Projektarbeit

zur Gesellenprüfung zum Informationselektroniker für Geräte und Systeme

Entwicklung einer Helligkeit und Tageszeitabhängigen Hühnerklappensteuerung

-Bild-

Nils Karst

Vidit-Systems GmbH

Inhaltsverzeichnis

MEIN AUSBILDUNGSBETRIEB

Mein Ausbildungsbetrieb ist die Firma Vidit Systems GmbH.

Die Vidit Systems GmbH entwickelt und produziert Komplettlösungen für die amtliche Verkehrskontrolle, automatische

Kennzeichenlesesysteme und Dokumentations- und Beweissicherungskamerasysteme für die Polizei. Unser Tätigkeitsfeld erstreckt sich von der Entwicklung- über die Produktion von Elektronik, -Software, -Hardware bis hin zur Integration der Technologie in von uns ausgebaute Sonderfahrzeuge. Zu unseren Kunden zählen Polizeidienststellen aus jedem Bundesland in Deutschland und einige in Österreich.

Aktuell befinden wir uns in der Zulassungsphase für ein neues Produkt. VKS4 soll das bereits seit über 10 Jahren verwendete VKS3.2 ablösen. In diesem Entwicklungs- und Zulassungsprozess bin ich seit ca. 1,5 Jahren ein Fester Bestandteil. Meine Aufgaben erstrecken sich von Planungen, Problemlösungen, testen der Funktionen und Präsentation von unserer Umsetzung bei der PTB (Physikalisch-Technische Bundesanstalt) in Berlin und Braunschweig.

Abbildung 1Vidit Banner

Oben zu sehen von links nach rechts:

- VKS Stationär
  - Stationäres Verkehrsüberwachungssystem in Thüringen
- VKS Bus außen
  - Mobiles Verkehrsüberwachungssystem
  - Über 70 Mal in ganz Deutschland
- VKS Bus innen
- HDSNK
  - Robustes 360 ° Kamerasystem mit Richtmikrofonen
- Catchken innen
  - automatische Erkennen von KFZ Kennzeichen in polizeilichen Fahndungsapplikationen
- Polizeiauto mit Catchken von außen

Die Ausgangslage

Bei mir Zuhause halten wir aktuell 5 Hühner. Der Hühnerstall soll sich morgens bei Tagesanbruch öffnen und abends wieder schließen. Eine Erfassung der Hühner ist nicht nötig. Ich habe das Verhalten über mehrere Tage beobachtet, mit dem Ergebnis, dass die Hühner abends mit Beginn der Dämmerung in ihr Haus gehen. In dem Hühnerhaus wurde ein Motor mit Seilzug vorinstalliert. Dieser wird aktuell Manuel über einen Taster gesteuert, dieser soll nun durch meine Entwicklung ersetz und dadurch automatisiert werden. Der Motor ist ein DC 12V Motor mit Gewindestab.

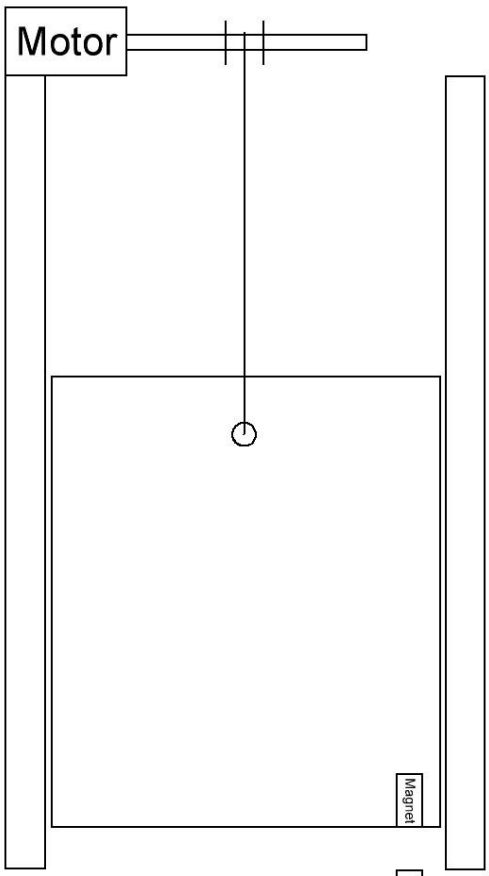


Abbildung 2 Hühnerklappe Skizze

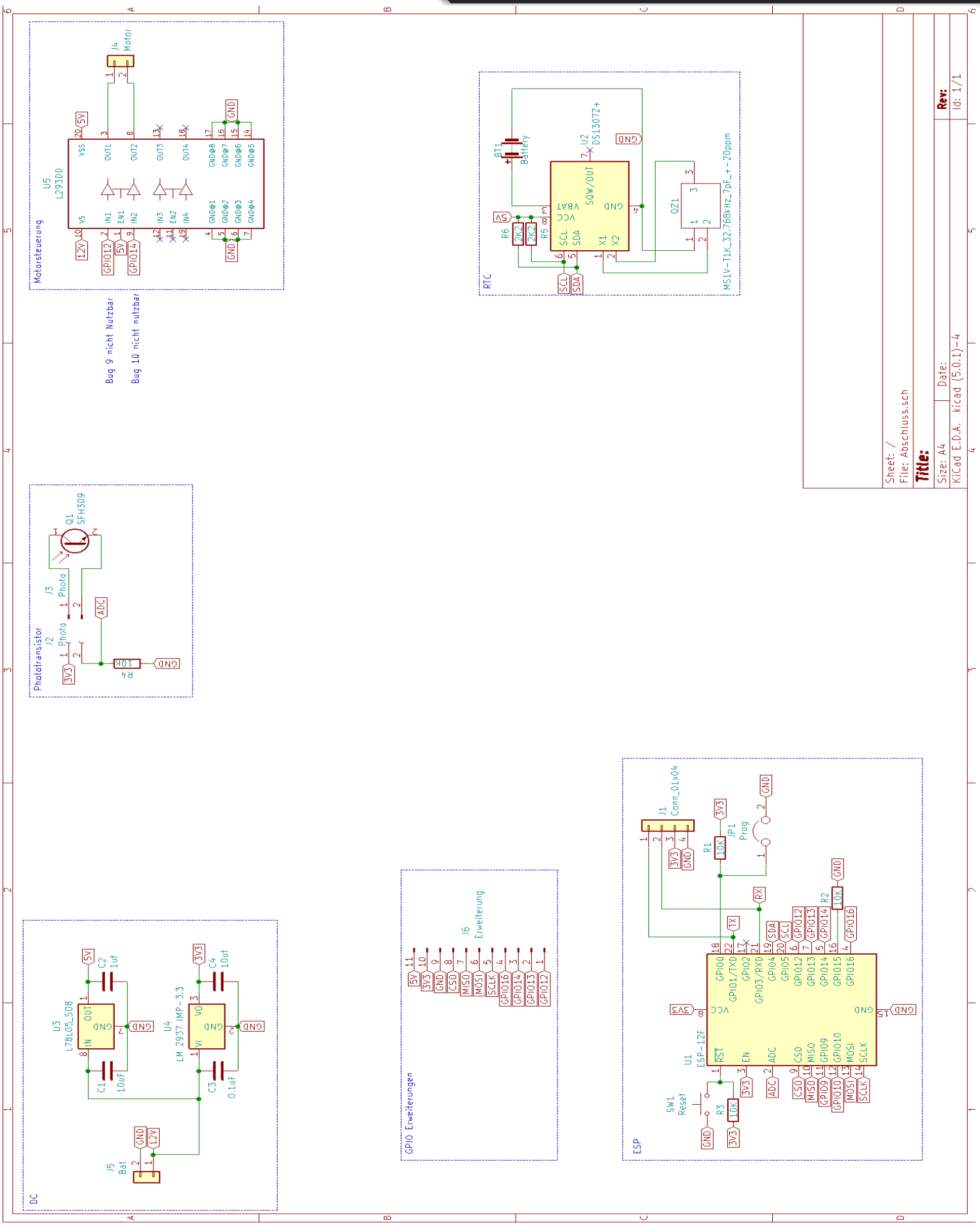
Links zu sehen ist eine Skizze zur Mechanik der Hühnerklappe. Die Tür an sich sitzt in zwei Führungsschienen und wird durch ein Seil hochgezogen bzw. heruntergelassen. Das Seil wickelt sich um den Gewindestab am Motor.

