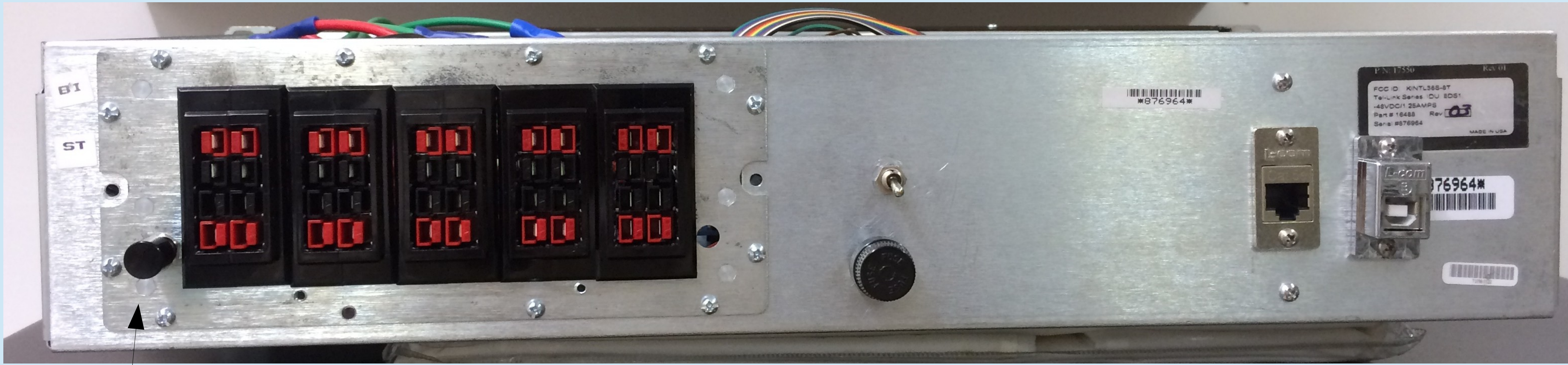
# Arduino Ethernet Device Control Example

## Arduino Ethernet Device Control Example

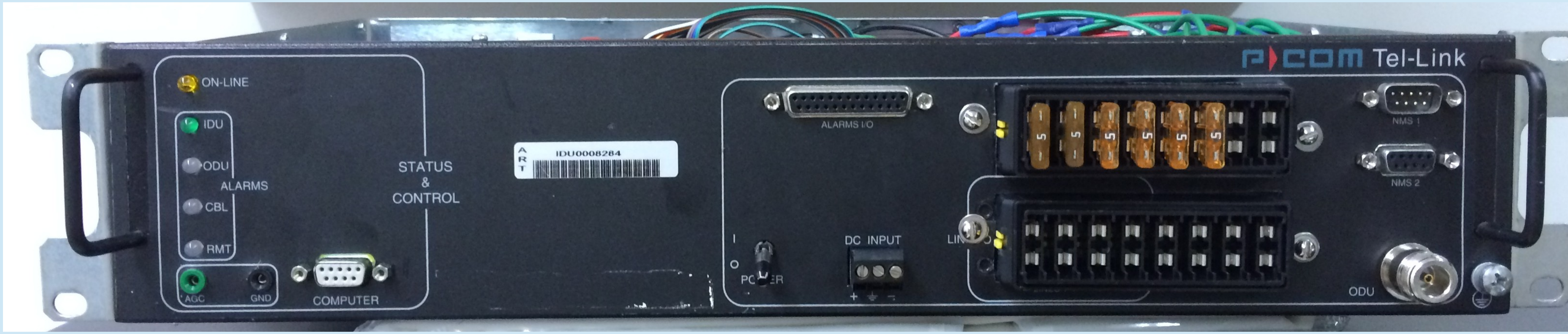
- Use Arduino to create a web page, provide on/off control for 16 devices via the Ethernet
  - Can use for power control,
  - transverter or antenna bandswitching,
  - switching mic, receive audio, foot switch, CW key, etc. among IF rigs
  - turning cameras on/off or switching between cameras

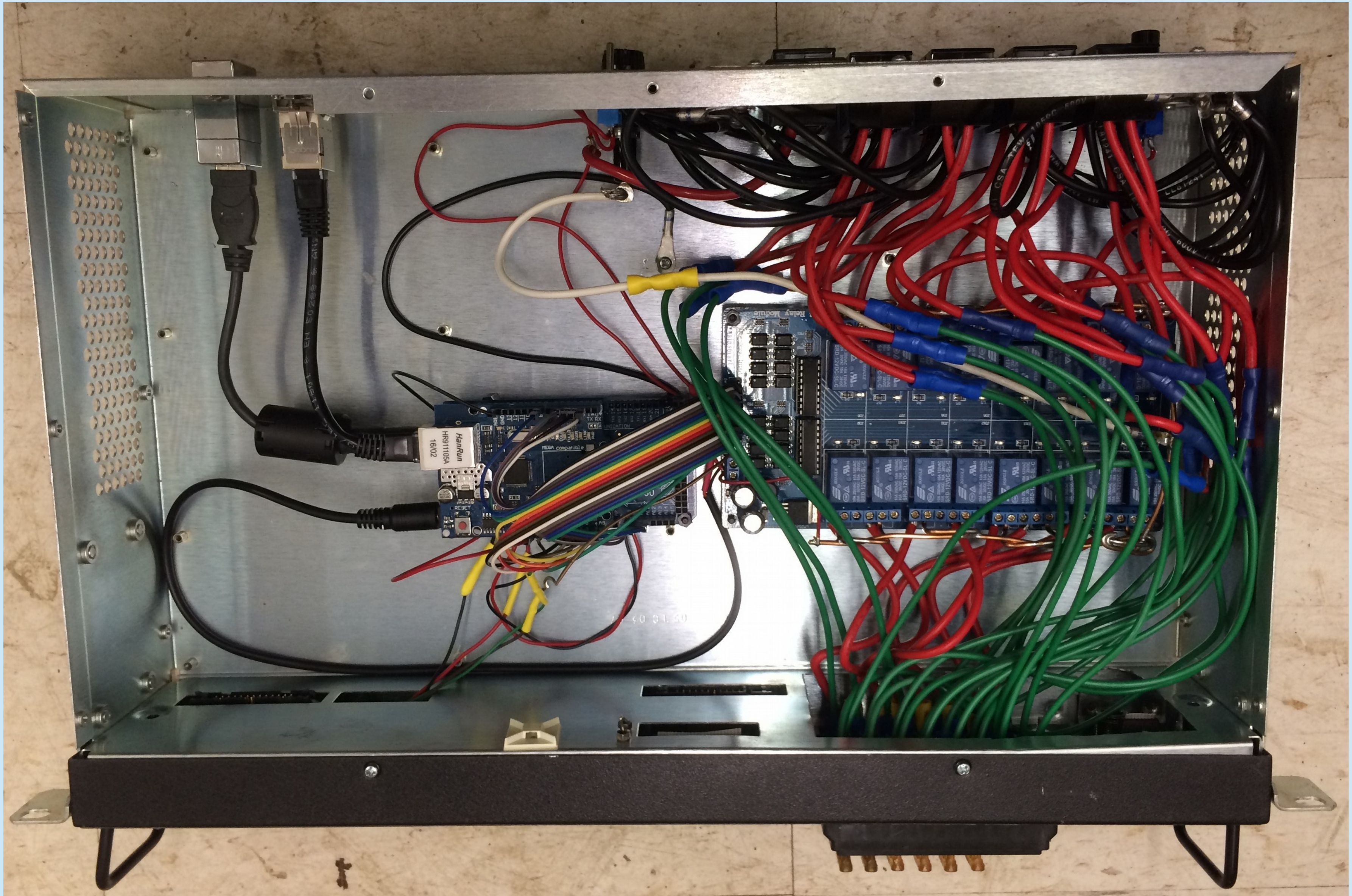
## Arduino Ethernet Device Control Example

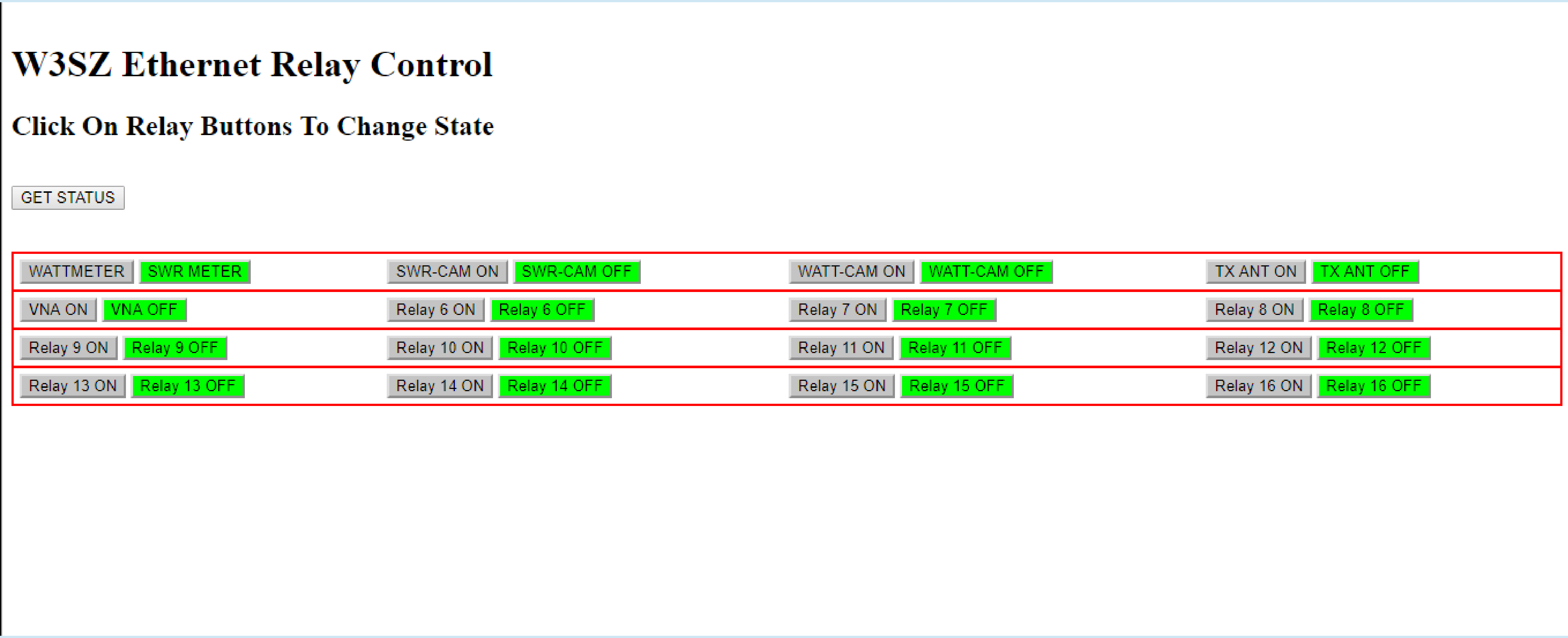
- Originally needed to use MEGA due to memory requirements:
  - Used 4084 bytes of SRAM (dynamic memory)
  - UNO only has 2048 bytes of SRAM
- Subsequent coding changes reduced SRAM to 1598
- Arduino MEGA and ethernet shield from eBay
  - Cost $13.66 with free shipping



RESET Button


P COM Tel-Link
ON-LINE
IDU
ODU
ALARMS
CBL
RMT
AGC
GND
COMPUTER
STATUS & CONTROL
IDU0008284
ALARMS I/O
POWER
DC INPUT
NMS 1
NMS 2
ODU





