

HTML

Objektleiste

↓
1/TV 4/PLAY
Sevent/Typ/ Befehl

- TV
- Steckdose
- Hifi

Objekte/subklasse

- | | | |
|-------------|-------------|----------|
| • TV | • Steckdose | • Hifi |
| - (Status) | - Ein | - Lauter |
| - Play | - Aus | - Leiser |
| - Lauter | - Alle Aus | - Mute |
| - Leiser | | - Kanal |
| - Kanal 0-9 | | |

Klassen

IR

Funk

Methode

IR SEND(MEC(32/x84))

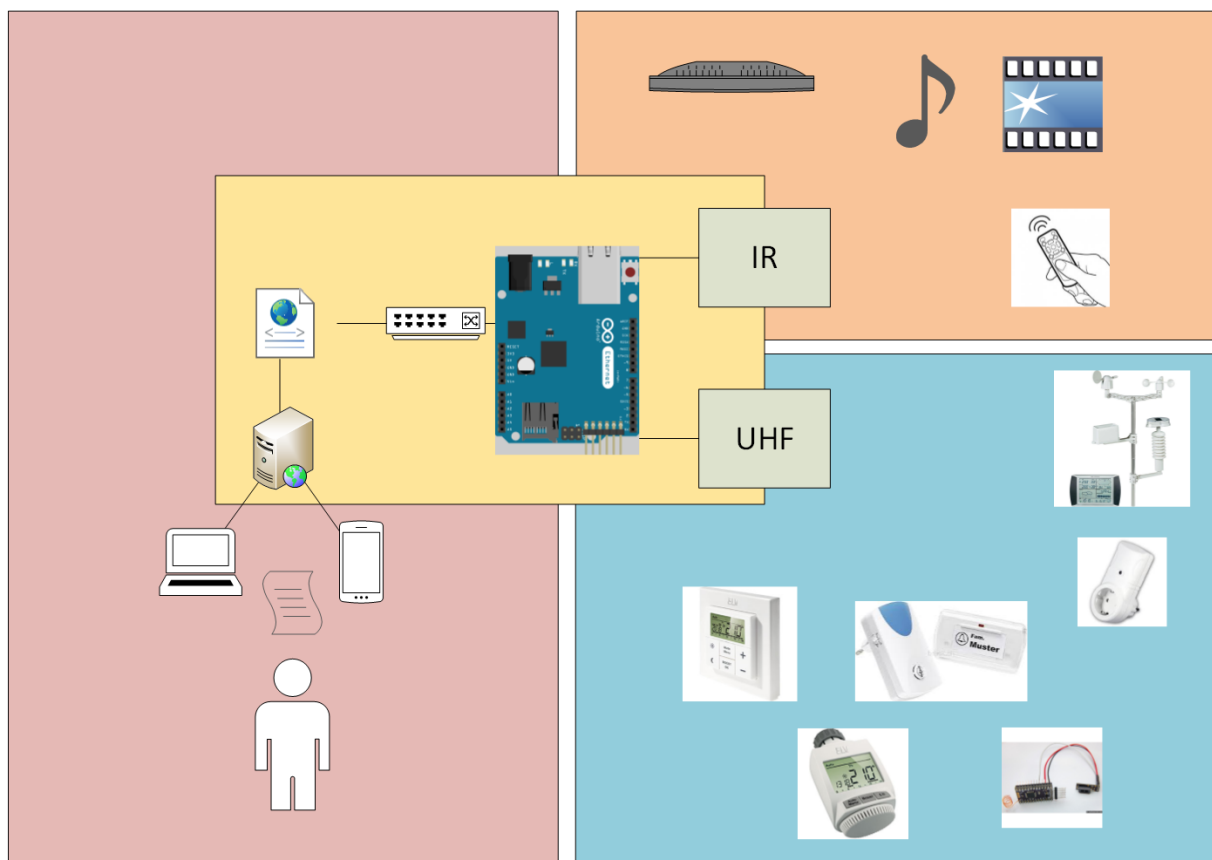
Arduino Projekt

David Arenz & Matthias Lehmann

Ziel des Projektes:

Schaffung einer universellen Plattform zur Hausautomatisierung

1. Einbinden verschiedener Steuerungssystemen aus dem Konsumerbereich
 - Verwirklichung eines Universeller Infrarot Sender/Empfängers
 - TV, Hifi, Lampen usw.
 - Beliebige Fernbedienungen als Bedienelemente
 - Einbindung von UHF Funktransceivern (433/866 MHz)
 - Funksteckdosen, Funkdimmer
 - Empfang von Wetterdaten
2. Ansteuerung verschiedener Systeme bündeln
 - API ähnliche Befehle
 - Abarbeiten von Befehlsketten
 - Ggf. Überwachung und Regelung von Parametern
3. HMI Schnittstelle per Webserver



1 SD Karte

Problem: Fehler: Initialisierung der SD-Karte fehlgeschlagen!

