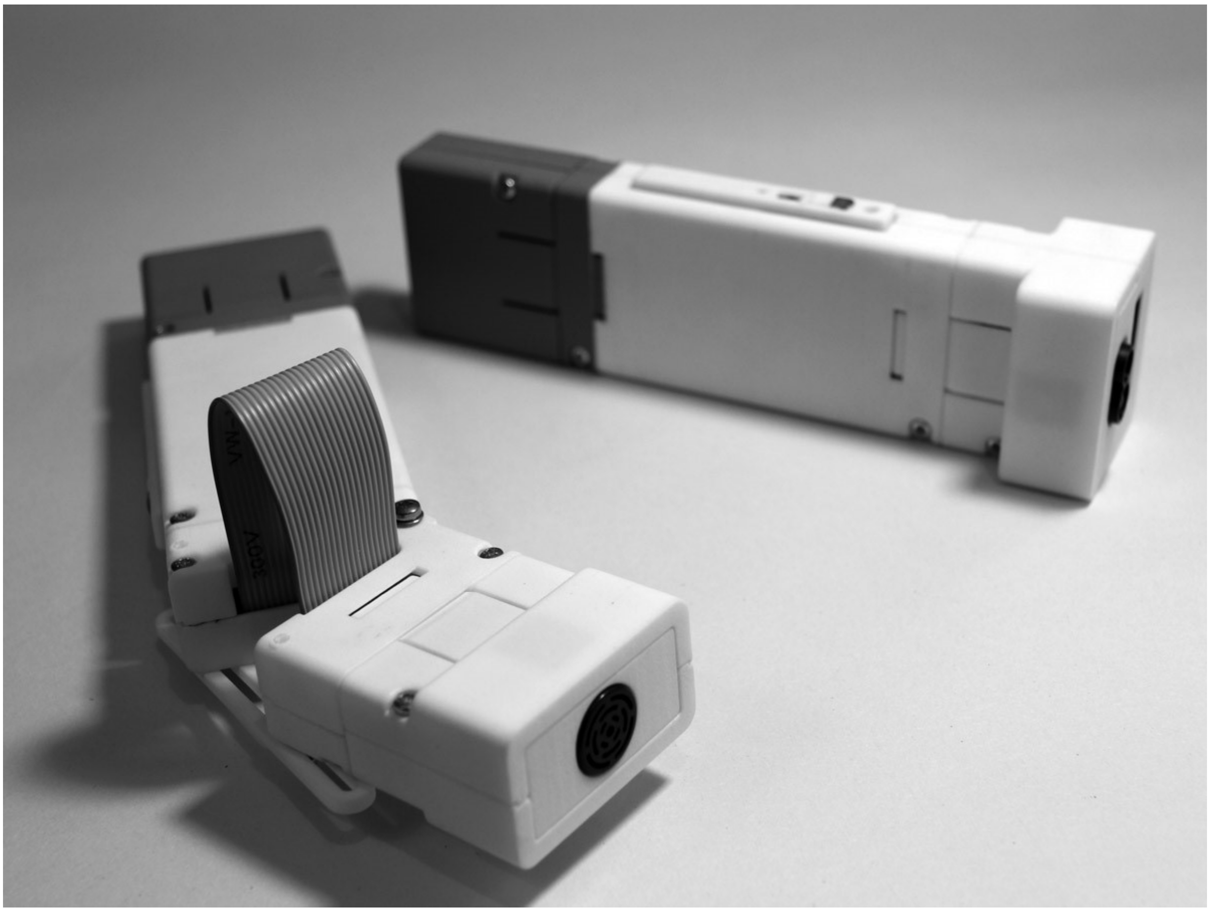
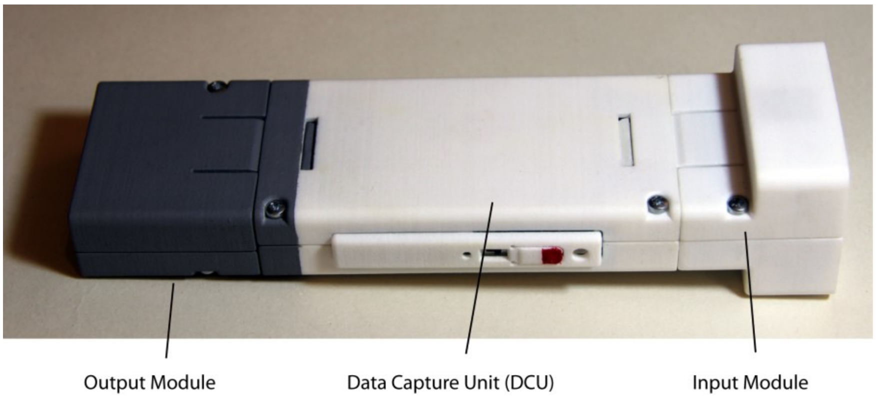
ENACTIVE TORCH RT 2 USER GUIDE



