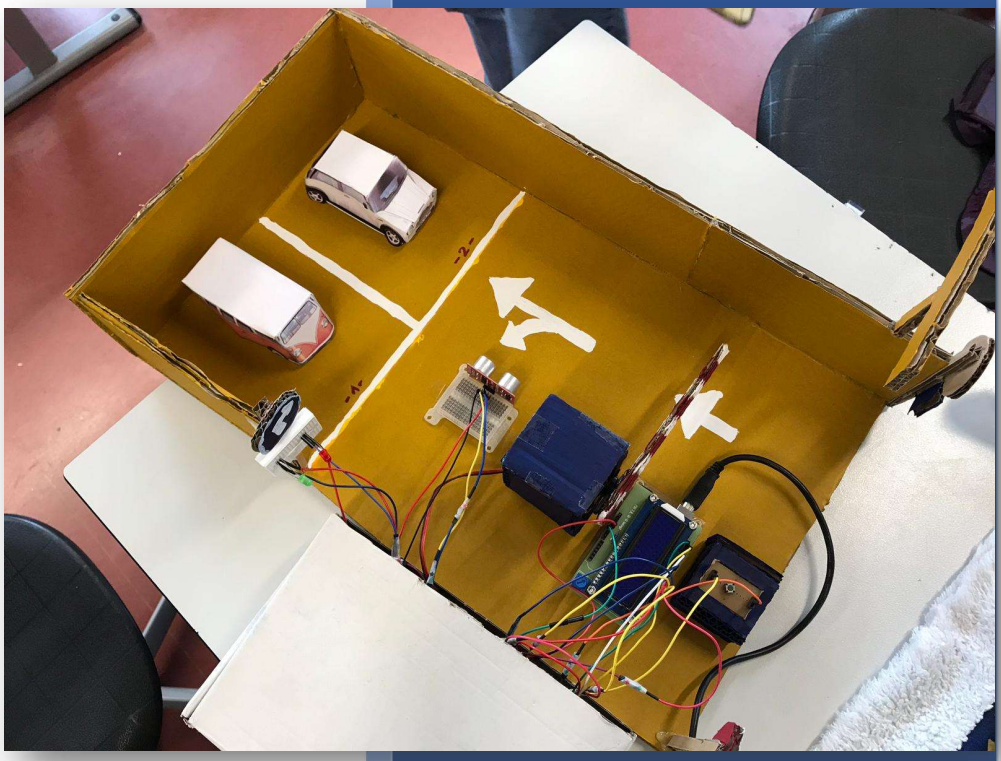
NwT-Projekt

Beginn: 21.05.2019
Abgabe: 11.07.2019

Das Parkhaus- ein Arduino-Projekt



Beteiligte:
Alisa Queck, Carina Steil, Paola Sorce und Simon Wödl

Kepler Gymnasium
Klasse 9c
Herr Glatthaar

Inhaltsverzeichnis

1. VORBEREITUNGSPHASE

1.1 Projektauftrag

Baue ein Parkhaus mit diversen Funktionen, die vom Arduino angetrieben werden.

1.2 Projektziele

- Das Parkhaus hat eine Ampel, mit rotem und grünem Licht
- Das Parkhaus besitzt eine Schranke, die sich automatisch öffnet und schließt, nach Betätigen eines Tasters
- Die Zeit des Parkhausaufenthaltes soll gestoppt werden und auf dem LCD ausgegeben werden
- Neben dem Parkhaus befindet sich eine Anzeige mit Uhrzeit und Datum
- Der Zustand des Parkhauses soll auf dem LCD wiedergegeben werden

1.3 Projektbeschreibung

Ein Auto will in ein Parkhaus fahren und hält vor der Schranke an. Neben der Schranke befinden sich zwei Lampen, eine rote und eine grüne.

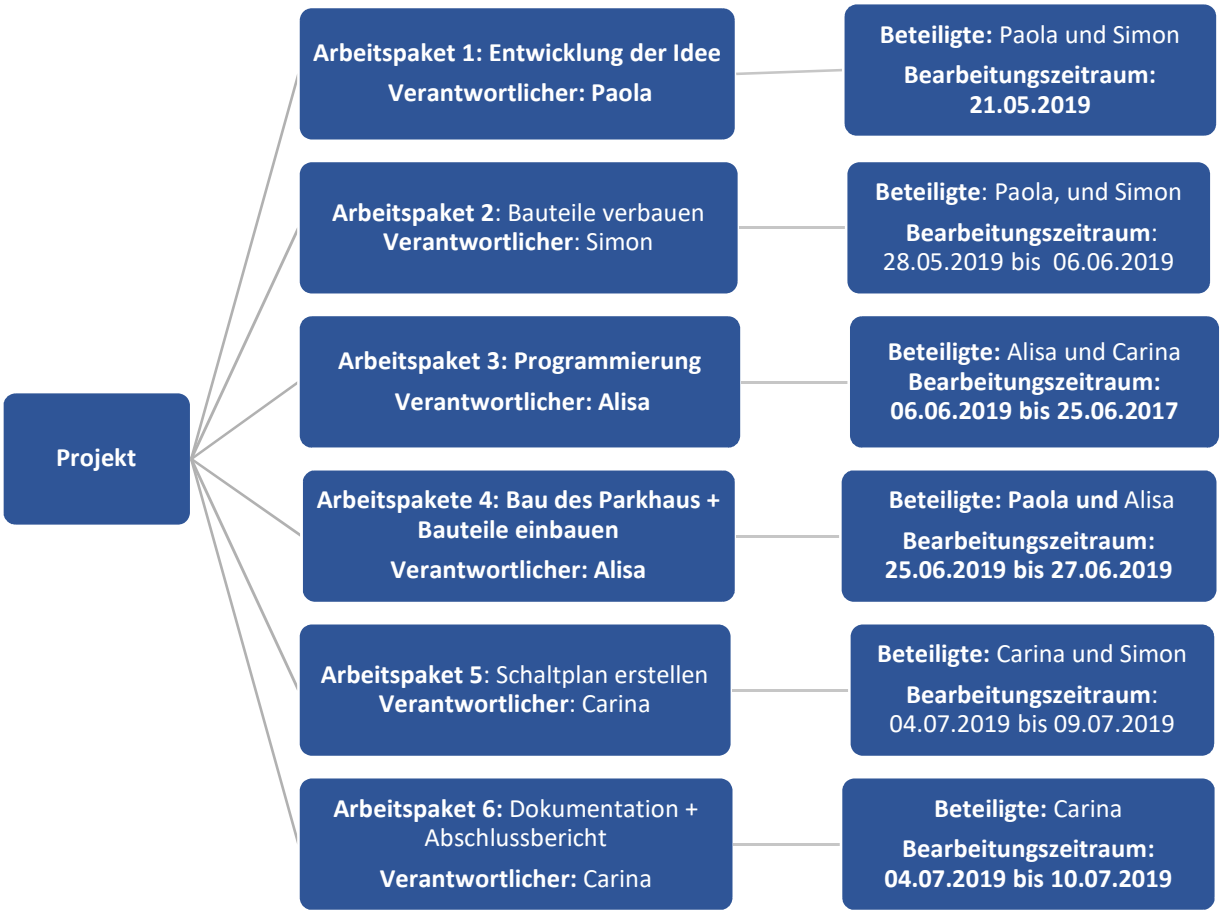
Auf dem LCD steht die Uhrzeit, das Datum und frei. Außerdem leuchtet die Lampe rot. Sobald der Taster von dem Autofahrer gedrückt wird, geht die grüne Lampe an. Die Schranke öffnet sich und schließt sich automatisch nach 5 Sekunden wieder, somit wird das Lämpchen wieder rot.