Die Automatisierung soll Folgende Funktionen enthalten:

- Tageszeitgesteuert
- Helligkeitsgesteuert
- Vorhandenen DC Motor ansteuern
- Uhrzeit speichern (RTC)

TECHNISCHE DATEN

- Versorgungsspannung: 12V
- Fototransistor zur Messung der Helligkeit
- RTC um die Zeit zu speichern
- Magnetschalter um Position der Tür zu erfassen
- DC Motor zum Antrieb der Tür


Motorsteuerung
L293DD
Motor
Bug 9 nicht Nutzbar
Bug 10 nicht nutzbar
RTC
DS1307Z+
Battery
MS1V-T1K_32.768kHz_7pF_+-20ppm
Photatransistor
SFH309
Photo
ADC
10K
DC
L78L05_SOB
LM 2937 IMP-3.3
10uf
0.1uF
Bat
GPIO Erweiterungen
Erweiterung
ESP
ESP-12F
Conn_01x04
Prog
Reset
Sheet: /
File: Abschluss.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (5.0.1)-4
Id: 1/1

ESP

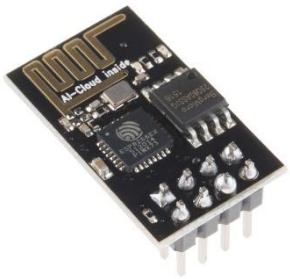


Abbildung 3 ESP12F

Was ist der ESP 12F?

Der ESP12F ist eine Kleine Platine mit minimal Konfiguration eines ESP 8266 und wird häufig für kleinere Projekte wie dieses verwendet.

ESP 8266:

Der ESP8266 ist ein kostengünstiger 32-Bit Mikrocontroller mit geringem Leistungsbedarf. Unterstützt wird unter anderem die Programmiersprache C++, welche in diesem Fall angewandt wird.

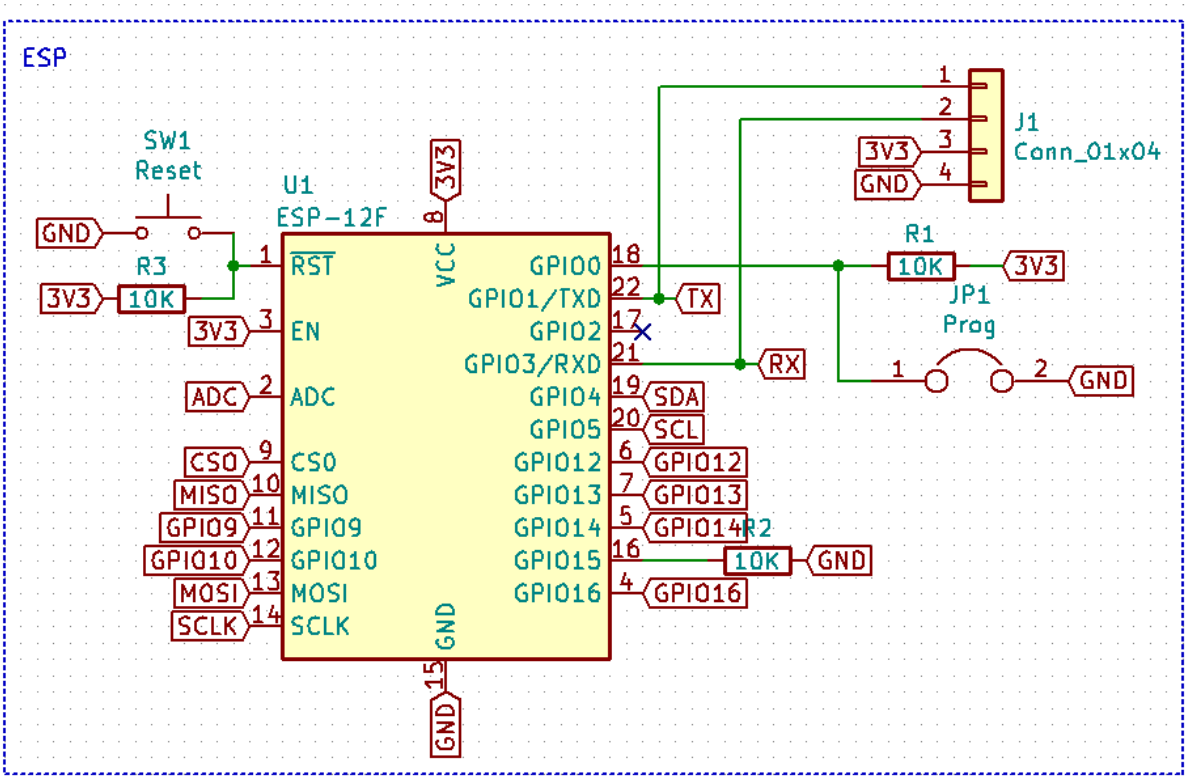


Abbildung 4 Schaltplan ESP

Der ESP12F ist wie im Datenblatt angegeben verschaltet:

An Pin 1 wird ein Resettaster (SW1) verwendet um diesen Pin von 3V3 auf Masse zu ziehen und somit ein Reset auszulösen.
An Pin 2 ist der interne ADC dieser wird mit dem Fototransistor verschaltet.
An Pin 3 werden 3V3 angelegt um den Chip zu aktivieren (Chip Enable).
Pin 18 muss beim Start auf Masse gelegt werden um den Programmiermodus zu aktivieren (JP1).
An Pin 21(Rx) und 22 (Tx) werden auf eine Stiftleiste (J1) geführt um einen UART zu USB Programmieradapter anzuschließen.
An Pin 13 wird der Magnetschalter angeschlossen.
Pin 16 muss zum Start auf Masse liegen. Um ihn dennoch verwenden zu können wird ein 10K Widerstand

zwischengeschaltet.
Die Pins 4,5,6,7,9,10,13 und 14 wurden für eventuelle Erweiterungen auf eine Pinleiste geführt.
Pin 11 und 12 sollten ursprünglich für die Ansteuerung des Motortreibers verwendet werden, hierbei ist allerdings ein Problem aufgetreten:
Die Beiden Pins werden intern für den Flash verwendet und können somit nur eingeschränkt (als Eingang) verwendet werden. Aufgrund einer nicht idealen Dokumentation im Datenblatt ist dieser Fehler erst bei der Inbetriebnahme der Platine aufgefallen. Diese Verbindungen werden jetzt getrennt und alternativ Pin 5 und 6 verwendet. Diese wurden mit Fädeldraht nachträglich verbunden (siehe Abbildung XXX).

DC Versorgung

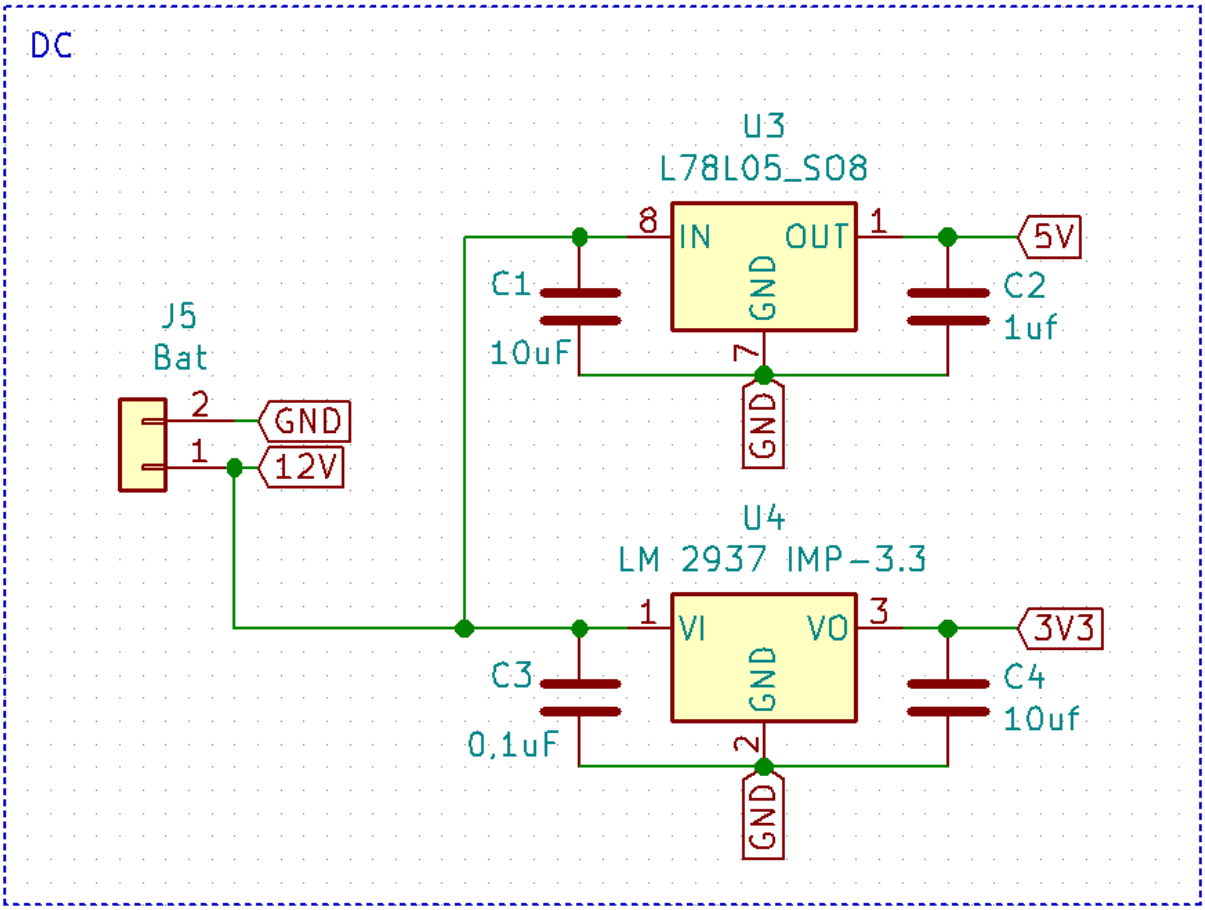
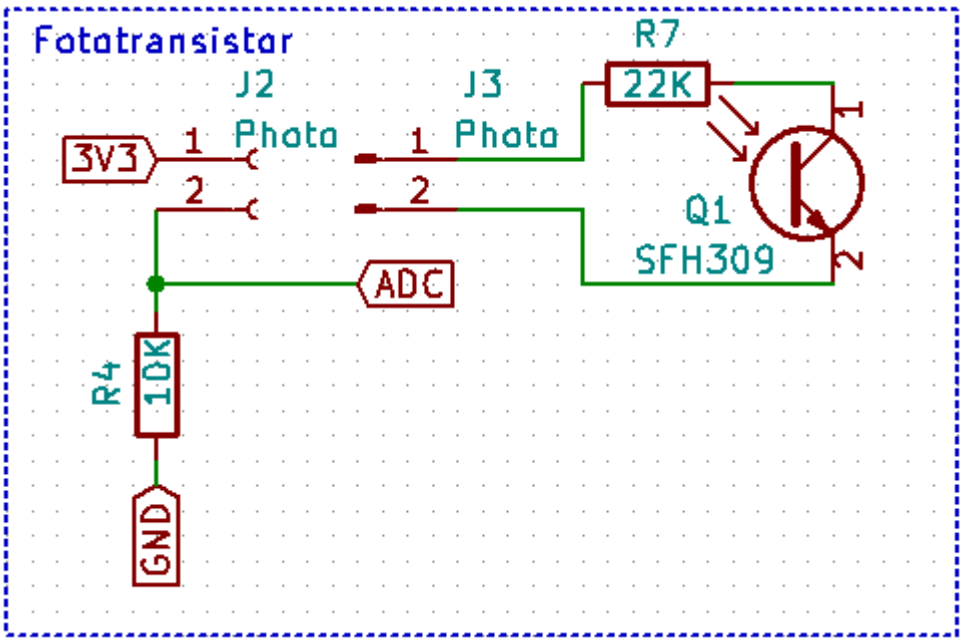


