Písomný výstup pedagogického klubu

1. Prioritná os	Vzdelávanie
2. Špecifický cieľ	1.1.1 Zvýšiť inkluzívnosť a rovnaký prístup ku kvalitnému vzdelávaniu a zlepšiť výsledky a kompetencie detí a žiakov
3. Prijímateľ	Stredná priemyselná škola informačných technológií, Nábrežná 1325, Kysucké Nové Mesto
4. Názov projektu	Učme efektívnejšie pre prax
5. Kód projektu ITMS2014+	312011AMJ5
6. Názov pedagogického klubu	Informatika v praxi
7. Meno koordinátora pedagogického klubu	Ing. Peter Remiš
8. Školský polrok	február 2022 – jún 2022
9. Odkaz na webové sídlo zverejnenia písomného výstupu	www.spsknm.sk

10.

Úvod:

Stručná anotácia

Výmena vedomostí a skúseností medzi členmi pedagogického klubu, prehlbovanie vedomostí z oblasti IoT a informatiky obecne, zlepšenie kompetencií a vytvorenie učebného materiálu (príloha).

Kľúčové slová

Internet vecí, IoT, robotické rameno, robot, Fanuc, programovanie, PLC, Xiaomi ECO systém, smart house, Node-RED, server, ESP32, Arduino, Raspberry Pi, softvér, hardvér, sieť, protokol, služba

Zámer a priblíženie témy písomného výstupu

Zámerom písomného výstupu je tvorba odborných materiálov, ktoré budú slúžiť pre výučbu predmetu Internet veci a iných príbuzných predmetov oblasti IoT. Tieto materiály sú vytvorené v dokumente označenom ako príloha. Témy, ktoré sú spracované v dokumente sú: robotika a robotické ramená,

PLC systém, systém pre inteligentnú domácnosť, serverové služby pre spracovanie a prezentovanie dát. V závere dokumentu sa nachádza praktický príklad na oblasť IoT, ktorý demonštruje zlúčenie jednotlivých vrstiev architektúry Internetu vecí. Aby sa čitateľom/žiakom jednotlivé témy čo najviac priblížili k praxi, zvolil sa v každej téme v praxi bežne využívaný systém. Tým je napríklad Fanuc, PLC S7 1200, Xiaomi ECO systém atď.

Druhotným zámerom je priblíženie činnosti pedagogického klubu počas roka a zhrnutie získaných záverov.

Jadro:

Popis témy/problém

Pedagogický klub sa v tomto polroku zaoberal dvoma oblasťami. Jednu tvorila technická oblasť problematiky Internetu vecí a druhú tvorila pedagogická oblasť – prístup ku žiakom. Príloha, ktorú členovia klubu tvorili priebežne počas celej doby spadá do prvej oblasti. Tvorí učebnú pomôcku – učebný materiál, ktorý opisuje reálne technológie používajúce sa praxi. Konkrétne sa v prílohe opisujú nasledovné problematiky. Prvá kapitola opisuje robotické ramena a konkrétne sa zameriava na rameno Fanuc. Ide o robotické rameno českej firmy, takže u nás je veľmi využívané. Robotické ramená tvoria prvú vrstvu IoT – vytvárajú akcie a snímajú koncové produkty. Preto je nutné žiakom priblížiť technológiu, s ktorou sa s veľkou pravdepodobnosťou stretnú po skončení školy. Druhá kapitola prílohy sa zamerala na riadiaci systém PLC od firmy Siemens (konkrétne model S7-1200). PLC systémy sú vďaka svojej licencii a odolnosti veľmi využívané pre riadenie zariadení. Práve PLC od Siemensu sa v našej oblasti využíva najčastejšie, keďže ide o kvalitného nemeckého výrobcu. Tretia kapitola sa zamerala na Xiaomi ECO systém. Ide o systém pre správu inteligentnej domácnosti. Firma Xiaomi je čínska firma, ktorá sa vďaka pomeru ceny a kvality svojich výrobkov dostala v posledných rokoch medzi celosvetovú špičku. Množstvo typov ich produktov dokáže pokryť komplexne celú domácnosť. Predposledná kapitola sa venuje technológii Nore-RED. Táto technológia slúži na jednoduché grafické (blokové / uzlové) programovanie aplikácii na spracovanie dát a ich prezentovanie užívateľovi. Beží ako serverová služba a ponúka množstvo blokov ako napr. Facebook správy, Instagram notifikácie a pod. Po doinštalovaní nových blokov dokáže dokonca spolupracovať aj so zariadeniami ako je napr. Arduino. Nástroj Node-RED je veľmi obľúbený v IoT aplikáciách. Posledná kapitola obsahuje komplexný projekt na tému IoT – Internet vecí. Projekt je zameraný na ovládanie koncových zariadení (viacfarebných svetiel) pomocou internetovej stránky – webového rozhrania. Jadrom projektu je využitie MQTT servera pre prenos a uschovanie dát. Uvedené témy pokrývajú 50% z obsahu pedagogických klubov. Členovia, ktorí sú odborníci na dané oblasti, spracovali podklady a vyučili ostatných členov. Ďalších 50% sa zameralo na to, ako zvládnuť

výučbu žiakov. Učitelia spracovali problematiku a prakticky predviedli na stretnutiach ped. klubu, ako zlepšiť prístup ku žiakom. Prvou z „pedagogických tém“ bola téma zameraná na žiakov s poruchami učenia. Danú problematiku je nutné zvládať nie len preto, že čoraz viac žiakov trpí tzv. „dis-„ poruchami, ale aj preto, lebo i bežným žiakom takýto špeciálny prístup dokáže zlepšiť učenia schopnosť. Druhou tému boli organizačné formy učenia. Kolega poukázal, ako rozmanito a s akými výhodami sa dá realizovať vyučovací proces. Kolegyňa zas spracovala problematiku motivácie žiakov. Táto téma síce môže pôsobiť bežne, ale stále sa vyvíja, nakoľko sa mení aj doba, v ktorej žijeme. Deti majú už odmalička mobily a tie im dokážu poskytnúť takmer čokoľvek, takže je ich oveľa ťažšie namotivovať, ako tomu bolo v minulosti. Bez motivácie učiteľ nedokáže žiaka takmer nič naučiť, či už ide o vnútornú alebo vonkajšiu. Posledná téma sa zamerala na dištančné vzdelávanie. Kolega poukázal na rôzne nástroje a spôsoby výučby. Pred pár rokmi išlo o ojedinelú formu výučby, no kvôli pandémii sa stala témou číslo jeden v školstve. Preto bolo nutné zosumarizovať aktuálne poznatky a trendy. Kolega sa následne zameral na Moodle systém, ktorý bol pre túto formu výučby navrhnutý už dávno pred pandémiou.

Pedagogický klub teda naplnil všetky ciele, ktoré si na začiatku stanovil a síce obohatil členov o poznatky z oblasti IoT v praxi a o poznatky z oblastí pedagogiky, ktoré sú v tejto dobe pre učiteľa najaktuálnejšie.

Záver:

Zhrnutia a odporúčania pre činnosť pedagogických zamestnancov

Odporúčaním z pedagogického klubu je zamerať sa vo výučbe na praktické veci – prepojenie učiva s praxou a teda technológiami, ktoré sa v praxi využívajú. Taktiež je ale dôležité nezabúdať na to byť dobrým pedagógom a nie len technickým odborníkom. Bez pedagogického prístupu sa žiakom totiž odborné vedomosti nepodarí odovzdať. Preto sa pedagogický klub zameral okrem poukázania na technológie z praxe i na témy z pedagogiky.

Vďaka pedagogickému klubu si zvýšili učitelia svoje odborné vedomosti zo širokej oblasti. Zároveň sa vytvorili učebné materiály, ktoré slúžia pre výučbu v predmetoch s podobným zameraním, ako je oblasť IoT. Tým sa zmodernizovala výučba, ktorá napomôže k zvýšeniu kvality žiakov a ich uplatneniu v praxi. Obsah študijného materiálu tvorili všetci pedagogický členovia klubu a vychádzali pritom z aktuálneho stavu praxe, no zároveň volili nástroje, ktoré sú pre žiakov oveľa jednoduchšie na výučbu a voľne dostupné.

11. Vypracoval (meno, priezvisko)	Ing. Peter Remiš
12. Dátum	20.06.2022
13. Podpis	[redacted]
14. Schválil (meno, priezvisko)	Ing. Milan Valek
15. Dátum	11-07-2022
16. Podpis	[redacted]

Stredná priemyselná škola informačných technológií
Nábrežná 1325, 024 01 Kysucké Nové Mesto

INFORMATIKA V PRAXI

Autor: Ing. Peter Remiš február 2022 – jún 2022

OBSAH

SLUŽBY A SYSTÉMY V IOT

Pojem IoT pokrýva širokú oblasť. Od koncových zariadení slúžiacich na zber dát a ovládanie zariadení, cez prenos dát cez sieť, až po služby bežiace na cloude či užívateľských zariadeniach. Táto kapitola sa bude zaoberať jednotlivými oblasťami, s ktorými sa človek/žiak dokáže stretnúť v svojom okolí.

Ako koncové zariadenie sa v prvej kapitole opíše robotické rameno Fanuc. Toto rameno slúži ako akčný prvok v automatizačných linkách, ktorý dokáže pohybovať prvkami, alebo spínať rôzne zariadenia – zváračka, pumpa, lakovač atď. Okrem vykonávania akcie dokáže aj snímať veci v okolí, a preto sa dá považovať za ideálne koncové zariadenie.

V druhej kapitole sa opíše riadiaca jednotka PLC. Podobne ako Fanuc, i tento riadiaci systém je bežne nasadzovaný v praxi. Slúži na ovládanie zariadení jednoduchšou formou programovania. Oproti Arduinu má výhodu najmä v certifikácii a dlhodobej spoľahlivosti. PLC dokáže ovládať koncové zariadenia a následne cez sieť informovať centrálnu jednotku o ich stave.

Tretia kapitola je venovaná Xiaomi Eco systému, čo je systém na inteligentnú správu domácnosti. Xiaomi je veľmi rozšírená značka, ktorá je známa kvalitou, širokospektrálnosťou a prijateľnou cenou. Inteligentné domácnosti sú jednoznačné neoddeliteľnou súčasťou IoT.

Štvrtá kapitola sa venuje serverovej službe Node-RED. Ide o niečo ako programovací grafický jazyk slúžiaci na spracovanie dát a ich následnú prezentáciu užívateľom. Svojou jednoduchosťou je vhodný pre výučbu cloudových služieb. Hoci je grafické programovanie jednoduché, rozhodne nie je veľmi limitované a ponúka bloky pre správu zariadení ako je Arduino, no i internetových služieb ako je Instagram, či Twiter. V IoT ide o veľmi populárny nástroj tvorby cloudových aplikácii.

Posledná kapitola sa venuje praktickému príkladu IoT. Ide o demonštráciu všetkých oblastí: od ovládania systému užívateľom, cez sieťové IoT služby (MQTT), až po ovládanie koncového zariadenia. Príklad sa zameria na riadiaci systém ESP32, zobrazovací prvok NEOPIEXEL, protokol MQTT, web rozhrnie atď.

1 ROBOTICKÉ RAMENO FANUC

Priemyselný robot (LR Mate 200iD-4S) je univerzálny pohybový viacosový manipulátor, ktorý má voľne programovateľný spôsob pohybu. Tieto roboty môžu byť vybavené chápadlami, nástrojmi alebo inými výrobnými prostriedkami a môžu prevádzať manipulačné, technologické alebo montážne úlohy.

1.1 Rozdelenie priemyselných robotov

Priemyselné roboty môžeme všeobecne nazvať ako manipulátory. Dnes je najčastejšie používané rozdelenie do troch hlavných skupín:

- Ručné manipulačné zariadenia uvedené do prevádzky pomocou operátora, označované aj ako teleoperátory, slúžia k uľahčeniu opakovaných operácií s ťažkými bremenami. Ich využitie je často jednoúčelové, ale sú aj viacúčelové manipulátory tejto skupiny.

- Robot s pevným programom je manipulátor, ktorý vykonáva činnosť na základe pevne daného programu, bez priameho zásahu človeka. Zmena pracovného cyklu je veľmi obtiažna a často sa neobíde bez zásahu v podobe výmen a úprav konštrukčných prvkov.

- Robot s premenným programom a ľahkou zmenou pracovného cyklu. Je to voľne programovateľný manipulátor, poskytujúci vysoký stupeň univerzálnosti.

1.2 Základné časti priemyselných robotov

Robotické rameno – je to mechanická časť robota, skladá sa z kĺbov a väzieb, pričom kĺby slúžia k realizácii pohybu robota a väzby tvoria tuhé telesa medzi nimi. Každý kĺb poskytuje stupeň voľnosti. Väčšina robotov má 5 resp. 6 stupňov voľnosti. Mechanická časť robota sa skladá z podstavy, karuselu a ramien.

Riadiaca jednotka – riadiaca časť robota. Vysiela príkazy pre ovládanie pohonov a ostatných mechanizmov podľa zadaného programu. Je schopná spracovávať vstupné signály od senzorického systému, na základe ktorých je schopná vykonávaný algoritmus meniť a upravovať ho.

Programovacia časť robota (programovacia jednotka, Teach pendant) – umožňuje prístup medzi človekom a robotom. Vďaka nej môžeme vizuálne kontrolovať parametre a ostatné dôležité informácie o zariadení, takisto umožňuje ručne riadiť robota alebo ho programovať.

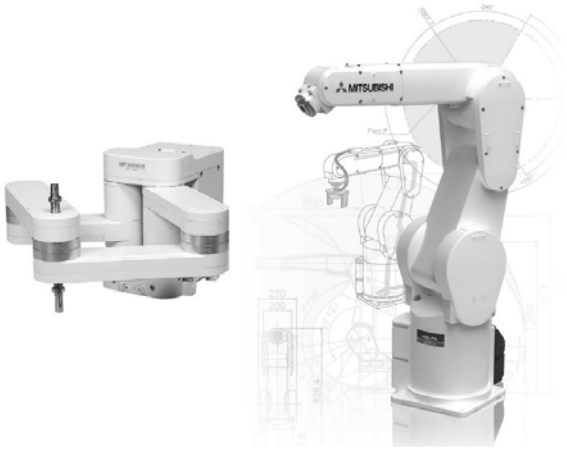
Zostava priemyselného robota

- Manipulátor – robotické rameno
- Riadenie robota – riadiaca jednotka
- Programovacia jednotka – Teach Pendant
- Spojovacie vedenia
- Softvér
- Voliteľné príslušenstvo

1.3 Základné delenie robotov

Roboty môžu byť klasifikované podľa rôznych kritérií. Ako príklad uvedieme rozdelenie podľa:

- kinematickej štruktúry,
- počtu stupňov voľnosti,
- spôsobu riadenia, programovania,
- bezpečnosti robotov a iné.