Ab jetzt soll die Zeit des Aufenthaltes gestoppt werden. Sind alle Parkplätze besetzt, so steht auf dem LCD „voll“.

Fährt das Auto wieder aus dem Parkhaus heraus, so kommt es an dem Ultraschallsensor vorbei. Die Schranke geht automatisch wieder auf und das Lämpchen wird grün. Nach 3 Sekunden schließt sich automatisch die Schranke wieder. Auf dem LCD wird nun die Zeit des Aufenthaltes angegeben.

2. PLANUNGSPHASE

2.1 Projektzeitplan und Meilensteine



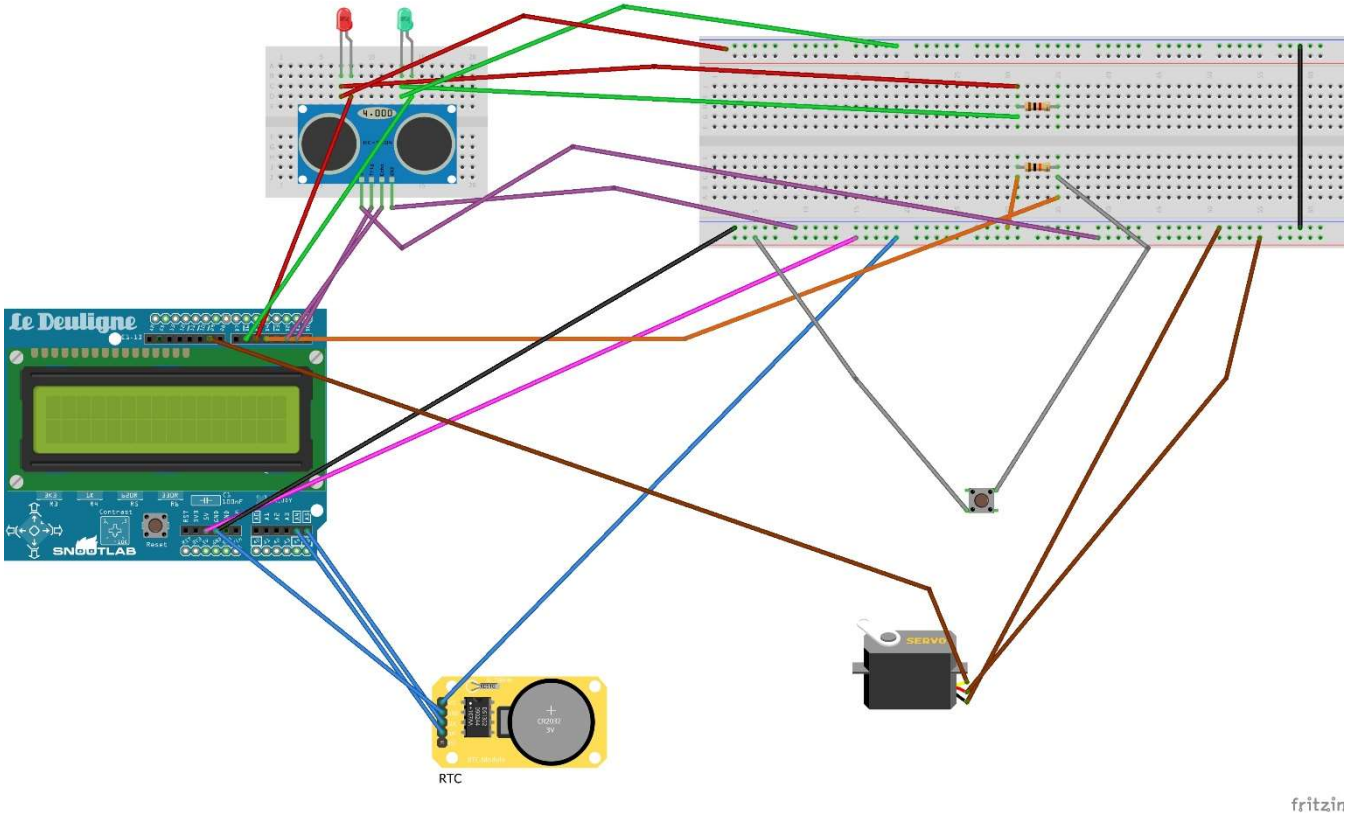
3. DURCHFÜRUNGSPHASE

3.1Bestandteile

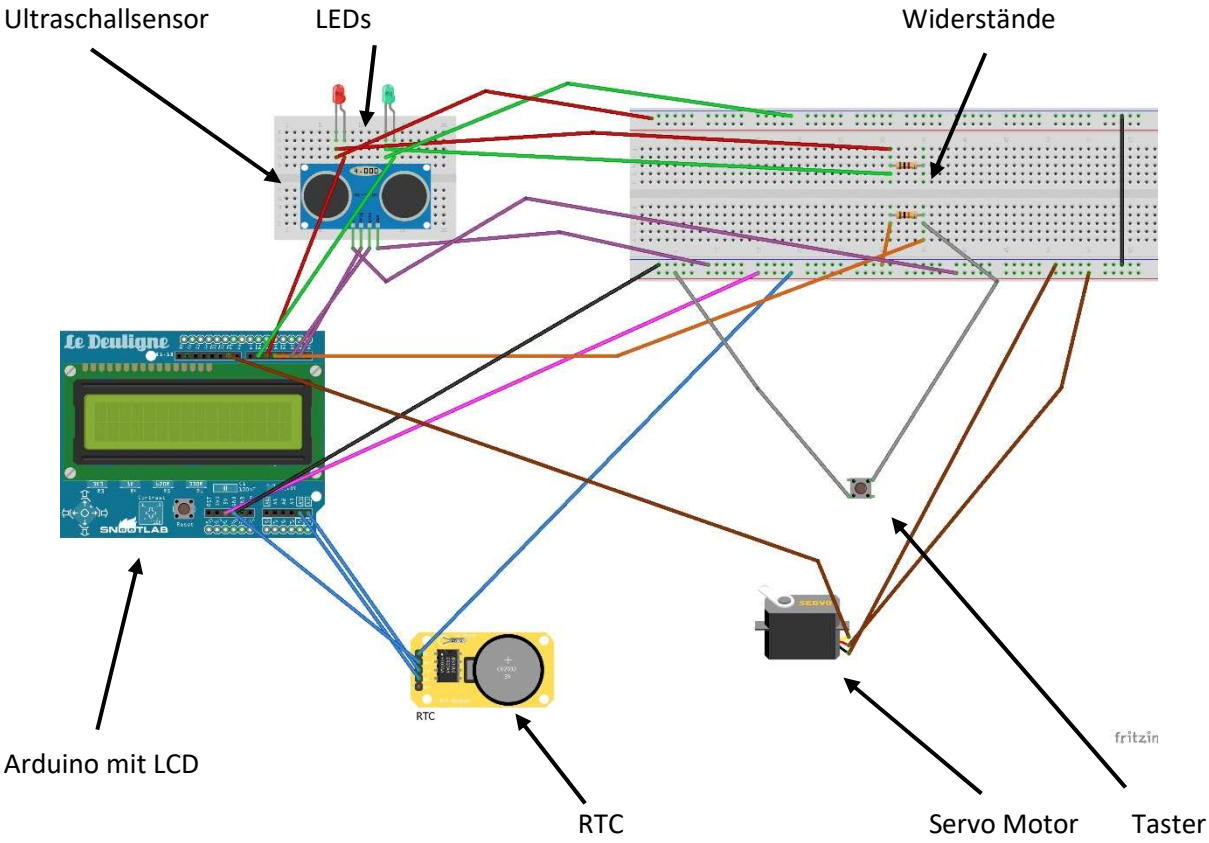
Für unser Projekt haben wir folgendes benötig:

- 1 Arduino Uno
- 1 LC Display
- 1 Ultraschallsensor (HC-SR04)
- 1 Taster
- 1 Servo Motor ()
- 1 Real Time Clock (HW-84)
- 1 großes Steckbrett
- 1 kleines Steckbrett
- 1 rote LED mit integriertem Widerstand
- 1 grüne LED mit integriertem Widerstand
- 1 10k Ohm Widerstand
- 1 1k Ohm Widerstand
- Jumper Kabel

3.2Schaltplan



