluha jeho události „OnButton1KeyDown“ bude vypadat následovně.

```
event OP60kbd1_OnButton1KeyDown()
     Hlavni.Show();
end;
```

Pro uložení volby a odchod z obrazovky se využije čtvrté tlačítko („Enter“). Obsluha jeho události „OnButton4KeyDown“ bude vypadat následovně.

```
event OP60kbd1_OnButton4KeyDown()
     SerialBusN.ProtocolMode = RadioButton1.SelectedIndex;
     Hlavni.Show();
end;
```

8 Dodatek C

8.1 Práce s rozhraním Poseidon© na AMR-OP70RHP

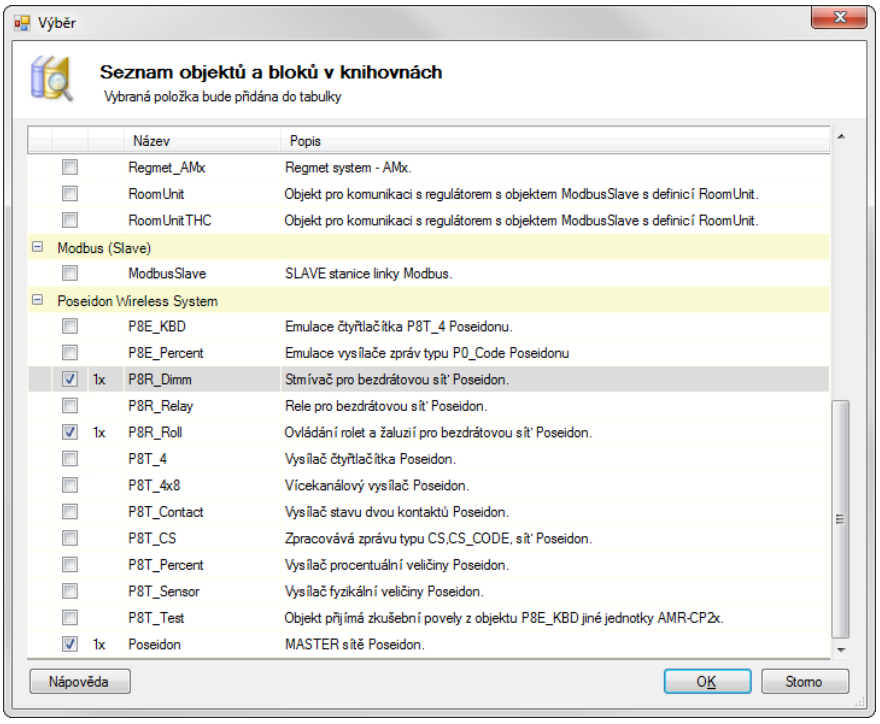
Nástěnný ovladač **AMR-OP70RHP** je mimo jiné osazen rozhraním Poseidon. Bližší informace o práci v síti Poseidon lze nalézt v nápovědě k části EsiDet, vývojového prostředí DetStudio a v aplikační poznámce AP0051 „Komunikace v bezdrátové síti Poseidon“.

Pozor
*Uvedeným příkladem lze řešit pouze nastavení základních vazeb. Pokud jsou vyžadovány složitější konfigurace (konfigurace sítě DALI, konfigurace časů pro otevření/zavření rolet či žaluzií apod.), je nutné využít již zmiňovaný SW nástroj Poseidon Asistent v kombinaci s hardware (**AMR-CP2x**, **AMR-CP4x**, **P8 TR IP**, **P8 TR USB**), který umožňuje konfiguraci sítě Poseidon.*

Následující příklad je k dispozici v příloze této aplikační poznámky. Jedná se o soubor s názvem op70rhp_p1_cz_xx.dsox.

8.1.1 Vytvoření obsluhy sítě Poseidon

Obsluhu sítě Poseidon lze do projektu **AMR-OP70RHP** vložit stejně, jako obsluhu jiných komunikačních protokolů – pomocí kontextového menu, vyvolaného nad složkou „Komunikace“ (v okně „Projekt“). V okně se seznamem komunikačních objektů (otevře se pomocí položky „Přidat objekt“ z kontextového menu) vybereme hlavní objekt **`Poseidon`** a periferie, se kterými má ovladač komunikovat.



