SHPI – Kurzanleitung

Einführung

Vielen Dank, dass Sie uns ihr Vertrauen geschenkt und sich für einen SHPI.zero oder One entschieden haben.

Der SHPI.zero und SHPI.one ist ein hochflexibles und multifunktionales Gerät. Wir sind uns sicher, dass Sie nach kurzer Einarbeitung, viel Freude mit dem Gerät haben werden. Die Regelungs- und Steuerungs- und Erweiterungsmöglichkeiten sind nahezu unbegrenzt.

Es wurde für den schnellen Einsatz als Drop-In-Replacement für ein Wandthermostat entworfen. Sie können mit jedem SHPI aber auch Rollladenschalter, Lichtschalter oder jedes andere Unterputzgerät ersetzen. In Kürze wird auch ein Tischstandgehäuse verfügbar sein.

Konzept

Der SHPI.zero und One verfügt über eine

- duale Prozessorhierarchie (außer Lite Version!)
- ein Sensornetzwerk
- ein berührungssensitives Display
- und modulares Netzteil mit integrierten Schaltrelais, sowie Stromsensor

Abgerundet wird das Gerät mit einer Vielzahl an Schnittstellen und der notwendigen Peripherie für die Sprachein- und -ausgabe.

Optional kann jedes SHPI mit einer Kamera (normales Raspberry Pi Zero Kameramodul) und flexiblen Zusatzmodulen ausgestattet werden.

Duale Prozessorhierachie (außer Lite!)

Der SHPI.zero und SHPI.one verfügt über einen Raspberry Pi Zero W und einen ATmega32u4 als sekundären Mikrocontroller.

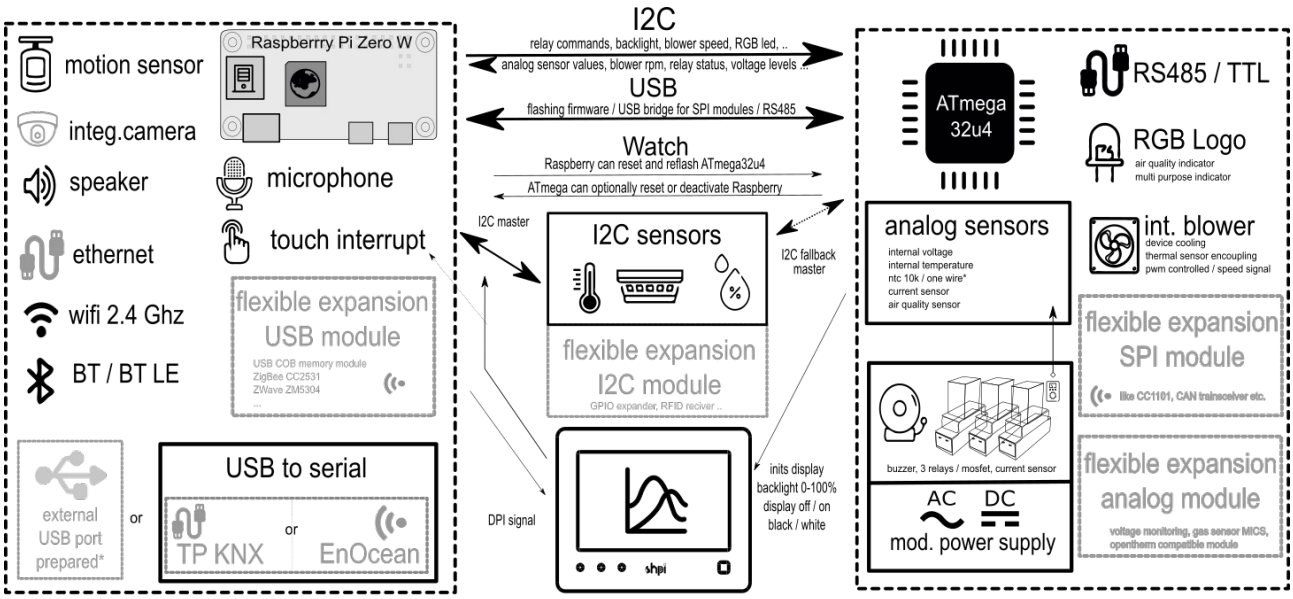
Der Raspberry stellt mit seinem ARM Prozessor und dem VideoCore IV genügend Rechenleistung für Multimedianwendungen zur Verfügung und der ATmega32u4 erweitert die Funktionalität um Analogeingänge, Echtzeitanwendungsmöglichkeiten und industrietaugliches Fallback-System.

