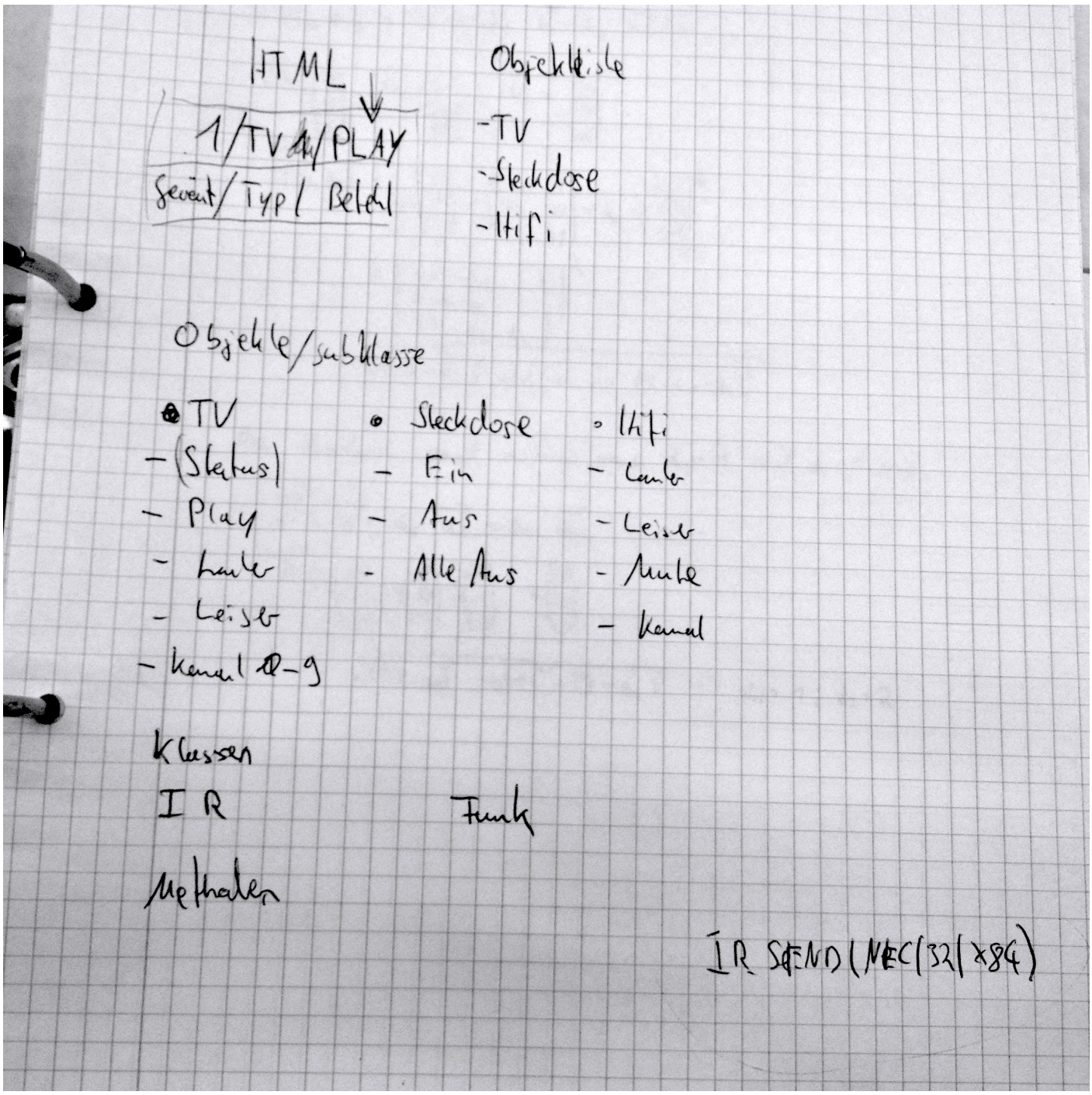

HTML
1/TV/PLAY
Sender/Typ/Befehl
Objektliste
-TV
-Steckdose
-Hifi
Objekte/subklasse
TV
-(Status)
- Play
- Lauter
- Leiser
- Kanal 0-9
Steckdose
- Ein
- Aus
- Alle Aus
Hifi
- Lauter
- Leiser
- Mute
- Kanal
Klassen
IR
Funk
Methoden
IR SEND(NEC/32/x84)

# Arduino Projekt

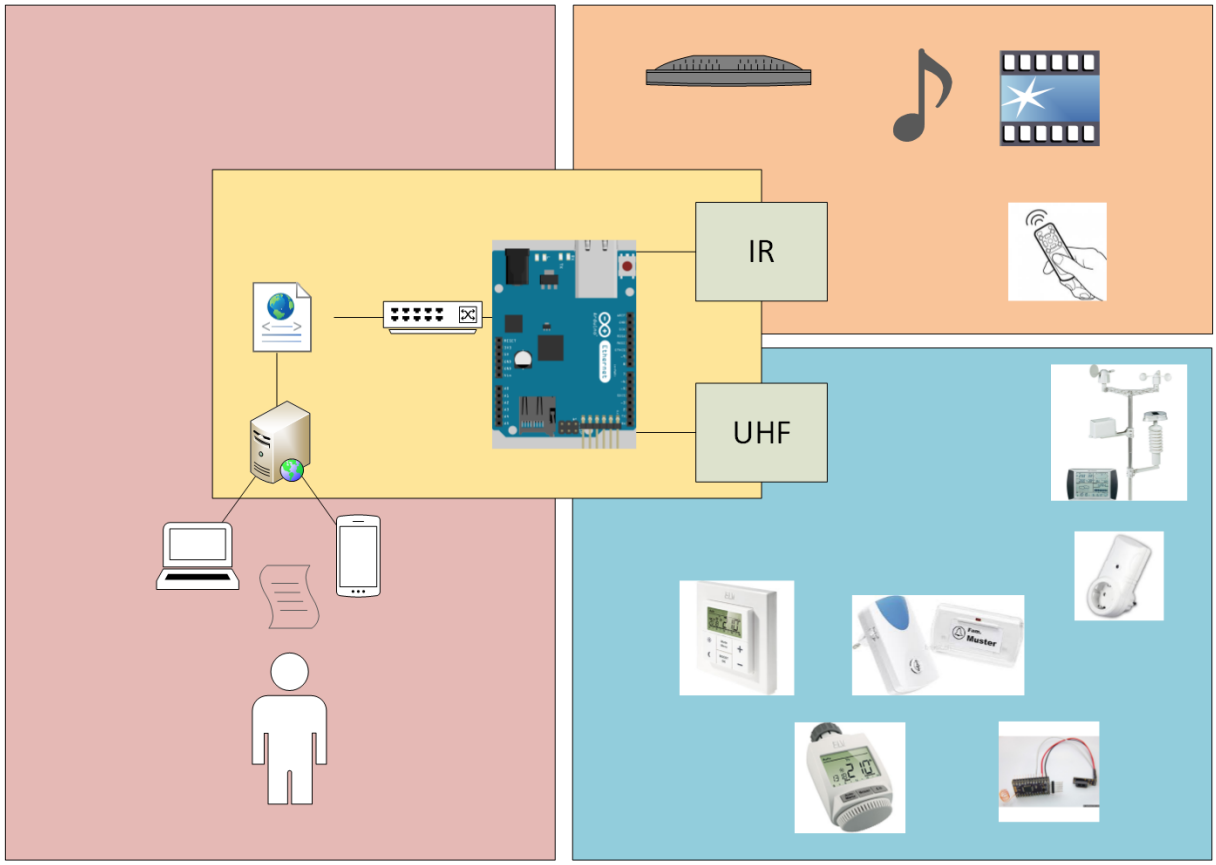
David Arenz & Matthias Lehmann

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

# 2 Code

Listing 1: `../code/IR_Proxy/IR_Proxy.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)
53  {
        const int MAX_LINE = 256;
55      char line[MAX_LINE];
        EthernetClient client = server.available();
57      if (client)
        {
59          while (client.connected())
            {
61              if (client.available())
                {
63                  read_line(client, line, MAX_LINE);
                    Serial.println(line);
65                  if (line[0] == 'G' && line[1] == 'E' && line[2] == 'T')
                        handle_command(line);
67                  if (!strcmp(line, ""))
                    {
69                      client.println("HTTP/1.1 200 OK\n");
                        break;
71                  }
                }
73          }
            delay(1);
75          client.stop();
        }
77  }
};
79  //--- ENDE DER DEKLARATION ---
const unsigned int PROXY_PORT = 80;
81  const unsigned int BAUD_RATE = 19200;
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xDB, 0xAE }; // MAC Arduino Ethernet (David)
83  byte ip[] = { 192, 168, 3, 100 };
EthernetServer server(PROXY_PORT);
85  InfraredProxy ir_proxy;
void setup()
87  {
// Open serial communications and wait for port to open:
89      Serial.begin(BAUD_RATE);
        while (!Serial)
91      {
            ; // wait for serial port to connect. Needed for Leonardo only
93      }
        // start the Ethernet connection and the server:
95      Ethernet.begin(mac);
        server.begin();
97      Serial.print("server is at ");
        Serial.println(Ethernet.localIP());
99  }
void loop()
101 {
        ir_proxy.receive_from_server(server);
103 }
```

# 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();
    }
```

```
}
```

### 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);
    }
```