```
/*
  პროგრამული კოდი, "NodeMCU" დაფაზე არსებული მიკროკონტროლერისთვის
*/

// ბიბლიოთეკების დამატება
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

// WiFi-ს სახელი (SSID) და პაროლი...
#ifndef STASSID
#define STASSID " WiFi SSID "
#define STAPSK  " WiFi Password "
#endif

// RGB შუქდიოდისთვის განკუთვნილი პინები
#define redPin D3
#define greenPin D4
#define bluePin D5
// დემულტიპლექსორთან დაკავშირებული შუქდიოდებისთვის...
#define redDEMUX D0
#define greenDEMUX D1
#define blueDEMUX D2

// დემულტიპლექსორის ციფრული პინები
#define DEMUX_A D6
#define DEMUX_B D7
```

#define DEMUX_C D8

// base64 სურათები

#define RGB_ICON
"
LGPC/xhBQAAACBjSFJNAAB6JgAAgIQAAPoAAACA6AAAdTAAAOpgAAA6mAAAF3CculE8AAACo1BMVEUA
AADYACfYACfYACf/AAfaACbrABT/AADbEinVCDWW2lgRZufYACfYACfYACfYACfYACfYACfYACfYACfYA
CfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYACfYAC
fYACfYACfXACfYACbYACaR3FrLMzLcJizXCjiwGVQAbfCQ3VqW1ljdyEewPZsVaOgAbvKR3FqR3FqP3FoAbfE
AbfAAbfCR3FqR3FoAbfAAbfCR3FoAbfCR3FoAbfCR3FoAbfCR3FqR3FoAbfCR3FqR3FoAbfAAbfCR3FqR3FqR
3FqR3FqR3FqR3FiQ3X4sjusAa/AAbfAAbfAAbfAAbfAAbfCR3FqR3FqR3FqR3FqR3FqR3FqR3FqR2E0BYOwA
bfAAbfAAbfAAbfAAbfAAbfAAbfDYACfYACfYACfYACfYACfXACbXACbYACfYACfYACfYACbYACXYACXgOy7gO
C3bGSrXAybXACXZBjDbDT3cDj/90UL80EL3uD7pazzhJFnmKnnoMYbsMYW83FD82kT/2kP/3Un+7Kn1psvr
O4/qM4rpNItlVcaQ3FqX21jX2kv+2UH/5Hv+/fX+9frydK/rMIieRqwSaekAbfCR3FqR3FqQ3Fqa21jK2k7w77
T5/v/5///jtNeCTrkYZ+YAbfAAbfAAbfCR3FqR3FqR3FqQ3FqR3F6k68qy8/e09PWF0PEOcu8AbfAAbfAAbfAA
bfCR3FqR3FqR3FyV6MGX7/Sa8fJtyfAHcO8AbfAAbfCR3FqR3FeU5JyZ7/Ka8PJKqvAAa+8AbfAAbfCR3FqR3F
qR3FmR3mmY68550/IVfO8AbPAAbfAAbfCR3FoAbfAAAACyg1poAAAAcnRSTlMAAAAAAAAAAAAAAAAAAAA
AEdZJaXZB0AADy/9f//9b88AAAqzc0qAAOF//+FAwARvr4RABnQ0BkAHo/y85MhHK/+/7Qghvr7jd3j/P7u8
67/tj3Z3UMCS8z5//XAvvT/+c5PAgABJnGhnGUbGmKbonQoAQCRDVnFAAAAAWJLR0QAiAUdSAAAAAd0S
U1FB+QMEA4iFRQZxmsAAAEbSURBVBjTARAB7/4AAAEMDQ4PEBESExQODQwBAAABARUWFxgZGhscHR
4fFQEBAAEgASEicnJycnJyIyQBIAEAAiUmJyhzdHV1dHYpKissAwAEBS0ud3h5ent8fX4vMAYHADEIMjN/gIG
Cg4SFhjQ1CTYACjc4OYeIiYqLjI2OOjs8CwA9Pj+PkJGSk5SVlpeYQEFCAENEmZqbnJ2en6ChoqOkRUYAR6W
mp6ipqqusra6vsLGySABJs7S1tre4ubq7vL2+v8BKAEuzs8HCw8TFxsflycrAwEwATU6zs8vMzc7P0NHS08DA
TwBQUbPU1dbX2Nna29zdwFJTAFRVVlfeWFlaW1xd315fYGEAYmNkZWZnaGlqa2xtbm9wcW45bFjvpW6I
AAAAJXRFWHRkYXRlOmNyZWF0ZQAyMDIwLTEyLTE2VDE0OjM0OjIxLTA1OjAwARKrnwAAACV0RVh0ZGF
0ZTptb2RpZnkAMjAyMC0xMi0xNlQxNDozNDoyMS0wNTowMHBPEyMAAABjdEVYdHN2Zzpjb21tZW50AC
BHZW5lcmF0b3I6IEFkb2JlIElsbHVzdHJhdG9yIDE5LjAuMCwgU1ZHIEV4cG9ydCBQbHVnLUluIC4gU1ZHIFZlc
nNpb246IDYuMDAgQnVpbGQgMCkgIM5IkAsAAAAASUVORK5CYII="

#define NODEMCU_ICON
"
goKygrKDhVNT41NT41VUtbSkVKW0uHal5eaoecg3yDnL2pqb3u4u7///8BEBAQEBEQEhQUEhkbGBsZJSIfH
yIlOCgrKCsoOFU1PjU1PjVVS1tKRUpbS4dqXl5qh5yDfIOcvampve7i7v/////AABEIACAAIAMBIgACEQEDEQH
/xABmAAEBAQEAAAAAAAAAAAAAAAAFBgQCEAACAgIBAwQDAAAAAAAAAAABAgMEAAURBjFhEjJCUSEic
QEAAgMAAAAAAAAAAAAAAAAAAIBBAURAQEBAQEBAAAAAAAAAAAAAAAECAAMSEf/aAAwDAQACEQM
RAD8AHs2ZrUrSysSWOIU9JeuIJFCoh7M+darUzXXSSsVZCS7hVV1HZfHP3lyrRJnZ8QJ6vA/sT8scnQJbnZS7hVV1HZfHP3lyrRJnZ8QJ6vA
3NJepoZGCug7smH1rM1WVZYmIKnK6jeSBLKPWccPGCpdW9yeeMD2upmpO8gUGBnPpI+ORN/VmvmLg
D1GqdBZiloRxKQHi5DLmO/QWstdlsupDycERqfcuSjpa19kjl45EPcYrF1JsUHDiN/LDFedFepRHOdZZJsRMrr6
C2Fss1mRi0iEkooP4Xz6vvNm/sxRUJImILy8BVyel6k2DjhREn8XC0S1sLIHLySOe5wOdNeqQDD1klmBV3/9k
="

#define ARDUINO_ICON
"

goKygrKDhVNT41NT41VUtbSkVKW0uHal5eaoecg3yDnL2pqb3u4u7///8BEBAQEBEQEhQUEhkbGBsZJSIfH
yIlOCgrKCsoOFU1PjU1PjVVS1tKRUpbS4dqXl5qh5yDfIOcvampve7i7v/////AABEIACAAIAMBIgACEQEDEQH
/xABiAAADAQEAAAAAAAAAAAAAAAAAAwYFAhAAAgIDAQEBAQAAAAAAAAAAwQAAgERMhMUITEBAQ
EBAAAAAAAAAAAAAAAAAAAABAxEBAAIDAQEBAAAAAAAAAAAAAQASAxExIgJB/9oADAMBAAIRAxEAPwC/
hMFhhllooAFwEQuyReGXEdXubDK29Ztj+1l1MXMC+Wp+yihJsjbTeLlqb5la57z1aPXYZVaEA5cGEXgkVgz/A
CvHW+zk4mE2jGGD2CbukV5tv6Fhf5lt7t+azaUkI3I4R2WaryTVwMpVuCy+WVc866rHAEw40ExAeIQ8Um/
CLQYQT01Hk//Z"