Lösung: SD Karte komplett entfernen, SD Karte einstecken, 3s lang den REST Button drücken

2 Code

2.1 WEB_SD_IR.ino

Listing 1: ../code/WEB_SD_IR/WEB_SD_IR.ino

```
1  #include <SPI.h>
   #include <Ethernet.h>
3  #include <IRremote.h>
   #include <TextFinder.h>
5  #include <SD.h>

7  // #### Voraussetzungen ####
   // TSOP Signal-Pin <==> Arduino - Pin 11
9  // IR-LED Anode <==> Arduino - Pin 3
   // Test-LED <==> Arduino - Pin 6

11
   class ControlProxy
13  {
       IRsend _infrared_sender;
15  void read_line(EthernetClient& client, char* buffer, const int buffer_length)
       {
17         int buffer_pos = 0;
           while (client.available() && (buffer_pos < buffer_length - 1))
19         {
               const char c = client.read();
21               if (c == '\n')
                   break;
23               if (c != '\r')
                   buffer[buffer_pos++] = c;
25         }
           buffer[buffer_pos] = '\0';
27     }
       bool send_ir_data(const char* protocol, const int bits, const long value)
29     {
           bool result = true;
31         if (!strcasecmp(protocol, "NEC"))
             _infrared_sender.sendNEC(value, bits);
33         else if (!strcasecmp(protocol, "SONY"))
             _infrared_sender.sendSony(value, bits);
35         else if (!strcasecmp(protocol, "RC5"))
             _infrared_sender.sendRC5(value, bits);
37         else if (!strcasecmp(protocol, "RC6"))
             _infrared_sender.sendRC6(value, bits);
39         else
             result = false;
41         return result;
       }
43     bool handle_command(char* line)
       {
45         strsep(&line, " ");
           char* path = strsep(&line, " ");
47         char* args[3];
           for (char** ap = args; (*ap = strsep(&path, "/")) != NULL;)
49             if (**ap != '\0')
                 if (++ap >= &args[3])
```

```

51         break;
52     const int bits = atoi(args[1]);
53     const long value = atol(args[2]);
54     return send_ir_data(args[0], bits, value);
55 }
56 public:
57     void receive_from_server(EthernetServer server)
58     {
59         const int MAXLINE = 256;
60         char line[MAXLINE];
61         EthernetClient client = server.available();
62         File webFile; // Datei mit HTML-Inhalt
63         if (client)
64         {
65             while (client.connected())
66             {
67                 if (client.available())
68                 {
69                     read_line(client, line, MAXLINE);
70                     Serial.println(line);
71                     if (line[0] == 'G' && line[1] == 'E' && line[2] == 'T')
72                         handle_command(line);
73                     if (!strcmp(line, ""))
74                     {
75                         // Standard HTTP Header senden
76                         client.println("HTTP/1.1 200 OK");
77                         client.println("Content-Type: text/html");
78                         client.println("Connection: close");
79                         client.println();
80
81                         // Website von SD-Karte laden
82                         webFile = SD.open("sd.htm"); // Website laden
83                         if (webFile)
84                         {
85                             while(webFile.available())
86                             {
87                                 client.write(webFile.read()); // Website an Client schicken
88                             }
89                             webFile.close();
90                         }
91                         break;
92                     }
93                 }
94             }
95             delay(1);
96             client.stop();
97         }
98     }
99 };
100 //—— ENDE DER DEKLARATION ——
101 const unsigned int PROXY_PORT = 80;
102 const unsigned int BAUD_RATE = 19200;
103 byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xDB, 0xAE }; // MAC Arduino Ethernet (David)
104 byte ip[] = { 192, 168, 3, 100 };
105 byte sdPin = 4; // Pin der SD-Karte
106 EthernetServer server(PROXY_PORT);
107 ControlProxy ir_proxy;
108 void setup()
109 {
110     // Open serial communications and wait for port to open:
111     Serial.begin(BAUD_RATE);

```

```

// start the Ethernet connection and the server:
113 Ethernet.begin(mac);
    Serial.print("Server is at: ");
115 Serial.println(Ethernet.localIP());
server.begin(); // Server starten
117
Serial.println("ARDUINO - STEUERUNG");
119 Serial.println("Initialisiere SD-Karte...");
if (!SD.begin(sdPin))
121 {
    Serial.println(" - Initialisierung der SD-Karte fehlgeschlagen!");
123 return;
}
125 Serial.println(" - SD-Karte erfolgreich initialisiert.");

127 if (!SD.exists("sd.htm"))
{
129 Serial.println(" - Datei (sd.htm) wurde nicht gefunden!");
return;
131 }
Serial.println(" - Datei (sd.htm) wurde gefunden.");
133
Serial.println();
135 Serial.println("Verbraucher schalten");
}
137 void loop()
{
139 ir_proxy.receive_from_server(server);
}

