How to Build a Comfort Monitoring Sensor Station

Originalartikel

Backup

<html> <section id=„step1“ class=„step“ data-stepid=„SHHDK4NJEUKNUD6“ readability=„32“><h2 class=„step-title“>Step 1: Background – Thermal and Visual Comfort</h2> <noscript> </noscript> Thermal and visual comfort have become more and more important topics, especially in office and workplace environments, but also in the residential sector. The main challenge in this field is that the thermal perception of individuals often varies in a wide range. One person might feel hot in a certain thermal condition while another person feels cold in the same. That’s because the individual thermal perception is influenced by many factors, including the physical factors of air temperature, relative humidity, air velocity, and radiant temperature of surrounding surfaces. But also, clothing, metabolic activity, and an individual aspect of age, sex, body mass, and more, influence the thermal perception. While the individual factors remain an uncertainty in terms of heating and cooling controls, the physical factors can be determined precisely by sensor devices. Air temperature, relative humidity, air velocity, and globe temperature can be measured and used as a direct input to building controls. Further, in a more detailed approach, they can be used as input to calculate the so called PMV-index, where PMV stands for Predicted Mean Vote. It describes how people in average would be likely to rate their thermal sensation under given ambient room conditions. PMV can take on values from -3 (cold) to +3 (hot), with 0 being a neutral state. Why do we mention that PMV-thing here? Well, because in the field of personal comfort it is a commonly used index that can serve as a quality criterion for the thermal situation in a building. And with CoMoS, all ambient parameters required for PMV calculation can be measured. If you're interested, find out more about thermal comfort, the context of globe and mean radiant temperature, the PMV-index, and the implementing ASHRAE-standard at Wikipedia: Thermal Comfort ISO 7726 Ergonomics of the thermal environment ASHRAE NPO By the way: There are long existing, but also plenty of newly developed gadgets in the field of personalized environment to provide individual thermal and visual comfort. Small desktop fans are a well-known example. But also, footwarmers, heated and ventilated chairs, or office partitions for IR-radiative heating and cooling are being developed or even already available on market. All these technologies influence the local thermal condition, at a workplace for example, and they can be controlled automatically based on local sensor data, too, as illustrated in this step's pictures. More information about the gadgets of personalized environment and the ongoing research is available at Living Lab smart office space: Personalized Environment University of California, Berkeley ZEN report on personal heating an cooling devices [PDF] SBRC University of Wollongong </section><section id=„step2“ class=„step“ data-stepid=„S95WITBJGWJF8UE“

readability=„15“><h2 class=„step-title“>Step 2: System Scheme</h2> <noscript> </noscript> One of the main goals in the development process was to create a wireless, compact, and inexpensive sensor device to measure indoor environmental conditions of at least ten individual workplaces in a given open office space. Therefore, the station uses an ESP32-WROOM-32 with on-board WiFi connectivity and with a large variety of connector pins and supported bus types for all kinds of sensors. The sensor stations use a separate IoT-WiFi and send their data readings to a MariaDB database through a PHP script that runs on the database server. Optionally, an easy-to-use Grafana visual output can be installed as well. The scheme above shows the arrangement of all peripheral components as an overview on the system setup, but this instructable focuses on the sensor station itself. Of course, the PHP file and a description of the SQL connection is included later on, too, to provide all necessary information to build, connect, and use CoMoS. Note: at the end of this instructable you can find instructions on how to build an alternative stand-alone version of CoMoS with SD-card storage, internal WiFi access point, and a web app for mobile devices. </section><section id=„step3“ class=„step“ data-stepid=„SBI3OL1JGWJFPQ0“ readability=„5“><h2 class=„step-title“>Step 3: Supply List</h2> <noscript> </noscript> Electronics Sensors and controller, as shown in the picture: ESP32-WROOM-32 mikrocontroller (espressif.com) [A] Si7021 or GY21 temperature and humidity sensor (adafruit.com) [B] DS18B20+ temperature sensor (adafruit.com) [C] Rev C. air velocity sensor (moderndevice.com) [D] WS2812B 5050 status LED (adafruit.com) [E] BH1750 illuminance sensor (amazon.de) [F] More electric parts: 4,7k pull-up resistor (adafruit.com) 0,14 mm² (or similar) standard wire (adafruit.com) 2x Wago compact splicing connectors (wago.com) Micro USB cable (sparkfun.com) Case parts
(Find more detailed information on these parts and sizes in the next Step. If you have a 3D-printer available, you only need a table tennis ball. Skip the next Step and find all info and files for printing in Step 5.) Acrylic plate round 50×4 mm [1] Steel plate round 40×10 mm [2] Acrylic tube 50x5x140 mm [3] Acrylic plate round 40×5 mm [4] Acrylic tube 12x2x50 mm [5] Table tennis ball [6] Miscellaneous White paint spray Black matte paint spray Some tape A little insulation wool, a cotton pad, or anything similar Tools