Abbildung 5 Dc Versorgung

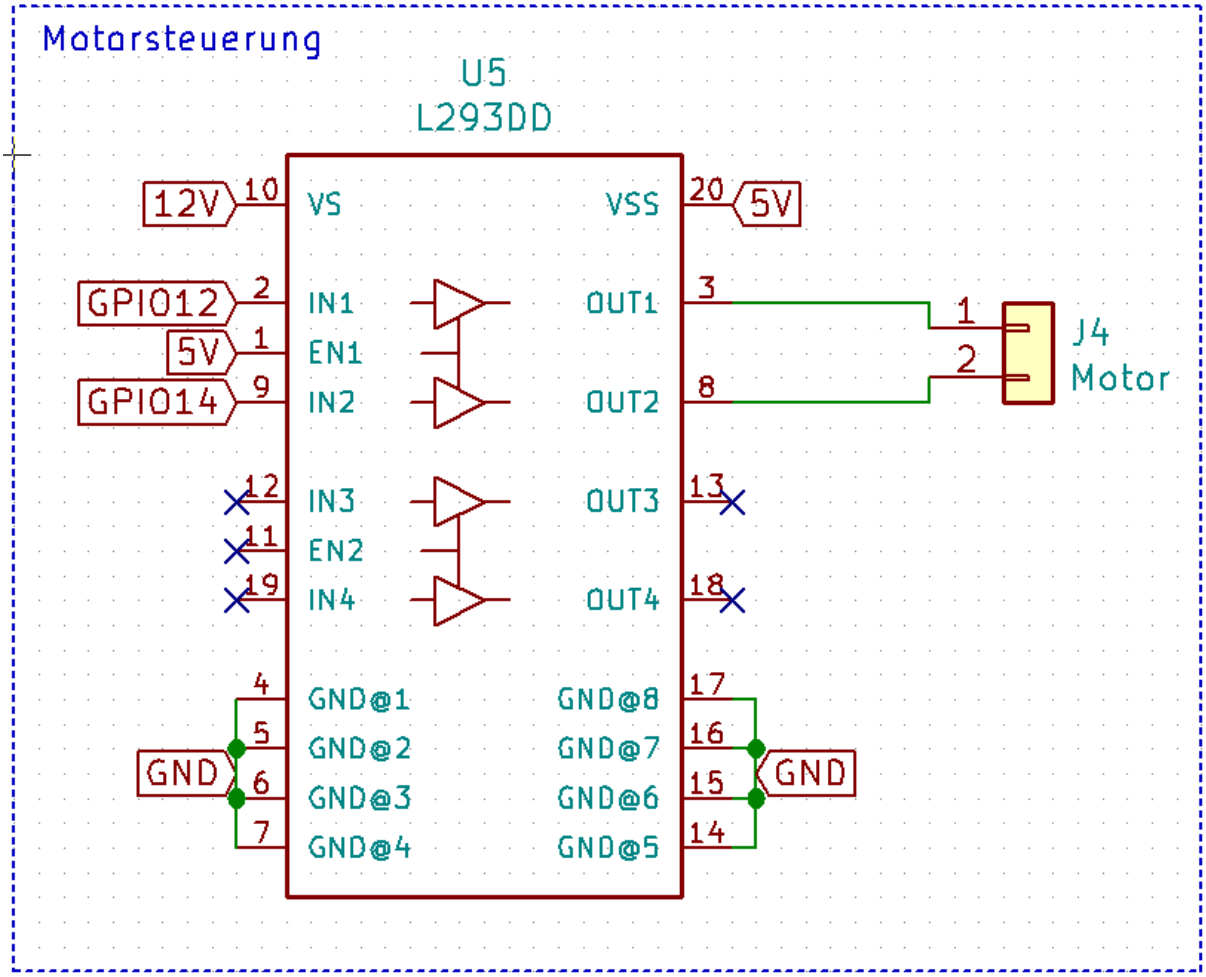
- J5 = Eingangsspannung DC 12 V
- U3 = DC/DC Wandler von 12V zu 5V
- U4 = DC/DC Wandler von 12V zu 3,3V
- Die Kondensatoren dienen der Entstörung bzw. Stabilisierung

Fototransistor



Der Fototransistor wird verwendet um die Helligkeit im Außenbereich zu ermitteln. Mittels R4 und R7 wird ein Spannungsteiler realisiert um die Maximale Spannung am ADC Pin auf 1V zu begrenzen. Dies ist Notwendig, da der interne ADC des ESP nur Spannungen zwischen 0V und 1V Messen kann. Die Messung wird von 0V = 0 bis 1V =1024 angegeben.

