```
-- LUA Script für Taranis Q X7 mit OpenTX 2.3.11                                    --
-- ******************************************************************************** --
-- Guden, 03.06.2021
--
--
--
-- Script zur freien Verwendung, keine Garantie auf Funktion                        --
--
--
--
-- Das Script fügt eine Telemetrie-Seite in die Taranis Q X7 ein und zeigt Funktionen für    --
-- einen TRX-4 an.
--
--
--
-- Kapazität des Akku's, Schalterzuweisungen und Telemetire-Werte können in der ersten       --
-- Sektion angepasst werden. Je nach OpenTX Einstellungen sind die entsprechenden Schalter,  --
-- Kanäle und Telemetrie-Namen anzupassen.                                          --
--                                                                                  --
-- ******************************************************************************** --

-- *** Funktion zum runden von Zahlen ********************************************** --
function round(num, decimals)
        local mult = 10^(decimals or 0)                         -- Multiplikator entsprechend der
Anzahl gewünschter Kommastellen
        return math.floor(num * mult + 0.5) / mult      -- Zahl multiplizieren, 0.5 addieren
(damit korrekt gerundet wird), auf ganze Zahl abrunden und dann durch Multiplikator teilen
end
-- *** Funktion zum runden von Zahlen ********************************************** --

-- *** Konstanten und Signale definieren ******************************************* --
-- Eingänge für Spannungs- und Verbrauchswerte --
local telemetry_cell_min = 'Low'                                -- Telemetrie-Sensor für tiefste
Spannung der Einzelzellen
local telemetry_cell_diff = 'Diff'                              -- Telemetrie-Sensor für Differenz
zwischen höchster und tiefster Einzelzellen
local telemetry_transmitter = 'tx-voltage'              -- Telemetrie-Sensor für Spannung des
Senders
local telemetry_bec = 'RxBt'                                    -- Telemetrie-Sensor für Spannung
des BEC

-- Eingänge für Differentialsperren --
local diff_fwd = 'sf'                                                   -- Schaltkanal für
sperren/entsperren des vorderen Differentials
local diff_rear = 'sh'                                                  -- Schaltkanal für
sperren/entsperren des hinteren Differentials

-- Eingänge für die Temperaturen --
local telemetry_temp_mot = 'TmpM'                               -- Telemetrie-Sensor für
Temperatur des Motors
local telemetry_temp_esc = 'TmpR'                               -- Telemetrie-Sensor für
Temperatur des ESC

-- Eingänge für Gangschaltung --
local gear = 'ch4'                                                              -- Schaltkanal für
Umschaltung 1/2 Gang

-- Eingänge für Geschwindigkeit --
local telemetry_speed = 'VSpd'                                  -- Telemetrie-Sensor für
Geschwindigket

-- Eingänge für Seilwinde --
local winch = 's2'                                                              -- Schaltkanal für
Umschaltung Winde aus / Winde ein

-- Eingänge für RSSI Signal --
local telemetry_rssi = 'RSSI'                                   -- Telemetrie-Sensor für RSSI
Signal
```

```
-- Display-Daten definieren --
local min_x, min_y = 0, 0                                          -- obere linke Ecke des
Displays
local max_x, max_y = LCD_W - 1, LCD_H - 1               -- untere rechte Ecke des Displays
local cell_width = round(max_x / 3, 0)                  -- Zellenbreite berechnen
local cell_height = round(max_y / 3, 0)                 -- Zellenhöhe berechnen
-- *** Konstanten und Signale definieren ********************************************** --

-- *** Bildschirmunterteilung zeichnen ********************************************** --
local function drawGrid()
        lcd.drawLine(min_x, min_y + cell_height, max_x, min_y + cell_height, SOLID, FORCE)
-- erste Zeile
        lcd.drawLine(min_x, min_y + cell_height * 2, max_x, min_y + cell_height * 2, SOLID, FORCE)
-- zweite Zeile

        -- die obere und untere Zeile haben eine Höhe von 21 Pixel. Die mittlere von lediglich 20
Pixel (geht rein rechnerisch nicht anders)

        lcd.drawLine(min_x + cell_width, min_y, min_x + cell_width, max_y, SOLID, FORCE)
-- erste Spalte
        lcd.drawLine(min_x + cell_width * 2 + 1, min_y, min_x + cell_width * 2 + 1, max_y, SOLID,
FORCE)  -- zweite Spalte

        -- alle Spalten haben eine Breite von 42 Pixel
end
-- *** Bildschirmunterteilung zeichnen ********************************************** --

-- *** Uhr anzeigen ***************************************************************** --
local function drawClock()
        lcd.drawText(1, 1, "Uhr", SMLSIZE + INVERS)
-- Text "Uhr" invertiert anzeigen

        local datenow = getDateTime()
-- Uhrzeit in Variable einlesen
        lcd.drawText(4, 12, string.format("%.2d", datenow.hour) .. ":" ..
