VALTRACK-V4-VTS Documentation

> 📱 You should find majority of the details needed for using the hardware here

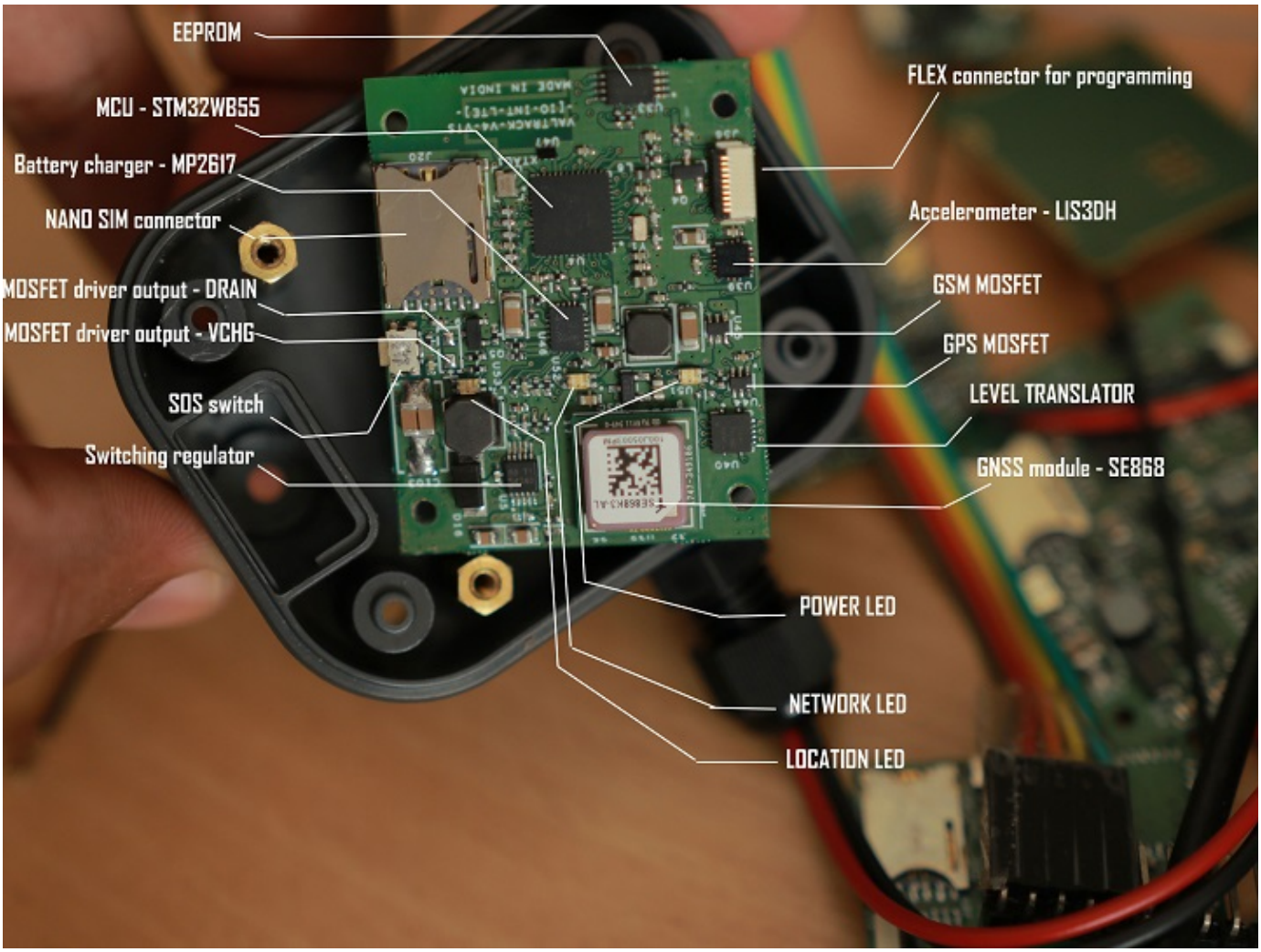
⚡ Introduction

VALTRACK-V4-VTS as the name itself depicts, is a fully configurable low power Vehicle Tracking System device. It was designed to be versatile and flexible enough to fit into any vehicle tracking scenario, be it for tracking 🚲 Bikes, 🚗 Cars or 🚚 Trucks.

Specifications

Model No.	VALTRACK-V4-VTS [IO-INT-LTE]
Operating Voltage	• Main Power Input : 12V to 42V DC [Connect to 12V Lead Acid Battery] Has Reverse Polarity Protection
	• Backup Battery Input : 3.7V to 4.2V DC [Connect a single cell 3.7V-4.2V Li-Po or Li-Ion Battery] Doesn't have Reverse Polarity Protection
Dimensions	• With Enclosure - Length: 52mm * Width: 65mm * Height: 30 mm
	• PCB Dimensions - Length: 34mm * Width: 43mm * Thickness: 1.6 mm
Battery Support	• 3.7V-4.2V DC Li-Po backup battery is supported. • Use > 400mAH capacity battery. • MP2617 Charger chip is used to handle the battery charging.
Cellular module	SIM7600G - LTE 3G 2G - For Global use
	SIM7600E - LTE 3G 2G - For Europe
	SIM7600A - LTE 3G 2G - For North America
Navigation hardware	• SE868K3AL GNSS module comes with inbuilt patch antenna
	• SIM7600x has inbuilt GNSS hardware which needs external active antenna [Bias is already provided to the GNSS U.FL connector]
Aux Inputs	None
Aux Outputs	1x Relay Output (MOSFET DRIVERS) [Optional]
Operating Modes	HTTP, SMS, MQTT/TCP
Configuration Methods	Bluetooth 5.0 available on board can be used to configure the device using the VALTRACK-V4 Setup application.
Processor	STM32WB55CEU6 ARM Cortex M4 , 64MHz, 512KB of Flash memory
Motion Sensor	MMA865x 12-bit, 3-axis Accelerometer
Memory	1-Mbit / 128KB EEPROM for storing parameters and lost pings.
Antenna	Cellular : U.FL Connector • 1.5 dBi gain Flexible PCB antenna comes attached
	SIM7600x GNSS : U.FL Connector • Patch antenna comes attached Available on request only !
	SE868K3L GNSS : Inbuilt antenna
Connectivity	Bluetooth, GPRS, SMS, Call
Flashing options	• SWD Debug port available on a FFC connector [10 pin, 0.5mm pitch]
	• MCU can be flashed Over The Air (OTA) using Bluetooth interface
Enclosure	Device ships in a standard IP67 rated enclosure.

Getting Started



Opening the enclosure

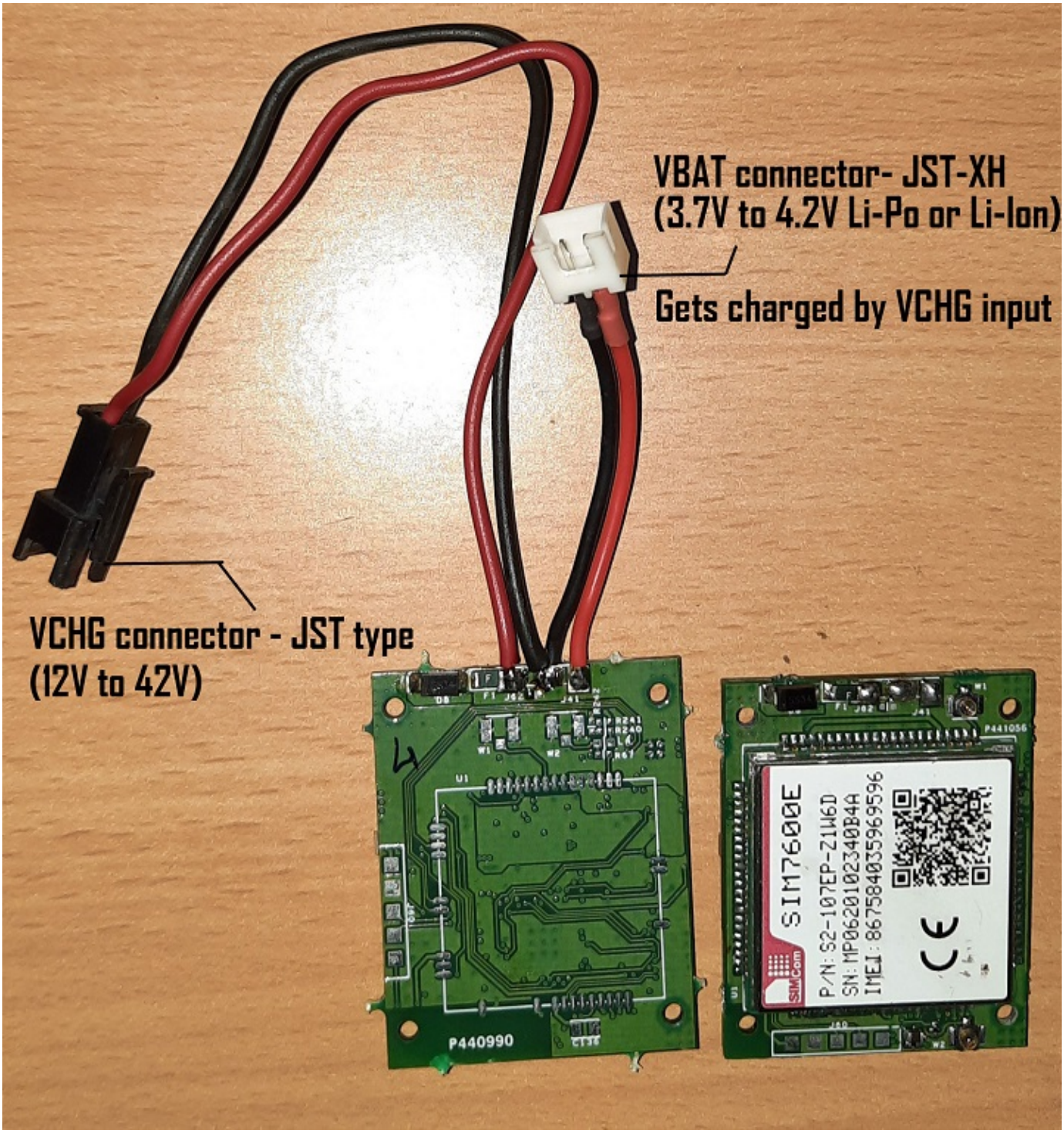
You need to remove the four screws present in the bottom of the enclosure to open it. Use a star head screw driver.

Inserting the SIM card

Get a Nano-SIM card and insert it into the boards push-pull type SIM card slot.

> ⚠ Power should not be connected to the device during SIM card insertion

Powering the device



VALTRACK-V4-VTS can run from any of the two power sources,

Main Power Input **[VCHG connector]**	• Can take 12V to 42V DC [Connect to 12V Lead Acid Battery] • Its a JST type connector Has Reverse Polarity Protection

Backup Battery Input **[VBAT connector]**	• Can take 3.7V to 4.2V DC [Connect a single cell 3.7V-4.2V Li-Po or Li-Ion Battery] • Use at least 500mAH and above capacity batteries • The battery connected to this port automatically gets charged by VCHG. • Its a JST-XH type connector ⚠ Doesn't have Reverse Polarity Protection

Device can start functioning with any of above power sources.

LED Indicators

- Once the device is powered ON, The LED will show up and start with all **RED** .
- Once the SIM is registered to the network, the NETWORK LED turns **GREEN** .
- Once the GNSS module gets a location sync, the LOCATION LED turns **GREEN** .
- After 30 seconds of inactivity, all LED will turn OFF to save power and turn ON again on movement detected by Accelerometer.
- POWER LED remains **RED** all the time

Programming

Here you will find the details needed for develop your own firmware for the device

If you are interested in writing firmware for the VALTRACK-V4-VTS device, you will need to know where is each pin of MCU is connected to.

Since the schematics of the device is not yet openly available, We are providing the MCU pin connection details, which should be able to help you in determining how is the whole architecture laid out. Watching our device intro video would also help to get an overall idea on the hardware present on board.

MCU Pinout Details

Pin Number	Pin Name	Net Name	Connected to
1	VBAT	3VDC	3VDC
2	PC14-OSC32_IN	CLK_IN	32.768 KHz crystal
3	PC15-OSC32_OUT	CLK_OUT	32K.768 KHz crystal
4	PH3-BOOT0	BOOT0	BOOT0 pull down resistor
5	PB8	NET_LED_R	Network - RED LED - Cathode pin
6	PB9	NET_LED_G	Network - GREEN LED - Cathode pin
7	NRST	RESET	Debug connector MCU Reset lines via RC network • J60 pin no 1 • J56 pin no 6
8	VDDA	3VDC	3VDC
9	PA0	SIM_PWRKEY_3V3	SIM7600x PWRKEY pin through N channel MOSFET. • Making this pin HIGH pulls PWRKEY pin to GND
10	PA1	DTR_3V3	SIM7600x DTR input pin through level translator.
11	PA2 / LPUART1_TX	LPUART1_TX	SIM7600x RXD input pin through level translator.
12	PA3 / LPUART1_RX	LPUART1_RX	SIM7600x TXD output pin through level translator.
13	PA4	ANALOG_IN	VCHG input through voltage divider resistor network. • R22,R33 govern the voltage at this pin. • Default values : R22 = 100K, R33 = 23.7K, effectively giving 2.87V for VCHG = 15V
14	PA5	GEN_LED_B	Location - BLUE LED - Cathode pin
15	PA6	NET_LED_B	Network - BLUE LED - Cathode pin
16	PA7	BAT_LED_R	Battery - RED LED - Cathode pin
17	PA8	RELAY	RELAY MOSFET driver Gate Input • Open drain driver with Drain pin of MOSFET exposed on a connector
18	PA9 / USART1_TX	UART_TX1	SE868K3AL RX0 input pin
19	PB2	GEN_LED_G	Location - GREEN LED - Cathode pin
20	VDD	3VDC	3VDC
21	RF1	RF1	Bluetooth PCB antenna via matching network
22	VSSRF	GND	System Ground
23	VDDRF	3VDC	3VDC
24	OSC_OUT	OSC_OUT	32 MHz crystal
25	OSC_IN	OSC_IN	32 MHz crystal
26	AT0	ATO	Not connected
27	AT1	AT2	Not connected

28	PB0	INT1	INT1 interrupt output of LIS3DH Accelerometer
29	PB1	GPS_ENABLE	Enable input of power gating MOSFET for SE868K3AL GNSS module • Making this pin high provides 3VDC to SE868KAL module
30	PE4	RELAY1	Not connected
31	VFBSMPS	VFBSMPS	3VDC
32	VSSSMPS	GND	System Ground
33	VLXSMPS	VLXSMPS	3VDC
34	VDDSMPS	VDDSMPS	3VDC
35	VDD	3VDC	3VDC
36	PA10 / USART1_RX	UART1_RX1	SE868K3AL TX0 output pin
37	PA11	GSM_ENABLE	Enable input of power gating MOSFET for SIM7600x LTE module • Making this pin high provides ~4VDC to SIM7600x module
38	PA12	GEN_LED_R	Location - RED LED - Cathode pin
39	PA13 / JTMS_SWDIO	SWDIO	Debug connector SWDIO lines • J60 pin no 3 • J56 pin no 8
40	VDDUSB	VDDUSB	3VDC
41	PA14 / JTMS_SWCLK	SWCLK	Debug connector SWDIO lines • J60 pin no 2 • J56 pin no 7
42	PA15	SOS	Tactile switch input • Pulled up, filtered and Active LOW
43	PB3	BAT_LED_B	Battery - BLUE LED - Cathode pin
44	PB4	TPS_ENABLE	Enable input of switching regulator TPS54240 • Making this pin high powers the system via VCHG connector. Not connected by default as it will cause system into reset loop if no alternate backup battery power available
45	PB5	BAT_LED_G	Battery - BLUE LED - Cathode pin
46	PB6 / I2C1_SCL	IIC_CLK	I2C clock of LIS3DH Accelerometer and M24M01 EEPROM
47	PB7 / I2C1_SDA	IIC_DATA	I2C data of LIS3DH Accelerometer and M24M01 EEPROM
48	VDD	3VDC	3VDC

J60 - MCU Debug Connector [SMT pads] - Pinout Details

Pin Number	Pin Name	Connected to
1	RESET	MCU Reset pin
2	SWCLK	MCU SWCLK pin
3	SWDIO	MCU SWDIO pin
4	GND	System Ground

5	VCC	3VDC

J56 - Flex Debug Connector [0.5mm 10 pin FFC] - Pinout Details

Pin Number	Pin Name	Connected to
1	SIM_USB_DN	SIM7600x USB_DN pin
2	SIM_USB_DP	SIM7600x USB_DP pin
3	SIM_USB_VBUS	SIM7600x USB_VBUS pin
4	UART_RX1	• SE868K3AL TX0 output pin • MCU USART1_RX pin
5	UART_TX1	• SE868K3AL RX0 input pin • MCU USART1_TX pin
6	RESET	MCU Reset pin
7	SWCLK	MCU SWCLK pin
8	SWDIO	MCU SWDIO pin
9	GND	System Ground
10	VCC	3VDC

J57 - SIM7600x USB Connector [SMT pads] - Pinout Details

Pin Number	Pin Name	Connected to
1	SIM_USB_DN	SIM7600x USB_DN pin
2	SIM_USB_DP	SIM7600x USB_DP pin
3	SIM_USB_VBUS	SIM7600x USB_VBUS pin
4	UART_RX1	• SE868K3AL TX0 output pin • MCU USART1_RX pin
5	UART_TX1	• SE868K3AL RX0 input pin • MCU USART1_TX pin

J62 - VCHG Connector [SMT pads] - Pinout Details

Pin Number	Pin Name	Connected to
1	VCHG	VCHG input of system through FUSE and diode • 12VDC to 42VDC input
2	GND	System Ground

J41 - VBAT Connector [SMT pads] - Pinout Details

Pin Number	Pin Name	Connected to
1	VBAT	VBAT input of system or Backup battery input • 3.7V to 4.2V battery input
2	GND	System Ground

Configuration

You will find information about parameter configuration by Bluetooth here

There is a mobile application presently only available for Android being developed, which supports updating of parameters using the on board Bluetooth 5 interface.

When you power on the device, it presents itself with the name **P2PSRV1**

The device runs a Custom P2P server Bluetooth profile code which exposes a few characteristics to be written to or read from to interact with the device.

Here are the Bluetooth interface details you need to be able to read and write from the device,

Service UUID : 0000fe40-cc7a-482a-984a-7f2ed5b3e58f

TX Characteristic : 0000fe41-8e22-4541-9d4c-21edae82ed19

Rx Characteristic : 0000fe42-8e22-4541-9d4c-21edae82ed19

Transmission and reception is from phones perspective

List of parameters supported :