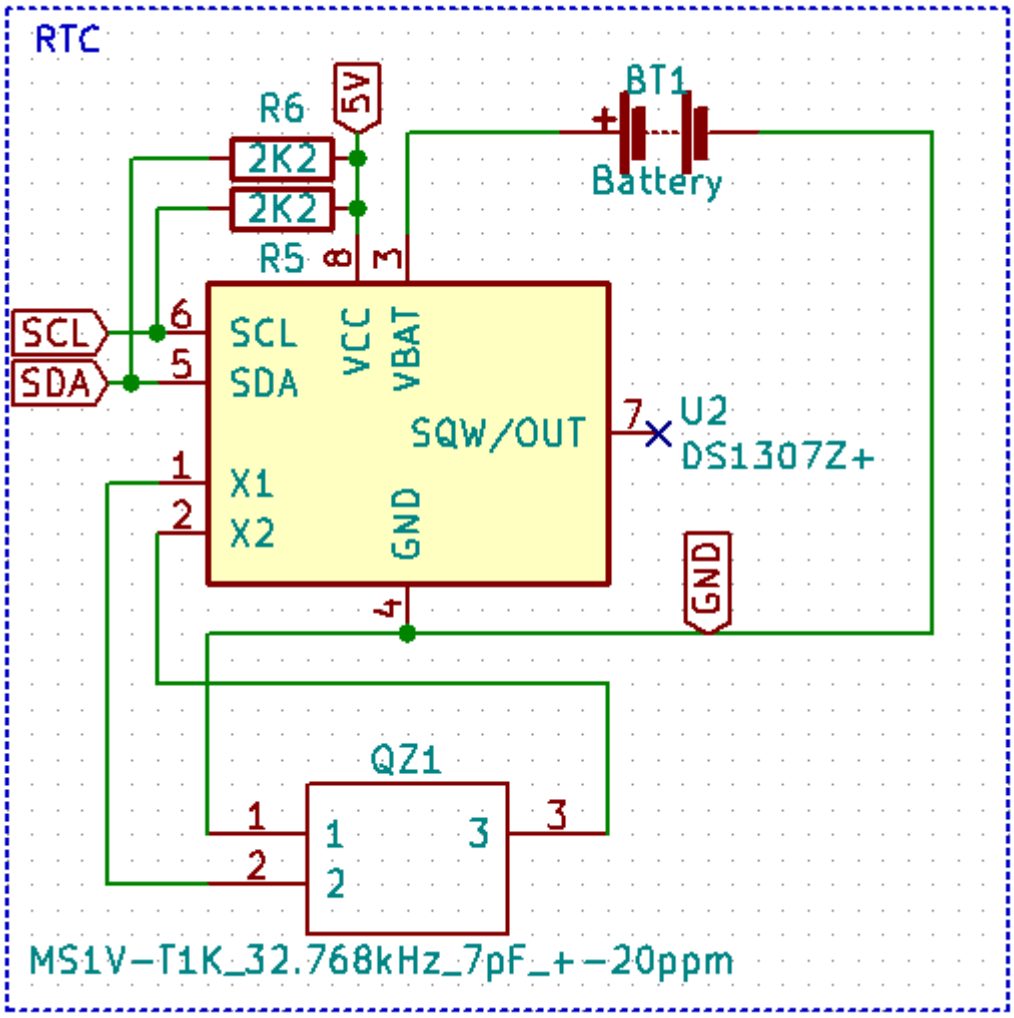
Motorsteuerung



Die Ansteuerung des DC Motors wird mithilfe des Motortreibers L293DD realisiert. Dieser wurde wie im Datenblatt beschrieben beschaltet. Die Spannung für den Motor wird an PIN 10 VS angelegt. Der Motor wird an PIN3 und 8 bzw. an J4 angeschlossen. GPIO 12 und 14 steuern den L293DD nach folgender Tabelle:

GPIO 12	GPIO14	J4 PIN1	J4 PIN2
H	L	12V	GND
L	H	GND	12V

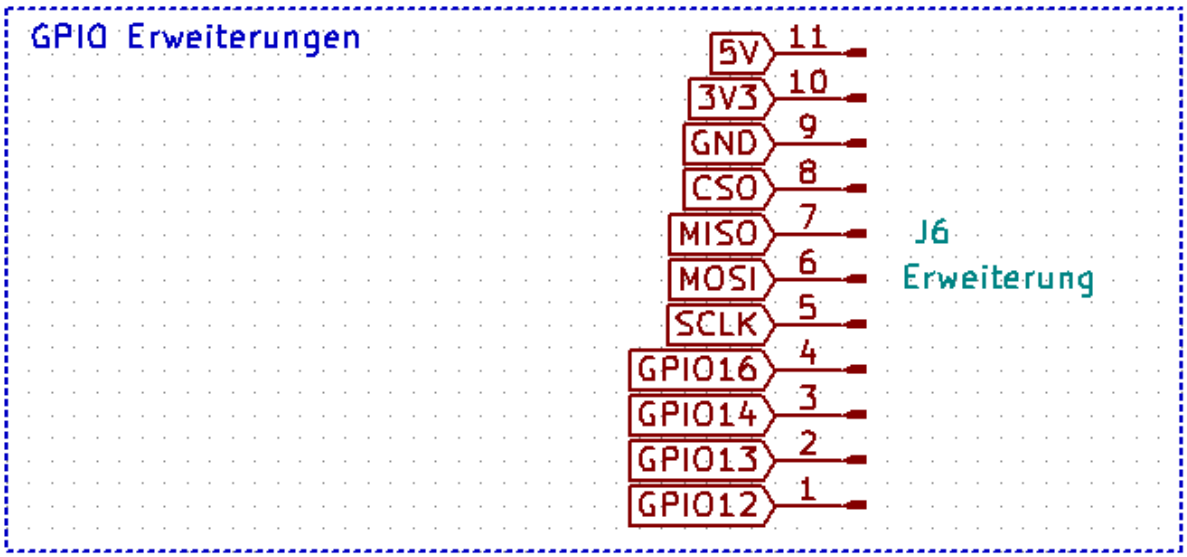
RTC



Die RTC DS1307 wird über I2C angesteuert diese PINs (SDA und SCL) sind mit dem ESP verbunden. Mittels R5 und R6 wird der für die I2C benötigte Pullup auf 5V realisiert. Der externe Quarz (QZ1) schwingt mit 32.768 kHz +-20ppm und wird verwendet um den Taktzähler der RTC zu triggern.

BT1 ist eine Knopfzelle (CR2032) die verwendet wird um die Zeit zu speichern und weiterlaufen zu lassen während die Platine nicht extern mit Spannung versorgt wird.

