Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Fakultät Design, Medien, Information

Kunst- und Mediencampus Hamburg

Department Medientechnik

Studiengang Media Systems

Projekt:

Arduino-Smart-Home

„Entwicklung und Konstruktion einer Steuerung für Funksteckdosen via Web-App“

Projektteilnehmer: **Marko Vukadinovic, Sebastian Bohn, Johannes Bagge, Florian Langhorst**

Projektleiter: **Prof. Dr. Andreas Plaß, Dipl. Ing. Michael Berens**

Inhaltsverzeichnis

1 Projektbeschreibung

Das Ziel dieses Projekts ist es mit Hilfe einer Arduino Uno Entwicklungsplattform und einer selbstentwickelten Webseite, eine oder mehrere Funksteckdosen ein- und ausschalten zu können. Erweitert wird dies um eine Steuerung für den Betrieb einer Pflanzenbewässerungsanlage.

2 Projektplanung

2.1 Definition der Projektumsetzung

Eine möglichst genaue Definition der zu realisierenden Ziele ist wichtig, um das Projekt effektiv umzusetzen. Folgende Meilensteine wurden in der Planung definiert:

- Netzwerkkommunikation durch den Arduino
- Steuerung der Stromquellen durch den Arduino
- Webinterface für kabellose Kommunikation
- Ansteuern einer Pflanzenbewässerung

Vorab ist es ratsam sich im Internet über ähnliche Projekte zu informieren, um eine Liste der benötigten Komponenten zu erstellen. Von Vorteil sind Grundkenntnisse der Elektrotechnik, Elektronik, Netzwerktechnik und objektorientierten Programmierung. Es sollte auch genügend Zeit für Testphasen eingeplant werden, um sich den sicheren Umgang mit den einzelnen Elementen anzueignen. Eine gute Quelle für den Einstieg ist die Webseite http://www.arduino.cc, auf der viele nützliche Tutorials und ein hilfreiches Forum zu finden sind.

3 Hardware

3.1 Arduino Uno R3

Arduino Uno R3 ist ein Mikrocontroller Entwicklungsboard das auf dem ATmega328 basiert. Dieser wird für sogenanntes „Prototyping“ genutzt, um zum Beispiel den Input von verschiedenen Sensoren und Schaltern zu verarbeiten und über den Output Stromkreise, Lichtquellen, Relais und Motoren zu kontrollieren.

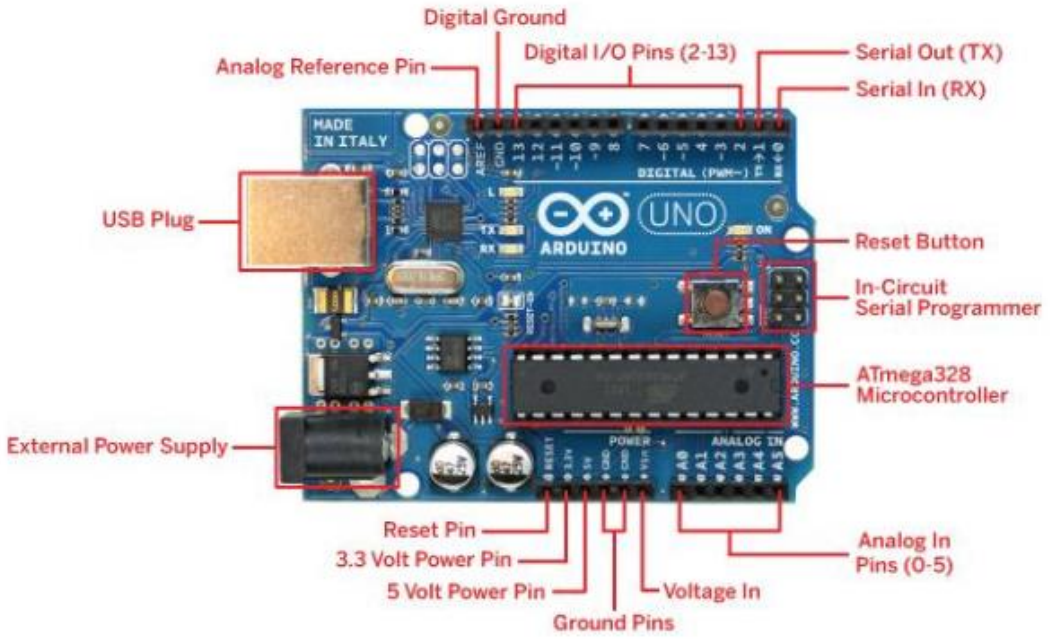

Abbildung 1: Arduino Uno R3 und Erläuterung der Elemente

Datenblatt Arduino Uno R3	
Mikrocontroller	ATmega328
Betriebsspannung	5V
Eingangsspannung (empfohlen)	7-12V
Eingangsspannung (maximum)	6-20V
Digitale I/O Pins	14 (6 bieten PWM/Pulse-width modulation)
Analoge Input Pins	6
DC pro I/O Pin	40 mA
DC für 3.3V Pin	50 mA
Flash Speicher	32 KB (ATmega328), davon 0.5 KB für den Bootloader belegt
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)