Obr. 60 – Výběr objektů pro komunikaci v síti Poseidon

Důležité vlastnosti objektů P8x_xxx pro zprovoznění sítě jsou:
- ID – ID periferie v síti Poseidon, které je uvedeno na nálepce periferie,
- InitDevice – pokyn na vyslání či příjem speciálního rámce pro vytvoření vazeb,
- Error – hodnota případného chybového hlášení.

Vložením objektů pro obsluhu periferií sítě Poseidon do projektu jsou výše uvedené vlastnosti automaticky k dispozici u každého objektu pro komunikaci s periferií v síti Poseidon. Lze je tedy přímo použít na obrazovkách ovladače.

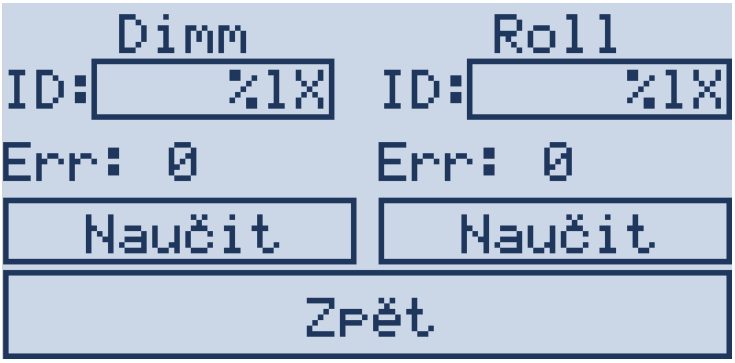
8.1.2 Obrazovka pro vytvoření vazeb v síti Poseidon

Při tvorbě servisní obrazovky pro vytvoření vazeb v síti Poseidon použijeme prvek **`NumericEdit`** pro případné zadání ID periferie, se kterou chceme provést vazbu. V prvku lze s výhodou využít vlastnost **`CustomFormat`** pro definici zobrazení hodnoty ID periferie v hexadecimálním tvaru (hodnota ID je na periferiích uvedena taktéž v hexadecimálním tvaru). Do vlastnosti **`CustomFormat`** tedy vložíme text „%IX“. Aby ovladač nabízel pro hexadecimální formát také znaky A ž F, nastavíme vlastnost **`SystemEditor`** na „NumericUpDn“.

Pokyn k vytvoření vazby mezi ovladačem a periferií v síti Poseidon lze naprogramovat pomocí prvku **`BitSwitchButton`**, kterému do vlastnosti **`Variable`** vložíme odkaz na vlastnost **`InitDevice`** odpovídajícího objektu definice sítě Poseidon. Ve vlastnostech **`TextDown`** a **`TextUp`** změníme texty na „Učím“ a „Naučit“. Při stisku tlačítka tak dojde k vyslání speciálního rámce pro vytvoření vazby a zároveň se zobrazí text „Učím“. Tento stav přetrvá buď do doby, než dojde k vytvoření vazby mezi ovladačem a periferií nebo do doby, než dojde k timeoutu.

Chybu při komunikaci lze zobrazit pomocí prvku **`NumericView`**, kterému do vlastnosti **`Variable`** vložíme odkaz na vlastnost **`Error`** odpovídajícího objektu definice sítě Poseidon.

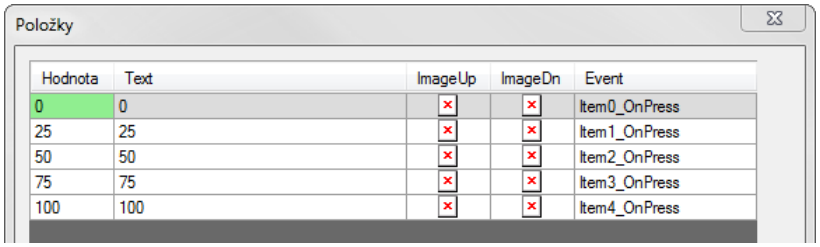
Výsledná obrazovka v pozici Landscape bude pro dva přijímače v síti Poseidon vypadat následovně.



Obr. 61 – Obrazovka pro vytvoření vazeb

8.1.3 Obrazovka pro nastavení požadované úrovně osvětlení

Při požadavku na nastavení úrovně osvětlení v přesně definovaných krocích lze využít prvek **`SelectButton`**. Po jeho vložení na obrazovku mu pomocí vlastnosti **`Items`** nastavíme jednotlivé položky dle následujícího obrázku.



