



Extensible USB HID Emulation Layer for Embedded Devices

Background

Why was Isotope developed?

Professor Niesler's work on Automatic Speech Recognition has led to the development and testing of embedded speech recognition systems capable of being run on commercially available prototyping hardware like the Raspberry Pi. A significant limitation of these devices is the lack of a straightforward way in which to convey information to the host machine.

This project's goal was to develop an interface layer which would enable the emulation of input devices such as the mouse and keyboard in a universally compatible manner without the need for custom drivers on host machines.

The USB Human Interface Device protocol was selected to fulfil this requirement as it has been implemented by most major operating system vendors, is an established standard and compatible hardware ports are available on most personal computers.

The primary intention of this project was to provide this functionality while keeping the task of implementing additional libraries and features as simple as possible. Ideally it would be possible to add support to an existing or new application within a very short timeframe and with absolutely no knowledge of the USB HID protocol.

Architecture

What does Isotope consist of?



The Isotope device consists of a microprocessor with integrated USB connection and a voltage level switcher capable of converting the Master Device's logical voltage levels to ones compatible with the Isotope processor, and vice-versa.

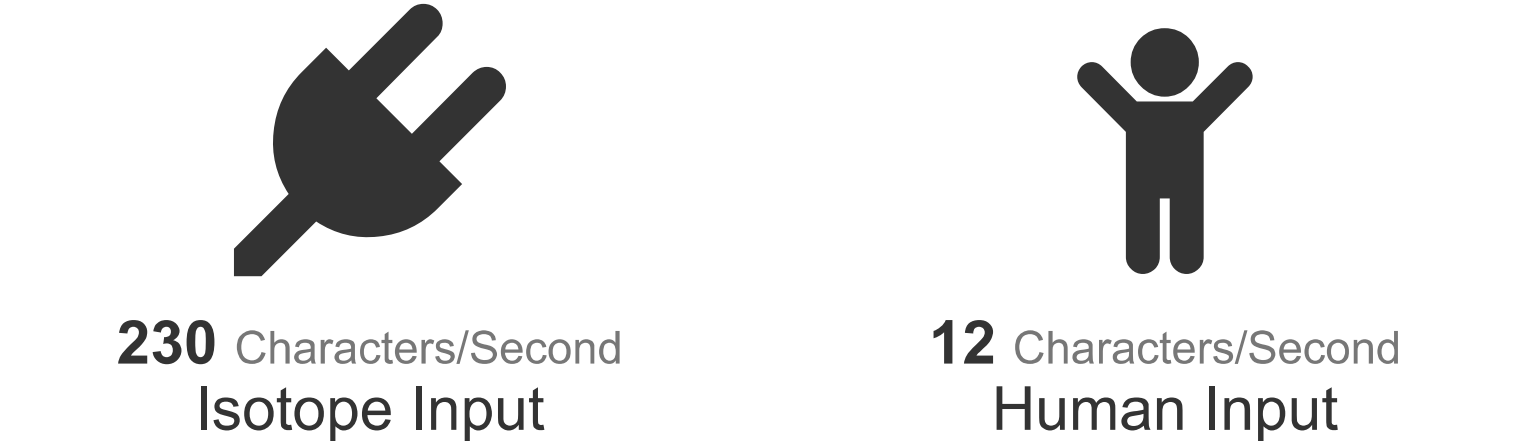
Custom firmware on the microprocessor interprets commands sent by the master device into emulation instructions, converts them into USB HID compatible packets and sends them to the host device.

Commands are sent using the Master Device's built in Universal Asynchronous Receiver/Transmitter (UART) configured to operate at 115 200 baud. These commands are encoded using a binary protocol developed to ensure maximum performance and flexibility for this task.

Performance

Performance of the device is artificially limited by a rate limiter. This rate limiter was implemented to avoid problems introduced when exceeding the maximum command rate allowed by the USB HID protocol.

Even with this rate limiter though, Isotope's performance in common tasks like text transcription still far exceeds that possible by any human.



Unfortunately, this excessively high input rate can cause issues in some applications which are designed with the assumption that input can only be received at a certain rate. This includes applications which implement spell checkers or syntax highlighters, which may freeze or even crash at high input rates.

For this reason, the rate limiter used by Isotope has been designed to be configurable. This allows the developer to select the input rate best suited to the task at hand.

Applications

Isotope was originally intended for use in ASR applications, allowing a device to emulate typing on the user's behalf without custom software or drivers on their computer. However it offers a number of advantages over alternative input emulation methods which make it significantly more useful in certain applications.

Security Penetration

Isotope operates at the hardware level, making it virtually indistinguishable from a valid USB input device to the host device. This allows it to be used to bypass security features like User Access Control on Windows Vista and later, something not possible with software level emulators.

Remote Administration

Most large server providers offer a remote administration toolkit which relies on prohibitively expensive hardware and software packages to allow control of a system from the BIOS level upwards. Isotope's hardware nature allows it to fulfil the same role at a fraction of the cost, on any system with a USB port.




Project Information

Personnel Responsible for Project



Isotope was developed by Benjamin Pannell under the supervision, and with the assistance of, Professor Thomas Niesler of the University of Stellenbosch's Electrical & Electronic Engineering Department.

To foster adoption and simplify future modifications, the source code and design have been released to the public under an MIT Open Source licence which grants permission to distribute, modify and use Isotope commercially.

Benjamin Pannell

 16447972
 benjamin.pannell@gmail.com
 sierrasoftworks.com

Professor Thomas Niesler

 trn@sun.ac.za
 dsp.sun.ac.za/~trn

Node.js Module

Using Isotope in Node.js

The Node.js module can be installed from the Node.js package repository by using the **npm** command line application. It will automaticaly build all required modules.

```
npm install libisotope
```

You can then easily use Isotope from within your next Node.js project as follows.

```
var Isotope = require('libisotope'),
    isotope = new Isotope('/dev/ttyAMA0');
isotope.keyboard.write("Hello World!");
isotope.mouse.move(-10,10);
isotope.keyboard.shift.press(Isotope.keyboard.keys.a).then.releaseAll
    .then.press(Isotope.keyboard.keys.l).then.releaseAll;
```

C Library

Using Isotope from C/C++

The C Library can be compiled from source using the **make** command. You will need to have **git**, **gcc** and **cmake** installed on your system before running these commands.

```
git clone https://github.com/SierraSoftworks/Isotope.git
cd Isotope
make
```

Using Isotope from within a C project is then relatively straightforward.

```
#include <isotope.h>
int main() {
    int isotope;
    if(isotope = isotope_open("/dev/ttyAMA0")) {
        isotope_write(isotope, "Hello World!");
        isotope_close(isotope);
        return 0;
    } else return 1;
}
```

Command Line Tools

Using Isotope from the Command Line

The command line tools are automatically built when you compile the C library. They are designed to make prototyping and testing incredibly straightforward by wrapping common functions in their own executables.

```
git clone https://github.com/SierraSoftworks/Isotope.git
cd Isotope
make
sudo cp build/apps/rpi/* /usr/local/bin/
```

You can then use the various command line applications to perform emulation commands.

```
isowrite Hello World
isokey -s h
isomouse x10 y-10
```