```

3 Beispiele

3.1 InfraredDumper.ino

Listing 2: ../example/InfraredDumper/InfraredDumper.ino

```
1  #include <IRremote.h>

3  const unsigned int IR_RECEIVER_PIN = 11;
   const unsigned int BAUD_RATE = 19200;

5
   IRrecv ir_receiver(IR_RECEIVER_PIN);
7   decode_results results;

9   void setup()
   {
11       Serial.begin(BAUD_RATE);
        ir_receiver.enableIRIn();
13   }

15   void dump(const decode_results* results)
   {
17       const int protocol = results->decode_type;
        Serial.print("Protocol: ");
19       if (protocol == UNKNOWN)
        {
21         Serial.println("not recognized.");
        }
23       else
        {
25         if (protocol == NEC)
            {
27             Serial.println("NEC");
            }
29         else if (protocol == SONY)
            {
31             Serial.println("SONY");
            }
33         else if (protocol == RC5)
            {
35             Serial.println("RC5");
            }
37         else if (protocol == RC6)
            {
39             Serial.println("RC6");
            }
41         Serial.print("Value: ");
            Serial.print(results->value, HEX);
43         Serial.print(" (");
            Serial.print(results->bits, DEC);
45         Serial.println(" bits)");
        }
47   }

49   void loop()
   {
51       if (ir_receiver.decode(&results))
        {
53         dump(&results);
            ir_receiver.resume();
55       }
```


3.2 InfraredProxy.ino

Listing 3: ../example/InfraredProxy/InfraredProxy.ino

```
#include <SPI.h>
2 #include <Ethernet.h>
#include <IRremote.h>
4 // ### Voraussetzungen ###
// TSOP Signal-Pin <—> Arduino - Pin 11
6 // IR-LED Anode <—> Arduino - Pin 3
class InfraredProxy
8 {
    IRsend _infrared_sender;
10 void read_line(EthernetClient& client, char* buffer, const int buffer_length)
    {
12         int buffer_pos = 0;
        while (client.available() && (buffer_pos < buffer_length - 1))
14         {
            const char c = client.read();
16             if (c == '\n')
                break;
18             if (c != '\r')
                buffer[buffer_pos++] = c;
20         }
        buffer[buffer_pos] = '\0';
22     }
    bool send_ir_data(const char* protocol, const int bits, const long value)
24     {
        bool result = true;
26         if (!strcasecmp(protocol, "NEC"))
            _infrared_sender.sendNEC(value, bits);
28         else if (!strcasecmp(protocol, "SONY"))
            _infrared_sender.sendSony(value, bits);
30         else if (!strcasecmp(protocol, "RC5"))
            _infrared_sender.sendRC5(value, bits);
32         else if (!strcasecmp(protocol, "RC6"))
            _infrared_sender.sendRC6(value, bits);
34         else
            result = false;
36         return result;
    }
38 bool handle_command(char* line)
    {
40         strsep(&line, " ");
        char* path = strsep(&line, " ");
42         char* args[3];
        for (char** ap = args; (*ap = strsep(&path, "/")) != NULL;)
44             if (**ap != '\0')
                if (++ap >= &args[3])
46                 break;
        const int bits = atoi(args[1]);
48         const long value = atol(args[2]);
        return send_ir_data(args[0], bits, value);
50     }
public:
52 void receive_from_server(EthernetServer server)
    {
54         const int MAXLINE = 256;
        char line[MAXLINE];
56         EthernetClient client = server.available();
        if (client)
```



```

58     {
60         while (client.connected())
61         {
62             if (client.available())
63             {
64                 read_line(client, line, MAX_LINE);
65                 Serial.println(line);
66                 if (line[0] == 'G' && line[1] == 'E' && line[2] == 'T')
67                     handle_command(line);
68                 if (!strcmp(line, ""))
69                 {
70                     client.println("HTTP/1.1 200 OK\n");
71                     break;
72                 }
73             }
74         }
75         delay(1);
76         client.stop();
77     }
78 };
79 //— ENDE DER DEKLARATION —
80 const unsigned int PROXY_PORT = 80;
81 const unsigned int BAUD_RATE = 19200;
82 byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xDB, 0xAE }; // MAC Arduino Ethernet (David)
83 byte ip[] = { 192, 168, 3, 100 };
84 EthernetServer server(PROXY_PORT);
85 InfraredProxy ir_proxy;
86 void setup()
87 {
88     // Open serial communications and wait for port to open:
89     Serial.begin(BAUD_RATE);
90     while (!Serial)
91     {
92         ; // wait for serial port to connect. Needed for Leonardo only
93     }
94     // start the Ethernet connection and the server:
95     Ethernet.begin(mac);
96     server.begin();
97     Serial.print("server is at ");
98     Serial.println(Ethernet.localIP());
99 }
100 void loop()
101 {
102     ir_proxy.receive_from_server(server);
103 }

```