Power drill 8 mm steal drill 6 mm wood/plastic drill 12 mm wood/plastic drill Thin hand saw Sandpaper Wire cutting pliers Wire stripper Soldering iron and tin Power-glue or hot glue gun Software and libraries
(The numbers indicate the library versions we used and tested the hardware with. Newer libraries should work as well, but we faced some issues occasionally while trying different / newer versions.) Arduino IDE (1.8.5) ESP32 Core library BH1750FVI library Adafruit_Si7021 library (1.0.1) Adafruit_NeoPixel library (1.1.6) DallasTemperature library (3.7.9) OneWire library (2.3.3) </section><section id=„step4“ class=„step“ data-stepid=„SYRZ8GJJGWJG56B“ readability=„38“><h2 class=„step-title“>Step 4: Case Design and Construction – Option 1</h2> <noscript> <img alt=„Picture of Case Design and Construction – Option 1“

src=„https://cdn.instructables.com/FZ1/J07C/JGWJGBY5/FZ1J07CJGWJGBY5.LARGE.jpg“ width=„2052“ height=„2052“/> </noscript> CoMoS' design features a slim, vertical case with most of the sensors mounted in the top area, with only the temperature and humidity sensor mounted near the bottom. The sensor positions and arrangements follow specific requirements of the measured variables: The Si7021 temperature and humidity sensor is mounted outside the case, near its bottom, to allow free air circulation around the sensor and to minimize the influence of waste heat evolved by the microcontroller inside the case. The BH1750 illuminance sensor is mounted on the flat top of the case, to measure the illumination on a horizontal surface as required by common standards on workplace illumination. The Rev. C wind sensor is also mounted in the top of the case, with its electronics hidden inside the case, but its tines, which carry the actual thermal anemometer and temperature sensor, exposed to the air around the top. The DS18B20 temperature sensor is mounted on the very top of the station, inside a black painted table tennis ball. The position on top is necessary to minimize the view factors and therefore the radiative influence of the sensor station itself to the globe temperature measurement. Additional resources about the mean radiant temperature and the use of black table tennis balls as globe temperature sensors are: Wang, Shang & Li, Yuguo. (2015). Suitability of Acrylic and Copper Globe Thermometers for Diurnal Outdoor Settings. Building and Environment. 89. 10.1016/j.buildenv.2015.03.002 . de Dear, Richard. (1987). Ping-pong globe thermometers for mean radiant temperature. H & Eng.,. 60. 10-12.
 The case is designed simple, to keep the manufacturing time and effort as low as possible. It can easily be built from standard parts and components with just a few simple tools and skills. Or, for those lucky enough to have a 3D-printer at their service, all case parts can be 3D-printed as well. For printing the case, the rest of this Step can be skipped and all required files and instructions can be found in the next Step. For the construction from standard parts, fitting dimensions are chosen for most of them: The main body is an acrylic (PMMA) pipe of 50 mm outer diameter, 5 mm wall thickness, and a height of 140 mm. The bottom plate, which serves as a light conductor for the status LED, is an acrylic round plate of 50 mm diameter and a thickness of 4 mm. A steel round with a diameter of 40 mm and a thickness of 10 mm is installed as a weight on top of the bottom plate and fit inside the lower end of the main body tube to prevent the station from toppling over and to hold the bottom plate in place. The top plate fits inside the main body tube as well. It’s made of PMMA and has a diameter of 40 mm and a thickness of 5 mm. Finally, the top riser tube is PMMA, too, with an outer diameter of 10 mm, a wall thickness of 2 mm, and a length of 50 mm. The manufacturing and assembling process is simple, beginning with some holes to drill. The steel round needs an 8 mm continuous hole, to fit the LED and cables. The main body tube needs some 6 mm holes, as cable feed-through for the USB and sensor cables, and as ventilation holes. The number and positions of holes can be varied up to your preference. The developers’ choice is six holes on the backside, close to top and bottom, and two on the front side, one top, one bottom again, as a reference. The top plate is the most tricky part. It needs a centered, straight and continuous 12 mm whole to fit the top riser tube, another off centered 6 mm hole to fit the illuminance sensor cable, and a thin slit of approximately 1,5 mm width and 18 mm length to fit the wind sensor. See the pictures for reference. And finally, the table tennis ball needs a 6 mm whole, too, to fit the globe temperature sensor and cable. In the next step, all PMMA parts, except the bottom plate,