#define UGLIMES_ICON
"
GPC/xhBQAAACBjSFJNAAB6JgAAgIQAAPoAAACA6AAAdTAAAOpgAAA6mAAAF3CculE8AAAABmJLR0QA/
wD/AP+gvaeTAAAAB3RJTUUH5AwEERgLA74B9AAABvtJREFUSMeVVltsXEcZ/v45Zy/ei9fr7HrX6/W1Thync
RKS1A5JmyrQFkSgSIiKXoUKEpSCRHmCx/BQCfpaKixBm6hCqEgUSFGrtmlJE5JmnYtzD3Zcx3Hs9a4dx7H34v
XunpmfhzlebxyK0nk4Z86cf77/9s0/P0kpmZmIABBRZa4HMy/PQIL0L2ZmxViRwqotGkfPzc9D1xLLiBAGpUZ
nhwcmAKzrbY51hlZ03ylcwdFPklKuQq+WttEFDZ0Y//jNM/H1DQAmhma+8uzWDbvalOIq01fZZKOZq76rnd
BahSCrJBPvXHngm929e7sBnHpvaOCdK2u3xx0u8+6NtuHLURL/E33FFQaA7NxiuWh1bI5pxR1bYlZZZucWd
W6Y+e4QVaBEJV4VzXc4SwDgC9aYDmP0bFIrGD2bNJ2Gf40HAAi0PCq7Kp9EZK76sSoBAJRih8vse3zD4T8N
3ppcYGDsfOqr39/mcJqrclBt6MpTKVUdNSJiBpHthKYjgUjQXCrzbn9CSbX3xztCzXWsmMGavsvo0PqqmWre
mZNlHYoZEIIgbNOWFkuBBl99c0BJVRf1FwtlV40DbIdIKQYzCRtktQfVkdEJIwIRFXLF8SvTN4ZmZqcyi9liacmy
A8JweRxun3NNtLZlfbh1Q9Tjd1VvrPZgRUEFXQiav5kbeH/42oWUYYpISzDSFgw1+r11HofTIEBaMjtfmJ3Kp
Mfmpm/MW5a6b1Nj39e76sI+nZXP9UAH9NyxsY/+ejES8/c9srbt/qiziuzVQy+Witb4f2YGPhqZTi7s+c7GrQ9
1VKqIvUUnWSklpWLmgY+vvvzi304fGdWLel1aUkoppVJSKamkrCxqEcXMZ45ee/knbycOXWVmvW6fD/3S6
DPJ+VdeOngxMc7MVllaZanlNGJF5aoVLcnM5xPXf/vzf6QmblcAmVlopiulAJw5fj3SFtzY1yKlEgYZpn3OhSAis
KpmM4TNGRimEAZJqTb1tcY61gweu64Do9QKTWEYAsDU5EL7ujAAQ9Ctm/kjHwynxucBjsQDux/raojVVsg
2k8r8+8Or6YkFANHmwO6vdYWjfgAt68JXL6UBG5CZjX379o2NzP7rvaGLg8nJGwud3Q3NbfXjo7fe+N1xwx
C9D7W3dIbGr80dfn+4uTVYH/IS0fWR2TdePW46jN6H25vb60evzh49NNLSXl9X75lOZS9fSCVvzA9dSrtrHPU
hr/Hc0z99a/9pv9/lq3Onp3OdXeGmluCB/sT6jdGnftgbaQpEmwJbd7RmM8VjR0Z7d7WB+UB/YtO2+Peefy
ASCzQ2B7bvbJufL5w4OrZjd0cquTAyMhtvDebzpeOHR+OtQfP8ualwc90zP+pTzMMjtwynOZ3OzswV9qwLp
6YyS4UyCC6X2dEdTpycmJqcB9GtTPG+DZF0OlssWmC4a8y190dPnppMpzLCYbj87r3f7RFEr/cnBgeTpiXZ43
UCkJZSzBCUzxUNQ/zz75ctS1aKq8NhkKBcriRMQYL+8udz5bIUgjRHnA6DhMhkioYpFCAtJRyGx+e0LGWWA
YsAQBhkGUIy19bVFIBnntrc0lyXz5dA8NQ40tO5/j+c9AXcpiGKRM89+6VIgy+/WALg9Tin09k/7j9dW+fO5kv
SIK24zDAMEkSkay4rkKBiSYVC3sZY7fHEhMfrDIW9oZDX63MdS0yEwt54PBCL1YYjvhMDE4GAuzHqb4z6Aw
H3sYGJhqg/GvEvLlkg0mWBBDHIhCFkSQIQAkxUlgrAk0/09O8//etXjmzfEgNw9nxqsVB+4QfbBRGAp5/Y9Nr
rp/b95pPerU0MnBxMSku98Pw2AJZiJhICAEqKXSaZTU21Bw99dv7KzJqguyCZBQGINfp/+dKDRz4dH7k2x4y
enuieXa21fpc+B01Ntb/6xYOffDo+fG2OCL3bmh7e2er1ODT9C4qT07nbC8UrY7cff7TT3NkbT84uHnj7ksNp
5BbLFrNSnM2XXE7jG490lqUCw2GKYklm8yWhyxfD6TS+9djasqVAcBiiWJKZXMnvdZYVz+XLr7551rLUti2xn
dvjpJQiIL9klS31+7cuTKZzDlMw7DKl72hlX0rQPQAI+q8g4pU2AgSyLNXY4H3xyU0ul+Fxm8wwtUUet0lEZUZ
Pd/jLPZHCknV3ff7/g5lr3ObJyzPJmXww4Nb0JbL7IrtHy0nujQc2d4WlYkN8MQV6y9TtpZFUbvm2ABGZy00ki
BCP+N49lWwMeUxTMAP3roIBgiD64MxUW8RbAbSvTC0giG4uLL12cOizVNYQxPcMbvcsgFTcHvH97NvrI8E
axaw5bV+ZqGqkc0uWVF8UHwAMQV63qd0mIh0C2wMsswWAuKuJv8ckE9EqBGb+L428lzb9CTDsAAAAJX
RFWHRkYXRlOmNyZWF0ZQAyMDIwLTEyLTA0VDE3OjIzOjU2KzAwOjAwsWnJcAAAACV0RVh0ZGF0ZTptb2
RpZnkAMjAyMC0xMi0wNFQxNzoyMzo1NiswMDowMMA0ccwAAAAASUVORK5CYII="

#define ABGDEVZT_ICON
"

BQQEBQoHBwYIDAoMDAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8XGBYUGBIUFRT/2wBDAQMEBAU
EBQkFBQkUDQsNFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUF
BQUFBT/wAARCAAgACADASIAAhEBAxEB/8QAGQAAAgMBAAAAAAAAAAAAAAAAAABAgFBgkH/8QALRAAA
QMEAQMDAgYDAAAAAAAAAQIDBAUGERIHAAghExQiMUEkMlFhgZFCUuL/xAAZAQACAwEAAAAAAAAAA
AAAAAABBQACBgf/xAAmEQACAQMDAwQDAAAAAAAAAAABAhEDBAUABhIhMUETMlFxYcHw/9oADAM
BAAIRAxEAPwBve63uwjcHx2aHRGW6jdk1vcBfyZgMnOHXRkZzggD+T4+qH1y5+QeVrkDlx3HP9tIY9wiVNlll
geT4Q0AlCB+/joK4qpUuZeZq3X3nXXjImqfU2kboLOdEIUk/HARgef0/XqMnwL05NvObY9nUtNxz4yVuz3PV
9il0Nubq2cQtPpoSdAAFeTr9usna0Lzcty9G0jhTBZpJACg9zHk+PvXQ8vm6mwL2jhsVRV7zir1XcTwBHtUf3g
+dT1NvbkDhGtSpNIuefHEb0z+GkF5hzfYp3bXsjCsfmI+P+3WgXbF3MtcywhSa61Gp13sRUS1Mx3AWpLKvH
qI8nBB8KQfKcp+x6zKkQ+S+KOXk2zd1LDMqpPl5txHoyfVbcThBaJcQ2oZCs7IyCnCurjYtwTOIOb7cqkX3rPtZ
zalqlsJZ9VpxejwQlB10IKwn/nqXFG725dJQuYKOJBEwwmJE6piMzU37c1sTlaaJeBGqU2QRy491P6/H1qxUa
y6lxfzHddJmR9GIMh2K4rb5BCjuytI/Qo1P89R3G1Yvbt87gK5fFo0uBcdGrDS0S4E2WmOVhSw4QCclBC05B
wRqenz7l+257lKMut23Jap90MMhr03hq1NQNsIWsfIFOxwfp9j4+iB3xS7/ALBXJpNyW1U6OsjQyGwsEfu28
Nkf1npxhrmntO4uLhV5rWXiykEqRMjtBBEdDOl9fA3e/c42dtmUE/c42dtmUV/TVCvIKW4j3Qx6mfjpHQ6K5Ird6c+9wF
Evu8abT7aolFZSiJAhykyNEJWXACv47lSzknAGo6Nh2+9zDzra9KpRRJYVLajLUj8yEtn1Hiof44Gf6657bfCl18u
iVRKXRbouCHNKA6iVOeUz8TsN3FYQBnSrSXtf7YWeGYUir18q3BbVqxUhJ9oY9WmOVh5Sw4QCclBC05B
ddxQuGXgtFeKgAhQJk95JJnqZOhQwFzsPNDXKa5RqYXkrFeQ90L2M/JiOg1//2Q=="