# W3SZ Ethernet Relay Control

## Click On Relay Buttons To Change State

# Arduino Ethernet Device Control
Live Demo

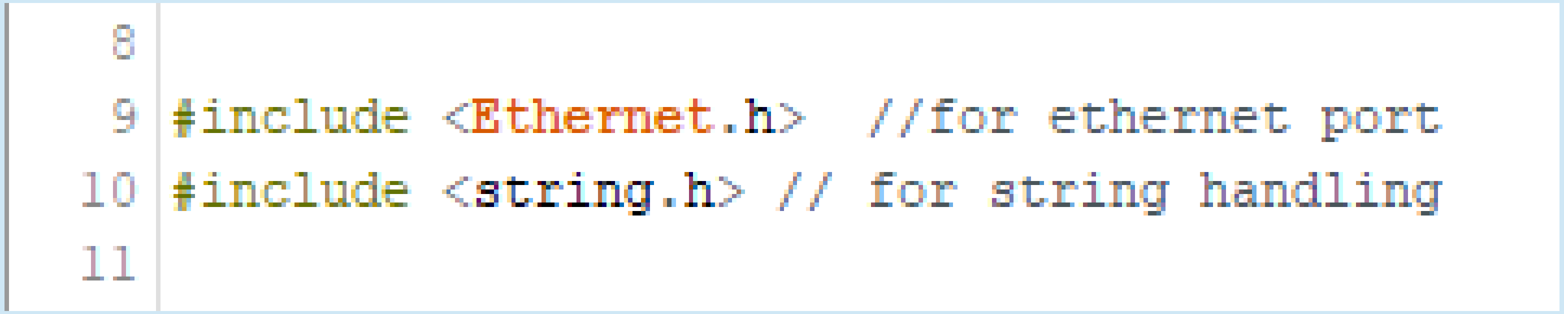
## Arduino Ethernet Device Control Example: Arduino Code

1) Include Libraries that are needed
2) Define/initialize constants and variables
3) Setup()
    - Define and initialize output pins
    - Start ethernet port and serial port
4) Loop()
    - Get ethernet data
    - Parse ethernet data
    - Switch relays on or off
5) Call procedure “sendReply” to:
    - Send relay status back to client and re-write web page at client
    - (Web page uses HTML buttons to send commands to Arduino to control relays and read relay status)

# Arduino Ethernet Device Control Example: Arduino Code

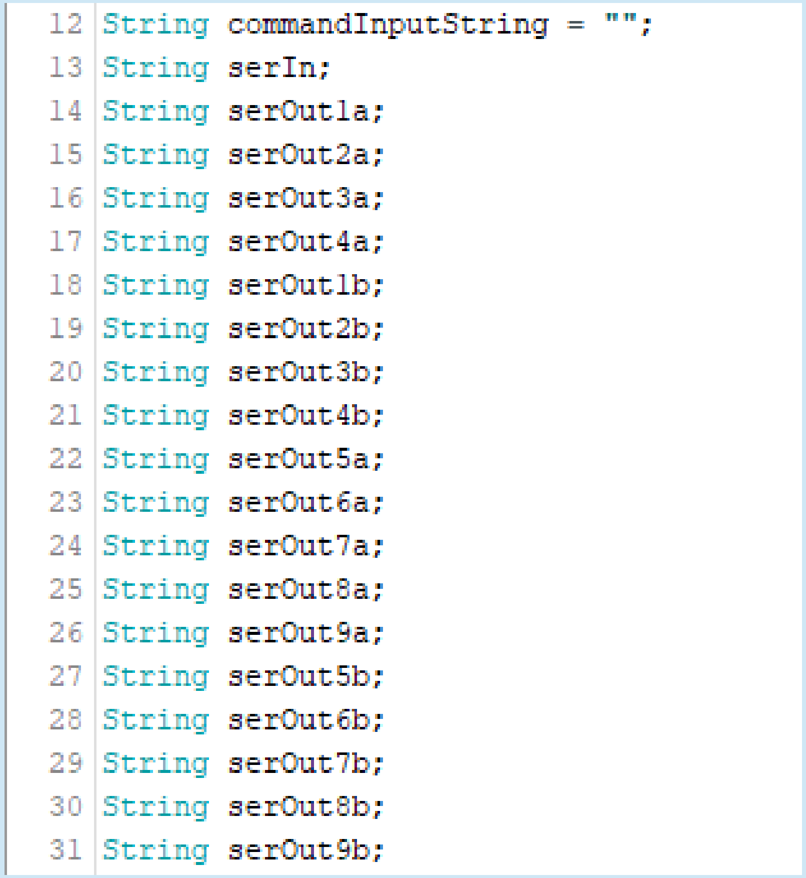
- For relay control uses GPIO pins 2-6, 8, A0-A5, A8-A11
- Depending on characteristics of relay board, may need to use reverse logic for relay control
- For this example we will NOT use reverse logic

# Include Libraries

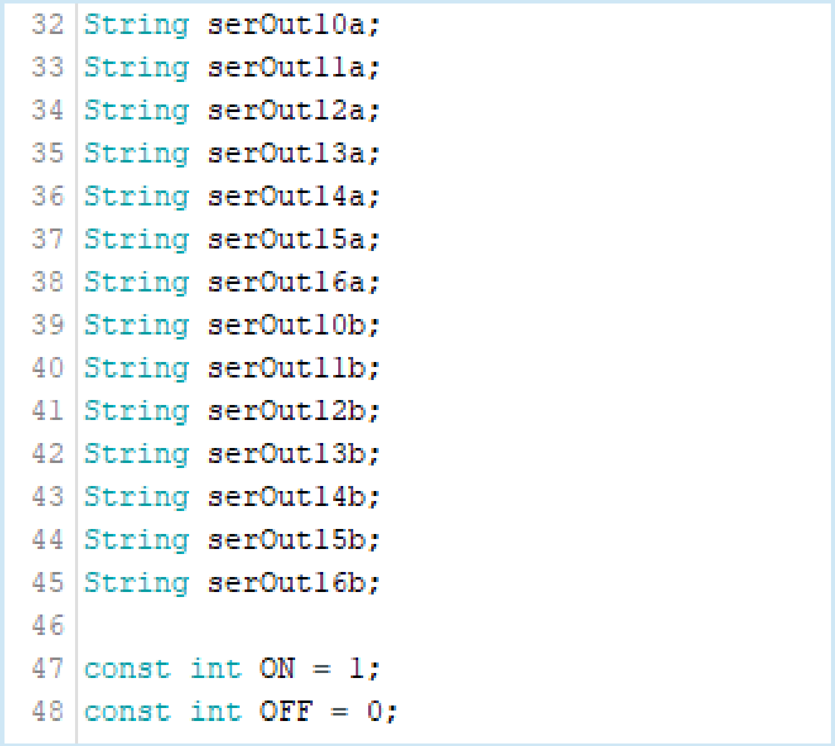


```

#include <Ethernet.h>  //for ethernet port
#include <string.h> // for string handling

```

# Define Variables & Constants



```
String commandInputString = "";
String serIn;
String serOut1a;
String serOut2a;
String serOut3a;
String serOut4a;
String serOut1b;
String serOut2b;
String serOut3b;
String serOut4b;
String serOut5a;
String serOut6a;
String serOut7a;
String serOut8a;
String serOut9a;
String serOut5b;
String serOut6b;
String serOut7b;
String serOut8b;
String serOut9b;
```



```
String serOut10a;
String serOut11a;
String serOut12a;
String serOut13a;
String serOut14a;
String serOut15a;
String serOut16a;
String serOut10b;
String serOut11b;
String serOut12b;
String serOut13b;
String serOut14b;
String serOut15b;
String serOut16b;

const int ON = 1;
const int OFF = 0;
```

# Ethernet.h

- Library to work with Ethernet Shield, Ethernet Shield 2, and Leonardo Ethernet. Contains the classes:

  Ethernet: members begin(), localIP(), maintain()

  IPAddress: member IPAddress()

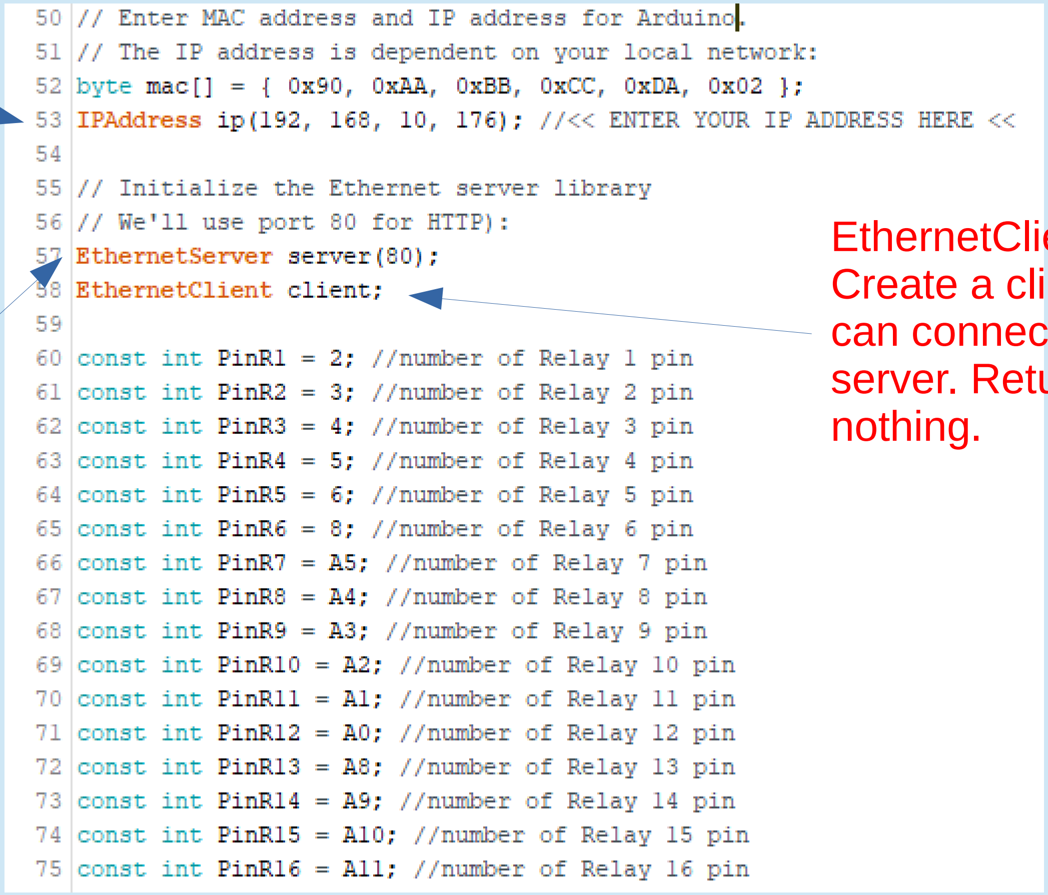
  Server: members Server, EthernetServer(), begin(), available(), write(), print(), println()

  Client: members Client, EthernetClient(), if(EthernetClient), connected(), connect(), write(), print(), println(), available(), read(), flush(), stop()

  EthernetUdp members begin(), read(), write(), beginPacket(), endPacket(), parsePacket(), available(), stop(), remoteIP(), remotePort()

# Define Variables & Constants

IPAddress(address): a comma delimited list representing the address (4 bytes, ex. 192, 168, 1, 1). Returns nothing.

