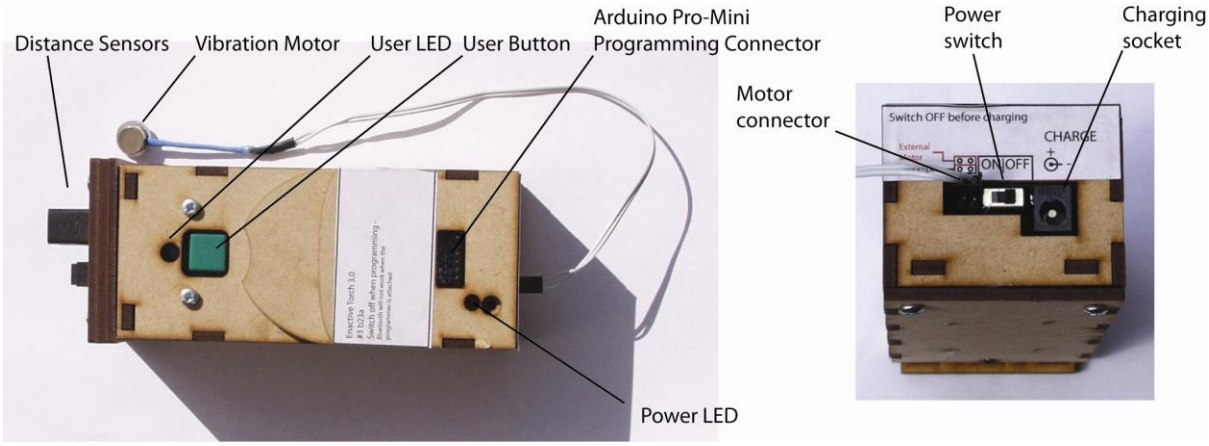
# Enactive Torch 3.x

## Instructions

### Operation:

Switch it on and after a few seconds it will start working!




### Charging the batteries:

The ET uses six AAA Batteries and has a charger socket included. When you plug the charger in it will automatically disconnect the power so the ET will stop working. It is possible to re-program the Arduino when the ET is charging because the USB programming adapter will supply power just to the Arduino when it is plugged in. Please read the charger instructions!

### Processor:

The ET is controlled with an Arduino microcontroller board - the Arduino Pro-mini 5V from Sparkfun. The Arduino can be programmed via USB using a C like language and an easy to use IDE that will work on Windows, Mac or Linux. Details of this specific board can be found here:

https://www.sparkfun.com/products/11113

More general information on the Arduino is available here:

http://www.arduino.cc/

To write or change the software for the Enactive torch you need to download and install the Arduino IDE from here: https://www.arduino.cc/en/Main/Software

Instructions on installing the Arduino IDE are available here:
https://learn.sparkfun.com/tutorials/installing-arduino-ide

To compile software for the Enactive Torch you will need to ensure you have selected the correct board and processor.

In the Arduino IDE first select ***Tools->Boards->Arduino Pro Mini*** Then select ***Tools->Processor->ATmega328 (5V, 16Mhz)***

## Sensors:

There are two infra red rangefinders at the front. The smaller one covers a range of 8-80cm (Sharp GP2D12) and the larger one covers 20-150cm (Sharp GP2Y0A02YK0F). The sensors have a non-linear response (they are more sensitive at close ranges). The basic software uses the longer range sensor only and maps it straight to the motor. The sensors interface via a socket and are mounted on their own board with a dedicated 5V regulator. It is possible to interface different sensors to the main board if required.

The ET also includes a three axis accelerometer for measuring motion. The device is an ADXL335 from Analog devices and is supplied on a daughterboard made by Sparkfun:

http://www.sparkfun.com/products/9269

## Vibration feedback:

The ET comes with a vibration motor on the end of a cable. This can be strapped to a wrist, hand or other part of the body with the wrist strap. Optionally it is possible to fit a vibrating motor internally, or to strap the external motor onto the body of the ET. Using an internal motor or a motor attached to the ET will interfere with the accelerometer readings because the motor will vibrate the body of the ET, and consequently the motion sensor.