Clock Frequenz	16 MHz
Kosten	ca. 22,00 €

3.2 Ethernet-Shield

Ein Shield ist eine Erweiterung des Boards, welches zusätzliche Schnittstellen für den Mikrocontroller bietet. Mit dem Ethernet-Shield ist es möglich durch ein RJ45 LAN-Kabel eine Netzwerkverbindung aufzubauen. Zusätzlich bietet dieser noch einen Steckplatz für eine Mikro-SD Karte zur Auslagerung größerer Daten. Der Shield wird direkt auf den Arduino aufgesteckt, so dass die Pins über das Shield zu erreichen sind. Jedoch werden für die einige Funktionen mehrere Pins belegt. Zu Beginn wurde ein Nachbau des Originals verwendet, welches jedoch Probleme mit der Stabilität der Stromversorgung bereitete. Daraufhin wurde ein Original Arduino Ethernet-Shield nachgekauft und verwendet.

Datenblatt Arduino Ethernet-Shield	
Betriebsspannung	5V (über den Arduino)
Ethernet Controller	W5100 mit internem 16K Buffer
Verbindungsgeschwindigkeit	10/100 Mb
Pin 4	Mikro-SD Kartenslot
Pin 10-13	W5100 und Netzwerkkommunikation
Bibliotheken	Ethernet.h; SD.h; SPI.h;...usw.
LEDs	
PWR	Stromversorgung aktiv
LINK	leuchtet bei vorhandener Netzwerkverbindung; blinkt wenn das Shield Daten sendet oder empfängt
100M	bei einer 100 Mb/s Verbindung
RX	Daten werden empfangen
TX	Daten werden gesendet
FULLD	Netzwerkverbindung ist Vollduplex
COLL	blinkt bei Netzwerkkollisionen
Kosten Nachbau	ca. 8,00 €
Kosten Original	ca. 30,00 €

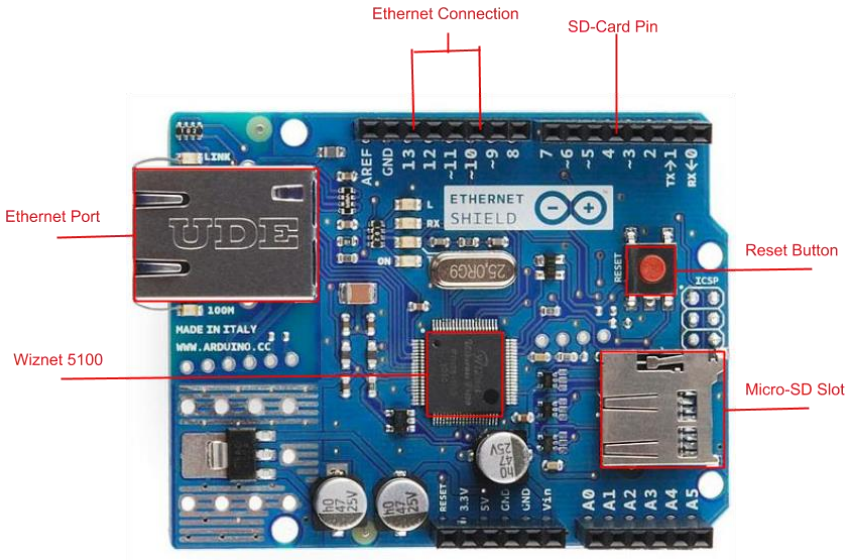

Abbildung 2 : Arduino Ethernet-Shield mit Erläuterung der Elemente

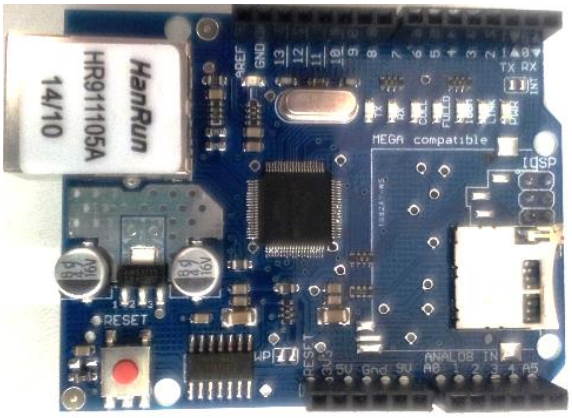

Abbildung 3: Ethernet-Shield Nachbau

3.3 Relais

Ein Relais ist ein durch elektrischen Strom betriebener Schalter, mit dem man andere Stromkreise aktivieren oder deaktivieren kann. Für dieses Projekt wurden zwei 4-Kanal Relais verwendet, welche einen Default-Zustand auf HIGH haben und auf LOW schalten.