#define QVEITISOS_ICON
"
AAAAAAIAcAigAYkZCTUQwMTAwMGFhNDAxMDAwMDI3MWEwMDAwMDYzNjAwMDA2NjNhMDAwMD
QzNDEwMDAwY2Y2MTAwMDA1Njk3MDAwMDgyOWQwMDAwMzZhMjAwMDAyOWE4MDAwMGRmZ
mIwMDAwHAJnABQ5ckxjdlJHMzRCb0NidEl0b0lZUf/bAEMABQMEBAQDBQQEBAUFBQYHDAgHBwcHDws
LCQwRDxISEQ8RERMWHBcTFBoVEREYIRgaHR0fHx8TFyIkIh4kHB4fHv/bAEMBBQUFBwYHDggIDh4UERQe
Hh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHv/AABEIACAAIAM
BIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQU
EBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc
4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKm
qsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQ
EBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQ
gUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc
3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo
6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APGvD8NtLfLFcxSS+YNkaRgks54XGPc16b8GfAmm+KvHkljc2F5Hp
+lAzalJOxVU2sQI3XqMkYPI4D+2PL7AXAuke0WRpo2Dp5aksCDwQPrivqrUPDeteCP2dNbsrbRrfVda1nTLrU
ddu70/6PGpjJZCRy77TtVRwTuYkDr9tmuMeGp2i/ee3+Z8pl+G+sVNdlueKfFnwBN4a+JEmiabbNJZagftGluT
xJEQSQD32kMPpjrmuI1fTrrTbsW93F5TtGsgXn7rdDzg9q+j9M07UPiF+y/pOpy2NnpWs+HbWOXSri2GyGa
AQr8q7uFypKEZI3p17V84Xl1c3cwkupWkdVCAsAMKOgwKnKazEPaN6rR77/l/TNsyvSXs7Ll3Wi/O1/6R3n
whtYrHRtQ8WXcYgtbG4S3a6kb5Xd8FY1X+M8EkAHgivUviF8S54NI8R/D7UvD02q22i6VBday0M6oiQugc27
YHynjGc8lwMV5/ZeJn0vw14d0k+BNL1CHSfB0vjC0e7yzC9DZ+0nbKFaPgfKQXAGBXR3HjmeXVfE0U/gDR2/
tbwIniHUWKHdczYLB5D53MGmDxRvDh/w3GsdzpTeJNBs
xdqVlTb53kBdpO5GaTGcZMfGMivD9dtjd2Nn4rsLYx6RqhZEw5fyrhABKjHA2sTlwvoa9ds/G9zL4k0O4/4V/o
q3Wi/D5dd0yVAwe2b+8p83Hkck7Dl/esO71v8AtLQH8Mw+BNN02x1rwh/wmFybllG+2jn7Sm6VlWPBYFM
biMj0r1MvzSphbU38F9e/5mmNw/t6bmn+/3nmmNw/t6Lgtz/9k="

#define WEB_ICON
"
GPC/xhBQAAAAFzUkdCAK7OHOkAAAAMUExURQAAAAAAAAAAAAAADXpN5YAAAAEdFJOUwT1nUejq
WCSAAAAsUlEQVQY02NgQADmpVEHwAzV0NAIsEBobW0oSIgz9KBzaAJI5hmjw2yQ3HRxRgeRa0AlMTMY
HdRNDzAwFsxhdDDjdWDgbIiaGrqUKYGB9WQoECQHMJhOjT7o/DUshkE1qoDRgXdpBINqE4jRFMEgzgRiN
FxhEIEwRBAMMZiU6iaoYrh2kIEOP4EGwq3gbIZaylgQw+hgCnQG2GFqQIcxTL8CcSrC8XDvwD0I9zIiECAAA
Gn6QwFLsee3AAAAAElFTkSuQmCC"

#define CIRCUIT_ICON
"
GPC/xhBQAAAAFzUkdCAK7OHOkAAAAMUExURQAAAAAAAAAAAAAADXpN5YAAAAEdFJOUwLMSIha7A
8qAAAAuElEQVQYGQXAIU5CcRgA8B+PuUEy+kXGvinBKhqpJtzcfxYDFume4EWKGQMn4AhaKN5AwLfpHX
wneA4AACqjgRons8O6d4d8mbevgfPF8+/wDcN6M60W6Ik0Q79Elh1sI7eYEMnEkUiOnkqJLOXWfjyOXK0e
XBLJDd8iXdBffkS+Nzv9z31kud/xKNIViASnXWT3h+susvtBiJQIkRLh60xiU83beoqlZaOhmttMtTUHWGMEA
wAA/AMJoTAlBaJ5RQAAAABJRU5ErkJggg=="

#define cookieName "89231509304388565377620051418977"

#define sendEnableCookieValue "893546151587282722677975252505171"

#define sendDisableCookieValue "10820127834194290893276099662506"

#define maxInputsNumber 100// მუდმივა, სადაც ინახება რიცხვი, თუ რამდენი რიცხვის შენახვა მოხდება 1 ჯერზე, რომლებიც მომხმარებლების მიერ არის გამოგზავნილი

// მუდმივები, სადაც ინახება დრო და თარიღი, რის შემდეგაც რიცხვის http-ით გამოგზავნა და მისი შუქდიოდურ წერტილოვან მატრიცაზე გატანა შეუძლებელი იქნება

#define allowedInputTime_second 0// წამი

#define allowedInputTime_minute 59// წუთი

#define allowedInputTime_hour 23// საათი

#define allowedInputDate_day 31// დღე

#define allowedInputDate_month 12// თვე

#define allowedInputDate_year 2020// წელი

// მუდმივები, სადაც ინახება დრო და თარიღი, რის შემდეგაც სერვერთან დაკავშირება შეუძლებელი იქნება (მომხმარებელი მიიღებს შესაბამის ინფორმაციას)

#define allowedConnectTime_second 30// წამი

#define allowedConnectTime_minute 59// წუთი

#define allowedConnectTime_hour 23// საათი

#define allowedConnectDate_day 31// დღე

#define allowedConnectDate_month 12// თვე

#define allowedConnectDate_year 2020// წელი

const char* ssid = STASSID;

const char* password = STAPSK;

unsigned long t;// უნიშნო, მთელი რიცხვების ტიპის (unsigned long) ცვლადი, რომელიც განკუთვნილია დროის შესანახად...

byte countRGBmodes;// ცვლადი, რომლის სიდიდის მიხედვით ხდება RGB შუქდიოდზე ფერის ცვლილება

boolean WEB_RGB_DISABLE = 0;// თუ ინახება ლოგიკური 0, ვებ-გვერდზე შესვლისას RGB შუქდიოდი ფერს შეიცვლის.. ლოგიკური 1 - გაიზრდება "countVisits" ცვლადის მნიშვნელობა

unsigned long countVisits;// ცვლადი, სადაც ინახება ვებ გვერდზე შესვლების რაოდენობა მანამ, სანამ "WEB_RGB_DISABLE" ცვლადი 1-ის ტოლია

String userInput;// ცვლადი, სადაც ინახება მომხმარებლის მიერ გამოგზავნილი სიმბოლო(ები)

String inputNumber;// ცვლადი, სადაც ინახება "Arduino UNO R3"-თან გასაგზავნი რიცხვი...

boolean whiteColourEnable;// ცვლადი, სადაც ინახება ლოგიკური 0 ან ლოგიკური 1..

String inputNumbers[maxInputsNumber];// მასივი, რომელიც განკუთვნილია მომხმარებლის მიერ გამოგზავნილი ციფრების შესანახად

unsigned int countInputs;// უნიშნო მთელი რიცხვების ტიპის ცვლადი, რომელიც განკუთვნილია მიღებული რიცხვების რაოდენობის შესანახად

String received;// ცვლადი, რომელიც ინახავს UART კავშირის საშუალებით მიღებულ წინადადებებს

boolean inputDisabled;

boolean connectDisabled;

unsigned long t2[8];

byte countRGBmodes2[8];// ცვლადი, რომლის სიდიდის მიხედვით ხდება RGB შუქდიოდზე ფერის ცვლილება (2)

boolean whiteColourEnable2[8];// ცვლადი, სადაც ინახება ლოგიკური 0 ან ლოგიკური 1.. (2)

int RGB_COLOUR_TRANSITION_VALUE[8];

unsigned int RGB_DEMUX_ALL_TIME;

boolean RGB_DEMUX_ALL_ENABLED;

unsigned int softRGBtime;

boolean softRGBenabled;

byte GLOBAL_RGB_INDEX;

boolean RGB_DEMUX_softMix_ENABLED;

int RGB_COLOURS[8][3];

unsigned long t3;

boolean DEMUX_RGB_RANDOM_ENABLED;

int redValue;

int greenValue;

int blueValue;

boolean RGB_DIRECTION;

boolean BIDIRECTIONAL_RANDOM_RGB_DEMUX_ENABLED;

boolean doubleRGB;

byte GLOBAL_RGB_INDEX_2;

unsigned long t4;

unsigned long t5;

byte DEMUX_RGB_MODE;

byte RGB_TIME;

byte RGB_SOFT_MIX_TIME;

boolean RGB4enabled;

unsigned int RGB_MODE_CHANGE_TIME;

unsigned long t6;