Der ATmega überwacht den Raspberry optional und resettet ihn Fehlerfall*. Im permanenten Fehlerfall kann er als Fallbacksystem dienen und die Betriebssicherheit erhöhen.

Beide Prozessoren sind per I2C an die Sensoren angeschlossen. Standardmäßig ist der Raspberry I2C Master. Im Fehlerfall kann der ATmega auch Masterfunktionalität bereitstellen.*

* Noch nicht in Standardfirmware implementiert.

Beide Prozessoren sind per USB verbunden und die Firmware kann on-the-fly auf dem ATmega geändert werden.



Raspberry Pi Firmware installieren (fertiges Image)

Für Anfänger empfiehlt es sich unser fertiges Image zu nutzen, welches bereits unsere Standardanwendung installiert hat.

http://shpi.de/shpi-universal.img.gz

Aus Linux / MacOS:

Hierbei handelt es sich nur um ein Beispiel, bitte passen Sie die Befehle an ihre Pfade an und vergewissern Sie sich, dass Sie eventuell vorhanden Partitionen auf der SD-Karte unmounted haben.

```
gunzip shpi-universal.img.gz
```

```
dd if=shpi-universal.img of=/dev/mmcblk0 bs=4m oflag=direct
```

Ersetzen Sie /dev/mmcblk0 mit dem Pfad ihrer SD-Karte!

Aus Windows:

Wir empfehlen die die Programme „Win32DiskImager" oder „BalenaEchter".

W-Lan einrichten:

Bitte erstellen Sie die Datei „wpa_supplicant.conf" (Unix Format, UTF-8) in der Boot-Partition der SD-Karte mit ihren W-Lan Daten mit folgendem Inhalt:

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
     ssid="NETWORK-NAME"
     psk="NETWORK-PASSWORD"
}
```

Image aktualisieren:

Nach dem ersten Start des SHPIs empfiehlt es sich, sich per SSH auf dem Gerät einzuloggen **(Nutzername: pi, Passwort: raspberry)** und die Pakete zu aktualisieren.

```
sudo apt-get update

sudo apt-get upgrade
```

SHPI Software aktualisieren:

Unsere Software ist OpenSource und wird weiterentwickelt. Sie sollten daher regelmäßig unsere Software aktualisieren um von neuen Funktionen zu profitieren.

Gehen Sie hierfür in die entsprechenden Ordner und führen „git pull" aus.

```
cd old_main_application (oder zero_main_application)
git pull

cd ..

cd one_avr_firmware_std
git pull
```

Hinweis: Wie der Name es schon verrät wird in kürzerer Zeit die als Beispielprogramm (für Entwickler) gedachte **old_main_application** ersetzt. Die neue Anwendung ist auf den Endanwender zugeschnitten und wird wesentlich umfangreicher sein.

SHPI – Old main application konfigurieren (*zero_main_application)

Die alte Applikation wird über die Datei

```
/home/pi/old_main_application/shpi/config.py
```

konfiguriert. Öffnen Sie Datei über SSH mit nano oder vim.

```
nano /old_main_application/shpi/config.py
```

Die Relais konfigieren Sie mit folgenden Variablen:

```
COOLINGRELAY = 0  # off
HEATINGRELAY = 1  # on relay 1
SHUTTERDOWN = 2  # relay 2
SHUTTERUP = 3  # relay 3
```

0 bedeutet, dass kein Relais für die Kühlfunktion ausgewählt wurde.
1 Relais 1 ist als Heizungsrelais eingestellt (Heizen: ein)

Hinweis: Der Wetter-Slide benötigt eine **Internetverbindung** und muss konfiguriert werden. Registrieren Sie sich bitte hierfür bei OpenWeatherMap (https://openweathermap.org/) oder entfernen Sie den Slide in der Config.

```
OWMKEY = '20f7aab0a600927a8486b220200ee694'
OWMLANGUAGE = 'de'
OWMCITY = 'Berlin, DE'
```

Wetter-Slide entfernen:

```
slides = ['overview', 'dial_thermostat', 'weather', 'ical2', 'status',
'shutter','livegraph', 'amperemeter', 'rrdgraph', 'settings']
```

Slides hinzufügen:

Sie können weitere Slides hinzufügen, indem Sie in den Unterordner „slides" und „subslides" schauen und die verfügbaren Slides manuell den Arrays hinzufügen:

z.B. demo_colorpicker.py:

```
slides = ['overview', 'dial_thermostat', 'ical2', 'status',
'shutter','livegraph', 'amperemeter', 'rrdgraph', 'settings',‘demo_colorpicker‘]
```

Sie können sich den Aufbau der Slide-Dateien anschauen und eigene Slides hinzufügen. Slides werden automatisch dem Hauptprogramm hinzugefügt. Subslides hingegen sind nur über andere Hauptslides aufrufbar.

HTTP Zugriff

Ein Fernzugriff auf den SHPI ist mit der Old Main App per HTTP und MQTT möglich. Es lässt sich auf alle internen Variablen zugreifen und die Applikation fernsteuern.

HTTP Server aktivieren (config.py):

```
START_HTTP_SERVER = True
HTTP_PORT = 9000
```

Beispiele:

http://ipshpi:port/?relay1 -> Liest Relais 1 eins aus (an): relay1:1;
http://ipshpi:port/?relay2 -> Liest Relais 2 aus (aus): relay2:0;
http://ipshpi:port/?relay1=1 -> Schaltet Relais 1 ein: relay1:1;relay1>1;

Bitte beachten Sie, wenn Sie von extern einRelais steuern, dass Sie vorher die internen Funktionen deaktivieren (z.B.: heatingrelay = 0), sonst wird der Relaiswert nach einem bestimmten Zeitpunkt wieder überschrieben.

http://ipshpi:port/?led=255,255,255 -> led>['255', '255', '255'];

Schaltet LED auf weiß, maximale Helligkeit.

http://ipshpi:port/?screenshot -> Grafikausgabe

Zeigt aktuellen SHPI.One Bildschirm an.

http://ipshpi:port/?all -> Ausgabe aller Werte

Zeigt alle Werte aus dem eg_Object an.

MQTT Zugriff - Client aktivieren (config.py):