Jedno zo základných delení je práve delenie podľa kinematickej štruktúry. Podľa nej delíme priemyselné roboty do nasledovných skupín.

Sériové roboty

Sériové roboty - sériová kinematika robota ja charakteristická otvoreným kinematickým reťazcom, v ktorom sú jednotlivé kinematické dvojice usporiadané za sebou. Pohyb koncového člena kinematického reťazca je výsledkom na seba nadväzujúcich pohybov jednotlivých kinematických členov. Pričom jednotlivé členy sa môžu pohybovať nezávislé na sebe. Patria tu:

- Angulárne kĺbové roboty – najpoužívanejšie priemyselné roboty
- SCARA roboty
- Kartézske roboty
- Cylindrické roboty
- Sférické roboty

Paralelné roboty

Paralelná kinematika je špecifická uzatvoreným kinematickým reťazcom. Takýto mechanizmus sa skladá zo základne (bázy) prepojenej s pohyblivou platformu prostredníctvom minimálne dvoch paralelných členov (ramien). Pohyblivá platforma

môže mať od troch do šiestich stupňov voľnosti. Jej pohyb je daný súčasným pohybom všetkých ramien. Patria tu:

- Delta roboty
- Tripody
- Hexapody

1.4 Súradnicové systémy priemyselného robota

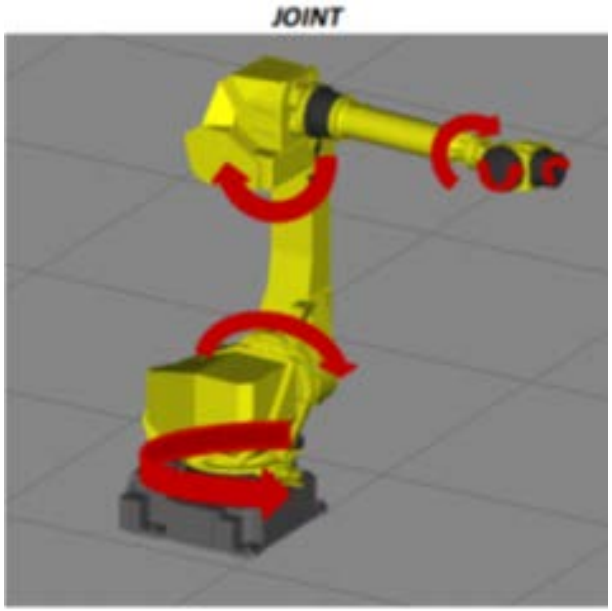
Robot môže pracovať v niekoľkých súradnicových systémoch, vďaka ktorým je schopný presne určovať svoju polohu. Informácie o polohe je potrebné poznať z dôvodu jeho riadenia. Bez súradných systémov by nebol schopný polohovať svoj koncový bod.

U priemyselných robotov používame štyri základné súradnicové systémy:

- Systém JOINT
- Systém WORLD/JGFRM
- Systém TOOL
- Systém USER

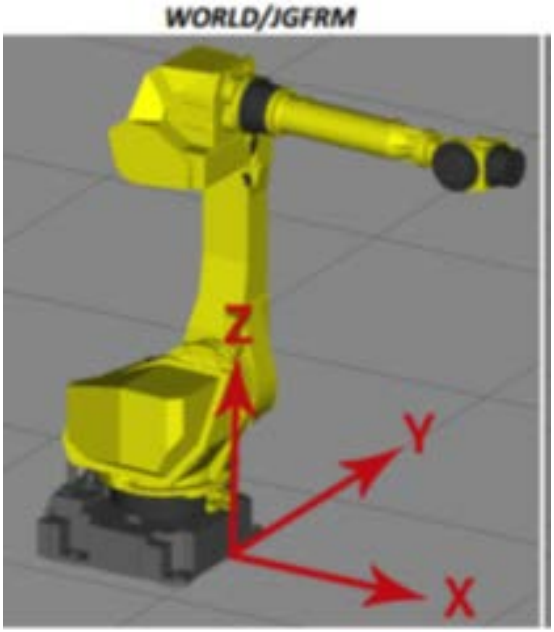
Súradnicový systém JOINT

- Jedná sa o osový súradnicový systém. Tento systém je nezameniteľne votknutý v každej osi robota. Poloha je potom vyjadrovaná pomocou uhlov natočení jednotlivých kĺbov.



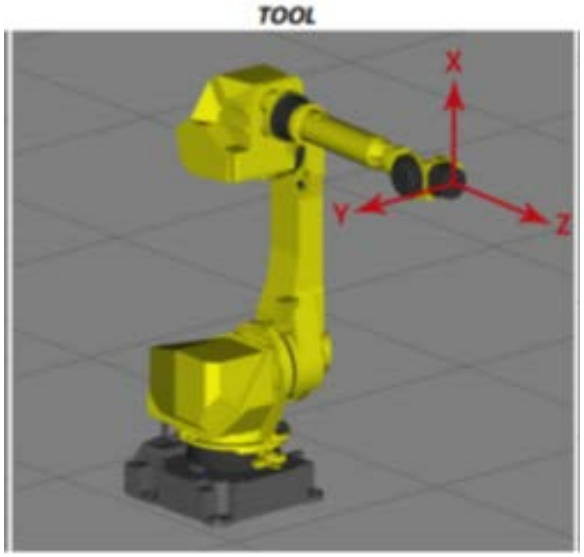
Súradnicový systém WORLD/JGFRM

- Jedná sa o svetový súradnicový systém. Je nezamenitelne definovaný kartézsky súradný systém votknutý do päty robota. Robot sa pohybuje v smere osi X, Y a Z.



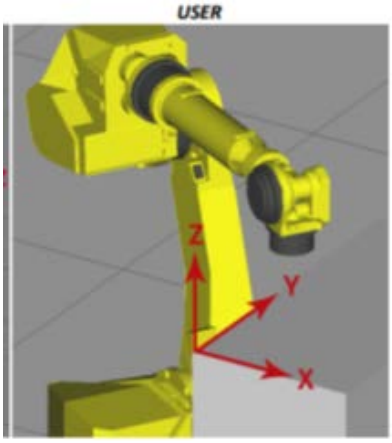
Súradnicový systém TOOL

- Systém TOOL, je súradný systém nástroja umiestnený na strede zápästia robota. Je definovaný užívateľom. Definuje polohu stredového bodu nástroja, skrátene TCP z anglického Tool Center Point. Stredový bod nástroja je nutný k ďalšiemu určovaniu údajov o polohe. Ak nie je systém TOOL definovaný, je ako platný systém koncového nástroja braný systém zápästia robota.



Súradnicový systém USER

- Jedná sa o užívateľský súradnicový systém. Tento súradný systém je definovaný užívateľom. Po nadefinovaní je robot a jeho polohy späté so začiatkom tohto systému. Začiatok je tiež často označovaný ako origin.



1.5 Druhy pohybov robota

- Dosiahnutie jednotlivých polôh koncového bodu robot realizuje pomocou pohybov. U priemyselného robota rozlišujeme tri základné pohyby:

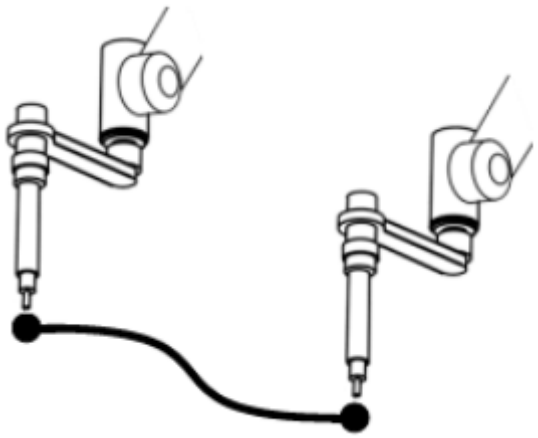
- Pohyb JOINT

- Pohyb LINEAR

- Pohyb po kružnici

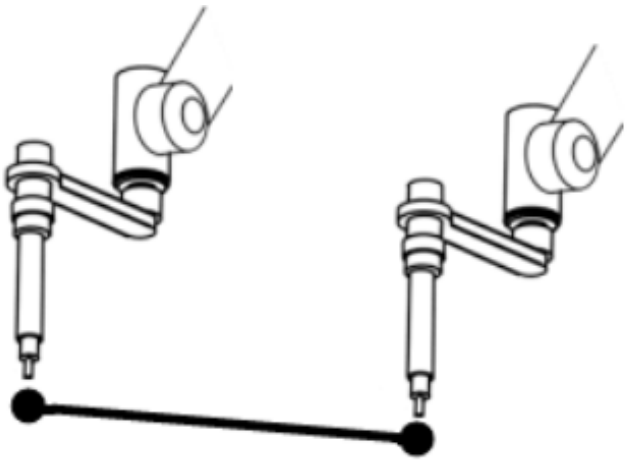
Pohyb JOINT

- Pohyb Joint, je zadávaný pomocou dvoch bodov, východiskového a cieľového. Robot do definovanej cieľovej polohy príde po ľubovolnej trajektórii v najkratšom čase. Pri tomto pohybe môže byť správanie robota v istých situáciách nepredvídateľné.



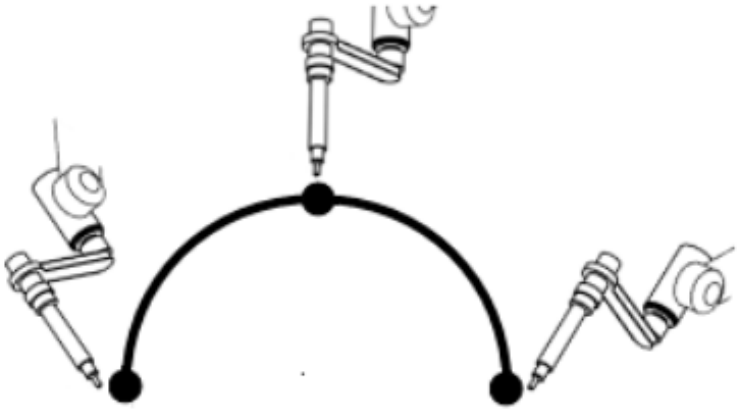
Pohyb LINEAR

- Robot do koncovej polohy príde po priamke definovanou rýchlosťou. Takisto je zadávaný pomocou východiskového a cieľového bodu. Tento pohyb je vhodný, ak potrebujme presnejšie spoznať, ako sa bude robot pohybovať priestorom.



Pohyb po kružnici

- Robot do definovanej koncovej polohy príde po kružnici. Na rozdiel
- od predchádzajúcich pohybov nemožno pohyb po kružnici definovať iba východiskovým a koncovým bodom, ale aj tretím medzibodom, cez ktorý TCP (nástroj robota) prechádza. Na základe tohto bodu je dopočítavaný polomer kruhového pohybu.



Syntax pohybovej inštrukcie robota

Pohyb robota sa vykoná po vložení príslušnej pohybovej inštrukcie. Syntax pohybovej inštrukcie po jej vložení vyzerá nasledovne:

L P[1] 100mm/s FINE

- L Druh pohybovej inštrukcie
- P [1] Pozičné údaje - číslo pozície
- 100mm/s Rychlosť pohybu -mm/s (default)-ďalšie možnosti sú cm/min, inch/min, deg/sec, sec, msec
- FINE Spôsob dosiahnutia daného bodu (priame, s obídením s určitým polomerom)

1.6 Druhy pohybových inštrukcií

- **Inštrukcia JOINT** - robot sa pohybuje z bodu do bodu bez stanovenia dráhy (najrýchlejšia dráha).

  J (P1) 100% FINE

- **Inštrukcia LINEAR** - robot sa pohybuje z východzieho do cieľového bodu programovanou rýchlosťou po priamke (lineárne).

  L (P1) 4000mm/sec FINE

- **Inštrukcia CIRCULAR** - robot sa pohybuje po kružnici z východzieho do cieľového bodu stanovenou rýchlosťou cez pomocný bod (max.180°).

  J P [1] 100% FINE

  C P [2] (medzibod)

  P [3] 2000 mm/s FINE

- **Inštrukcia ARC Circle** znamená pohyb robota po kruhovom oblúku. Efektívnejšie sa programuje ako klasický pohyb CIRCULAR. Na tento pohyb sú potrebné aspoň tri body.

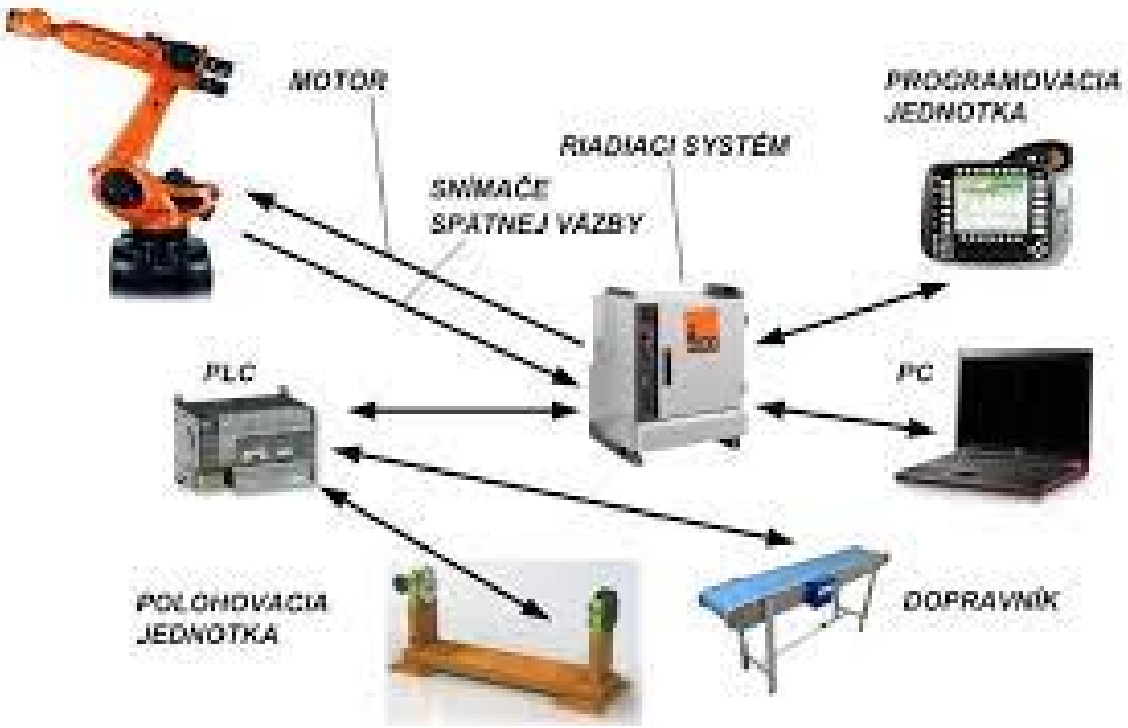
  A (P1) 4000mm/sec FINE

1.7 Vstupy a výstupy robota

- U priemyselných robotov FANUC máme nasledovné druhy vstupov a výstupov:
- Digitálne vstupy/výstupy – DI/DO
- Vstupy a výstupy robota – RI/RO
- Skupinové vstupy a výstupy – GI/GO
- Systémové vstupy a výstupy – SI/SO

Digitálne vstupy a výstupy

- Slúžia pre spoluprácu priemyselného robota s okolitými zariadeniami, napr. so snímačmi, PLC automatmi príp. na spoluprácu s inými robotmi.
- Označenie digitálnych vstupov/výstupov: DI/DO.