should be spray painted, the reference is white. The table tennis ball must be painted in matte black to establish its estimated thermal and optical attributes. The steel round is glued centered and flat to the bottom plate. The top riser tube is glued into the 12 mm hole of the top plate. The table tennis ball is glued on the top end of the riser, with its hole matching the inner opening of the riser tube, so the temperature sensor and cable can be inserted to the ball afterwards through the riser tube. With this step done, all parts of the case are ready to be assembled by putting them together. If some fit too tight, sand them down a bit, if too loose, add a thin layer of tape. </section><section id=„step5“ class=„step“ data-stepid=„S4WVC60JKPWLUHH“ readability=„16“><h2 class=„step-title“>Step 5: Case Design and Construction – Option 2</h2> <noscript> </noscript> While Option 1 of building CoMoS' case is still a fast and simple one, letting a 3D-printer do the job might be even easier. Also for this option, the case is divided into three parts, top, case body, and bottom part, to allow easy wiring and assembly as described in the next Step. The files and further info on printer settings are provided at Thingiverse: CoMoS files on Thingiverse Following the instructions to use white filament for the top and case body parts is highly recommended. This prevents the case from heating up too quickly in sunlight and avoids false measurements. Transparent filament should be used for the bottom part to allow LED indicator illumination. Another variation from Option 1 is that the metal round is missing. To prevent CoMoS from toppling over, any kind of weight like bearing balls or a bunch of metal washers should be placed in/on the transparent bottom part. It is designed with an edge around to fit and hold some weight. Alternatively, CoMoS can be taped to its place of installation by using double-sided tape. Note: The Thingiverse folder includes files for a micro SD card reader case which can be mounted to the CoMoS case. This case is optional and part of the stand-alone version described in the last step of this instructable. </section><section id=„step6“ class=„step“ data-stepid=„ST3K2XHJGWJGCP8“ readability=„24“><h2 class=„step-title“>Step 6: Wiring and Assembly</h2> <noscript> <img alt=„Picture of Wiring and Assembly“

