Home Assistant

- Measure time the TV is ON
- ESPHome code for 7,5 inch Waveshare ePaper Display - Home Automation

Measure time the TV is ON

To track how long your TV is on in Home Assistant, you can use the **template sensor** and **history statistics** integration. Assuming your TV has an entity (e.g., `switch.tv` or `media_player.tv`), here's a YAML setup to calculate the time it's on each day.

Step 1: Create a Template Binary Sensor

First, create a binary sensor to detect when your TV is on:

```
binary_sensor:
  - platform: template
    sensors:
      tv_status:
        friendly_name: "TV Status"
        value_template: "{{ is_state('media_player.tv', 'on') }}"
```

Replace `media_player.tv` with the actual entity ID of your TV. This sensor will be `on` when the TV is on.

Step 2: Use the History Statistics Sensor

Now, use the **history statistics sensor** to calculate the total time the TV has been on each day:

```
sensor:
  - platform: history_stats
    name: TV On Time Today
    entity_id: binary_sensor.tv_status
    state: "on"
    type: time
    start: "{{ now().replace(hour=0, minute=0, second=0) }}"
    end: "{{ now() }}"
```

This setup will provide a `TV On Time Today` sensor, which resets every day and accumulates the time your TV was on.

Optional: Adding a Weekly or Monthly Tracker

You can duplicate the **history_stats** sensor and adjust the `start` time to track weekly or monthly usage:

```
- platform: history_stats
  name: TV On Time This Week
  entity_id: binary_sensor.tv_status
  state: "on"
  type: time
  start: "{{ now().replace(hour=0, minute=0, second=0) - timedelta(days=now().weekday()) }}"
  end: "{{ now() }}"
```

This will give you daily, weekly, and customizable duration insights on your TV's usage. Let me know if you'd like to add automations or notifications based on this data!

ESPHome code for 7,5 inch Waveshare ePaper Display - Home Automation


```
# 15-10-2023 Updated the code due to changes in the FordPass Api
# Also changes the deep-sleep method.
# Made a input_boolean in Home Assistant to be able to prevent deep sleep like:
#
# prevent_deep_sleep:
# name: "Prevent deep sleep"
# initial: false
```

```
#
esphome:
name: "epaper1"
on_boot:
priority: -200.0
then:
- component.update: eink_display
- wait_until:
condition:
lambda: 'return id(data_updated) == true;'
# Wait a bit longer so all the items are received
- delay: 5s
- logger.log: "Initial sensor data received: Refreshing display..."
- lambda: "id(initial_data_received) = true;"
- script.execute: update_screen
- delay: 20s
- script.execute: consider_deep_sleep


esp32:
board: esp32dev
framework:
type: arduino


# Enable logging
logger:



# Enable Home Assistant API
api:
encryption:
key: "xDIfV9w1+pU9uTXCGcNpG2tas8UgDeiNflfczJYJvnA="


ota:
- platform: esphome
```

```
wifi:
ssid: !secret wifi_ssid_iot
password: !secret wifi_password_iot
# Enable fallback hotspot (captive portal) in case wifi connection fails
# ap:
# ssid: "Epaper1 Fallback Hotspot"
# password: !secret fallback_pw


captive_portal:


web_server:


# Global variables for detecting if the display needs to be refreshed. (Thanks @paviro!)
globals:
- id: data_updated
type: bool
restore_value: no
initial_value: 'false'
- id: initial_data_received
type: bool
restore_value: no
initial_value: 'false'
- id: recorded_display_refresh
type: int
restore_value: yes
initial_value: '0'



# Script for updating screen - Refresh display and publish refresh count and time. (Thanks @paviro!)
script:
- id: update_screen
then:
- lambda: 'id(data_updated) = false;'
- component.update: eink_display
- lambda: 'id(recorded_display_refresh) += 1;'
- id: consider_deep_sleep
```

```
mode: queued
then:
- delay: 15s
- if:
condition:
binary_sensor.is_on: prevent_deep_sleep
then:
- logger.log: 'Skipping sleep, per prevent_deep_sleep'
else:
- deep_sleep.enter: deep_sleep_control


- script.execute: consider_deep_sleep



time:
- platform: sntp
id: ntp
timezone: Europe/Amsterdam
servers:
- 0.pool.ntp.org
- 1.pool.ntp.org
- 2.pool.ntp.org
# Check whether the display needs to be refreshed every minute,
# based on whether new data is received or motion is detected. (Thanks @paviro!)
- platform: homeassistant
id: homeassistant_time
on_time:
- seconds: 0
minutes: /45
then:
- script.execute: update_screen


# Pins for Waveshare ePaper ESP Board
spi:
clk_pin: GPIO13
mosi_pin: GPIO14
```

```
# Deep Sleep
########################################################################
deep_sleep:
id: deep_sleep_control
sleep_duration: 45min



font:
- file: "fonts/GoogleSans-Medium.ttf"
id: xtra_large_font
size: 80
glyphs:
[ '-', '.','°', '0','1', '2', '3', '4', '5', '6', '7', '8', '9']
- file: "fonts/GoogleSans-Medium.ttf"
id: large_font
size: 56
glyphs:
['&', '@', '!', ',', '.', '"', '%', '(', ')', '+', '-', '_', ':', '°', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E',
'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ' ', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z','å', 'ä', 'ö', '/']
- file: "fonts/GoogleSans-Bold.ttf"
id: medium_fontb
size: 36
glyphs:
['&', '@', '!', ',', '.', '"', '%', '(', ')', '+', '-', '_', ':', '°', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E',
'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ' ', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z', 'å', 'ä', 'ö', '/']
- file: "fonts/GoogleSans-Medium.ttf"
id: medium_fontm
size: 40
glyphs:
['&', '@', '!', ',', '.', '"', '%', '(', ')', '+', '-', '_', ':', '°', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E',
```