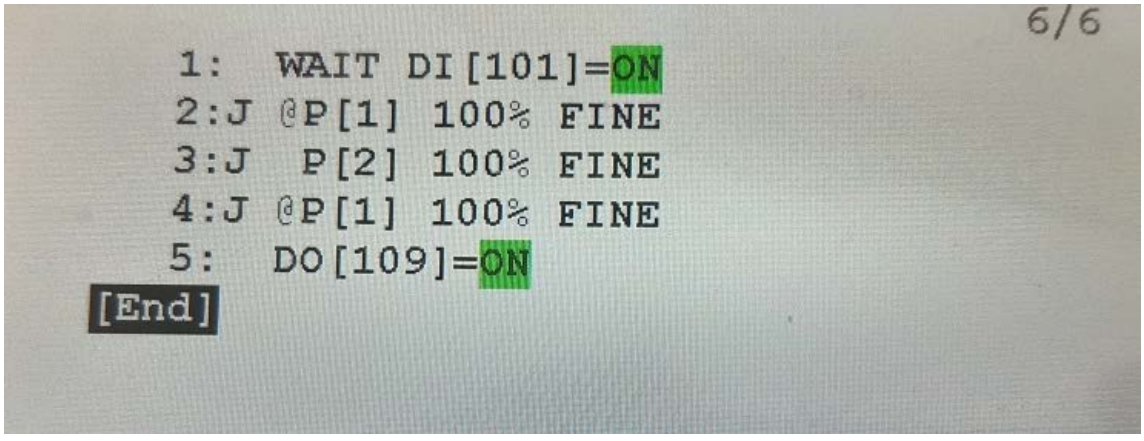


Digitálne vstupy a výstupy – príkazy

- Druhy príkazov pre digitálny vstup/výstup robota DI/DO.
- DO[x] = ON / OFF: Tento príkaz dá určitý digitálny výstupný signál na ON alebo na OFF.
- DO[x] PULSE [hodnota] : Týmto príkazom je digitálny výstupný signál na stanovenú dobu zopnutý na ON. DĺŽka impulzu(PULSE) je v sekundách (0,1-25s).
- WAIT DI[x]= ON: Týmto príkazom sa v spracovaniu programu čaká na signál ON.
- IF DI[x]= .....
- IF DO[x]= .....
- IF DO[x]=(.....)

Príklad programu s využitím riadenia digitálnych vstupov/výstupov

- Program zapísaný v programovacej jednotke i-Pendant:



- Popis činnosti programu: Robot čaká, až bude aktívny digitálny vstup DI (101). Ten sa aktivuje signálom z externého zariadenia (snímača, PLC automatu). Až po jeho aktivácii začne robot vykonávať vlastný program, tvorený pohybovými inštrukciami. Po ich vykonaní aktivuje príslušný digitálny výstup (DO (109)). Tým vyšle signál do externého zariadenia (napr. PLC automatu, signalizačného zariadenia a pod.).

2 PLC SIEMENS S7 -1200

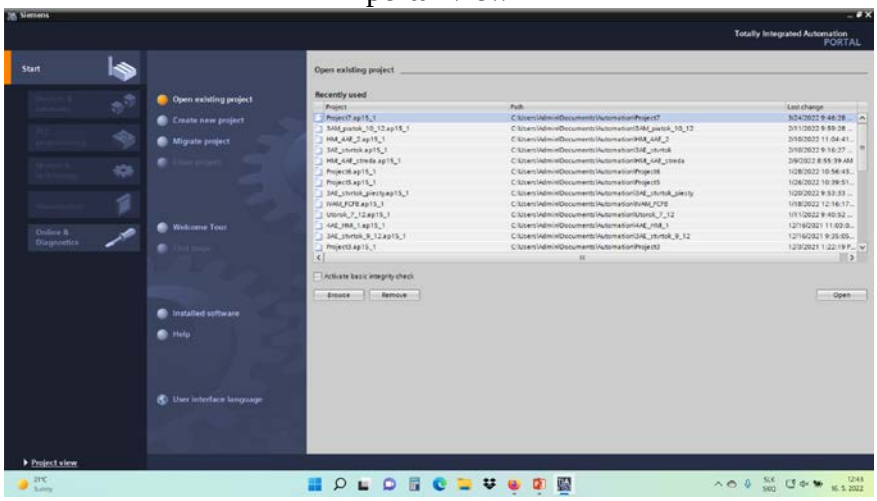
Pojem PLC znamená Programable logic controller. Ide o riadiaci systém pre priemyselné využitie. Určený je na riadenie a komunikáciu rôznych zariadení v priemyselných podmienkach (systémy bez vlastného riadenia, systémy rôznych výrobcov...)

- Modulárny – prispôsobiteľný (rozšíriteľný systém)
- Komunikácia cez Ethernet (ProfiNet) – s PC, HMI, ďalšími PLC...
- Rozširujúce I/O moduly
- Riadiaca jednotka
- Komunikačné moduly

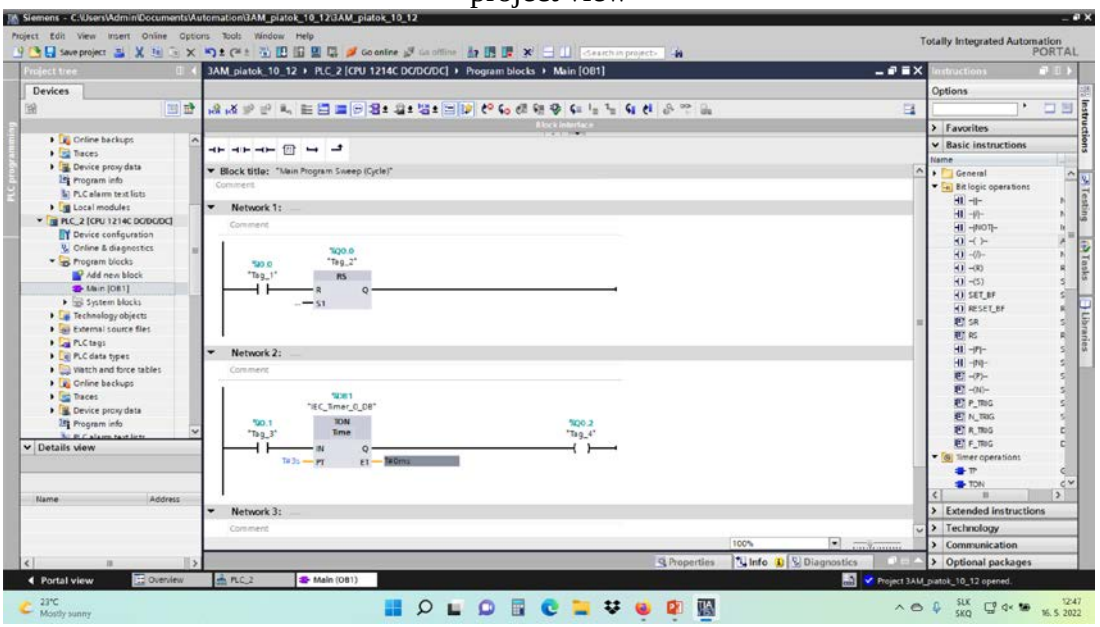
2.1 Programové prostredie TIA Portal

- Spoločné prostredie pre programovanie všetkých Siemens PLC zariadení, dotykových obrazoviek a I/O zariadení rôznych výrobcov.
- Obsahuje aj simulačný program PLC SIM.

portal view

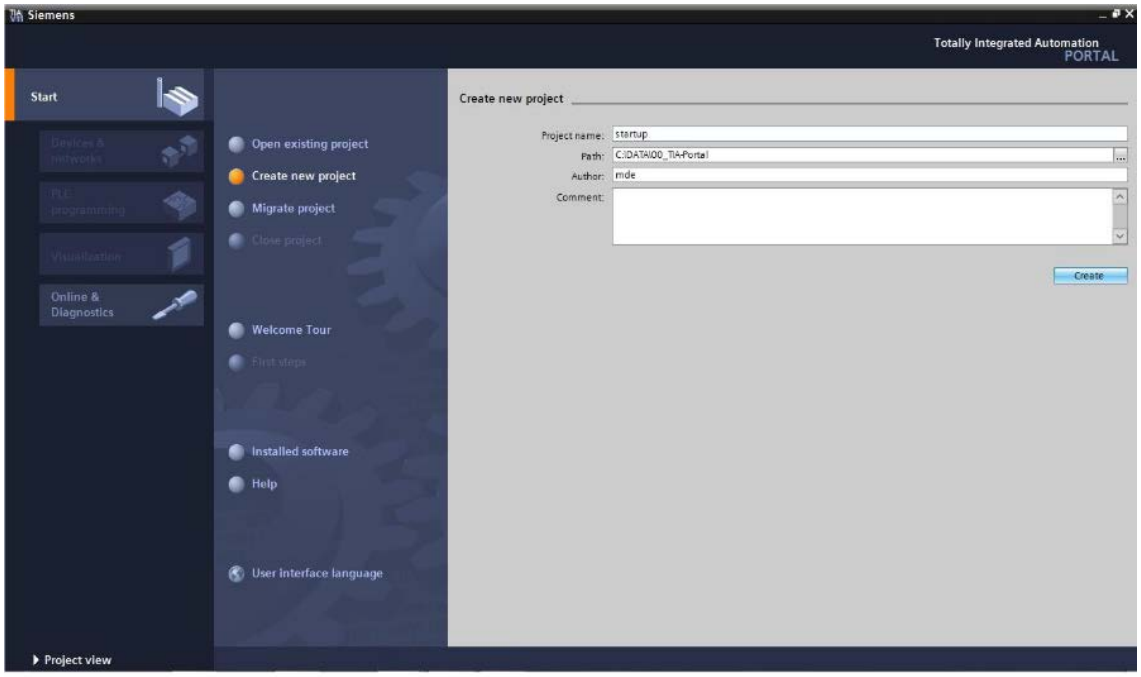


project view



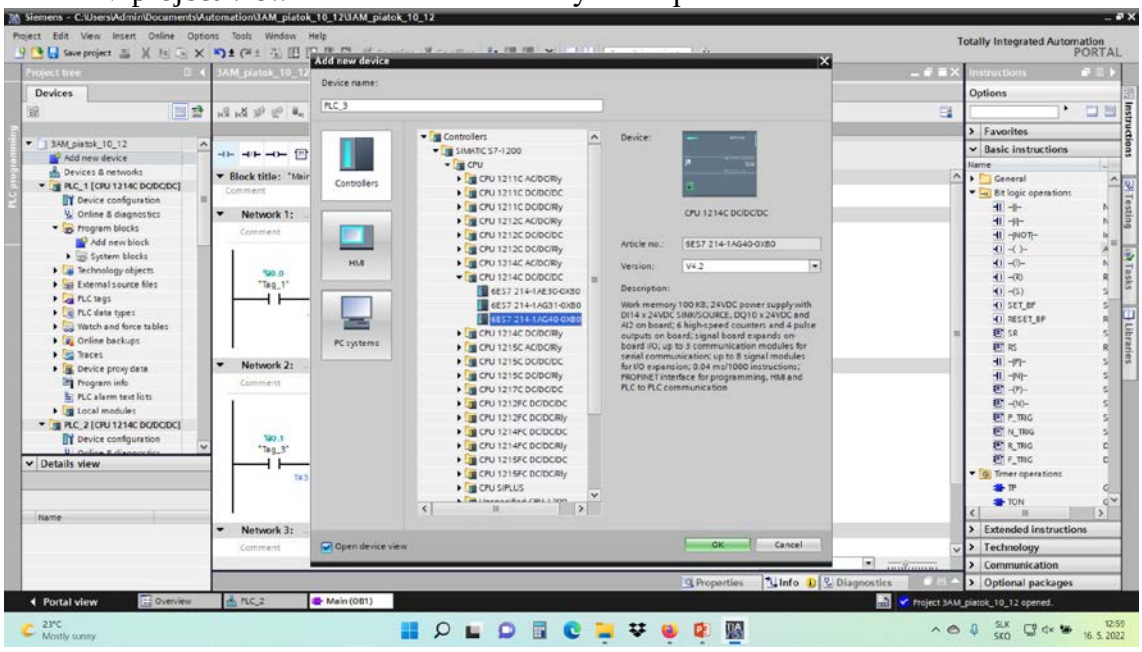
2.2 Inštalácia

Programy pre SIMATIC S7-1200 sú spravované v projektoch. Nový projekt. (- Create new project - startup - Create)



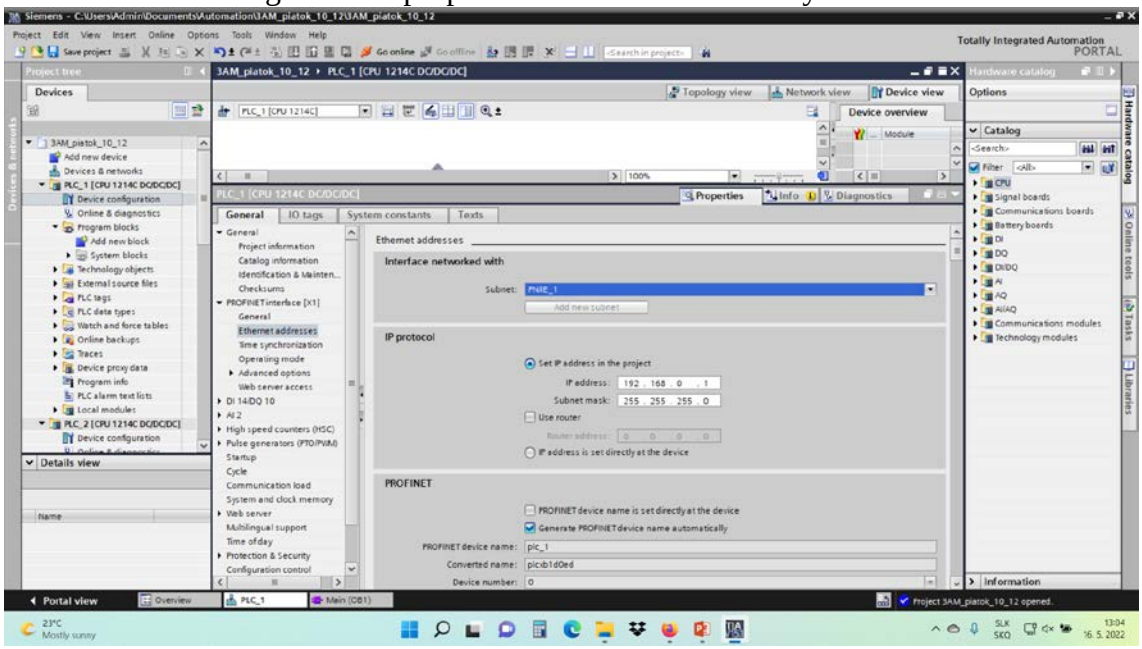
Vloženie CPU do projektu.

- V project view - Add new device. Vybrať a potvrdiť.



Základná konfigurácia.

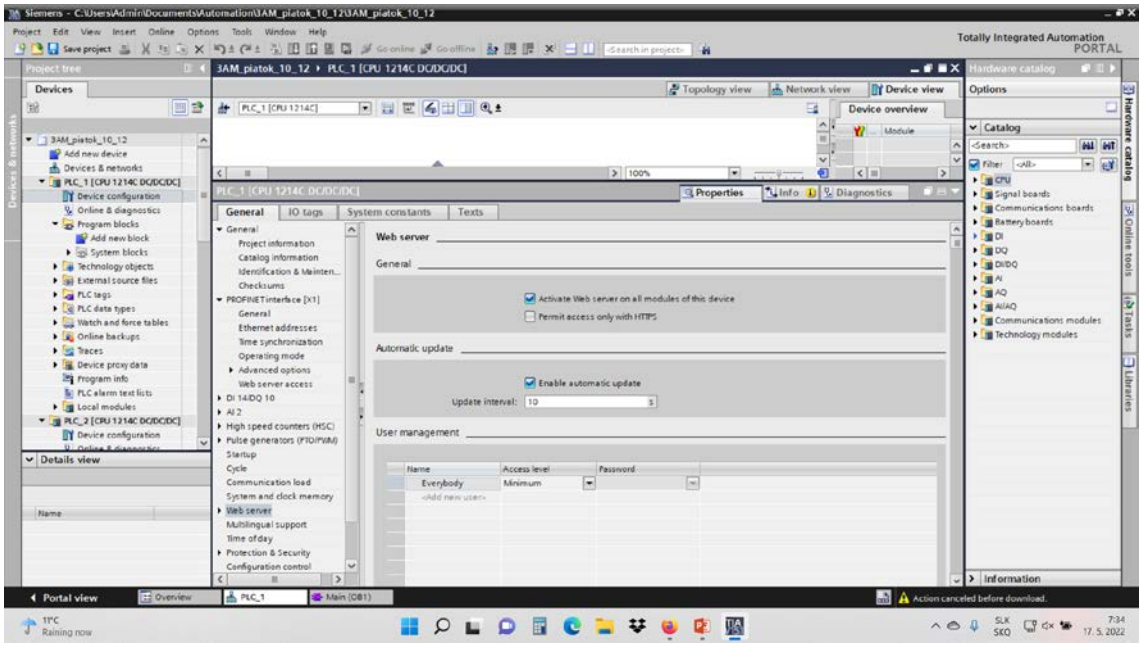
- Device configuration – properties. Nastavenie IP adresy...



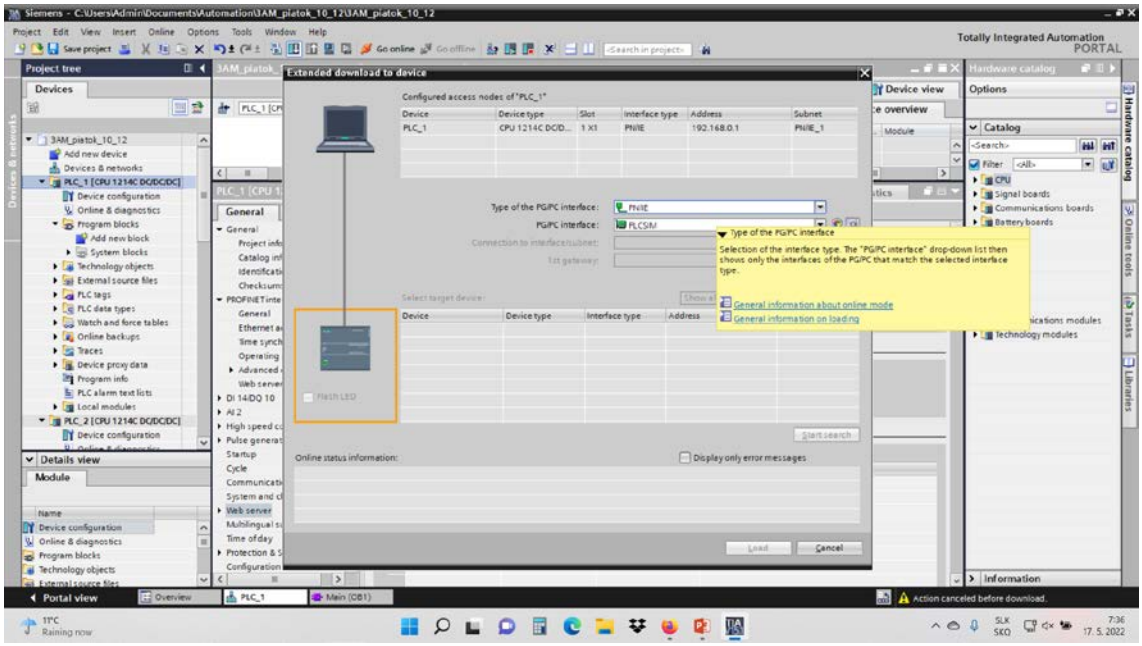
2.3 Úloha – ovládanie LED cez web rozhranie.

Postup:

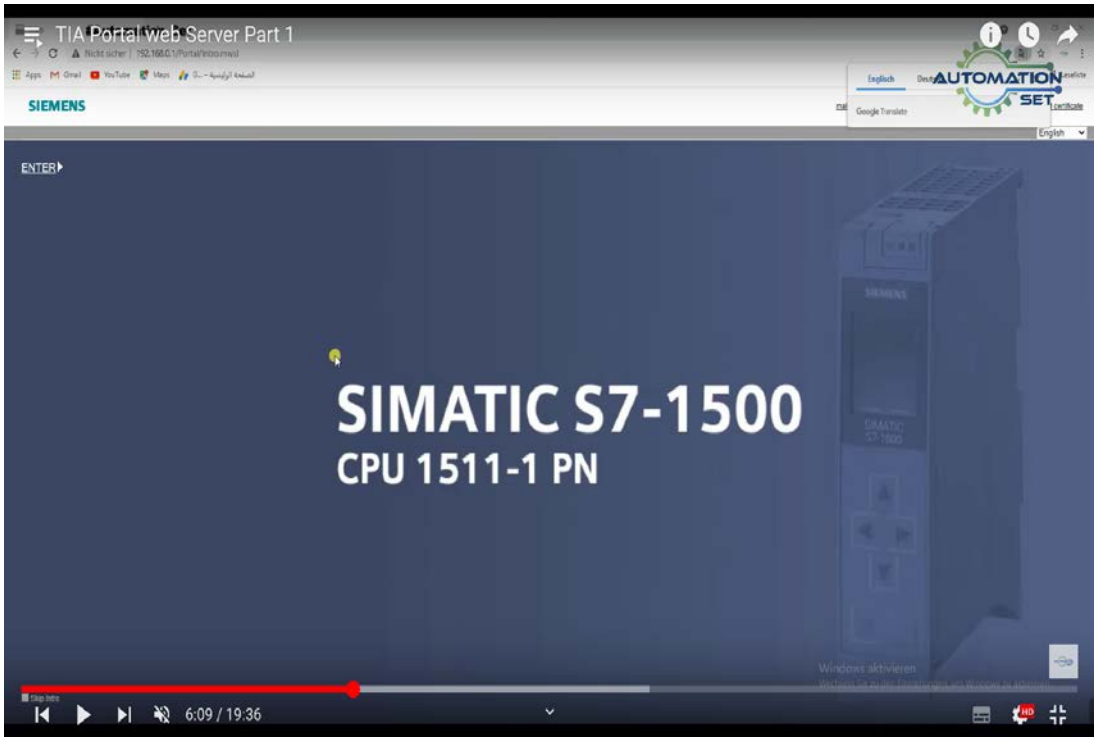
- **povolenie web servera v TIA Portal** – device configuration – properties – web server – Activate web server on all modules of this device

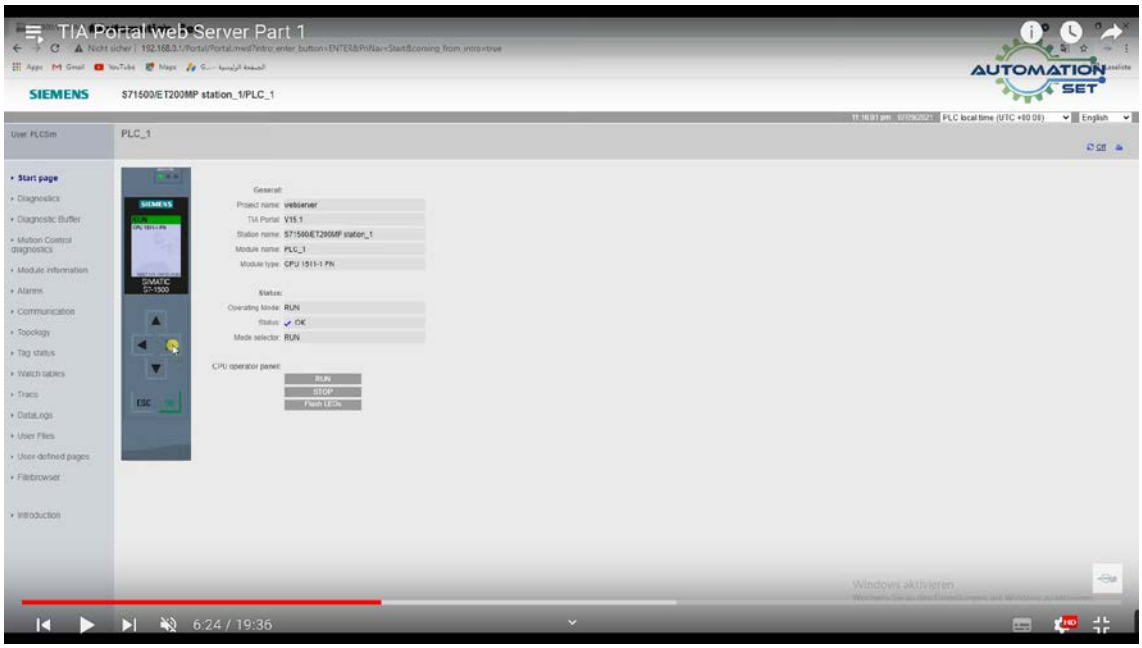


- **nahranie hardvérovej konfigurácie do PLC** – download to device – start search – load – load – finish



- **test spojenia cez prehliadač –** v prehliadači zadáme IP adresu PLC (192.168.0.1)
- **cez ENTER sprístupníme základné informácie o PLC**

