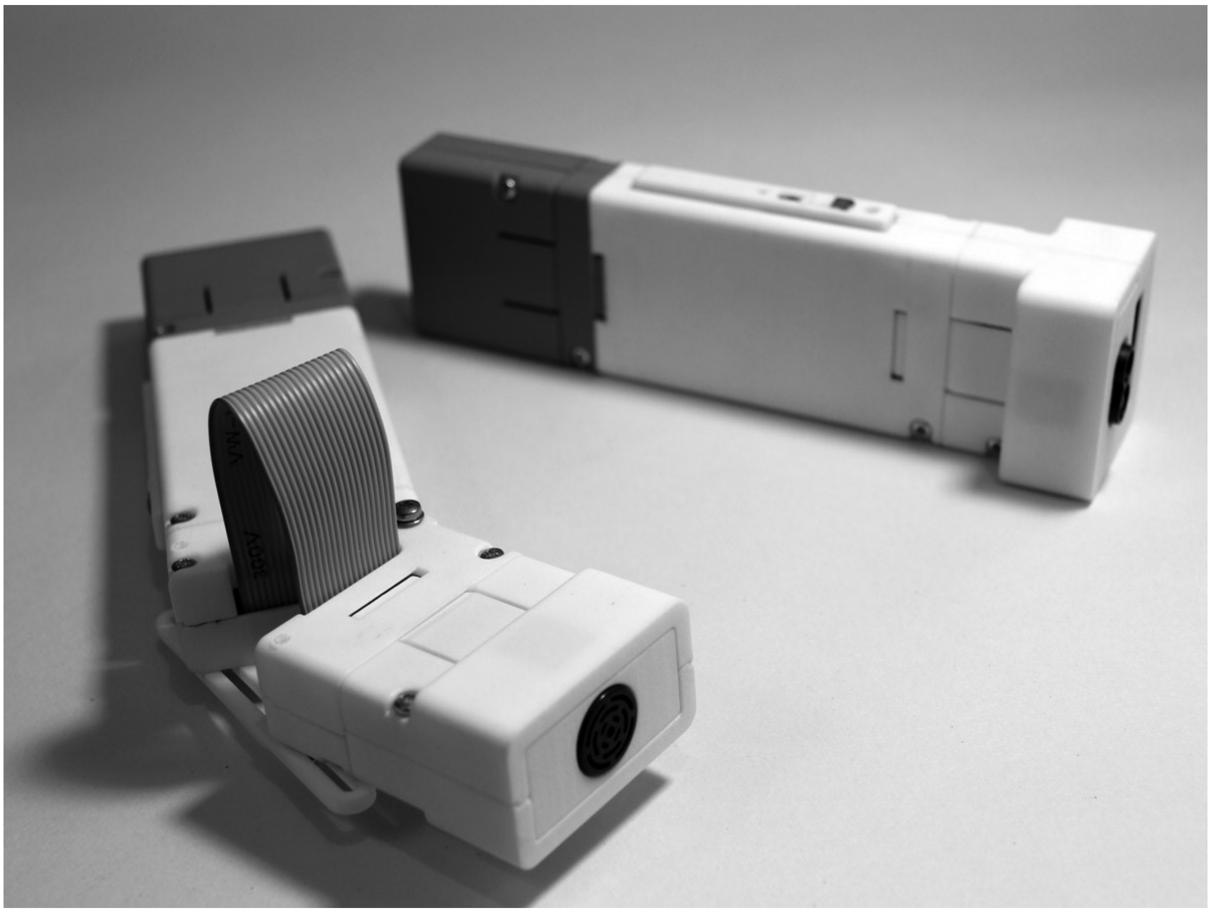
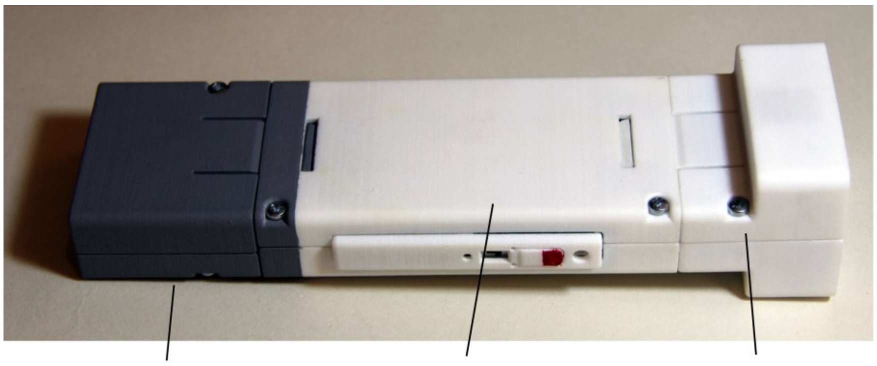
ENACTIVE TORCH RT 2 USER GUIDE



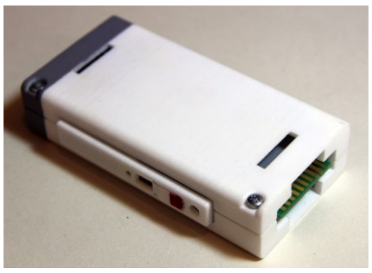
Contents

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.