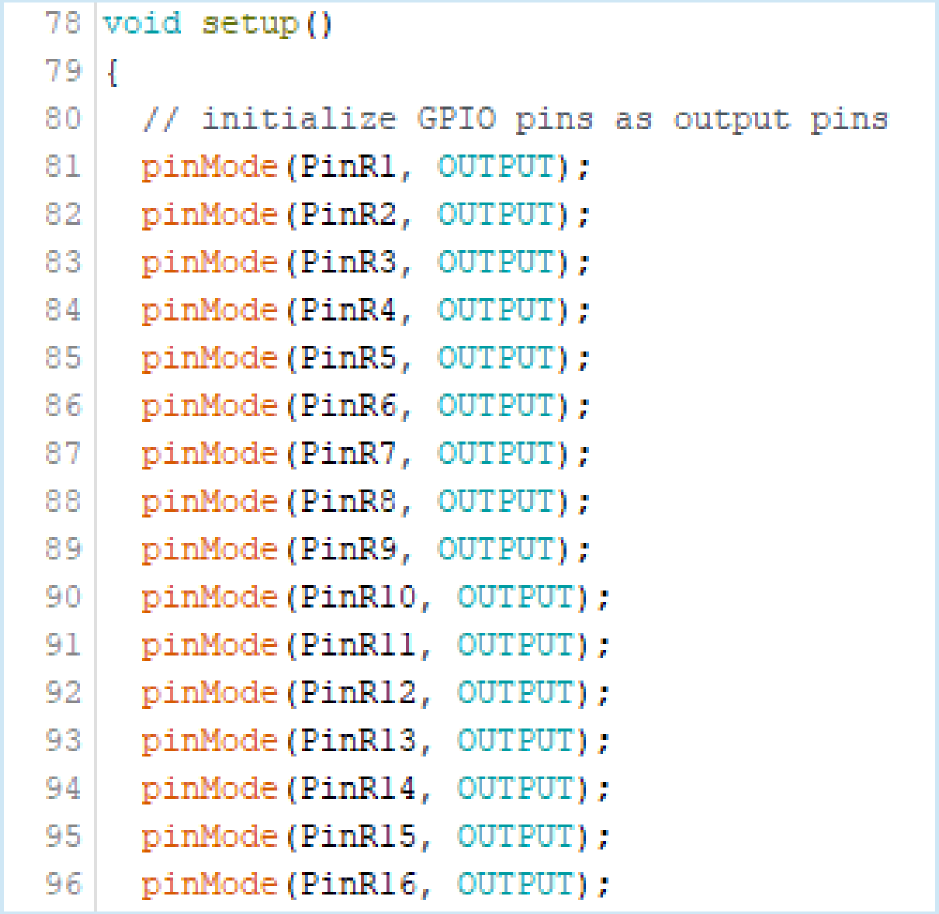


```
// Enter MAC address and IP address for Arduino.
// The IP address is dependent on your local network:
byte mac[] = { 0x90, 0xAA, 0xBB, 0xCC, 0xDA, 0x02 };
IPAddress ip(192, 168, 10, 176); //<< ENTER YOUR IP ADDRESS HERE <<

// Initialize the Ethernet server library
// We'll use port 80 for HTTP):
EthernetServer server(80);
EthernetClient client;

const int PinR1 = 2; //number of Relay 1 pin
const int PinR2 = 3; //number of Relay 2 pin
const int PinR3 = 4; //number of Relay 3 pin
const int PinR4 = 5; //number of Relay 4 pin
const int PinR5 = 6; //number of Relay 5 pin
const int PinR6 = 8; //number of Relay 6 pin
const int PinR7 = A5; //number of Relay 7 pin
const int PinR8 = A4; //number of Relay 8 pin
const int PinR9 = A3; //number of Relay 9 pin
const int PinR10 = A2; //number of Relay 10 pin
const int PinR11 = A1; //number of Relay 11 pin
const int PinR12 = A0; //number of Relay 12 pin
const int PinR13 = A8; //number of Relay 13 pin
const int PinR14 = A9; //number of Relay 14 pin
const int PinR15 = A10; //number of Relay 15 pin
const int PinR16 = A11; //number of Relay 16 pin
```

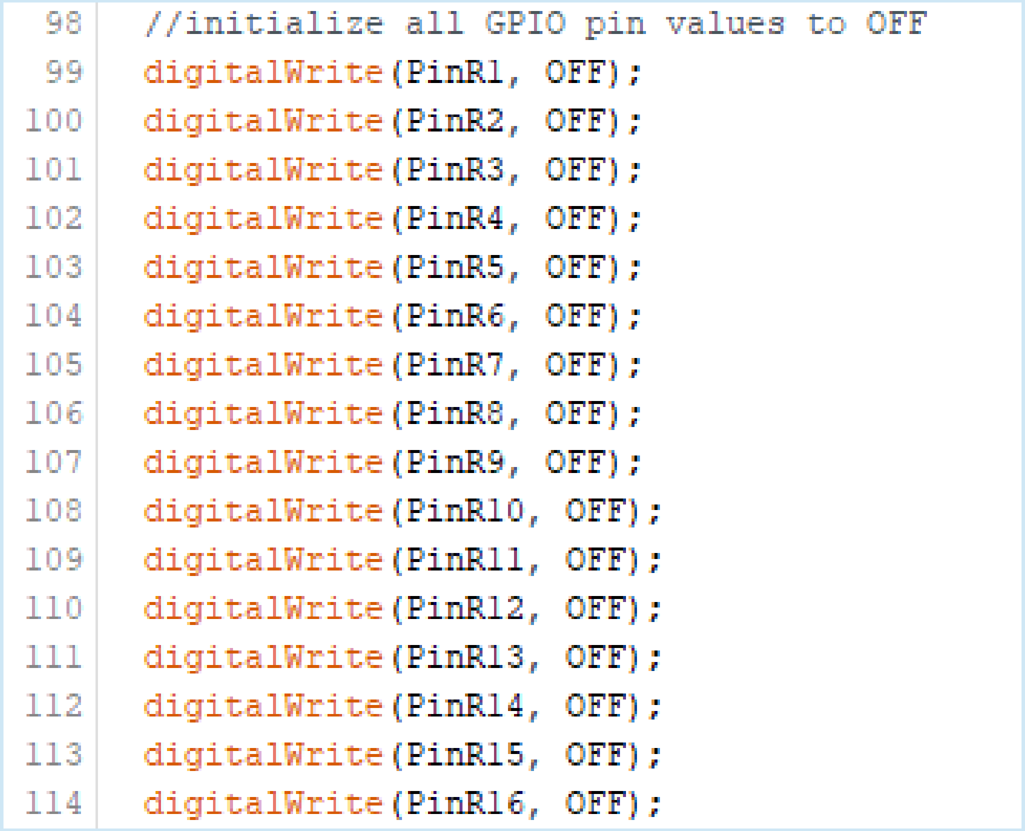
EthernetServer(port): Create a server that listens for incoming connections on the specified port. Returns nothing.

EthernetClient: Create a client that can connect to a server. Returns nothing.

## **Setup:** Initialize GPIO Pins

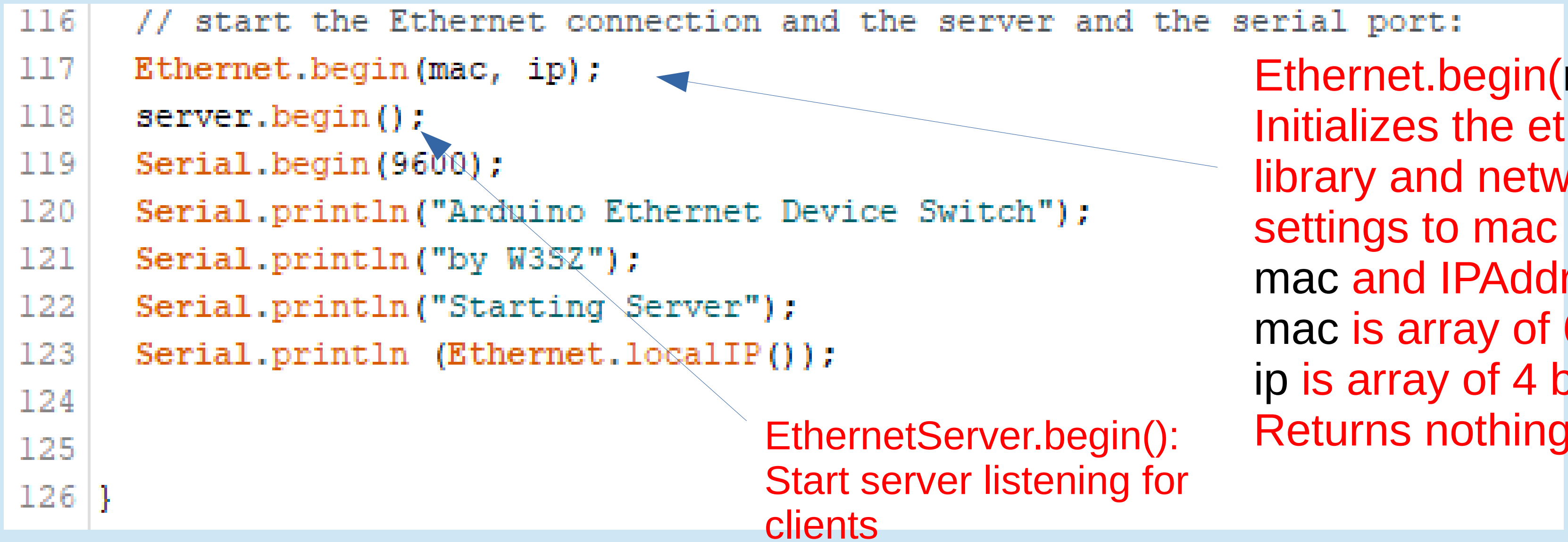


```
void setup()
{
  // initialize GPIO pins as output pins
  pinMode(PinR1, OUTPUT);
  pinMode(PinR2, OUTPUT);
  pinMode(PinR3, OUTPUT);
  pinMode(PinR4, OUTPUT);
  pinMode(PinR5, OUTPUT);
  pinMode(PinR6, OUTPUT);
  pinMode(PinR7, OUTPUT);
  pinMode(PinR8, OUTPUT);
  pinMode(PinR9, OUTPUT);
  pinMode(PinR10, OUTPUT);
  pinMode(PinR11, OUTPUT);
  pinMode(PinR12, OUTPUT);
  pinMode(PinR13, OUTPUT);
  pinMode(PinR14, OUTPUT);
  pinMode(PinR15, OUTPUT);
  pinMode(PinR16, OUTPUT);
```



```
  //initialize all GPIO pin values to OFF
  digitalWrite(PinR1, OFF);
  digitalWrite(PinR2, OFF);
  digitalWrite(PinR3, OFF);
  digitalWrite(PinR4, OFF);
  digitalWrite(PinR5, OFF);
  digitalWrite(PinR6, OFF);
  digitalWrite(PinR7, OFF);
  digitalWrite(PinR8, OFF);
  digitalWrite(PinR9, OFF);
  digitalWrite(PinR10, OFF);
  digitalWrite(PinR11, OFF);
  digitalWrite(PinR12, OFF);
  digitalWrite(PinR13, OFF);
  digitalWrite(PinR14, OFF);
  digitalWrite(PinR15, OFF);
  digitalWrite(PinR16, OFF);
```