unsigned long t7;

```
// დროის შესაანახი ცვლადები
byte second;
byte minute;
byte hour;

// თარიღის შესანახი ცვლადები
byte day;
byte month;
int year;

ESP8266WebServer server(80);// მე-80 (http) პორტის დაყენება

String getConnectDateTimeString()  {
  return " 🕒" + String(allowedConnectTime_hour) + ":" + String(allowedConnectTime_minute) + ":" + String(allowedConnectTime_second) + " 📅" + String(allowedConnectDate_year) + "-" + String(allowedConnectDate_month) + "-" + String(allowedConnectDate_day);
}

// ფუნქცია, რომელიც აბრუნებს HTML კოდს
String sendHTML()  {
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta charset=\"UTF-8\" name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=yes\">\n";
  ptr += "<title>2020-2021 წელი</title>\n";
  ptr += "<link rel=\"icon\" href=\"" + String(RGB_ICON) + "\"></link>\n";
  ptr += "<span id=\"images\"></span>\n";
  ptr += "<link rel=\"stylesheet\" href=\"/page_style\">\n";
  ptr += "</head>\n";
```

```
  ptr += "<h1>ინტერნეტ სერვერი - Web Server</h1>\n";

  ptr += "<h1><span style=\"color:#ffffff;background-color:#8040ff;padding:4px;border-radius:8px;padding: 2px;\">ESP8266 (NodeMCU)</span></h1>\n";

  ptr += "<span style=\"display:inline-block;padding:4px;border-radius:8px;padding: 2px;border 2px solid #808080;border-style:double;border-radius:8px\">🕒დრო და 📅თარიღი (RTC): <span id=\"rtc_datetime\"></span></span><br><br>\n";

  ptr += "<span>🛈 სერვერთან დაკავშირება შესაძლებელია" + getConnectDateTimeString() + " - მდე</span><br><br>\n";

  ptr += "<div style=\"animation: leftToRight 0.512s;\">\n";

  ptr += "<div id=\"RGB\"><p style=\"background-color:#ffffff80;border-radius:4px; animation: leftToRight 0.512s;\"><b>RGB შუქდიოდის ფერი შეიცვალა 🎨➡</b></p></div>\n";

  ptr += "</div>\n";

  ptr += "<p id=\"info\"></p>\n";

  ptr += "<strong style=\"color:red;\">ერთმა მომხმარებელმა რიცხვი უნდა გააგზავნოს მაქსიმუმ ერთხელ!</strong>\n";

  ptr += "<br><span>🛈 რიცხვის გაგზავნა შესაძლებელია 🕒" + String(allowedInputTime_hour) + ":" + String(allowedInputTime_minute) + ":" + String(allowedInputTime_second) + " 📅" + String(allowedInputDate_year) + "-" + String(allowedInputDate_month) + "-" + String(allowedInputDate_day) + " - მდე</span>\n";

  ptr += "<br><input type=\"tel\" id=\"inputNumber\" minlength=\"1\" maxlength=\"10\" placeholder=\"ჩაწერეთ ციფრ(ებ)ი...\"><button id=\"sendButton\" disabled>→</button>\n";

  ptr += "<br><span style=\"font-size:16px; font-style:italic;\">ჩაწერილი ციფრ(ებ)ი გავა შუქდიოდურ წერტილოვან მატრიცაზე</span>\n";

  ptr += "<br><span style=\"font-size:16px; color:red;\">რიცხვი უნდა იყოს მთელი, არაუარყოფითი და მისი ციფრების რაოდენობა 10-ს არ უნდა აღემატებოდეს.. უნდა შედგებოდეს მხოლოდ ციფრებისგან და სხვა სიმბოლოს არ უნდა შეიცავდეს..</span>\n";

  ptr += "<br><br>\n";

  ptr += "<p style=\"width: 100%;height: 32px;background: linear-gradient(45deg, #ff0000, #ffff00, #00ff00, #00ffff, #0000ff, #ff00ff, #ffffff);\"></p>\n";

  ptr += "<body>\n";

  ptr += "<div id=\"fb-root\"></div><script async defer crossorigin=\"anonymous\" src=\"https://connect.facebook.net/ka_GE/sdk.js#xfbml=1&version=v9.0\" nonce=\"i3R5OLsw\"></script>\n";
```

```
  ptr += "</body>\n";

  ptr += "<br><br><a href=\"http://uglimes.ge/\" target=\"_blank\" style=\"background-color:rgb(128,
50, 195);width:250px;height=56px;display:inline-block;padding:4px;border-radius:4px;\"><img
src=\"http://uglimes.ge/img/0.%20%E1%83%9B%E1%83%97%E1%83%90%E1%83%95%E1%83%90%E1
%83%A0%E1%83%98/01_01_0000s_0000_Logo.png\" /></a>\n";

  ptr += "<div class=\"fb-page\" data-href=\"https://www.facebook.com/UGLIMES\" data-tabs=\"\"
data-width=\"\" data-height=\"\" data-small-header=\"true\" data-adapt-container-width=\"true\"
data-hide-cover=\"false\" data-show-facepile=\"true\"><blockquote
cite=\"https://www.facebook.com/UGLIMES\" class=\"fb-xfbml-parse-ignore\"><a
href=\"https://www.facebook.com/UGLIMES\">UG-Limes აკადემია</a></blockquote></div>\n";

  ptr += "<div class=\"fb-page\" data-href=\"https://www.facebook.com/pedestrian.sos\" data-tabs=\"\"
data-width=\"\" data-height=\"\" data-small-header=\"true\" data-adapt-container-width=\"true\"
data-hide-cover=\"false\" data-show-facepile=\"true\"><blockquote
cite=\"https://www.facebook.com/pedestrian.sos\" class=\"fb-xfbml-parse-ignore\"><a
href=\"https://www.facebook.com/pedestrian.sos\">ქვეითი SOS</a></blockquote></div>\n";

  ptr += "<br><br><p style=\"width: 100%;height: 32px;background: linear-gradient(#aaaaaa,
#555555);\"></p>\n";

  ptr += "<body onload=\"getJavaScript()\">\n";

  ptr += "<script>\n";

  ptr += "function loadJS(url){let script = document.createElement('script');script.src =
url;document.body.appendChild(script);}\n";

  ptr += "function getJavaScript(){loadJS(\"/get_script\");}\n";

  ptr += "</script>\n";

  return ptr;

}


// ფუნქცია, რომელიც აბრუნებს CSS კოდს

String sendCSS()  {

  String ptr = "html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align:
center;}\n";

  ptr += "#RGB {background: linear-gradient(to right, ";

  if (countRGBmodes == 1)  {

    ptr += "#ff";

    if (whiteColourEnable)  {
```

```
    ptr += "ffff";
   }
   else  {
    ptr += "0000";
   }
   ptr += ", #ffff00";
 }
 else if (countRGBmodes == 2)  {
   ptr += "#ffff00, #00ff00";
 }
 else if (countRGBmodes == 3)  {
   ptr += "#00ff00, #00ffff";
 }
 else if (countRGBmodes == 4)  {
   ptr += "#00ffff, #0000ff";
 }
 else if (countRGBmodes == 5)  {
   ptr += "#0000ff, #ff00ff";
 }
 else if (countRGBmodes == 0 || countRGBmodes == 6)  {
   ptr += "#ff00ff, #ff0000";
 }
 else if (countRGBmodes == 7)  {
   ptr += "#ff0000, #ffffff";
 }
 ptr += "); font-size: 25px; padding: 16px;border-radius:8px;}\n";
 ptr += "@keyframes leftToRight{from{width:0%; font-size: 0px;} to{width:100%; font-size: 25px;}}\n";
```

```
  ptr += "#inputNumber  {width: 25%;padding: 6px 6px;margin: 6px 0;display: inline-block;border: 1px solid #ccc;border-radius: 4px;box-sizing: border-box;} #inputNumber:disabled{cursor: not-allowed;} #inputNumber:disabled::placeholder{color: red; text-decoration-line: line-through;}\n";

  ptr += "#sendButton{ border: none; color: white; padding: 2px 16px; text-align: center; text-decoration: none; display: inline-block; font-size: 25px; margin: 4px 2px; transition-duration: 0.4s; cursor: pointer; background-color: white; color: black; border-radius: 8px; background-color: rgba(192, 192, 192, 0.25); } #sendButton:disabled{ color: #c0c0c0; cursor: not-allowed; } #sendButton:hover{ background-color: rgba(224, 224, 224, 0.1); }\n";

  return ptr;

}


// ფუნქცია, რომელიც აბრუნებს JavaScript კოდს

String sendScript() {

  String ptr = "var getImages = new XMLHttpRequest();\n";

  ptr += "getImages.open(\"GET\", \"/images\");\n";

  ptr += "getImages.send();\n";

  ptr += "getImages.onreadystatechange = function(){if(this.responseText != \"\"){document.getElementById(\"images\").innerHTML = this.responseText;}};\n";

  ptr += "inputNumberElement = document.getElementById(\"inputNumber\");\n";

  ptr += "sendButtonElement = document.getElementById(\"sendButton\");\n";

  ptr += "function setInputFilter(textbox, inputFilter) { [\"input\", \"keydown\", \"keyup\", \"mousedown\", \"mouseup\", \"select\", \"contextmenu\", \"drop\"].forEach(function(event) { textbox.addEventListener(event, function() { if (inputFilter(this.value)) { this.oldValue = this.value; this.oldSelectionStart = this.selectionStart; this.oldSelectionEnd = this.selectionEnd; } else if (this.hasOwnProperty(\"oldValue\")) { this.value = this.oldValue; this.setSelectionRange(this.oldSelectionStart, this.oldSelectionEnd); } else { this.value = \"\"; } }); }); }\n";

  ptr += "setInputFilter(inputNumberElement, function(value) {return /^\\d*$/.test(value); });\n";

  ptr += "inputNumberElement.oninput = function(){sendButtonElement.disabled = (inputNumberElement.value.length == 0);};\n";

  ptr += "var number;\n";

  ptr += "function setCookie(cname, cvalue, exdays) { var d = new Date(); d.setTime(d.getTime() + (exdays*24*60*60*1000)); var expires = \"expires=\"+ d.toUTCString(); document.cookie = cname + \"=\" + cvalue + \";\" + expires + \";path=/\"; }\n";
```

```
  ptr += "function getCookie(cname) { var name = cname + \"=\"; var decodedCookie =
decodeURIComponent(document.cookie); var ca = decodedCookie.split(';'); for(var i = 0; i <ca.length;
i++) { var c = ca[i]; while (c.charAt(0) == ' ') { c = c.substring(1); } if (c.indexOf(name) == 0) { return
c.substring(name.length, c.length); } } return \"\"; }\n";

  ptr += "function disableNumberSend(){inputNumberElement.disabled=1;}\n";

  ptr += "if(document.cookie == ''){\n";

  ptr += "setCookie(\"" + String(cookieName) + "\", \"" + String(sendEnableCookieValue) + "\", 1);\n";

  ptr += "}\n";

  ptr += "if(getCookie(\"" + String(cookieName) + "\") == \"" + String(sendEnableCookieValue) + "\"){\n";

  ptr += "sendButtonElement.onclick = function(){\n";

  ptr += "setCookie(\"" + String(cookieName) + "\", \"" + String(sendDisableCookieValue) + "\", 1);\n";