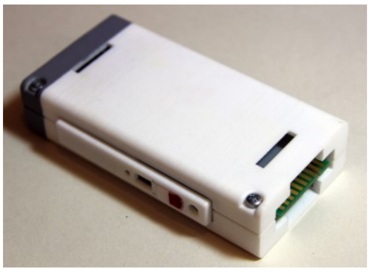
Contents

Introduction

The Enactive Torch RT 2 consists of three modules, a Data Capture Unit (DCU), input module and output module. The three modules clip together to form a working device.



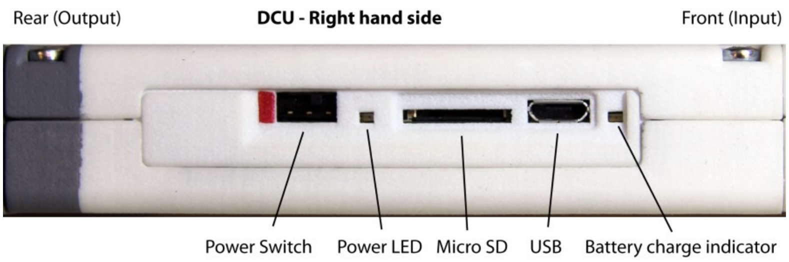
Data Capture Unit (DCU)

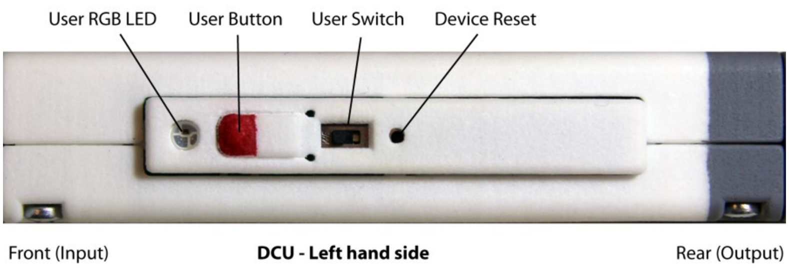


The Data Capture Unit is the core of the device. It contains the main processor, wifi module, inertial sensors and battery along with the user button and switch, USB port and micro SD card.

Input and output ports are located at each end of the DCU, one for a sensor input module and the other for the output module. The two ports use polarised connectors to prevent the modules being connected the wrong way around, and the case uses clip connectors to secure them in place. The two ends are also colour coded with the output module end coloured grey.

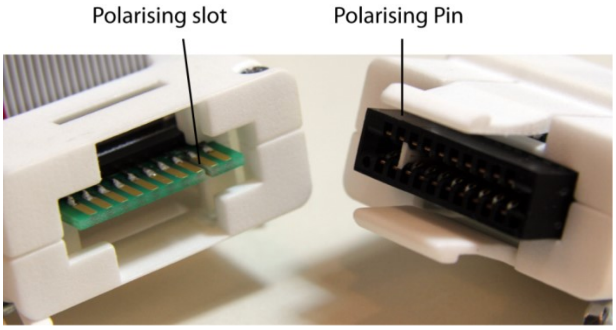
The USB port, power switch and SD card is located on one side of the DCU whilst the User LED, User button and switch and a device reset button are located on the opposite side.