## **Setup:** Start Ethernet Port



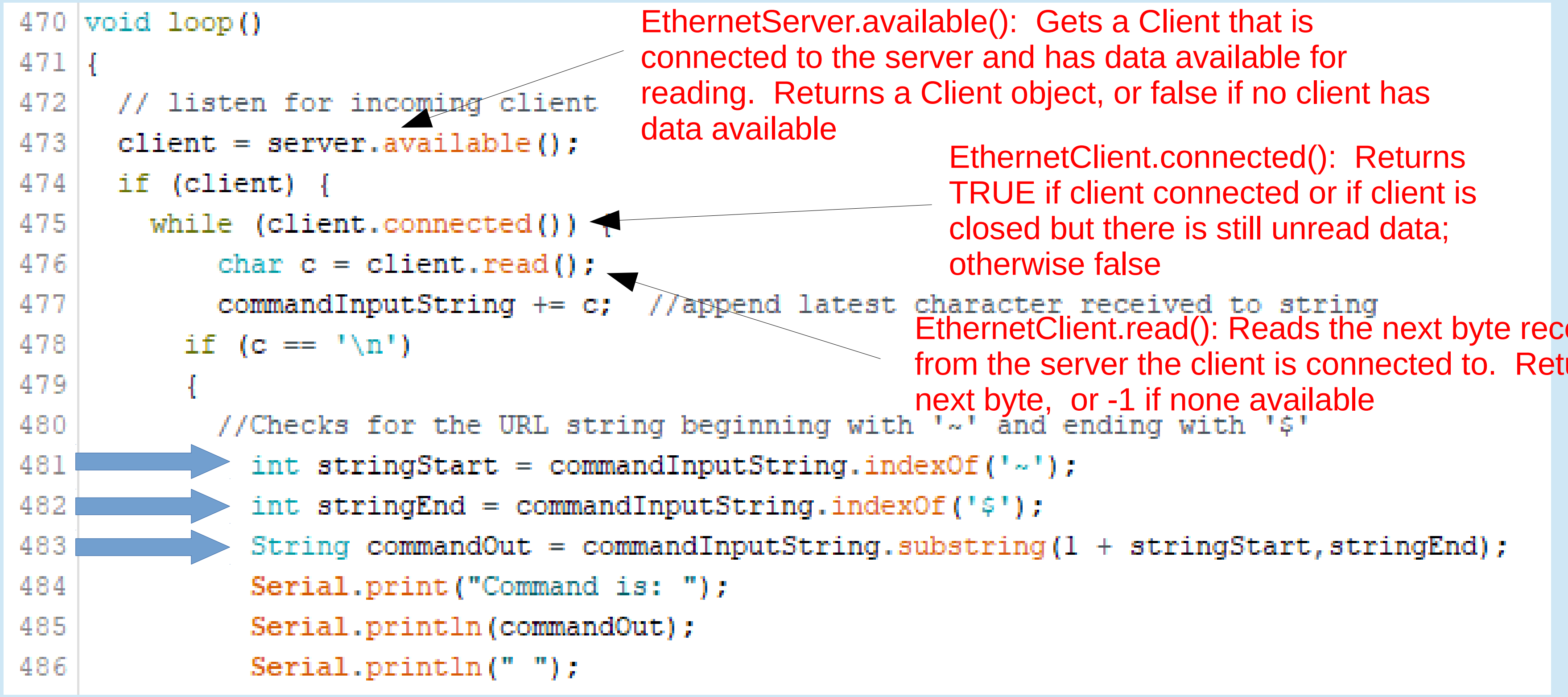
```
  // start the Ethernet connection and the server and the serial port:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.begin(9600);
  Serial.println("Arduino Ethernet Device Switch");
  Serial.println("by W3SZ");
  Serial.println("Starting Server");
  Serial.println (Ethernet.localIP());


}
```

Ethernet.begin(mac, ip): Initializes the ethernet library and network settings to mac address mac and IPAddress ip. mac is array of 6 bytes. ip is array of 4 bytes. Returns nothing.

EthernetServer.begin(): Start server listening for clients

# Arduino Ethernet Device Control Example:

## Loop to Get Ethernet Data, Parse It, Switch Relays, Send Status Back to HTML Client and Refresh Web Page



```
void loop()
{
  // listen for incoming client
  client = server.available();
  if (client) {
    while (client.connected()) {
        char c = client.read();
        commandInputString += c;  //append latest character received to string
      if (c == '\n')
      {
        //Checks for the URL string beginning with '~' and ending with '$'
          int stringStart = commandInputString.indexOf('~');
          int stringEnd = commandInputString.indexOf('$');
          String commandOut = commandInputString.substring(1 + stringStart,stringEnd);
          Serial.print("Command is: ");
          Serial.println(commandOut);
          Serial.println(" ");
```

EthernetServer.available(): Gets a Client that is connected to the server and has data available for reading. Returns a Client object, or false if no client has data available

EthernetClient.connected(): Returns TRUE if client connected or if client is closed but there is still unread data; otherwise false

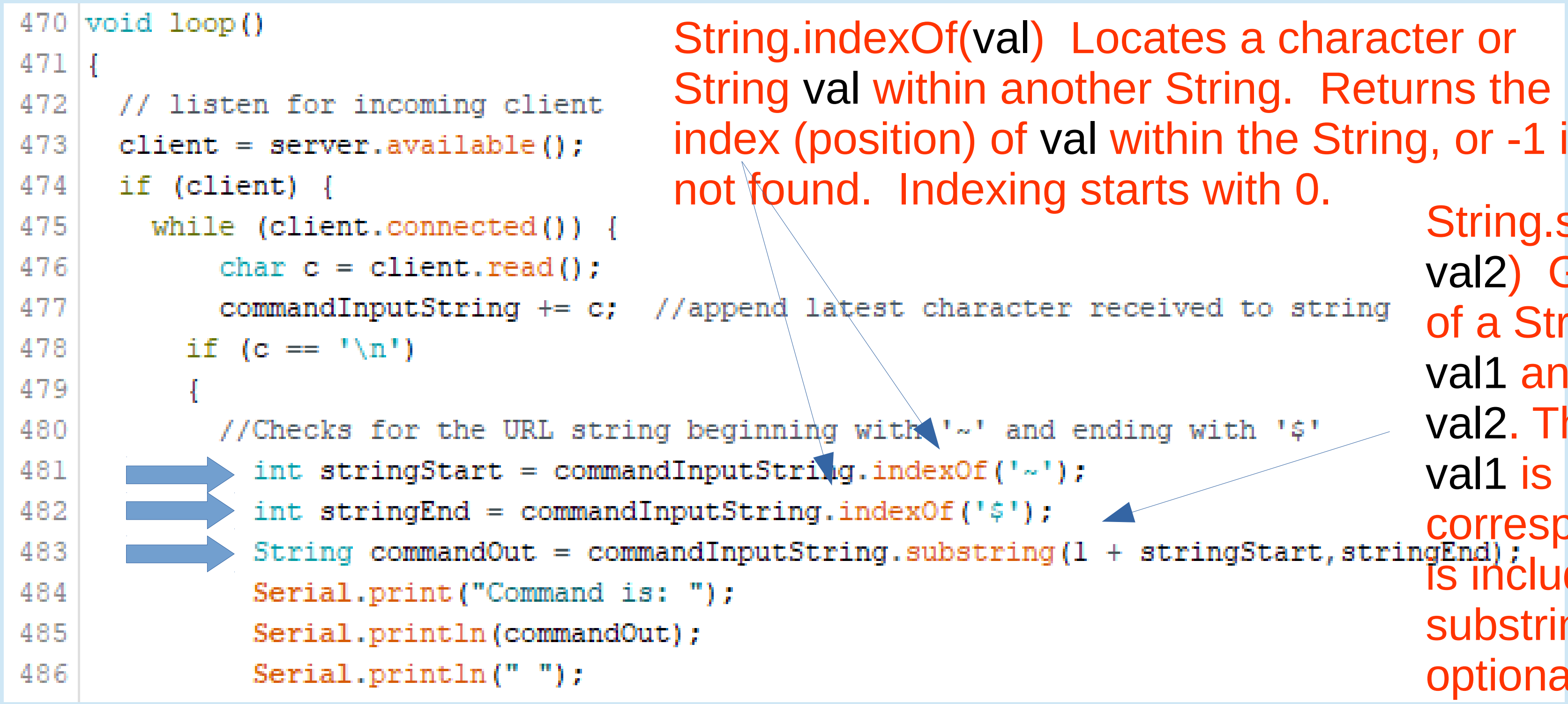
EthernetClient.read(): Reads the next byte received from the server the client is connected to. Returns the next byte, or -1 if none available

# Arduino String class

- Members include:
  charAt
  compareTo
  concat
  c_str
  endsWith
  equals
  equalsIgnoreCase
  getBytes
  indexOf
  lastIndexOf
  length
  remove
  replace
  reserve
  setCharAt
  startsWith
  substring
  toCharArray
  toInt
  toFloat
  toLowerCase
  toUpperCase
  trim

# Arduino Ethernet Device Control Example:

## Loop to Get Ethernet Data, Parse It, Switch Relays, Send Status Back to HTML Client and Refresh Web Page



```
void loop()
{
  // listen for incoming client
  client = server.available();
  if (client) {
    while (client.connected()) {
        char c = client.read();
        commandInputString += c;  //append latest character received to string
      if (c == '\n')
      {
        //Checks for the URL string beginning with '~' and ending with '$'
          int stringStart = commandInputString.indexOf('~');
          int stringEnd = commandInputString.indexOf('$');
          String commandOut = commandInputString.substring(1 + stringStart,stringEnd);
          Serial.print("Command is: ");
          Serial.println(commandOut);
          Serial.println(" ");
```

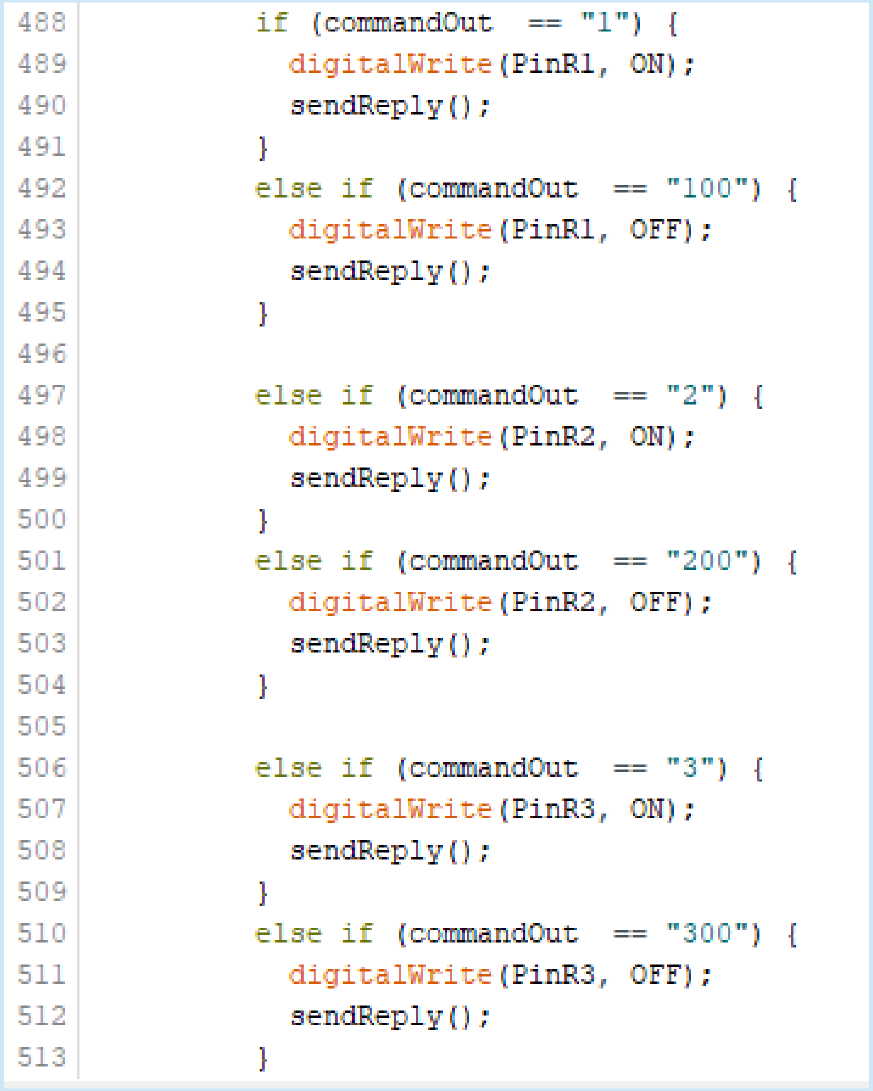
String.indexOf(val) Locates a character or String val within another String. Returns the index (position) of val within the String, or -1 if not found. Indexing starts with 0.

String.substring(val1, val2) Gets a substring of a String, starting with val1 and ending before val2. The starting index val1 is inclusive (the corresponding character is included in the substring), but the optional ending index val2 is exclusive. Returns the substring.