  ptr += "disableNumberSend();\n";

  ptr += "document.getElementById(\"info\").innerHTML = \"გთხოვთ დაიცადოთ...\";\n";

  ptr += "var ajax = new XMLHttpRequest();\n";

  ptr += "number = inputNumberElement.value;\n";

  ptr += "sendButtonElement.disabled = 1;\n";

  ptr += "inputNumberElement.value = '';\n";

  ptr += "ajax.open(\"GET\", \"/get_number?num=\" + number, true);\n";

  ptr += "ajax.send();\n";

  ptr += "ajax.onreadystatechange = function(){\n";

  ptr += "if(this.readyState == 4 && this.status == 200){\n";

  ptr += "if(this.responseText == \"1\"){\n";

  ptr += "document.getElementById(\"info\").innerHTML = \"<p style=background-
color:#008000;color:#ffffff>გაგზავნილი ციფრ(ებ)ი (\" + number + \") გამოისახება
მატრიცაზე...</p>\";\n";

  ptr += "}else if(this.responseText == \"0000\"){document.getElementById(\"info\").innerHTML =
'';disableNumberSend();}\n";

  ptr += "}\n";

  ptr += "};\n";

  ptr += "};\n";

  ptr += "}else{disableNumberSend();}\n";
```

```
  ptr += "function getDateTime(){var ajax2 = new XMLHttpRequest(); ajax2.open(\"GET\", \"/get_datetime\", true); ajax2.send(); ajax2.onreadystatechange = function(){if(this.readyState == 4 && this.status == 200){document.getElementById(\"rtc_datetime\").innerHTML = this.responseText;}};}\n";

  ptr += "getDateTime();\n";

  ptr += "setInterval(getDateTime, 1000);\n";

  if (inputDisabled)  {

    ptr += "disableNumberSend();\n";

  }

  return ptr;

}


String datetime_html()  {

  String ptr = "<!DOCTYPE html> <html>\n";

  ptr += "<head><meta charset=\"UTF-8\" name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=yes\">\n";

  ptr += "<title>RTC 🕘 & 📅</title>\n";

  ptr += "<center>\n";

  ptr += "RTC - დრო და თარიღი: <span id=\"rtc_datetime\"></span><br><br>\n";

  ptr += "<input type=\"time\" step=\"1\" id=\"timeinput\">\n";

  ptr += "<input type=\"date\" id=\"dateinput\">\n";

  ptr += "<button id=\"sendbutton\">გაგზავნა</button>\n";

  ptr += "<br><br><button id=\"automaticdatetime\">დროს და თარიღის ავტომატურად დაყენება</button>\n";

  ptr += "</center>\n";

  ptr += "<script>\n";

  ptr += "function time_second(time){if(time.indexOf(':', time.indexOf(':') + 1) == -1){time += \":0\";} return time;}\n";

  ptr += "function getDateTime(){var ajax2 = new XMLHttpRequest(); ajax2.open(\"GET\", \"/get_datetime\", true); ajax2.send(); ajax2.onreadystatechange = function(){if(this.readyState == 4 && this.status == 200){document.getElementById(\"rtc_datetime\").innerHTML = this.responseText;}};}\n";

  ptr += "getDateTime();\n";
```

```
  ptr += "setInterval(getDateTime, 1000);\n";

  ptr += "function sendDateTime(DateTime) {var ajax = new XMLHttpRequest(); ajax.open(\"GET\", \"/set_datetime?datetime=\" + DateTime, true); ajax.send(); ajax.onreadystatechange = function(){if(this.readyState == 4 && this.status == 200 && this.responseText == \"1\"){alert(\"დრო და თარიღი გაგზავნილია: \" + DateTime.replace(\"%20\", \" \"));}};}\n";

  ptr += "document.getElementById(\"sendbutton\").onclick = function(){sendDateTime(time_second(document.getElementById(\"timeinput\").value) + \"%20\" + document.getElementById(\"dateinput\").value);};\n";

  ptr += "document.getElementById(\"automaticdatetime\").onclick = function(){var t = new Date(); sendDateTime(t.getHours() + ':' + t.getMinutes() + ':' + t.getSeconds() + \"%20\" + t.getFullYear() + '-' + (t.getMonth() + 1) + '-' + t.getDate());};\n";

  ptr += "</script>\n";

  ptr += "</html>\n";

  return ptr;

}


// სურათების შესაბამისი base64 კოდის დაბრუნების ფუნქცია

void webImages()  {

  if (!serverConnect_checkTime()) {

    return;

  }

  server.send(200, "text/html", "<img src=\"" + String(WEB_ICON) + "\">\t<img src=\"" + String(CIRCUIT_ICON) + "\">\t<img src=\"" + String(NODEMCU_ICON) + "\">\t<img src=\"" + String(ARDUINO_ICON) + "\">\t<img src=\"" + String(UGLIMES_ICON) + "\">\t<img src=\"" + String(ABGDEVZT_ICON) + "\">\t<img src=\"" + String(QVEITISOS_ICON) + "\">\n");

}


// ფუნქცია, რომელიც მთავარი ვებ გვერდის გახსნისას სრულდება

void handleRoot() {

  if (!serverConnect_checkTime()) {

    return;

  }
```

```
  if (!WEB_RGB_DISABLE) {
    WEB_RGB_DISABLE = 1;
    RGBcoloursTransition(500);// თუ "WEB_RGB_DISABLE" ცვლადი 0-ის ტოლია (შუქდიოდზე ფერის ცვლილება არ ხდება), მოხდება შუქდიოდზე ფერის ცვლილება
    if (countVisits)  {// თუ "countVisits" ცვლადი არ უდრის 0-ს
      // "RGBcoloursTransition" ფუნქციის გამოძახება (შუქდიოდის ფერის ცვლილება) მოხდება იმდენჯერ, რა რიცხვიც არის შენახული "countVisits" ცვლადში
      for (byte n = 0; n < countVisits; n++) {
        if (connectDisabled)  {
          return;
        }
        RGBcoloursTransition(500);
      }
      countVisits = 0;// "countVisits" ცვლადის განულება
    }
    WEB_RGB_DISABLE = 0;// "WEB_RGB_DISABLE" ცვლადი ხდება 0-ის ტოლი
  }
  else  {
    countVisits++;// წინააღმდეგ შემთხვევაში გაიზრდება "countVisits" ცვლადის სიდიდე
  }
  server.send(200, "text/html", sendHTML());// html კოდის გაგზავნა
}


// ფუნქცია, რომელიც ბმულიდან (ლინკიდან) ამოიკაითხავს გამოგზავნილ დროს და თარიღს, რომელიც შემდეგ UART-ით გადაეგზავნება "ATmega328P" მიკროკონტროლერს
void handleSetDateTime()  {
  if (server.args() == 1 && server.argName(0) == "datetime")  {// თუ http-ით მიღებულია მხოლოდ 1 პარამეტრი, რომლის სახელია "datetime"
    Serial.print('t' + server.arg(0));// UART-ით იგზავნება მიღებული ინფორმაცია, "Arduino"-ს დაფაზე არსებულ მიკროკონტროლერთან
```

```
   server.send(200, "text/plain", "1");// მოწყობილობასთან, რომელიც NodeMCU-ს დაუკავშირდა http-ით, იგზავნება String "1", იმის ნიშნად რომ მონაცემები მიღებულია
  }
  else  {// თუ წინა პირობა მცდარია
   server.send(200, "text/plain", "0");// მოწყობილობასთან, რომელიც NodeMCU-ს დაუკავშირდა http-ით, იგზავნება String "0", იმის ნიშნად რომ მონაცემები არ არის (სწორად) გამოგზავნილი
  }
}

// ფუნქცია, რომელიც http-ით დაკავშირებულ მოწყობილობას უგზავნის დროს და თარიღს, რომელიც მიღებულია Arduino-დან UART კავშირით.. Arduino აღნიშნულ დროს და თარიღს იღებს საათის (RTC) მოდულიდან, I2C კავშირის საშუალებით..
void handleGetDateTime()  {
  server.send(200, "text/plain", String(hour) + ':' + String(minute) + ':' + String(second) + ' ' + String(year) + '-' + String(month) + '-' + String(day));
}

void handleDateTime() {
  server.send(200, "text/html", datetime_html());
}

void handlePageStyle()  {
  if (!serverConnect_checkTime()) {
    return;
  }
  server.send(200, "text/css", sendCSS());// css კოდის გაგზავნა
}

void handlePageScript() {
  if (!serverConnect_checkTime()) {
```

```
  return;
 }
 server.send(200, "text/script", sendScript());// script კოდის გაგზავნა
}

// ბმულიდან (ლინკიდან) რიცხვების ამოკითხვის და შესაბამის ცვლადში შენახვის ფუნქცია...
void handleGetNumber()  {
 if (!serverConnect_checkTime()) {
  return;
 }
 if (inputDisabled)  {
  server.send(200, "text/plain", "0000");
  return;
 }
 if (server.args() == 1 && server.argName(0) == "num")  {
  userInput = server.arg(0);
  userInput = makeStringDigitsOnly(userInput);
  if (userInput.length())  {
   inputNumber = userInput.substring(0, 10);
   saveInput(inputNumber);
   server.send(200, "text/plain", "1");
  }
  else  {
   server.send(200, "text/plain", "0");
  }
 }
 else  {
  server.send(200, "text/plain", "-1");
 }
```

```
}

// ფუნქცია, რომელიც სრულდება ისეთი ვებ მისამართის გახსნისას, რომელიც არ არსებობს
void handleNotFound() {
  if (!serverConnect_checkTime()) {
    return;
  }
  String message = "Error 404 Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
}

//// ფუნქცია, რომელსაც პარამეტრად გადაეცემა String და თუ ის მხოლოდ ციფრებისგან შედგება, აბრუნებს ლოგიკურ 1-ს.. წინააღმდეგ შემთხვევაში - ლოგიკურ 0-ს..
//boolean isNumber(String input)  {
//  for (byte n = 0; n < input.length(); n++) {
//    if (!isDigit(input.charAt(n)))  {
//      return 0;
//    }
//  }
```

```
//  return 1;
//}


// ფუნქცია, რომელიც მისთვის პარამეტრად გადაცემული String-დან შლის ყველა სიმბოლოს, ციფრების გარდა და აბრუნებს მიღებულ შედეგს
String makeStringDigitsOnly(String input) {
  for (byte n = 0; n < input.length(); n++) {
    if (!isDigit(input.charAt(n)))  {
      input.remove(n, 1);
    }
  }
  return input;
}


// მომხმარებლების მიერ გამოგზავნილი რიცხვების შენახვის ფუნქცია
void saveInput(String input)  {
  if (countInputs < maxInputsNumber)  {
    countInputs++;
    String tempArray[maxInputsNumber];
    // მასივში რიცხვების არსებობის შემთხვევაში, ხდება მათი გადანაცვლება მომდევნო ელემენტში (მაგალითად: 0 ინდექსის მქონე ელემენტში არსებული რიცხვი გადადის ინდექს 1-ში, ინდექს 1-ში არსებული რიცხვი - ინდექს 2-ში, ინდექს 2-ში - ინდექს 3-ში...)
    for (byte i = 0; i < countInputs; i++) {
      tempArray[i + 1] = inputNumbers[i];
    }
    tempArray[0] = input;// ახალი მიღებული რიცხვი ინახება მასივის პირველ ელემენტში (ინდექსი 0)
    // "tempArray" ლოკალურ მასივში არსებული ელემენტების გადატანა, "inputNumbers" გლობალურ მასივში
    for (byte i = 0; i < countInputs; i++) {
```

```
    inputNumbers[i] = tempArray[i];
   }
 }
}
```