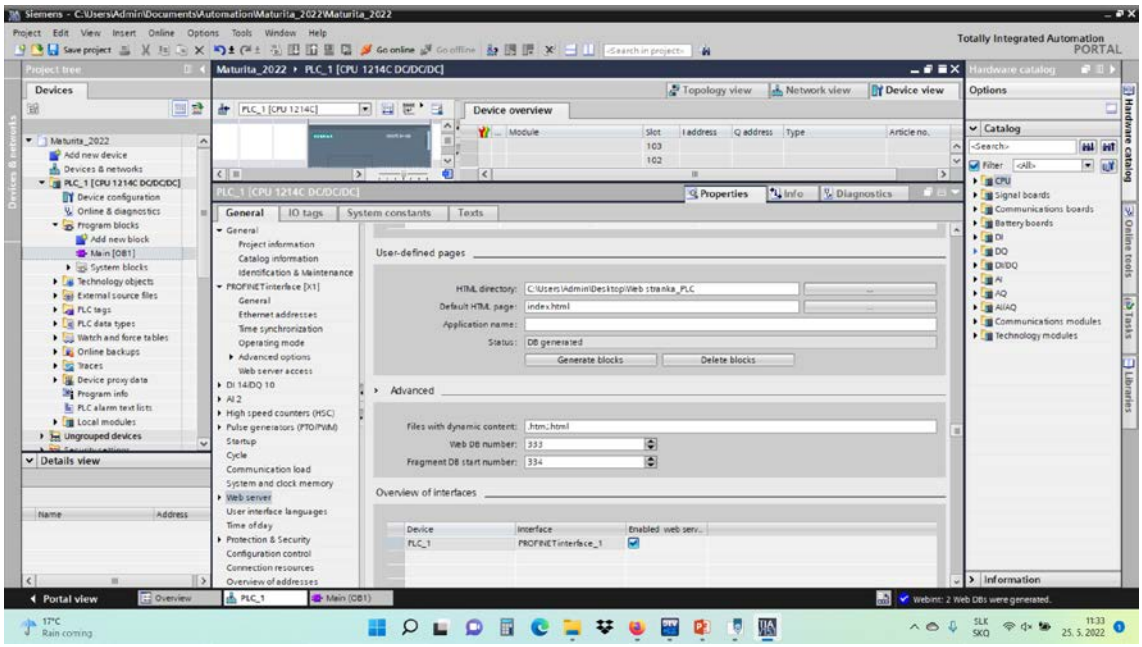




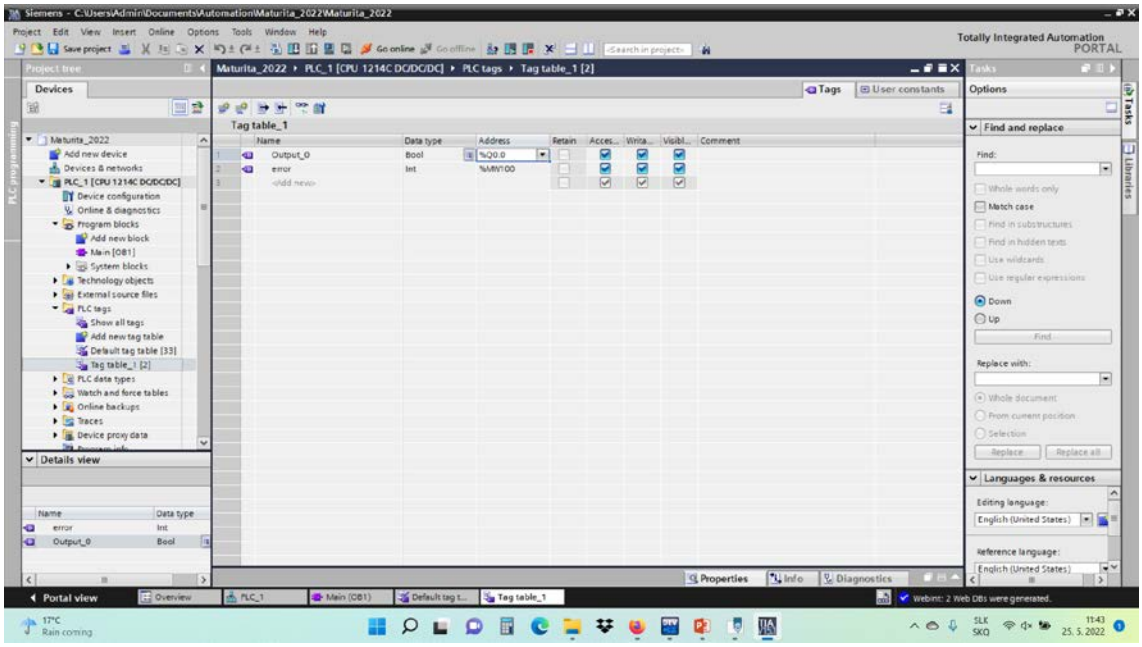
Vytvoríme webstránku na komunikáciu s PLC

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
 <head>
 <meta http-equiv="content-type" content="text/html; charset=utf-8">
 <meta name="generator" content="PSPad editor, www.pspad.com">
 <title>WEB server</title>
 </head>
 <body>
     <form>
        <input type="submit" value="Set Output_0">
        <input type="hidden" name='"Output_0"' value="1">
     </form>
     <form>
        <input type="submit" value="Reset Output_0">
        <input type="hidden" name='"Output_0"' value="1">
     </form>

