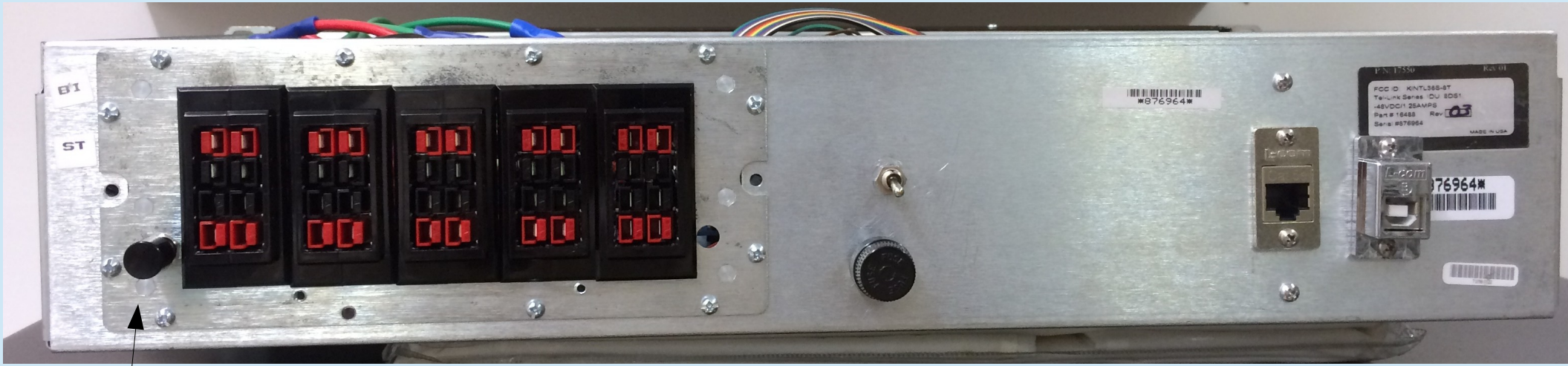
# Arduino Ethernet Device Control Example

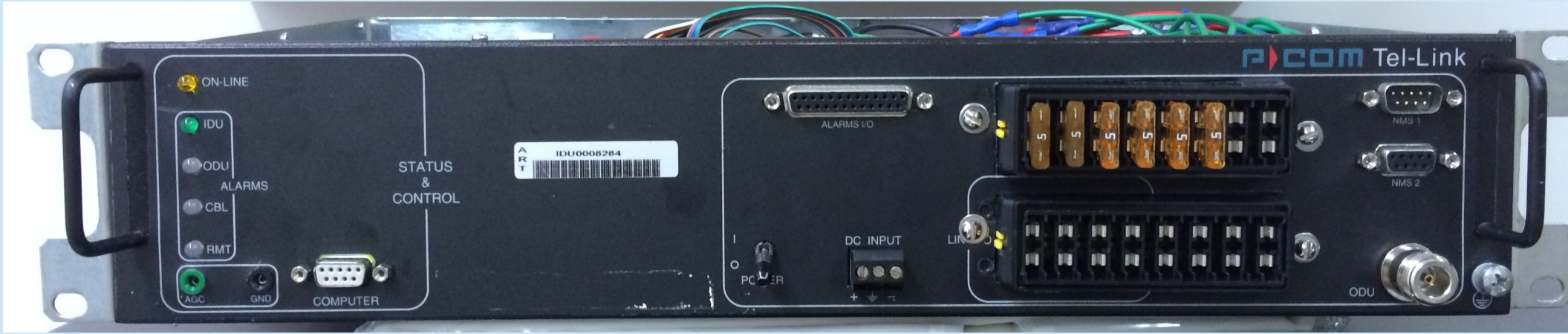
# Arduino Ethernet Device Control Example