Datenblatt Relais	
Betriebsspannung über Arduino	5V
Strom pro Kanal	15 - 20mA

VCC	Integrated Circuit (IC) Stromversorgung (positiv)
GND	Ground
IN1-IN4	Relais 1-4; je eine Verbindung zu einem I/O Pin
NC	Normally Closed; Ruhekontakt; Break
NO	Normally Open; Arbeitskontakt; Make
CO	Change Over; Wechselkontakt; Break-Make
Kosten pro Modul	ca 3,60 €

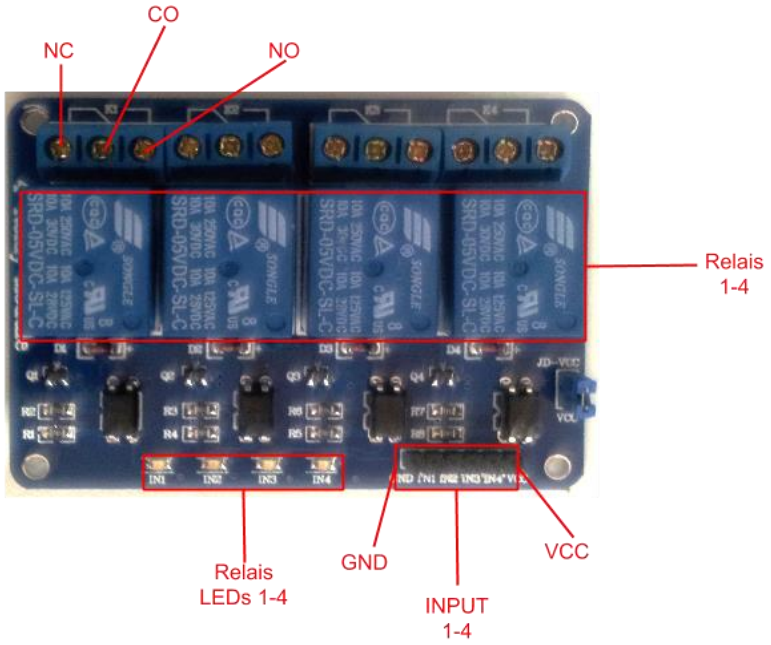

Abbildung 4: 5V 4-Kanal Relais mit Erläuterung der Elemente

3.4 Funksteckdosen mit Sender

Verwendet wurden handelsübliche 433 MHz Funksteckdosen mit vier Kanälen an der Fernbedienung und drei Steckdosen (ohne Dip Schalter). Der Aufbau der Platine der Fernbedienung unterscheidet sich je nach Hersteller.

Datenblatt Funksteckdosen-Set	
Betriebsspannung Fernbedienung	12V Batterie
Spannung Steckdosen	230V / 50 Hz
maximale Schaltleistung	1000 W
Reichweite	ca. 30 m
Frequenz	433.92 MHz
Verwendetes Modell	Mumbi m-FS300

Kosten	ca. 17,00 €

Abbildung 5: Funksteckdosen-Set von Mumbi

3.5 Verbrauchsmaterial, Kabel und Zusatzteile

- Schrumpfschlauch
- Lötplatinen
- Schrauben und Muttern
- Abstandshalter
- Jumper Kabel
- Kabel-Steckbrücken
- 8-poliges RJ-45 Netzwerkkabel
- 9V Netzteil
- Breadboard, LEDs, Widerstände (für Testaufbauten)

3.6 Werkzeug

- Schraubendreher
- Crimpzange
- Multimeter

- Heißklebepistole und Kleber
- Bohrmaschine

4 Software

Arduino bietet eine eigene, auf Java basierende, Open-Source Entwicklungsumgebung, welche vorab schon viele Bibliotheken und Beispielprogramme enthält. Als Programmiersprache wird Processing verwendet, welche an eine vereinfachte Variante von C++ erinnert und objektorientiert ist.
Mit der Entwicklungsumgebung können Programme geschrieben, getestet, auf den Mikrocontroller aufgespielt und während der Laufzeit debugged werden. Diese läuft auf allen üblichen Systemen wie Windows, Mac OS X und Linux.
http://arduino.cc/en/Main/Software