'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ' ', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z', 'å', 'ä', 'ö', '/']
- file: "fonts/GoogleSans-Medium.ttf"
id: small_font
size: 26
glyphs:
['&', '@', '!', ',', '.', '"', '%', '(', ')', '+', '-', '_', ':', '°', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E',
'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ' ', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z', 'å', 'ä', 'ö', '/']
- file: "fonts/GoogleSans-Medium.ttf"
id: xtra_small_font
size: 18
glyphs:
['&', '@', '!', ',', '.', '"', '%', '(', ')', '+', '-', '_', ':', '°', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E',
'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ' ', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z', 'å', 'ä', 'ö', '/']

https://pictogrammers.github.io/@mdi/font/5.3.45/
- file: "fonts/materialdesignicons-webfont.ttf"
id: font_icons_small
size: 26
glyphs:
- "\U000F140B" # Energy
- "\U000F0E1B" # Car
- "\U000F03C7" # Car oil
- "\U000F010C" # Car battery
- "\U000F07CA" # mdi-fuel
- "\U000F109D" # Car doors closed
- "\U000F0B6B" # Car doors open
- "\U000F05B1" # Car windows open
- "\U000F05AE" # Car windows closed

```
- "\U000F029A" # mdi-gauge
- "\U000F058E" # mdi-water-percent
- "\U000F07E4" # mdi-molecule-co2


- file: "fonts/materialdesignicons-webfont.ttf"
id: font_icons_medium
size: 36
glyphs:
- "\U000F10C2" # Temperature High
- "\U000F10C3" # Temperature Low
- "\U000F050F" # mdi-thermometer
- "\U000F029A" # mdi-gauge
- "\U000F058E" # mdi-water-percent
- "\U000F07E4" # mdi-molecule-co2
- "\U000F059D" # mdi-weather-windy
- "\U000F04E0" # mdi-sunglasses


- file: "fonts/materialdesignicons-webfont.ttf"
id: font_icons_large
size: 80
glyphs:
- "\U000F0599" # mdi-weather-sunny (clear)
- "\U000F0595" # mdi-weather-partly-cloudy (partlycloudy)
- "\U000F0590" # mdi-weather-cloudy (cloudy)
- "\U000F0591" # mdi-weather-fog (partlycloudy-fog)
- "\U000F0F33" # mdi-weather-partly-rainy (partlycloudy-light-rain)
- "\U000F0F32" # mdi-weather-partly-lightning (partlycloudy-rain)
- "\U000F0597" # mdi-weather-rainy (light-rain)
- "\U000F0596" # mdi-weather-pouring (rainy)
- "\U000F067F" # mdi-weather-snowy-rainy (snowy-rainy)
- "\U000F0F35" # mdi-weather-partly-snowy-rainy (partlycloudy-light-snow)
- "\U000F0F34" # mdi-weather-partly-snowy (partlycloudy-snow)
- "\U000F0598" # mdi-weather-snowy (light-snow)
- "\U000F0F36" # mdi-weather-snowy-heavy (snowy)
- "\U000F067E" # mdi-weather-lightning-rainy (partlycloudy-lightning)
- "\U000F0593" # mdi-weather-lightning (lightning)
```

```
sensor:


# Buienradar
- platform: homeassistant
entity_id: sensor.temperatuur
id: br_temperature
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: weather.buienradar
id: br_humidity
attribute: humidity
unit_of_measurement: "%"
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'


- platform: homeassistant
entity_id: weather.buienradar
id: br_pressure
attribute: pressure
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.wind_force
id: br_wind_force
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.wind_speed
id: br_wind_speed
```

```
unit_of_measurement: "km/h"
state_class: "measurement"
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: wind_speed
entity_id: weather.buienradar
attribute: wind_speed
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
# Ford
- platform: homeassistant
entity_id: sensor.fordpass_odometer
id: odometer
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.distance_to_empty
id: dis_to_empty
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.front_left
id: front_left
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.front_right
id: front_right
internal: true
```

```
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.rear_left
id: rear_left