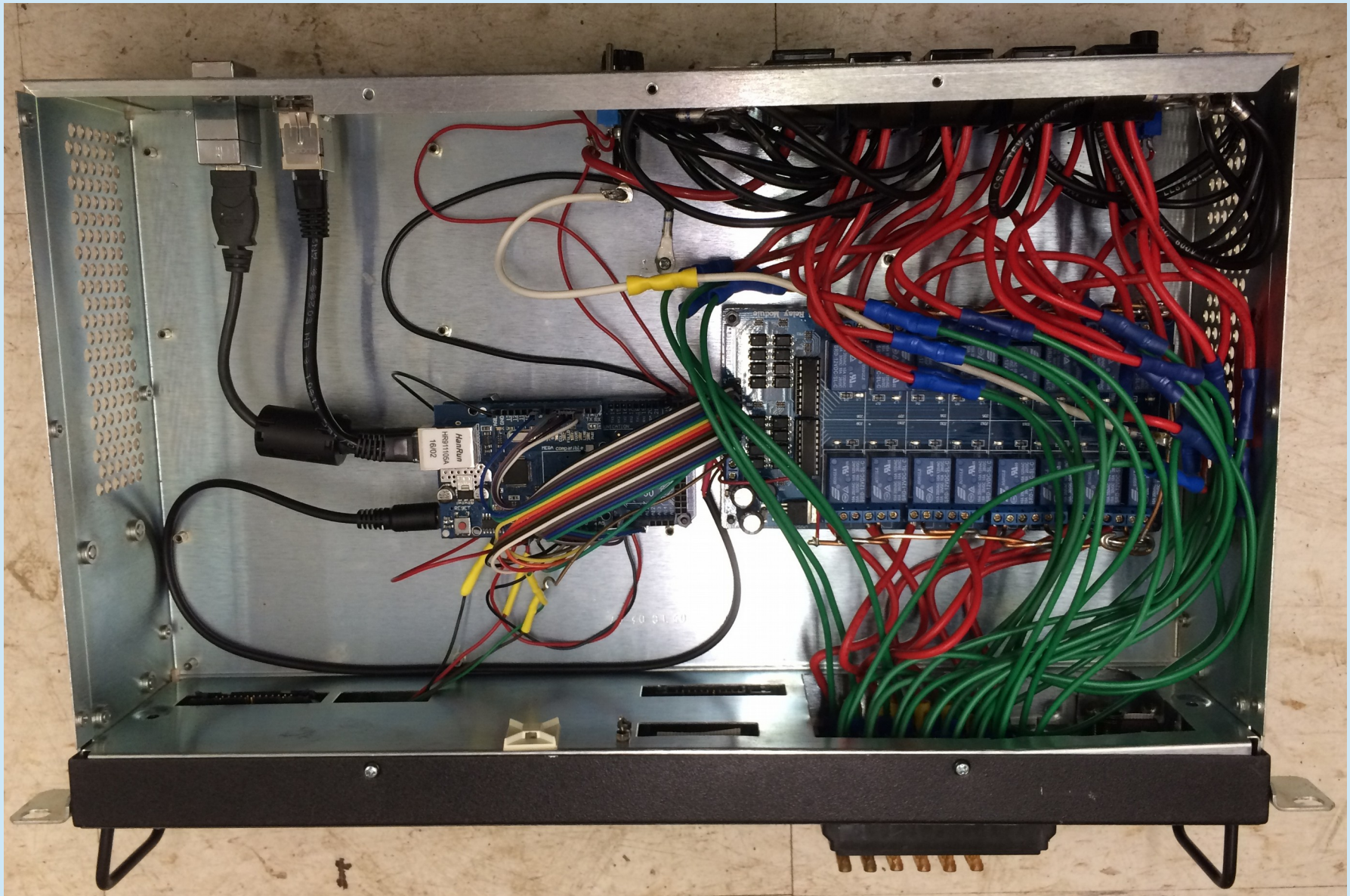
- Use Arduino to create a web page, provide on/off control for 16 devices via the Ethernet
  - Can use for power control,
  - transverter or antenna bandswitching,
  - switching mic, receive audio, foot switch, CW key, etc. among IF rigs
  - turning cameras on/off or switching between cameras

# 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


BI
ST
RESET Button


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


HanRun
HR911105A
16/02

# W3SZ Ethernet Relay Control

## Click On Relay Buttons To Change State

GET STATUS

| | | | |
|---|---|---|---|
| WATTMETER SWR METER | SWR-CAM ON SWR-CAM OFF | WATT-CAM ON WATT-CAM OFF | TX ANT ON TX ANT OFF |
| VNA ON VNA OFF | Relay 6 ON Relay 6 OFF | Relay 7 ON Relay 7 OFF | Relay 8 ON Relay 8 OFF |
| Relay 9 ON Relay 9 OFF | Relay 10 ON Relay 10 OFF | Relay 11 ON Relay 11 OFF | Relay 12 ON Relay 12 OFF |
| Relay 13 ON Relay 13 OFF | Relay 14 ON Relay 14 OFF | Relay 15 ON Relay 15 OFF | Relay 16 ON Relay 16 OFF |