Obr. 62 – Položky prvku SelectButton

Pro obsluhu osvětlení lze v síti Poseidon využívat např. objekt **P8R_Dimm** (analogové řízení úrovně osvětlení). Do vlastnosti **Variable** prvku **SelectButton** dosadíme vlastnost **OutX** objektu **P8R_Dimm**.

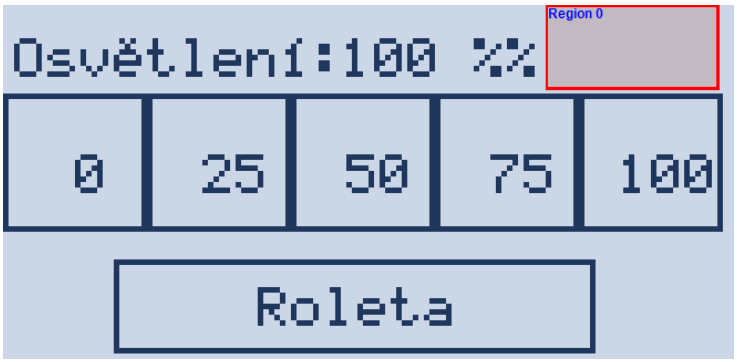
Dále na obrazovku vložíme prvek Label, kterému změníme text na „Osvětlení“.

Aktuálně nastavenou úroveň osvětlení lze u objektu **P8R_Dimm** načíst pomocí vlastnosti **ActualOutX**. Zobrazení nastavené úrovně provedeme pomocí prvku **NumericView**, kterému do vlastnosti **Variable** vložíme odkaz na vlastnost **ActualOutX**.

Na obrazovku naprogramujeme také přechod na obrazovku s parametry pro vytvoření vazeb. Pro přechod využijeme stejný postup jako v kapitole 5.3 „Přechod mezi obrazovkami“.

Pro přechod na obrazovku s ovládáním další periferie v síti Poseidon použijeme prvek **ButtonScreen**, kterému za jeho vlastnost **GoToScreen** dosadíme obrazovku, na kterou se má přejít a za jeho vlastnost **Text** dosadíme např. text „Roleta“ (přechod na ovládání rolet).

Výsledná obrazovka bude vypadat následovně.



Obr. 63 – Obrazovka pro nastavení žádané úrovně osvětlení

8.1.4 Obrazovka pro nastavení požadované polohy rolet

Obrazovku vytvoříme duplikací obrazovky pro nastavení požadované úrovně osvětlení. Na obrazovce pak změníme texty prvků a odkazy na jednotlivé vlastnosti.

Text prvku **Label** změníme na „Pozice“.

Prvku **SelectButton** dosadíme za vlastnost **Variable** odkaz na vlastnost **Position** objektu **P8R_Roll**.

Aktuálně nastavenou pozici rolety lze u objektu **P8R_Roll** načíst pomocí vlastnosti **ActualPosition**. Prvku **NumericView** tedy dosadíme za vlastnost **Variable** odkaz na vlastnost **ActualPosition**.

Pro návrat na obrazovku s ovládáním osvětlení dosadíme za vlastnost **GoToScreen** na odkaz na obrazovku s ovládáním osvětlení. Za jeho vlastnost **Text** dosadíme např. text „Světlo“ (přechod na ovládání osvětlení).

Výsledná obrazovka bude vypadat následovně.

Obr. 64 – Obrazovka pro nastavení žádané pozice rolet

9 Technická podpora

Veškeré informace ohledně programování nástěnných ovladačů Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím emailu na adrese **support@amit.cz**.

10 Upozornění

AMiT, spol. s r.o. poskytuje informace v tomto dokumentu, tak jak jsou, nepřejímá žádné záruky, pokud se týče obsahu tohoto dokumentu a vyhrazuje si právo měnit obsah dokumentu bez závazku tyto změny oznámit jakékoli osobě či organizaci.

Tento dokument může být kopírován a rozšiřován za následujících podmínek:

1. Celý text musí být kopírován bez úprav a se zahrnutím všech stránek.
2. Všechny kopie musí obsahovat označení autorského práva společnosti AMiT, spol. s r.o. a veškerá další upozornění v dokumentu uvedená.
3. Tento dokument nesmí být distribuován za účelem dosažení zisku.

   V publikaci použité názvy produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.