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.rear_right
id: rear_right
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
# Huis
- platform: homeassistant
id: temp_woonk
entity_id: climate.woonkamer
attribute: current_temperature
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: temp_woonk_set
entity_id: climate.woonkamer
attribute: temperature
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: hum_woonk
entity_id: sensor.woonkamer_humidity
internal: true
on_value:
then:
```

```
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: co2_woonk
entity_id: sensor.senseair_co2_value
unit_of_measurement: "ppm"
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: temp_zolder
entity_id: climate.zolder
attribute: current_temperature
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: hum_zolder
entity_id: sensor.zolder_slk_hum
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
id: co2_zolder
entity_id: sensor.airquality2_co2_value
unit_of_measurement: "ppm"
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'



# UV index (https://www.home-assistant.io/integrations/openuv/), updated via automation.yaml
- platform: homeassistant
id: cur_uv_index
entity_id: sensor.current_uv_index
internal: true
on_value:
```

```
then:
- lambda: 'id(data_updated) = true;'


- platform: homeassistant
id: max_uv_index
entity_id: sensor.max_uv_index
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'


- platform: homeassistant
entity_id: sensor.fordpass_oil
id: car_oil
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.fordpass_battery
id: car_battery
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'


text_sensor:
# sun/moon
- platform: homeassistant
entity_id: sun.sun
id: sun
internal: true


- platform: homeassistant
id: br_condition
entity_id: sensor.condition
```

```
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'


- platform: homeassistant
entity_id: sensor.wind_direction
id: br_windrichting
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'
- platform: homeassistant
entity_id: sensor.fordpass_doorstatus
id: car_doors
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'


- platform: homeassistant
entity_id: sensor.fordpass_windowposition
id: car_windows
internal: true
on_value:
then:
- lambda: 'id(data_updated) = true;'


binary_sensor:
- platform: homeassistant
id: prevent_deep_sleep
entity_id: input_boolean.prevent_deep_sleep
on_press:
then:
- deep_sleep.prevent: deep_sleep_control
```

```
# Now render everything on the ePaper screen.
display:
- platform: waveshare_epaper
id: eink_display
cs_pin: GPIO15
dc_pin: GPIO27
# THE BUSY PIN MUST BE INVERTED ON 7.50inv2 ePaper Screens!
busy_pin:
number: GPIO25
# inverted: True
reset_pin: GPIO26
# 800x480 px
# model: 7.50inV2
model: 7.50inV2alt
update_interval: never
reset_duration: 2ms
rotation: 270°
lambda: |-
#define xres 480
#define yres 800
#define x_pad 20 // border padding
#define y_pad 60 // border padding
#define val_pad 70 // padding before value
#define icon_y_pad 8 //padding after icons
#define y_header_weer 70 // Plaats van de Weer header
#define y_header_huis 340 // Plaats van de Huis header
#define y_header_ford 570 // Plaats van de Ford header
#define weather_icon_x xres/4-x_pad


int y = 10;


if(isnan(id(br_temperature).state))
{
it.printf(20, y+90, id(small_font), TextAlign::LEFT, "Waiting for data .....");
}
else
{
```

```
// HET WEER

//Header with a line from e.g. [x=0,y=0] to [x=50,y=50]
it.printf(x_pad, y_header_weer, id(medium_fontb), TextAlign::BASELINE_LEFT, "Weer");
it.line(x_pad+100, y_header_weer-10, xres-x_pad, y_header_weer-10);

// it.printf(x_pad, y_header_weer+40, id(medium_fontm), TextAlign::BASELINE_LEFT, "%s",
id(br_symbol).state.c_str());

// Buiten temperatuur
// if (id(br_temperature).state < 10)
// {
// it.printf(xres-x_pad-180, y_header_weer+68, id(font_icons_medium), TextAlign::BASELINE_LEFT,