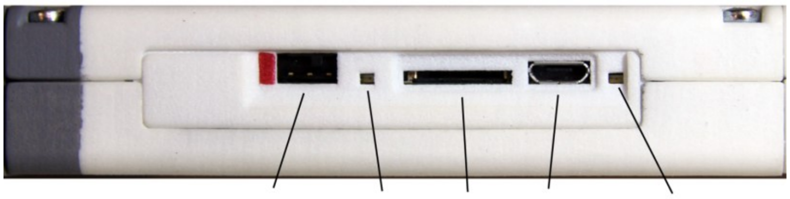
Data Capture Unit (DCU)

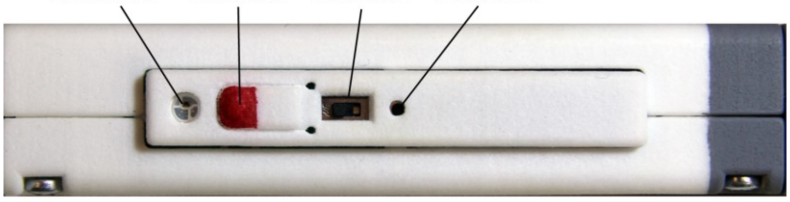


The Data Capture Unit is the core of the device. It contains the main processor, wifi module, inertial sensors and battery along with the user button and switch, USB port and micro SD card.

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.

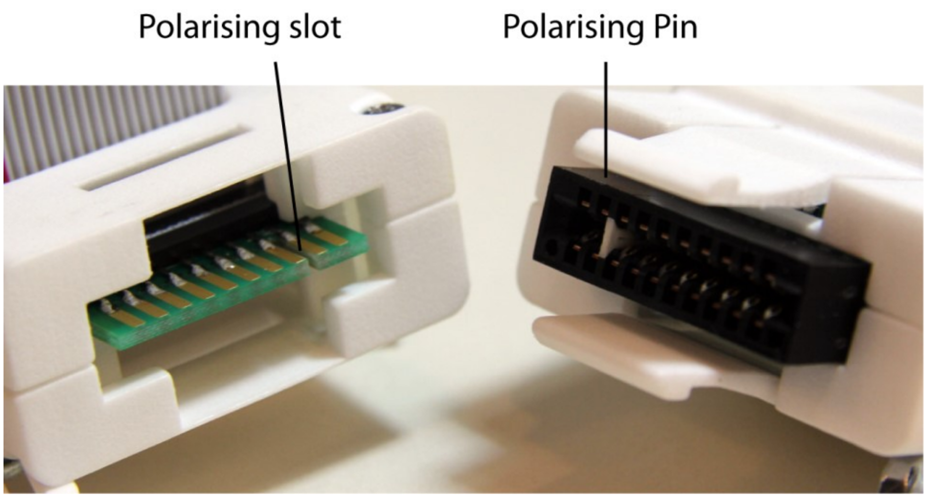
The USB port, power switch and SD card is located on one side of the DCU whilst the User LED, User button and switch and a device reset button are located on the opposite side.





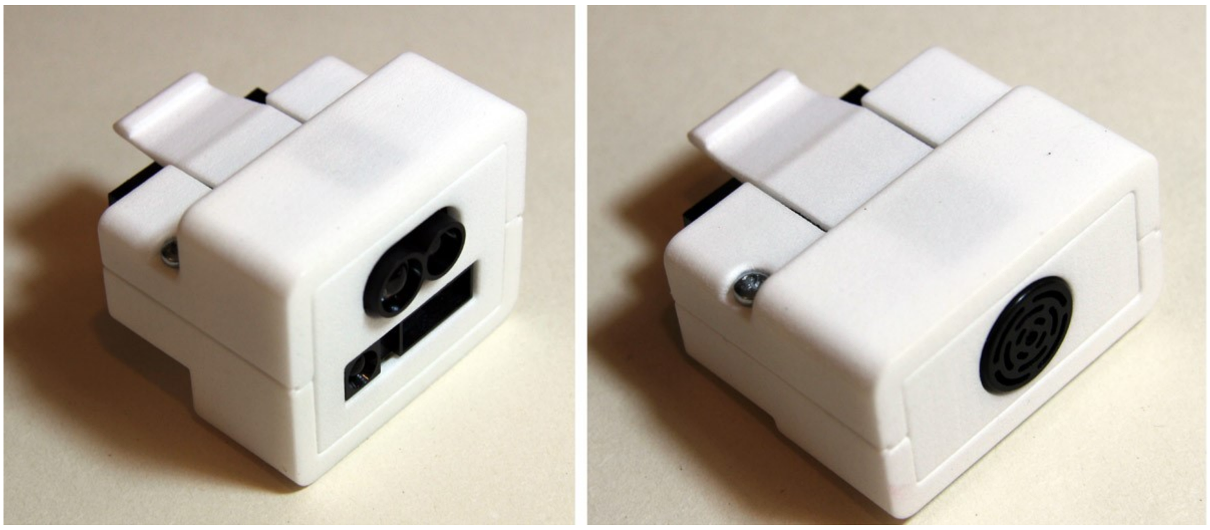
Clip connectors

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

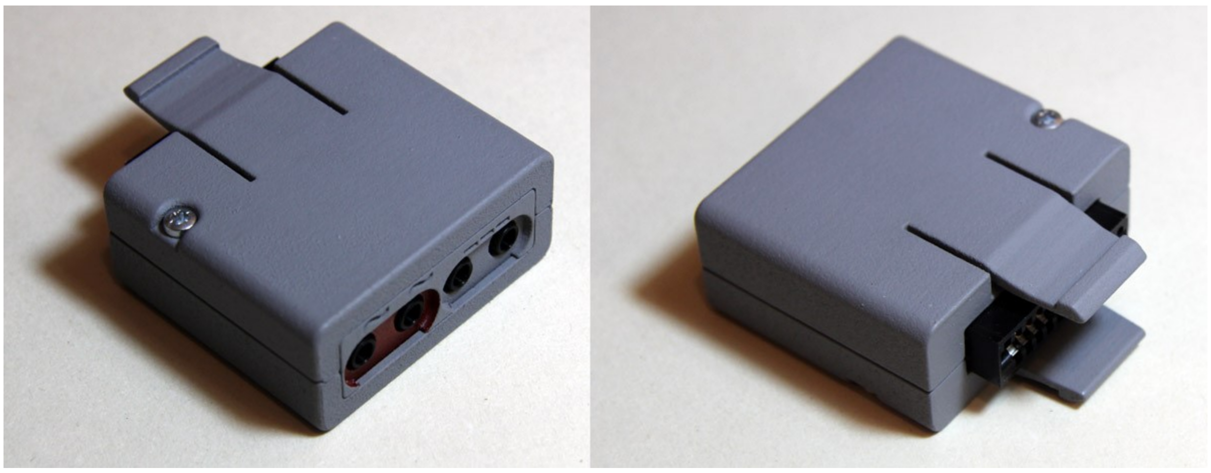
Input modules consist of a sensor or sensors that connect to the front of the device. A number of different input sensors can be connected to provide different functionality.



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.



Basic operation

Start up procedure

When the DCU is switched on it will go through a series of steps as it initialises the various functions and configures its self.

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:

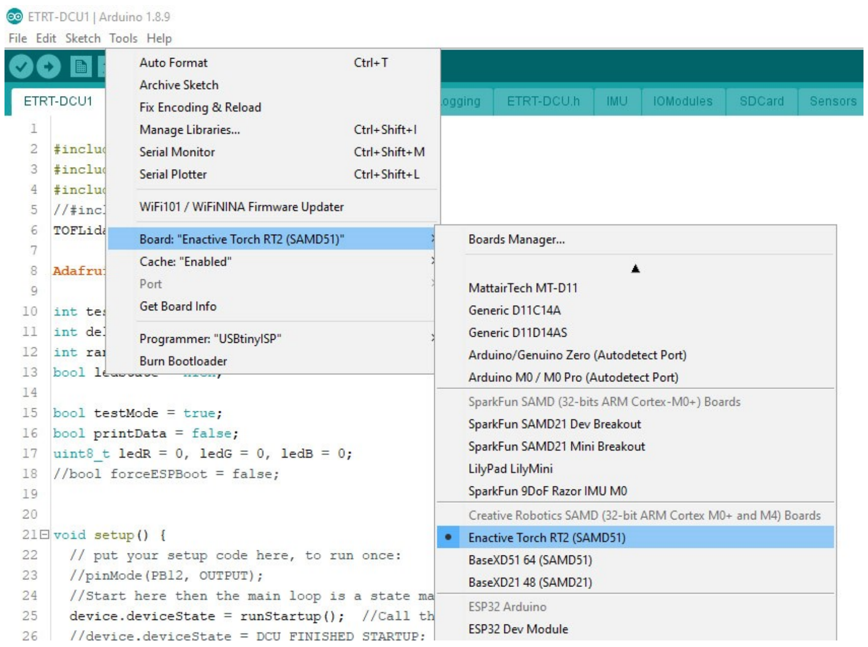
https://www.arduino.cc/en/Guide/HomePage

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.


ETRT-DCU1 | Arduino 1.8.9
File Edit Sketch Tools Help
ETRT-DCU1
ogging
ETRT-DCU.h
IMU
IOModules
SDCard
Sensors
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Enactive Torch RT2 (SAMD51)"
Cache: "Enabled"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
Boards Manager...
MattairTech MT-D11
Generic D11C14A
Generic D11D14AS
Arduino/Genuino Zero (Autodetect Port)
Arduino M0 / M0 Pro (Autodetect Port)
SparkFun SAMD (32-bits ARM Cortex-M0+) Boards
SparkFun SAMD21 Dev Breakout
SparkFun SAMD21 Mini Breakout
LilyPad LilyMini
SparkFun 9DoF Razor IMU M0
Creative Robotics SAMD (32-bit ARM Cortex M0+ and M4) Boards
● Enactive Torch RT2 (SAMD51)
BaseXD51 64 (SAMD51)
BaseXD21 48 (SAMD21)
ESP32 Arduino
ESP32 Dev Module

```

#inclu
#inclu
#inclu
//#incl
TOFLida

Adafrui

int te
int del
int rai
bool le

bool testMode = true;
bool printData = false;
uint8_t ledR = 0, ledG = 0, ledB = 0;
//bool forceESPBoot = false;


void setup() {
  // put your setup code here, to run once:
  //pinMode(PB12, OUTPUT);
  //Start here then the main loop is a state ma
  device.deviceState = runStartup();  //Call th
  //device.deviceState = DCU_FINISHED_STARTUP;
```

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 source code for the DCU firmware can be downloaded from here:

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.

?

Arguments: None

Returns: ack

Query – Returns ack.

Help

Arguments: None

Returns: Command List

Returns command list.

Ack

Arguments: None

Returns: Nothing

Send an acknowledge.

Nack

Arguments: None

Returns: Nothing

Send a NOT acknowledge.

DEBUG:

Arguments: Debug message

Returns: Nothing

Identifies an incoming message as debug information, for example coming from the ESP32

---

get status

Arguments: None

Returns: Device status summary

Request a page of status information from the device

---

set time

Arguments: HH:MM:SS

Returns: Nothing

Set the RTC clock time in hours, minutes and seconds

---

get time

Arguments: None

Returns: RTC time in hours, minutes, seconds and milliseconds.

Request the RTC Clock time

---

set date

Arguments: DD:MM:YYYY

Returns: Nothing

Sets the date of the RTC Clock

---

get date

Arguments: None

Returns: RTC date in day, month and year.

Request the RTC Date

---

set power5

Arguments: ‘on’ or ‘off’

Returns: Nothing

Turn on or off the 5V power supply

Sleep

Arguments: None

Returns: Nothing

Put the device to sleep

P

Arguments: None

Returns: Nothing

Toggle printing over USB

---

set SSID

Arguments: SSID (Network Name)

Returns: Nothing

Sets the WiFi network name.

Example: *set SSID VM12345*

---

set Pass

Arguments: WiFi Password

Returns: Nothing

Sets the WiFi password.

Example: *set Pass mypassw0rd*

ESP Boot

Arguments: None

Returns:

Set the ESP to bootloader mode

ESP RESET

Arguments: None

Returns: Nothing

Resets the ESP32 module by toggling the reset line.

ESP Status:

Arguments: ESP Status message

Returns: Nothing

Marks a status message from ESP

ESP Get:

Arguments: ESP32 server client request message (For example “GET /http”)

Request from the ESP32 for an HTML page for the server to send to a client.

The reply can have multiple lines and each line starts with *serverSend:*

When all the lines have been sent the command *serverEnd* must be sent.

ESP:

Arguments: Any ESP Command and argument

route a command to the ESP module

ESP32 command list

?

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.