## Communication:

The ET contains a Bluetooth serial device that can connect wirelessly with a computer equipped with a Bluetooth dongle capable of functioning as a serial port. The ET Bluetooth module is the Sparkfun BlueSMiRF Gold unit and communicates at 115200 Baud. To pair a Bluetooth dongle with the device use the code 1234

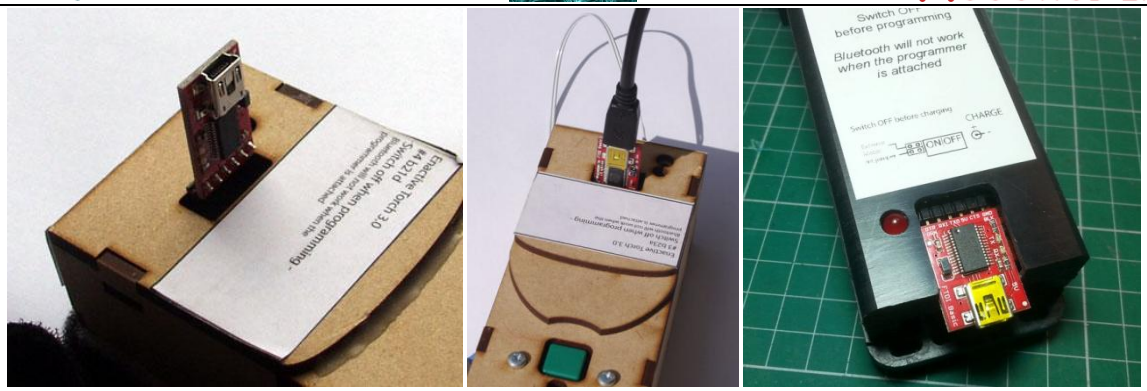
http://www.sparkfun.com/products/582

## Programming:

The ET uses an Arduino Pro-Mini controller which can be re-programmed using a suitable USB-Serial adapter. Instructions on installing the drivers are available here:

https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers

The adapter is made by Sparkfun ( http://www.sparkfun.com/products/9716 ) and plugs into the top of the ET, or from the read depending on which style case was supplied. There are two versions of the Sparkfun adapter, an old version and a newer version. The photos below illustrate how to correctly plug the adapter in. The only real difference is that the newer version has its connector mounted underneath the circuit board instead of at the end which makes it a little shorter and neater. In both instances they should be plugged in so that the side of the circuit board that the Mini USB connector is attached to is facing the front of the torch.




*Left: Old style Sparkfun USB Serial Converter. Right: New style converter. Bottom:alternative case design.*

The Arduino is programmed through its serial port which is also used to transmit and receive data to the Bluetooth module. Unfortunately this means that the programmer and Bluetooth module will not work at the same time.

The Arduino can draw power directly from the programmer so in order to programme it with the Bluetooth module in place you have to turn the ET's power switch off. This ensures that the Bluetooth module is off when the programmer is being used, but it also means that none of the sensors will work when you are programming it or using the programmer as a serial link. You can download and run a program, read sensor signals and use the programmer as a serial link to print the data, but because the sensors won't have power the data you get won't mean anything. The only way to get proper sensor data is to use the Bluetooth link when the programmer has been removed. Hopefully future versions will include a way around this!

## Software

The Enactive Torch is supplied with demonstration software maps the long range sensor values straight to the vibration motor – the closer and object the greater the vibration intensity. Pressing the user button will also generate a tone via the internal speaker (if it is installed). The pitch of the tone is proportional to the sensor value. The sensor data is also streamed over the serial port as a set of tab seperated values.

An alternative piece of software, documented below, impliments a data capture system that can be started and stopped by pressing the button. The rate at which data is transmitted is set with a variable, and each line of data begins with a value indicating the elapsed time in milliseconds since the startof a data capture session. This code is documented below.

