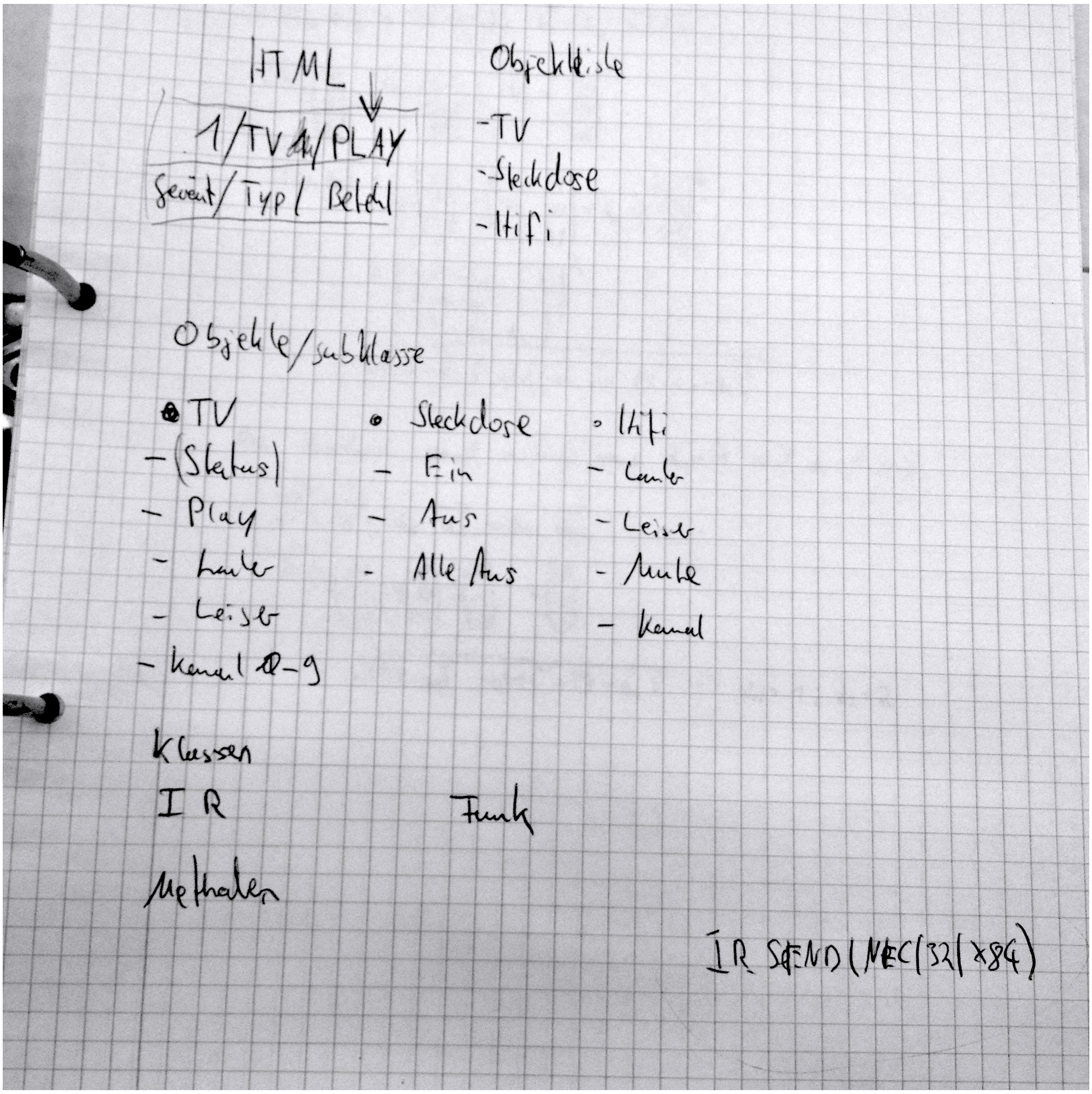


HTML

1/TV ~~4~~/PLAY

Gerät/ Typ/ Befehl

Objektliste

- TV
- Steckdose
- Hifi

Objekte/subklasse

● TV	● Steckdose	● Hifi
– (Status)	– Ein	– Laut
– Play	– Aus	– Leiser
– Lauter	– Alle Aus	– Mute
– Leiser		– Kanal
– Kanal ~~0~~–9		