```
START_MQTT_CLIENT = True
MQTT_USER = 'USERNAME'
MQTT_PW = 'PASSWORD'
MQTT_SERVER = "mqtt.eclipse.org"
MQTT_PORT = 1883
MQTT_PATH = "shpi"
MQTT_QOS = 0
```

MQQT veröffentlichte Werte

atmega_volt, d13, hwb, a0, a1, a2, a3, a4, a5, a7, atmega_temp, vent_rpm, vent_pwm, atmega_ram, buzzer, relay1current, mlxamb, mlxobj, bmp280_temp, pressure, lightlevel, sht_temp, humidity, motion, set_temp, backlight_level, gputemp, cputemp, act_temp, useddisk, load, freespace, wifistrength, ipaddress, led_red, led_green, led_blue, ssid, uhrzeit, relay1, relay2, relay3, lastmotion, max_backlight, usertext, usertextshow, alert

MQTT Steuerkanäle unter /set

relay1, relay2, relay3, buzzer, d13, alert, max_backlight, set_temp, vent_pwm, led

ON | OFF	für relay1, relay2, relay3,buzzer, d13, alert
1 .. 31	für max_backlight
0.0 .. 88.5	für set_temp
0 .. 255	für vent_pwm
255,255,255	für led

Variablenerläuterung

atmega_volt	Eingangsspannung in mV	
d13	Wert D13 ATmega	Reset Option Raspberry
hwb	HWB ATmega	Luftgütesensorheizung an / aus
a0	A0 Analogeingang ATmega 0-1023	A0 an SPI-Modul, CC1101 GD0
a1	A1 Analogeingang ATmega 0-1023	A1 an Analogmodul
a2	A2 Analogeingang ATmega 0-1023	A2 an Analogmodul
a3	A3 Analogeingang ATmega 0-1023	A3 an Analogmodul
a4	A4 Analogeingang ATmega 0-1023	A4 = MP503 Luftgüte
a5	A5 Analogeingang ATmega 0-1023	A5, NTC 10K optional
a7	A7 Analogeingang ATmega 0-1023	ACS712 Stromsensor
atmega_temp	ATmega Chiptemperatur	
vent_rpm	aktuelle Lüfterdrehzahl	
atmega_ram	freier RAM ATmega	
relay1current	A7 AVERAGE Wert / Effektivwert	
vent_pwm	Lüfteranforderung	0: 100%, 254: auto min, 255: off
buzzer	Piezo Buzzer	
mlxamb	Interne Temperatur MLX 90615 Infrarotsensor	
mlxobj	Raumtemperatur MLX 90615 Infrarotsensor	
bmp280_temp	Temperatur BMP 280 in °C	
pressure	Luftdruck BMP 280 in hPa	
lightlevel	Lichtlevel BH1750 (ungenau)	
sht_temp	Temperatur SHT3x / AHT10 in °C	
humidity	Luftfeuchtigkeit in %	
motion	Bewegungsmelder	
set_temp	gewünsche Raumtemperatur	
backlight_level	Hintergrundbeleuchtung Status 0-31	
i2cerrorrate	I2C CRC Fehlerrate	
gputemp	Raspberry GPU Temperatur	
cputemp	Raspberry CPU Temperatur	
act_temp	**gemittelte Raumtemperatur**	
useddisk		
load	Linux Load value 1 = 100%	
freespace		
wifistrength	Wifi Signal in dbm	
ipaddress	aktuelle IP-Adresse	
led_red	Rotwert der LED	
led_green	Grünwert der LED	
led_blue	Blauwert der LED	
ssid	Wifi SSID	
tempoffset	aktueller Temperaturoffset Wochenprogramm	
tempoffsetstr		
uhrzeit	Uhrzeit	
relay1	Relais 1	
relay2	Relais 2	
relay3	Relais 3	
lastmotion	letzte Bewegung	
max_backlight	Hintergrundhelligkeit bei Auto	
alert	Alarmfunktion 1 oder 0	
led	LED Werte im Array [R,G,B]	

ATmega Fimware

Open Source: https://github.com/shpi/one_firmware_std
https://github.com/shpi/zero_avr_firmware_std

Der ATmega ist exklusiv mit folgender Peripherie verbunden:

- Backlight IC
- Display SPI Bus zur Initialisierung des Displays
- RS485 Transceiver
- RGB LED(s)

- interner Lüfter
- Stromsensor für Relais 1
- Relais 1, Relais 2, Relais 3
- Buzzer
- SPI-Modul z.B. CC1101 Sub 1-Ghz Transceiver
- Analog-Modul

Unsere Standardfirmware ermöglicht dem Raspberry PI die einfache Steuerung der Peripherie mittels I2C Befehlen. Der ATmega meldet sich auf dem I2C Bus mit der Adresse **0x2A** an.

Die Standardfirmware kann an individuelle Bedürfnisse on-the-fly angepasst werden. Clonen Sie hierzu das Git Verzeichnis:

```
git clone https://github.com/shpi/one_firmware_std
          https://github.com/shpi/zero_avr_firmware_std
```

Oder aktualisieren Sie ihr repository mit:

```
git pull
```

Modifizieren Sie die main.c nach ihren Anforderungen und installieren Sie die neue Firmware mit dem Befehl:

```
sudo make flash
```

Starten Sie danach den SHPI bitte neu.

CRC-Kontrolle ausschalten für Tests

Zur Erhöhung der Übertragunsicherheit ist eine CRC8 Kontrolle eingebaut, diese kann für manuelle Tests deaktiviert werden.