src=„https://cdn.instructables.com/FEF/4XS0/JP8Z7BBT/FEF4XS0JP8Z7BBT.LARGE.jpg“ width=„1600“ height=„1200“/> </noscript> The ESP, sensors, LED, and USB cable are soldered and connected according to the schematic circuit shown in the pictures of this step. The PIN-assignment matching the example code described later is: 14 - Reset bridge (EN) - [grey] 17 - WS2811 (LED) - [green] 18 - pullup resistor for DS18B20+ 19 - DS18B20+ (One Wire) - [purple] 21 - BH1750 & SI7021 (SDA) - [blue] 22 - BH1750 & SI7021 (SCL) - [yellow] 25 - BH1750 (V-in) - [brown] 26 - SI7021 (V-in) - [brown] 27 - DS18B20+ (V-in) - [brown] 34 - Wind sensor (TMP) - [cyan] 35 - Wind sensor (RV) - [orange] VIN - USB cable (+5V) - [red] GND - USB cable (GND) - [black] The Si7021, BH1750, and DS18B20+ sensors are powered through an IO-pin of the ESP32. This is possible because their max current draft is below the ESP's max current supply per pin, and necessary to be able to reset the sensors by cutting off their power supply in case of sensor communication errors. See the ESP code and comments for more information. The Si7021 and BH1750 sensors, same as the USB cable, should be soldered with the cables already put through the dedicated case holes to allow assembly in the next step. WAGO compact splicing connectors are used to connect devices to the power supply by the USB cable. All are powered at 5 V DC by USB, which works with the logic level of the ESP32 at 3,3 V. Optionally, the data pins of the micro USB cable can be reconnected to the micro USB plug and connected to the ESP's micro USB socket, as power input and data connection to transfer code to the ESP32 while the case is closed. Else, if connected as shown in the scheme, another intact micro USB cable is needed to initially transfer code to the ESP before assembling the case. The Si7021 temperature sensor is glued to the back side of the case, close to the bottom. It's very important to attach this sensor close to the bottom, to avoid false temperature readings caused by heat evolved within the case. See Epilogue step for more Information about this issue. The BH1750 illuminance sensor is glued to the top plate, and the wind sensor is inserted and fit mounted to the slit on the opposite side. If it fits too lose, a little bit of tape around the center part of the sensor helps to keep it in position. The DS18B20 temperature sensor is inserted through the top riser into the table tennis ball, with a final position in the center of the ball. The inside of the top riser is filled with isolation wool and the lower opening is sealed with tape or hot glue, to prevent conductive or convective heat transfer to the globe. The LED is attached into the steel round hole facing down to illuminate the bottom plate. All wires, the splicing connectors, and the ESP32 go inside the main case and all case part are put together in final assembly. </section><section id=„step7“ class=„step“ data-stepid=„SILU9CMJGZMFBZ9“ readability=„26“><h2 class=„step-title“>Step 7: Software – ESP, PHP, and MariaDB Configuration</h2> <noscript> </noscript> The ESP32 micro controller can be programmed by using the Arduino IDE and the ESP32 Core library provided by Espressif. There are plenty of tutorials available

online on how to set up the IDE for ESP32 compatibility, for example here. Once set up, the attached code is transferred to the ESP32. It is commented throughout for easy understanding, but some key features are: It has a „user configuration“ section at the beginning, in which individual variables must be set up, such as WiFi ID and password, database server IP, and desired data readings and send period. It also includes a „zero wind adjustment“ variable that can be used to adjust zero wind speed readings to 0 in case of a non-stable power supply. The code includes average calibration factors determined by the authors from calibration of ten existing sensor stations. See Epilogue step for more information and possible individual adjustment. Various error handling is included at several sections of the code. Especially an effective detection and handling of bus communication errors that occur often on ESP32 controllers. Again, see Epilogue step for more information. It has an LED color output to show the current state of the sensor station and any errors. See the Results step for more information. The attached PHP file has to be installed and accessible in the root folder of the database server, at serverIP/sensor.php. The PHP file name and content of data handling must match the call function code of the ESP and, on the other side, match the database table setup, to allow storage of data readings. The example codes attached are matched, but in case you change some variables, they have to be changed throughout the system. The PHP file includes an adjusting section at the beginning, in which individual adjustments are made according to the environment of the system, especially database username and password, and the database name. A MariaDB or SQL database is set up on the same server, according to the table setup used in the sensor station code and the PHP script. In the example code, the MariaDB database name is „sensorstation“ with a table named „data“, which contains 13 columns for UTCDate, ID, UID, Temp, Hum, Globe, Vel, VelMin, VelMax, MRT, Illum, IllumMin, and IllumMax. A Grafana analytics and monitoring platform can be installed additionally on the server as an option for direct database visualization. This is not a key feature of this development, so it's not further described in this instructable. <aside class=„downloads“><h4 class=„sr-only“>Attachments</h4> </aside> </section><section id=„step8“ class=„step“ data-stepid=„S32KHF5JGZMFC0I“ readability=„9“><h2 class=„step-title“>Step 8: Results – Data Reading and Verification</h2> <noscript> </noscript> With all wiring, assembly, programming, and environmental setup done, the sensor station sends data readings periodically to the database. While powered, several operation states are indicated through the bottom LED color: During boot, the LED lights in yellow color to indicate the pending connection to WiFi. When and while connected, the indicator is blue. The sensor station runs sensor readings and sends it to the server periodically. Each successful transfer is indicated by a green light impulse of 600 ms. In case of errors, the indicator will color red, purple, or yellowish, according to the error type. After a certain time or number of errors, the sensor station resets all sensors and reboots automatically, again indicated by a yellow light on boot. See the ESP32 code and comments for more information about the indicator colors. With this final step done, the sensor station runs and operates continuously. To date, a network of 10 sensor stations is installed and running in the beforehand mentioned Living Lab smart office space. </section><section id=„step9“ class=„step“ data-stepid=„SOCJE7EJMGL10JT“ readability=„25“><h2 class=„step-title“>Step 9: Alternative: Stand-alone Version</h2> <noscript> <img alt=„Picture of Alternative: Stand-alone Version“ src=„https://cdn.instructables.com/FMJ/WDJO/JP8Z90XT/FMJWDJOJP8Z90XT.LARGE.jpg“ width=„2000“

