ESENDER:

1. reconnectmqttserver():
   * 1. táto funkcia opätovne každých 5 sekúnd pripája ESP8266 k MQTT brokeru
2. callback():
   * 1. je volaná, keď nejaké zariadenie zverejní správu k téme, ku ktorej je ESP8266 pripojený
3. setup():
   * 1. zahájenie sériovej komunikácie s ESP32
     2. pripojenie na zadanú wifi sieť
     3. nastavenie mqtt brokeru
4. loop():
   * 1. odpočúvanie sériovej komunikácie s ESP32
     2. oddeľovanie jednotlivých údajov
     3. posielanie na mqtt server
5. Rozšírenia:
   * 1. bolo by rozumné informovať EServer o dostupnosti MQTT servera. V EServeri je to možné pridať do mainThreadu pre kontrolu. Resp treba odkomentovať a prípadne odladiť else.

ESTACK:

1. trieda StackItem:
   * 1. trieda uchovávajúca uložený prvok zásobníka a ukazovateľ na ďalší prvok
        1. setNext():

nastavuje ukazovateľ na ďalší prvok v zásobníku

* + - 1. getData():

v reťazci vracia prvok zvolený zásobníka

* + - 1. getNext():

vracia ukazovateľ na zvolený prvok zo zásobníka

1. trieda Estack:
   * 1. samotný zásobník zložený z objektov StackItem. Ak by som chcel celistvú pamäť, musel by som implementovať obdobu vectora, pričom by bolo rozumné riešiť aj resize pri zmenšovaní. kvôli alokácií celistvej pamäte a teda možnej FRAGMENTÁCII, čo by mohlo znamenať spadnutie programu. Toto je jeden z hlavných dôvodov, prečo som zvolili cestu zreťazenej pamäte.
        1. vypisZasobnika():

vypíše celý obsah zásobníka

* + - 1. getSize():

vráti aktuálnu veľkosť zásobníka

* + - 1. getMaxSize():

vráti veľkosť zásobníka rezervovanú v pamäti

* + - 1. isEmpty():

vráti pravdivostnú hodnotu, či v zásobníku niečo je

* + - 1. isFull():

vráti pravdivostnú hodnotu, či je zásobník plný

* + - 1. pushFront():

pridáva nové prvky spredu zásobníka

* + - 1. push():

pridáva nové prvky zozadu

* + - 1. pop():

vracia postupne prvky zásobníka odzadu a maže ich

* + - 1. clear():

vyčistí celý zásobník

* + - 1. Deštruktor:

vyvolá metódu vyčistí všetky itemy, resp pokiaľ nasledujúci je rôzny od nullptr. Je to z dôvodu zreťazenia pamäte. Táto pamäť má špecifikum, že nedokážeme pristúpiť priamo na prvok, iba pomocou predchodcov. Ak by som mal obojstranne zreťazený zoznam mohlo by sa pristupovať aj z konca.

* + - 1. Info:

V zásobníku nie sú implementované iterátory, teda buď ich treba doimplementovať, alebo nie je možné použiť foreach cyklus

ESENSOR:

1. createData():
   * 1. zostaví paket (dátový alebo autorizačný) z poľa
2. sendPacket():
   * 1. odošle vytvorený paket na zvolenú ip a port
3. getBatteryCapacity():
   * 1. vráti napätie batérie z analog. portu, tuto treba doladiť koeficienty, konkrétne zadané sa použili pre testovacie odpory
4. setup():
   * 1. pripojenie na prepojovacie zariadenie
     2. vytvorenie a poslanie autorizačného paketu
5. loop():
   * 1. vyčítanie teploty a vlhkosti zo senzora
     2. vyčítanie stavu batérie a následný prepočet
     3. vytvorenie dátového paketu
     4. posielanie dát. paketu
     5. uspanie zariadenia
     6. Info : v rámci senzora treba zistiť, či je EServer online a prípadne dáta ukladať do EEPROM. Spôsob zápisu keďže EEPROM má obmedzený garantovaný počet zápisov je možné riešiť príznakovým bitom. Teda nebude prepisovať začiatok a EEPROM vydrží dlhšie.

ESERVER:

1. Popis Tried:
   1. EDevice: uchováva informácie o zariadení
   2. ESD: trieda, komunikujúca s kartou, resp zapisujúca dáta a čítajúca ich.
      1. getData: naplní zásobník s údajmi z aktuálnej pozície, ktorú vyčíta z pomocného súboru
      2. writeData: zapíše dáta zo zásobníka do daného súboru a aktualizuje pomocný súbor na dané číslo riadku
      3. getLIStored: vráti posledné uložené ID, teda zároveň aj odoslané. ID kvôli databáze na serveri. Keďže ID pre každé zariadenie kvôli jednoznačnému určeniu môže byť rôzne teda napr. pre zariadenie s ID 1 : 100000012345. pričom 1000000 bude kvôli pretečeniu počtu packetov. Pri inicializácii sa použije ako prvá metóda pre getLastSentPosition, a následne sa pomocou seek dostaneme na aktuálnu pozíciu. Toto spôsobí ušetrenie N zapísaných prvkov.
      4. replaceLISentFile: musím prepísať posledne odoslané ID packetu v prípade, že server je nedostupný
      5. createFile: vytvorí súbor s daným názvom
      6. openFile: otvorí súbor s názvom, podľa parametra type buď ako append alebo read
      7. getLastSentPosition: pre použitú metódu seek, vráti pozíciu v súbore (na pamäťovej karte) na ktorej sa nachádza posledne odoslané ID paketu
      8. closeFile: zatvorí daný súbor.
      9. deštruktor: zatvorí súbor a zruší objekt \_file, pre čistotu pamäte
   3. EServer: trieda spravujúca chod prepojovacieho zariadenia - viem, že sa to dalo aj pomocou generalizácie, no pre projekt tohto rozsahu mi to neprišlo dôležité
      1. Konštruktor:
         1. inicializuje IP adresu, bránu a masku. vytvorí Software Serial komunikačný kanál, vytvorí objekt ESD, EStack pre buffer paketov, EStack pre packet čakajúce na odoslanie a EStack pre parametre aktuálne spracúvaného packetu.
      2. mainThread:
         1. funkcia hlavného vlákna s nekonečným while cyklom, ktorý som pre čistotu kódu dali do samotnej funkcie aj napriek tomu, že mohol byť v konštruktore. Tento while switchuje aby plnil všetky funkcie servera. Viem, že je možné manuálne riešiť vlákna v ESP32, no dočítal som sa, že kompilátor to automaticky rieši za mňa. Vznikal by tu problém s plánovaním procesov, ktorý by som musel riešiť.
      3. checkPackets:
         1. zistí príchodzie packety a automaticky ich rozparsuje do \_bufferu
      4. writeToCard:
         1. zapíše do požadovaného súboru parametre paketu, samozrejme za použitia už implementovaných funkcií a následne vyčistí parametre.
      5. writeToDataToSendFile:
         1. Je možné doimplemetovať, avšak pre aktuálne použitie to nie je potrebné
      6. continueRecievePacket:
         1. slúžila ako debugovacia metóda, v súčasnom stave nie je potrebná ako aj väčšina serialprintov.
      7. printDevices:
         1. vypíše na seriálový monitor všetky pripojené zariadenia. Odoberanie týchto zariadení by bolo rozumné vyriešiť. Napríklad: ak maximálny počet odoslaných paketov od ostatných zariadení je menší o napr. 2 od tohto. T eda časové okno je 30 minút od posledného odoslania paketu tohoto zariadenia, vieme povedať, že je odpojené a automaticky upovedomiť server o jeho prípadne chybovosti.
      8. tryAddDevice:
         1. skúsi pridať ďalšie zariadenie do poolu obsluhovaných zariadení. Tento počet som defaultne obmedzili na 15, avšak v knižnici je obmedzený na 4, teda v našom prípade je možné ho povoliť napr na defaultných 15. ak je plný počet zariadení nepridá. S touto funkcionalitou sa taktiež spája autorizácia zariadenia pomocou paketu, kedy sa s vyparsovaných parametrov vytvorí EDevice, ktoré je následne tomuto ID pridelené. Automaticky sa taktiež zavolajú funkcie spolupracujúce s ESD pre podrobnejšiu inicializáciu zariadenia. Tu chcem vyzdvihnúť, že tento packet je možno menej dôležitý avšak v prípade prelomenia hesla na EServer je dôležitý, teda hacker nebude schopný toto ESP používať ako dátovú banku.
      9. tryFindDevice:
         1. Štandardná funkcia slúžiaca na vyhľadanie devicu z už uložených zariadení.
      10. parseParams:
          1. automaticky vyparsuje parametre do \_packetParams, pričom oddeľovač medzi parametrami som použil “$” v prípade potreby je možné ho zameniť, avšak bude to nutné urobiť aj v ESenderi.
      11. trySendDataToSerial:
          1. ak je dostupný ESender, postupne mu pošle dáta z \_sendingBuffer cez sériovú komunikáciu
      12. sendPackets:
          1. v prípade, že je server dostupný a môže posielať, kontaktuje server a pošle príslušné pakety. doStuffPacket
      13. podľa druhu paketu buď priradí nové zariadenie pomocou tryAddDevice, alebo rieši zapisovanie, prípadne odosielanie dát na ESender.V prípade, že ESender nie je dostupný, tieto dáta uloží na SD kartu. Akonáhle zistí jeho dostupnosť, automaticky kvôli chronologickému poradiu pristúpi k dátam najskôr neodoslaným. Tento prístup som zvolil aj kvôli tomu, že MQTT server môže za predpokladu, že ani jeden s ESenderov nevypadol približne vypočítať čas, kedy boli dáta zozbierané. Samozrejme, že ak budeme riešiť problém s odpájaním zariadení, bude o tom treba upovedomiť server kvôli prepočtom
      14. listenFromServer:
          1. P2P jednoduchá komunikácia, na ktorú odpovedá ESender, aby informoval o stave prípadne možnosti prijatia nových paketov pre odoslanie na MQTT server
      15. contactServer:
          1. jednoduchá funkcia, kontaktuje server s dotazom “HelloMyFriend” lebo poznám CS GO.
      16. startAccessPoint:
          1. spustí access point s prednastavenými parametrami, resp Makrami, ktoré sú ve header file, Heslá, názov wifi, a pod.
      17. Deštruktor:
          1. vymaže \_device, buffer a packetParams. Som si vedomý, že kvôli čistote pamäti by bolo nutné vymazať aj ostatné veci alokujúce sa v konštruktore, avšak keďže na tomto zariadení beží iba táto aplikácia a nie je priebežne ukončená, nie je to potrebné. V prípade, že by som chcel viac aplikácií swapujúcich sa na tomto zariadení, už by som musel deštruktor dopísať aj o ostatné objekty.