# 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

```
 8
 9 #include <Ethernet.h>  //for ethernet port
10 #include <string.h> // for string handling
11
```

# Define Variables & Constants

```
12 String commandInputString = "";
13 String serIn;
14 String serOut1a;
15 String serOut2a;
16 String serOut3a;
17 String serOut4a;
18 String serOut1b;
19 String serOut2b;
20 String serOut3b;
21 String serOut4b;
22 String serOut5a;
23 String serOut6a;
24 String serOut7a;
25 String serOut8a;
26 String serOut9a;
27 String serOut5b;
28 String serOut6b;
29 String serOut7b;
30 String serOut8b;
31 String serOut9b;
```

```
32 String serOut10a;
33 String serOut11a;
34 String serOut12a;
35 String serOut13a;
36 String serOut14a;
37 String serOut15a;
38 String serOut16a;
39 String serOut10b;
40 String serOut11b;
41 String serOut12b;
42 String serOut13b;
43 String serOut14b;
44 String serOut15b;
45 String serOut16b;
46
47 const int ON = 1;
48 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.

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.

```
// 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
```

# **Setup:** Initialize GPIO Pins

```
78  void setup()
79  {
80    // initialize GPIO pins as output pins
81    pinMode(PinR1, OUTPUT);
82    pinMode(PinR2, OUTPUT);
83    pinMode(PinR3, OUTPUT);
84    pinMode(PinR4, OUTPUT);
85    pinMode(PinR5, OUTPUT);
86    pinMode(PinR6, OUTPUT);
87    pinMode(PinR7, OUTPUT);
88    pinMode(PinR8, OUTPUT);
89    pinMode(PinR9, OUTPUT);
90    pinMode(PinR10, OUTPUT);
91    pinMode(PinR11, OUTPUT);
92    pinMode(PinR12, OUTPUT);
93    pinMode(PinR13, OUTPUT);
94    pinMode(PinR14, OUTPUT);
95    pinMode(PinR15, OUTPUT);
96    pinMode(PinR16, OUTPUT);
```

```
 98    //initialize all GPIO pin values to OFF
 99    digitalWrite(PinR1, OFF);
100    digitalWrite(PinR2, OFF);
101    digitalWrite(PinR3, OFF);
102    digitalWrite(PinR4, OFF);
103    digitalWrite(PinR5, OFF);
104    digitalWrite(PinR6, OFF);
105    digitalWrite(PinR7, OFF);
106    digitalWrite(PinR8, OFF);
107    digitalWrite(PinR9, OFF);
108    digitalWrite(PinR10, OFF);
109    digitalWrite(PinR11, OFF);
110    digitalWrite(PinR12, OFF);
111    digitalWrite(PinR13, OFF);
112    digitalWrite(PinR14, OFF);
113    digitalWrite(PinR15, OFF);
114    digitalWrite(PinR16, OFF);
```

# **Setup:** Start Ethernet Port

```
116   // start the Ethernet connection and the server and the serial port:
117   Ethernet.begin(mac, ip);
118   server.begin();
119   Serial.begin(9600);
120   Serial.println("Arduino Ethernet Device Switch");
121   Serial.println("by W3SZ");
122   Serial.println("Starting Server");
123   Serial.println (Ethernet.localIP());
124
125
126 }
```

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

