Technische Dokumentation Projekt Huehnerklappe ("HenDroid")

1. ## Elektroinstallation allgemein

1.1. Komponenten

1. Spannungsversorgung
    - Netzteil 5V 5A
    - eingestellte Leerlaufspannung: 5.21V (lt. Wikipedia: V_usb=5V +/-5%; Pi wirft Under Voltage Errors)
    - Verbraucherberechnung

Verbraucher	**Typ. Stromaufnahme**	**Max. Stromaufnahme**
Raspberry Pi	1,0A	2,5A
Camera Module	0,25A	0,25A
PiFace Digital 2	???	???
Lichtrelais	0,03A	0,03A
Motor	1,7A	2,5A
Handy Akku	???	???
Summe	*2,98A+*	*5,28A+*

1. Schaltelemente
    - Relais zum Schalten der Raumbeleuchtung
        - Steuerspannung: 5VDC (direkt durch PiFace Open Collector Ausgänge schaltbar)
        - Schaltspannung: 230VAC
        - Schaltleistung: 30W
        - Spulenwiderstand: 167Ω
    - 2 Leuchttaster grün zur Bedienung vor Ort
        - Funktion:
            - Eingangssignal für Pi(Face) → Initiierung Öffnen/Schließen,
            - Ausgangssignal von Pi(Face) → Dauerleuchten zeigt Erreichen der Endlage an (offen/geschlossen); → Blinken zeigt Bewegung an (öffnend/schließend)
    - Leuchtschalter rot als Hauptschalter
        - Schaltspannung 230VAC
        - 2 Phasen → Schaltung von L und N (garantiert vollständige Spannungsfreiheit)
    - Reedkontakte als Endschalter für Klappe
        - billig
        - feuchtigkeitsunempfindlich

        - Berechnung Leitungswiderstand
            - Länge Zuleitung (einfach): ~6m
            - Querschnitt Zuleitung: 0,34mm² / 0,5mm²
            - spezifischer Widerstand Cu: 0,0175 Ω*mm²/m
            - Leitungswiderstand: 0,62 Ω / 0,42 Ω

1.2. Sicherungsauslegung

- Eingangssicherung Netzteil: 230V-5A-t
    - entspr. Herstellerdoku Netzteil
- Leitungsschutz 5V-Kreis (= Ausgangssicherung Netzteil): 5V-2A-ff
    - hält Nennstrom (2A) für >=1h
    - hält 1,5-fachen Nennstrom (3A) für ⇐30min
    - löst bei 2,75-fachem Nennstrom (5,5A) in ⇐150ms aus

Dauerverbraucher	**Typ. Stromaufnahme**	**Max. Stromaufnahme**
Raspberry Pi	1,0A	2,5A
Camera Module	0,25A	0,25A
PiFace Digital 2	???	???
Summe	*1,25A+*	*2,75A+*

- → hält Dauerstrom, sicher
- → löst bei max. Stromabgabe des Netzteils sicher aus
- → mgl. Verbesserung: einzelne Absicherung der einzelnen Stränge (löst Sicherung bei maximaler normaler Stromaufnahme von Pi und Motor bereits aus?)

1.3. Leitungsauslegung

1. Zuleitung Spannungsversorgung, Schaltleitung Raumbeleuchtung
    - 5G1
    - Eingangssicherung 5A-t
        - → löst bei 4-fachem Nennstrom (20A) nach ⇐3s aus
    - Stromführungsvermögen Leitung: >=10A
2. Verdrahtung Schaltkasten: 0,5mm²
    - Absicherung 5V-Kreis: 2A-ff
    - max. Stromführungsvermögen Leiter: >=5A
3. Zuleitung Motor: 4x0,34

    - Absicherung 5V-Kreis: 2A-ff
    - max. Stromführungsvermögen Leiter: ~6A

1.4. Schaltelemente

- Relais zum Schalten der Raumbeleuchtung
    - Ansteuerung über Outputs des PiFace (5V Open Collector)
        - maximaler Strom???
    - Spulenwiderstand: 125 Ω
    - → Steuerstrom: 40mA