                                            string.format("%.2d", datenow.min) .. ":" ..
                                            string.format("%.2d", datenow.sec), SMLSIZE)    --
Uhrzeit als hh:mm:ss ausgeben
end
-- *** Uhr anzeigen ***************************************************************** --

-- *** Trip anzeigen **************************************************************** --
local function drawTrip()
        lcd.drawText(1, 22, "Trip", SMLSIZE + INVERS)                                          --
Text "Trip" invertiert (Bezeichnung für Timer 1) anzeigen

        timer = model.getTimer(0)
-- Timer 1 des aktuellen Modells in Variable einlesen
        s = timer.value
-- Wert von Timer 1 einlesen
        time = string.format("%.2d:%.2d:%.2d", s/(60*60), s/60%60, s%60)        -- Wert in
hh:mm:ss formatieren
        lcd.drawText(4, 33, time, SMLSIZE)
-- Timer anzeigen
end
-- *** Trip anzeigen **************************************************************** --

-- *** Run anzeigen ***************************************************************** --
local function drawRun()
        lcd.drawText(1, 43, "Run", SMLSIZE + INVERS)                                           --
Text "Run" invertiert (Bezeichnung für Timer 2) anzeigen

        timer = model.getTimer(1)
-- Timer 2 des aktuellen Modells in Variable einlesen
        s = timer.value
-- Wert von Timer 2 einlesen
        time = string.format("%.2d:%.2d:%.2d", s/(60*60), s/60%60, s%60)        -- Wert in
hh:mm:ss formatieren
        lcd.drawText(4, 54, time, SMLSIZE)
```

```
-- Timer anzeigen
end
-- *** Run anzeigen ********************************************************************** --

-- *** Zellenspannung anzeigen ********************************************************** --
local function drawCellVoltage()
        local min_cell = getValue(telemetry_cell_min)
-- Spannung der niedrigsten Zelle einlesen
        local cell_diff = getValue(telemetry_cell_diff)
-- Differenz zwischen niedrigster und höchster Zelle einlesen

        -- Spannung der höchsten Zelle --
        local max_cell = round(min_cell + cell_diff, 2)
-- höchste Zellspannung aus kleinster und Differenz berechnen und auf 2 Nachkommastellen runden
        lcd.drawText(45, 3, "high" .. " " .. string.format("%.2f", max_cell), SMLSIZE)          --
Zellspannung ausgeben im Format x.xx

        -- Spannung der niedrigsten Zelle --
        min_cell = round(min_cell, 2)
-- auf 2 Nachkommastellen runden
        lcd.drawText(47, 12, "low" .. " " .. string.format("%.2f", min_cell), SMLSIZE)          --
Zellspannung ausgeben im Format x.xx
end
-- *** Zellenspannung anzeigen ********************************************************** --

-- *** Motor und ESC Temperaturen anzeigen ********************************************** --
local function drawTemp()
        -- Temperatur Motor --
        local temp_mot = getValue(telemetry_temp_mot)
-- Motortemperatur einlesen
        lcd.drawText(45, 24, "Motor" .. " " .. string.format("%.2d", temp_mot), SMLSIZE)        --
Motortemperatur ausgeben im Format xx

        -- Temperatur ESC --
        local temp_esc = getValue(telemetry_temp_esc)
-- ESC Temperatur einlesen
        lcd.drawText(51, 33, "ESC" .. " " .. string.format("%.2d", temp_esc), SMLSIZE)          --
ESC Temperatur ausgeben im Format xx
end
-- *** Motor und ESC Temperaturen anzeigen ********************************************** --

-- *** Differentiale anzeigen *********************************************************** --
local function drawDiff()
        local diff_fwd_stat = getValue(diff_fwd)
-- Status vorderes Differential einlesen
        local diff_rear_stat = getValue(diff_rear)
-- Status hinteres Differential einlesen
        if (diff_fwd_stat < 0 and diff_rear_stat < 0) then
-- wenn Wert für beide Differentiale <0 (offen), dann...
                lcd.drawPixmap(49, 45, "/SCRIPTS/TELEMETRY/GUDEN/diff_open.bmp")
-- Symbol für beide Differentiale offen anzeigen
        elseif (diff_fwd_stat >= 0 and diff_rear_stat < 0) then
-- wenn Wert für vorderes Differential >0 (geschlossen), dann...
                lcd.drawPixmap(49, 45, "/SCRIPTS/TELEMETRY/GUDEN/diff_fwd_closed.bmp")          --
Symbol für vorderes Differential geschlossen anzeigen
        elseif (diff_fwd_stat < 0 and diff_rear_stat >= 0) then
-- wenn Wert für hinteres Differential >0 (geschlossen), dann...