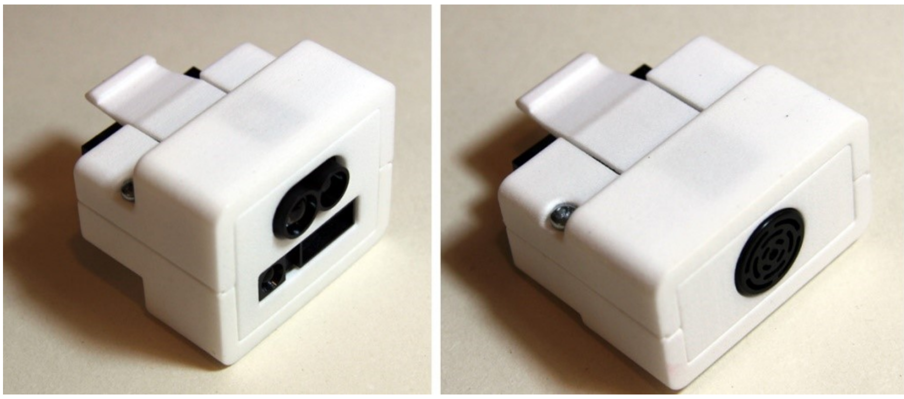
Clip connectors

The input and output modules connect to each end of the DCU. The connectors have white polarising pins that prevent them from being connected the wrong way.



Input Module

Input modules consist of a sensor or sensors that connect to the front of the device. A number of different input sensors can be connected to provide different functionality. More details can be found on page Sensor Modules9.



1: Two input modules - Dual LIDAR (Left) and Sonar (Right)

Output Module

The output module connects to the rear of the DCU and is used to produce output signals from the DCU, for example haptic feedback signals to drive different types of haptic actuator. More details can be found on page 10.

Basic operation

Startup procedure

When the DCU is switched on it will check for the presence of an SD card and, if found, it will then look for a file called settings.txt.

If the settings file is found it will open it and read the contents which should be a series of commands for configuring the device.

The command system

The DCU has a text based command system that can be used to configure the device. Commands can be sent over a USB connection, by Bluetooth and by WiFi using Telnet (Some of these functions are not working yet)

Documentation for the commands can be found on page 11.

The settings file can also be used to issue commands when the device is switched on and is the primary way to control the devices configuration. In particular it can be used to instruct the WiFi module to connect to a WiFi access point using a specified password.

Entering commands via the USB port.

To enter commands using the USB port, first connect the DCU to a computer using a USB cable. Connect to the DCU using a terminal emulator or serial port application. If the Arduino IDE is installed then the in built Serial monitor can be used. The USB serial port settings are:

- 115200 baud
- 1 Stop bit
- No parity

The DCU will appear as an “Adafruit Metro M4 (SAMD51)” USB device.

When the DCU is connected and switched on and the Serial monitor is opened up, type the command ‘get status’ and press enter to read the device configuration status. To get a list of commands type ‘help’ and press enter.

Software Installation

The DCU can be programmed using the Arduino development environment. The Arduino IDE is open source and free to download for PC, MAC and Linux. The DCU requires some additional files in order to work.

Installing Arduino

Download the version of Arduino for your computer from here:

https://www.arduino.cc/en/Main/Software

Follow the Arduino installation instructions for your system here:

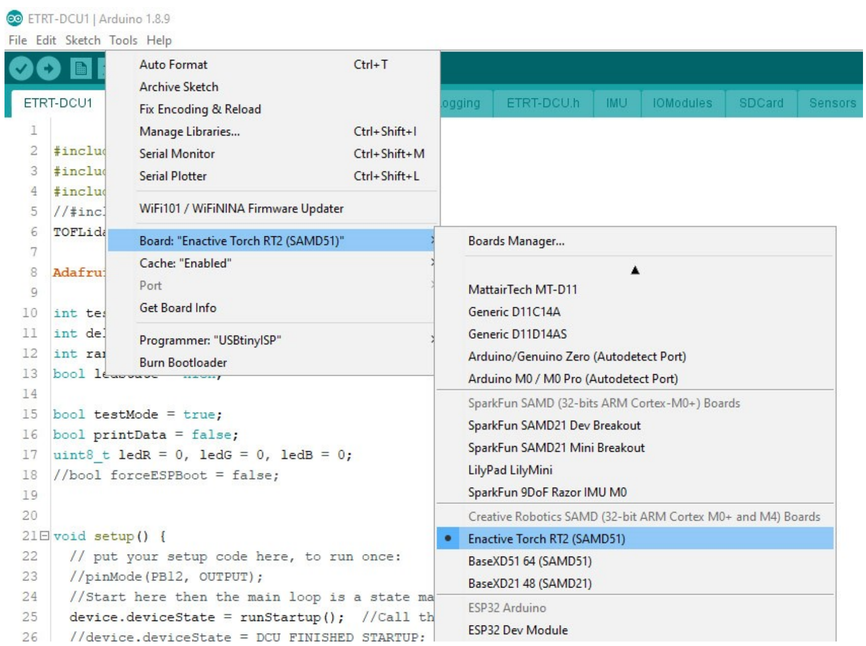
https://www.arduino.cc/en/Guide/HomePage