1.5. Sensoren

- Reedsensoren als Endschalter für Klappe

2. ### Steuerung durch Raspberry Pi

2.1. Hauptkomponenten

1. Raspberry Pi 3B
2. Camera Module v1
    - Auflösung: 5MP
3. Erweiterungsplatine PiFace Digital 2
    - 2 Wechsler-Relais (Schaltspannung 20 V, Schaltstrom 5 A)
    - 4 Taster
    - 4 DI
    - 4 GPIO
    - 8 LED

2.2. Software-Architektur

1. UI-Client
    - dynamisch durch js
    - Schalten und Anzeigen des aktuellen Bewegungszustandes
    - Parametrierung des Timers
2. Server
    - nodejs
    - hendroid.zosel.ch (Remote Server)

    - localhost:3030 bzw. 192.168.43.66:3030 (Lokaler Server)
    - Ausliefern der dynamischen Website an Browser
    - Weiterleiten der Socket-Events zwischen Pi und UI-Client
3. Pi-Logik
    - Python 3.5
    - Starten der Logik, Initialisierung: >>python3 ~/hendroid/client-raspi/init.py
    - Schalten der HW-Outputs (outputs.py)
    - Handling des aktuellen Status + Statusübergänge (state_handler.py)
    - Handling des Timers zum automatischen Öffnen/Schließen zu einstellbarer Uhrzeit (timer_handler.py)
    - Kommunikation mit Servern per Websockets (client.py)
4. OS
    - Raspbian
    - Starten von Python-Logik und Node Server als systemd services:
        - Service Unit-Datei: /etc/systemd/system/hendroid-<client/server>d.service
        - Aktivieren der Services: >>systemctl enable hendroid-<client/server>d.service
        - Überwachen der Services: >>systemctl status hendroid-<client/server>d.service
    - Log output: >> journalctl -u service-name.service

3. ### Seilwinde

3.1. Hauptkomponenten

1. Gleichstrom-Bürstenmotor Motoraxx X Drive 5600 1/min
    - Last-Drehzahl: 4500 1/min
    - Maximalmoment: 12,5 Nmm
    - Betriebsspannung: 3 – 15 VDC
    - maximale Stromaufnahme: 1,7 A
    - Wellendurchmesser: 3,17 mm
2. Zahnradstufe
    - Übersetzung: 1:1,67
    - Modul: 0,5
    - Material: Polyacetat
3. Schneckenradstufe
    - Übersetzung: 1:60
    - Gang: 1

    - Material: Stahl/Messing
4. Seilwinde
    - effektiver Durchmesser: 10mm
    - Seillänge: ca. 40cm
    - Material: Kunststoff

3.2. Funktionsbeschreibung

- Belastungsannahmen:
    - Zugkraft am Seil: 30N
    - Zug-Geschwindigkeit am Seil: ca. 2,5 cm/s (entspr. 12s Wartezeit bei 30cm Weg)
- Berechnungen Winde
    - effektiver Durchmesser: 10 mm
    - → Lastmoment: 0,3 Nm
    - → Drehzahl: 0,8 1/s = 48 1/min
- Auslegung Getriebe
    - muss selbsthemmend sein, um ohne Antriebsmoment die Dauerlast (Klappengewicht) zu halten → Schneckenradstufe
        - Übersetzung 1:60 (aus Liefergründen: zweite Zahnradstufe war leichter in niedriger Übersetzung zu beschaffen)
        - → Lastmoment am Schneckenrad (bei Annahme Wirkungsgrad 0,5): 10 Nmm
        - → Drehzahl am Eingang: 2870 1/min
    - weitere Übersetzung nötig, da Motordrehzahl viel höher → weitere Zahnradstufe
        - benötigte Übersetzung: 1:1,57, vorliegende Übersetzung: 1:1,67
        - → Drehzahl am Eingang: 4780 1/min
        - → Lastmoment am Eingang: 6 Nmm
        - verwendeter Werkstoff: Kunstsoff → keine Schmierung benötigt