     Output 0 : :="Output_0":
 </body>
</html>
```

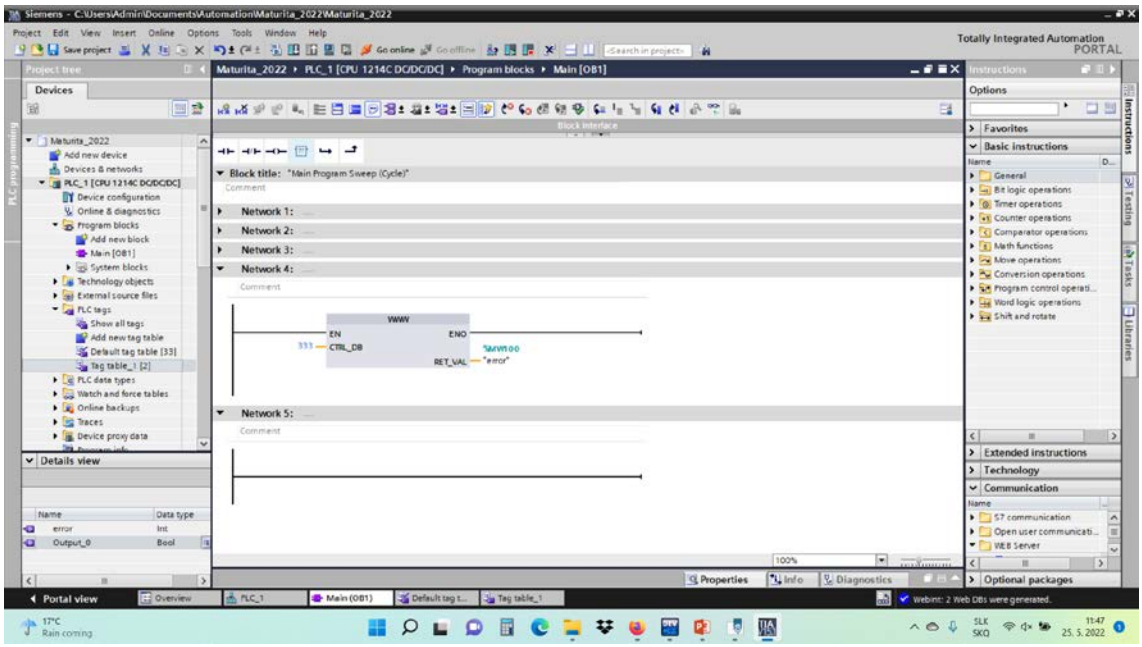
Nastavíme v hardvérovej konfigurácii Domovský web priečinok a názov nášho HTML súboru a vygenerujeme databázu údajov pre www.



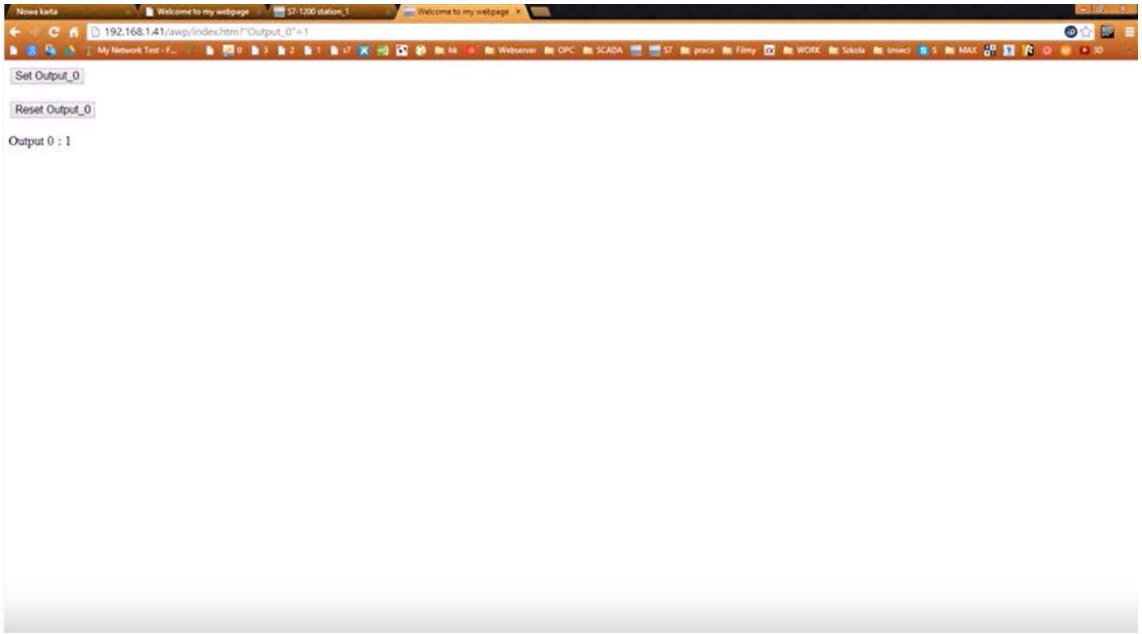
Vytvoríme premennú pre výstup Q0.0 a premenná typu Int pre prípadné uloženie chybových stavov.



Vložíme blok pre komunikáciu s WEB serverom a web stránkou (inštrukcia z Communication).

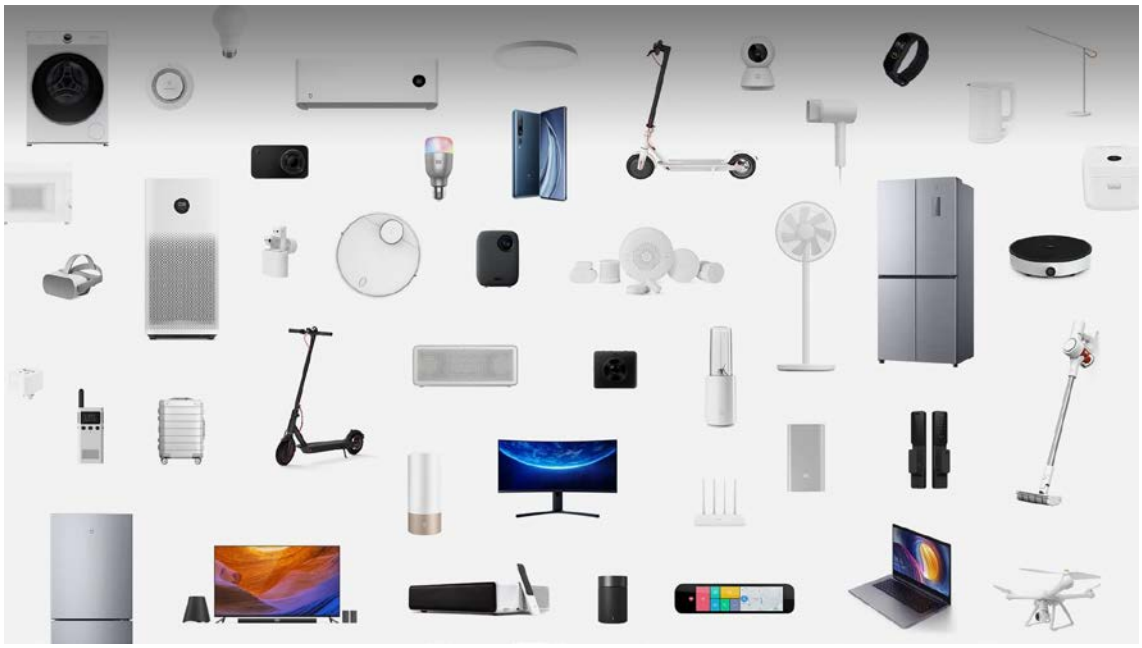


V prehliadači zadáme IP adresu PLC a následne prejdeme na stránku s komunikáciu a môžeme riadiť výstup Q0.0 cez web rozhranie (zapínať / vypínať).



3 XIAOMI ECO-SYSTEM

Ide o systém, ktorý spravuje množstvo zariadení používaných v domácnosti.



3.1 Spôsoby komunikácie:

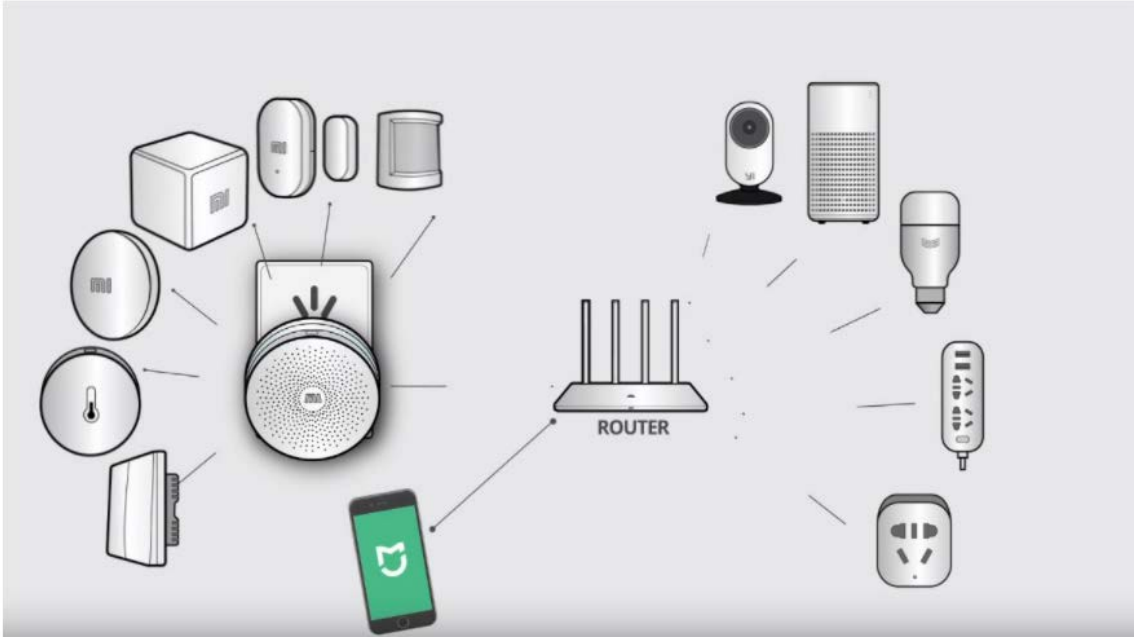
- WIFI
- Bluetooth
- BLE(Bluetooth Low Energy)

3.2 Najrozšírenejším spôsobom prepojenia je BLE:

Pre Ble brána pracovať, musíte k nim pripojiť alebo spárovať miniaplikácie podporujúce Bluetooth. Toto spárovanie umožňuje bráne prenášať údaje z pripojeného zariadenia do cloudu.

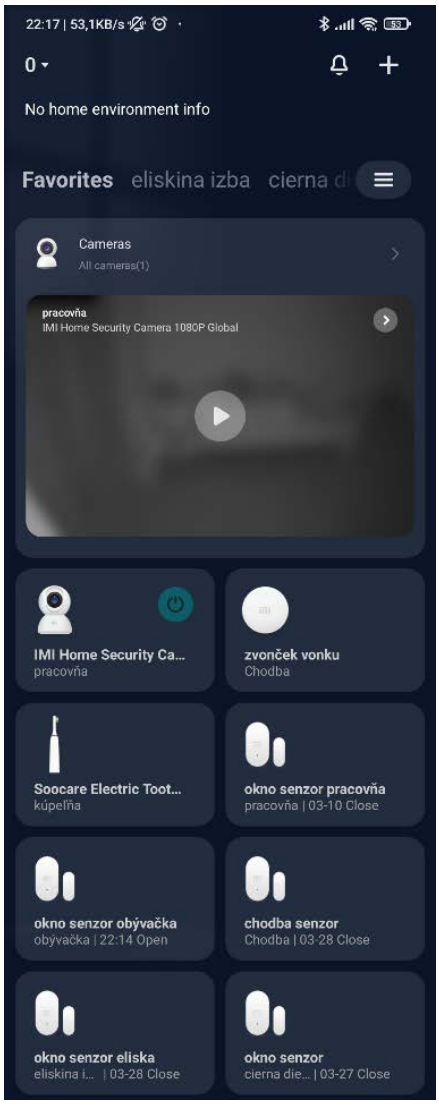
Akonáhle sa tieto informácie dostanú na cloudový server, okamžite zaregistruje pripojené zariadenie Bluetooth. Potom začne brána Bluetooth s nízkou energiou vyhľadávať všetky blízke zariadenia Bluetooth. V okamihu, keď nájde zariadenie v jeho dosahu, zhromažďuje všetky informácie, vrátane jeho funkcií.

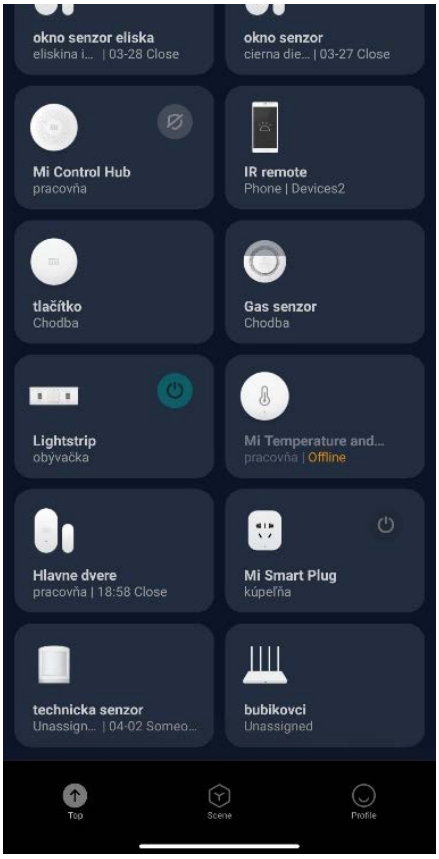
Potom začne dekódovať všetky požiadavky HTTPS z údajov v pamäti. Tieto informácie potom použije, napríklad, vlastnosti zariadenia, odpovedať na tieto žiadosti. Niekedy, požiadavky HTTPS sa môžu týkať skutočných údajov pripojeného zariadenia Bluetooth. V tomto prípade, automaticky sa pripojí k zariadeniu Bluetooth a zhromaždí potrebné informácie.



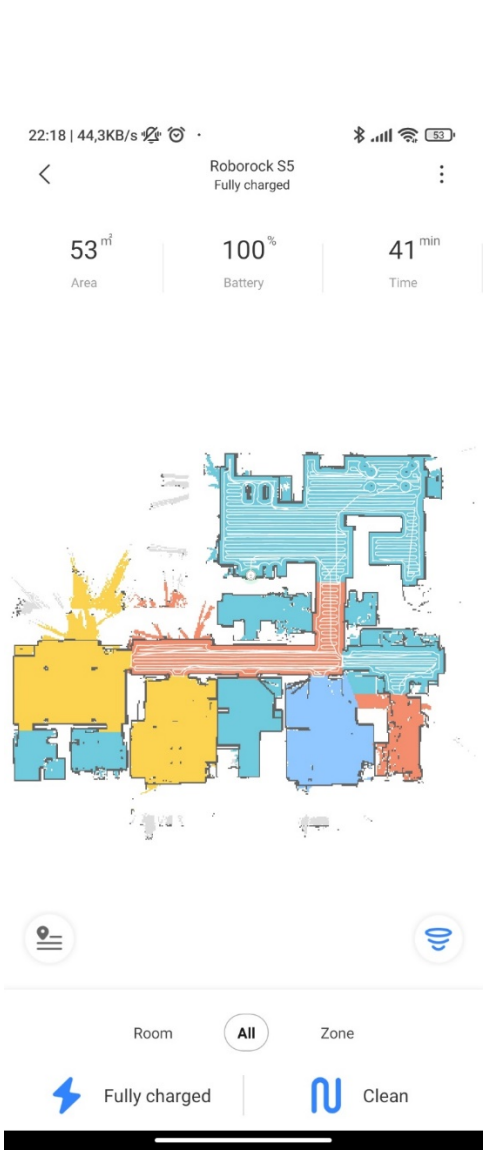
3.3 Rozhranie pre správu systému

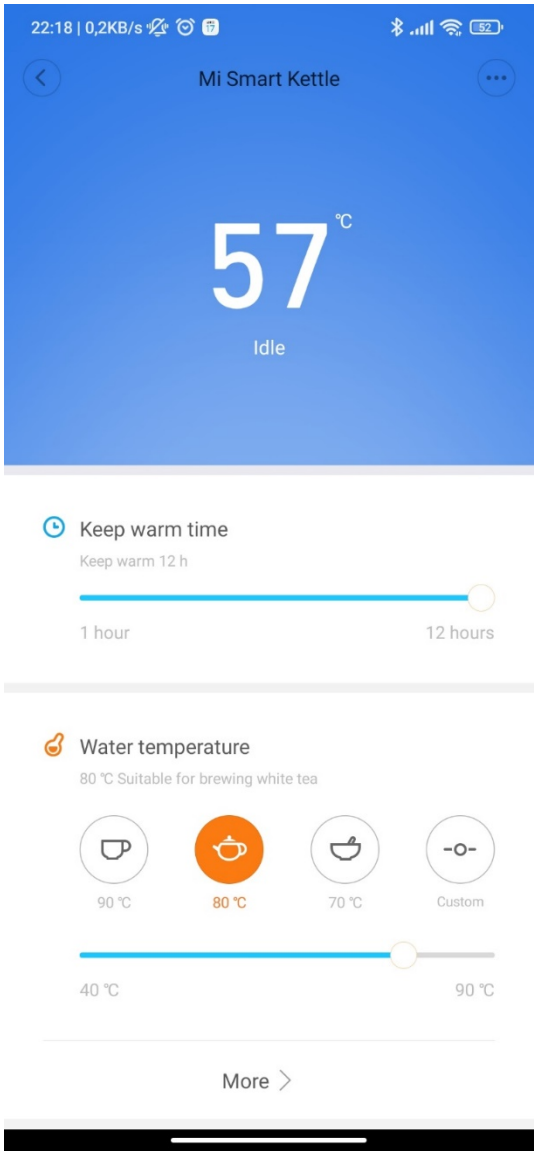
Správa pomocou Android aplikácie je nasledovná:





Rozhranie pre inteligentný vysávač a rýchlovarnú kanvicu:

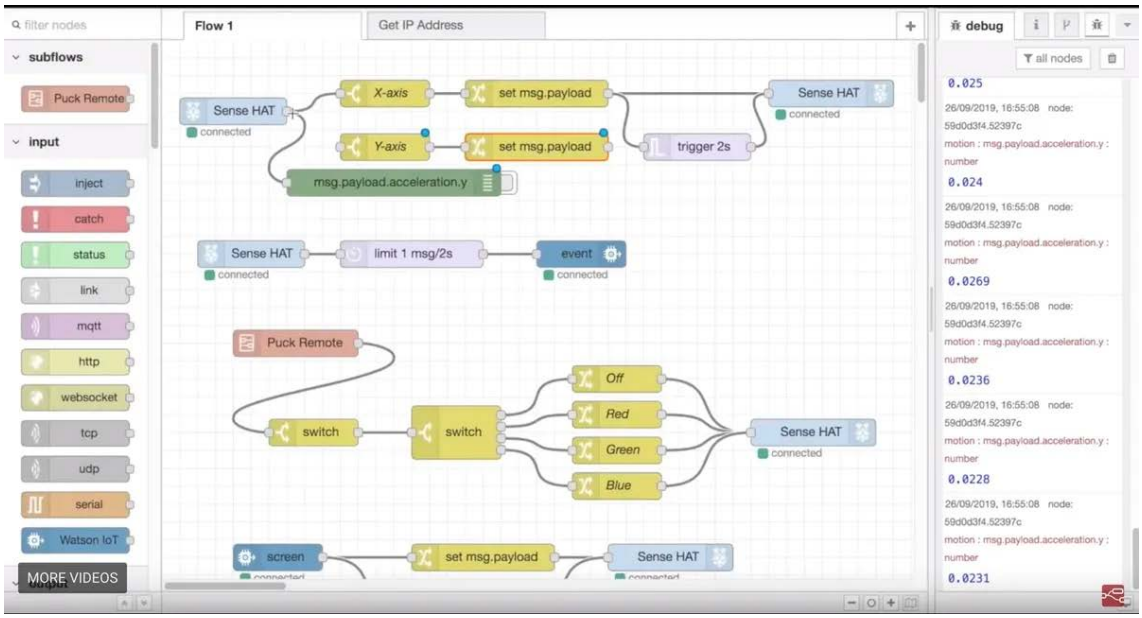




Viac informácii: https://www.mistores.sk/xiaomi-produkty/inteligentna-domacnost/

4 SERVEROVÁ SLUŽBA NODE-RED

Node-RED je programovací nástroj, ktorý umožňuje tvoriť program pomocou uzlov a blokov. Slúži najmä na tvorbu web-GUI, ale dokáže aj debugovanie, programovanie v JS atď.



4.1 Spustenie Node-RED

Existuje niekoľko spôsobov, ako sa dá táto služba spustiť:

- Online server (napr. FRED (https://fred.sensetecnic.com/), ktorý je zadarmo, avšak nutnosť denného prihlasovani sa)
- Inštalácia na Windows (pomerne komplikovaný spôsob, avšak ponúka možnosť ovládania zariadení cez COM)
- Inštalácia na Raspberry Pi (predinštalovaný v RP OS, ovládanie GPIO pinov)

4.2 Príklad 1 - demonštrovanie práce s Node-RED

Program vypisuje každú sekundu číslo 0/1. (Týmto programom sa demonštruje práca s Node-RED, dashoard prvky, zobrazenie web GUI, debugovanie, ...)

Postup:

1- V ľavom panely vybrať prvok INJECT a nastaviť na ňom interval opakovania 1 sekundu.

2- Výstup prvého prvku prepojiť s prvkom TRIGGER, na ktorom sa nastaví posielanie 1, počkanie 500 ms, poslanie 0.

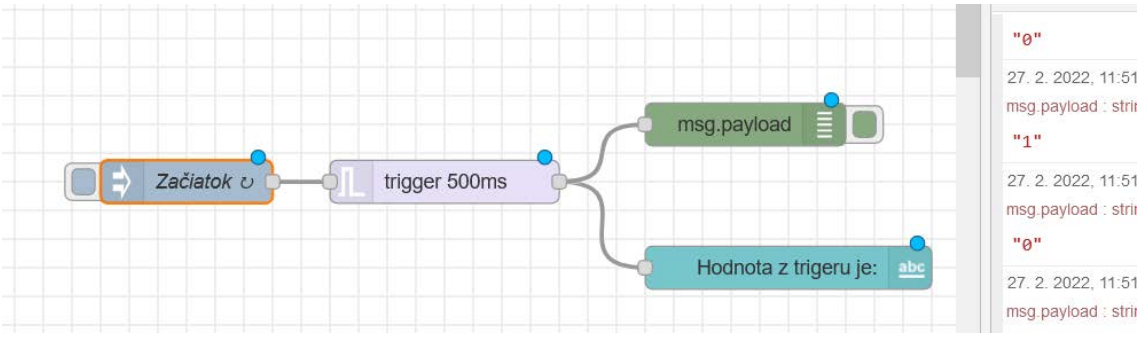