Installing the Creative Robotics Arduino core

The DCU works with a customised version of the Arduino software core which can be downloaded from here:

https://github.com/CreativeRobotics/ArduinoCore-samd/archive/master.zip

1. Unzip the contents of the file.
2. The unzipped folder will be called ‘ArduinoCore-samd-master’ – Rename this to ‘samd’.
3. Locate the Arduino Sketch folder on your computer
4. inside the sketch folder look for a folder called ‘hardware’ – if it does not exist then create it.
5. Inside the ‘hardware’ folder create another folder called ‘CreativeRobotics’.
6. Copy the ‘samd’ folder from step 2 into the ‘CreativeRobotics’ folder.



To check that the installation has worked properly start the Arduino IDE and look in the Tools->Board menu. You should see a list of boards under the heading “Creative Robotics SAMD (32-bit ARM Cortex M0+ and M4) Boards”

Select the “Enactive Torch RT2 (SAMD51)” option.

Installing the DCU firmware source code for Arduino

The full set of hardware and software for the ETRT2 including source code for the DCU firmware can be downloaded from here:

https://github.com/CreativeRobotics/EnactiveTorchRT2/archive/master.zip

To set up the software to work with Arduino first unzip the files and locate the 'ETRT-DCU1' folder inside 'Firmware->DCU' and copy this into your Arduino Sketchbook folder.

In order to compile the software you also need to install some third party libraries. This can be done with the Arduino Libraries manager.

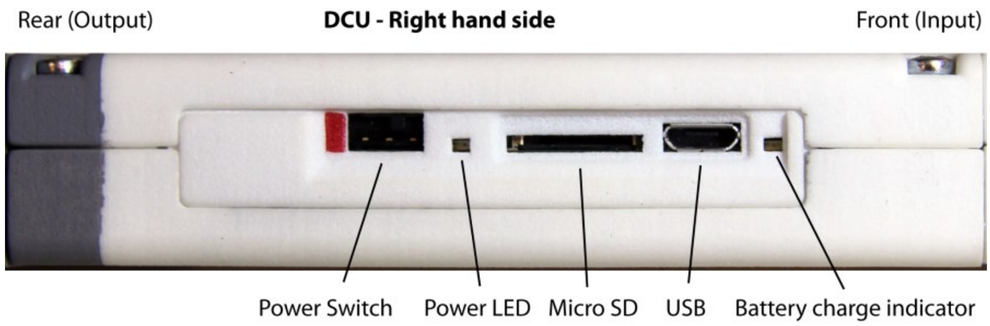
In the Arduino IDE open the 'Sketch->Include Library' menu and select 'Manage Libraries…'.

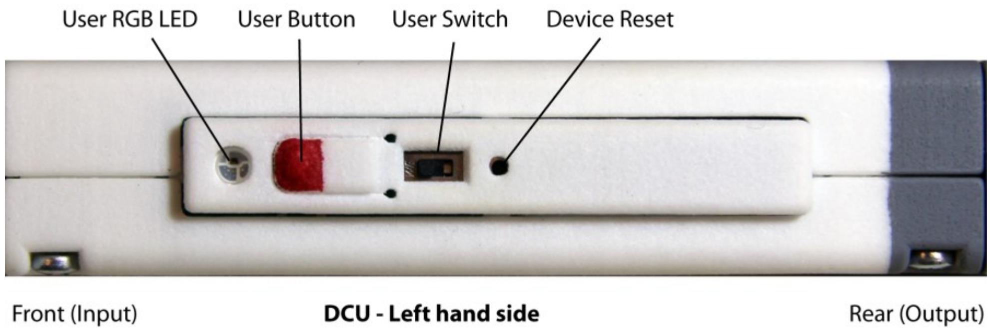
The Library manager will open and you can use the search option to find and install the following libraries:

- Adafruit ZeroDMA
- Adafruit NeoPixel ZeroDMA
- SparkFun MPU9250
- Bounce2

Alternatively these libraries can be copied from the 'Libraries' folder in zip file and placed in the libraries folder inside the Arduino sketchbook.