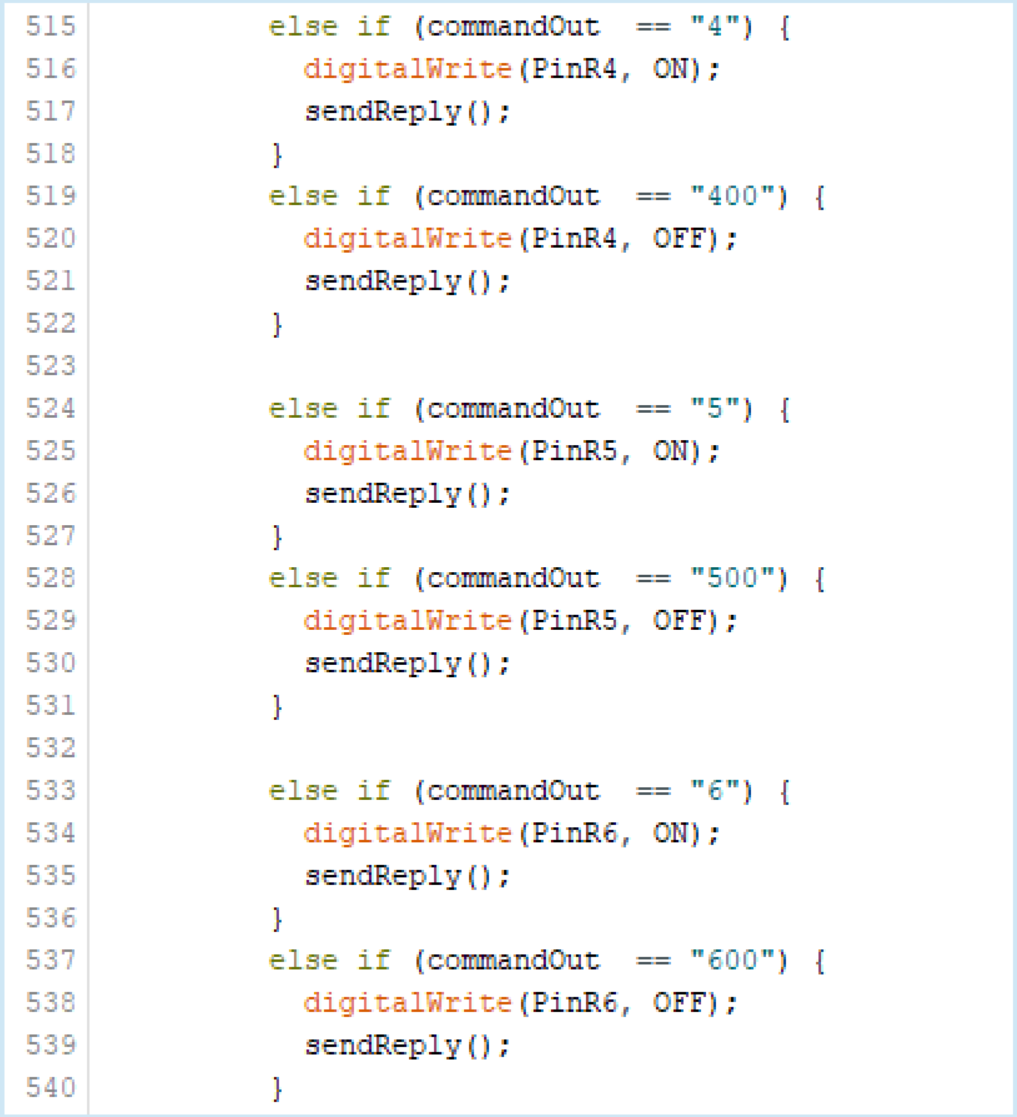
# Where did this arcane client/server stuff come from?

- http://www.instructables.com/id/Arduino-Ethernet-Shield-Tutorial/ has an example that turns LED on and off via the ethernet...a perfect beginning for this project!
- Original Arduino code is here: http://w3sz.com/EthernetLED_Switch.ino
- Remember, if you start by stealing someone else’s code, you will progress much more quickly

# Switch the relays

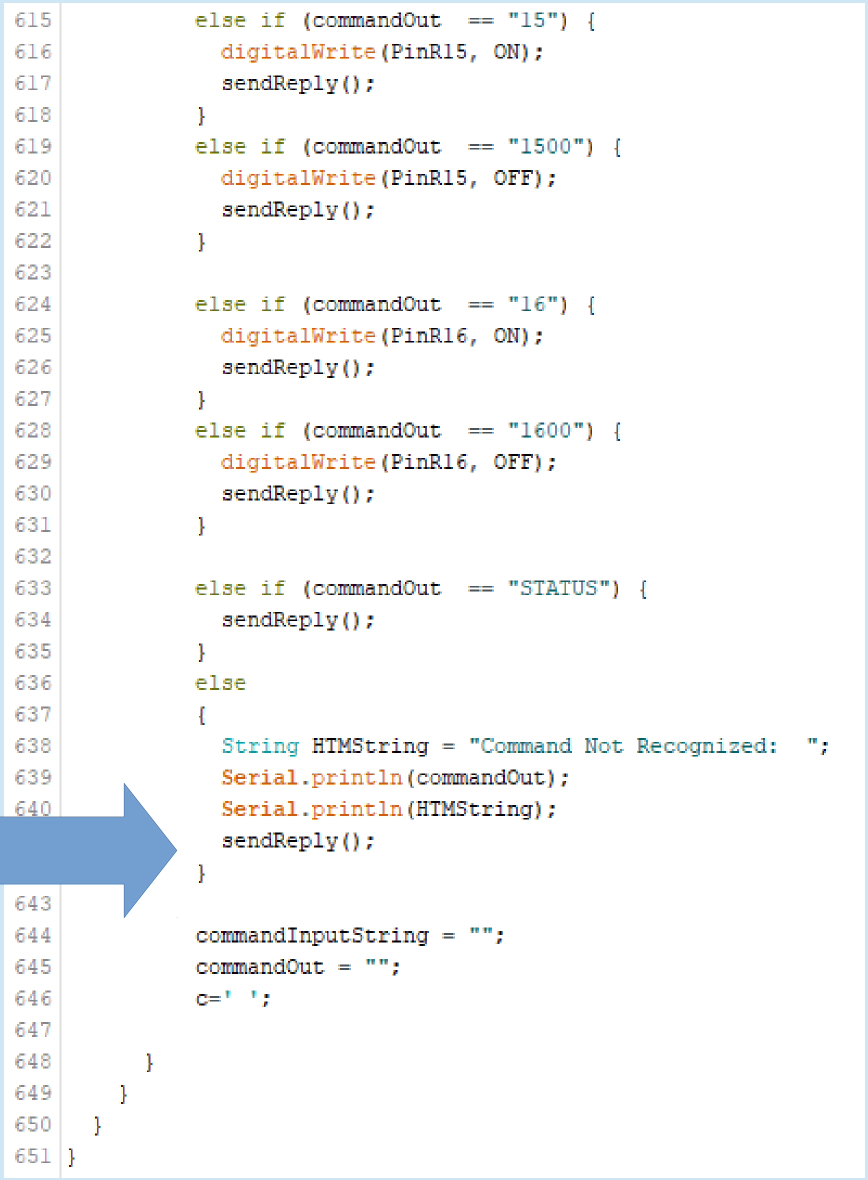


```
if (commandOut  == "1") {
  digitalWrite(PinR1, ON);
  sendReply();
}
else if (commandOut  == "100") {
  digitalWrite(PinR1, OFF);
  sendReply();
}

else if (commandOut  == "2") {
  digitalWrite(PinR2, ON);
  sendReply();
}
else if (commandOut  == "200") {
  digitalWrite(PinR2, OFF);
  sendReply();
}

else if (commandOut  == "3") {
  digitalWrite(PinR3, ON);
  sendReply();
}
else if (commandOut  == "300") {
  digitalWrite(PinR3, OFF);
  sendReply();
}
```



```
      else if (commandOut  == "4") {
        digitalWrite(PinR4, ON);
        sendReply();
      }
      else if (commandOut  == "400") {
        digitalWrite(PinR4, OFF);
        sendReply();
      }

      else if (commandOut  == "5") {
        digitalWrite(PinR5, ON);
        sendReply();
      }
      else if (commandOut  == "500") {
        digitalWrite(PinR5, OFF);
        sendReply();
      }

      else if (commandOut  == "6") {
        digitalWrite(PinR6, ON);
        sendReply();
      }
      else if (commandOut  == "600") {
        digitalWrite(PinR6, OFF);
        sendReply();
      }
```

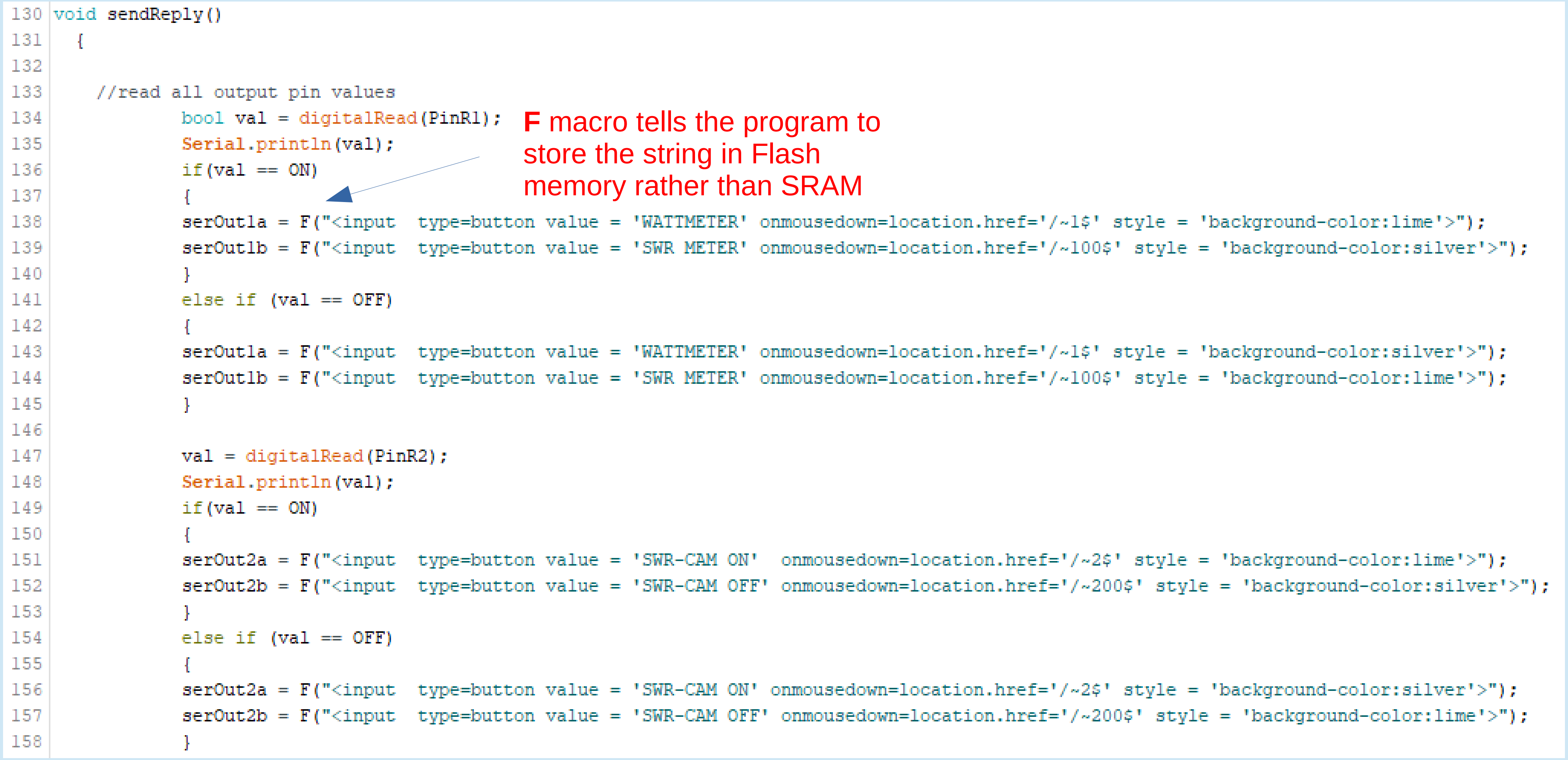
What about SendReply?



```
          else if (commandOut  == "15") {
            digitalWrite(PinR15, ON);
            sendReply();
          }
          else if (commandOut  == "1500") {
            digitalWrite(PinR15, OFF);
            sendReply();
          }

          else if (commandOut  == "16") {
            digitalWrite(PinR16, ON);
            sendReply();
          }
          else if (commandOut  == "1600") {
            digitalWrite(PinR16, OFF);
            sendReply();
          }

          else if (commandOut  == "STATUS") {
            sendReply();
          }
          else
          {
            String HTMString = "Command Not Recognized:  ";
            Serial.println(commandOut);
            Serial.println(HTMString);
            sendReply();
          }

          commandInputString = "";
          commandOut = "";
          c=' ';

      }
    }
  }
}
```

