```
bool initialization = false;
int ElectricityMeter[15][2]; //pin; countPulse
long electricityMeterTime[15]; //time detect of pulse
int electricityMeterDelayPulse = 15;
//int pinPushButton[15];
int pinRelay[15];

int countElectricityMeter;
int pushButton[15][2]; //pin; defaultState
long timeOffPushBtn[15]; //time to Off
int countPushButton;

int pinVoltageSignal[15];
boolean stateVoltageSignal[15];
int countVoltageSignal;

int countRelay;
boolean nacteno = false;



void setup() {
  // put your setup code here, to run once:
  Serial.begin(38400);
}

int findPushBtnPosition(int PIN){
  int found = 0;
  int numCols = sizeof(pushButton[0])/sizeof(pushButton[0][0]);
  for(int i = 0; i<=numCols; i++){
    if(PIN == pushButton[i][0]){
      found = i;
    }
  }
```

```
  return found;
}

void sendEchoVoltageSignal(int Pin, boolean state){
  int State;
  if(state){
    State = 1;
  }else{
    State = 0;
  }
  SendMsg('#' + String(Pin) + "-4-" + String(State) + '-');
}

void checkDevice(){
  for(int i = 0; i<countPushButton; i++){
    if(timeOffPushBtn[i]<millis()){
      digitalWrite(pushButton[i][0], LOW);
    }
  }

  for(int i = 0; i<countVoltageSignal;i++){
    boolean state = digitalRead(pinVoltageSignal[i]);
    if(state != stateVoltageSignal[i]){
      stateVoltageSignal[i] = state;
      sendEchoVoltageSignal(pinVoltageSignal[i], state);
    }
  }

  checkElectricityMeter();
}

void checkElectricityMeter(){
  for(int i = 0; i<countElectricityMeter; i++){
```

```
   if(digitalRead(ElectricityMeter[i][0]) == HIGH){//pokud se prepne ze stavu LOW na HIGH
     if(electricityMeterTime[i] < millis()){
       electricityMeterTime[i] = millis() + electricityMeterDelayPulse;
       ElectricityMeter[i][1]++;
       //Serial.println("Zaznamenan pulse2");
     }
   }
  }
  sendEchoElectricity();
}

void sendEchoElectricity(){
  for(int i = 0; i<countElectricityMeter; i++){
    if(ElectricityMeter[i][1]>=2){
      int pin = ElectricityMeter[i][0];
      int state = ElectricityMeter[i][1];
      SendMsg('#' + String(pin) + "-2-" + String(state) + '-');
      ElectricityMeter[i][1] = 0;
    }
  }
}

int booleanToInt(boolean value){
  if(value == true){
    return 1;
  }else{
    return 0;
  }
}

void responseStateDevice(int Pin, int DeviceType){
  int State;
  switch(DeviceType){
```

```
    case 1:
      State = booleanToInt(digitalRead(Pin));
      break;
    case 2:
      State = ElectricityMeter[findPushBtnPosition(Pin)][1];
      break;
    case 3:
      State = booleanToInt(digitalRead(Pin));
      break;
    case 4:
      State = booleanToInt(digitalRead(Pin));
      break;
  }

    SendMsg('#' + String(Pin) + '-'+ String(DeviceType) +'-' + String(State) + '-');

}

void addDevice(int Pin, int DeviceType, int state){
  switch(DeviceType){
    case 1:
      pinMode(Pin, OUTPUT);
      if(String(state) == "1"){
        digitalWrite(Pin, HIGH);
      }else{
        digitalWrite(Pin, LOW);
      }
      SendMsg("Nastavuji relay");
      break;
    case 2:
      pinMode(Pin, INPUT_PULLUP);
      ElectricityMeter[countElectricityMeter][0] = Pin;
      ElectricityMeter[countElectricityMeter][1] = 0;
```

```
      countElectricityMeter++;
      SendMsg("Nastavuji electricityMeter");
      break;
    case 3:
      pinMode(Pin, OUTPUT);
      pushButton[countPushButton][0] = Pin;
      countPushButton++;
      pushButton[countPushButton][1] = 0;
      if(String(state) == "1"){
        digitalWrite(Pin, HIGH);
      }else{
        digitalWrite(Pin, LOW);
      }
      SendMsg("Nastavuji PushSwitch");
      break;
    case 4:
      pinMode(Pin, INPUT_PULLUP);
      break;

  }
}



void SendMsg(String str){
  Serial.println(str);
  Serial.flush();
  delay(10);
}