Data Capture Unit





The Data Capture Unit (DCU) is the core of the device, it provides power, USB and wireless connectivity and supports the Micro SD card reader. It also incorporates a 9 axis inertial measurement unit to track the orientation of the unit.

The DCU has two identical connectors at each end, one for attaching a distance sensing module, and the other for attaching a haptic driver module. With these modules attached, the DCU can read range data from the distance sensor module and translate it to haptic signals. It can also log this and other sensor data to the SD card and broadcast it over a Bluetooth connection, or over WiFI using the UDP data protocol.

The power switch ON position is where the switch is slit towards the rear of the device (the grey end).

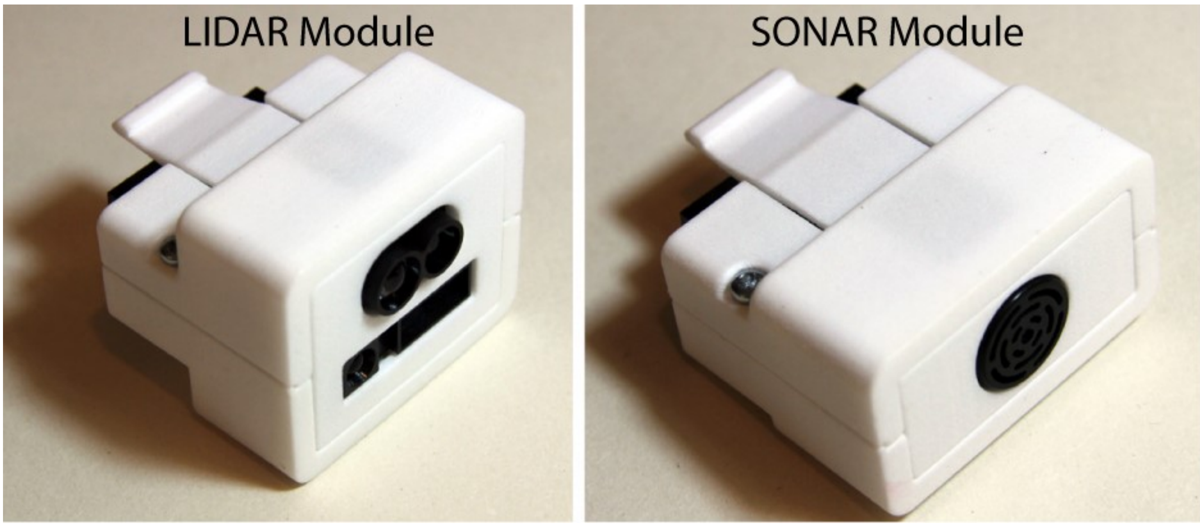
The user button activates the sensor to haptic mapping and will trigger data capture. The user switch controls whether data is just streamed over USB and wireless connections, or whether it is also logged to an SD card. The streaming options for the device are controlled using the startup configuration which the device reads from the ‘settings.txt’ file on the SD card.

When the user switch is in the OFF position (towards the rear) and the user button is pressed, the DCU will stream data over USB and any wireless connection it has been configured to use.

When the switch is ON (toward the front) the DCU will create an SD log file whenever the user button is pressed, and log data to this file for as long as the user keeps the button pressed. When the user releases the button the log file is closed. The DCU will also stream this data over any connection it has been configured to use.

Sensor Modules

There are currently two sensor modules that can be used with the DCU. The DCU will automatically detect the type of sensor that is attached to it when powered up.



Lidar

The Lidar module uses a time of flight lidar sensor that measures distance by sensing out a pulse of infra-red light and measuring the time it takes to receive a return signal. This sensor can measure distances between 30cm and 1200cm with a resolution of 1cm and a frequency of 100Hz.

The module also incorporates a second infra-red rangefinder that can measure distances from 4cm to 30cm.

Details of the Lidar and IR rangefinder cab be found here:

https://www.seeedstudio.com/Seeedstudio-Grove-TF-Mini-LiDAR.html

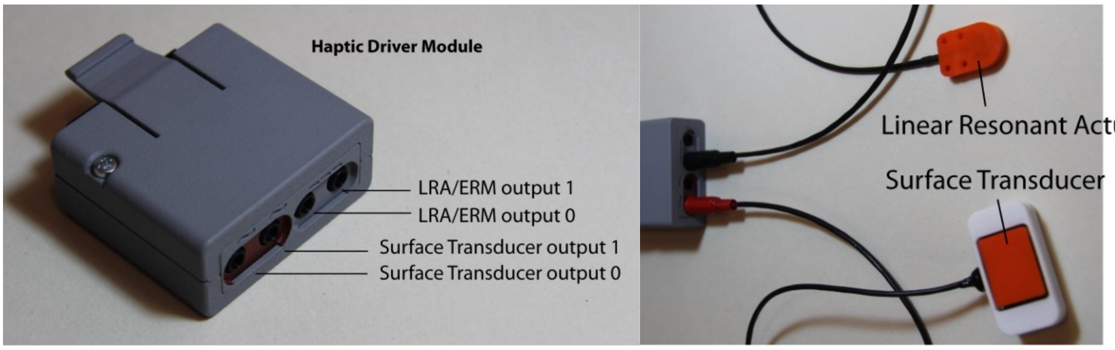
http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a41sk_e.pdf

Sonar

The sonar module uses ultrasonic sound to measure distance. The sensor is an MB1010 LV-MaxSonar-EZ1 sensor from Maxbotics. It can measure distances between 15cm and 645cm with a resolution of 2.54cm and a frequency of 20Hz.

https://www.maxbotix.com/documents/MB1010_Datasheet.pdf

Haptic Modules



ERM/LRA and Surface Transducer module

This module combines two types of haptic driver, each with two channels.

ERM/LRA drivers

The ERM/LRA drivers control conventional vibration motors and can be configured in software for two modes of operation:

In LRA mode it will drive Linear Resonant Actuators, these are a type of vibration motor containing a magnetic coil, mass and spring, and will resonate at a certain frequency when driven correctly and allow the intensity to be controlled. The LRA actuators supplied with the module will resonate at approximately 200Hz.

In ERM mode the device can drive Eccentric Rotating Mass actuators. These are conventional miniature motors with an eccentric mass attached to their output shaft which causes them to vibrate when the motor is spinning. The speed of the motor, and consequently the intensity of the vibration, can be controlled in software. With these actuators the vibration frequency and intensity are coupled together so increasing intensity also increases frequency.

Surface Transducer drivers

The surface Transducer drivers produce class B audio signals using a 2.5 audio Watt amplifier. They can be used to drive surface transducers. These are a type of actuator consisting of a magnetic coil and magnet and operate in exactly the same way as audio loudspeakers.

Surface transducers allow for independent control over frequency and intensity so it is possible to map two different sensor signals into their haptic output.

The default mode of operation for the surface transducers is conventional sensor to intensity mapping where the actuators are driven by a sine wave of varying intensity. A number of different signals can be chosen for driving the module this way, for example triangle waves and saw tooth waves.

Configuring the DCU

The DCU has a set of commands that can be used to control configuration. These Commands can be accessed using the USB port and a Serial Terminal application, or through a wireless connection such as Bluetooth or Telnet.

The settings file

The default configuration when the device starts up can be controlled with the same command set, but with a file on the SD card. When the DCU is switched on it will look for an SD card and for a file called 'settings.txt'. If it finds this file it will read and process any commands in the file.

The settings file MUST include a blank line at the end of the file.

A full list of commands can be found in the next section.

Example settings file

```
#ETRT SETTINGS
#DCU SETTINGS
#Lines starting with HASH (#) are ignored
DEBUG:if the settings file is empty thats fine
#Data log settings - what to log
setlog quaternion false
setlog YPR true
setlog gyro false
setlog accellerometer false
setlog magnetometer false
setlog heading false
setlog inputs true
setlog outputs true
#Data log settings - where to log
setlog USB false
setlog UDP false
setlog SD false
setlog bluetooth true
#Set biases for IMU
#set gyro bias 0,0,0
#set accel bias 0,0,0
#set mag bias 0,0,0
#use for configuring wifi
#set SSID myssid
#set set Pass mypassword
enable bluetooth
```

Anything preceded by the # symbol will be ignored so this symbol can be used to insert comments or disable commands. The command DEBUG can be used to print messages – anything after a DEBUG command will be printed over the USB port.

The first group of settings all use the 'setlog' command to configure which different parameters will be logged to file, and streamed over any data connections.

The second set of 'setlog' commands will enable or disable different channels for data logging. I the example only Bluetooth is enabled by default – but during operation logging to SD card can be switched ON or OFF with the user switch.