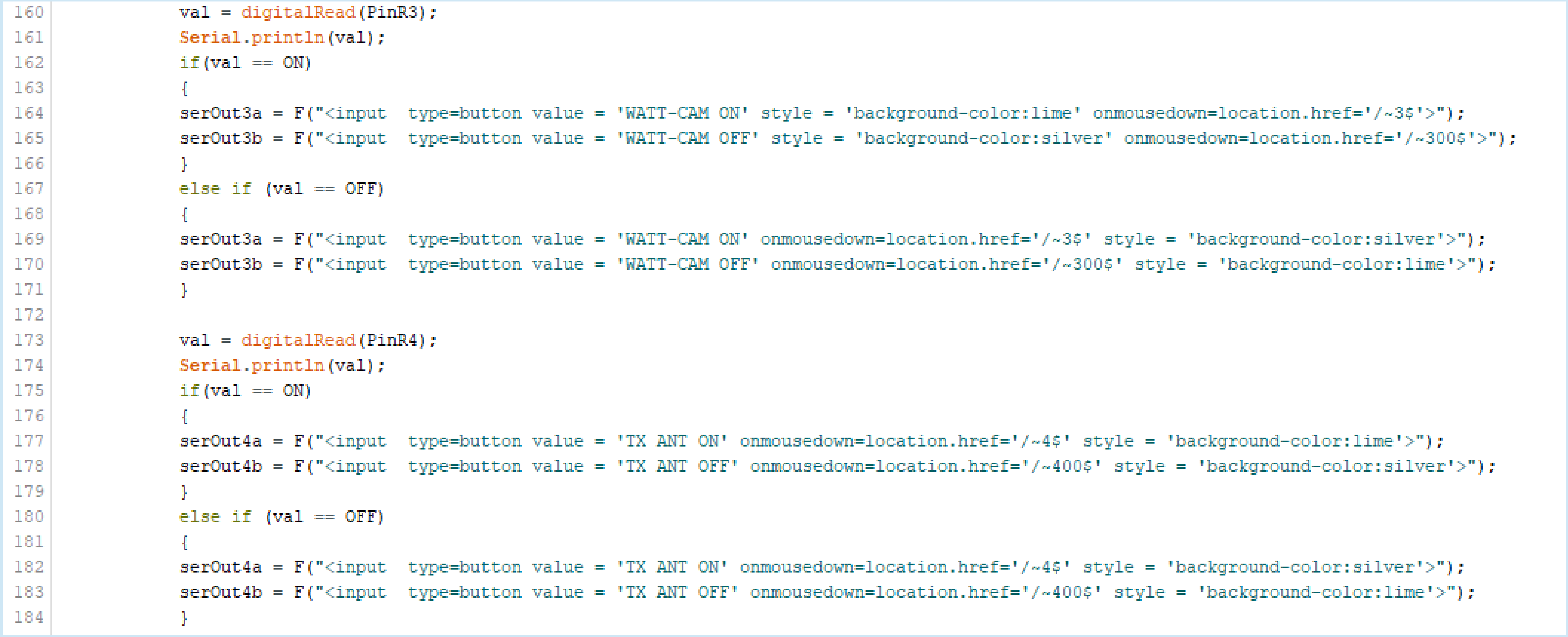
## SendReply() Function

- This routine reads the GPIO pin values and reports them both through the serial port and to the HTML client
- It also creates the web page for the HTML client, including the HTML buttons on the web page and defines what is sent to the Arduino when each button is clicked on the web page

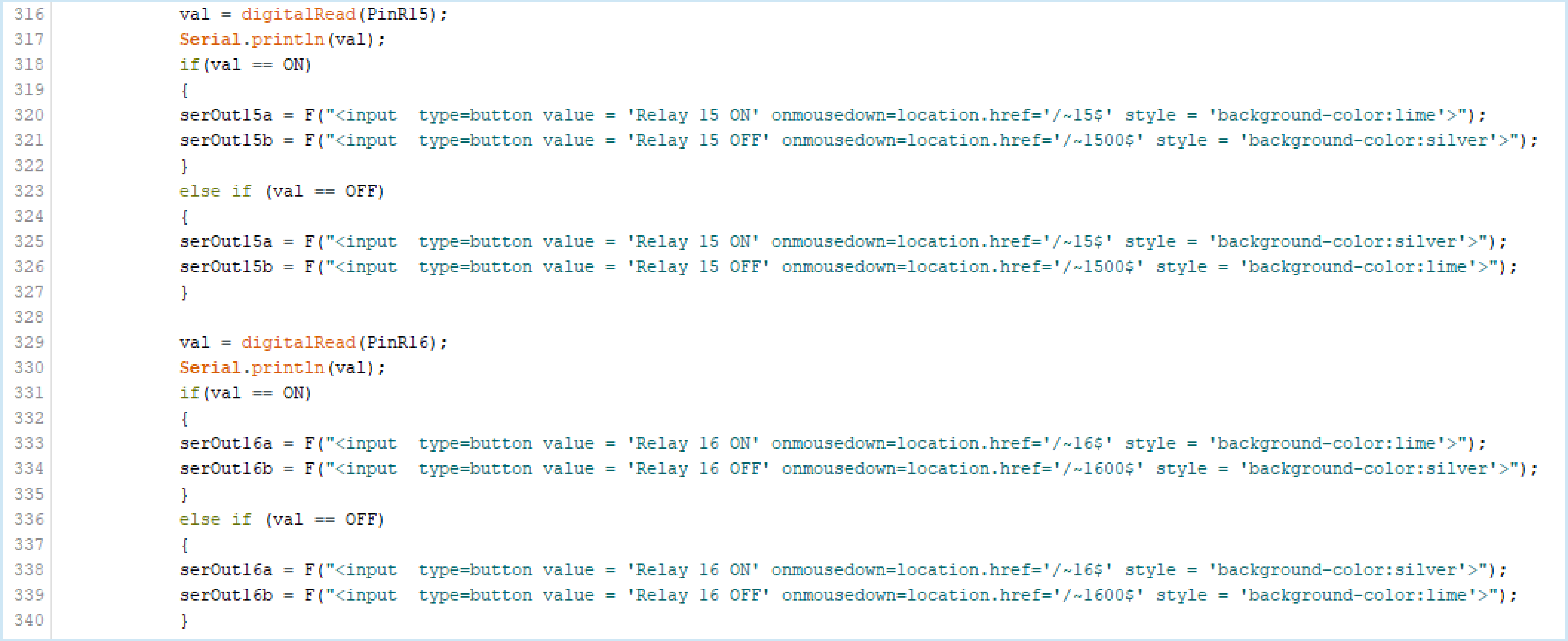


```
void sendReply()
  {

    //read all output pin values
            bool val = digitalRead(PinR1);
            Serial.println(val);
            if(val == ON)
            {
            serOut1a = F("<input  type=button value = 'WATTMETER' onmousedown=location.href='/~1$' style = 'background-color:lime'>");
            serOut1b = F("<input  type=button value = 'SWR METER' onmousedown=location.href='/~100$' style = 'background-color:silver'>");
            }
            else if (val == OFF)
            {
            serOut1a = F("<input  type=button value = 'WATTMETER' onmousedown=location.href='/~1$' style = 'background-color:silver'>");
            serOut1b = F("<input  type=button value = 'SWR METER' onmousedown=location.href='/~100$' style = 'background-color:lime'>");
            }

            val = digitalRead(PinR2);
            Serial.println(val);
            if(val == ON)
            {
            serOut2a = F("<input  type=button value = 'SWR-CAM ON'  onmousedown=location.href='/~2$' style = 'background-color:lime'>");
            serOut2b = F("<input  type=button value = 'SWR-CAM OFF' onmousedown=location.href='/~200$' style = 'background-color:silver'>");
            }
            else if (val == OFF)
            {
            serOut2a = F("<input  type=button value = 'SWR-CAM ON' onmousedown=location.href='/~2$' style = 'background-color:silver'>");
            serOut2b = F("<input  type=button value = 'SWR-CAM OFF' onmousedown=location.href='/~200$' style = 'background-color:lime'>");
            }
```

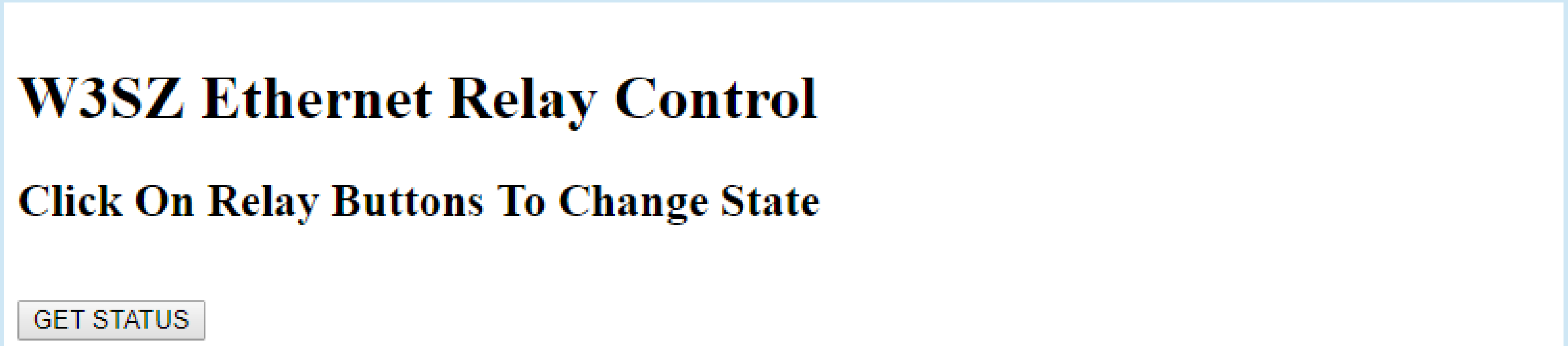
**F** macro tells the program to store the string in Flash memory rather than SRAM



```
            val = digitalRead(PinR3);
            Serial.println(val);
            if(val == ON)
            {
            serOut3a = F("<input  type=button value = 'WATT-CAM ON' style = 'background-color:lime' onmousedown=location.href='/~3$'>");
            serOut3b = F("<input  type=button value = 'WATT-CAM OFF' style = 'background-color:silver' onmousedown=location.href='/~300$'>");
            }
            else if (val == OFF)
            {
            serOut3a = F("<input  type=button value = 'WATT-CAM ON' onmousedown=location.href='/~3$' style = 'background-color:silver'>");
            serOut3b = F("<input  type=button value = 'WATT-CAM OFF' onmousedown=location.href='/~300$' style = 'background-color:lime'>");
            }

            val = digitalRead(PinR4);
            Serial.println(val);
            if(val == ON)
            {
            serOut4a = F("<input  type=button value = 'TX ANT ON' onmousedown=location.href='/~4$' style = 'background-color:lime'>");
            serOut4b = F("<input  type=button value = 'TX ANT OFF' onmousedown=location.href='/~400$' style = 'background-color:silver'>");
            }
            else if (val == OFF)
            {
            serOut4a = F("<input  type=button value = 'TX ANT ON' onmousedown=location.href='/~4$' style = 'background-color:silver'>");
            serOut4b = F("<input  type=button value = 'TX ANT OFF' onmousedown=location.href='/~400$' style = 'background-color:lime'>");
            }
```

