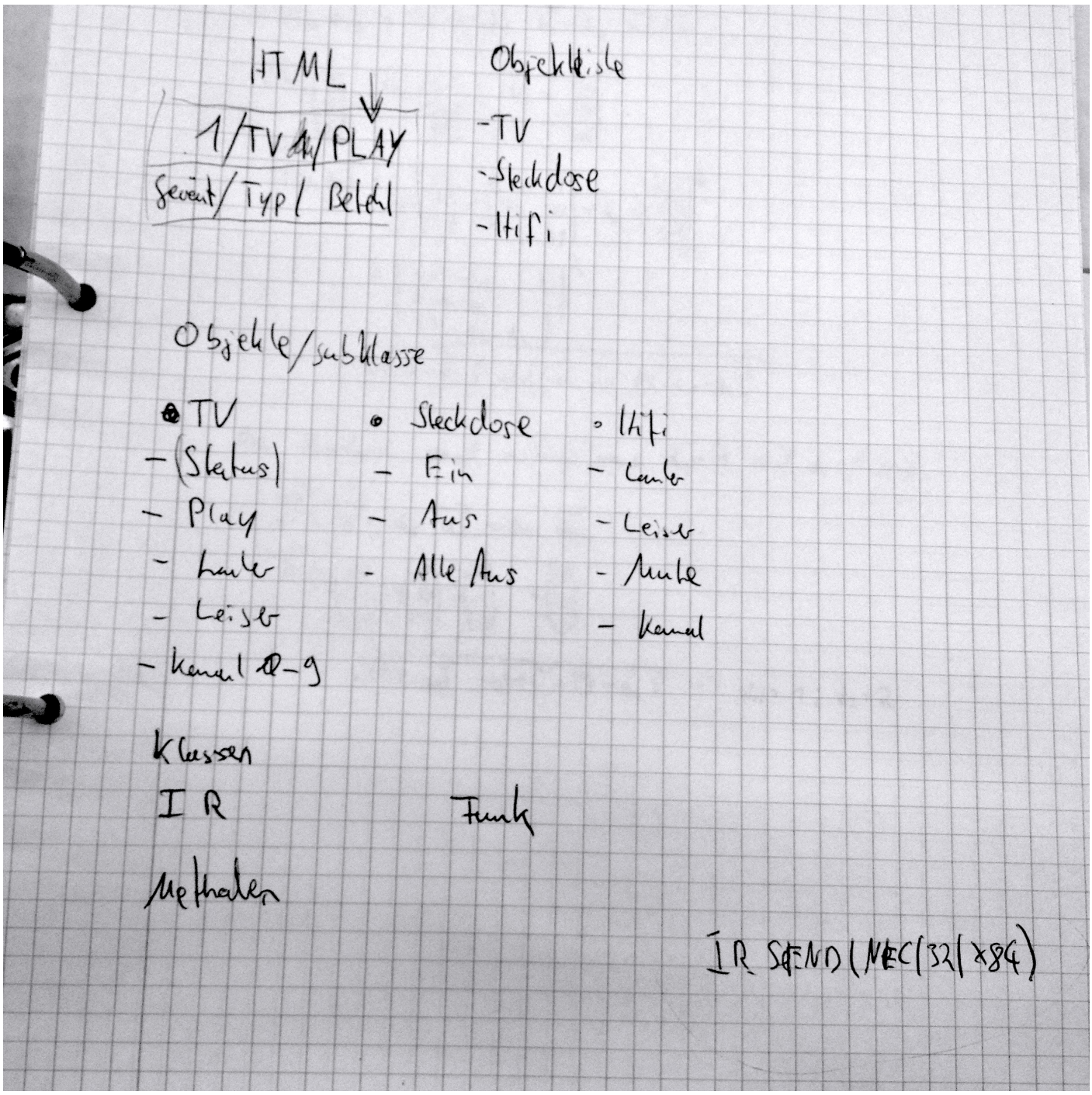

HTML
1/TV/PLAY
Sender/Typ/Befehl
Objektliste
-TV
-Steckdose
-Hifi
Objekte/subklasse
TV
-(Status)
- Play
- Lauter
- Leiser
- Kanal 0-9
Steckdose
- Ein
- Aus
- Alle Aus
Hifi
- Lauter
- Leiser
- Mute
- Kanal
Klassen
IR
Funk
Methoden
IR SEND(NEC/32/x84)

Arduino Projekt

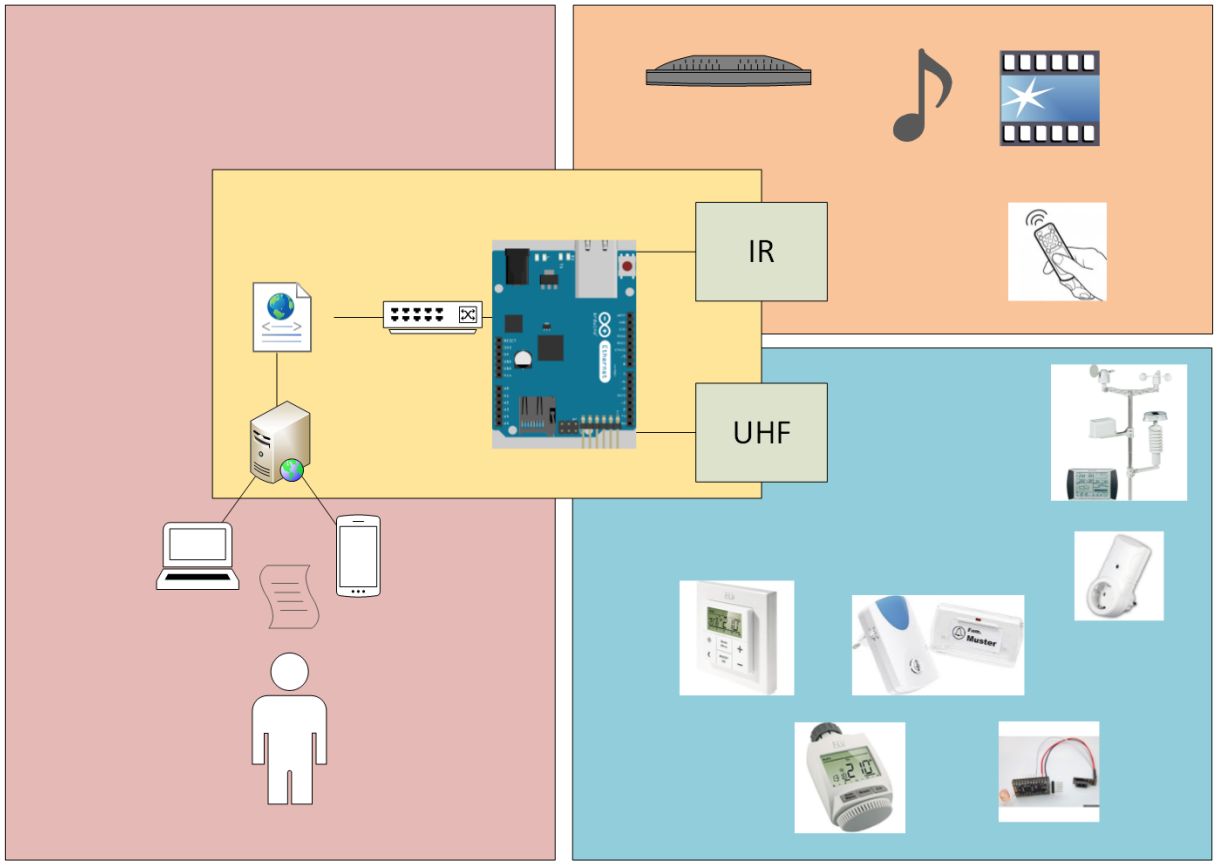
David Arenz & Matthias Lehmann

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!
Loesung: SD Karte komplett entfernen, SD Karte einstecken, 3s lang den REST Button druecken