```
i2cset -y 2 0x2A 0xFE 0x00   (für main app wieder aktivieren mit 0x01)
```

Beispielbefehl: Relais 1 anschalten

```
i2cset -y 2 0x2A 0x8D 0xFF
```

Beispielbefehl: Relais 1 Status auslesen

```
i2cget -y 2 0x2A 0x0D
```

0x00 = Relais 1 aus, 0xFF = Relais 1 an

I2C Befehlsübersicht

Funktion / Pin	Leseadressen	Schreibadressen	Bytes	Interpretation	Werte
	0x01-0x7F	0x80-0xFF			
A0	0x00		2	10bit in 2 Byte, LOW→HIGH	0 – 1023
A1	0x01		2	10bit in 2 Byte, LOW→HIGH	0 – 1023
A2	0x02		2	10bit in 2 Byte, LOW→HIGH	0 – 1023
A3	0x03		2	10bit in 2 Byte, LOW→HIGH	0 – 1023
A4 (MP503, Luftgüte)	0x04		2	10bit in 2 Byte, LOW→HIGH	Sehr gut:0 ... Schlecht:1023
A5 (NTC 10K)	0x05		2	10bit in 2 Byte, LOW→HIGH	0 – 1023
A7 (ACS712, Stromsensor)	0x06		2	10bit in 2 Byte, LOW→HIGH	0 – 1023
Backlight Level	0x07	0x87	1	1 Byte	Aus: 0-31: heller
Lüfterdrehzahl (VENT_PWM)	0x08		2	2 Byte	UPM
Eingangsspannung ATmega	0x09		2	2 Byte	Spannung in mV
Interne ATmega Temperatur	0x0A		2	2 Byte	°C
FREERAM ATmega	0x0B		2	2 Byte	in Bytes
RGB LED Farbe / Helligkeit	0x0C	0x8C	3	3 Byte	R, G, B 0-254, 0-254, 0-254
Relais 1	0x0D	0x8D	1	1 Byte	0x00=off 0xFF=on
Relais 2	0x0E	0x8E	1	1 Byte	0x00=off 0xFF=on
Relais 3	0x0F	0x8F	1	1 Byte	0x00=off 0xFF=on
D13 (Reset RPI optional)	0x10	0x90	1	1 Byte	0x00=off 0x01=1sec for reset* 0XFF=on
HWB (Heizung MP503)	0x11	0x91	1	1 Byte	0x00=off 0xFF=on
Buzzer	0x12	0x92	1	1 Byte	0x00=off 0xFF=on Wird erweitert*
Lüfterdrehzahl Einstellung	0x13	0x93	1	1 Byte	0=max speed ... 253=minimum 254=autominimum 255=off
LED Rotwert	0x14	0x94	1	1 Byte	0 – 255
LED Grünwert	0x15	0x95	1	1 Byte	0 – 255
LED Blauwert	0x16	0x96	1	1 Byte	0 – 255
A7_AVG (AC curr)	0x17 (prev.:0x14)		2	2 Byte	0 – 1023
Display Controller	0x18	0x98	1	1 Byte	0x00=off 0xFF=on
Display weiß/schwarz	0x19	0x99	1	1 Byte	0x00=off 0x01=normal 0xFF=on
Watchdog	0x20	0xA0	1	1 Byte	0x00=off 0x01=only LED 0xF1=hard reset* 0xFF= hard reset with fallback*
Gewählte RGB LED	0x21	0xA1	1	1 Byte	0x00=LOGO 0x01=signal LED 0x02 … custom
DFU Bootlader		0xFD	1	1 Command	0xFF=DFU active
CRC Check	0x7E	0xFE	1	1 Byte	0x00=off 0xFF=on
FW_VERSION	0x7F		1	1 Byte	0xFF=Pre 2.0 0x01=2.0