"\U000F050F");
// }
//else
// {
// it.printf(xres-x_pad-200, y_header_weer+68, id(font_icons_medium), TextAlign::BASELINE_LEFT,
"\U000F050F");
// }

it.printf(xres-x_pad, y_header_weer+70, id(xtra_large_font), TextAlign::BASELINE_RIGHT, "%2.1f°",
id(br_temperature).state);

// Buiten vochtigheid
it.printf(x_pad, y_header_weer+60, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F058E");
it.printf(x_pad+40, y_header_weer+60, id(medium_fontm), TextAlign::BASELINE_LEFT, "%2.0f%%",
id(br_humidity).state);

// Luchtdruk
it.printf(x_pad, y_header_weer+105, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F029A");
```

```
it.printf(x_pad+40, y_header_weer+105, id(medium_fontm), TextAlign::BASELINE_LEFT, "%4.0f hPa",
id(br_pressure).state);


// Wind
it.printf(x_pad, y_header_weer+150, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F059D");
it.printf(x_pad+40, y_header_weer+150, id(medium_fontm), TextAlign::BASELINE_LEFT, "(%1.0f)",
id(br_wind_force).state);
it.printf(x_pad+100, y_header_weer+150, id(medium_fontm), TextAlign::BASELINE_LEFT, "%2.1f km/h",
id(wind_speed).state);


//UV Index
it.printf(x_pad, y_header_weer+195, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F04E0");
it.printf(x_pad+45, y_header_weer+195, id(medium_fontm), TextAlign::BASELINE_LEFT, "%1.1f",
id(cur_uv_index).state);
it.printf(x_pad+100, y_header_weer+195, id(medium_fontm), TextAlign::BASELINE_LEFT, "/%1.1f",
id(max_uv_index).state);



// Windrichting
it.printf(xres-x_pad-10, y_header_weer+200, id(medium_fontm), TextAlign::BASELINE_RIGHT, "%s",
id(br_windrichting).state.c_str());


// Icoon
if (id(br_condition).state == "clear") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0599");}
if (id(br_condition).state == "partlycloudy") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0595");}
if (id(br_condition).state == "cloudy") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0590");}
if (id(br_condition).state == "partlycloudy-fog") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0591");}
if (id(br_condition).state == "partlycloudy-light-rain") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0F33");}
if (id(br_condition).state == "partlycloudy-rain") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0F32");}
if (id(br_condition).state == "light-rain") {
```

```
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0597");}
if (id(br_condition).state == "rainy") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0596");}
if (id(br_condition).state == "snowy-rainy") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F067F");}
if (id(br_condition).state == "partlycloudy-light-snow") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0F35");}
if (id(br_condition).state == "partlycloudy-snow") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0F34");}
if (id(br_condition).state == "light-snow") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0598");}
if (id(br_condition).state == "snowy") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0F36");}
if (id(br_condition).state == "partlycloudy-lightning") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F067E");}
if (id(br_condition).state == "lightning") {
it.printf(xres-x_pad, y_header_weer+150, id(font_icons_large), TextAlign::BASELINE_RIGHT, "\U000F0593");}


// HUIS
//Header with a line from e.g. [x=0,y=0] to [x=50,y=50]
it.printf(x_pad, y_header_huis, id(medium_fontb), TextAlign::BASELINE_LEFT, "Huis");
it.line(x_pad+100, y_header_huis-10, xres-x_pad, y_header_huis-10);


it.printf(x_pad, y_header_huis+30, id(xtra_small_font), TextAlign::BASELINE_LEFT, "Woonkamer");


// Binnen vochtgehalte woonkamer
it.printf(x_pad, y_header_huis+70, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F058E");
it.printf(x_pad+40, y_header_huis+70, id(medium_fontm), TextAlign::BASELINE_LEFT, "%2.0f%%",