5 Umsetzung

5.1 Analyse und Tests der Bauteile

Im ersten Schritt der Projektrealisierung, ist es von Vorteil sich mit den einzelnen Komponenten und ihren Funktionen vertraut zu machen. Zum Einstieg kann man sich mit einfachen Versuchsaufbauten einarbeiten. Hier können die verschiedenen Basic-Beispielprogramme der Entwicklungsumgebung helfen. Ein Beispielaufbau wäre, eine einfache LED über den Arduino und ein Relais zum leuchten zu bringen. Dafür ist es nötig sich mit grundlegenden Regeln der Elektrotechnik auseinanderzusetzen, wie Stromstärke, Spannung, Widerständen und Flussrichtung. Das Programm oder Sketch wie es in der Entwicklungsumgebung heißt, bestimmt die Pins (Aus-/Eingänge) und was dort passieren soll. Dieser kann aus einem oder mehreren Tabs bestehen, wodurch objektorientierter Code möglich ist. Dieser darf beim Uno jedoch maximal 32 KB groß sein. Wenn das Programm

erfolgreich auf den Arduino geladen wurde, läuft dieses in einer Endlosschleife, solange die Stromzufuhr aktiv ist. Die Kabelverbindungen können alle mit simplen Jumper Wires (Kabelbrücken) verdrahtet werden. Bei den Tests der Relais, kann es sein das die Stromzufuhr über das USB-Kabel des Computers nicht ausreicht und ein instabiles oder gar kein Signal bei den Relais ankommt. Hierfür ist ein zusätzliches Netzteil von Vorteil.
Der nächste Schritt ist die Analyse der Funkfernbedienung des Steckdosen-Sets. Diese können je nach Hersteller variieren. In diesem Fall besteht die Platine aus acht Kontaktpunkten (vier ON; vier OFF), drei Kontaktbrücken, in denen bestimmte Kontakte zusammenlaufen, und einem Logik-Gatter. Die acht Kontaktpunkte sind jeweils gespalten und das Auslösen des jeweiligen Funksignals erfolgt durch Überbrückung des Kontakts. Nun testet man, welche Kombinationen aus Punkten und Brücken welches Signal auslösen und schreibt das Ergebnis in eine Tabelle. Aus dieser Tabelle ermittelt man die kleinstmögliche Anzahl der benötigten Lötpunkte.

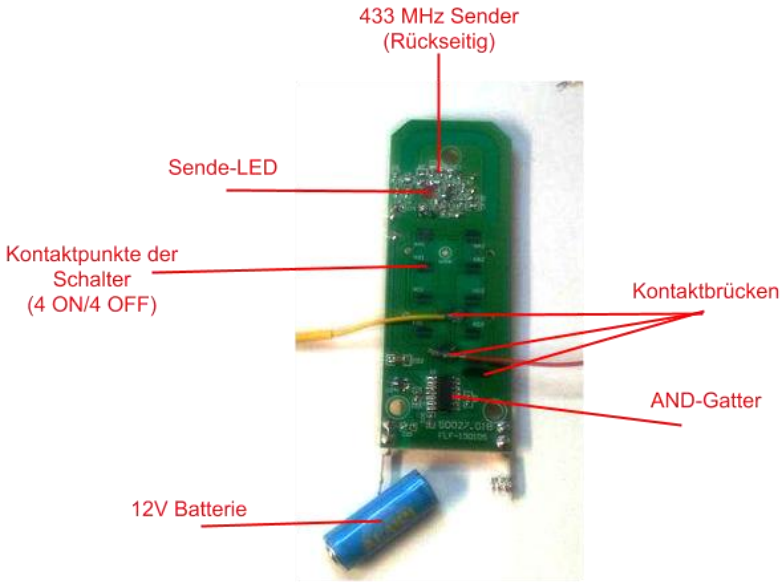

Abbildung 6: Platine der Funkfernbedienung und Erläuterung der Elemente

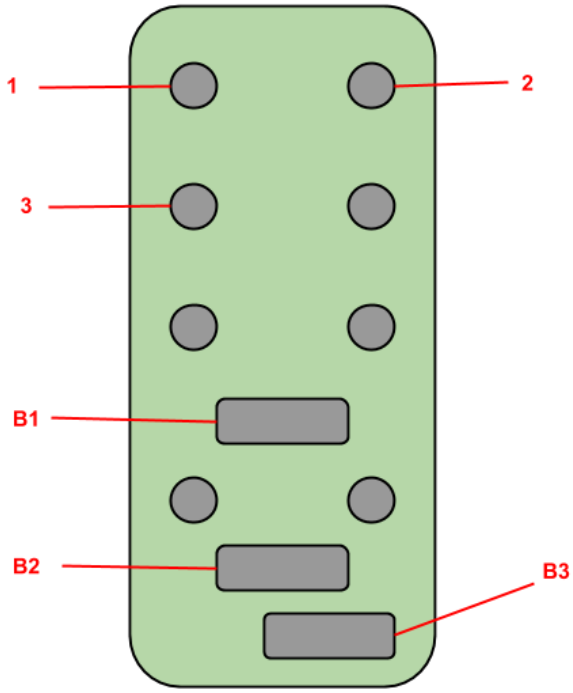

Abbildung 7: Deklaration der gewählten Kontakte

Schalter	AND-Verknüpfung von Kontakt A und Kontakt B	
A ON	2	1
A OFF	2	B3
B ON	2	3
B OFF	2	B2
C ON	B1	1
C OFF	B1	B3
D ON	B1	3
DOFF	B1	B2

5.2 Verkabelung der Sende-Platine

Für das verkabeln der gewählten Punkte ist löten leider nicht immer möglich, da Lötzinn auf dieser Art Platinen nicht haftet. Eine Alternative ist Heißkleber, welcher nach anpressen genügend halt gibt, aber sich auch rückstandsfrei lösen lässt.