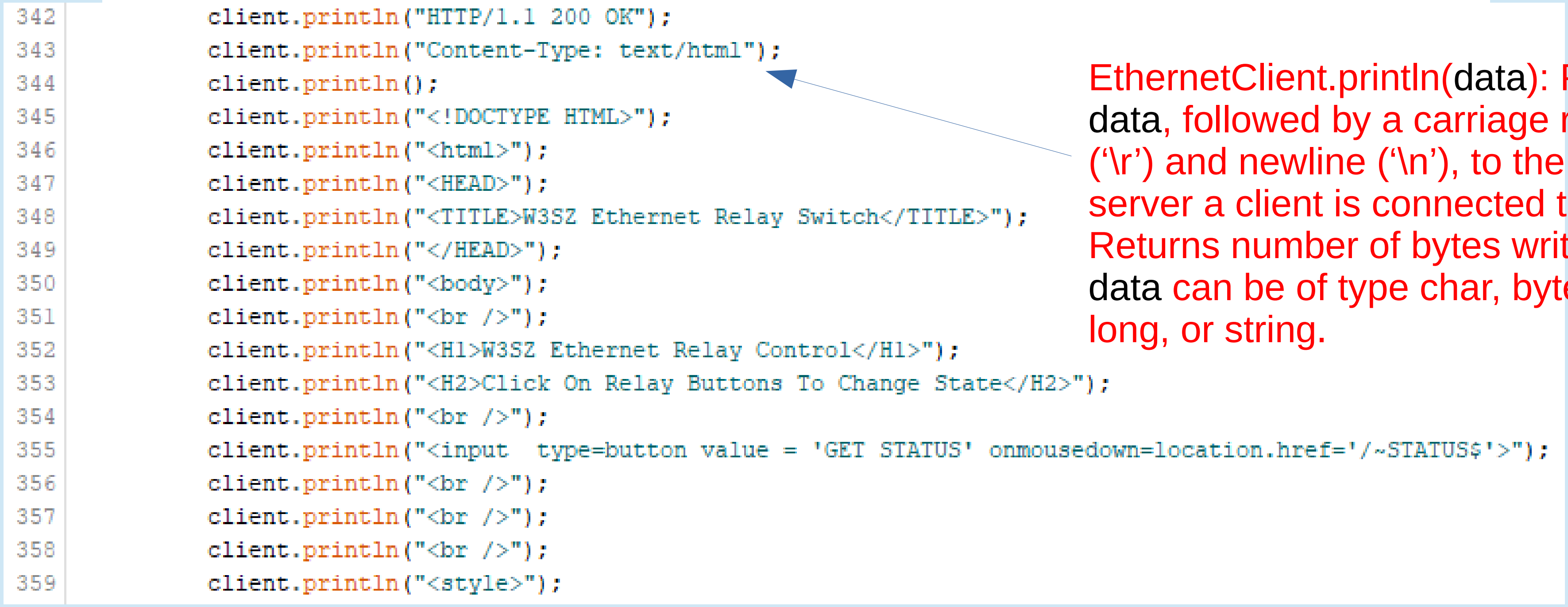


```
        val = digitalRead(PinR15);
        Serial.println(val);
        if(val == ON)
        {
        serOut15a = F("<input  type=button value = 'Relay 15 ON' onmousedown=location.href='/~15$' style = 'background-color:lime'>");
        serOut15b = F("<input  type=button value = 'Relay 15 OFF' onmousedown=location.href='/~1500$' style = 'background-color:silver'>");
        }
        else if (val == OFF)
        {
        serOut15a = F("<input  type=button value = 'Relay 15 ON' onmousedown=location.href='/~15$' style = 'background-color:silver'>");
        serOut15b = F("<input  type=button value = 'Relay 15 OFF' onmousedown=location.href='/~1500$' style = 'background-color:lime'>");
        }

        val = digitalRead(PinR16);
        Serial.println(val);
        if(val == ON)
        {
        serOut16a = F("<input  type=button value = 'Relay 16 ON' onmousedown=location.href='/~16$' style = 'background-color:lime'>");
        serOut16b = F("<input  type=button value = 'Relay 16 OFF' onmousedown=location.href='/~1600$' style = 'background-color:silver'>");
        }
        else if (val == OFF)
        {
        serOut16a = F("<input  type=button value = 'Relay 16 ON' onmousedown=location.href='/~16$' style = 'background-color:silver'>");
        serOut16b = F("<input  type=button value = 'Relay 16 OFF' onmousedown=location.href='/~1600$' style = 'background-color:lime'>");
        }
```



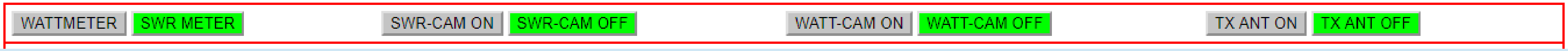
W3SZ Ethernet Relay Control

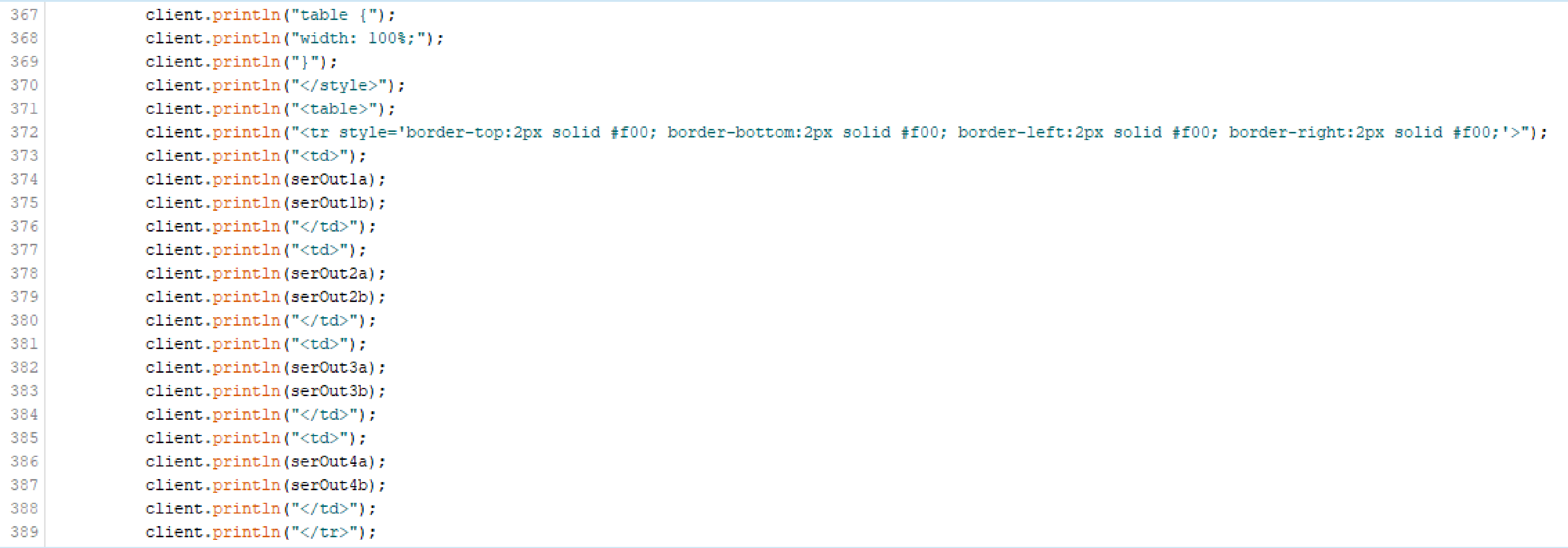
Click On Relay Buttons To Change State



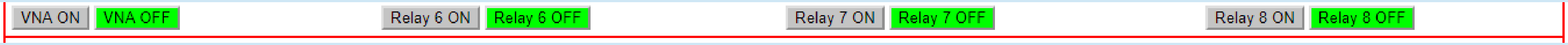
```
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();
        client.println("<!DOCTYPE HTML>");
        client.println("<html>");
        client.println("<HEAD>");
        client.println("<TITLE>W3SZ Ethernet Relay Switch</TITLE>");
        client.println("</HEAD>");
        client.println("<body>");
        client.println("<br />");
        client.println("<H1>W3SZ Ethernet Relay Control</H1>");
        client.println("<H2>Click On Relay Buttons To Change State</H2>");
        client.println("<br />");
        client.println("<input  type=button value = 'GET STATUS' onmousedown=location.href='/~STATUS$'>");
        client.println("<br />");
        client.println("<br />");
        client.println("<br />");
        client.println("<style>");
```

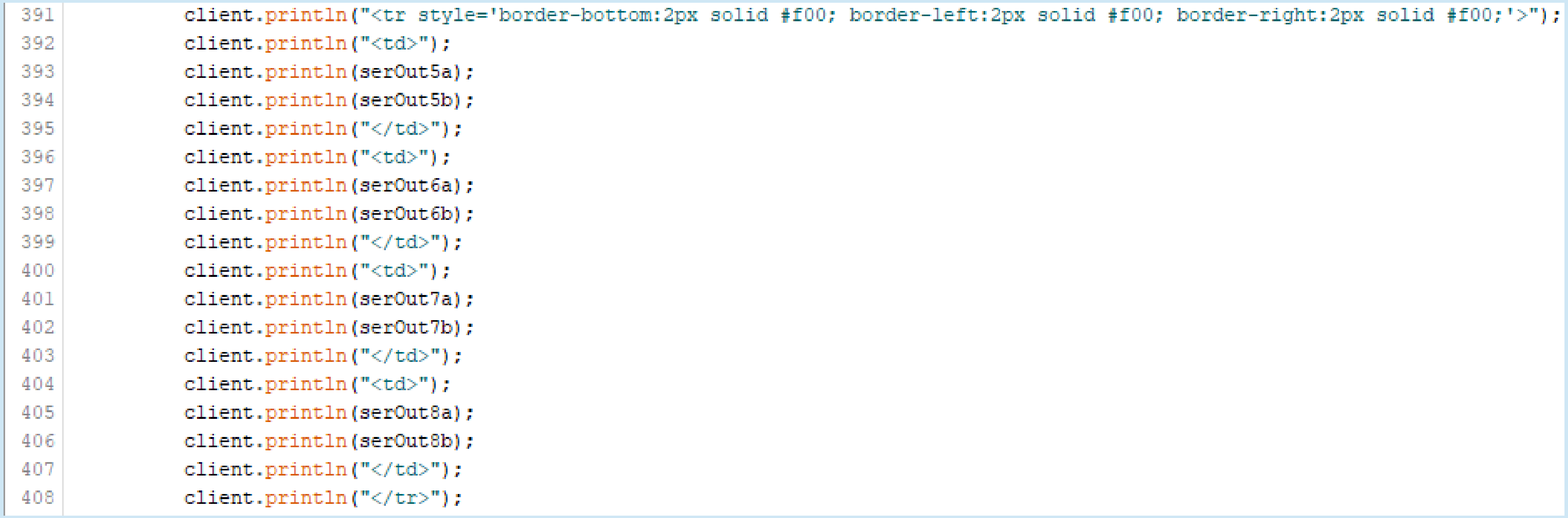
EthernetClient.println(data): Prints data, followed by a carriage return ('\r') and newline ('\n'), to the server a client is connected to. Returns number of bytes written. data can be of type char, byte, int, long, or string.



