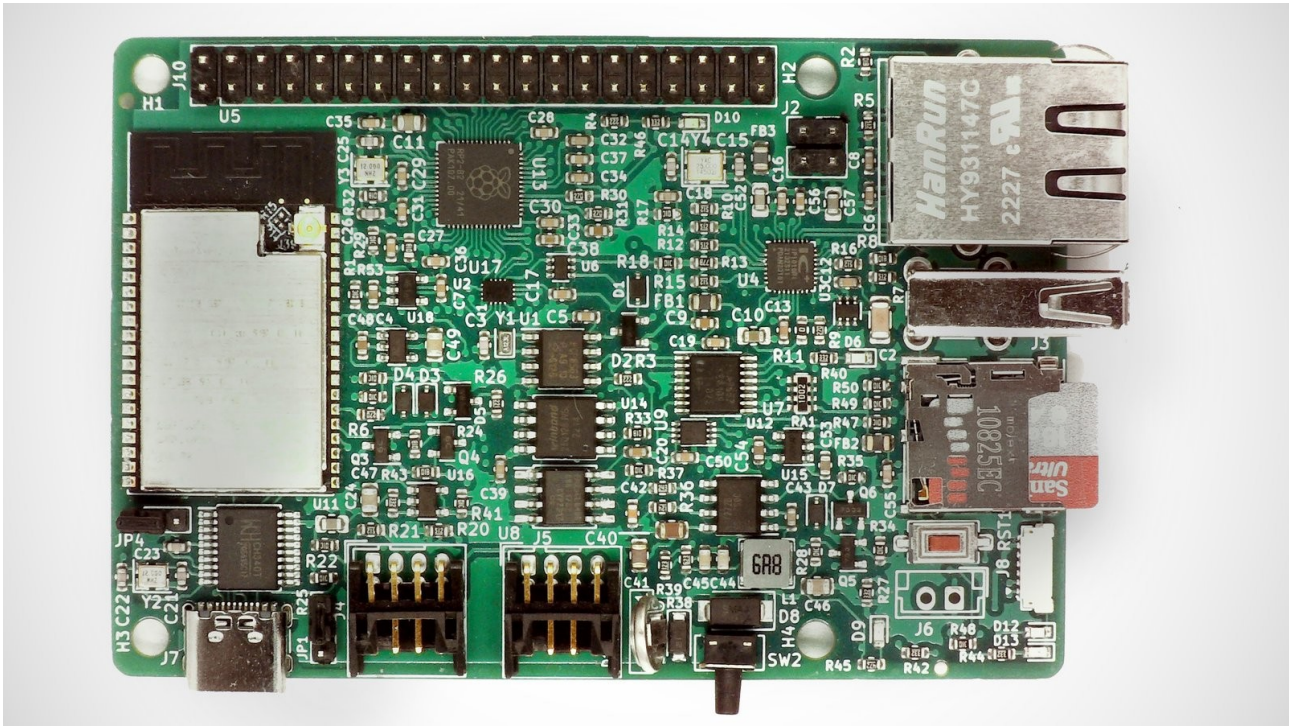
3Preliminary



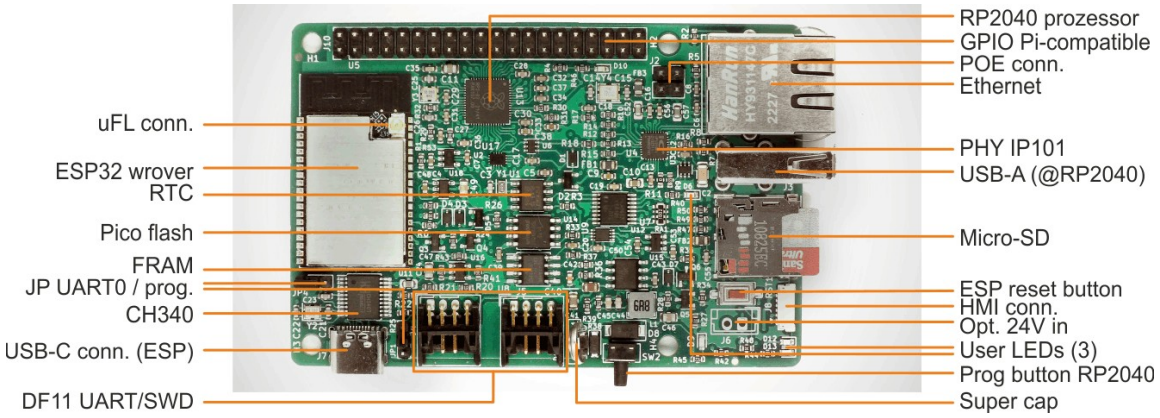
EsPiFF is an ESP32 in a Raspberry Pi 4 form factor, capable of utilizing nearly every Pi 4 enclosure and HAT. EsPiFF packs an additional punch with wired and wireless Ethernet, an SD card socket, and a RP2040 co-processor.