3- Výstup druhého prvku prepojiť s prvkom DEBBUG + check.

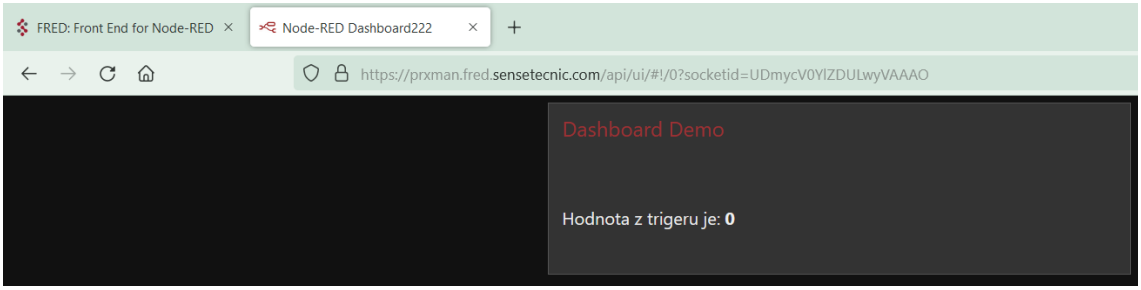
4- Výstup druhého prvku prepojiť s prvkom TEXT.

5- Potvrdiť zmeny tlačidlom DEPLOY vpravo hore.

6- Pre sledovanie DEBUG zvoliť v pravom okne ikonu chrobáka, alebo na konci šípku a tam DEBUG MESSAGES.

7- Pre zobrazenie web-GUI zvoliť na konci šípku a zvoliť DASHBOARD. Tuná upraviť výzor webu a poslednou ikonou (obdĺžnik so šípkou) otvoriť na novom okne web-GUI.





Program je možné rôzne modifikovať, napríklad použiť iný grafický prvok na zobrazenie hodnoty z TRIGGERU, alebo upraviť typ posielaných správ, alebo umiestniť DEBUG na iné miesto v toku (FLOWu).

4.3 Príklad 2 – demonštrovanie programovania

Program náhodné mení číslo v rozsahu 20 – 30. (Týmto programom sa demonštruje využitie programovania v JS, náhody, vizualizácia hodnoty cez dashboard, ...)

Postup:

1- V ľavom panely vybrať prvok INJECT a nastaviť na ňom interval opakovania 1 sekundu („msg“ sa nespracováva)

2- Výstup prvého prvku prepojiť s prvkom FUNCTION, do ktorého sa zapíše program:

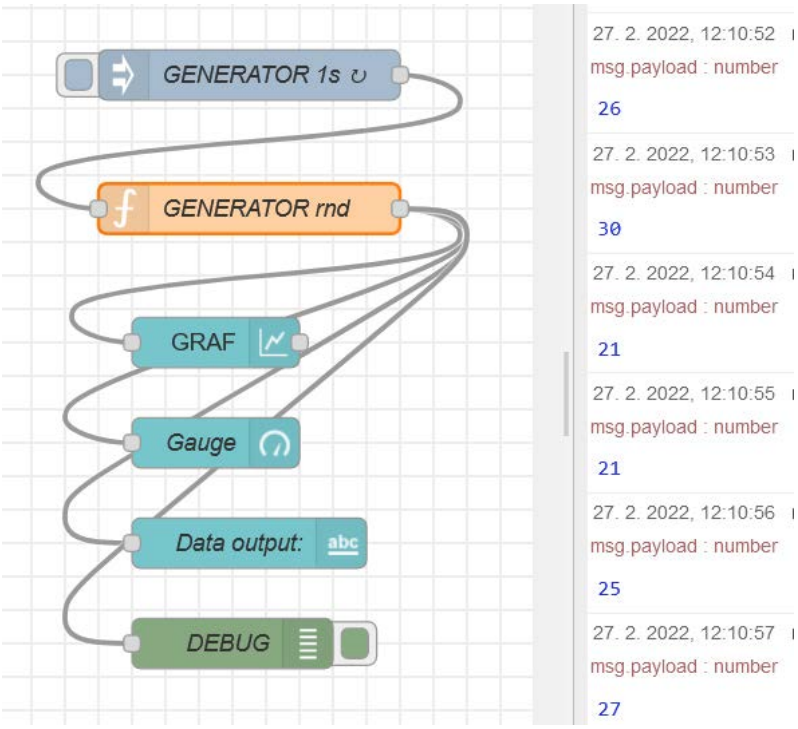
```
msg.payload = Math.round(  Math.random()*11 ) + 20;
return msg;
```

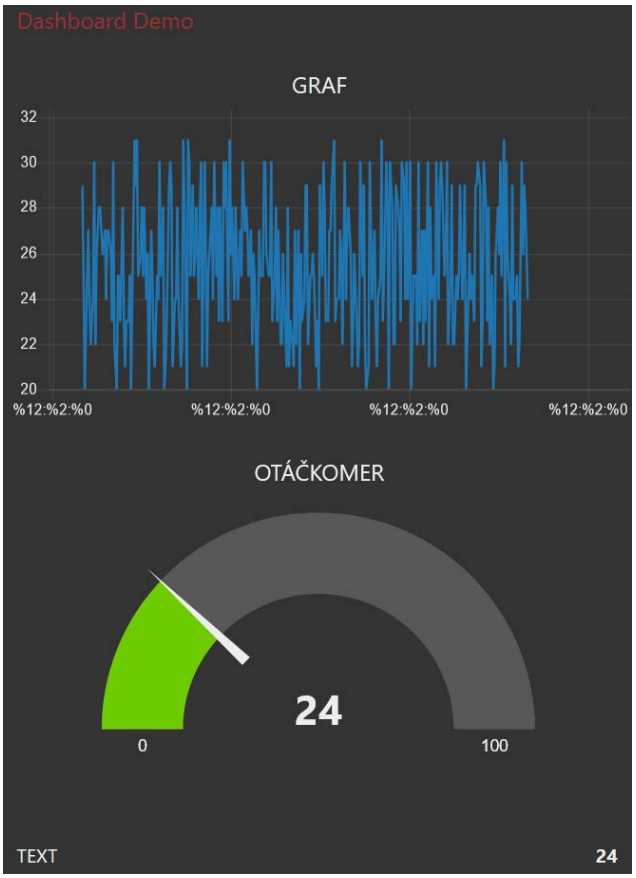
3- Výstup druhého prvku prepojiť s prvkami CHART, GAUGE, TEXT a DEBUG. Možnosť ľubovoľne editovať ich zobrazenie.

4- Potvrdiť zmeny tlačidlom DEPLOY vpravo hore.

V JS kóde sa zapíše do správy MSG náhodné číslo 20 až 30. Táto správa sa celá pošle na výstup. WEB-GUI prvky z nej čítajú obsah (payload).

Funkcia Math.random() navráti náhodné číslo 0 až 1. Prenásobením 11 sa číslo zmení na 0 až 10. Math.round() toto číslo upraví na celé a pričítaním 20 sa upraví na požadovaný rozsah 20 až 30.


GENERATOR 1s
GENERATOR rnd
GRAF
Gauge
Data output:
abc
DEBUG
27. 2. 2022, 12:10:52
msg.payload : number
26
27. 2. 2022, 12:10:53
msg.payload : number
30
27. 2. 2022, 12:10:54
msg.payload : number
21
27. 2. 2022, 12:10:55
msg.payload : number
21
27. 2. 2022, 12:10:56
msg.payload : number
25
27. 2. 2022, 12:10:57
msg.payload : number
27


Dashboard Demo
GRAF
32
30
28
26
24
22
20
%12:%2:%0
%12:%2:%0
%12:%2:%0
%12:%2:%0
OTÁČKOMER
24
0
100
TEXT
24

4.4 Príklad 3 - demonštrovanie komunikácie s ESP32

Program prijíma dáta z MQTT (naň odosiela ESP32). (Týmto programom sa demonštruje komunikácia s riadiacou jednotkou (konkrétne ESP32 s programom v jazyku C++), práca s MQTT protokolom a serverom HiveMQ, ...)

Postup:

1- V ľavom panely vybrať prvok MQTT IN a nastaviť na ňom adresu servera (broker.hivemq.com), číslo portu (1883) a topic (SPSKNM/TOPIC)

2- Výstup prvého prvku prepojiť s prvkom GAUGE, v ktorom sa môže nastaviť rozsah hodnôt 20 až 30.

3- Potvrdiť zmeny tlačidlom DEPLOY vpravo hore.

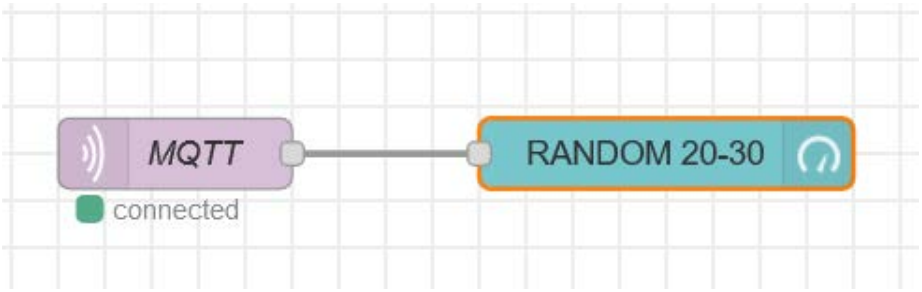
K ESP32 stačí pripojiť napájanie. To sa pripojí na nastavenú sieť a začne odosielať náhodné čísla v rozsahu 20-30 každú sekundu.

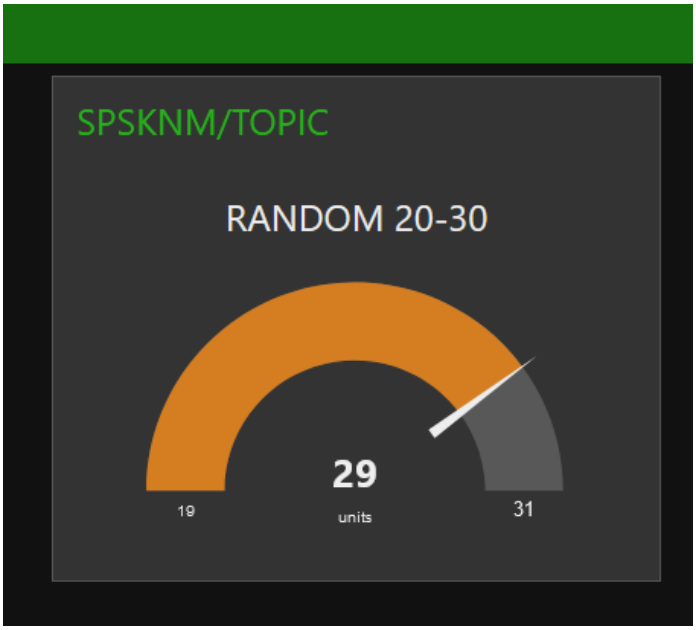
Výsledok posielania dát sa dá overiť buď na broker servery HiveMQ po pripojení sa do topiku SPSKNM/TOPIC, alebo vo web-GUI servera FRED.

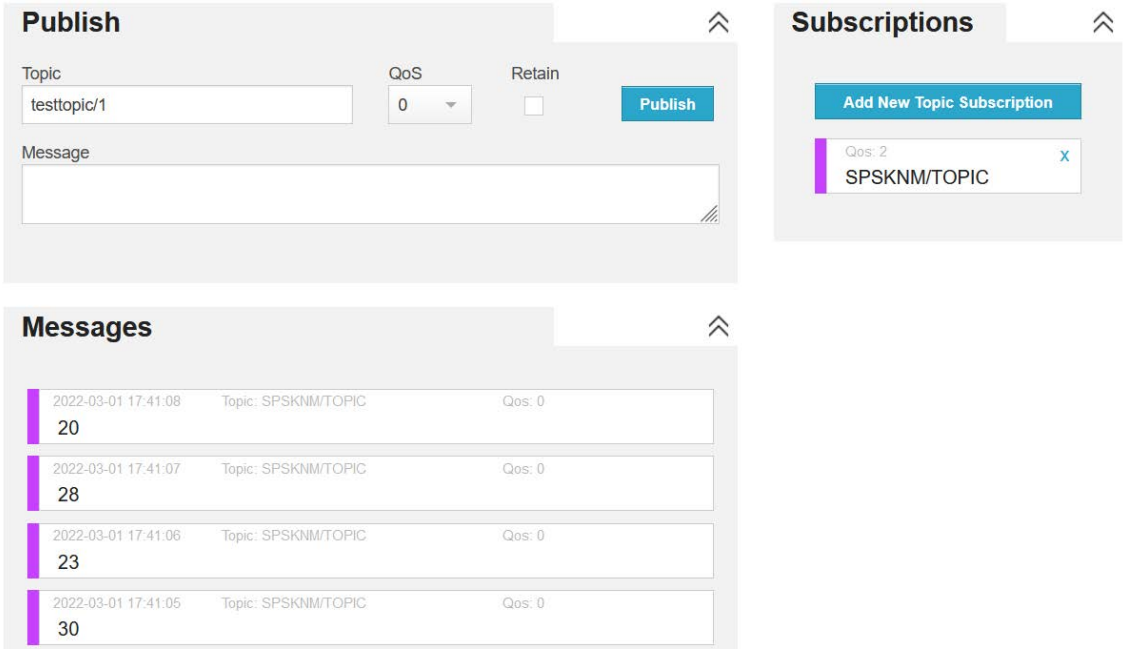
```
// ZDOJOVÝ KÓD PRE ESP32 ->
#include <WiFi.h>
#include <PubSubClient.h>