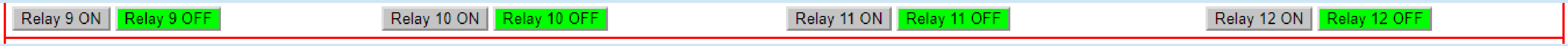


```
            client.println("table {");
            client.println("width: 100%;");
            client.println("}");
            client.println("</style>");
            client.println("<table>");
            client.println("<tr style='border-top:2px solid #f00; border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>");
            client.println("<td>");
            client.println(serOut1a);
            client.println(serOut1b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut2a);
            client.println(serOut2b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut3a);
            client.println(serOut3b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut4a);
            client.println(serOut4b);
            client.println("</td>");
            client.println("</tr>");
```

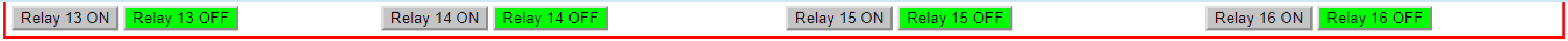


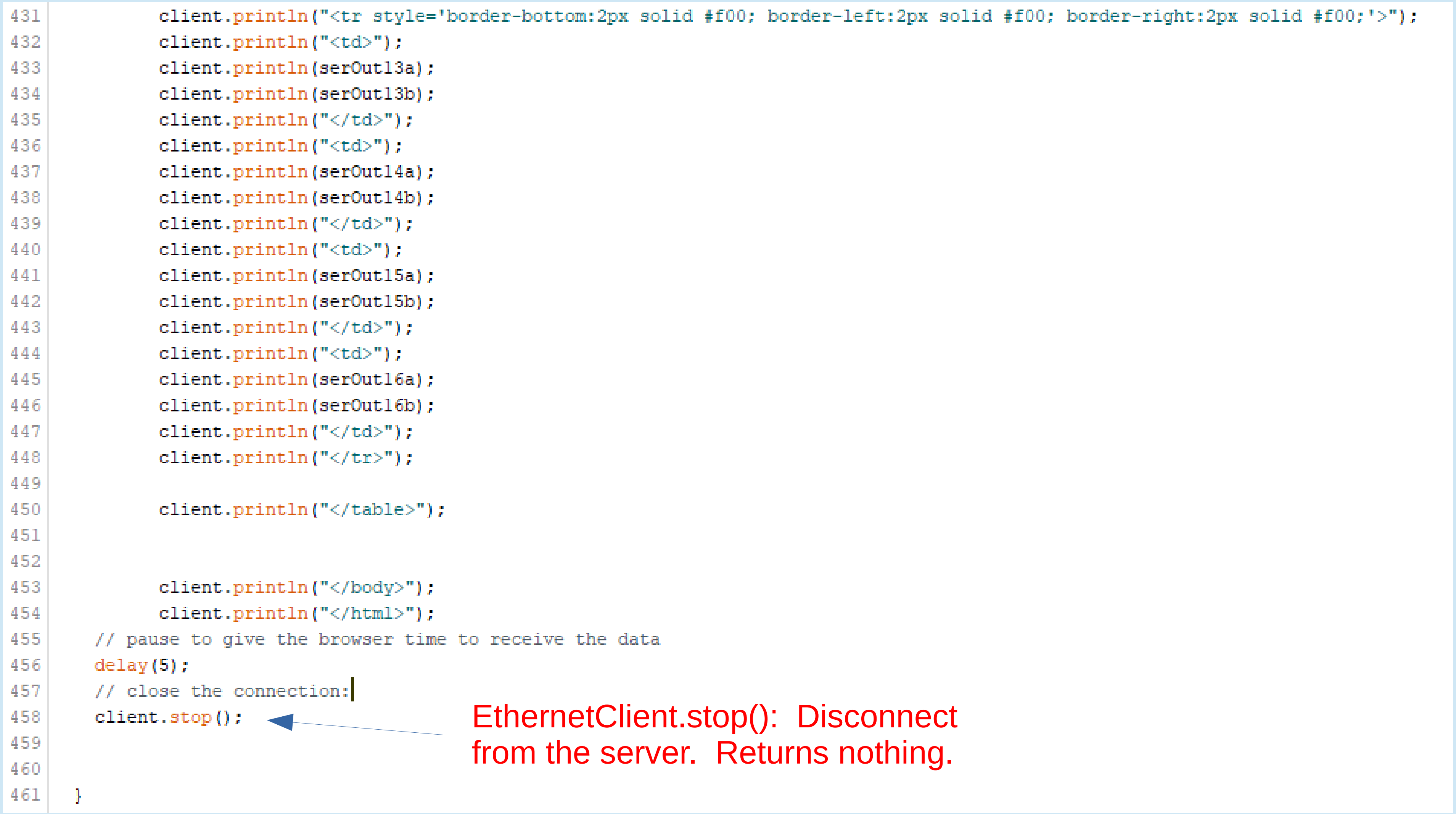


```
            client.println("<tr style='border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>");
            client.println("<td>");
            client.println(serOut5a);
            client.println(serOut5b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut6a);
            client.println(serOut6b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut7a);
            client.println(serOut7b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut8a);
            client.println(serOut8b);
            client.println("</td>");
            client.println("</tr>");
```



```
            client.println("<tr style='border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>
            client.println("<td>");
            client.println(serOut9a);
            client.println(serOut9b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut10a);
            client.println(serOut10b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut11a);
            client.println(serOut11b);
            client.println("</td>");
            client.println("<td>");
            client.println(serOut12a);
            client.println(serOut12b);
            client.println("</td>");
            client.println("</tr>");
```



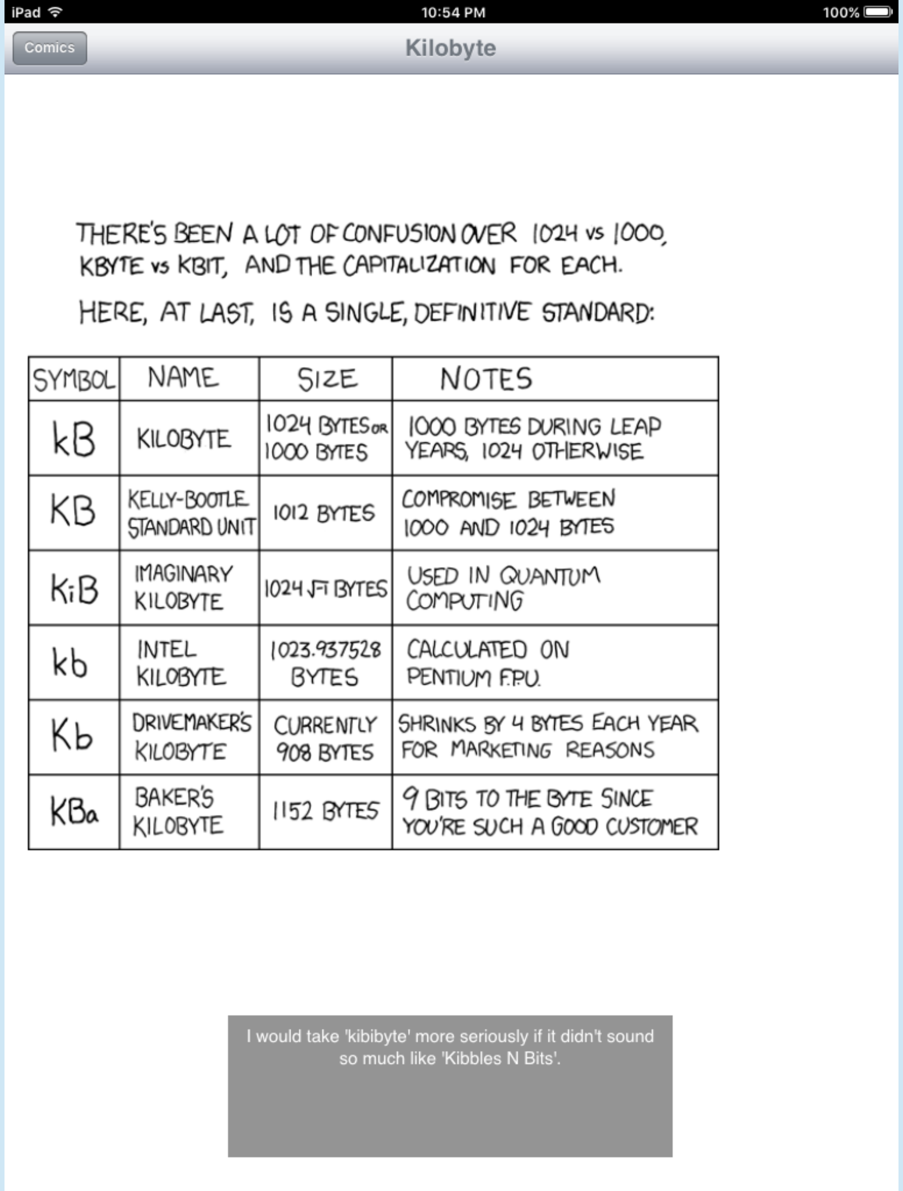


```
        client.println("<tr style='border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>");
        client.println("<td>");
        client.println(serOut13a);
        client.println(serOut13b);
        client.println("</td>");
        client.println("<td>");
        client.println(serOut14a);
        client.println(serOut14b);
        client.println("</td>");
        client.println("<td>");
        client.println(serOut15a);
        client.println(serOut15b);
        client.println("</td>");
        client.println("<td>");
        client.println(serOut16a);
        client.println(serOut16b);
        client.println("</td>");
        client.println("</tr>");

        client.println("</table>");


        client.println("</body>");
        client.println("</html>");
    // pause to give the browser time to receive the data
    delay(5);
    // close the connection:
    client.stop();


}
```

EthernetClient.stop(): Disconnect from the server. Returns nothing.

# Arduino Ethernet Device Control Example: Arduino Code

1) Included Libraries that are needed
2) Defined/initialized constants and variables
3) Setup()
   - Defined and initialized output pins
   - Started ethernet port and serial port
4) Loop()
   - Got ethernet data
   - Parsed ethernet data
   - Switched relays on or off
5) Called procedure “sendReply” to:
   - Send relay status back to client and re-write web page at client
   - (Web page used HTML buttons to send commands to Arduino to control relays and read relay status)



THERE'S BEEN A LOT OF CONFUSION OVER 1024 vs 1000, KBYTE vs KBIT, AND THE CAPITALIZATION FOR EACH.

HERE, AT LAST, IS A SINGLE, DEFINITIVE STANDARD:

| SYMBOL | NAME | SIZE | NOTES |
|---|---|---|---|
| kB | KILOBYTE | 1024 BYTES OR 1000 BYTES | 1000 BYTES DURING LEAP YEARS, 1024 OTHERWISE |
| KB | KELLY-BOOTLE STANDARD UNIT | 1012 BYTES | COMPROMISE BETWEEN 1000 AND 1024 BYTES |
| KiB | IMAGINARY KILOBYTE | 1024 $\sqrt{-1}$ BYTES | USED IN QUANTUM COMPUTING |
| kb | INTEL KILOBYTE | 1023.937528 BYTES | CALCULATED ON PENTIUM F.P.U. |
| Kb | DRIVEMAKER'S KILOBYTE | CURRENTLY 908 BYTES | SHRINKS BY 4 BYTES EACH YEAR FOR MARKETING REASONS |
| KBa | BAKER'S KILOBYTE | 1152 BYTES | 9 BITS TO THE BYTE SINCE YOU'RE SUCH A GOOD CUSTOMER |

I would take 'kibibyte' more seriously if it didn't sound so much like 'Kibbles N Bits'.