Abbildung 8: Verbindung der Kabel mit Heißkleber auf den Kontakten

Mit Hilfe der erstellten Schalttabelle, können die minimal benötigten Kontaktpunkte ermittelt werden und entsprechend viele Abzweigungen des Kabels vorgenommen werden.

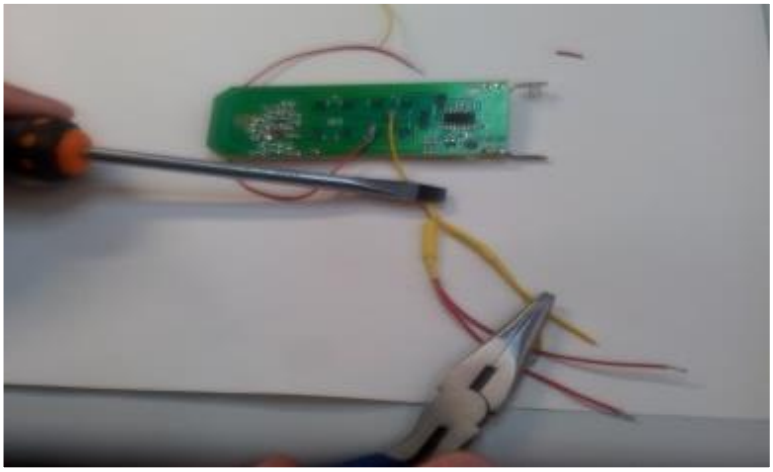

Abbildung 9: Verzweigungen der Kabel für einzelne Kontakte

Farbige Kabel können zur Übersicht beitragen. Um die Abzweigungen zu isolieren und festigen, sollte Schrumpfschlauch verwendet werden.
Um die Platine und die Klebepunkte vor äußeren Einwirkungen zu schützen, kann die Plastikhülle weitervervendet werden. Dabei reicht ein seitlich eingefräster Schlitz als Durchlass und Zugentlasstung.

Abbildung 10: Funkfernbedienung mit fertiger Verkabelung

5.3 Verbindung von Arduino, Relais und Sender

Als nächstes wird das Auslösen eines Funksignals über ein Relais getestet. Ein kurzer Impuls reicht dem Schalter in der Regel aus. Dieser liegt in etwa bei 500 ms, da bei zu kurzen Impulsen der Empfänger nicht immer ein Signal empfängt. Ein Pin ist mit einem Relais, für einen Schalter zuständig.

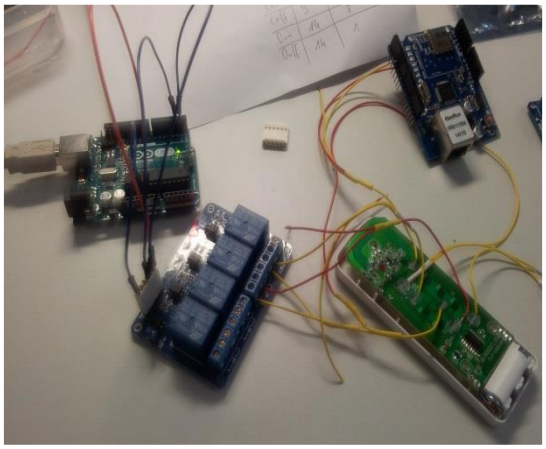

Abbildung 11: Testen der Steuerung des Senders über den Arduino und Relais

Um alle Schalter anzusteuern, benötigt man also acht freie I/O Pins und acht Relais. Da in einem Set aber nur drei Steckdosen enthalten sind, reichen sechs Pins und Relais aus. Entsprechend kann man einen ON und zugehörigen OFF Schalter frei lassen und diesen isolieren. Es sollte ebenfalls bedacht werden, das Pin 10 – 13 für den Ethernet-Shield und Pin 4 für die Mikro-SD Karte (wenn benötigt) reserviert sind. Um beide Relais-Module zu nutzen, muss vom Arduino ausgehend eine Parallelschaltung zu den beiden Relais gelegt werden. Dies erreicht man mit der Splittung des 5V und GND Aus-/Eingangs. Unterstützend ist hier ein Netzteil anzuraten, um die Stromzufuhr der Relais zu stabilisieren.

