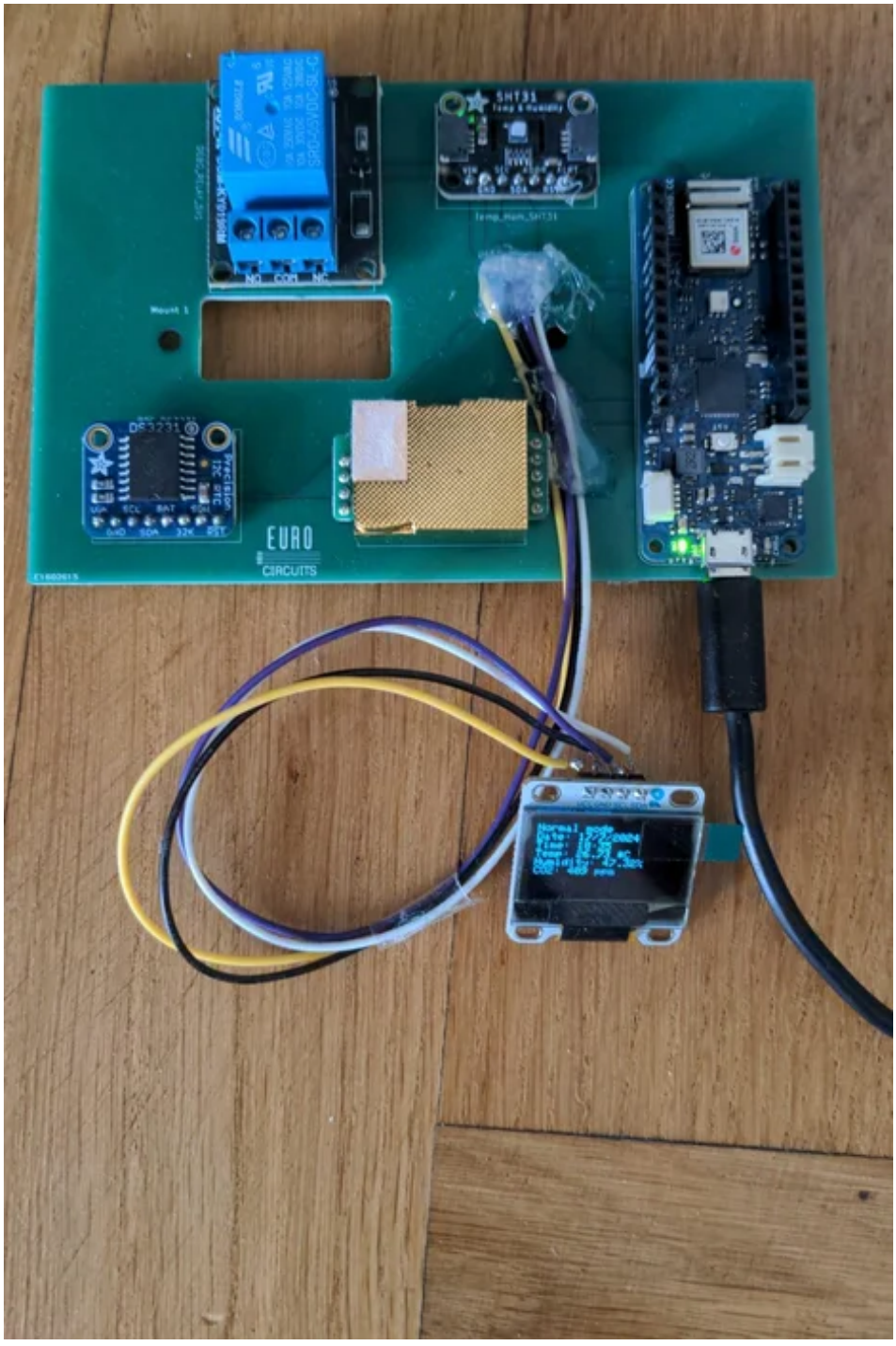
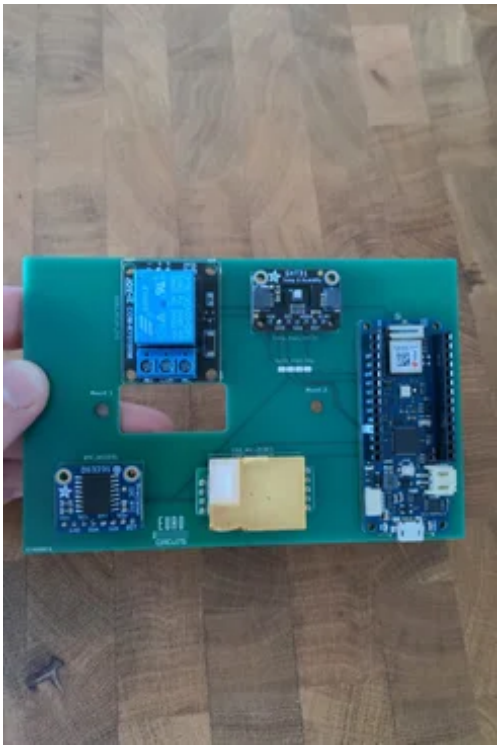
AUTODESK
Instructables