So sieht die Schaltung von unserem Projekt aus.



Zum Aufbau:

Das LCD wird auf den Arduino gesteckt. Der Ultraschallsensor und die LED kommen auf die kleine Steckplatte und die beiden Widerstände werden auf die große Steckplatte gesteckt.

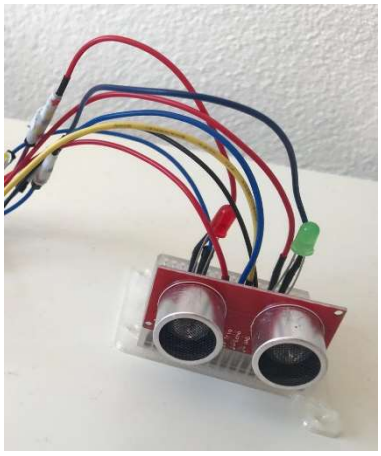
Ultraschallsensor:

VCC → Board 5V

Trig → Pin 1 am Arduino

Echo → Pin 2 am Arduino

GND → Board GND



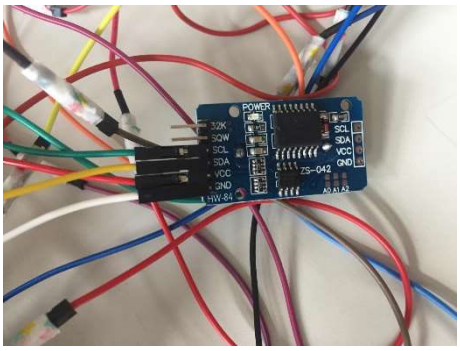
Real Time Clock

GND → GND Arduino

VCC → Board 5V

SDA → A4 am Arduino

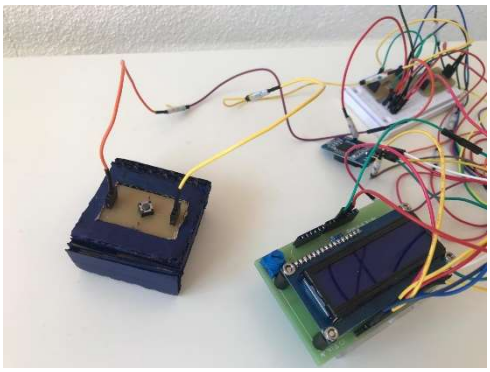
SCL → A5 am Arduino



Arduino

GND→ Board (groß)

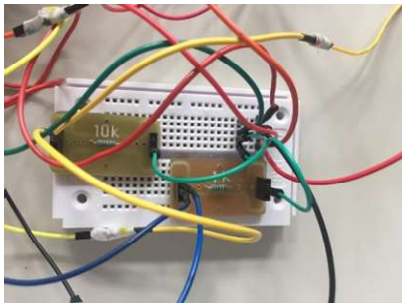
5V → Board (groß)



Taster

Board → 5 V

Widerstand 10k Ohm → Board → GND



Widerstand

10k Ohm → Pin 4 am Arduino

Lampen

Rot:

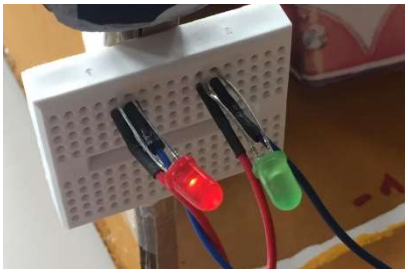
Beinchen mit Widerstand → Pin 5 am Arduino

Kurzes Beinchen → GND (Board)

Grün:

Beinchen mit Widerstand → Pin 6 am Arduino

Kurzes Beinchen → GND (Board)