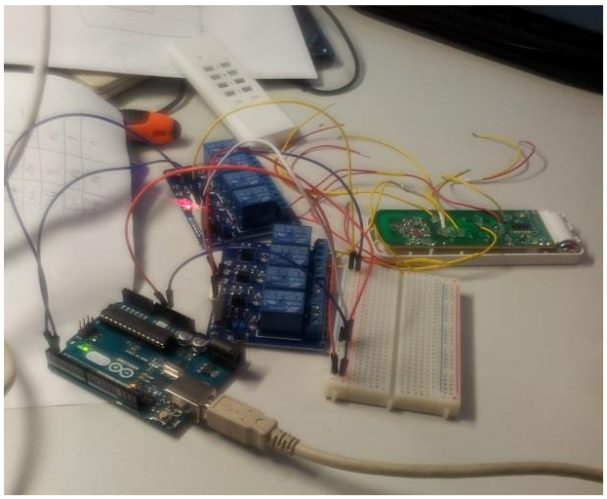

Abbildung 12: Test mit beiden Relais-Modulen

Wenn alle Relais zufriedenstellend schalten, kann die Anordnung der Bauteile für eine kompakte Fixierung geplant werden. Um die Kabelwege nicht unnötig lang zu gestalten und die Fläche klein zu halten, empfiehlt es sich auf zwei oder mehreren Ebenen zu arbeiten.

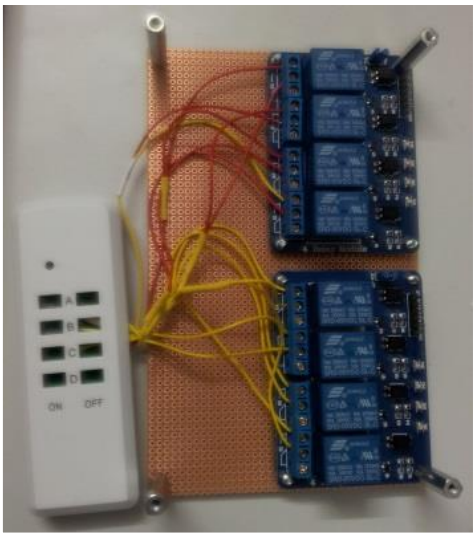

Abbildung 13: Relais auf einer Lötplatte

In diesem Beispiel werden zwei Ebenen verwendet. Als Platte können Leiterplatinen dienen, welche durch aufgebohrte Löcher verschraubt werden. Auf der unteren Platine sind die beiden Relais-Module und die Fernbedienung angeordnet. Das Batteriefach der Fernbedienung sollte allerdings noch einfach zu erreichen sein.
Auf der oberen Platine sind Arduino Uno und Ethernet-Shield aufeinander gesteckt und verschraubt. Die Position sollte auch idealerweise am Rand der Platine, mit den Anschlüssen für Ethernet, Strom und USB nach außen sein. Das ermöglicht den vereinfachten Zugang der Anschlüsse, falls die Konstruktion in eine Box gesetzt werden soll.

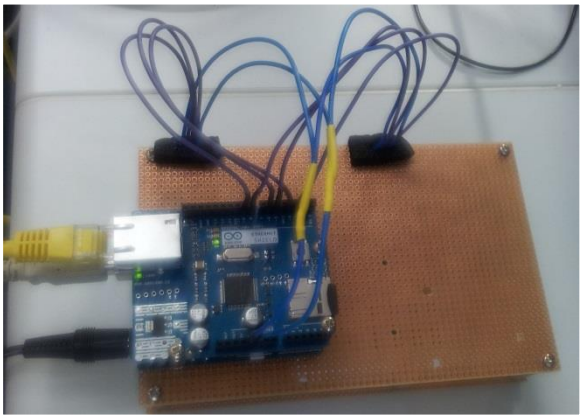

Abbildung 14: Oberer Teil des Zusammenbaus mit Arduino und Shield

6 Programmierung

Das Programm mit dem Webserver, den Webseiten und der Pin-Steuerung besteht aus einer Klasse.

```
#include <SPI.h>
#include <Ethernet.h>
```

Einbinden der benötigten Bibliotheken, die der Shield-Erweiterung und der Netzwerkkommunikation.

```
//Schalter mit Adressen deklarieren
int a_ON = 2;
int a_OFF = 3;
int b_ON = 5;
int b_OFF = 6;
int c_ON = 8;
int c_OFF = 9;

boolean a_enabled = false;
boolean b_enabled = false;
boolean c_enabled = false;
```