                lcd.drawPixmap(49, 45, "/SCRIPTS/TELEMETRY/GUDEN/diff_rear_closed.bmp")         --
Symbol für hinteres Differential geschlossen anzeigen
        elseif (diff_fwd_stat >= 0 and diff_rear_stat >= 0) then
-- wenn Werte für beide Differentiale >0 (geschlossen), dann...
                lcd.drawPixmap(49, 45, "/SCRIPTS/TELEMETRY/GUDEN/diff_closed.bmp")
-- Symbol für beide Differentiale geschlossen anzeigen
        else
-- sonst...
                lcd.drawText(50, 45, "ERR", DBLSIZE)
-- Text "ERR" anzeigen
        end
end
```

```
-- *** Differentiale anzeigen ************************************************************ --

-- *** Gang und Geschwindigkeit anzeigen ************************************************** --
local function drawGear()
        -- Gangschaltung --
        local gear_state = getValue(gear)
-- Schaltzustand der Gangschaltung (1/2) einlesen
        if (gear_state == -512) then
-- wenn Zustand -512 (2 Gang) entspricht, dann...
                lcd.drawText(88, 3, "Gear" .. " " .. "2nd", SMLSIZE)
-- Text "Gear 2nd" (2 Gang) ausgeben
        elseif (gear_state == 512) then
-- wenn Zustand 512 (1 Gang) entspricht, dann...
                lcd.drawText(88, 3, "Gear" .. " " .. "1st", SMLSIZE)
-- Text "Gear 1st" (1 Gang) ausgeben
        else
-- sonst..
                lcd.drawText(88, 3, "Gear" .. " " .. "???", SMLSIZE)
-- Text "Gear ???" (in Transit) ausgeben
        end

        -- Geschwindigkeit --
        local speed = getValue(telemetry_speed)
-- Geschwindigkeit einlesen
        speed = round(speed, 1)
-- Wert auf 1 Nachkommastellen runden
        lcd.drawText(87, 12, string.format("%.1f", speed) .. " " .. "km/h", SMLSIZE)    -- Wert
ausgeben im Format x.x km/h
end
-- *** Gang und Geschwindigkeit anzeigen ************************************************** --

-- *** Windenstatus und RSSI anzeigen ***************************************************** --
local function drawRssi()
        -- Zustand der Seilwinde --
        local winch_state = getValue(winch)
-- Zustand der Seilwinde (aus/ein) einlesen
        if (winch_state <= -350 or winch_state >= 350) then                              -- wenn
Wert <-350 oder >350 entspricht (Winde aus), dann...
                lcd.drawText(92, 24, "Wch off", SMLSIZE)                                    --
Text "Wch off" anzeigen
        else
-- sonst...
                lcd.drawText(92, 24, "Wch on", SMLSIZE)                                     --
Text "Wch on" anzeigen
        end

        -- RSSI --
        local rssi_value = getValue(telemetry_rssi)                                         --
RSSI Wert einlesen
        lcd.drawText(92, 33, "RSSI" .. " " .. rssi_value, SMLSIZE)              -- Text "RSSI" und
Wert anzeigen
end
-- *** Windenstatus und RSSI anzeigen ***************************************************** --

-- *** Sender- und BEC-Spannung anzeigen ************************************************** --
local function drawVoltage()
        -- Sender-Spannung --
        local voltage_transmitter = getValue(telemetry_transmitter)
-- Spannung des Senders einlesen
        voltage_transmitter = round(voltage_transmitter, 1)
-- auf 1 Nachkommastellen runden
        lcd.drawText(90, 46, "Send" .. " " .. string.format("%.1f", voltage_transmitter), SMLSIZE)
-- Spannung ausgeben im Format x.x

        -- BEC-Spannung --
        local voltage_bec = getValue(telemetry_bec)
-- Spannung des BEC einlesen
        voltage_bec = round(voltage_bec, 1)
-- auf 1 Nachkommastellen runden
```

```
        lcd.drawText(92, 55, "BEC" .. " " .. string.format("%.1f", voltage_bec), SMLSIZE)
-- Spannung ausgeben im Format x.x
end
-- *** Sender- und BEC-Spannung anzeigen ********************************************** --


-- *** Run-Funktion ******************************************************************* --
local function run(event)
        lcd.clear()                             -- Dispaly löschen

        drawClock()                             -- Uhr anzeigen
        drawTrip()                              -- Trip Timer anzeigen
        drawRun()                               -- Run Timer anzeigen

        drawCellVoltage()               -- Einzelzellspannungen max/min anzeigen
        drawTemp()                              -- Temperatur von Motor und ESC anzeigen
        drawDiff()                              -- Schaltzustand der Differentiale anzeigen

        drawGear()                              -- Ganganzeige und Geschwindigkeit anzeigen
        drawRssi()                              -- Windenstatus und RSSI Wert anzeigen
        drawVoltage()                   -- Spannug von Sender und BEC anzeigen

        drawGrid()                              -- Liniengitter darstellen
end
-- *** Run-Funktion ******************************************************************* --

return{run=run}
```