```
470 void loop()
471 {
472   // listen for incoming client
473   client = server.available();
474   if (client) {
475     while (client.connected()) {
476         char c = client.read();
477         commandInputString += c;  //append latest character received to string
478       if (c == '\n')
479       {
480         //Checks for the URL string beginning with '~' and ending with '$'
481          int stringStart = commandInputString.indexOf('~');
482          int stringEnd = commandInputString.indexOf('$');
483          String commandOut = commandInputString.substring(1 + stringStart,stringEnd);
484          Serial.print("Command is: ");
485          Serial.println(commandOut);
486          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

```
470 void loop()
471 {
472   // listen for incoming client
473   client = server.available();
474   if (client) {
475     while (client.connected()) {
476         char c = client.read();
477         commandInputString += c;  //append latest character received to string
478       if (c == '\n')
479       {
480         //Checks for the URL string beginning with '~' and ending with '$'
481         int stringStart = commandInputString.indexOf('~');
482         int stringEnd = commandInputString.indexOf('$');
483         String commandOut = commandInputString.substring(1 + stringStart,stringEnd);
484           Serial.print("Command is: ");
485           Serial.println(commandOut);
486           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?

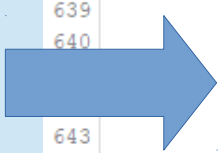
- http://www.instructables.com/id/Arduino-Ethernet-Shield-Tutorial/ has an example that turns LED on and off via the ethernet...a perfect beginning for this project!
- Original Arduino code is here: http://w3sz.com/EthernetLED_Switch.ino
- Remember, if you start by stealing someone else's code, you will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

# Switch the relays

```
488           if (commandOut  == "1") {
489             digitalWrite(PinR1, ON);
490             sendReply();
491           }
492           else if (commandOut  == "100") {
493             digitalWrite(PinR1, OFF);
494             sendReply();
495           }
496
497           else if (commandOut  == "2") {
498             digitalWrite(PinR2, ON);
499             sendReply();
500           }
501           else if (commandOut  == "200") {
502             digitalWrite(PinR2, OFF);
503             sendReply();
504           }
505
506           else if (commandOut  == "3") {
507             digitalWrite(PinR3, ON);
508             sendReply();
509           }
510           else if (commandOut  == "300") {
511             digitalWrite(PinR3, OFF);
512             sendReply();
513           }
```

```
515         else if (commandOut  == "4") {
516           digitalWrite(PinR4, ON);
517           sendReply();
518         }
519         else if (commandOut  == "400") {
520           digitalWrite(PinR4, OFF);
521           sendReply();
522         }
523
524         else if (commandOut  == "5") {
525           digitalWrite(PinR5, ON);
526           sendReply();
527         }
528         else if (commandOut  == "500") {
529           digitalWrite(PinR5, OFF);
530           sendReply();
531         }
532
533         else if (commandOut  == "6") {
534           digitalWrite(PinR6, ON);
535           sendReply();
536         }
537         else if (commandOut  == "600") {
538           digitalWrite(PinR6, OFF);
539           sendReply();
540         }
```

What about SendReply?



```
615           else if (commandOut  == "15") {
616             digitalWrite(PinR15, ON);
617             sendReply();
618           }
619           else if (commandOut  == "1500") {
620             digitalWrite(PinR15, OFF);
621             sendReply();
622           }
623
624           else if (commandOut  == "16") {
625             digitalWrite(PinR16, ON);
626             sendReply();
627           }
628           else if (commandOut  == "1600") {
629             digitalWrite(PinR16, OFF);
630             sendReply();
631           }
632
633           else if (commandOut  == "STATUS") {
634             sendReply();
635           }
636           else
637           {
638             String HTMString = "Command Not Recognized:  ";
639             Serial.println(commandOut);
640             Serial.println(HTMString);
641             sendReply();
642           }
643
644           commandInputString = "";
645           commandOut = "";
646           c=' ';
647
648       }
649     }
650   }
651 }
```

# 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