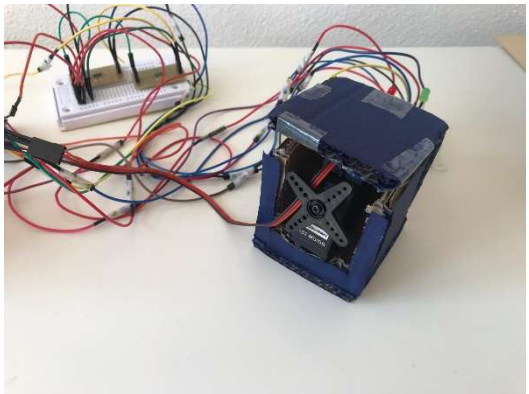


Servo Motor (Schranke)

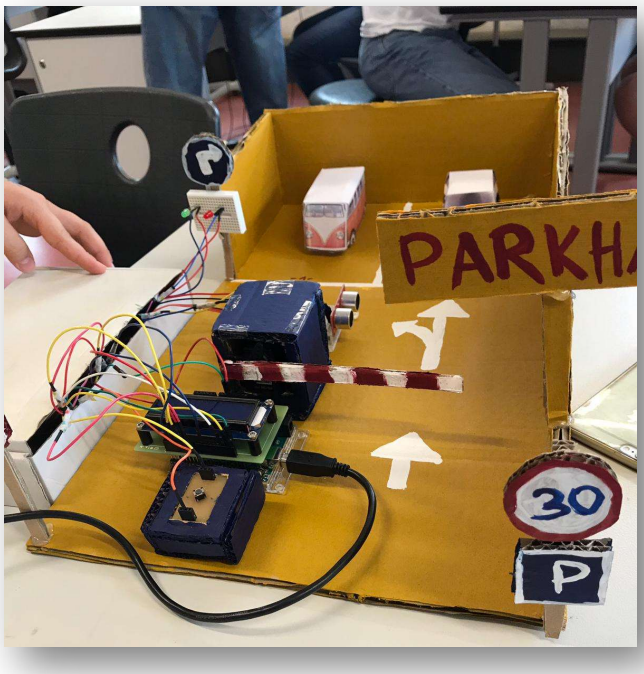
VCC → 5V am Board

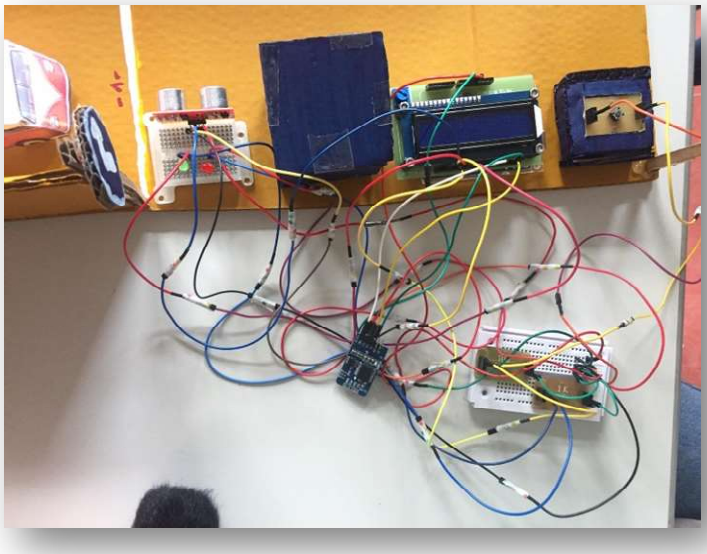
GND → GND am Board

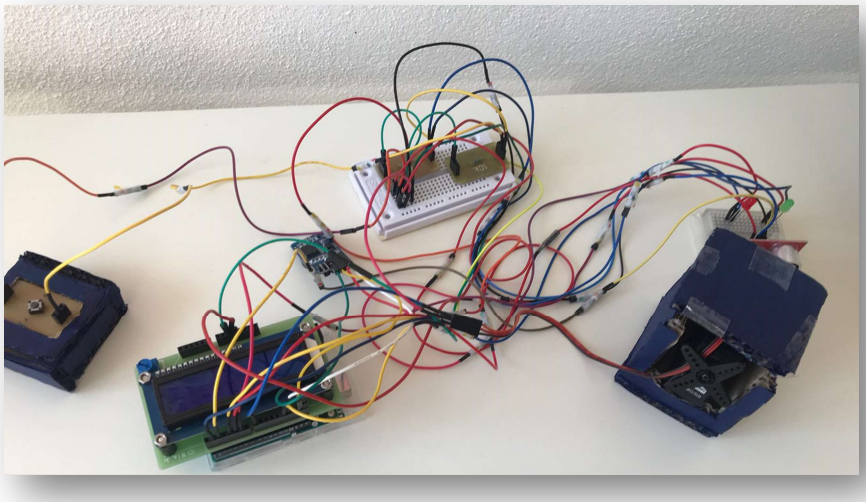
J1 → Pin 9 am Arduino



Aufbau:







3.3Das Protokoll

21.05.2019, Tübingen Kepler Gymnasium

Anwesende: Paola und Simon

- Entwicklung der Idee (Simon und Paola)
- Erste Skizzen zum Aufbau (Simon und Paola)

23.05.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Erste Skizzen zum Aufbau (Paola und Simon)
- Genauere Planung (Alisa, Carina und Simon)

28.05.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Beschaffung des Materials
- Bauteile verbauen (Arduino, LCD, Steckbretter und LEDs) (alle)

04.06.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Bauteile verbauen (Ultraschallsensor und Taster) (Simon und Paola)
- Erste Schritte: Beginn der Programmierung (Alisa und Carina)

06.06.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Bauteile verbauen (RTC und Servo Motor) (Simon und Carina)
- Verlängerung der Abstände zwischen den Bauteilen (Alisa und Paola)
- Bau des Parkhauses + anmalen (Alisa)

25.06.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- PROBLEM: Servo-Motor kaputt → Beschaffung eines neuen Servo Motors
- Problem behoben
- Programmierung (Alisa und Carina)

04.07.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Beginn der Dokumentation (Carina)
- Erstellung des Schaltplans (Carina und Simon)
- Bauteile in das Parkhaus einsetzten (Alisa und Paola)

09.07.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Erstellung des Schaltplans (Carina und Simon)
- Erstellung der Dokumentation (Carina)
- Bauteile in das Parkhaus einsetzten (Alisa und Paola)

11.07.2019, Tübingen Kepler Gymnasium

Anwesende: Alisa, Carina, Paola und Simon

- Einzelheiten (alle)
- Letzten Tests (alle)
- Video des Projekts (alle)
- Abgabe der Dokumentation und des Projekts