2 Code

2.1 WEB_SD_IR.ino

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

```
// http://fluuux.de/2013/03/arduino-als-webserver-einrichten-und-webpage-von-sd-karte-laden/

#include <Ethernet.h>
#include <TextFinder.h>
#include <SD.h>
#include <IRremote.h>

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

class AppleRemote
{
    enum
    {
        CMD_LEN = 32,
        UP = 0x77E1D01D,
        DOWN = 0x77E1B01D,
        PLAY = 0x77E1201D,
        PREV = 0x77E1101D,
        NEXT = 0x77E1E01D,
        MENU = 0x77E1401D
    };

    IRsend mac;

    void send_command(const long command)
    {
        mac.sendNEC(command, CMD_LEN);
    }

public:
    void menu()
    {
        send_command(MENU);
    }
    void play()
    {
        send_command(PLAY);
    }
    void prev()
    {
        send_command(PREV);
    }
    void next()
    {
        send_command(NEXT);
    }
```

```
    void up()
51  {
        send_command(UP);
53  }
    void down()
55  {
        send_command(DOWN);
57  }
};
59
AppleRemote apple_remote;
61
const unsigned int PROXY_PORT = 80;
63 const unsigned int BAUD_RATE = 19200;
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xDB, 0xAE };
65      // MAC Arduino Ethernet (David)
byte sdPin = 4;
67      // Pin der SD-Karte

69 EthernetServer server(PROXY_PORT);
        // Server port
71
File webFile;
73
void setup()
75 {
    Serial.begin(BAUD_RATE);
77      // Open serial communications and wait for port to open:
    Ethernet.begin(mac);
79      // start the Ethernet connection and the server:
    Serial.print("Server is at: ");
81  Serial.println(Ethernet.localIP());
    server.begin();
83      // Server starten
    Serial.println("ARDUINO - STEUERUNG");
85  Serial.println("Initialisiere SD-Karte...");
    if (!SD.begin(sdPin))
87  {
        Serial.println(" - Initialisierung der SD-Karte fehlgeschlagen!");
89      return;
    }
91  Serial.println(" - SD-Karte erfolgreich initialisiert.");

93  if (!SD.exists("aprm.htm"))
    {
95      Serial.println(" - Datei (aprm.htm) wurde nicht gefunden!");
        return;
97  }
    Serial.println(" - Datei (aprm.htm) wurde gefunden.");
99
    Serial.println();
101 Serial.println("Verbraucher schalten");
}
103
void loop()
105 {
    EthernetClient client = server.available();
107     // Auf Anfrage warten

109 if(client)
    {
```

```
111         /*************************************
              Ausgaenge ueber das Webformular steuern  *
113         *************************************/
            TextFinder finder(client);
115
            if(finder.find("GET"))
117         {
                while(finder.findUntil("cmd-", "\n\r"))
119             {
                    char befehl = client.read();
                    Serial.print(" - D"+String(befehl));
                    switch(befehl)
123                 {
                    case 'm':
125                     apple_remote.menu();
                        break;
127                 case 'u':
                        apple_remote.up();
129                     break;
                    case 'd':
131                     apple_remote.down();
                        break;
133                 case 'l':
                        apple_remote.prev();
135                     break;
                    case 'r':
137                     apple_remote.next();
                        break;
139                 case 'p':
                        apple_remote.play();
141                     break;
                    default:
143                     Serial.print(" - Falscher Befehl");
                        break;
145                 }
                }
147         }

149         /************************
              Webformular anzeigen  *
151         ************************/

153         boolean current_line_is_blank = true;
                        // eine HTTP-Anfrage endet mit einer Leerzeile und einer neuen
                           Zeile
155         while (client.connected())
            {
157             if (client.available())
                    // Wenn Daten vom Server empfangen werden
159             {
                    char c = client.read();
161                         // empfangene Zeichen einlesen
                    if (c == '\n' && current_line_is_blank)
163                         // wenn neue Zeile und Leerzeile empfangen
                    {
165                     // Standard HTTP Header senden
                        client.println("HTTP/1.1 200 OK");
167                     client.println("Content-type: text/html");
                        client.println("Connection: close");
169                     client.println();
                        // Website von SD-Karte laden
```

```
171                     webFile = SD.open("aprm.htm");
                            // Website laden
173                     if (webFile)
                        {
175                         while(webFile.available())
                            {
177                             client.write(webFile.read());
                                    // Website an Client schicken
179                         }
                            webFile.close();
181                     }
                        break;
183                 }
                    if (c == '\n')
185                 {
                        current_line_is_blank = true;
187                 }
                    else if (c != '\r')
189                 {
                        current_line_is_blank = false;
191                 }
                }
193         }
            delay(1);
195         client.stop();
        }
197 }
```