void changeDevice(int Pin, int DeviceType, int state){
    switch(DeviceType){
      case 1:
```

```
      if(String(state) == "1"){
        digitalWrite(Pin, HIGH);
        SendMsg("Zapinam relay");
       }else{
        digitalWrite(Pin, LOW);
        SendMsg("Vypinam relay");
       }
       break;
     case 3:
      digitalWrite(Pin, HIGH);
      SendMsg("Menim PushSwitch#");
      timeOffPushBtn[findPushBtnPosition(Pin)] = millis() + pushButton[findPushBtnPosition(Pin)][1];
      break;
   }
}

void doCommand(int Pin, int DeviceType, int State){
   if(Pin == 1 && DeviceType == 0 && State == 0){ //string = 1-0-0 -> inicializace
    // ----- ODESLANI INFORMACE O INICIALIZACI
    if(initialization){
     SendMsg("CHYBA: Inicializace byla jiz spustena");
    }else{
      SendMsg("Spoustim inicializaci");
    }
    // ----- KONEC ODESILANI
    initialization = true;
   }else if(Pin == 0 && DeviceType == 0 && State == 0){ // string = 0-0-0 -> konec inicializace
    // ----- ODESLANI INFORMACE O INICIALIZACI
    if(!initialization){
     SendMsg("CHYBA: Inicializace byla jiz ukoncena");
    }else{
      SendMsg("Ukoncuji inicializaci");
    }
```

```
    // ----- KONEC ODESILANI
    initialization = false;
    nacteno = true;
   }

   if(initialization){
    //Serial.println(State);
    addDevice(Pin, DeviceType, State);
   }else if(nacteno){
    changeDevice(Pin, DeviceType, State);
   }
}

void loop() {
  bool cteni = false;
  //char receiveChar; // přečtený příchozí znak

  char receiveChar; // přečtený příchozí znak
  String readText; // zde se skládá sekvence přečtených znaků
  int pin; // zapíše se hodnota prvního parametru
  int deviceType; // druhy parametr
  String state; // treti parametr
  boolean processRead = false; // proces započne až po přečtení prvního znaku, kterym je #
  int processPart = 0; // cislo zpracovane casti - 1. je pin, pak prejde na deviceType a state
  long timeToRead = 0; // po precteni prvniho znaku se nastavi na 100, jedna se timeOut
delay(10);

 /* if(timeToRead > millis()){
  readText = "";
  pin = 0;
  deviceType = 0;
  state = "";
  processRead = false;
```

```
 processPart = 0;
 timeToRead = 0;
 }*/

 while(Serial.available() > 0 || timeToRead > millis()) { //timeToRead nastavi cas, po který se čeká na konec sekvence
  if(Serial.available() > 0){
   if(timeToRead != 0){
    timeToRead = millis()+100;
   }
   receiveChar = Serial.read();
   //Serial.print("Znak: ");
   //Serial.println(receiveChar);

   //Serial.print("Process: ");
   //Serial.println(processRead);
   if(processRead == true && receiveChar != '-'){
    readText = String(readText + receiveChar);
    //Serial.println("platim");
   }

   if(receiveChar == '-'){
    switch(processPart){
     case 0:
     pin = readText.toInt();
     break;
     case 1:
     deviceType = readText.toInt();
     break;
     case 2:
     state = readText;
     break;
    }
```

```
    readText = "";
    processPart++;
    delay(10);
     if(processPart == 3){//pokud nacte treti pomlcku, pak pokracuje dal v kodu
       processPart = 0;
       timeToRead = 0;
       processRead = false;
       if(String(state) != "?"){
         doCommand(pin, deviceType, state.toInt());
         //Serial.println("Delam command");
       }else{
         responseStateDevice(pin, deviceType);
       }
     }
    }
    if(receiveChar == '#'){
      processRead = true;
      //Serial.print(processRead);
      //Serial.println("menim process");
      //Serial.print(processRead);
      //Serial.println("platim hash");

    }
   }else{

   }
  }

   if(nacteno){
      checkDevice();
    }
}
```

```
//ROZDELUJE prichozi data
String getValue(String data, char separator, int index)
{
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```