```
130 void sendReply()
131   {
132
133     //read all output pin values
134             bool val = digitalRead(PinR1);
135             Serial.println(val);
136             if(val == ON)
137             {
138             serOut1a = F("<input  type=button value = 'WATTMETER' onmousedown=location.href='/~1$' style = 'background-color:lime'>");
139             serOut1b = F("<input  type=button value = 'SWR METER' onmousedown=location.href='/~100$' style = 'background-color:silver'>");
140             }
141             else if (val == OFF)
142             {
143             serOut1a = F("<input  type=button value = 'WATTMETER' onmousedown=location.href='/~1$' style = 'background-color:silver'>");
144             serOut1b = F("<input  type=button value = 'SWR METER' onmousedown=location.href='/~100$' style = 'background-color:lime'>");
145             }
146
147             val = digitalRead(PinR2);
148             Serial.println(val);
149             if(val == ON)
150             {
151             serOut2a = F("<input  type=button value = 'SWR-CAM ON'  onmousedown=location.href='/~2$' style = 'background-color:lime'>");
152             serOut2b = F("<input  type=button value = 'SWR-CAM OFF' onmousedown=location.href='/~200$' style = 'background-color:silver'>");
153             }
154             else if (val == OFF)
155             {
156             serOut2a = F("<input  type=button value = 'SWR-CAM ON' onmousedown=location.href='/~2$' style = 'background-color:silver'>");
157             serOut2b = F("<input  type=button value = 'SWR-CAM OFF' onmousedown=location.href='/~200$' style = 'background-color:lime'>");
158             }
```

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

```
160             val = digitalRead(PinR3);
161             Serial.println(val);
162             if(val == ON)
163             {
164             serOut3a = F("<input  type=button value = 'WATT-CAM ON' style = 'background-color:lime' onmousedown=location.href='/~3$'>");
165             serOut3b = F("<input  type=button value = 'WATT-CAM OFF' style = 'background-color:silver' onmousedown=location.href='/~300$'>");
166             }
167             else if (val == OFF)
168             {
169             serOut3a = F("<input  type=button value = 'WATT-CAM ON' onmousedown=location.href='/~3$' style = 'background-color:silver'>");
170             serOut3b = F("<input  type=button value = 'WATT-CAM OFF' onmousedown=location.href='/~300$' style = 'background-color:lime'>");
171             }
172
173             val = digitalRead(PinR4);
174             Serial.println(val);
175             if(val == ON)
176             {
177             serOut4a = F("<input  type=button value = 'TX ANT ON' onmousedown=location.href='/~4$' style = 'background-color:lime'>");
178             serOut4b = F("<input  type=button value = 'TX ANT OFF' onmousedown=location.href='/~400$' style = 'background-color:silver'>");
179             }
180             else if (val == OFF)
181             {
182             serOut4a = F("<input  type=button value = 'TX ANT ON' onmousedown=location.href='/~4$' style = 'background-color:silver'>");
183             serOut4b = F("<input  type=button value = 'TX ANT OFF' onmousedown=location.href='/~400$' style = 'background-color:lime'>");
184             }
```

```
316             val = digitalRead(PinR15);
317             Serial.println(val);
318             if(val == ON)
319             {
320             serOut15a = F("<input  type=button value = 'Relay 15 ON' onmousedown=location.href='/~15$' style = 'background-color:lime'>");
321             serOut15b = F("<input  type=button value = 'Relay 15 OFF' onmousedown=location.href='/~1500$' style = 'background-color:silver'>");
322             }
323             else if (val == OFF)
324             {
325             serOut15a = F("<input  type=button value = 'Relay 15 ON' onmousedown=location.href='/~15$' style = 'background-color:silver'>");
326             serOut15b = F("<input  type=button value = 'Relay 15 OFF' onmousedown=location.href='/~1500$' style = 'background-color:lime'>");
327             }
328
329             val = digitalRead(PinR16);
330             Serial.println(val);
331             if(val == ON)
332             {
333             serOut16a = F("<input  type=button value = 'Relay 16 ON' onmousedown=location.href='/~16$' style = 'background-color:lime'>");
334             serOut16b = F("<input  type=button value = 'Relay 16 OFF' onmousedown=location.href='/~1600$' style = 'background-color:silver'>");
335             }
336             else if (val == OFF)
337             {
338             serOut16a = F("<input  type=button value = 'Relay 16 ON' onmousedown=location.href='/~16$' style = 'background-color:silver'>");
339             serOut16b = F("<input  type=button value = 'Relay 16 OFF' onmousedown=location.href='/~1600$' style = 'background-color:lime'>");
340             }
```

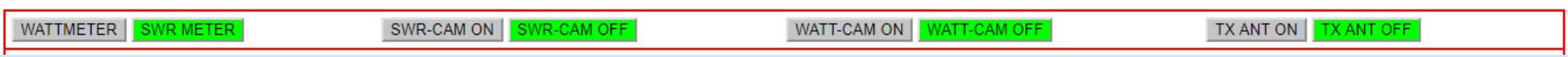
# W3SZ Ethernet Relay Control

## Click On Relay Buttons To Change State

GET STATUS