// ფუნქცია, რომელსაც პარამეტრებად გადაეცემა დრო და თარიღი, რის მიხედვითაც აბრუნებს შესაბამის რიცხვს (დაახლოებით დროს წამებში)

```
//unsigned long dateTime2number2000(byte local_second, byte local_minute, byte local_hour, byte local_day, byte local_month, int local_year) {
//  local_year -= 2000;// ათვლა იწყება 2000 წლიდან...
//  return (local_second + local_minute * 60 + local_hour * 3600 + local_day * 86400 + local_month * 86400 * 30 + local_year * 86400 * 365);
//}


//boolean isInputDisabled_time() {
//  return dateTime2number2000(second, minute, hour, day, month, year) >= dateTime2number2000(allowedInputTime_second, allowedInputTime_minute, allowedInputTime_hour, allowedInputDate_day, allowedInputDate_month, allowedInputDate_year);
//}
//
//boolean isConnectDisabled_time() {
//  return dateTime2number2000(second, minute, hour, day, month, year) >= dateTime2number2000(allowedConnectTime_second, allowedConnectTime_minute, allowedConnectTime_hour, allowedConnectDate_day, allowedConnectDate_month, allowedConnectDate_year);
//}
```

// ფუნქცია, რომელიც წყვეტს სერვერთან კავშირს და აბრუნებს შესაბამის გაფრთხილებას, თუ მიმდინარე დრო და თარიღი მეტია კავშირის დასრულების დროზე და თარიღზე

```
boolean serverConnect_checkTime()  {
  if (connectDisabled)  {
```

```
   server.send(200, "text/html", "<!DOCTYPE html> <html><head><meta charset=\"UTF-8\" name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=yes\"><title>2020-2021 წელი</title><center><strong>სერვერთან დაკავშირება შესაძლებელი იყო" + getConnectDateTimeString() + " - მდე</strong><br><br>დრო და თარიღი (RTC): <span id=\"rtc_datetime\"></span></center><script>function getDateTime(){var ajax2 = new XMLHttpRequest(); ajax2.open(\"GET\", \"/get_datetime\", true); ajax2.send(); ajax2.onreadystatechange = function(){if(this.readyState == 4 && this.status == 200){document.getElementById(\"rtc_datetime\").innerHTML = this.responseText;}};}getDateTime();setInterval(getDateTime, 1000);</script>");
    return 0;
  }
  return 1;
}

// RGB შუქდიოდის ფერის დაყენების ფუნქცია
void RGB(int red, int green, int blue)  {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}

// დემულტიპლექსორთან დაკავშირებული RGB შუქდიოდების ფერის დაყენების ფუნქცია
void DEMUX_RGB_COLOUR(int red, int green, int blue) {
  analogWrite(redDEMUX, red);
  analogWrite(greenDEMUX, green);
  analogWrite(blueDEMUX, blue);
}

// დემულტიპლექსორთან დაკავშირებული RGB შუქდიოდებიდან, ერთ-ერთი შუქდიოდის ანთების ფუნქცია (ანთებული შუქდიოდის ინდექსის მითითება)
void DEMUX_RGB_INDEX(byte RGBindex)  {
  digitalWrite(DEMUX_A, RGBindex & 1);
```

```
  digitalWrite(DEMUX_B, RGBindex & 2);
  digitalWrite(DEMUX_C, RGBindex & 4);
}

void DEMUX_RGB(int red_value, int green_value, int blue_value, byte index)  {
  DEMUX_RGB_COLOUR(0, 0, 0);// RGB შუქდიოდის ჩაქრობა, რადგან არ მოხდეს უკვე ანთებული შუქდიოდის ახალი ფერით განათება...
  DEMUX_RGB_INDEX(index);
  DEMUX_RGB_COLOUR(red_value, green_value, blue_value);
}

// დაყოვნების ფუნქცია
void delayFunction(unsigned int delayTime)  {
  /*
     პროგრამის მსვლელობის დაყოვნება, ამ ფუნციისთვის პარამეტრად გამდოცემული დროით
     თუმცა დაყოვნების პერიოდში, "loopFunction" ფუნქცია აგრძელებს მუშაობას
  */
  for (t = millis(); millis() - t <= delayTime; loopFunction());
}

// დაყოვნების ფუნქცია (მიკროწამები)
void delayMicrosecondsFunction(unsigned int delayTime)  {
  /*
     პროგრამის მსვლელობის დაყოვნება, ამ ფუნციისთვის პარამეტრად გამდოცემული დროით
     თუმცა დაყოვნების პერიოდში, "loopFunction" ფუნქცია აგრძელებს მუშაობას
  */
  for (t = micros(); micros() - t <= delayTime; loopFunction());
```

```
}

// RGB შუქდიოდის ფერის ნელი ცვლილების ფუნქცია...
void RGBcoloursTransition(unsigned long delayTime) {
  for (int value = 0; value < 1024; value++) {
    if (connectDisabled && WEB_RGB_DISABLE)  {
      WEB_RGB_DISABLE = 0;
      return;
    }
    // წითელი-ყვითელი
    if (countRGBmodes == 0)  {
      RGB(1023, value, 0);
    }
    // ყვითელი-მწვანე
    else if (countRGBmodes == 1) {
      RGB(1023 - value, 1023, 0);
    }
    // მწვანე-ცისფერი
    else if (countRGBmodes == 2) {
      RGB(0, 1023, value);
    }
    // ცისფერი-ლურჯი
    else if (countRGBmodes == 3) {
      RGB(0, 1023 - value, 1023);
    }
    // ლურჯი-იისფერი
    else if (countRGBmodes == 4) {
      RGB(value, 0, 1023);
```

```
  }
  // იისფერი-წითელი
  else if (countRGBmodes == 5) {
    RGB(1023, 0, 1023 - value);
  }
  else if (whiteColourEnable) {
    // წითელი-თეთრი
    if (countRGBmodes == 6)  {
      RGB(1023, value, value);
    }
    // თეთრი-ყვითელი
    else  {
      RGB(1023, 1023, 1023 - value);
    }
  }
  delayMicrosecondsFunction(delayTime);// "delayTime" პარამეტრში არსებული რიცხვის მიხედვით (მიკროწამები) დაყოვნება "delayMicrosecondsFunction" ფუნქციის გამოყენებით...
 }
 ++countRGBmodes;// იზრდება "countRGBmodes" ცლადის სიდიდე 1 ერთეულით
 if (countRGBmodes == 6)  {
  whiteColourEnable = !whiteColourEnable;// "whiteColourEnable" ცვლადის ლოგიკური შებრუნება
 }
 if ((countRGBmodes > 5) && !whiteColourEnable)  {
  countRGBmodes = 0;
 }
 else if (countRGBmodes > 7)  {
  countRGBmodes = 1;
 }
```

```
}


void DEMUX_RGB_SETUP(byte LOCAL_RGB_INDEX)  {
  if (countRGBmodes2[LOCAL_RGB_INDEX] == 6)  {
    whiteColourEnable2[LOCAL_RGB_INDEX] = !whiteColourEnable2[LOCAL_RGB_INDEX];//
"whiteColourEnable2" ცვლადის ლოგიკური შებრუნება
  }
  if ((countRGBmodes2[LOCAL_RGB_INDEX] > 5) && !whiteColourEnable2[LOCAL_RGB_INDEX])  {
    countRGBmodes2[LOCAL_RGB_INDEX] = 0;
  }
  else if (countRGBmodes2[LOCAL_RGB_INDEX] > 7)  {
    countRGBmodes2[LOCAL_RGB_INDEX] = 1;
  }
}


// დემულტიპლექსორთან დაკავშირებული 8 ცალი RGB შუქდიოდის ფერის ნელი
ცვლილების ფუნქცია...