4. ### Kommunikation

4.1. Komponenten

1. Smartphone als mobiler Hotspot
    - htc ONE S
    - PIN: 0000
    - Passwort für Bildschirmsperre: 0000
    - Netzwerkname: hendroid

    - Netzwerkpasswort: s. ./pwds

4.2. Mögliche Kommunikationswege (Brainstorming)

- mobiles Internet mit Surfstick
    - Mobilfunkanbieter - Netzauslastung
        - Vodafone: alle Netze gut
        - Telekom: alle Netze gut
        - Telefonica (O2, Eplus): nur 2G, 3G (kein LTE)
    - Vorteile:
        - Kommunikation von überall aus
    - Nachteile:
        - begrenzte Bandbreite → langsam
        - begrenztes Datenvolumen → nur einfache Informationen austauschbar (Textfiles, Statusinfos, Kommandos; kein remote Desktop!)
- Remote Desktop via lokalem Netzwerk
    - WLAN-Router vor Ort benötigt (z.B. mobiler Hotspot Smartphone), Zugriff nur aus selbem Netzwerk möglich
    - mögliche Technologien:
        - VNC (desktop sharing): 'vncviewer' → connect to 'hendroid.local'
        - SSH und Erweiterungen (SFTP (browse, exchange, edit files), SCP (exchange files), SSHFS (mount Pi-files on local machine), rsync (synchronisation))
    - Vorteile:
        - große Bandbreite,
        - unbegrenztes Datenvolumen
    - Nachteile:
        - nur von vor Ort zugreifbar (wahrscheinlich nicht einmal aus dem Büro, da zu weit weg…)
- Remote Access via Internet
    - beide Teilnehmer benötigen Internetzugriff
    - mögliche Technologien:
        - Team Viewer
        - Remote.it
- lokales Netzwerk ohne Router
    - mit Android Smartphone (nicht gerootet) kein ad-hoc Netzwerk erstellbar
    - Android Smartphone kann nicht selbst Teilnehmer des eigenen mobilen Hotspot Netzwerkes sein

- Bluetooth → keine App für generische Kommunikation (außer File Trasfer) vorhanden
- Hendroid kann nicht dauerhaft lokales Netzwerk anbieten und gleichzeitig per Surfstick im mobilen Netz sein -→ Umschalten im Bedarfsfall zwischen Wifi-client mode und Hotspot mode nötig

4.3. Kommunikation Pi <→ Benutzer

4.3.1. Architektur

Pi-Logik:-------[python3] ^ |(python-intern) v Pi-Client:-[python3-webclient]----[Flask server] | ^ |(HTTP) |(HTTP) v | Pi-Proxy:---[Express server]-----[js-webclient] ^ ^ (SocketIO)| |(SocketIO) v v Remote server:----------[js-webclient] | [js-webclient]----------:Local Server ^ | ^ (SocketIO)| | |(SocketIO) v | v Web-UI:-----------------[js-webclient] | [js-webclient]----------------:Web-UI

- Gründe für komplizierte Architektur
  - js-Proxy: stellt stabilen js-SocketIO Websocket zur Kommunikation mit Remote Server (python-Variante socketIO-client-nexus unbrauchbar)
  - Local Server: stellt UI access auch ohne Internetverbindung zur Verfügung

4.3.2. Vor Ort via Hotspot

- Smartphone stellt dauerhaften Hotspot bereit:
  - Netzwerkname: hendroid
  - Passwort: s. ./pwds
- Jedes WLAN-fähige Nutzergerät kann sich vor Ort mit Hotspot verbinden
- auf Pi läuft ab Startup node server:
  - 'node ~/hendroid/server/index.js'
  - Website incl Websocket-Funktionalität aufrufbar per URL: 'hendroid.local/:3030' bzw. '192.168.43.66:3030'
  - Node-Server bietet exakt selbe Funktionalität an wie zosel.ch-Server
  - client hört auf Node-server events

4.3.3. Remote.it

1. Ablauf
   - remote.it im client öffnen
   - per E-Mail und Passwort anmelden
   - Gerätename auswähren
   - ssh Service Name auswählen
   - Strg-C Strg-V mit angegebenem Befehl → Zack fertig