Vital Desk

Introduction

Vital Desk is an interactive embedded system designed to check and prevent the health risks of modern remote work environments. In a context where sedentary time has increased by an average of 2 hours per day , this project serves as a companion that bridges environmental monitoring with behavioral intervention.

- What it does: The device monitors air quality (MQ8), climate (DHT11), and user proximity (HC-SR04). It uses a servo to break the user's static posture and displays real-time health metrics on an I2C LCD.
- Purpose: To combat the lack of an optimal microclimate in improvised workspaces and reduce cardiovascular risks associated with prolonged sitting and bad posture.
- The Concept: Moving away from passive measurement toward a proactive, artistic guardian that interacts physically with the user.
- Utility: It provides a stand-alone health assistant that functions independently of a desktop PC, ensuring a healthier and more focused workflow.

General Description

The project is based on three main components: data acquisition, local processing and intervenions based on alerts.

1. The sensors are collecting the data, such as the temperature, humidity, gas, proximity
2. The data is checked and based on the limits that we set at first, we can see wether the working environment is good or not.
3. If something is above or below limits, we alert through the LCD and/or servo sensor movement
4. Also, there’s an alert if the user is sitting way too close to the PC that they work on, or if the user hasn’t been standing for a longer period of time.

Hardware Design

Electrical Scheme (done in Wokwi)

The physical build is a frame where the sensors are integrated as facial features of a robot assistant.
Component List:

- Microcontroller: Arduino-UNO board.
- Climate Sensor: DHT11 for temperature and humidity tracking.
- Air Quality Sensor: MQ8 for gas and CO2 detection.
- Proximity Sensor: HC-SR04 Ultrasonic sensor to detect user presence/absence and how close the user is sitting to their computer.
- Actuator: SG90 Servomotor for physical intervention.
- Display: LCD 1602 with I2C interface for status updates.
- Connectivity: USB cable.

Proposed pinout

Arduino UNO:

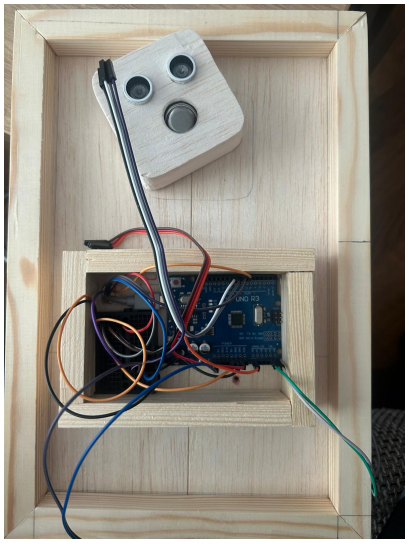
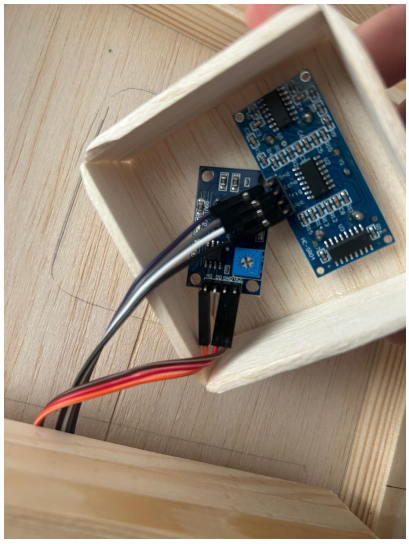
Component	Arduino Pins
DHT11	DATA → D7 (PD7), VCC → 5V, GND → GND
HC-SR04	TRIG → D9 (PB1), ECHO → D8 (PB0), VCC → 5V, GND → GND
MQ-8 Gas Sensor	AO → A0 (PC0 / ADC0), VCC → 5V, GND → GND
SG90 Servo	Signal → D6 (PD6), VCC → 5V, GND → GND
LCD 1602 (I2C)	SDA → A4 (PC4), SCL → A5 (PC5), VCC → 5V, GND → GND
Buttons (x4)	D2, D3, D4, D5 (PD2-PD5), INPUT_PULLUP, common pin → GND

Button mapping:

- **Button 1 (D2):** ARM / DISARM the system.
- **Button 2 (D3):** Monitoring screen (temperature, humidity, gas, distance).
- **Button 3 (D4):** Temperature min / max screen.
- **Button 4 (D5):** Humidity min / max screen.

Power Supply: The project is directly linked to the Laptop, using a USB cable, making the data collection easier. All sensors are powered through 5V from the Arduino UNO.

Servo power note. The SG90 draws short current spikes (hundreds of mA) when it starts moving. When the servo is powered from the Arduino 5V pin (especially over USB), these spikes cause brief voltage dips that can disturb the I2C bus and make both the LCD and the servo behave erratically. For a stable build, the servo should ideally be powered from a separate 5V source with a common ground to the Arduino, with a decoupling capacitor (470-1000 uF) close to the servo. In firmware, a short “quiet window” is inserted before the servo starts moving (no I2C traffic, no sensor reads) to reduce this interference.

Software Design

Development environment

- **PlatformIO** with the Arduino framework (board: `uno`, platform: `atmelavr`).
- Build flags: `-Os` (optimize for size) and `-Wall` (all warnings).
- Serial monitor speed: 9600 baud.

3rd-party libraries

The firmware uses a small set of well-known Arduino libraries (declared in `platformio.ini` under `lib_deps`):

- **DHT sensor library** (Adafruit) + **Adafruit Unified Sensor** - reading the DHT11 temperature and humidity.
- **LiquidCrystal_I2C** (marcoschwartz) - driving the 1602 LCD over I2C.
- **Servo** (arduino-libraries) - generating the control pulses for the SG90.

The GPIO access for the HC-SR04 and the buttons, as well as the MQ-8 ADC reads, are done **bare-metal** (direct register manipulation), in the PM lab style, instead of using `digitalRead`/ `analogRead`.

Alert thresholds:

Alert Trigger When? |

Temperature	T > 28 C	immediate
Humidity	H > 65 %	immediate
Gas	gas > 250 ppm	immediate
Too far	distance > 30 cm	held for 10 s

Too close	distance < 10 cm	held for 30 s

For the distance alerts, a generic **temporal timer** requires the condition to hold continuously for the configured duration before the alert fires; an ultrasonic timeout reading is treated as “very far” (nobody at the desk) so that walking away is detected reliably.

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Conclusions

Vital Desk meets its original goal: a stand-alone health assistant that monitors the work environment and intervenes physically and visually when something is off, without depending on a host PC for its logic. The most useful lessons from the project were on the debugging side rather than the feature side. The servo jitter was instructive: it showed clearly where the boundary between a software fix and a hardware fix lies. Spacing out the I2C traffic and the servo activity in firmware reduced the symptom, but the residual jitter is a current/power problem that software cannot fully solve.

If the project were continued, the natural next steps would be: a dedicated 5V supply for the servo to eliminate the remaining jitter, replacing the `_delay_ms`-based tick with a hardware timer so the 10 s / 30 s timers are exact.

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF