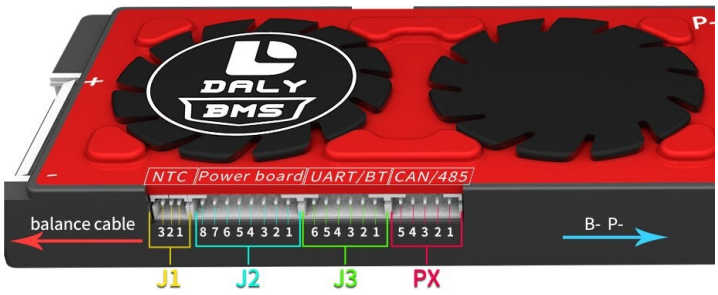
How to connect a DALY BMS via UART with a Raspberry Pi

Connecting a DALY BMS with a computer using the supplied USB-adapter is well known. But on a Rasperry Pi Zero W, there is just one USB interface and if power should be saved, why use the additional USB adapter when the RasPi has a native UART interface after all?
So the process is quite simple and straight forward:

1. Check the pinout of the DALY BMS UART connector
2. Understand what to do with it
3. Enable UART in the RasPi config and
4. Write a program to read out the BMS data

1. Check the pinout of the DALY BMS connector

A Smart DALY BMS comes with at least the DALY Bluetooth dongle. This Bluetooth interface interacts with the same interface that is being used by the DALY USB-UART interface – hence it has to speak UART.



DEFINITION OF SIGNAL INTERFACE OF ACQUISITION MODULE

J1 (NTC port) : 3PIN

1	2	3

1	NTC1	1#NTC
2	GND	GND
3	NTC2	2#NTC

J2 (External light board port) : 8PIN

1	2	3	4	5	6	7	8

1	GND	GND
2	POW_3.3V	Light board supply 3.3V
3	S1	Keyboard
4	LED_51	Fifth light
5	LED_41	4th light
6	LED_31	3rd light
7	LED_21	2nd light
8	LED_11	1st light

J3 (BT/UART) : 6PIN

1	2	3	4	5	6

1	GND	GND
2	POW_3.3V	Power supply 3.3V+
3	15V	Power supply 15V+
4	S1	Button activation
5	UART_TX2	Communication sender
6	UART_RX2	Communication receiver

PX (485/CAN port) : 5PIN

1	2	3	4	5

1	485_B	485 receiver
2	485_A	485 sender
3	GND	GND
4	CAN_H	CAN higher
5	CAN_L	CAN lower

*Above item based on 100A as an example

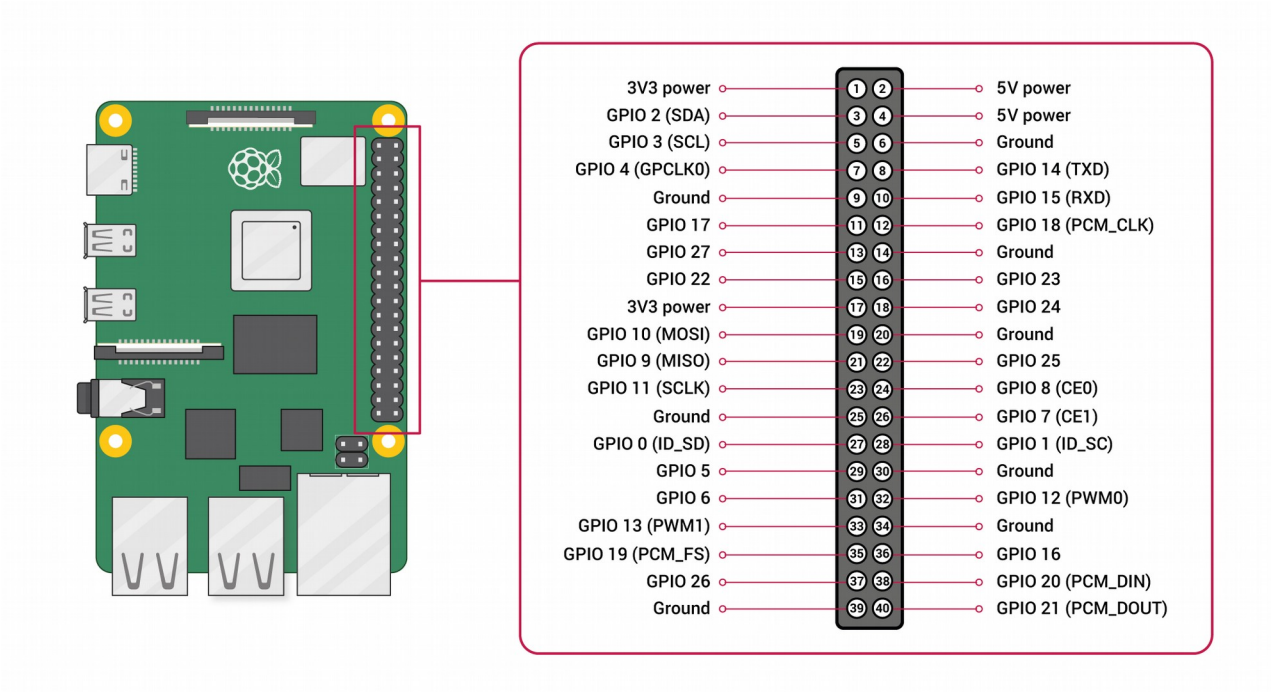
In this picture, the connector in question is J3 and for UART communication, there are three wires needed: GND (pin 1), UART_TX2 (pin 5) and UART_RX2 (pin 6). Also note POW_3.3V (pin 2) and S1 (pin 3) – we‘ll come to them later.

2. Understand what to do with it

Although it seems obvious, I rather mention this here. In any communication, one party transmits something that has to be received by the other party. So therefore the TX pin of one party needs to be connected to the RX pin of the other and vice versa:

	DALY	Raspberry Pi
Ground	GND – pin 1	Ground e.g. pin 9
Data 1	UART_TX2 – pin 5	RXD – pin 10
Data 2	UART_RX2 – pin 6	TXT – pin 8

A normal cable will do if it is not too long (in my case 2m were no problem). Just find a way to get the right receptacles to both sides of the cable ...



3. Enable UART in the RasPi config
By default, the UART interface on the Raspberry Pi is disabled. It needs to get enabled in the RasPi config. Although there might be a nice way on the GUI, when using a headless setup, there is only the command line interface. So here are the steps:

1. Open the config by typing `sudo raspi-config`
2. Scroll down `5 Interfacing Options`
3. Go to `P6 Serial`
4. Stay with **No** on `Would you like a login shell …`
5. Say **Yes** to `Would you like the serial port hardware to be enabled?`
6. The systems wants to reboot after this, so lets have it its way.

4. Write a program to read out the BMS data
Now the serial interface is enabled, it can get accessed by `/dev/serial0`. Of course, since the PC program supplied by DALY will not run on a Raspberry Pi, one has to come up with something DIY.
I use a python script to talk with the BMS. The protocol is well documented (can be found in this forum). There are no tricks involved, it works just as described.

5. Now to the other two pins
As mentioned, there is pin 2 called `Power Supply 3.3V` and pin 4 called `Button Activation`.
It is well known that the BMS only starts communication when charged (some models also wake up on discharge, but I‘m not sure on that). This entails also that pin 2, the 3.3V power supply is only part-time active (as the BT dongle shall go to sleep as well). This scraps the idea to use this pin to continuously power an ESP8266 for protocol translation and/or monitoring over larger distances.
However, for the Raspberry Pi being powered otherwise, pin 4, the `Button Activation` pin can be used to wake up the BMS and read out its values. All that needs to be done is to short that pin 4 shortly to ground. In my case, I use an opto coupler (want to be on the safe side) with the opto coupler output transistor‘s emitter connected to ground and collector connected to pin 4 on the BMS. The opto coupler input is connected via a resistor to one of the GPIO‘s (e.g. GPIO 4 which is pin 7 on the RasPI) and the other input pin connected to ground as well.

One could now argue, that when using an ESP8266, it can use pin 4 for continuously sending a „heartbeat“ to keep the BMS UART awake would work too. This is correct – however, when it fails for some reason, it is dead until the BMS wakes its UART again when recognizing charging.

So have fun with whatever you can take out of this!