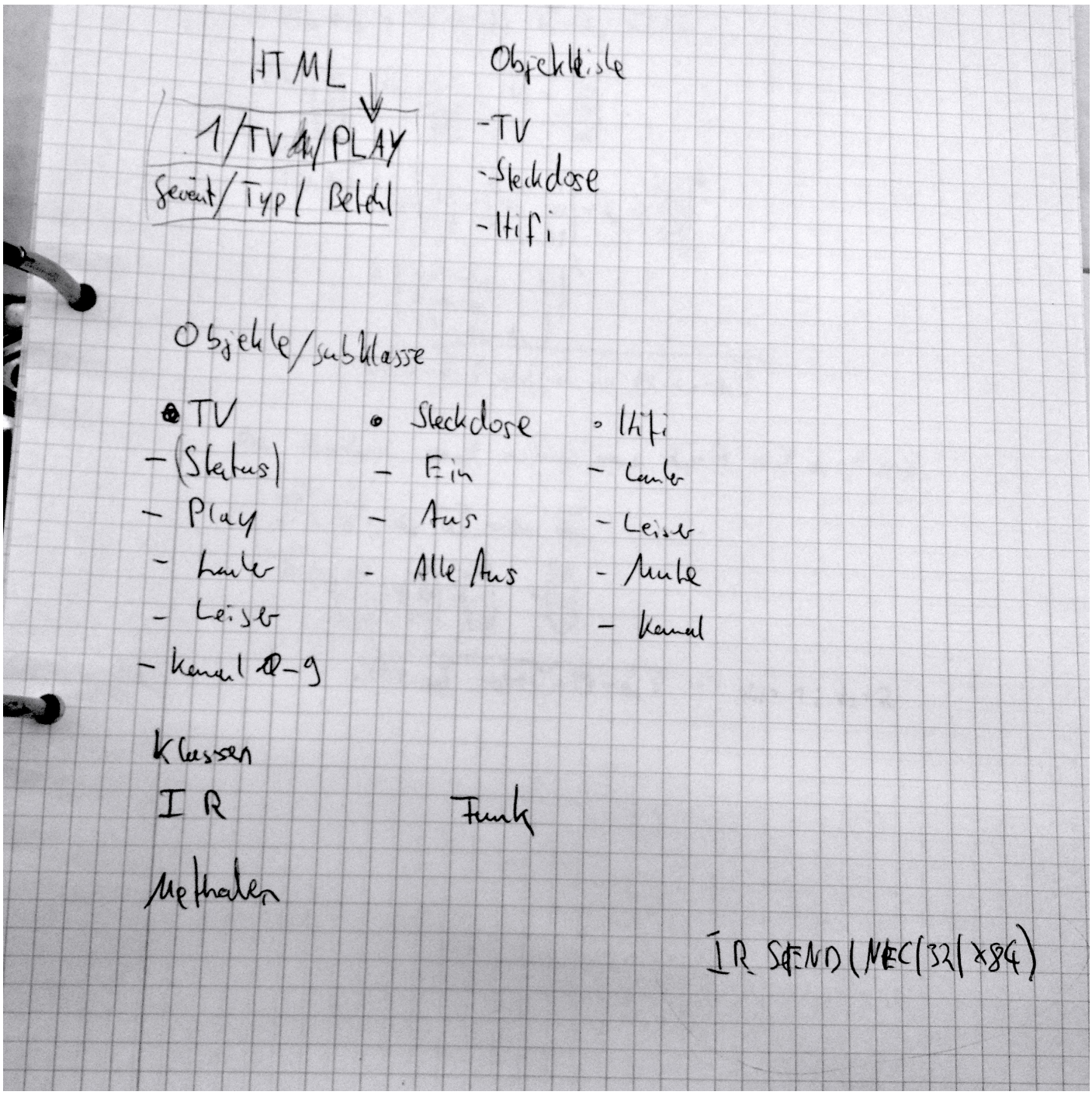

HTML
1/TV/PLAY
Sender/Typ/Befehl
Objektliste
-TV
-Steckdose
-Hifi
Objekte/subklasse
TV
-(Status)
- Play
- Lauter
- Leiser
- Kanal 0-9
Steckdose
- Ein
- Aus
- Alle Aus
Hifi
- Lauter
- Leiser
- Mute
- Kanal
Klassen
IR
Funk
Methoden
IR SEND(NEC/32/x84)