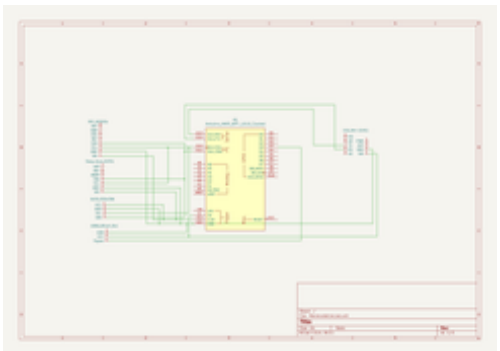
Build Your Own Smart Thermostat (Open Source)

By os-smart-thermostat in CircuitsAssistive Tech
Published Jul 18th, 2024

Introduction: Build Your Own Smart Thermostat (Open Source)







Welcome to this step-by-step guide on building a Smart Thermostat. This project aims to create an accessible, user-friendly, and open-source smart thermostat that you can customize to fit your specific needs.

This project is part of the "Media and Human-Centered Computing" study course at the Vienna University of Technology. This project aimed to create an open-source smart thermostat following a human-centred approach. By following this, we wanted to create an accessible design that everyone can adapt to their specific needs (if needed) to build a working thermostat.

Find more information on our public GitHub Repository: Open-Source-Smart-Thermostat

In case you want to contact us, please do so via github (not via instructables)!

Supplies

Hardware Components:

1. **Microcontroller:**Arduino MKR WiFi 1010 (The microcontroller that processes sensor data and controls the heating system. It features WiFi and Bluetooth. We only used Wifi for this project. Raspberry Pi would be also possible if you want to run a fully fledged OS.)
2. **Sensors**
3. *Humidity & Temperature Sensor:*Adafruit Sensirion SHT31-D
4. *CO2 Sensor (optional, we used it to collect additional data of air quality):* MH-Z19C Infrarot CO2 Sensor
5. **OLED Display:**0,96 Zoll OLED SSD1306 Display I2C
6. **Relay Module:**Relais SRD-05VDC-SL-C, 5V DC/250V AC 10A for controlling heating systems by acting as a switch
7. **Real Time Clock:** Real Time Clock RTC DS3231 I2C (for keeping track of exact time)
8. **Power Supply:** 5V/2A power adapter
9. **Enclosure:** 3D printed (find our draft in this instruction) or repurposed box for housing components
10. **Circuit Board,** either:
11. *Breadboard and Jumper Wires*(if you want to run it as a prototype and potentially extend it), *or*
12. *Custom Printed Circuit Board*(find our PCB draft attached to this instruction, we designed it with KiCad EDA and printed it with Eurocircuits)

Software Requirements:

1. **IDE:** Arduino IDE (you can use any you like as long as you manage to upload code to the Arduino)
2. **Programming:** You only need basic knowledge to assemble and make configuaration adjustments. If you want to extend the project some experience with the listed technologies might be helpful.
3. *Arduino:* C++
4. *Database:* SQL (MariaDB)
5. *Frontend:* TypeScript (Angular17 Frontend)
6. *Backend*: TypeScript (Node.js, Express)

Tools:

1. **Soldering Iron and Solder**
2. **Wire Strippers and Cutters**
3. **Screwdrivers**
4. **Multimeter**
5. **3D Printer** (optional, you could also print with an online service)

Prerequisite Check – Is This Usable for Me?

In most heating systems, a thermostat controls the heater by closing (shorting) a contact, which completes an electrical circuit and activates the heating unit. This thermostat works in the same way, using a relay to close the electrical circuit, if your current thermostat operates like this you will be able to use this thermostat too.

Additionally, you will need a server to utilise all the thermostat's features, such as remote monitoring and control. This server handles communication between you and the thermostat and collects the data. However, if you don't have a server, you can still build a standalone thermostat based on the physical components we used and maybe you can reuse some of our code.

SAFETY PRECAUTIONS

Working with high voltage components, such as those involved in controlling heating systems, can be dangerous. It is crucial to exercise caution and follow all safety guidelines to prevent accidents, injuries, or damage to your property. Always ensure that the power supply is disconnected before making any adjustments to the wiring or components. Use proper insulation and avoid touching live wires or exposed connections. If you are not experienced or comfortable working with high voltage, please seek assistance from a qualified professional. By following this guide, you acknowledge that you are doing so at your own risk. We, the creators of this guide, are not liable for any injuries, damages, or losses that may occur during the construction or operation of your smart thermostat. Always prioritize safety and take necessary precautions to protect yourself and others.

Step 1: 3D Print Housing (optional)

We designed a simple case to make the thermostat look more polished and resemble a typical thermostat. You can download our STL file and import it into Tinkercad or any other 3D modelling software. You can adapt it to your liking and then 3D print it. Our design is quite basic and serves as a prototype, so you can improve and customize it as needed.

If you want to use the PCB (we designed) to fit exactly in the case you can export the PCB layout from KiCad as a SVG file and import it into your 3D modelling software. This way you can design the case around the PCB and make sure everything fits.

Step 2: Print Your PCB (optional)

Why optional?

You could continue using your prototype running on a prototype. We designed a custom PCB in order to have a neat working solution that can be placed in the housing we designed.

Prerequisite: PCB Design

Adapt the PCB to your own needs if you want to integrate other sensors, buttons, and components or extend the design.

To modify the files we provide in our GitHub Repository – PCB Design you need to use KiCad.

Print the PCB

Print it with your preferred provide, we used Eurocircuits and were satisfied with the results. PCBWay works quite well too, but takes longer if you happen to live outside of China.

Bonus Tip

You may want to do this step later if you make any adjustments to your setup. We list it here as the printing of a custom PCB takes some time.

Step 3: Assembling Hardware & Microcontroller Setup

Hint: You can find a wiring diagram (of how we wired it) attached to this guide or in our public github repository. It is recommended to do the basic setup on a breadboard before printing our provided PCB (if you want to go for the neat solution, otherwise just leave it on a breadboard).

Microcontroller Setup

Begin the assembly process by setting up the microcontroller. Mount it on a breadboard, which will make it easier to prototype and make connections. Connect the microcontroller to the power supply, ensuring it receives a stable 5V/2A power source. This initial setup forms the foundation of your thermostat, providing the necessary control and processing capabilities for the rest of the components. Find more information on how to upload code and which libraries are required for setup here: Github Repository – Arduino Code Upload

Sensor Integration

Temperature & Humidity Sensor

Next, integrate the temperature & humidity sensor with the microcontroller. Depending on the type of sensor you choose, follow the specific wiring diagram to connect it correctly. For the Adafruit Sensor listed above, you will find it on their website. The sensor will provide real-time data about the ambient temperature and humidity, which the microcontroller will use to regulate the heating system. Secure the sensor in a position where it can accurately measure the room temperature.

CO2 Sensor (optional)

Repeat this approach for the CO2 Sensor if you want to collect data regarding the air quality of your room. The datasheet MH-Z19C Infrared CO2 Sensor can be found at berrybase.

OLED Display Connection

Connect the OLED display to the microcontroller, ensuring the pin configuration is correct. The display will serve as the user interface, displaying the current temperature, set temperature, and other relevant information. Properly wiring the display is crucial for clear and accurate display output.

Relay Module Setup

Set up the relay module, which will control the heating system based on the temperature readings and user input. Connect the relay to the microcontroller, ensuring it can handle the load of your heating system. The relay acts as a switch that the microcontroller can turn on or off to regulate the temperature. Test the relay operation to ensure it responds correctly to control signals from the microcontroller.

Step 4: Programming the Thermostat

How Does The Thermostat Work?

The main loop function continuously runs and performs the following tasks:

1. Sends a heartbeat to the server at regular intervals.
2. Enqueues sensor data requests if the heartbeat was successful.
3. Processes any pending requests in the queue.
4. Attempts to reconnect if in fallback mode and sufficient time has passed.
5. Controls the heating relay based on temperature in fallback mode.
6. Updates the display at regular intervals.

Setting Up the Development Environment

Set up the development environment on your computer by installing the necessary software. If you are using an Arduino, download and install the Arduino IDE. For a Raspberry Pi, set up the appropriate development environment. Ensure you have all the required libraries for the temperature sensor, LCD display, and any other components you are using. This setup will provide the tools you need to write, upload, and debug the code for your thermostat.

Find more information on how to upload code and which libraries are required for setup here: Github Repository – Arduino Code Upload

Uploading Code to the Arduino MKR WiFi 1010

To upload the code to the Arduino, use the Arduino IDE.

Configure the Arduino IDE

1. Install the Arduino IDE: Download and install the Arduino IDE from the official website.
2. Install the Board: Go to Tools -> Board -> Boards Manager. Search for "Arduino SAMD Boards" and install it.
3. Open Library Manager: Go to Sketch -> Include Library -> Manage Libraries.
4. Install the following Libraries:

- WiFiNINA
- Adafruit GFX Library
- Adafruit SSD1306
- RTClib
- MHZ19
- LinkedList
- Adafruit SHT31

Connect Arduino MKR 1010 WiFi

1. Connect the Board: Plug your Arduino MKR 1010 WiFi into your computer using a USB cable.
2. Select the Board: Go to Tools -> Board and select Arduino MKR WiFi 1010.
3. Select the Port: Go to Tools -> Port and select the port corresponding to your connected board (e.g., COM3, /dev/ttyUSB0).

Prepare the Sketch

1. Open the Sketch: Open the Arduino sketch file (.ino) in the Arduino IDE.
2. Configure the Sketch: Modify the sketch to match your server IP, WiFi credentials, and other settings.

Upload the Sketch

1. Verify the Sketch: Click the checkmark icon at the top left of the Arduino IDE to compile and verify the code. This ensures there are no syntax errors.
2. Upload the Sketch: Click the right-arrow icon next to the checkmark to upload the code to the Arduino MKR 1010 WiFi. The IDE will compile the code again and then upload it to the board.

Monitor Serial Output

1. Open Serial Monitor: Go to Tools -> Serial Monitor to open the Serial Monitor.
2. Set Baud Rate: Ensure the baud rate at the bottom of the Serial Monitor is set to 9600 to match the Serial.begin(9600); setting in your code.
3. View Output: You should see the output from your Arduino, which includes debug messages and sensor readings.

Adjust the Configuration

You need to adjust the WiFi and Server configuration to your own setup. These have to be adjusted to match your server's IP address and WiFi network.

```
const char *ssid = "WIFI_SSID";
const char *password = "WIFI_PASSWORD";
const char *server = "SERVER_API";
const String username = "SERVER_USERNAME";
const String userPassword = "SERVER_PASSWORD";
const int port = 443;


const String authenticateURL = "/api/authenticate";
const String temperatureURL = "/api/sensor/temperature";
const String humidityURL = "/api/sensor/humidity";
const String co2URL = "/api/sensor/co2";
const String heartbeatURL = "/api/heartbeat";
const String heatingStatusURL = "/api/heating/status";
```

Optional: Adjust Variables for Operation

```
String token;
#define FALLBACK_TEMPERATURE 15.0
bool isHeatingOn = false;
bool fallbackMode = false;
bool heartbeatSuccess = true;

unsigned long lastHeartbeatMillis = 0;
unsigned long lastReconnectMillis = 0;
unsigned long lastDisplayMillis = 0;

const unsigned long WIFI_TIMEOUT = 30000;        // 30 seconds
```

```
const unsigned long HEARTBEAT_INTERVAL = 180000; // 3 minutes
const unsigned long DISPLAY_INTERVAL = 5000;     // 5 seconds
const unsigned long RECONNECT_INTERVAL = 300000; // 5 minutes
```

These variables and constants manage the operational state, such as connection status, heating status, and timing intervals for the tasks. The FALLBACK_TEMPERATURE is used when the server is unreachable, and the thermostat operates in fallback mode.
The HEARTBEAT_INTERVAL determines how often the Arduino sends heartbeats to the server.

Optional: Extend the Code

If you want to make any adjustments to the provided code feel free to do so. If you think your adjustments might be helpful for others, please provide a pull request to our GitHub Repository.

Step 5: Setup DB, Server and Frontend

Database

For the server of the smart thermostat to function correctly it needs a database. For this we used MariaDB.

Find the information on how to set it up here: GitHub Repository – Database Setup

Frontend

The frontend is developed using Angular17, allowing users to view data and configure the temperature settings.

Find the information on how to set it up here: GitHub Repository – Frontend Setup

Backend

The backend is developed using Node.js and Express, providing APIs for sensor data, authentication, and heating control.

Find the information on how to set it up here: GitHub Repository – Backend Setup

Step 6: Testing and Calibration (Optional)

Initial Testing

Power on your thermostat and conduct initial testing to ensure all components are working correctly. Check the OLED display for proper output, and verify that the temperature sensor is providing accurate readings. Test the relay module by setting various temperature thresholds and observing if the relay activates or deactivates the heating system accordingly. This initial testing phase is crucial for identifying any issues early on and making necessary adjustments.

Calibration

Calibrate your thermostat to ensure it operates accurately and reliably. Compare the sensor readings with a trusted thermometer to verify accuracy. Adjust the code or sensor placement if necessary to match the actual temperature more closely. Ensure the relay activates and deactivates at the correct temperature settings, providing precise control over the heating system. This calibration process will optimize the performance of your thermostat and ensure it meets your desired specifications.

Step 7: Share Your Results With the Community

Thank you for building our thermostat. Let us know of any issues you encounter and share your results here with the community. We would be grateful if you let the community know of your progress by clicking the "I Made It"-Button.