## Code:

```
//global variables
int Sensor1, Sensor2;
int AccX, AccY, AccZ;
int MotorValue = 0;
int ToneFrequency = 500;
//pin input and output constants
const int MotorOutputPin = 9; //digital pin 4 - PWM OUTPUT
const int SpeakerPin = 4; //digital pin 4 - OUTPUT
const int ButtonPin = 8; //digital pin 8 - INPUT
const int GreenLED = 7; //digital pin 7 - OUTPUT
```

```
const int Sensor1Pin = 1; //analog input pin 1 - Long range sensor 20-150cm
const int Sensor2Pin = 2; //analog input pin 2 - Short range sensor 8-80cm
const int AccXPin = 5; //analog input pin 5
const int AccYPin = 4; //analog input pin 4
const int AccZPin = 3; //analog input pin 3
 /*
 X axis is along the length of the ET
 Y axis across the width
 Z axis from bottom to top
 */

//a high voltage on the LED pin switches it off
#define LEDOFF HIGH
#define LEDON LOW

bool printEnabled =0;
int printLog = 0;
long int lastTime = 0;
int logInterval = 10;
//logInterval is the time in milliseconds between data samples:
//10= 10 milliseconds between samples or 100Hz
//100 = 100millisecs between samples or 10Hz
long int startTime = 0;

//-------------------------------------------------
void setup(){
  Serial.begin(115200);
  //set up the digital inputs and outputs
  pinMode(SpeakerPin, OUTPUT);
  pinMode(ButtonPin, INPUT);
  pinMode(GreenLED, OUTPUT);
  playTone();
}
//-------------------------------------------------
void loop() {
  readSensors();
  checkButton();
  MotorValue = map(Sensor1, 0, 600, 0, 255);//Map the long range sensor (in the range 0-600) to the motor PWM
value (in the range 0-255)
  //MotorValue = map(Sensor1, 0, 600, 0, 255);//Map sensor2 (short range) to the motor PWM value
  analogWrite(MotorOutputPin, MotorValue);
  if(printEnabled) logData();
}

//-------------------------------------------------
void checkButton(){
  if(!digitalRead(ButtonPin)){
    //button is pressed - toggle print enable state
    printEnabled = 1-printEnabled;
    //set the LED (0 = ON) so it is ON when printing
    digitalWrite(GreenLED, !printEnabled);
    //now wait in a loop until the button is released
    while(!digitalRead(ButtonPin)) delay(10);
    //Now print a message so we mark the start and end of any data captured
    //and include an iterative number marking each time we start capturing
    if(printEnabled) {
      Serial.println("START DATA: ");
      Serial.println(printLog);
      printLog++;
      startTime = millis(); //record the time that the sample collection started
                  //Note: the millis() function returns the number of milliseconds elapsed since the Arduino started
up
    }
```

```
    if(!printEnabled) Serial.println("END DATA:");
  }
}
//-----------------------------------------------
void logData(){
  //call print data but only if a specified time has elapsed
  long int currentTime = millis();
  if(currentTime-lastTime >= logInterval){
    lastTime = currentTime; //reset for the next time
    //print the elapsed time in milliseconds since the trial was started
    Serial.print(currentTime-startTime);
    Serial.print("\t");
    //now print the sensor data;
    printData();
  }
}
//-----------------------------------------------
void readSensors(){
    Sensor1 = analogRead(Sensor1Pin);
  // wait 10 milliseconds for the analog-to-digital converter to settle after the last reading:
  delay(10);
  Sensor2 = analogRead(Sensor2Pin);
  delay(10);
  AccX = analogRead(AccXPin);
  delay(10);
  AccY = analogRead(AccYPin);
  delay(10);
  AccZ = analogRead(AccZPin);
}
//-----------------------------------------------

void printData(){
  Serial.print(Sensor1);
  Serial.print("\t");
  Serial.print(Sensor2);
  Serial.print("\t");
  Serial.print(AccX);
  Serial.print("\t");
  Serial.print(AccY);
  Serial.print("\t");
  Serial.print(AccZ);
  Serial.print("\t");
  Serial.println(MotorValue);
}
//-----------------------------------------------

void playTone(){
  //play a quick tune on startup
  digitalWrite(GreenLED, LEDON);
  tone(SpeakerPin, 392); //Start with a tone
  delay(100);
  tone(SpeakerPin, 440); //Up a full tone
  delay(100);
  tone(SpeakerPin, 349); //Down a major third
  delay(100);
  tone(SpeakerPin, 174); //Now drop an octave
  delay(100);
  tone(SpeakerPin, 261); //Up a perfect fifth
  delay(100);
  noTone(SpeakerPin);
  digitalWrite(GreenLED, LEDOFF);
 }
```

## Circuit Schematic

The schematic for the main board is below. The sensor head is not shown. It contains a 5V regulator to power the distance sensors.