EsPiff is particularly well suited to measurement, control, and automation projects where the current consumption and heat generation of a Pi—or the potential instability of its SD card—could be problematic.

Features & Specifications

- ESP32-WROVER with 8 MB PSRAM and 16 MB Flash in a Raspberry Pi 4 form factor
- Wi-Fi connectivity (requires an external U.FL antenna)
- 10/100 wired Ethernet via IP101 PHY
- PoE header, to use Raspberry Pi PoE HATs. EsPiFF can be powered from a HAT or can power the HAT
- Micro SD card socket for storage
- Up to 3 UARTs
- USB Type-C connector on CH340 USB-UART for programming as well as power draw up to 5 V / 3 A for power-hungry HATs

- 40-pin Rasperry Pi header, compatible with all Raspberry Pi HATs
- RP2040 co-processor to emulate the Pi on the 40 pin connector, with 8 MB Flash
- External realtime clock, watchdog, and supervisor for high-availability, 24/7 applications
- On-board supercap to keep the realtime clock running for days, even without power. The supercap has, in contrast to a battery, a practically infinite lifetime
- USB-Host on the USB Type-A connector
- BOOT button for the RP2040, to switch between USB-Programming and USB-Host/Device
- ESP32 reset button and three user LEDs



Specifications	**Value**	**Remarks**
Dimensions	86 * 56mm	RasPi 4 compatible
Power supply	5V, up to 3A via USB 3 connector Optional 24V input via Pin Header	
Connectivity		
Ethernet	RJ45	
Wifi	External uFL antenna connector	PCB antenna not usable
Serial	2 DF11 connector with a total of 3 UART (TTL 3.3V)	Cable assemblies available (Mouser etc.)
Programming ESP32	USB-C (CH340 USB to UART)	
Programming RP2040	USB-A	
USB-Host/device	USB-A	
Raspberry Pi HAT	40 pol pin header	
UART TFT display (HMI)	6 pol JST	
ESP32-WROVER specs		
Dual core LX6 MCU	240 MHz	
Flash	16 MByte (W25Q128JVSIQ)	
PSRAM	8 MByte	

Pi Pico RP2040 specs		
Dual core M0+ MCU	133 MHz	
Flash	16 MByte	
ESP32 – RP2040 connectivity	SPI or UART	
Current consumption		
Wi-Fi Tx packet 13dBm~21dBm	170 to 295 mA	TBD
Wi-Fi/BT Tx packet 0dBm	140 mA	TBD
Wi-Fi/BT Rx and listening	85 to 115 mA	TBD
ESP32 modem sleep mode	3 to 28 mA	TBD
Sdcard	SPI mode	
FRAM	FM25CL64B – 1 MByte	
Supervisor	APX823	
Real time clock	PCF8563T	
Ethernet PHY	IP101	
PoE header	4 pol header	Identical to Pi 4
Supercap for real time clock	220mF	
Buttons		
PROG button for RP2040	On the position of the Pi audio connector	Can be operated when inside a Pi 4 enclosure
Reset button for the ESP32	On the position of the Pi USB	
Operating temperature	-20 to +70 degree Celsius	
Storage temperature	-40 to +85 degree Celsius	
Weight	37g	
CE/RoHS	compliant	
FCC	pending	

Pin outs

J4 – DF11

Pin number	Function	Remark
1	+5V	Max. 2.7 A, when USB powered
2	+5V	Max. 1.5A, when 24V powered
3	GND	
4	GND	
5	UART0 RXD	Labled HF_MISO in schematic
6	UART1 RXD	Labled TFT_MISO in schematic
7	UART0 TXD	Labled HF_MOSI in schematic
8	UART1 TXD	Labled TFT_MOSI in schematic

J5 – DF11

Pin number	Function	Remark
1	+5V	Max. 2.7 A, when USB powered
2	+5V	Max. 1.5A, when 24V powered
3	GND	
4	GND	
5	UART2 RXD	Labled UHF_MISO in schematic
6	Pico SWCLK	Debug clock pin for PicoProbe
7	UART2 TXD	Labled UHF_MOSI in schematic
8	Pico SWDIO	Debug data pin for PicoProbe

Cable assemblies for J4, J5: Hirose DF11-8DS-2C(17), Mouser No 798-DF11-8DS-2C17

J8 – SM06B-SURS-TF(LF)(SN)

Pin number	Function	Remark
1	UART1 TXD	Labled TFT_MOSI in schematic
2	UART1 RXD	Labled TFT_MISO in schematic
3	GND	
4, 5	nc	Not connected
6	+5V	

Cable assemblies for J8: A06SUR06SUR32Wxxx (replace xxx with the cable length in mm). For example, A06SUR06SUR32W305B, Digikey part number 455-2995-ND.

Jumper JP1 - U0TXD

1-2	CH340 to ESP32	position to program
2-3	DF11 connector to ESP	To access the 3th UART

Jumper JP4 - U0RXD

1-2	CH340 to ESP32	position to program
2-3	DF11 connector to ESP	To access the 3th UART

Pin mapping 40 pol Raspberry Pi HAT to RP2040 pins

HAT pin on 40 pol header	Pi HAT function	RP2040	Remarks
1	3.3V (from EsPiFF to HAT)	-	HAT can take up to 150 mA
2	5V (both directions possible)	-	HAT can take up to 2.7 A
3	Pi_GPIO2_SDA	IO10	
4	5V (both directions possible)	-	
5	Pi_GPIO3_SCL	IO11	
6	GND	-	
7	Pi_GPIO4_CLK0	IO23	
8	Pi_GPIO14_TXD	IO8	
9	GND	-	
10	Pi_GPIO15_RXD	IO9	
11	Pi_GPIO17	IO6	
12	Pi_GPIO18_PWM0	IO0	
13	Pi_GPIO27	IO22	
14	GND	-	
15	Pi_GPIO22	IO5	
16	Pi_GPIO23	IO7	
17	3.3V (from EsPiFF to HAT)	-	
18	Pi_GPIO24	IO20	
19	Pi_GPIO10_MOSI0	IO16	
20	GND	-	

21	Pi_GPIO9_MISO0	IO19
22	Pi_GPIO25	IO3
23	Pi_GPIO11_SCK0	IO18
24	Pi_GPIO8_nCS0	IO17
25	GND	-
26	Pi_GPIO7_nCS1	IO13
27	Pi_GPIO0_ID_SD	Not connected to RP2040 Connect to ESP32-I2C0
28	Pi_GPIO1_ID_SC	Not connected to RP2040 Connect to ESP32-I2C0
29	Pi_GPIO5_CLK1	IO24
30	GND	-
31	Pi_GPIO6_CLK2	IO25
32	Pi_GPIO12_PWM0	IO1
33	Pi_GPIO13_PWM1	IO2
34	GND	-
35	Pi_GPIO19_MISO1	IO15
36	Pi_GPIO16	IO4
37	Pi_GPIO26	IO21
38	Pi_GPIO20_MOSI1	IO12
39	GND	-
40	Pi_GPIO21_SCK1	IO14

The I2C lines to read out the HAT EEPROM is not connected to the RP2040, but instead to the ESP32-WROVER.
To expose the HAT to the ESP, a open source project called Configurable Firmata (https://github.com/firmata/ConfigurableFirmata) can be used. At the time of writing, Configurable Firmata just support UART as transport between the EsPiFF and the RP2040. In a future version, SPI will be supported as transport, additional to UART.

ESP32 – RP2040 interconnection

To establish a SPI connection between the ESP32 and the RP2040, the IO expander PCA9557 need to set the line nSPI_PICO to low, additional to the SPI_nCS0 lines. While reading/writing to the RP2040, the ESP32 can not use the SDcard.

ESP32-WROVER		RP2040	Remarks
GPIO14	SPI_SCK	IO26	
GPIO13	SPI_nCS0	IO29	
GPIO15	SPI_MOSI	IO28	
GPIO2	SPI_MISO	IO27	

ESP32 – FRAM

To establish a SPI connection between the ESP32 and the FRAM, the nSPI_CS1 line (ESP32-GPIO12) is to use. The nSPI_CS1 chip select line is exclusive for the FRAM.

ESP32 – SDcard

To establish a SPI connection between the ESP32 and the RP2040, the IO expander PCA9557 need to set the line mSPI_SDcard to low, additional to the SPI_nCS0 lines. While reading/writing to the Sdcard, the ESP32 can not communicate with the RP2040.

Software development

ESP32-WROVER

Important notice for Ubuntu/Linux users:

Ubuntu identify the CH340 USB to UART chip as braille device. To make the EsPiFF be recognized by Ubuntu need to disable the braille support.

systemctl stop brltty-udev.service
sudo systemctl mask brltty-udev.service
systemctl stop brltty.service
systemctl disable brltty.service

Espressif IDF

Please follow the instructions on https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html

Arduino

Please follow the instructions on https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/getting_started.html

uPython

Please follow the instructions on https://docs.micropython.org/en/latest/esp32/tutorial/intro.html

Javascript

For Espruino, please follow the instructions on https://www.espruino.com/ESP32. There are other Javascript interpreters for ESP32, you might check out:
- https://github.com/marcelkottmann/esp32-javascript
- https://github.com/Moddable-OpenSource/moddable

- https://www.neonious-iot.com/lowjs/
to name a few.

Rust

Please follow the instructions on https://github.com/espressif/rust-esp32-example

RP2040

Raspberry Pi Pico SDK

Please follow the instructions on https://github.com/raspberrypi/pico-sdk

Arduino – Pico

If you have the Arduino IDE installed, add the board support package for the Raspberry Pi Pico RP2040. At the time of writing, you have to add under settings, additional board URL,

https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

Then in boards, boards manager, select “Raspberry Pi Pico”. From here on, you can follow all tutorials from the Raspberry Pi Foundation for the Pi Pico and/or Arduino-Pico tutorials.

Version history

Document version	Author	Date
0.1	Michael Schmid	01.11.2022