3.4Das Programm

```
#include <Servo.h>
#include <DS3231.h>
#include <LiquidCrystal.h>

Servo myservo;

LiquidCrystal lcd(7, 8, 10, 11, 12, 13);

DS3231  rtc(SDA, SCL);

Time  trein;
Time  traus;
Time  tParkhaus;

int Sekundenrein = 0;
int Sekundenraus = 0;
int Parkzeit = 0;

int LEDgruen = 6;
int LEDrot = 5;
int taster = 4;
int tasterstatus = 0;
int voll = 0;
int rausfahren = 0;
int oeffnungswinkel = 0;

int trigger = 1;
int echo = 2;
long dauer = 0;
long entfernung = 0;
```

```
void setup()
{
  Serial.begin(9600);

  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  pinMode(LEDgruen, OUTPUT);
  pinMode(LEDrot, OUTPUT);
  pinMode(taster, INPUT);

  myservo.attach(9);
  myservo.write(0);


  lcd.begin(16, 2);
  lcd.setCursor(11, 1);
  lcd.println("frei ");

  rtc.begin();

  rtc.setTime(9, 55, 00);
  rtc.setDate(05, 07, 2019);

}



void loop()
{
  lcd.setCursor(0, 0);
  lcd.print(rtc.getDateStr());
  delay(10);
  lcd.setCursor(0, 1);
  lcd.print(rtc.getTimeStr());
  delay(10);

  if (voll == 0) digitalWrite(LEDrot, HIGH);


  tasterstatus = digitalRead(taster);


  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer / 2) * 0.03432;
```

```
if ((tasterstatus == 1) && (voll == 0))
{
  trein = rtc.getTime();
  Serial.println();
  Serial.print("Einfahrzeit: ");
  Serial.print(trein.hour, DEC);
  Serial.print(":");
  if (trein.min < 10) Serial.print('0');
  Serial.print(trein.min, DEC);
  Serial.print(":");
  if (trein.sec < 10) Serial.print('0');
  Serial.print(trein.sec, DEC);
  Serial.println();

  digitalWrite(LEDrot, LOW);
  digitalWrite(LEDgruen, HIGH);
  oeffnungswinkel = 90;
  myservo.write(oeffnungswinkel);
  Serial.println("Schranke offen - bitte einfahren");

  delay(5000);
  voll = 1;
  oeffnungswinkel = -90;
   myservo.write(oeffnungswinkel);
  Serial.println("Schranke wieder geschlossen");
  delay(10);
  digitalWrite(LEDrot, HIGH);
  digitalWrite(LEDgruen, LOW);
}


if ((tasterstatus == 1) && (voll == 1))
{
  Serial.println("Parkhaus besetzt. Keine Einfahrt möglich.");
  lcd.setCursor(11, 1);
  lcd.print("voll ");
  delay(10);
}

if ((entfernung > 0) && (entfernung < 15) && (voll == 1))
{
  digitalWrite(LEDrot, LOW);
  digitalWrite(LEDgruen, HIGH);
  oeffnungswinkel = 90;
  myservo.write(oeffnungswinkel);
  Serial.println("Schranke offen - bitte ausfahren");

  delay(3000);

  traus = rtc.getTime();
  Serial.print("Ausfahrzeit: ");
  Serial.print(traus.hour, DEC);
  Serial.print(":");
  if (traus.min < 10) Serial.print('0');
  Serial.print(traus.min, DEC);
  Serial.print(":");
  if (traus.sec < 10) Serial.print('0');
  Serial.print(traus.sec, DEC);
  Serial.println();
```

```
    Sekundenrein = trein.hour * 3600 + trein.min * 60 + trein.sec;
    Sekundenraus = traus.hour * 3600 + traus.min * 60 + traus.sec;

    Parkzeit = Sekundenraus - Sekundenrein;

    tParkhaus.hour = Parkzeit / 3600;
    Parkzeit = Parkzeit % 3600;
    tParkhaus.min = Parkzeit / 60;
    Parkzeit = Parkzeit % 60;
    tParkhaus.sec = Parkzeit;


    Serial.print("Parkzeit: ");
    if (tParkhaus.hour < 10) Serial.print('0');
    Serial.print(tParkhaus.hour, DEC);
    Serial.print(":");
    if (tParkhaus.min < 10) Serial.print('0');
    Serial.print(tParkhaus.min, DEC);
    Serial.print(":");
    if (tParkhaus.sec < 10) Serial.print('0');
    Serial.print(tParkhaus.sec, DEC);
    Serial.println();

   lcd.setCursor(0, 1);
   lcd.print(" Total: ");
   lcd.print(tParkhaus.hour);
   lcd.print(":");
   if (tParkhaus.min < 10) lcd.print('0');
   lcd.print(tParkhaus.min);
   lcd.print(":");
   if (tParkhaus.sec < 10) lcd.print('0');
   lcd.print(tParkhaus.sec);
   delay(5000);
   lcd.setCursor(8, 1);
   lcd.print("   frei ");

   oeffnungswinkel = -90;
   myservo.write(oeffnungswinkel);
   Serial.print("Schranke wieder geschlossen - Parkhaus frei");
   digitalWrite(LEDrot, HIGH);
   digitalWrite(LEDgruen, LOW);
   voll = 0;
  }

}
```

```
#include <Servo.h>
#include <DS3231.h>
#include <LiquidCrystal.h>
```

Die Bibliotheken für den Servo- Motor, die Real-Time-Clock (RTC) und das LC Display (LCD) werden eingelesen.

```
Servo myservo;

LiquidCrystal lcd(7, 8, 10, 11, 12, 13);

DS3231  rtc(SDA, SCL);
```

In der ersten Zeile wird ein Servo- Objekt erstellt, um den Servo- Motor zu steuern.

Als zweites wird festgelegt, welche Pins des Arduino an welche Pins des Displays angeschlossen werden.

Der dritte Befehl gibt an, dass der Pin SDA der Real-Time-Clock mit dem Pin A4 des Arduino verbunden wird und der Pin SCL der Real-Time-Clock mit dem Pin A5 des Arduino. Die Real-Time-Clock wird an den Arduino angeschlossen.

```
Time  trein;
Time  traus;
Time  tParkhaus;
```

Hier werden drei zeitliche Variablen definiert. Die erste Variable steht dabei für die Eingangszeit und die zweite für die Ausfahrtszeit. Und die letzte Variable steht für die Zeit des Parkhausaufenthaltes.

```
int Sekundenrein = 0;
int Sekundenraus = 0;
int Parkzeit = 0;
```

Anschließend werden die angeschlossenen Pins werden deklariert. Die erste Variable, setzt den Wert für "Sekundenrein" auf 0 setzt und speichert diesen.