Arduino Projekt

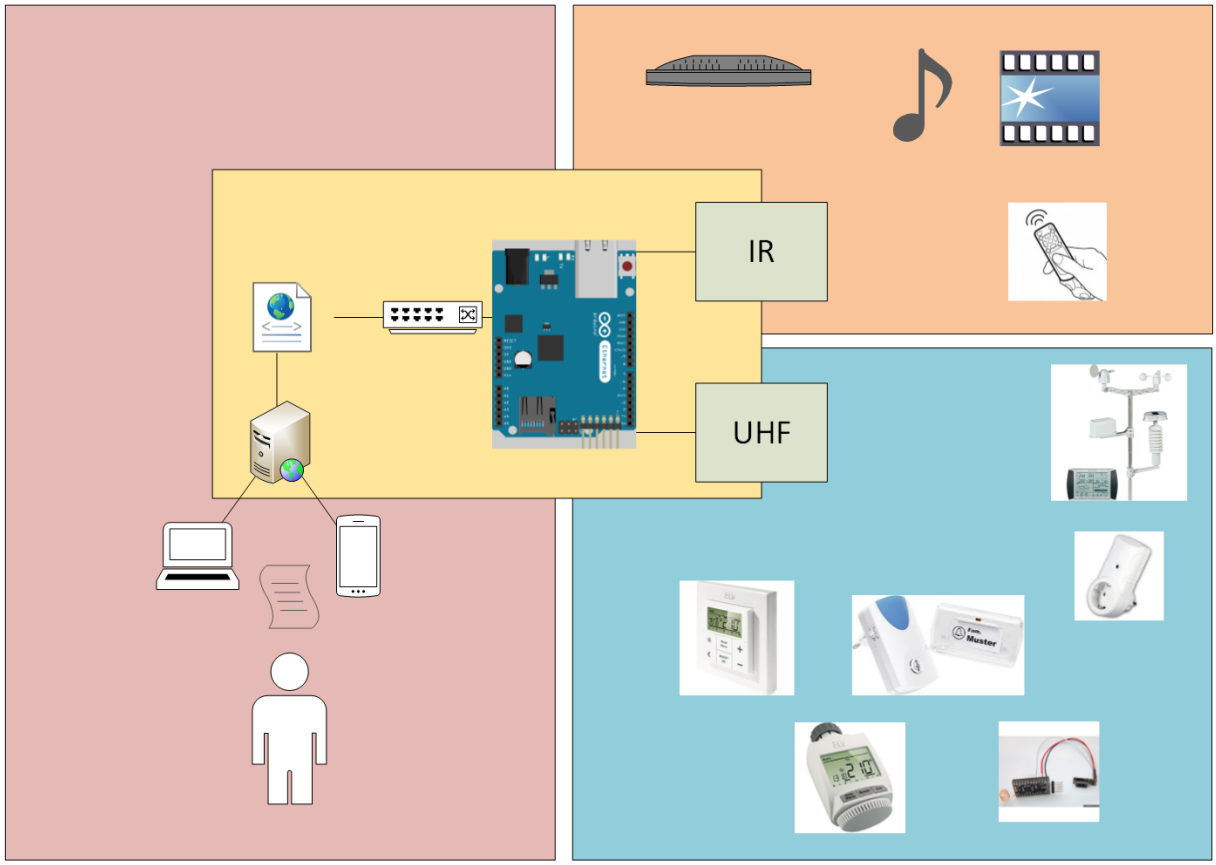
David Arenz & Matthias Lehmann

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 <SPI.h>
#include <Ethernet.h>
#include <IRremote.h>
#include <TextFinder.h>
#include <SD.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);
  }