Klassen

IR

Funk

Methoden

IR SEND(NEC(32(x84)

Arduino Projekt

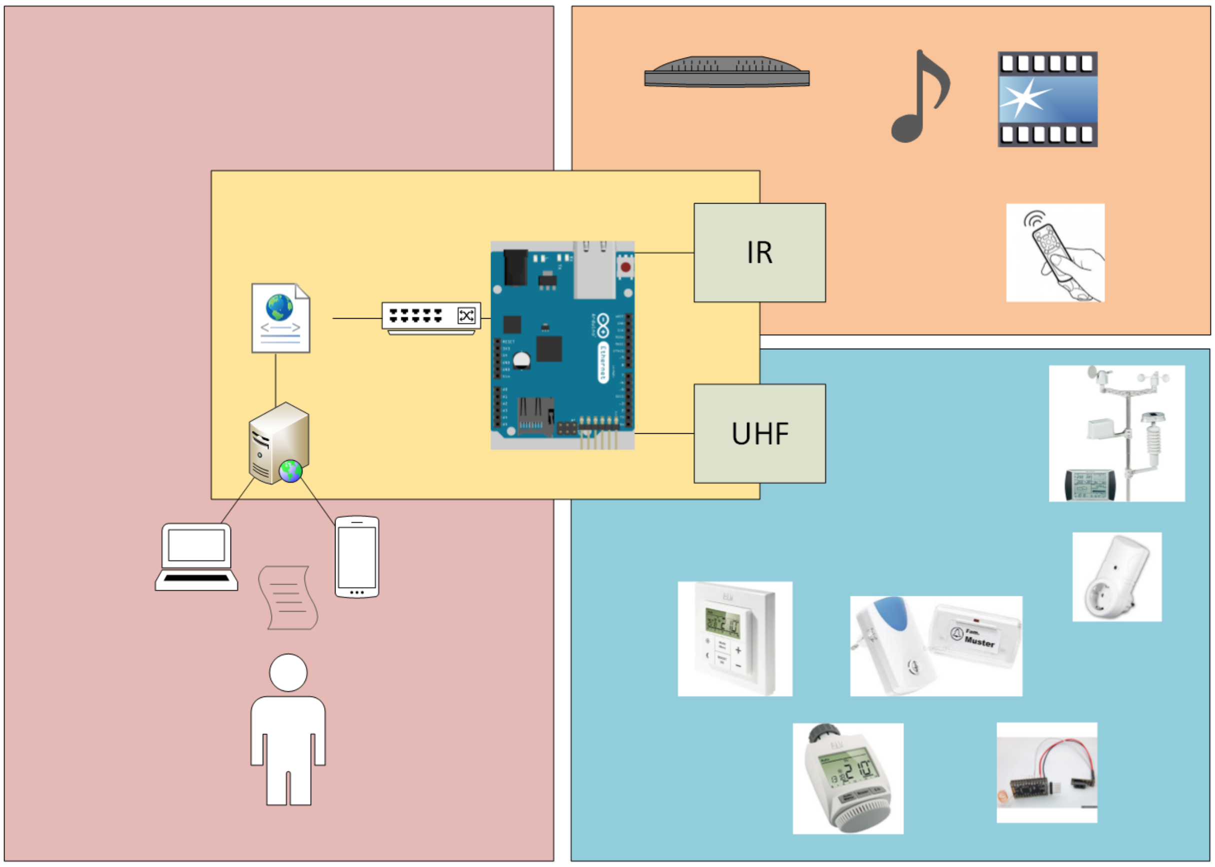
David Arenz & Matthias Lehmann

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 Troubleshoting

- Fehler: Initialisierung der SD-Karte fehlgeschlagen!

  Ursache: Die SD Karte ist nach dem Laden eines neuen Sketches noch initialisiert. Ein normaler Reset reicht scheinbar nicht aus.

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

- Symptom: Keine Ausgabe auf der Konsole.