Die zweite Variable, setzt den Wert für "Sekundenraus" auf 0 setzt und speichert diesen ebenfalls. Und die dritte Variable, setzt den Wert für "Parkzeit" auf 0 setzt und speichert diesen auch.

```
int LEDgruen = 6;
int LEDrot = 5;
int taster = 4;
int tasterstatus = 0;
int voll = 0;
int rausfahren = 0;
int oeffnungswinkel = 0;
```

Auch hier werden die angeschlossenen Pins deklariert.

- Variable, die den Wert (ganze Zahl) für "LEDgruen" auf 6 setzt und speichert. Die grüne LED ist mit dem Pin 6 des Arduino verbunden.
- Variable, die den Wert (ganze Zahl) für "LEDrot" auf 5 setzt und speichert. Die rote LED ist mit dem Pin 5 des Arduino verbunden.
- Variable, die den Wert (ganze Zahl) für "taster" auf 4 setzt und speichert. Der Taster ist mit dem Pin 4 des Arduino verbunden.
- Variable, die den Wert (ganze Zahl) für "tasterstatus" auf 0 setzt und speichert
- Variable, die den Wert (ganze Zahl) für "voll" auf 0 setzt und speichert
- Variable, die den Wert (ganze Zahl) für "rausfahren" auf 0 setzt und speichert
- Variable, die den Wert (ganze Zahl) für "oeffnungswinkel" auf 0 setzt und speichert

---

```
int trigger = 1;
int echo = 2;
long dauer = 0;
long entfernung = 0;
```

Angeschlossene Pins werden deklariert:

- Variable, die den Wert (ganze Zahl) für "trigger" auf 1 setzt und speichert. Der Ultraschall Sensor wird an Pin 1 des Arduino angeschlossen. (Pin am Ultraschall Sensor)
- Variable, die den Wert (ganze Zahl) für "echo" auf 2 setzt und speichert. Wird an Pin 2 des Arduino angeschlossen (Pin am Ultraschall Sensor).
- Variable, die den Wert (ganze Zahl) für "dauer" auf 0 setzt und speichert
- Variable, die den Wert (ganze Zahl) für "entfernung" auf 0 setzt und speichert. Das Wort „entfernung“ ist jetzt die variable, unter der die berechnete Entfernung gespeichert wird. Info: Anstelle von „int“ steht hier vor den beiden Variablen „long“. Das hat den Vorteil, dass eine größere Zahl gespeichert werden kann.

```
void setup()
{
  Serial.begin(9600);

  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  pinMode(LEDgruen, OUTPUT);
  pinMode(LEDrot, OUTPUT);
  pinMode(taster, INPUT);
```

VOID SETUP:

- Die serielle Kommunikation wird gestartet. Das Parameter 9600 legt die Übertragungsgeschwindigkeit (Baud-Rate) fest
- Digitale Pins werden als Ausgang definiert: „trigger“, „LEDgruen“ (grüne LED) und „LEDrot“ (rote LED)
- Digitale Pins werden als Eingang definiert: „echo“ und „taster“

```
myservo.attach(9);
myservo.write(0);
```

In der ersten Zeile wird festgelegt, an welchen Pin der Servo Motor angeschlossen wird. In diesem Fall Pin 9.

Schreibt einen Wert auf das Servo und steuert die Welle entsprechend. Bei einem Standard-Servo wird dadurch der Winkel der Welle (in Grad) eingestellt und die Welle in diese Ausrichtung bewegt. In diesem Fall ist der Winkel 0 Grad.

```
  lcd.begin(16, 2);
  lcd.setCursor(11, 1);
  lcd.println("frei ");
```

- Das Display wird initialisiert und die Anzahl der Spalten (16) und Zeilen (2) des LC Displays wird eingerichtet
- Die Spalte (11), an welcher der Text ausgegeben werden soll wird definiert; die Zeile (1), in welcher der Text ausgegeben werden soll wird definiert (0 oder 1)
- Schreibt Text oder Variablenwerte auf das LC Display in die nächste Zeile; in diesem Fall: "frei"

```
  rtc.begin();

  rtc.setTime(9, 55, 00);
  rtc.setDate(05, 07, 2019);
}
```

Der interne RTC wird initialisiert.

Die Uhrzeit wird auf 9:55:00 gesetzt, im 24 Stunden Format.

Das Datum wird auf den 05.07.2019 gesetzt.

```
void loop()
{
  lcd.setCursor(0, 0);
  lcd.print(rtc.getDateStr());
  delay(10);
  lcd.setCursor(0, 1);
  lcd.print(rtc.getTimeStr());
  delay(10);
```

VOID LOOP:

- Die Spalte (0), an welcher das Datum ausgegeben werden soll wird definiert; die Zeile (0), in welcher der Text ausgegeben werden soll wird definiert
- Das Datum wird auf dem LC Display ausgegeben
- 10 Millisekunden warten
- Die Spalte (0), an welcher die Uhrzeit ausgegeben werden soll wird definiert; die Zeile (1), in welcher der Text ausgegeben werden soll wird
- Die Uhrzeit wird auf dem LC Display ausgegeben
- 10 Millisekunden warten

```
if (voll == 0) digitalWrite(LEDrot, HIGH);

tasterstatus = digitalRead(taster);

digitalWrite(trigger, LOW);
delay(5);
digitalWrite(trigger, HIGH);
delay(10);
digitalWrite(trigger, LOW);
dauer = pulseIn(echo, HIGH);
entfernung = (dauer / 2) * 0.03432;
```

- Abfrage: ist das Parkhaus nicht "voll"? Wenn ja, dann wird die rote LED ausgeschalten
- Taster abfragen
- Ultraschall Sensor abschalten
- 5 Millisekunden warten

- Ultraschall Sensor anschalten
- 10 Millisekunden warten
- Ultraschall Sensor abschalten
- Mit dem Befehl „pulsen“ zählt der Mikrokontroller die Zeit in Mikrosekunden, bis der Schall zum Ultraschallsensor zurückkehrt
- Nun berechnet man die Entfernung in Zentimetern. Man teilt zunächst die Zeit durch zwei (Weil man ja nur eine Strecke berechnen möchte und nicht die Strecke hin- und zurück). Den Wert multipliziert man mit der Schallgeschwindigkeit in der Einheit Zentimeter/Mikrosekunde und erhält dann den Wert in Zentimetern

---

```
if ((tasterstatus == 1) && (voll == 0))
{
  trein = rtc.getTime();
  Serial.println();
  Serial.print("Einfahrzeit: ");
  Serial.print(trein.hour, DEC);
  Serial.print(":");
  if (trein.min < 10) Serial.print('0');
  Serial.print(trein.min, DEC);
  Serial.print(":");
  if (trein.sec < 10) Serial.print('0');
  Serial.print(trein.sec, DEC);
  Serial.println();
```