Setzen der Variablen, welche mit den gewählten Pins übereinstimmen müssen.

```
//---------------HEIM-Netz-------------------------
// Deklaration der MAC-Adresse des Ethernet-Ports
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// Deklaration einer statischen IP-Adresse innerhalb des Heimnetzwerks
byte ip[] = { 192,168,178,65 };
// DNS
byte dns1[] = {192,168,178,65};
// Gateway
byte gateway[] = { 192,168,178,1 };
// Subnetz-Maske
byte subnet[] = { 255, 255, 255, 0 };
```

Deklaration einer statischen, selbstgewählten MAC-Adresse. IP-Adresse, DNS, Gateway und Subnetz muss nur selbst vergeben werden wenn DHCP nicht genutzt wird.

```
// Default-Port 80
EthernetServer server(80);
EthernetClient client;

// String zum lesen der URL
String readString;
// login variablen
String user = "?id=Admin";
String password = "&pass=1234";
boolean security = false;
```

Variablen für den Server, das Speichern der Zeichen in der URL, sowie für den Login (hier inkl. Präfixe) werden deklariert.

```
void setup() {
 // initialisieren des Serial Monitors
  Serial.begin(9600);
   while (!Serial) {
    ; // warten bis der Serial-Port verbunden ist
  }

  // PINs als Ausgänge setzen

  pinMode(a_ON, OUTPUT);
  pinMode(a_OFF, OUTPUT);
  pinMode(b_ON, OUTPUT);
  pinMode(b_OFF, OUTPUT);
  pinMode(c_ON, OUTPUT);
  pinMode(c_OFF, OUTPUT);

  // Pegel der PINs auf HIGH setzen, da die Relais auf LOW-Pegel schalten
  digitalWrite(a_ON, HIGH);
  digitalWrite(a_OFF, HIGH);
  digitalWrite(b_ON, HIGH);
  digitalWrite(b_OFF, HIGH);
  digitalWrite(c_ON, HIGH);
  digitalWrite(c_OFF, HIGH);

  // starten der Ethernet-Verbindung und des Servers
  Ethernet.begin(mac, ip, dns1, gateway, subnet);
  server.begin();

  // Ausgabe der Adresse
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());

// Logout schaltet alle Schalter aus
}
```

Die **„setup()“**-Methode wartet zunächst solange bis der Serial-Port verbunden ist.
Danach werden die Pins für die drei Funksteckdosen als Ausgänge festgelegt und ihre Pegel standardmäßig auf High gesetzt.
Nun werden die Ethernetverbindung und der Server gestartet. Die Adresse wird im Serial Monitor ausgegeben.

```
void loop() {
  // Erstellen einer Client-Verbindung (Browser)
  client = server.available();
  // wenn die Verbindung besteht..
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        // auslesen der Chars aus der HTTP
        if (readString.length() < 100) {
          // speichern der Zeichen in einen String
          readString += c;
          // debug Ausgabe
          //Serial.print(c);
          //Serial.println(readString);
         }
```

In der **loop()**-Methode wird zunächst eine Clientverbindung (Browser) aufgebaut, welche so lange wartet, bis ein Client verbunden ist.
Danach werden die Zeichen aus der URL ausgelesen und in einer String-Variablen gespeichert.

Die Methode „**void login()“** baut die Seite auf, welche den Benutzer nach seinem Benutzernamen und dem zugehörigen Passwort abfragt um unautorisierten Benutzern keine Möglichkeit zu geben auf das Schaltinterface zugreifen zu können. Werden diese Daten korrekt eingegeben, gelangt der User zum Interface über welches er die Funksteckdosen bedienen kann. Die Methode **„void control()“** ist für den Aufbau der Bedieneroberfläche zuständig. Durch die Betätigung eines Schalters wird der URL eine bestimmte Zeichenkette angehängt.

```
client.println("<H2>Pflanzenbew&aumlsserung</H2>");
client.println("<br />");
client.println("<a href=\"/?aON\"\">Schalter A ON</a>");
client.println("<a href=\"/?aOFF\"\">Schalter A OFF</a><br />");
if(a_enabled){
  client.println("<p style= background-color:#00FF00>ON</p>");
}
else{
  client.println("<p style= background-color:#FF0000>OFF</p>");
}

client.println("<br />");
client.println("<H2>Licht</H2>");
client.println("<br />");
client.println("<a href=\"/?bON\"\">Schalter B ON</a>");
client.println("<a href=\"/?bOFF\"\">Schalter B OFF</a><br />");
if(b_enabled){
  client.println("<p style= background-color:#00FF00>ON</p>");
}
else{
  client.println("<p style= background-color:#FF0000>OFF</p>");
}

client.println("<br />");
client.println("<H2>Kaffemaschine</H2>");
client.println("<br />");
client.println("<a href=\"/?cON\"\">Schalter C ON</a>");
client.println("<a href=\"/?cOFF\"\">Schalter C OFF</a><br />");
if(c_enabled){
    client.println("<p style= background-color:#00FF00>ON</p>");
  }
  else{
    client.println("<p style= background-color:#FF0000>OFF</p>");
  }
```