void RGBcoloursTransition_DEMUX(unsigned long delayTime2, byte RGB_INDEX) {
  if (micros() - t2[RGB_INDEX] >= delayTime2) {
    // წითელი-ყვითელი
    if (countRGBmodes2[RGB_INDEX] == 0)  {
      DEMUX_RGB_COLOUR(1023, RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX], 0);
    }
    // ყვითელი-მწვანე
    else if (countRGBmodes2[RGB_INDEX] == 1) {
      DEMUX_RGB_COLOUR(1023 - RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX], 1023, 0);
    }
    // მწვანე-ცისფერი
    else if (countRGBmodes2[RGB_INDEX] == 2) {
```

```
  DEMUX_RGB_COLOUR(0, 1023, RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX]);
 }
 // ცისფერი-ლურჯი
 else if (countRGBmodes2[RGB_INDEX] == 3) {
  DEMUX_RGB_COLOUR(0, 1023 - RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX], 1023);
 }
 // ლურჯი-იისფერი
 else if (countRGBmodes2[RGB_INDEX] == 4) {
  DEMUX_RGB_COLOUR(RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX], 0, 1023);
 }
 // იისფერი-წითელი
 else if (countRGBmodes2[RGB_INDEX] == 5) {
  DEMUX_RGB_COLOUR(1023, 0, 1023 - RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX]);
 }
 else if (whiteColourEnable2[RGB_INDEX]) {
  // წითელი-თეთრი
  if (countRGBmodes2[RGB_INDEX] == 6)  {
    DEMUX_RGB_COLOUR(1023, RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX],
RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX]);
  }
  // თეთრი-ყვითელი
  else  {
    DEMUX_RGB_COLOUR(1023, 1023, 1023 - RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX]);
  }
 }
 if (++RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX] > 1023)  {
  RGB_COLOUR_TRANSITION_VALUE[RGB_INDEX] = 0;
  ++countRGBmodes2[RGB_INDEX];// იზრდება "countRGBmodes2" ცლადის სიდიდე 1
ერთეულით
```

```
    DEMUX_RGB_SETUP(RGB_INDEX);
   }
   t2[RGB_INDEX] = micros();
  }
  DEMUX_RGB_INDEX(RGB_INDEX);
}

void multiplexRGB() {
  if (++GLOBAL_RGB_INDEX > 7)  {
   GLOBAL_RGB_INDEX = 0;
  }
}

void DEMUX_RGB_RANDOM(byte enabled) {
  if (millis() - t3 >= RGB_TIME)  {
   RGB(random(0, 1024), random(0, 1024), random(0, 1024));
   for (byte index1 = 0; index1 < 8; index1++) {
    for (byte index2 = 0; index2 < 3; index2++) {
      RGB_COLOURS[index1][index2] = random(0, 1024);
    }
   }
   t3 = millis();
  }
  if (enabled & (1 << GLOBAL_RGB_INDEX))  {
   DEMUX_RGB(RGB_COLOURS[GLOBAL_RGB_INDEX][0], RGB_COLOURS[GLOBAL_RGB_INDEX][1],
RGB_COLOURS[GLOBAL_RGB_INDEX][2], GLOBAL_RGB_INDEX);
  }
}
```

```
void changeRGBindex() {
  if (millis() - t4 >= RGB_TIME)  {
    if (RGB_DIRECTION)  {
      GLOBAL_RGB_INDEX_2++;
    }
    else  {
      GLOBAL_RGB_INDEX_2--;
    }
    if (GLOBAL_RGB_INDEX_2 == 0)  {
      RGB_DIRECTION = 1;
    }
    else if (GLOBAL_RGB_INDEX_2 == 7)  {
      RGB_DIRECTION = 0;
    }
    t4 = millis();
  }
}


// დროს და თარიღის დაყენების ფუნქცია, "datetime" String პარამეტრის მიხედვით...
void setDateTime(String datetime)  {
  hour = datetime.substring(0, datetime.indexOf(':')).toInt();
  minute = datetime.substring(datetime.indexOf(':') + 1, datetime.indexOf(':', datetime.indexOf(':') + 1)).toInt();
  second = datetime.substring(datetime.indexOf(':', datetime.indexOf(':') + 1) + 1, datetime.indexOf(' ')).toInt();
  year = datetime.substring(datetime.indexOf(' ') + 1, datetime.indexOf('-')).toInt();
  month = datetime.substring(datetime.indexOf('-') + 1, datetime.indexOf('-', datetime.indexOf('-') + 1)).toInt();
  day = datetime.substring(datetime.indexOf('-', datetime.indexOf('-') + 1) + 1).toInt();
}
```

```
void UART() {

  if (Serial.available())  {// თუ UART "Serial" ინტერფეისის RX (მიმღებ) პინზე მოწოდებულია ინფორმაცია

    received = Serial.readString();// UART კავშირით მიღებული წინადადების ამოკითხვა და "received" String ტიპის ცვლადში შენახვა

    if (received.charAt(0) == 't') {

      setDateTime(received.substring(1));

      //     if (!inputDisabled)  {

      //       inputDisabled = isInputDisabled_time();

      //     }

      //     if (!connectDisabled)  {

      //       connectDisabled = isConnectDisabled_time();

      //     }

    }

    else if (received == "n" && countInputs && !inputDisabled)  {// თუ მიღებულია სიმბოლო "n" და "countInputs" ცვლადი 0-სგან განსხვავებულია

      Serial.print('n' + inputNumbers[--countInputs]);// UART კავშირის საშუალებით, მომხმარებლების მიერ გამოგზავნილი რიცხვების გადაცება "Arduino"-ს დაფაზე არსებული მიკროკონტროლერისთვის

      inputNumbers[countInputs] = "";// "inputNumbers" მასივის ელემენტში, რომლის ინდექსია "countInputs" ცვლადში შენახული რიცხვი, ინახება ცარიელი String

    }

  }

}


// ფუნქცია, რომელიც დააბრუნებს ლოგიკურ 1-ს იმ შემთხვევაში, თუ მისთვის პარამეტრებად გადაცემული დრო და თარიღი უდრის მიმდინარე დროს და თარიღს.. წინააღმდეგ შემთხვევაში, დააბრუნებს ლოგიკურ 0-ს..

boolean datetime_equals(byte local_second, byte local_minute, byte local_hour, byte local_day, byte local_month, int local_year) {
```

```
  return ((second == local_second) && (minute == local_minute) && (hour == local_hour) && (day == local_day) && (month == local_month) && (year == local_year));
}

void BIDIRECTIONAL_RANDOM_RGB_DEMUX() {
  if (millis() - t6 >= 250)  {
    redValue = random(0, 1024);
    greenValue = random(0, 1024);
    blueValue = random(0, 1024);
    RGB(redValue, greenValue, blueValue);
    if (!doubleRGB)  {
      DEMUX_RGB(redValue, greenValue, blueValue, GLOBAL_RGB_INDEX);
    }
    if (RGB_DIRECTION)  {
      GLOBAL_RGB_INDEX++;
    }
    else  {
      GLOBAL_RGB_INDEX--;
    }
    if (GLOBAL_RGB_INDEX == 0)  {
      RGB_DIRECTION = 1;
    }
    else if (GLOBAL_RGB_INDEX == 7)  {
      RGB_DIRECTION = 0;
    }
    t6 = millis();
  }
  if (doubleRGB)  {
    DEMUX_RGB(redValue, greenValue, blueValue, GLOBAL_RGB_INDEX);
```

```