Index	Command Name	Description	Size [Bytes]
0	Band	Network band to be selected, Best to leave default	30
1	Working Mode	Devices location sending mode HTTP / TCP / SMS	5
2	Motion Alert Mode	Alert CALL or SMS or NONE [Only in SMS Working mode]	5
3	Motion Threshold	Accelerometer threshold from 6 to 25	1
4	Contact Number	Contact number to be used for sending SMS or CALL	16
5	APN Name	Your network providers APN name	20
6	APN User Name	Your network providers APN user name if any	20
7	APN Password	Your network providers APN password if any	20
8	HTTP URL	URL of HTTP post request made in HTTP mode	150
9	HTTP Key	Any AUTH key of HTTP post request made in HTTP mode	100
A	Ping Interval	Location sending interval in seconds	4
B	MQTT Host	IP / Domain of MQTT broker in MQTT/TCP mode	30
C	MQTT Port	Port of MQTT broker accepting data in MQTT/TCP mode	10
D	MQTT Client ID	MQTT client ID of MQTT broker in MQTT/TCP mode	20
E	MQTT Topic	MQTT Topic of MQTT broker in MQTT/TCP mode	30
F	MQTT Protocol Name	Protocol name of MQTT broker in MQTT/TCP mode	10
G	MQTT LVL	LVL value of MQTT broker in MQTT/TCP mode	1
H	MQTT Flags	Flags used in MQTT packets in MQTT/TCP mode	1
I	MQTT Keep Alive	Keep alive interval for MQTT connection	4
J	MQTT User Name	MQTT authentication user name	30
K	MQTT Password	MQTT authentication password	35
Z	Return or Exit Bluetooth	Returns from the Bluetooth loop and restarts device	0

Writing new parameter values to the device :

When you want to update a parameters value, you need to write to the TX characteristic given above.

The format to write data is as follows,

```
InputData = '$VALETRON:' + InputIndex + '-' + $('#i'+InputID).val() + '#';
```

If you look at the above line, its a JavaScript line which forms the command to be sent to the device.

ex.,

if you want to update the Contact Number parameter to 1234567890, the command will become,

$VALETRON:4-1234567890#

Here the content between **-** (hyphen) and **#** (hash) characters which is **1234567890** will be written to the Contact Number parameter whose index is **4**.

“$VALETRON:” is the header and the **“#”** is like the footer which help the device to parse the command easily.

Once you have formed this command, you have to send the command, in a certain byte format to the device, Look at this code JavaScript code below,

```
for(var i=0;i<InputData.length;i++)
{
    data1[0] = 0x01; // Packet Identifer - Parameter Ppdate Packet
    data1[1] = InputData.charCodeAt(i);

    ble.writeWithoutResponse(
        deviceId,
        bluefruit.serviceUUID,
        bluefruit.txCharacteristic,
        data1.buffer, success, failure
    );
}
```

Here we are sending 0x01 as the first byte and our command byte as the second byte. Here 0x01 is a packet identifier that indicates to the device that the byte that follows is a parameter update data.

ex.,

Our data will be sent to device like this,

0x01, $

0x01, V

0x01, A etc

Reading values from the device :

When you want to read anything from the device, you subscribe to the RX characteristic given above.

Whenever a data is available, the phone is notified by Bluetooth.

When you want to manually read the parameters, Everything explained above holds good and you just need to replace the first byte, which is the packet identifier with `data1[0] = 0x02;` to indicate that its a parameter read command in below code snippet.

```
for(var i=0;i<InputData.length;i++)
{
    data1[0] = 0x02; // Packet Identifer - Parameter Read Packet
    data1[1] = InputData.charCodeAt(i);

    ble.writeWithoutResponse(
        deviceId,
        bluefruit.serviceUUID,
        bluefruit.txCharacteristic,
        data1.buffer, success, failure
    );
}
```

Here we are sending 0x02 as the first byte and our command byte as the second byte. Here 0x02 is a packet identifier that indicates to the device that the byte that follows is a parameter read data.

ex.,

Our data will be sent to device like this,

0x02, $

0x02, V

0x02, A etc

For example,

You can read the contact number parameter with **$VALETRON:4-000#** command.