id(hum_woonk).state);


// Binnen co2 woonkamer
it.printf(x_pad+160, y_header_huis+75, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F07E4");
it.printf(x_pad+200, y_header_huis+70, id(medium_fontm), TextAlign::BASELINE_LEFT, "%4.0f",
id(co2_woonk).state);
```

```
// Binnen temperatuur woonkamer
it.printf(xres-x_pad-120, y_header_huis+68, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F050F");
// it.printf(xres-110, y_header_huis+50, id(medium_fontm), TextAlign::BASELINE_RIGHT, "%2.1f",
id(temp_woonk_set).state);
it.printf(xres-x_pad, y_header_huis+70, id(medium_fontm), TextAlign::BASELINE_RIGHT, "%2.1f°",
id(temp_woonk).state);


it.printf(x_pad, y_header_huis+100, id(xtra_small_font), TextAlign::BASELINE_LEFT, "Zolder");


// Binnen vochtgehalte zolder
it.printf(x_pad, y_header_huis+140, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F058E");
it.printf(x_pad+40, y_header_huis+140, id(medium_fontm), TextAlign::BASELINE_LEFT, "%2.0f%%",
id(hum_zolder).state);


// Binnen co2 zolder
it.printf(x_pad+160, y_header_huis+145, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F07E4");
it.printf(x_pad+200, y_header_huis+140, id(medium_fontm), TextAlign::BASELINE_LEFT, "%4.0f",
id(co2_zolder).state);


// Binnen temperatuur Zolder
it.printf(xres-x_pad-120, y_header_huis+140, id(font_icons_medium), TextAlign::BASELINE_LEFT, "\U000F050F");
it.printf(xres-x_pad, y_header_huis+140, id(medium_fontm), TextAlign::BASELINE_RIGHT, "%2.1f°",
id(temp_zolder).state);




//FORD


//Header with a line from [x=0,y=0] to [x=50,y=50]
it.printf(x_pad, y_header_ford, id(medium_fontb), TextAlign::BASELINE_LEFT, "Ford");
it.line(x_pad+100, y_header_ford-10, xres-x_pad, y_header_ford-10);


if(isnan(id(odometer).state))
```

```
{
//Header with a line from [x=0,y=0] to [x=50,y=50]
it.printf(x_pad, y_header_ford+40, id(small_font), TextAlign::BASELINE_LEFT, "No data available...");
}
else
{


// Ford km
it.printf(x_pad, y_header_ford+40, id(font_icons_small), TextAlign::BASELINE_LEFT, "\U000F0E1B");
it.printf(x_pad+40, y_header_ford+40, id(small_font), TextAlign::BASELINE_LEFT, "%5.0f km",
id(odometer).state);


// Ford fuel
it.printf(x_pad, y_header_ford+65, id(font_icons_small), TextAlign::BASELINE_LEFT, "\U000F07CA");
if (id(dis_to_empty).state > 0 )
{
it.printf(x_pad+40, y_header_ford+65, id(small_font), TextAlign::BASELINE_LEFT, "%3.0f km",
id(dis_to_empty).state);
}


// Ford oil
it.printf(x_pad, y_header_ford+90, id(font_icons_small), TextAlign::BASELINE_LEFT, "\U000F03C7");
if (id(car_oil).state > 0 )
{
it.printf(x_pad+40, y_header_ford+90, id(small_font), TextAlign::BASELINE_LEFT, "%2.0f%%", id(car_oil).state);
}


// Ford battery
it.printf(x_pad, y_header_ford+115, id(font_icons_small), TextAlign::BASELINE_LEFT, "\U000F010C");
if (id(car_battery).state > 0)
{
it.printf(x_pad+40, y_header_ford+115, id(small_font), TextAlign::BASELINE_LEFT, "%2.0f%%",
id(car_battery).state);
}
```

```
// Ford tire pressure
// it.printf(xres-110, y_header_ford+105, id(xtra_small_font), TextAlign::BASELINE_LEFT, "Banden");
it.printf(xres-x_pad, y_header_ford+40, id(small_font), TextAlign::BASELINE_RIGHT, "LV %1.2f",
id(front_left).state);
it.printf(xres-x_pad, y_header_ford+65, id(small_font), TextAlign::BASELINE_RIGHT, "RV %1.2f",
id(front_right).state);
it.printf(xres-x_pad, y_header_ford+90, id(small_font), TextAlign::BASELINE_RIGHT, "LA %1.2f",
id(rear_left).state);
it.printf(xres-x_pad, y_header_ford+115, id(small_font), TextAlign::BASELINE_RIGHT, "RA %1.2f",
id(rear_right).state);



}



// Show date and time of last update
it.strftime((xres/2), yres-y_pad, id(xtra_small_font), TextAlign::BASELINE_CENTER, "Last update: %A %d %b %Y
%H:%M", id(ntp).now());



//Divider draw a line from [x=0,y=0] to [x=50,y=50]
it.line(x_pad, yres-y_pad-30, xres-x_pad, yres-y_pad-30);



}
```