WiFiClient WIFI_CLIENT;
PubSubClient MQTT_CLIENT;

const char* ssid = "spsknm";
const char* password = "tazkeheslo";
```

```
void setup()
{
  Serial.begin(9600);
  delay(10);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.print("\nWiFi conected. IP: ");
  Serial.println(WiFi.localIP());
}



void loop()
{
  if (!MQTT_CLIENT.connected())
    reconnect();

  char udaj[3];
  String( random(20,31) ).toCharArray(udaj,3);
  MQTT_CLIENT.publish("SPSKNM/TOPIC", udaj);
  delay(1000);

}
```

```
void reconnect()
{
  MQTT_CLIENT.setServer("broker.hivemq.com", 1883);
  MQTT_CLIENT.setClient(WIFI_CLIENT);
  while (!MQTT_CLIENT.connected())
  {
    Serial.println("Pripajanie sa ku MQTT brokeru...");
    MQTT_CLIENT.connect("UCITEL");
    delay(3000);
  }
  Serial.println("Pripojenie prebehlo uspesne!");
}
```







Adresy na správu jednotlivých okien:

http://hivemq.com/demos/websocket-client/

https://prxman.fred.sensetecnic.com/api/ui

5 REALIZÁCIA KOMPLEXNEJ IOT ÚLOHY

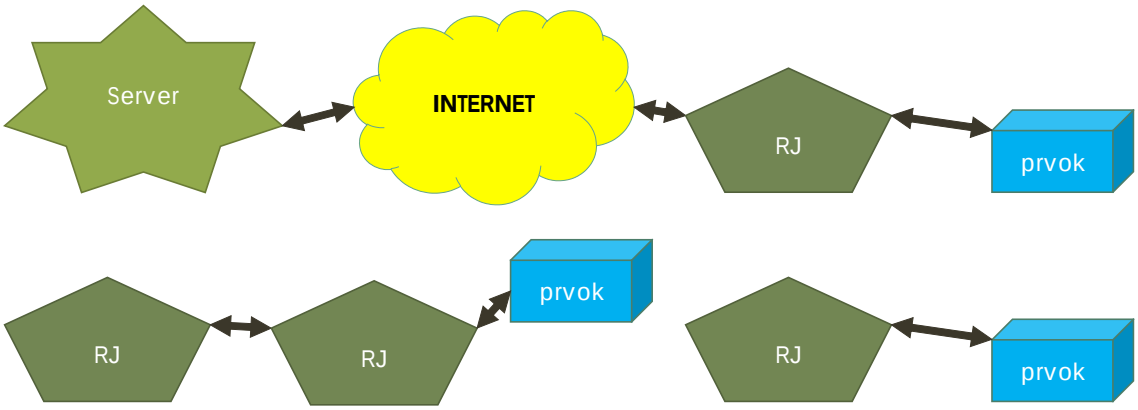
Nasledujúca kapitola sa zaoberá komplexným riešením úlohy pre oblasť IoT. Úloha teda obsahuje koncové zariadenie, prenos dát cez sieť a webové rozhranie pre spravovanie celého systému. Úloha demonštruje využitie MQTT protokolu na platforme Raspberry Pi a teda aj programovanie v jazyku Python.

Materiálne vybavenie

- Raspberry Pi (v3)
- Logický prevodník (3,3V-5V)
- Neopixel strip (ws2812)
- Zdroj napätia (5V/2A)

Spôsoby riadenia

- Server - Riadiaca jednotka – elek. prvky
- Riadiaca jednotka - riadiaca jednotka – elek. prvky
- Riadiaca jednotka – elek.prvky



Z uvedených spôsobov riadenia elek. prvok sa úloha zameriava na ten najkomplexnejší a teda prvý: server – riadiaca jednotka – elek. prvky.

5.1 Opis projektu

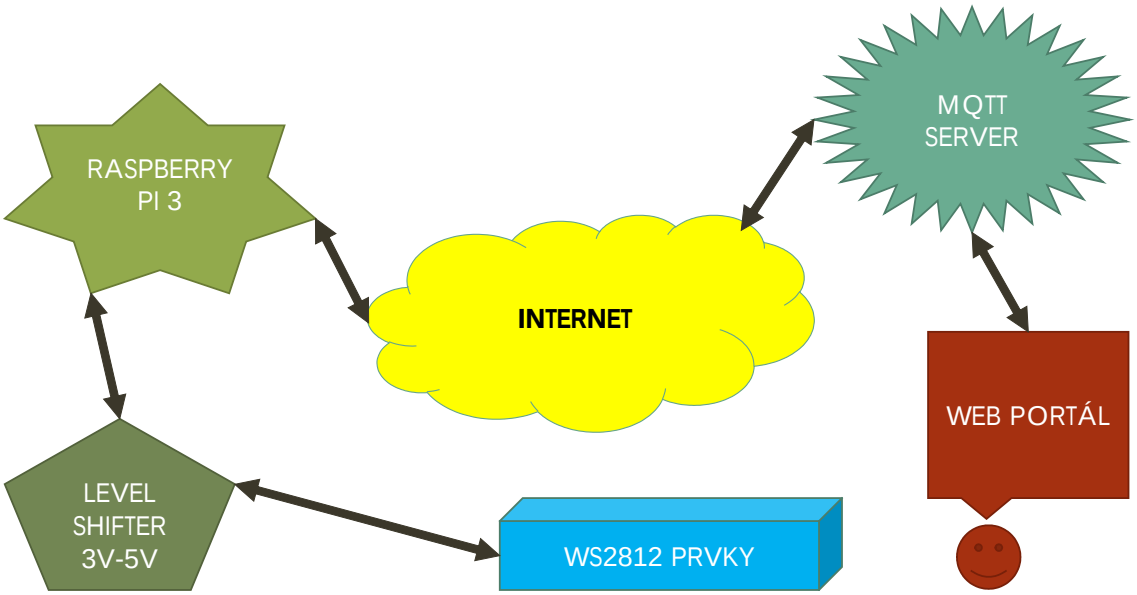
Na spustenie skriptu je nutné nainštalovať si na RP Python v3.

Okrem toho treba nainštalovať knižnicu pre prvky NeoPixel (rpi_ws281x). Link pre prácu s touto knižnicou nájdete na: https://github.com/rpi-ws281x/rpi-ws281x-python Druhu knižnicou je Paho-MQTT, ktorá slúži na prácu s protokolom MQTT. Link na prácu s touto knižnicou nájdete na: https://www.emqx.com/en/blog/use-mqtt-with-raspberry-pi

V skripte je nutné nadefinovať si počet NeoPixel prvkov a pin, na ktorý sa pripája dátový vstup zobrazovacieho prvku. (Defaultne je v programe D18).

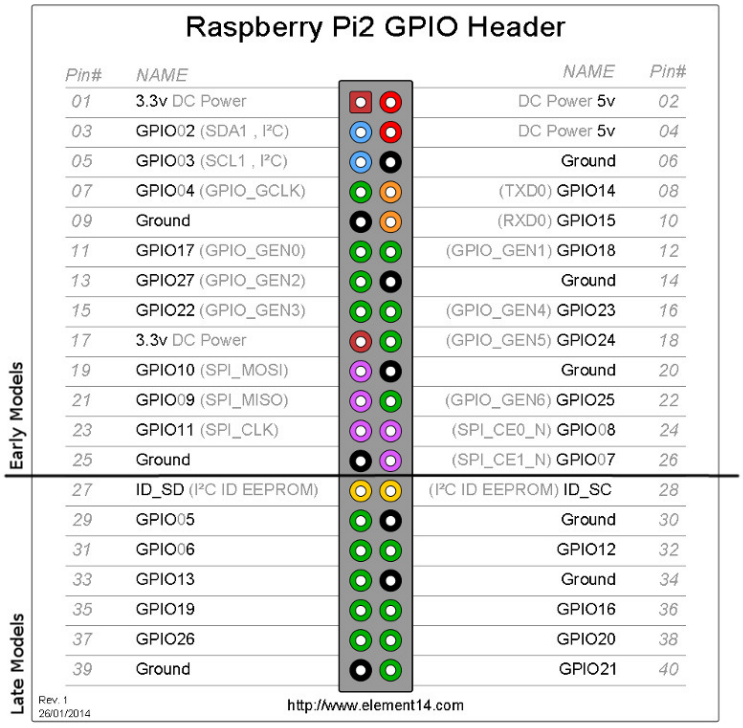
Keďže RP využíva 3,3V logiku a NeoPixel prvok 5V logiku, je nutné prispôsobiť tieto dva obvody. Riešení je niekoľko, základné schémy nájdete na odkaze: https://learn.adafruit.com/neopixels-on-raspberry-pi/raspberry-pi-wiring Spotreba jedného prvku sa síce udáva 60mA, ale v praxi býva menšia. I tak však treba dať pozor na zabezpečenie dostatočného zdroja prúdu, lebo inak dôjde k zmene farieb (napr. namiesto bielej môže vyžarovať prvok žltú/fialovú farbu a pod.) alebo k zmene poradia vysvieteného prvku, či iným anomáliám.

Keďže základom projektu je riadenie NeoPixel prvkov cez webové rozhranie, je logické, že RP musí byť pripojené na internet. Samotné ovládanie prvkov je možné cez webstránku: http://namakanyden.sk/2021/christmas.tree.html Ako alternatíva sa môže využiť aj HiveMQ klient: http://hivemq.com/demos/websocket-client/ Po pripojení sa do topicu „namakanyden/things/stromcek“ sa odošle príkaz v tvare: {"scenario":"fade","color":[154,80,31],"duration":"40"}



5.2 Elektrické zapojenie

Schéma zapojenia vychádza zo zdrojového kódu. Je veľmi jendoduchá, nakoľko stačí prepojiť riadiacu jednotku RP, zdroj napájania a zobrazovací prvok neopixel. Z toho dôvodu sa tu neuvádza a žiak musí prísť na jej tvar sám, čím sa prejaví jeho analytické myslenie. Pre zapojenie je však nutné poznať pinout RP, ktorý žiak nemusí ovládať z pamäti, preto sa tu ten uvádza.



5.3 Kód pre Raspberry Pi

Opis zdrojového kódu je čiastočne realizovaný formou komentára v kóde. Z veľkej časti sa však očakáva analýza žiaka a spoznanie kódu jeho čiastočným debugovaním:

```
import time

import random

import paho.mqtt.client as mqtt

import json

from rpi_ws281x import *


LED_COUNT      = 24      # Number of LED pixels.

LED_PIN        = 18      # GPIO pin connected to the pixels (18 uses PWM!).

LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)

LED_DMA        = 10      # DMA channel to use for generating signal (try 10)

LED_BRIGHTNESS = 200     # Set to 0 for darkest and 255 for brightest

LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)

LED_CHANNEL    = 0       # set to '1' for GPIOs 13, 19, 41, 45 or 53

MQTT_TOPIC     = "namakanyden/things/stromcek"

MQTT_BROKER    = "broker.hivemq.com"
```

```
def clear(pixels):
    print('>> running clear')
    while True:
            for i in range(pixels.numPixels()):
                pixels.setPixelColor(i, Color(0,0,0))
            pixels.show()
            yield


def set_color(pixels, color):
    print('>> running set color (' + str(color) + ")")
    while True:
            for i in range(pixels.numPixels()):
                pixels.setPixelColor(i, Color(color[0], color[1], color[2]))
            pixels.show()
            yield


def set_random_color(pixels, wait):
    print('>> running set_random_color')
    while True:
        for i in range(pixels.numPixels()):
            pixels.setPixelColor(i, Color(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)))
```

```
        pixels.show()
        time.sleep(wait/1000.0)
        yield

def bounce(pixels, color, wait, once=False):
    for i in range(pixels.numPixels()):
        pixels.setPixelColor(i, Color(color[0], color[1], color[2]))
    pixels.show()
    print('>> running bounce')
    while True:
        for i in range(pixels.numPixels()):
            pixels.setPixelColor(i, Color(0, 0, 0))
            if i-1 != -1:
                pixels.setPixelColor(i-1, Color(color[0], color[1], color[2]))
            else:
                pixels.setPixelColor(pixels.numPixels()-1, Color(color[0], color[1], color[2]))
            pixels.show()
            time.sleep(wait/1000.0)
            yield
        if once == True:
            break
```

```
def cycle(pixels, color, wait, once=False):
    next(clear(pixels))
    print('>> running cycle')
    while True:
        for i in range(pixels.numPixels()):
            pixels.setPixelColor(i%pixels.numPixels(), Color(color[0], color[1], color[2]))
            pixels.show()
            time.sleep(wait/1000.0)
            pixels.setPixelColor(i, Color(0, 0, 0))
            yield
        else:
            pixels.show()
        if once == True:
            break


def wheel(pos):
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
```

```
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)


def rainbow_cycle(pixels, wait, once=False):
    print('>> running rainbow_cycle')
    next(clear(pixels))
    while True:
        for j in range(255):
            for i in range(pixels.numPixels()):
                rc_index = (i * 256 // pixels.numPixels()) + j
                (r,g,b) = wheel(rc_index & 255)
                pixels.setPixelColor(i, Color(r,g,b))
            pixels.show()
            time.sleep(wait/1000.0)
            yield
        if once == True:
            break
```

```
def fade(pixels, color, wait, once=False):
    next(clear(pixels))
    print('>> running fade')
    while True:
        for i in range(0, 4 * 256, 8):
            for j in range(pixels.numPixels()):
                if (i // 256) % 2 == 0:
                    val = i & 0xff
                else:
                    val = 255 - (i & 0xff)
                pixels.setPixelColor(j, Color(val, color[1], color[2]))
            pixels.show()
            time.sleep(wait/1000.0)
            yield
        if once == True:
            break


def dim_color(color, width):
    for i in range(0,3):
        color[i] = color[i]/width
    return color
```

```
def knight_rider(pixels, color, wait):
    next(clear(pixels))
    print('>> running knight rider')
    pos = 2
    direction = 1
    while True:
        #pixels.setPixelColor(pos - 2, Color(16, 0, 0))  # Dark red
        #pixels.setPixelColor(pos - 1, Color(128, 0, 0))  # Medium red
        #pixels.setPixelColor(pos, Color(255, 0, 0))  # brightest
        #pixels.setPixelColor(pos + 1, Color(128, 0, 0))  # Medium red
        #if (pos + 2) < pixels.numPixels():
        #    pixels.setPixelColor(pos + 2, Color(16, 0, 0))
        pixels.setPixelColor(pos-2,Color(color[0]>>5, color[1]>>5, color[2]>>5))
        pixels.setPixelColor(pos-1,Color(color[0]>>2, color[1]>>2, color[2]>>2))
        pixels.setPixelColor(pos,Color(color[0], color[1], color[2]))
        pixels.setPixelColor(pos+1,Color(color[0]>>2, color[1]>>2, color[2]>>2))
        if(pos+2) < pixels.numPixels():
            pixels.setPixelColor(pos+2,Color(color[0]>>5, color[1]>>5, color[2]>>5))
        pixels.show()
        time.sleep(wait/1000.0)
```

```
        yield
        for j in range(-2, 2):
            pixels.setPixelColor(pos + j, Color(0, 0, 0))
            if (pos + 2) < pixels.numPixels():
                pixels.setPixelColor(pos + 2, Color(0, 0, 0))
        pos += direction
        if pos - 2 < 0:
            pos = 2
            direction = -direction
        elif pos >= (pixels.numPixels() - 1):
            pos = pixels.numPixels() - 2
            direction = -direction


def light_up_part_of_tree(pixels, color, parts, index):
    next(clear(pixels))
    print('>> running light up part of the tree')
    while True:
            group = pixels.numPixels() / parts
            start_idx = int(pixels.numPixels() - group * (index + 1))
            last_idx = int(start_idx + group)
            for i in range(start_idx, last_idx + 1):
```

```
                pixels.setPixelColor(i, Color(color[0],color[1],color[2]))
            pixels.show()
            yield



def on_connect(client, userdata, flags, rc):
        print(f"Connected with result code {rc}")
        client.subscribe(MQTT_TOPIC)


def on_message(client, userdata, msg):
        global scenario
        try:
                topic = msg.topic
                message = msg.payload.decode("utf-8")
                print("message received: '{}'".format(message))
                payload = json.loads(message)
                color = payload.get("color", [0,0,0])
                delay = int(payload.get("duration", 100))
                name = payload.get("scenario", "clear")
                if name == "clear":
                        scenario = clear(strip)
```

```
elif name == "rainbow":
        scenario = rainbow_cycle(strip, delay)
elif name == 'set color':
        scenario = set_color(strip, color)
elif name == 'bounce':
        scenario = bounce(strip, color, delay)
elif name == 'cycle':
        scenario = cycle(strip, color, delay)
elif name == 'random color':
        scenario = set_random_color(strip, delay)
elif name == 'fade':
        scenario = fade(strip, color, delay)
elif name == 'knight rider':
        scenario = knight_rider(strip, color, delay)
elif name == 'part of tree':
        index = int(payload.get('index', 0))
        parts = int(payload.get('parts', 10))
        scenario = light_up_part_of_tree(strip, color, parts, index)
else:
        print('Error: Unknown scenario "{}"'.format(name))
        return
```

```
        except Exception as ex:
                print('Error: {}'.format(ex))
                return
        #finally:
                #gc.collect()


strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT, LED_BRIGHTNESS, LED_CHANNEL)
strip.begin()
global scenario
scenario = clear(strip)


client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.will_set("rasperry/status", b"{'status': 'Off'}")
client.connect(MQTT_BROKER, 1883, 60)


print('waiting for message...')
while True:
        client.loop(.01)
        next(scenario)
```