height=„1222“/> </noscript> The development of CoMoS continues and the first result of this ongoing process is a stand-alone version. That version of CoMoS doesn't need a database server and WiFi network to monitor and record environmental data. The new key features are: Data readings are stored on internal micro SD-card, in Excel-friendly CSV format. Integrated WiFi access point for access to CoMoS by any mobile device. Web-based app (internal web server on ESP32, no internet connection required) for live data, settings, and storage access with direct file download from the SD card, as shown in the picture and screenshots attached to this step. This replaces the WiFi and database connection while all other features including calibration and all design and construction remain untouched from the original version. Still, the stand-alone CoMoS requires experience and further knowledge of how to access the internal file management system „SPIFFS“ of the ESP32, and a little awareness of HTML, CSS, and Javascript to understand how the web-app works. It also needs a few more / different libraries to work. Please check the Arduino code in the zip file attached for required libraries and the following references for further information on programming and uploading to SPIFFS file system: SPIFFS library by espressif SPIFFS file uploader by me-no-dev ESP32WebServer library by Pedroalbuquerque This new version would make a whole new instructable which might be published in the future. But for now, especially for more experienced users, we don't want to miss the chance to share the basic information and files you need to set it up. Quick steps to build a stand-alone CoMoS: <ul readability=„5“>
Build a case according to the step before. Optionally, 3D-print an additional case for the micro SC card reader to be attached to the CoMoS case. If you don't have a 3D printer available, the card reader can be placed inside the CoMoS main case as well, no worries. Wire all sensors as described before, but in addition, install and wire a micro SD card reader (amazon.com) and a DS3231 real time clock (adafruit.com) as indicated in the wiring scheme attached to this step. Note: The pins for the pull-up resistor and the oneWire differ from the original wiring scheme! <li readability=„5“> Check the Arduino code and adjust the WiFi access point variables „ssid_AP“ and „password_AP“ to your personal preference. If not adjusted, the standard SSID is „CoMoS_AP“ and the password is „12345678“. <li readability=„6“> Insert micro SD card, upload the code, upload the content of the „data“ folder to the ESP32 using the SPIFFS file uploader, and connect any mobile device to the WiFi access point. <li