- Abfrage: Taster ist gedrückt und das Parkhaus ist noch nicht "voll" (es sind noch Parkplätze vorhanden)
- Uhrzeit der Einfahrt wird auf dem seriellen Monitor ausgegeben
- Wert der Einfahrtszeit nach "Einfahrtszeit:" auf seriellen Monitor ausgegeben
- Ausgabe: Stunde der Einfahrtszeit in Dezimal

- Abfrage: Minuten der Eingangszeit: weniger als 10 Minuten, dann wird eine 0 auf seriellen Monitor ausgegeben
- Ausgabe: Minuten der Einfahrtszeit in Dezimal

- Abfrage: Sekunden der Eingangszeit: weniger als 10 Sekunden, dann wird eine 0 auf seriellen Monitor ausgegeben
- Ausgabe: Sekunden der Einfahrtszeit in Dezimal

```
digitalWrite(LEDrot, LOW);
digitalWrite(LEDgruen, HIGH);
oeffnungswinkel = 90;
myservo.write(oeffnungswinkel);
Serial.println("Schranke offen - bitte einfahren");
```

- Rote LED wird ausgeschalten
- Grüne LED wird angeschaltet
- Der Öffnungswinkel der Schranke wird definiert (90 Grad)
- Schranke öffnet sich (um den Öffnungswinkel)
- Auf dem seriellen Monitor wird "Schranke offen - bitte einfahren" ausgegeben

---

```
  delay(5000);
  voll = 1;
  oeffnungswinkel = -90;
  myservo.write(oeffnungswinkel);
  Serial.println("Schranke wieder geschlossen");
  delay(10);
  digitalWrite(LEDrot, HIGH);
  digitalWrite(LEDgruen, LOW);
}
```

- 5 Sekunden warten
- Das Parkhaus ist voll (es sind keine Parkplätze mehr vorhanden)
- Der Öffnungswinkel wird neu definiert (-90 Grad)
- Die Schranke wird wieder geschlossen
- "Schranke wieder geschlossen" wird auf dem seriellen Monitor ausgegeben
- 10 Millisekunden warten
- Rote LED wird ausgeschalten
- Grüne LED wird angeschaltet

```
if ((tasterstatus == 1) && (voll == 1))
{
  Serial.println("Parkhaus voll. Keine Einfahrt möglich.");
  lcd.setCursor(11, 1);
  lcd.print("voll ");
  delay(10);
}
```

- Abfrage: Taster gedrückt und Parkhaus voll
- Ausgabe auf dem seriellen Monitor "Parkhaus voll. Keine Einfahrt möglich."
- Die Spalte (11), an welcher der Text ausgegeben werden soll wird definiert; die Zeile (1), in welcher der Text ausgegeben werden soll wird definiert
- Auf dem LC Display wird "voll" ausgegeben
- 10 Millisekunden warten

```
if ((entfernung > 0) && (entfernung < 15) && (voll == 1))
{
  digitalWrite(LEDrot, LOW);
  digitalWrite(LEDgruen, HIGH);
  oeffnungswinkel = 90;
  myservo.write(oeffnungswinkel);
  Serial.println("Schranke offen - bitte ausfahren");

  delay(3000);
```

- Auto will ausfahren
- Abfrage: Die Entfernung liegt zwischen 0 und 15; das Parkhaus ist voll
- Rote LED anschalten
- Grüne LED ausschalten
- Öffnungswinkel wird auf 90 Grad definiert
- Schranke wird um den Öffnungswinkel geöffnet
- Ausgabe auf seriellen Monitor: "Schranke offen - bitte ausfahren"
- 3 Sekunden warten

```
traus = rtc.getTime();
Serial.print("Ausfahrzeit: ");
Serial.print(traus.hour, DEC);
Serial.print(":");
if (traus.min < 10) Serial.print('0');
Serial.print(traus.min, DEC);
Serial.print(":");
if (traus.sec < 10) Serial.print('0');
Serial.print(traus.sec, DEC);
Serial.println();
```

- Uhrzeit der Ausfahrt
- "Ausfahrzeit: " wird auf dem seriellen Monitor ausgegeben
- Ausgabe: Stunde der Ausfahrtszeit in Dezimal
- Abfrage: Minuten der Ausgangszeit: weniger als 10 Minuten, dann wird eine 0 auf dem seriellen Monitor ausgegeben
- Ausgabe: Minuten der Ausfahrtszeit in Dezimal
- Abfrage: Sekunden der Ausgangszeit: weniger als 10 Sekunden, dann wird eine 0 auf dem seriellen Monitor ausgegeben
- Ausgabe: Sekunden der Ausfahrtszeit in Dezimal
- Nächste Zeile

```
Sekundenrein = trein.hour * 3600 + trein.min * 60 + trein.sec;
Sekundenraus = traus.hour * 3600 + traus.min * 60 + traus.sec;

Parkzeit = Sekundenraus - Sekundenrein;
```

- Umrechnung der Eingangszeit in Sekunden (Stunden und Minuten) + Sekunden
- Umrechnung der Ausgangszeit in Sekunden (Stunden und Minuten) + Sekunden
- Formel für Zeit des Parkhausaufenthaltes

```
tParkhaus.hour = Parkzeit / 3600;
Parkzeit = Parkzeit % 3600;
tParkhaus.min = Parkzeit / 60;
Parkzeit = Parkzeit % 60;
tParkhaus.sec = Parkzeit;
```

Die Sekunden des Parkhausaufenthaltes werden durch 3600 geteilt somit in Stunden umgerechnet. Dabei bleibt ein Rest übrig. Dieser Rest wird durch 30 geteilt und somit erhält man die restlichen Minuten des Parkhausaufenthaltes. Es bleibt wieder ein Rest übrig, der die verbleibenden Sekunden angibt.