The next set of commands set bias values for the inertial measurement unit. These will normally be unique to each device and can be obtained by performing a calibration run (To Be Done)

The next set of commands set the network name (SSID) and password for WiFi conections. These are only used if the WiFi service is subsequently enabled. The command to enable WiFi must be used AFTER the password and SSID commands.

The final command enables the communications module as a Bluetooth device. This can be changed to enable WiFi and one of the WiFi services if required.

The DCU can use Bluetooth and WiFi simultaneously but the connections may become unreliable so it is best to use only one of these services.

DCU Command List

Note: This is a list of all command words that the device recognises. Some are only used for device to device communication, for example the ack and nack commands are simple acknowledge messages, and the DEBUG: command marks a command as containing debug information which it should simply ignore.

get status

Arguments: none

Returns:

get device status information

---

set time

Arguments: HH:MM:SS

Returns:

set the time. Syntax: set time [HH]:[MM]:[SS] – Hours, Minutes and Seconds separated with a colon.

---

get time

Arguments: none

Returns:

get the device time

---

set date

Arguments: DD:MM:YYYY

Returns:

Set the date Syntax: set time [DD]:[MM]:[YYYY]

---

get date

Arguments: none

Returns: none

Get the device date

---

get raw voltage

Arguments: none

Returns:

Get the device raw voltage reading

---

get settings

Arguments: none

Returns:

Print out the device settings

---

set precision

Arguments: integer - for example 4 (4 decimal places)

Returns:

Sets the number of decimal places for data logging. Example: set precision 5

---

set aux power

Arguments: on / off

Returns:

Turn the auxiliary 5V power regulator on or off. Syntax: set aux power on

---

sleep

Arguments: none

Returns: none

Puts the device to sleep – COMMAND NOT FUNCTIONAL

---

restart

Arguments: none

Returns: none

restarts the DCU

---

set SSID

Arguments: SSID name

Returns:

Sets the SSID for WiFI. Syntax: set SSID [WIFI NAME] //setSSID0=

---

set Pass

Arguments: Network Password

Returns:

Sets the password for WiFI. Syntax: set Pass [WIFI PASS] //setWiFiPass0=

---

enable server

Arguments: none

Returns:

Starts the HTTP server

---

disable server

Arguments: none

Returns:

Stope the HTTP server

---

enable telnet

Arguments: none

Returns: none

Starts the Telnet service

---

disable telnet

Arguments: none

Returns: none

Stops the Telnet service

---

enable bluetooth

Arguments: none

Returns: none

Starts the Bluetooth service

---

disable bluetooth

Arguments: none

Returns: none

Stops the Bluetooth service

---

enable UDP

Arguments: none

Returns: none

Starts the UDP service

---

disable UDP

Arguments: none

Returns: none

Starts the UDP service

startlog

Arguments: none

Returns: none

Starts logging data //start Logging Data

stoplog

Arguments: none

Returns: none

Stops logging data //stop Logging Data

setlog

Arguments: setlog sub command + true/false

Returns: none

Configure which types of data are to be logged and where to send it (SD, USB, UDP, Bluetooth).

Examples:

setlog quaternion true

setlog USB true

Sub command list:

quaternion	4 axis quaternion IMU data
YPR	Yaw Pitch and Roll data
gyro	Gyro XYZ data
accel	Accelerometer XYZ data
mag	Magnetometer XYZ data
heading	Heading (compass) data
inputs	Device sensor inputs
outputs	Device haptic outputs

SD	Enable/Disable SD card logging
UDP	Enable/Disable UDP data streaming
Bluetooth	Enable/Disable Bluetooth data streaming
USB	Enable/Disable USB data streaming

---

get log header

Arguments:

Returns:

Gets the header for the log file with names for each column of data.

Dir

Arguments: none

Returns: none

Print the filenames of files on the SD Card //print the SD directory

---

start haptest

Arguments: none

Returns: none

Start testing the haptics //start Logging Data

---

stop haptest

Arguments: none

Returns: none

Stop testing the haptics //stop Logging Data

---

set waveform

Arguments: Chanel WAVEFORM

Returns: none

Set the waveform type.

Example: set waveform 0 SINE

Sets the waveform for channel 0 to a sine wave.