Die **stringReader ()**-Methode vergleicht diese Zeichenkette und löst bei erfolgreichen Vergleich bestimmte Ereignisse aus, wie User und Passwort Abfrage oder Schalter betätigen.

```
// Methode für String-Vergleiche
void stringReader(){
    //prüfe ob Username und Passwort stimmen
      if (readString.indexOf(user+password)>0){
        security = true;
      }
      // Schalter AN und AUS
      if (readString.indexOf("?aON") >0){
          a_enabled = true;
          digitalWrite(a_ON, LOW);
          delay(500);
          digitalWrite(a_ON,HIGH);

      }
      if (readString.indexOf("?aOFF") >0){
          a_enabled = false;
          digitalWrite(a_OFF, LOW);
          delay(500);
          digitalWrite(a_OFF,HIGH);


      }
```

```
void shutdownAllSwitches(){
    digitalWrite(a_OFF, LOW);
    digitalWrite(b_OFF, LOW);
    digitalWrite(c_OFF, LOW);
    delay(500);
    digitalWrite(a_OFF,HIGH);
    digitalWrite(b_OFF,HIGH);
    digitalWrite(c_OFF,HIGH);
    a_enabled = false;
    b_enabled = false;
    c_enabled = false;
}
```

Durch die Methode „**shutdownAllSwitches()**“ werden alle Schalter ausgeschaltet. Dies soll zur Sicherheit passieren, wenn man sich ausloggt.

7 Weiterentwicklungsmöglichkeiten

Während der Entwicklung und Arbeit an diesem Projekt wurden die Möglichkeiten, aber auch die entstandenen Grenzen klar. Für die Erweiterung auf mehr als vier Steckdosen, könnten verschiedene Strategien in Betracht gezogen werden. Der Arduino Uno ist mit seinen vorhandenen Pins begrenzt und kann mit diesem Projekt nicht mehr als vier Funkkanäle ansteuern. Ein Wechsel auf ein größeres Entwicklungsboard wäre eine Variante zum Ausbau der Ausgänge. Die Nutzung eigener 433 MHz Sender und Empfänger als Bauteil und programmierbare Funksteckdosen mit Dip-Schalter, würde die Erstellung eigener Kanäle ermöglichen.

Ein Wechsel auf ein anderes Entwicklungsboard mit eigenem Betriebssystem und integrierter Schnittstelle für LAN oder WiFi könnte Kosten sparen und würde die Nutzung zusätzlicher Software ermöglichen. Somit wäre der Gebrauch einer Datenbank einfacher und könnte für mehr Sicherheit der Steuerung liefern.

Der Arduino selbst beherrscht leider kein Multi-Processing wie Threads, kann aber mit Bibliotheken ersetzt werden.

Die Integration von verschiedenen Open-Source Bibliotheken wie Arduino.js, etc. und die Auslagerung von Speicher erweitert das Potential von Mikrocontrollern.

Servomotoren, Sensoren oder eine Webcam wären gute Verbesserungen für die Bewässerungsanlage. Somit könnten die Kontrollen und Sicherheitsmechanismen ausgebaut werden.

8 Kosten

In der Kostenauflistung sind nur die grundlegenden Komponenten aufgelistet, nicht jedoch Werkzeug, Test- und Verbrauchsmaterial.

Teile	Kosten
Arduino Uno	22,00
Arduino Ethernet-Shield	30,00
Relais	7,20
Funksteckdosen-Set	17,00
Gesamtsumme	**ca. 76,20 €**

9 Zeitmanagement und Aufgabenverteilung

Personen	Planung	Materialbeschaffung	Einarbeitung
Sebastian Bohn	10	8	10
Florian Langhorst	10		10
Marko Vukadinovic	10	8	10
Johannes Bagge	10		10

Personen	Entwicklung	Testphase und Fehlerbeseitigung	Dokumentation
Sebastian Bohn	18,5	18,5	12
Florian Langhorst	17	17	12
Marko Vukadinovic	10	10	
Johannes Bagge	8	8	

Gesamtaufwand in Stunden	ca. 227 Std.

10 Fazit

Mit diesem Projekt wird aufgezeigt, das man mit wenigen Mitteln und geringen Vorkenntnissen, eine simple Heimautomation bauen kann. Das hierbei erworbene Wissen stellt eine gute Basis für mögliche Erweiterungen, Verbesserungen und weitere Projekte im Bereich „Prototyping“ dar. Durch eine riesige Variation an Bauteilen und einem großen freien Softwareangebot, sind den Ideen und ihren Umsetzungen nur wenige Grenzen gesetzt.

Der Source-Code ist unter dem Repository https://github.com/m4REK/Projekt-B zugänglich.

Abbildungsverzeichnis