    DEMUX_RGB(redValue, greenValue, blueValue, 7 - GLOBAL_RGB_INDEX);
  }
}

void RGB4() {
  switch (DEMUX_RGB_MODE)  {
    case 0:
      changeRGBindex();
      DEMUX_RGB_RANDOM(0xFF << GLOBAL_RGB_INDEX_2);
      break;
    case 1:
      changeRGBindex();
      DEMUX_RGB_RANDOM(0xFF >> GLOBAL_RGB_INDEX_2);
      break;
    case 2:
      changeRGBindex();
      DEMUX_RGB_RANDOM((0xFF << GLOBAL_RGB_INDEX_2) ^ (0xFF >> GLOBAL_RGB_INDEX_2));
      break;
    case 3:
      if (millis() - t7 >= RGB_TIME)  {
        GLOBAL_RGB_INDEX_2 = random(0, 256);
        t7 = millis();
      }
      DEMUX_RGB_RANDOM(GLOBAL_RGB_INDEX_2);
      break;
  }
  multiplexRGB();
  if (millis() - t5 >= RGB_MODE_CHANGE_TIME)  {
    if (++DEMUX_RGB_MODE > 3)  {
```

```
    DEMUX_RGB_MODE = 0;
   }
   t5 = millis();
  }
}

// ფუნქცია, სადაც დროს და თარიღის მიხედვით ხდება ელექტრული სქემის მართვა...
void functionDateTime() {
  if (datetime_equals(allowedInputTime_second, allowedInputTime_minute, allowedInputTime_hour,
allowedInputDate_day, allowedInputDate_month, allowedInputDate_year))  {
   inputDisabled = 1;
  }
  else if (datetime_equals(allowedConnectTime_second, allowedConnectTime_minute,
allowedConnectTime_hour, allowedConnectDate_day, allowedConnectDate_month,
allowedConnectDate_year))  {
   connectDisabled = 1;
   RGB(0, 0, 0);
  }
  else if (datetime_equals(0, 0, 0, 1, 1, 2021))  {
   softRGBtime = 1000;
   softRGBenabled = 1;
   RGB_DEMUX_ALL_TIME = 1000;
   RGB_DEMUX_ALL_ENABLED = 1;
  }
  else if (datetime_equals(30, 0, 0, 1, 1, 2021))  {
   softRGBtime = 250;
   RGB_DEMUX_ALL_TIME = 250;
  }
  else if (datetime_equals(0, 1, 0, 1, 1, 2021)) {
   for (byte arrayIndex = 0; arrayIndex < 8; arrayIndex++) {
```

```
    countRGBmodes2[arrayIndex] = arrayIndex;
    DEMUX_RGB_SETUP(arrayIndex);
  }
  RGB_DEMUX_ALL_ENABLED = 0;
  RGB_SOFT_MIX_TIME = 1;
  RGB_DEMUX_softMix_ENABLED = 1;
}
else if (datetime_equals(30, 1, 0, 1, 1, 2021)) {
  softRGBenabled = 0;
  RGB_DEMUX_softMix_ENABLED = 0;
  RGB_TIME = 100;
  DEMUX_RGB_RANDOM_ENABLED = 1;
}
else if (datetime_equals(0, 2, 0, 1, 1, 2021)) {
  DEMUX_RGB_RANDOM_ENABLED = 0;
  GLOBAL_RGB_INDEX = 0;
  RGB_DIRECTION = 1;
  BIDIRECTIONAL_RANDOM_RGB_DEMUX_ENABLED = 1;
}
else if (datetime_equals(15, 2, 0, 1, 1, 2021)) {
  doubleRGB = 1;
}
else if (datetime_equals(30, 2, 0, 1, 1, 2021)) {
  BIDIRECTIONAL_RANDOM_RGB_DEMUX_ENABLED = 0;
  RGB_DIRECTION = 1;
  RGB_TIME = 250;
  RGB_MODE_CHANGE_TIME = 15000;
  RGB4enabled = 1;
  t5 = millis();
```

```
  }
  else if (datetime_equals(32, 3, 0, 1, 1, 2021)) {
   RGB_TIME = 100;
   RGB_MODE_CHANGE_TIME = 8000;
  }
  else if (datetime_equals(15, 4, 0, 1, 1, 2021)) {
   RGB_TIME = 10;
   RGB_MODE_CHANGE_TIME = 800;
  }
  else if (datetime_equals(30, 4, 0, 1, 1, 2021)) {
   RGB4enabled = 0;
   softRGBtime = 750;
   softRGBenabled = 1;
   RGB_SOFT_MIX_TIME = 10;
   RGB_DEMUX_softMix_ENABLED = 1;
  }
}

// ფუნქცია, სადაც წერია ბრძანებები, რომლებიც უწყვეტად სრულდება...
void loopFunction() {
  server.handleClient();
  MDNS.update();
  UART();
  functionDateTime();
  if (RGB_DEMUX_ALL_ENABLED)  {
   // დემულტიპლექსორთან დაკავშირებული ყველა შუქდიოდის ანთება...
   RGBcoloursTransition_DEMUX(RGB_DEMUX_ALL_TIME, GLOBAL_RGB_INDEX);
   multiplexRGB();
  }
```

```
  else if (RGB_DEMUX_softMix_ENABLED)  {
    RGBcoloursTransition_DEMUX(RGB_SOFT_MIX_TIME, GLOBAL_RGB_INDEX);
    multiplexRGB();
  }
  else if (DEMUX_RGB_RANDOM_ENABLED) {
    DEMUX_RGB_RANDOM(0xFF);
    multiplexRGB();
  }
  else if (BIDIRECTIONAL_RANDOM_RGB_DEMUX_ENABLED) {
    BIDIRECTIONAL_RANDOM_RGB_DEMUX();
  }
  else if (RGB4enabled) {
    RGB4();
  }
}

void setup() {
  // გარკვეული პინები ცხადდება როგორც გამოსავალი პინები
  // RGB შუქდიოდის შესაბამისი პინები...
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(redDEMUX, OUTPUT);
  pinMode(greenDEMUX, OUTPUT);
  pinMode(blueDEMUX, OUTPUT);
  // დემულტიპლექსორის ციფრული პინების შესაბამისი პინები...
  pinMode(DEMUX_A, OUTPUT);
  pinMode(DEMUX_B, OUTPUT);
  pinMode(DEMUX_C, OUTPUT);
```

```
  Serial.begin(115200);// UART კავშირის დაწყება 115200kb/s (1000 ბიტი წამში) სიჩქარით

  Serial.setTimeout(2);// UART კავშირით ინფორმაციის მიღებისას, ლოდინის დრო იქნება 2 მილიწამი (0.002 წამი).. ეს წესი არ ეხება ისეთ შემთხვევებს, როცა ხდება მხოლოდ 1 სიმბოლოს ამოკითხვა (მაგალითად Serial.read() ბრძანება)..

  WiFi.mode(WIFI_STA);// WiFi-ს მუშაობის რეჟიმის მითითება

  WiFi.begin(ssid, password);// WiFi კავშირის დაწყება

  //Serial.println("");


  // Wait for connection

  // ციკლი მუშაობს მანამ, სანამ WiFi კავშირი არ დამყარდება...

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    //Serial.print(".");

  }

  //Serial.println("");

  //Serial.print("Connected to ");

  //Serial.println(ssid);

  //Serial.print("IP address: ");

  //Serial.println(WiFi.localIP());


  if (MDNS.begin("esp8266")) {

    //Serial.println("MDNS responder started");

  }

  // მითითება, თუ რომელი ფუნქციები შესრულდეს სერვერის ბმულების გახსნისას

  server.on("/", handleRoot);

  server.on("/page_style", handlePageStyle);

  server.on("/get_script", handlePageScript);

  server.on("/images", webImages);
```

```
  server.on("/get_number", handleGetNumber);
  server.on("/set_datetime", handleSetDateTime);
  server.on("/get_datetime", handleGetDateTime);
  server.on("/datetime", handleDateTime);
  server.onNotFound(handleNotFound);
  server.begin();// სერვერის კავშირის დაწყება
  //Serial.println("HTTP server started");
  RGB(1023, 0, 0);// RGB შუქდიოდის ანთება წითლად
}

void loop() {
  loopFunction();// "loopFunction" ფუნქციის გამოძახება
  if (softRGBenabled)  {
    RGBcoloursTransition(softRGBtime);
  }
}
```