3 Beispiele

3.1 InfraredDumper.ino

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

```
#include <IRremote.h>

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

IRrecv ir_receiver(IR_RECEIVER_PIN);
decode_results results;

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

void dump(const decode_results* results)
{
        const int protocol = results->decode_type;
        Serial.print("Protocol: ");
        if (protocol == UNKNOWN)
        {
        Serial.println("not recognized.");
        }
        else
        {
                if (protocol == NEC)
                {
                        Serial.println("NEC");
                }
                else if (protocol == SONY)
                {
                        Serial.println("SONY");
                }
                else if (protocol == RC5)
                {
                        Serial.println("RC5");
                }
                else if (protocol == RC6)
                {
                        Serial.println("RC6");
                }
                Serial.print("Value: ");
                Serial.print(results->value, HEX);
                Serial.print(" (");
                Serial.print(results->bits, DEC);
                Serial.println(" bits)");
        }
}

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

```
56 }
```

3.2 InfraredProxy.ino

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

```
#include <SPI.h>
#include <Ethernet.h>
#include <IRremote.h>
// ### Voraussetzungen ###
// TSOP Signal-Pin <--> Arduino - Pin 11
// IR-LED Anode <--> Arduino - Pin 3
class InfraredProxy
{
    IRsend _infrared_sender;
    void read_line(EthernetClient& client, char* buffer, const int buffer_length)
    {
        int buffer_pos = 0;
        while (client.available() && (buffer_pos < buffer_length - 1))
        {
            const char c = client.read();
            if (c=='\n')
                break;
            if (c!='\r')
                buffer[buffer_pos++] = c;
        }
        buffer[buffer_pos] = '\0';
    }
    bool send_ir_data(const char* protocol, const int bits, const long value)
    {
        bool result = true;
        if (!strcasecmp(protocol, "NEC"))
            _infrared_sender.sendNEC(value, bits);
        else if (!strcasecmp(protocol, "SONY"))
            _infrared_sender.sendSony(value, bits);
        else if (!strcasecmp(protocol, "RC5"))
            _infrared_sender.sendRC5(value, bits);
        else if (!strcasecmp(protocol, "RC6"))
            _infrared_sender.sendRC6(value, bits);
        else
            result = false;
        return result;
    }
    bool handle_command(char* line)
    {
        strsep(&line, " ");
        char* path = strsep(&line, " ");
        char* args[3];
        for (char** ap = args; (*ap = strsep(&path, "/")) != NULL;)
            if (**ap != '\0')
                if (++ap >= &args[3])
                    break;
        const int bits = atoi(args[1]);
        const long value = atol(args[2]);
        return send_ir_data(args[0], bits, value);
    }
public:
    void receive_from_server(EthernetServer server)
    {
        const int MAX_LINE = 256;
        char line[MAX_LINE];
        EthernetClient client = server.available();
        if (client)
```

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