Hands-On-Anleitung: Raspberry Pi mit Müdigkeits- & Gähn-Erkennung

1. Installation des Betriebssystems auf dem Raspberry Pi

• Raspberry Pi Imager von der offiziellen Webseite herunterladen: https://downloads.raspberrypi.org/imager/imager_latest.exe
• SD-Karte in den Computer einsetzen und mit dem Imager Raspberry Pi 4, empfohlenes Betriebssystem sowie die SD-Karte auswählen.
• Im Menü 'OS Customization Settings' Hostname, Benutzername, Passwort und WLAN-Zugangsdaten festlegen.
• SSH und VNC aktivieren.
• Einstellungen speichern und auf SD-Karte schreiben.
• SD-Karte in den Raspberry Pi einsetzen, Mini-HDMI, Tastatur, Maus und Stromversorgung anschließen.
• Mit 'sudo raspi-config' VNC aktivieren.
• Zugriff entweder per RealVNC Viewer oder per SSH-Verbindung über Windows PowerShell.

2. Installation der benötigten Bibliotheken

• Terminal öffnen (VNC oder SSH).
• System aktualisieren mit: sudo apt update && sudo apt upgrade
• Python-Version prüfen: python --version
• Virtuelle Umgebung anlegen: python3 -m venv yourName --system-site-packages
• Umgebung aktivieren: source youName/bin/activate
• Bibliotheken installieren: numpy, scipy, scikit-image, cmake, dlib, opencv-python, imutils, argparse
• Swap-Speicher auf 1024 MB erhöhen, Installation durchführen, danach wieder auf 512 MB zurücksetzen.

3. Programm ausführen

• Dateien von GitHub (https://github.com/Arijit1080/Drowsiness-and-Yawn-Detection-with-voice-alert-using-Dlib) herunterladen (bis auf drowsiness_yawn.py) und im Home-Verzeichnis speichern.
• Datei drowsiness_yawn.py zusätzlich speichern.
• Virtuelle Umgebung aktivieren und Programm starten: source drowsiness/bin/activate && python drowsiness_yawn.py

• Beenden mit Taste 'q'.

4. LEDs der Ampel anschließen und verwenden

• LEDs anschließen: Grün (GPIO 17), Gelb (GPIO 27), Rot (GPIO 22).
• Benötigte Dateien ins Home-Verzeichnis kopieren.
• Programm starten
• Beenden mit Taste 'q'.

5. Programm beim Systemstart automatisch ausführen

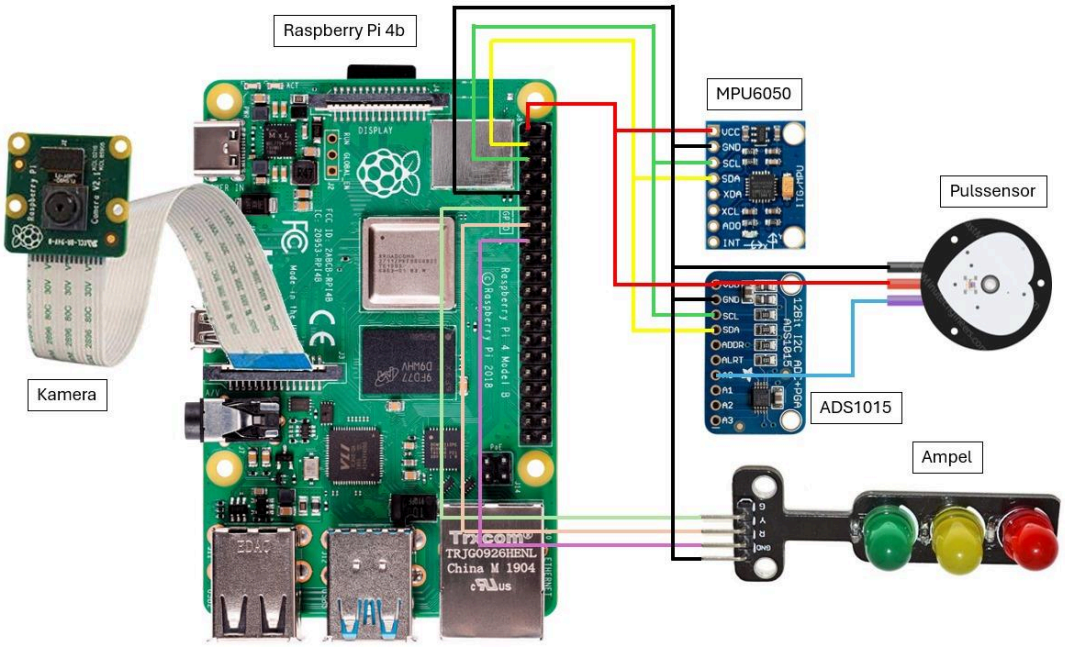
• Programm nach /home/USERNAME kopieren und ausführbar machen: chmod +x Programm.py
• systemd-Service-Datei erstellen: /etc/systemd/system/monitor.service
• Inhalt der Service-Datei anpassen (USERNAME ersetzen).
• Service aktivieren und starten: sudo systemctl enable monitor.service && sudo systemctl start monitor.service
• Neustart des Raspberry Pi durchführen.