```
342         client.println("HTTP/1.1 200 OK");
343         client.println("Content-Type: text/html");
344         client.println();
345         client.println("<!DOCTYPE HTML>");
346         client.println("<html>");
347         client.println("<HEAD>");
348         client.println("<TITLE>W3SZ Ethernet Relay Switch</TITLE>");
349         client.println("</HEAD>");
350         client.println("<body>");
351         client.println("<br />");
352         client.println("<H1>W3SZ Ethernet Relay Control</H1>");
353         client.println("<H2>Click On Relay Buttons To Change State</H2>");
354         client.println("<br />");
355         client.println("<input  type=button value = 'GET STATUS' onmousedown=location.href='/~STATUS$'>");
356         client.println("<br />");
357         client.println("<br />");
358         client.println("<br />");
359         client.println("<style>");
```

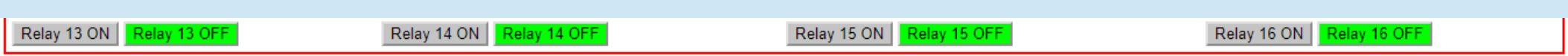
EthernetClient.println(data): Prints data, followed by a carriage return (‘\r’) and newline (‘\n’), to the server a client is connected to. Returns number of bytes written. data can be of type char, byte, int, long, or string.




```
367         client.println("table {");
368         client.println("width: 100%;");
369         client.println("}");
370         client.println("</style>");
371         client.println("<table>");
372         client.println("<tr style='border-top:2px solid #f00; border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>");
373         client.println("<td>");
374         client.println(serOut1a);
375         client.println(serOut1b);
376         client.println("</td>");
377         client.println("<td>");
378         client.println(serOut2a);
379         client.println(serOut2b);
380         client.println("</td>");
381         client.println("<td>");
382         client.println(serOut3a);
383         client.println(serOut3b);
384         client.println("</td>");
385         client.println("<td>");
386         client.println(serOut4a);
387         client.println(serOut4b);
388         client.println("</td>");
389         client.println("</tr>");
```


```
391         client.println("<tr style='border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>");
392         client.println("<td>");
393         client.println(serOut5a);
394         client.println(serOut5b);
395         client.println("</td>");
396         client.println("<td>");
397         client.println(serOut6a);
398         client.println(serOut6b);
399         client.println("</td>");
400         client.println("<td>");
401         client.println(serOut7a);
402         client.println(serOut7b);
403         client.println("</td>");
404         client.println("<td>");
405         client.println(serOut8a);
406         client.println(serOut8b);
407         client.println("</td>");
408         client.println("</tr>");
```


```
411         client.println("<tr style='border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>
412         client.println("<td>");
413         client.println(serOut9a);
414         client.println(serOut9b);
415         client.println("</td>");
416         client.println("<td>");
417         client.println(serOut10a);
418         client.println(serOut10b);
419         client.println("</td>");
420         client.println("<td>");
421         client.println(serOut11a);
422         client.println(serOut11b);
423         client.println("</td>");
424         client.println("<td>");
425         client.println(serOut12a);
426         client.println(serOut12b);
427         client.println("</td>");
428         client.println("</tr>");
```




```
431           client.println("<tr style='border-bottom:2px solid #f00; border-left:2px solid #f00; border-right:2px solid #f00;'>");
432           client.println("<td>");
433           client.println(serOut13a);
434           client.println(serOut13b);
435           client.println("</td>");
436           client.println("<td>");
437           client.println(serOut14a);
438           client.println(serOut14b);
439           client.println("</td>");
440           client.println("<td>");
441           client.println(serOut15a);
442           client.println(serOut15b);
443           client.println("</td>");
444           client.println("<td>");
445           client.println(serOut16a);
446           client.println(serOut16b);
447           client.println("</td>");
448           client.println("</tr>");
449
450           client.println("</table>");
451
452
453           client.println("</body>");
454           client.println("</html>");
455     // pause to give the browser time to receive the data
456     delay(5);
457     // close the connection:
458     client.stop();
459
460
461   }
```

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'.