Die Zeit zwischen Ein- und Ausfahrt wird gestoppt und hier in Stunden, Minuten und Sekunden umgerechnet.

```
Serial.print("Parkzeit: ");
if (tParkhaus.hour < 10) Serial.print('0');
Serial.print(tParkhaus.hour, DEC);
Serial.print(":");
if (tParkhaus.min < 10) Serial.print('0');
Serial.print(tParkhaus.min, DEC);
Serial.print(":");
if (tParkhaus.sec < 10) Serial.print('0');
Serial.print(tParkhaus.sec, DEC);
Serial.println();
```

- Ausgabe serieller Monitor "Parkzeit: "
- Abfrage: weniger als 10 Stunden Parkhausaufenthalt, dann Ausgabe am seriellen Monitor 0
- Ausgabe der Stunden des Parkhausaufenthaltes in Dezimal
- Abfrage: weniger als 10 Minuten Parkhausaufenthalt, dann Ausgabe am seriellen Monitor 0
- Ausgabe der Minuten des Parkhausaufenthaltes in Dezimal
- Abfrage: weniger als 10 Sekunden Parkhausaufenthalt, dann Ausgabe am seriellen Monitor 0
- Ausgabe der Sekunden des Parkhausaufenthaltes in Dezimal
- nächste Zeile im seriellen Monitor

```
    lcd.setCursor(0, 1);
    lcd.print(" Total: ");
    lcd.print(tParkhaus.hour);
    lcd.print(":");
    if (tParkhaus.min < 10) lcd.print('0');
    lcd.print(tParkhaus.min);
    lcd.print(":");
    if (tParkhaus.sec < 10) lcd.print('0');
    lcd.print(tParkhaus.sec);
    delay(5000);
    lcd.setCursor(8, 1);
    lcd.print("   frei ");
```

- Die Spalte (0), an welcher der Text ausgegeben werden soll wird definiert; die Zeile (1), in welcher der Text ausgegeben werden soll wird definiert
- Ausgabe am LC Display " Total: "
- Ausgabe am LC Display: Aufenthalt Stunden
- Abfrage: weniger als 10 Minuten Parkhausaufenthalt, dann auf dem LC Display eine 0 ausgeben
- Ausgabe am LC Display: Aufenthalt Minuten
- Abfrage: weniger als 10 Sekunden Parkhausaufenthalt, dann auf dem LC Display eine 0 ausgeben
- Ausgabe am LC Display: Aufenthalt Sekunden
- 5 Sekunden warten
- Die Spalte (8), an welcher der Text ausgegeben werden soll wird definiert; die Zeile (1), in welcher der Text ausgegeben werden soll wird definiert
- Ausgabe am LC Display " frei "

```
    oeffnungswinkel = -90;
    myservo.write(oeffnungswinkel);
    Serial.print("Schranke wieder geschlossen - Parkhaus frei");
    digitalWrite(LEDrot, HIGH);
    digitalWrite(LEDgruen, LOW);
    voll = 0;
  }


}
```

- Öffnungswinkel wird neu definiert (-90 Grad)
- Schranke schließen, um den Öffnungswinkel -90 Grad
- Ausgabe am seriellen Monitor "Schranke wieder geschlossen - Parkhaus frei"
- Rote LED ausschalten
- Grüne LED anschalten
- Parkhaus ist nicht voll; es sind noch Plätze vorhanden

---

4. ABSCHLUSSPHASE

4.1 Versuch auf Erfolg untersuchen

4.1.1 Projektauftrag

Projektauftrag: Baue ein Parkhaus mit diversen Funktionen, die vom Arduino angetrieben werden.

→ **Der Projektauftrag wurde erfolgreich ausgeführt**

4.1.2 Projektziele

ZIELE	ERREICHT/ NICHT ERREICHT
➢ Das Parkhaus hat eine Ampel, mit rotem und grünem Licht	• Vorhanden
➢ Das Parkhaus besitzt eine Schranke, die sich automatisch öffnet und schließt, nach betätigen eines Tasters	• Vorhanden
➢ Die Zeit des Parkhausaufenthaltes soll gestoppt werden und auf dem LCD ausgegeben werden	• Funktioniert
➢ Neben dem Parkhaus befindet sich eine Anzeige mit Uhrzeit und Datum	• Vorhanden
➢ Der Zustand des Parkhauses soll auf dem LCD wiedergegeben werden	• Funktioniert

→ **Alle Ziele wurden erreicht**

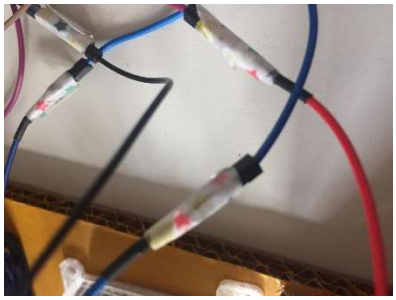
4.2 Ist-Soll-Vergleich

IST	SOLL	
LCD: die Uhrzeit, das Datum und frei + rote Lampe leuchtet	LCD: die Uhrzeit, das Datum und frei + rote Lampe leuchtet	✓
Taster wird gedrückt → die grüne Lampe geht an und Schranke öffnet sich	Taster wird gedrückt → die grüne Lampe geht an und Schranke öffnet sich	✓
Schranke schließt sich automatisch nach 5 Sekunden wieder → rote Lampe geht an	Schranke schließt sich automatisch nach 5 Sekunden wieder → rote Lampe geht an	✓
Zeit des Aufenthaltes soll gestoppt werden	Zeit des Aufenthaltes soll gestoppt werden	✓
Sind alle Parkplätze besetzt, so steht auf dem LCD „voll“	Sind alle Parkplätze besetzt, so steht auf dem LCD „voll“	✓
Ultraschallsensor: die Schranke geht automatisch wieder auf und das Lämpchen wird grün	Ultraschallsensor: die Schranke geht automatisch wieder auf und das Lämpchen wird grün	✓

Nach 3 Sekunden schließt sich automatisch die Schranke wieder	Nach 3 Sekunden schließt sich automatisch die Schranke wieder	✓
Auf dem LCD wird nun die Zeit des Aufenthaltes angegeben	Auf dem LCD wird nun die Zeit des Aufenthaltes angegeben	✓

→ Der Soll- Zustand stimmt mit dem Ist- Zustand überein

4.3Verbesserungsvorschläge



- Die Verbindungskabel mit Tape verbinden, damit sie nicht so leicht auseinandergehen können (siehe Bild)

4.4Probleme und ihre Lösungen

- Der Servo- Motor funktionierte am 25.06.2019 nicht mehr → Austausch des Servo Motors durch einen funktionierenden Servo Motor

- Da der Abstand zwischen zwei Bauteilen häufig ziemlich groß ist, verwendeten wir Verbindungskabel, um den Abstand zu vergrößern. PROBLEM: die Kabel sind oft auseinander gegangen → Verbindungen mit einem Tape fixieren

- Kleine Fehler im Programm → ließen sich leicht beheben

4.5Fazit

→ Erfolgreiches Projekt: Alles funktioniert, so wie es geplant war

4.6Quellen

- http://www.netzmafia.de/skripten/hardware/Arduino/Arduino_Programmierhandbuch.pdf
- https://www.arduino.cc/reference/de/language/functions/math/map/
- https://www.arduino-tutorial.de/lcd/
- https://www.arduino.cc/en/Reference/RTC
- https://www.arduino-tutorial.de/taster/

- **Verwendung:** Nachlesen von Funktionen, Schaltpläne und Programmen