    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:
```

```
    bool send_ir_data(const char* protocol, const int bits, const long value)
    {
        bool result = true;
        if (!strcasecmp(protocol, "NEC"))
            mac.sendNEC(value, bits);
        else if (!strcasecmp(protocol, "SONY"))
            mac.sendSony(value, bits);
        else if (!strcasecmp(protocol, "RC5"))
            mac.sendRC5(value, bits);
        else if (!strcasecmp(protocol, "RC6"))
            mac.sendRC6(value, bits);
        else
            result = false;
        return result;
    }
    };

AppleRemote apple_remote;

const unsigned int PROXY_PORT = 80;
const unsigned int BAUD_RATE = 19200;
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xDB, 0xAE };
        // MAC Arduino Ethernet (David)
byte sdPin = 4;
        // Pin der SD-Karte
const int MAX_LINE = 256;
char line[MAX_LINE];

EthernetServer server(PROXY_PORT);
        // Server port

File webFile;

void setup()
{
    Serial.begin(BAUD_RATE);
        // Open serial communications and wait for port to open:
    Ethernet.begin(mac);
        // start the Ethernet connection and the server:
    Serial.print("Server is at: ");
    Serial.println(Ethernet.localIP());
    server.begin();
        // Server starten
    Serial.println("ARDUINO - STEUERUNG");
    Serial.println("Initialisiere SD-Karte...");
    if (!SD.begin(sdPin))
    {
        Serial.println(" - Initialisierung der SD-Karte fehlgeschlagen!");
        return;
    }
    Serial.println(" - SD-Karte erfolgreich initialisiert.");

    if (!SD.exists("aprm2.htm"))
    {
        Serial.println(" - Datei (aprm2.htm) wurde nicht gefunden!");
        return;
    }
    Serial.println(" - Datei (aprm2.htm) wurde gefunden.");

    Serial.println();
    Serial.println("Verbraucher schalten");
```

```
111 }
113 void loop()
    {
115     EthernetClient client = server.available();
            // Auf Anfrage warten
117
        if(client)
119     {
            /***************************************
121           Ausgaenge ueber das Webformular steuern  *
            ***************************************/
123         TextFinder finder(client);
125         if(finder.find("GET"))
            {
127             while(finder.findUntil("cmd-", "\n\r"))
                {
129               //String prot = client.read();
                 // int  bits = finder.getValue();
131             //  int  value = finder.getValue();
               //   send_ir_data(prot, bits, value);
133               Serial.println("CMD-");
                }
135         }
137         /************************
              Webformular anzeigen  *
139         ************************/
141         boolean current_line_is_blank = true;
                        // eine HTTP-Anfrage endet mit einer Leerzeile und einer neuen
                           Zeile
143         while (client.connected())
            {
145             if (client.available())
                    // Wenn Daten vom Server empfangen werden
147             {
                    read_line(client, line, MAX_LINE);
149                 char c = client.read();
                            // empfangene Zeichen einlesen
151                 if (c == '\n' && current_line_is_blank)
                            // wenn neue Zeile und Leerzeile empfangen
153                 {
                        // Standard HTTP Header senden
155                     client.println("HTTP/1.1 200 OK");
                        client.println("Content-type: text/html");
157                     client.println("Connection: close");
                        client.println();
159                     // Website von SD-Karte laden
                        webFile = SD.open("aprm2.htm");
161                         // Website laden
                        if (webFile)
163                     {
                            while(webFile.available())
165                         {
                                client.write(webFile.read());
167                                 // Website an Client schicken
                            }
169                         webFile.close();
                        }
```

```
171                     break;
                    }
173                 if (c == '\n')
                    {
175                     current_line_is_blank = true;
                    }
177                 else if (c != '\r')
                    {
179                     current_line_is_blank = false;
                    }
181             }
            }
183         delay(1);
            client.stop();
185     }
    }
187
    IRsend _infrared_sender;
189     void read_line(EthernetClient& client, char* buffer, const int buffer_length)
        {
191         int buffer_pos = 0;
            while (client.available() && (buffer_pos < buffer_length - 1))
193         {
                const char c = client.read();
195             if (c=='\n')
                    break;
197             if (c!='\r')
                    buffer[buffer_pos++] = c;
199         }
            buffer[buffer_pos] = '\0';
201     }
```

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);
    }
```