Waveform arguments:

SINE	Sine wave
SQUARE	Square wave
TRIANGLE	Triangle wave
SAWF	Sawtooth (forwards)
SAWB	Sawtooth (reverse)
ONESHOT	(Single pulse sequence)
HALFSHOT	(Half pulse sequence)

---

set gyro bias

Arguments: x,y,z

Returns: none

Set the Gyro bias values with three comma seperated integers. Syntax:set gyro bias [x],[y],[z]

---

set accel bias

Arguments: x,y,z

Returns: none

Set the Accelerometer bias values with three comma seperated integers. Syntax:set gyro bias [x],[y],[z]

---

set mag bias

Arguments: x,y,z

Returns: none

Set the magnetometer bias values with three comma seperated integers. Syntax:set gyro bias [x],[y],[z]

ESP Boot

Arguments: none

Returns: none

Set the ESP32 to bootloader mode //go into bootloader and disable UART

ESP RESET

Arguments: none

Returns: none

Reset the ESP32 //restart ESP32 and UART

ESP Status:

Arguments: none

Returns: none

An ESP32 WiFi module Status message //Status message from ESP

ESP Get:

Arguments: none

Returns: none

Marks HTML data sent to the WiFI module for it to sent to a client //get an HTTP page

---

ESPTelnet:

Arguments: none

Returns: none

Marks data as coming from a Telnet client //

ESP:

Arguments: none

Returns: none

Route the command to the ESP32 WiFi Module //route a command to the ESP module

---

testSD

Arguments: none

Returns: none

Test the SD by opening writing and closing a file

---

button action

Arguments: none

Returns: none

Enable or disable various button actions

DEBUG:

Arguments: none

Returns: none

Debug message

---

ack

Arguments: none

Returns: none

Acknowledge

---

nack

Arguments: none

Returns: none

NOT Acknowledge

---

toggle print

Arguments: none

Returns: none

Toggle Printing (DEBUGGING)

?

Arguments: none

Returns: none

Query

---

help

Arguments: none

Returns: none

Get Help

ESP32 command list

Command list for the ESP32 WiFi Module. These commands are used by the WiFi/Bluetooth module and are used by the DCU to control the WiFi module..

?

Arguments: None

Query – Returns ack.

Help

Arguments: None

Returns command list.

Ack

Arguments: None

Send an acknowledge.

Nack

Arguments: None

Send a NOT acknowledge.

---

enable server

Arguments: None

Enable the HTTP Server.

---

disable server

Arguments: None

Disable the HTTP Server.

---

enable Bluetooth

Arguments: None

Enable the Bluetooth Serial service.

---

disable Bluetooth

Arguments: None

Disable the Bluetooth Serial service.

---

enable UDP

Arguments: None

Enable the UDP broadcast service.

---

disable UDP

Arguments: None

Disable the UDP broadcast service.

UDPSend:

Arguments: UDP data packet

Send a packet of data over the UDP service.

Example: *UDPSend:Sensor=123,Gyro=992*

---

serverSend:

Arguments: HTML line

Send a line of data for the HTTP server to send to a client.

Example: *serverSend:<p>some HTML</p>*

BTSend:

Arguments: Bluetooth serial data

Send a line of data for transmission via Bluetooth.

Example: *BTSend:Sensor=123,Gyro=992*

---

serverEnd

Arguments: None

Tell the server to close the connection to the client – When everything that needs to be sent has been sent.

---

set UDPPort

Arguments: UDP Port number

Sets the UDP Port number.

Example: *set UDPPort 6060*

---

set UDPAddress

Arguments: UDP network address

Sets the network address for UDP data. Default is broadcast address.

Example: *set UDPAddress 123.456.789.101*

---

set SSID

Arguments: SSID (Network Name)

Sets the WiFi network name to connect to.

Example: *set SSID VM12345*

---

set Pass

Arguments: WiFi Password

Sets the WiFi password.

Example: *set Pass mypassw0rd*

---

set Server Timeout

Arguments: time in milliseconds

Sets the timeout period in milliseconds after which the HTTP server will close a client connection.

---

set BTName

Arguments: Bluetooth device name.

Sets the Bluetooth device name.

Example: *set BTName ETNumber2*

---

connect to

Arguments: SSID and Password (Separated by one space)

Attempts to connect to a WiFi network using the SSID and Password in the arguments.

Example: *connect to VM12345 mypassw0rd*

---

set WiFi Timeout

Arguments: Time in milliseconds

Sets the timeout period after which an attempt to connect to WiFi is abandoned. Minimum is 1000 (1 Second).

Example: *set WiFi Timeout 10000*

Disconnect

Arguments: None

Disconnects from WiFi.