readability=„1“> Navigate to „192.168.4.1“ in your mobile browser and enjoy!   The app is all based on html, css, and javascript. It's local, no internet connection is involved or required. It features an in-app side menu to access a setup page and a memory page. On the setup page, you can adjust the most important settings like local date and time, sensor readings interval, etc. All settings will be stored permanently in the ESP32's internal storage and restored on next boot. On the memory page, a list of files on the SD card is available. Clicking a filename initiates a direct download of the CSV file to the mobile device. This system setup allows individual and remote monitoring of indoor environmental conditions. All sensor readings are stored on the SD card periodically, with new files being created for every new day. This allows a continuous operation for weeks or months without access or maintenance. As mentioned before, this is still an ongoing research and development. If you're interested in further details or assistance, please don't hesitate to contact the corresponding author through the comments or directly via LinkedIn. <aside class=„downloads“><h4 class=„sr-only“>Attachments</h4> </aside> </section><section id=„step10“ class=„step“ data-stepid=„SKE5X57JGZMFC21“ readability=„26“><h2 class=„step-title“>Step 10: Epilogue – Known Issues and Outlook</h2> <noscript> </noscript> The sensor station described in this instructable is the outcome of a long and ongoing research. The goal is to create a reliable, precise, yet low-cost sensor system for indoor environmental conditions. This held and holds some serious challenges, of which the most certain should be mentioned here: Sensor accuracy and calibration The sensors used in this project all offer relatively high accuracy at low or moderate cost. Most are equipped with internal noise reduction and a digital bus interfaces for communication, reducing the need for calibration or level adjustments. Anyway, because the sensors are installed in or on a case with certain attributes, a calibration of the complete sensor station was performed by the authors, as shown briefly by the pictures attached. A total of ten equally built sensor stations were tested in defined environmental conditions and compared to a TESTO 480 professional indoor climate sensor device. From these runs, the calibration factors included in the example code were determined. They allow a simple compensation of the case's and electronics' influence on the individual sensors. To reach the highest accuracy, an individual calibration for each sensor station is recommended. The calibration of this system is a second focus of the authors' research, besides the development and construction described in this instructable. It is discussed in an additional, connected publication, which is still in peer-review and will be linked here as soon as it goes online. Please find more information about this topic on the authors' website. ESP32 operation stability Not all Arduino-based sensor libraries used in this code are fully compatible with the ESP32 board. This issue has been widely discussed at many points online, especially regarding the stability of I2C and OneWire communication. In this development, a new, combined error detection and handling is carried out, based on powering the sensors directly through IO pins of the ESP32 to allow cutting their power supply for reset purpose. From today's perspective, this solution has not been presented or is not widely discussed. It was born of necessity, but to date is running smoothly for operation periods of several months and beyond. Yet it is still a topic of research. Outlook Together with this instructable, further written publications and conference presentations are carried out by the authors to spread the development and allow a wide and open source application. In meanwhile, the research is continued to further improve the sensor station, especially regarding system design and

manufacturability, and system calibration and verification. This instructable might be updated on important future developments, but for all up-to-date information, please visit the authors' website or contact the authors directly via LinkedIn: corresponding author: Mathias Kimmling second author: Konrad Lauenroth research mentor: Prof. Sabine Hoffmann </section> <div class=„contest-entries finalized-contests contest-entry“ readability=„23“> Second Prize in the
First Time Author <section id=„imadeits“ class=„imadeits“ readability=„0“><h2>Share</h2> </section><section class=„recommendations“><h2>Recommendations</h2> <ul class=„list-unstyled projects“> <article> <h3>3D Printer the C3Dt/bd (Big Delta)</h3> <footer readability=„-1“> by Core3D in Technology </footer> </article> <article> <h3>3D Printed Vacuum Cleaner for a CNC Machine</h3> <footer readability=„-1“> by Nikus in 3d Printing </footer> </article> <article> <h3>Opensource Ornithopter Prototype. Arduino Powered and Remote Controlled.</h3> <footer readability=„-1“> by gabbapeople in Arduino </footer> </article> <article> <h3>FeatureCAM Standard Class</h3> <footer> 1,486 Enrolled </footer> </article> <ul

class=„list-unstyled contests“> <h3 class=„sr-only“>Arduino Contest 2019</h3>   <h3 class=„sr-only“>Tape Contest</h3>   <h3 class=„sr-only“>Trash to Treasure</h3> </section> </html>