  Fehler: Arduino bekommt keine IP (vermutlich)

  Loesung:

  - Konsole schliessen
  - Arduino von USB und NW trennen
  - NW anschliessen
  - USB anschliessen
  - Arduino reseten
  - Konsole oeffnen

Important Note!
If an uninitialized SD card is left in the SD card socket of the shield, it can cause problems with code in the sketch that is accessing the Ethernet chip. This may cause symptoms such as the sketch running once or twice, then hanging up.
This is because both the Ethernet chip and the SD card are accessed by the Arduino using the same SPI bus.
If the SD card is not being used with an Ethernet application, either remove it from the socket or add the following code to disable the SD card:

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()
    {
        send_command(UP);
    }
    void down()
    {
        send_command(DOWN);
    }
};

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

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("aprm.htm"))
    {
        Serial.println(" - Datei (aprm.htm) wurde nicht gefunden!");
        return;
    }
    Serial.println(" - Datei (aprm.htm) wurde gefunden.");

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

void loop()
{
    EthernetClient client = server.available();
        // Auf Anfrage warten

    if(client)
    {
        /****************************************
          Ausgaenge ueber das Webformular steuern  *
        ****************************************/
        TextFinder finder(client);

        if(finder.find("GET"))
        {
            while(finder.findUntil("cmd-", "\n\r"))
            {
                char befehl = client.read();
                Serial.print(" - D"+String(befehl));
                switch(befehl)
                {
                case 'm':
                    apple_remote.menu();
                    break;
                case 'u':
                    apple_remote.up();
                    break;
                case 'd':
                    apple_remote.down();
                    break;
                case 'l':
                    apple_remote.prev();
                    break;
                case 'r':
                    apple_remote.next();
                    break;
                case 'p':
                    apple_remote.play();
                    break;
                default:
                    Serial.print(" - Falscher Befehl");
                    break;
                }
            }
        }

        /***********************
```

```
          Webformular anzeigen  *
        ***********************/

        boolean current_line_is_blank = true;
                        // eine HTTP-Anfrage endet mit einer Leerzeile und einer neuen
                           Zeile
        while (client.connected())
        {
            if (client.available())
                // Wenn Daten vom Server empfangen werden
            {
                char c = client.read();
                        // empfangene Zeichen einlesen
                if (c == '\n' && current_line_is_blank)
                        // wenn neue Zeile und Leerzeile empfangen
                {
                    // Standard HTTP Header senden
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-type: text/html");
                    client.println("Connection: close");
                    client.println();
                    // Website von SD-Karte laden
                    webFile = SD.open("aprm.htm");
                        // Website laden
                    if (webFile)
                    {
                        while(webFile.available())
                        {
                            client.write(webFile.read());
                                // Website an Client schicken
                        }
                        webFile.close();
                    }
                    break;
                }
                if (c == '\n')
                {
                    current_line_is_blank = true;
                }
                else if (c != '\r')
                {
                    current_line_is_blank = false;
                }
            }
        }
        delay(1);
        client.stop();
    }
}
```

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)
        {
            while (client.connected())
            {
                if (client.available())
                {
                    read_line(client, line, MAX_LINE);
                    Serial.println(line);
                    if (line[0] == 'G' && line[1] == 'E' && line[2] == 'T')
                        handle_command(line);
                    if (!strcmp(line, ""))
                    {
                        client.println("HTTP/1.1 200 OK\n");
                        break;
                    }
                }
            }
            delay(1);
            client.stop();
        }
    }
};
//---- ENDE DER DEKLARATION ----
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 ip[] = { 192, 168, 3, 100 };
EthernetServer server(PROXY_PORT);
InfraredProxy ir_proxy;
void setup()
{
// Open serial communications and wait for port to open:
    Serial.begin(BAUD_RATE);
    while (!Serial)
    {
        ; // wait for serial port to connect. Needed for Leonardo only
    }
    // start the Ethernet connection and the server:
    Ethernet.begin(mac);
    server.begin();
    Serial.print("server is at ");
    Serial.println(Ethernet.localIP());
}
void loop()
{
    ir_proxy.receive_from_server(server);
}
```