Erweiterungen

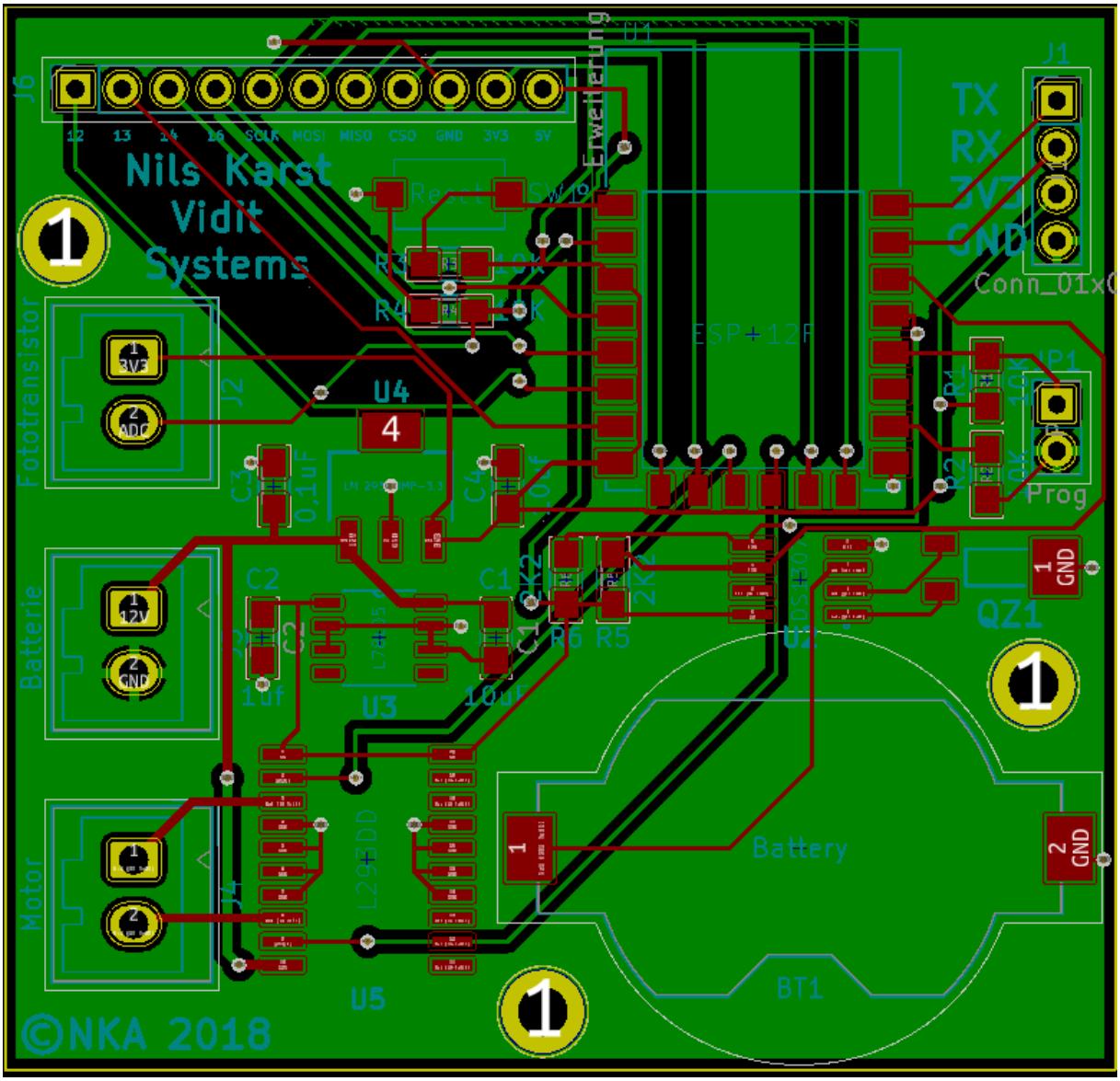


Um nachträgliche Erweiterungen und Anpassungen zu ermöglichen werden alle bis dato nicht benötigten Pins auf eine Stiftleiste geführt.

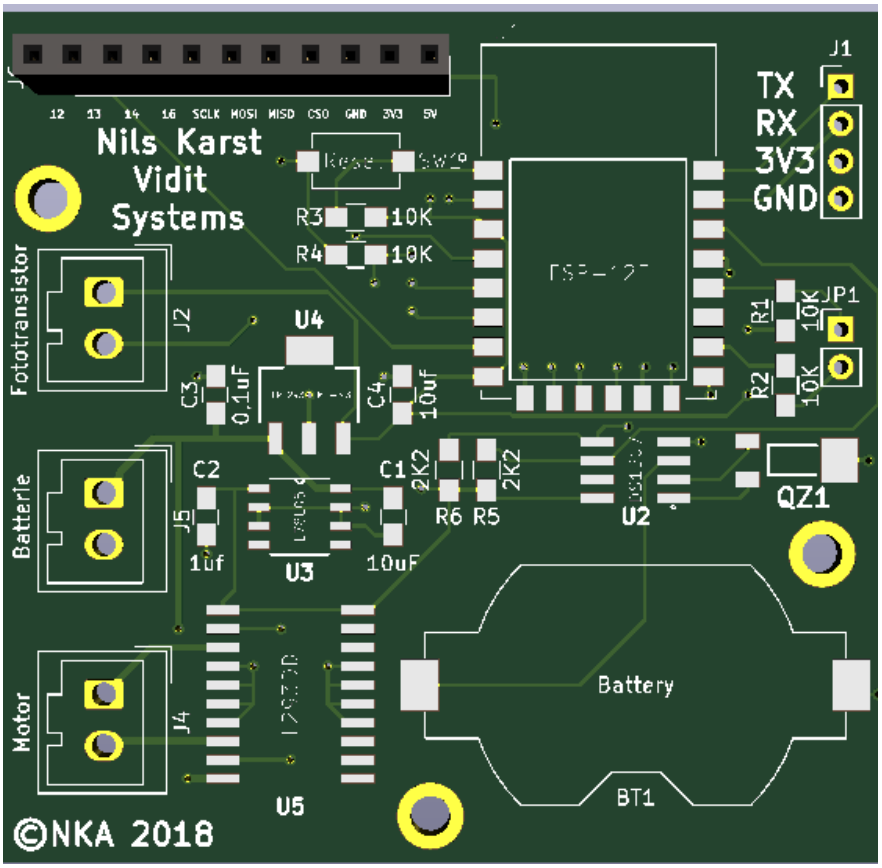
GPIO 13 wird mittlerweile als Anschlag für die Hühnerklappe benutzt. Hierzu wird ein Magnetschalter angeschlossen der bei ausreichend starkem Magnetfeld GPIO 13 auf Masse zieht. An der Hühnerklappe wurde unten ein Magnet angebracht.

GPIO 12 und 14 werden dank eines Bugs (siehe Schaltplan ESP) mittlerweile für die Ansteuerung des Motortreibers verwendet werden.

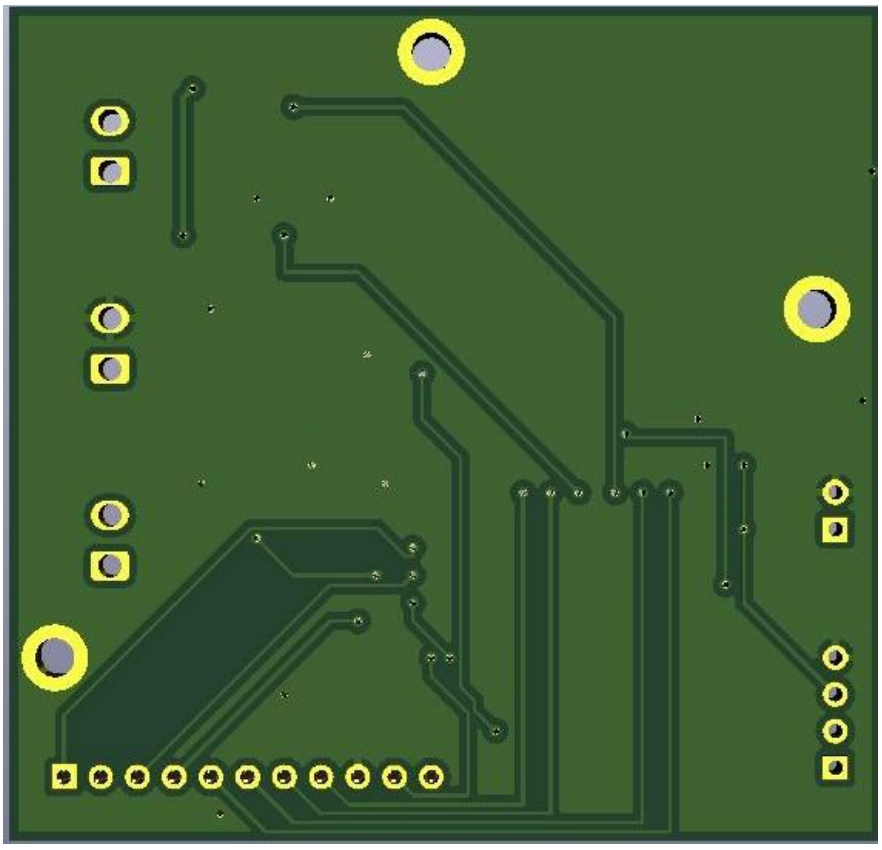
Gesamt


Nils Karst
Vidit
Systems
Erweiterung
J1
TX
RX
3V3
GND
Conn_01x0
ESP+12F
JP1
Prog
Fototransistor
Batterie
Motor
U4
U3
U5
U2
QZ1
Battery
BT1
GND
©NKA 2018