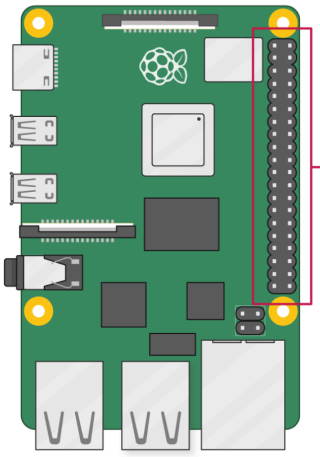
6. MPU6050-Sensor einrichten

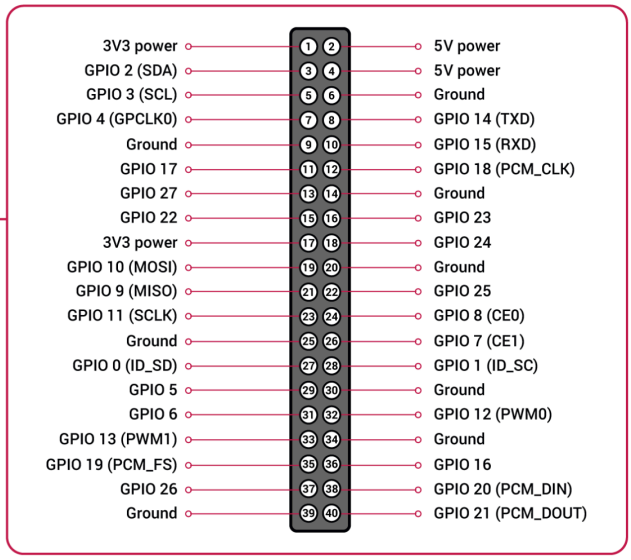
• Anschlüsse: SDA → GPIO 2, SCL → GPIO 3, VCC → 3,3 V, GND → GND.
• Bibliotheken installieren: python3-smbus, i2c-tools, mpu6050-raspberrypi.
• I^2C im Raspberry Pi aktivieren.
• Verbindung prüfen: i2cdetect -y 1 (Adresse 68 sichtbar).
• Testprogramm ausführen (Bewegungen werden erkannt).

7. Pulssensor einrichten

• Pulssensor mit ADS1015 verbinden: Rot → VCC, Schwarz → GND, Weiß → A0.
• ADS1015 mit Raspberry Pi verbinden: VCC → 3,3 V, SCL → GPIO 3, SDA → GPIO 2, GND → GND.
• Testprogramm starten und Herzfrequenz ausgeben lassen.
• MPU6050 und ADS1015 kombinieren und erweiterten Code ausführen.


Raspberry Pi 4b
MPU6050
Pulssensor
Kamera
ADS1015
Ampel




3V3 power
GPIO 2 (SDA)
GPIO 3 (SCL)
GPIO 4 (GPCLK0)
Ground
GPIO 17
GPIO 27
GPIO 22
3V3 power
GPIO 10 (MOSI)
GPIO 9 (MISO)
GPIO 11 (SCLK)
Ground
GPIO 0 (ID_SD)
GPIO 5
GPIO 6
GPIO 13 (PWM1)
GPIO 19 (PCM_FS)
GPIO 26
Ground
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
5V power
5V power
Ground
GPIO 14 (TXD)
GPIO 15 (RXD)
GPIO 18 (PCM_CLK)
Ground
GPIO 23
GPIO 24
Ground
GPIO 25
GPIO 8 (CE0)
GPIO 7 (CE1)
GPIO 1 (ID_SC)
Ground
GPIO 12 (PWM0)
Ground
GPIO 16
GPIO 20 (PCM_DIN)
GPIO 21 (PCM_DOUT)