Oben


12 13 14 16 SCLK MOSI MISO CS0 GND 3V3 5V
Nils Karst
Vidit
Systems
Reset
SW2
R3 10K
R4 10K
J1
TX
RX
3V3
GND
JP1
R1 10K
R2 10K
Fototransistor
J2
U4
C3 0.1uF
C4 10uF
Batterie
J5
C2
1uf
U3
C1 2K2
R6 2K2
R5
U2
QZ1
Motor
J4
U5
Battery
BT1
©NKA 2018

Unten



Bestückt oben

Bild

Bestückt unten

Bild

Nummer	Bezeichnung	Beschreibung	Kosten (vorhanden)
1	U2	DS 1307Z :: Real Time Clock I²C 56B NV SRAM, SO-8	1,45€
2	QZ1	32,768 MS5V-12,5 :: Uhrenquarz, Metallgehäuse, 1,15x1,15x4,7mm, 12,5pF	1€
3	BT1	Knopfzellenhalter 1x CR 2032	2€
4	BT1a	Knopfzelle CR 2032	0€
5	R5,6	2K2	0,1€
6	U1	ESP-12F	4€
7	J1	Programmierstecker	0€ (0,1€)
8	R1,2,3	10K	0,1€
9	SW1	Reset Taster	0,5€
10	JP1	Jumper	0€ (0,1€)
11	U3	5V L78L05ACD	0,2€
12	U4	3,3V LM 2937 IMP-3.3	1,5€
13	C1,4	10uF	0,1€
14	C2	1uF	0,1€
15	C3	0.1uF	0,1€

16	Q1	Photodiode SFH 309 FA	0,25€
17	R4	10K	0,1€
18	J3	Wago Buchse 2 Pol	0€ (0,5€)
19	U5	L293DD	2,5€
20	J4	Wago Buchse 2 Pol	0€ (0,5€)
21	J5	Wago Buchse 2 Pol	0€ (0,5€)
22	SW2	Magnetschalter	0€ (8€)
23	M1	DC Motor mit Gewindestab	0€ (15€)
24	Platine	2 Layer Platine	35€
Gesamtkosten:			49€ (73,70€)

Arduino Studio

Um den Quellcode zu entwickeln wird die Software „Arduino Studio“ in der Version 1.8.8 verwendet. Diese Entwicklungsumgebung basiert auf der Programmiersprache „C++“ und ist durch ihre Kompatibilität zu vielen gängigen Mikroprozessoren sehr einsteigerfreundlich, auch der ESP8266 wird unterstützt.

Erklärung des Quellcodes

Der gesamte Quellcode befindet sich im Anhang im folgendem Textabschnitt wird sich lediglich auf einzelne Ausschnitte daraus bezogen um die Funktion besser erklären zu können.

Der Code Stammt direkt aus der Arduino Umgebung und ist von mir selbst geschrieben und Kommentiert.

Zusätzlich verwendet wurden folgende Bibliotheken:

- Wire.h
  - https://www.arduino.cc/en/Reference/Wire
  - Diese Bibliothek wird verwendet um Kommunikation über den I2C Bus zu ermöglichen. Die RTC kommuniziert über I2C.

- RTClib.h
  - https://github.com/adafruit/RTClib
  - Diese Bibliothek wird für die Kommunikation mit der RTC verwendet.

Initialisierung

```
void setup() {
  pinMode(M1, OUTPUT);
  pinMode(M2, OUTPUT);
  pinMode(LDR, INPUT);
pinMode(T1, INPUT_PULLUP);

  Serial.begin(115200);

  if (! rtc.begin()) {
  Serial.println("Verbindung zur RTC
fehlgeschlagen");
    while (1);
  }

  if (! rtc.isrunning()) {
    Serial.println("Zeit nich enthalten");
    Serial.println("Setze Zeit...");
    rtc.adjust(DateTime(F(__DATE__),
F(__TIME__)));
  }



if ( digitalRead(T1) == HIGH ){
    Serial.println("Tür offen -->
kalibrierung");
    while( digitalRead(T1) == HIGH ){
      runter();
      delay(100);


    Serial.println("Fahre runter");
    }
    aus();
    State=0;
    Serial.println("Kalibriert");
```

Die „void setup“ Schleife wird beim Einschalten nur ein einziges Mal aufgerufen. Diese dient der Initialen Kalibrierung um einen definierten Ausgangspunkt zu haben, wenn die Main Routine gestartet wird. Zuerst werden die benutzten GPIOs als Aus. bzw. Eingänge konfiguriert.

In der ersten IF Schleife wird nur für Debugging verwendet, wenn die RTC Verbindung fehlgeschlagen ist, dann wird "Verbindung zur RTC fehlgeschlagen" ausgegeben. Bei erfolgreicher Verbindung hat diese Schleife keine Funktion.

Die zweite Schleife fragt ab, ob in der RTC eine Zeit enthalten ist, wenn nicht wird die Zeit vom Kompilieren gesetzt. Auch diese Schleife hat nur im Fehlerzustand eine Funktion.

Als nächstes wird in der 3. Schleife die Tür nach unten gefahren, wenn sie dies nicht bereits ist. Dies ist nötig um wie oben erwähnt einen definierten Startpunkt für die Main Routine zu haben.

Danach wir noch die Aktuelle Zeit über die COM Schnittstelle ausgegeben. Dies ist ebenfalls eine Debugging Funktion, deshalb hier rausgekürzt.

Main Routine

Uhrzeit

```
DateTime now = rtc.now();
int std = now.hour();
```

Die Zeit im ESP wird mit der Zeit aus der RTC gleichgesetzt. Dies ist Notwendig, da die interne RTC des ESP sehr ungenau ist. Um die Stunden später in einer Abfrage verwenden zu können wird die aktuelle Stunde in die variable „std“ eingelesen.

Fototransistor auslesen

Der aktuelle Wert des ADC wird in die Variable „sensorValue“ eingelesen um ihn später in einer Abfrage zu verwenden.

```
sensorValue = analogRead(LDR);
```

Hauptprogramm

```
if((State == 0 )&& (sensorValue >= 101) && (std >= 5 ) && (std <= 18 )) {

    hoch();

    delay(5000);

    aus();

    Serial.println("oben");

    State=1;
  }
```

Im Hauptteil der Main Routine wird die Tür letztendlich angesteuert.
Die erste IF Schleife fragt ab ob die Tür unten ist, ob es hell ist (>=101) und ob es zwischen 5 und 18 Uhr ist. Sind all diese Bedingungen „WAHR“ fährt der Motor die Tür für 5 Sekunden nach oben und setzt den Status auf „oben“.

Die zweite If Schleife fragt ab ob die Tür oben ist, ob es dunkel ist (<=100) und ob es nach 18 und vor 5 Uhr ist. Sind all diese Bedingungen „WAHR“ fährt der Motor die Tür nach unten bis der Reedkontakt auslöst und GPIO13 auf Masse zieht danach wird der Status auf „unten“ gesetzt.

```
 if((State == 1 )&& (sensorValue <= 101) && (std <= 4 ) && (std >= 19 )) {

    while( digitalRead(T1) == HIGH ){

      runter();

      delay(100);

      Serial.println("Fahre runter");
    }
    aus();

    Serial.println("unten");

    State=0;

  }
```

```
void hoch(){

  digitalWrite(M1, HIGH);

  digitalWrite(M2, LOW);

}

void runter(){

  digitalWrite(M1, LOW);

  digitalWrite(M2, HIGH);

}

void aus(){

  digitalWrite(M1, LOW);
  digitalWrite(M2, LOW);

}
```

Um die Motorsteuerung in der Main Routine übersichtlicher zu gestalten habe ich 3 Unterprogramme zum Hochfahren, Runterfahren und Anhalten geschrieben diese Steuern den Motortreiber über GPIO 12 und 14 an.

GPIO 12	GPIO14	J4 PIN1	J4 PIN2
H	L	12V	GND
L	H	GND	12V

AUSBLICK

Mögliche Optimierung und Erweiterungen in zukünftigen Versionen:

- Spannungsversorgung um Solarzelle und Pufferbatterie erweitern
- Layout anpassen
  - GPIO 12 und GPIO 14 nichtmehr per Fädeldraht an den Motortreiber anbinden
  - Eine Buchse für den Magnetschalter an GPIO 13 einfügen
- Steuerung und Statusausgabe per WLAN
- Android App statt / zusätzlich zur Webseite

QUELLEN

Quellcode

```
/* Hühnerklappe V1
 *  Nils Karst
 *  Vidit-systems
 *  15.01.19
 */

#include <Wire.h>                                                    //I2C Bus Lib
#include "RTClib.h"                                                  //RTC Lib
RTC_DS1307 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};                                                         //Variable für Anzeige des
                                                                     Wochentags


int sensorValue = 0;                                                 //Sensor Wert (Standard =)
int State = 0;                                                       //Status der Klapppe
#define M1 D5                                                        //Motorsteuerung Pin 1
#define M2 D6                                                        //Motorsteuerung Pin 2
#define LDR A0                                                       //Fototransistor am
                                                                     ADC (Pin A0)
#define T1 D7                                                        //Taster an GPIO13

void setup() {                                                       //Code läuft am Start
                                                                     einmal durch
  pinMode(M1, OUTPUT);                                               //M1,M2 als Ausgang
  pinMode(M2, OUTPUT);
  pinMode(LDR, INPUT);                                               //LDR als Eingang
  pinMode(T1, INPUT_PULLUP);                                         //Taster als Eingang
                                                                     mit Pullup


  Serial.begin(115200);                                              //Serielle Komunikation
                                                                     aktivieren Baudt 115200


  if (! rtc.begin()) {                                               //Für Debugging
    Serial.println("Verbindung zur RTC fehlgeschlagen");             //Wenn rtc.begin aus lib
                                                                     nicht erfolgreich ausgeführt
                                                                     wurde (--> keine Verbindung
                                                                     zur RTC)
    while (1);                                                       //wird seriell "Verbindung
                                                                     zur RTC fehlgeschlagen"
                                                                     ausgegeben


  }

  if (! rtc.isrunning()) {                                           //Für Debugging
    Serial.println("Zeit nich enthalten");
    Serial.println("Setze Zeit...");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));                  //Falls keine Zeit in der RTC
                                                                     gesetz war wird die Zeit des
                                                                     Kompilierens gesetzt

  }
```

```
if ( digitalRead(T1) == HIGH ){                                    //Wenn offen runter
                                                                   fahren bis geschlossen
    Serial.println("Tür offen --> kalibrierung");
    while( digitalRead(T1) == HIGH ){
      runter();
      delay(100);


    Serial.println("Fahre runter");
    }
    aus();
    State=0;
    Serial.println("Kalibriert");

    DateTime now = rtc.now();                                      //RTC Zeit auslesen
    Serial.print(now.year(), DEC);                                 //Aktuelle Zeit und
                                                                   Datum ausgeben (COM)
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" (");
    Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
    Serial.print(") ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

  }
}

void loop() {
  DateTime now = rtc.now();
  int std = now.hour();                                //aktuelle Stunde einlesen
  sensorValue = analogRead(A0);                        //Einlesen des ADC 0....1024

  if((State == 0 )&& (sensorValue >= 101) && (std >= 5 ) && (std <= 18 )) {     // Wenn die Tür
                                                                   unten, hell, zwischen
                                                                            5 und 18 Uhr
                                                                      --> Tür hochfahren

    hoch();
    delay(5000);
    aus();
    Serial.println("oben");
    State=1;
  }
```

```
if((State == 1 )&& (sensorValue <= 101) && (std <= 4 ) && (std >= 19 )) {  // Wenn die Tür
                                                                              oben, dunkel, vor
                                                                              5 und nach 18 Uhr
                                                                           --> Tür runterfahren
    while( digitalRead(T1) == HIGH ){
      runter();
      delay(100);
      Serial.println("Fahre runter");
    }
    aus();
    Serial.println("unten");
    State=0;


  }
}

void hoch(){                                                              //Unterprogramme
  digitalWrite(M1, HIGH);                                                 //Beschaltung
                                                                          siehe Datasheet LS293DD
  digitalWrite(M2, LOW);
}

void runter(){
  digitalWrite(M1, LOW);                                                  //Beschaltung
                                                                          siehe Datasheet LS293DD
  digitalWrite(M2, HIGH);
}
void aus(){
  digitalWrite(M1, LOW);                                                  //Beschaltung
                                                                          siehe Datasheet LS293DD
  digitalWrite(M2, LOW);
}
```