Exhibit 1

U.S. Patent No. 8,860,337 ("'337 Patent")

Accused Instrumentalities

Nintendo products supporting "HD Rumble" technology, including without limitation the Nintendo Switch ("Accused Products"), infringe at least Claim 2 of the '337 Patent. Each Accused Product infringes the claims in substantially the same way, and the evidence shown in this chart is similarly applicable to each Accused Product.

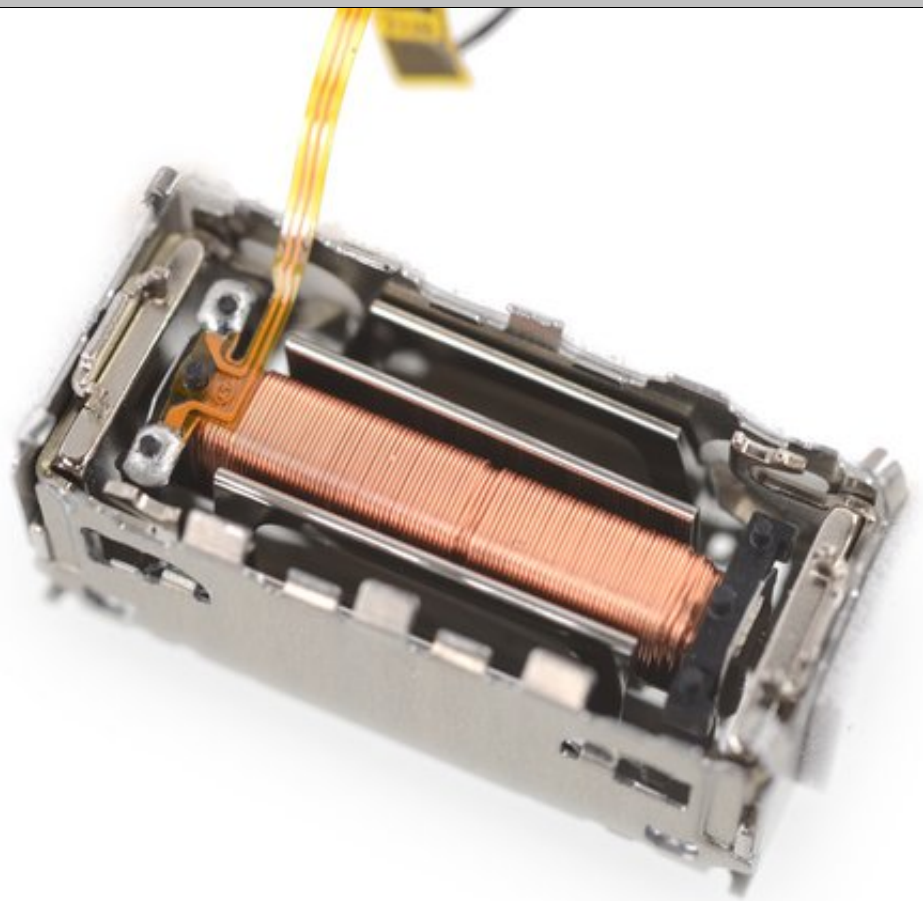
Claim 2

Claim 2	Accused Products
[2pre]. A linear vibration module comprising:	To the extent the preamble is limiting, each Accused Product includes or constitutes a linear vibration module. For example, the Switch comprises a module capable of linear vibration, as described in connection with the claim limitations below. For another example, the Switch comprises a pair of handheld controllers, each of which is capable of linear vibration, as described in the claim limitations below. *See, e.g.*:  https://www.nintendo.co.jp/corporate/release/en/2017/170113.html

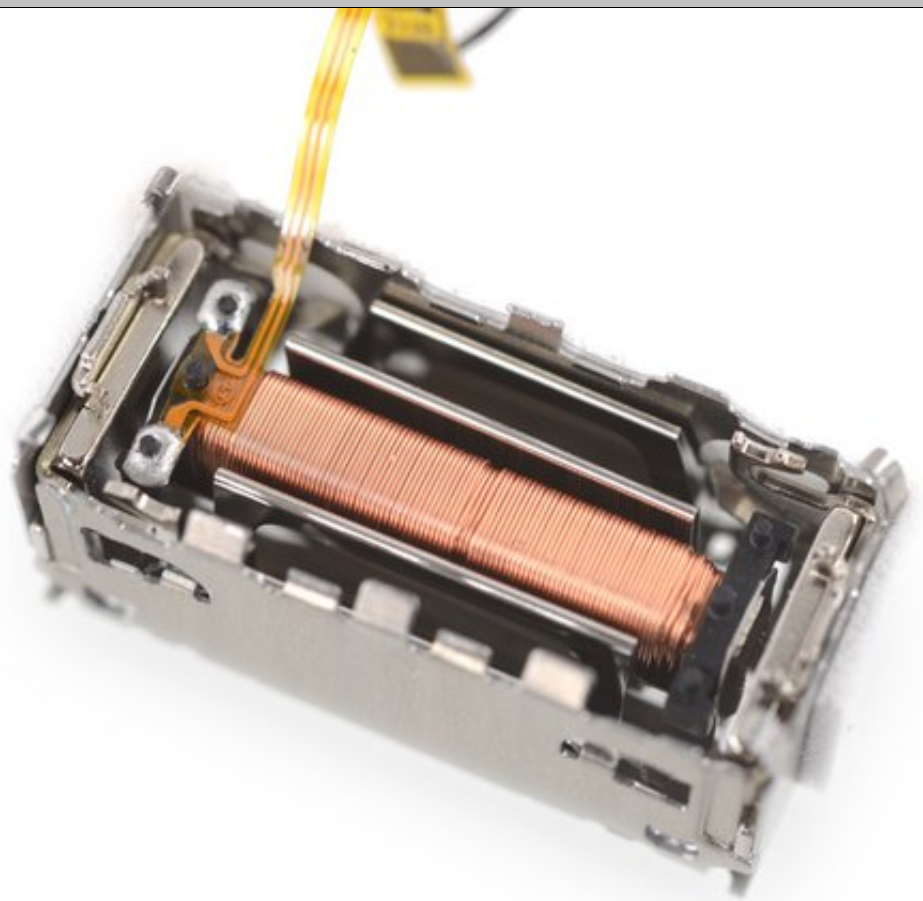
Claim 2	Accused Products
	 **The Joy-Con (L) and Joy-Con (R) controllers can be detached from the system for a variety of unprecedented playstyles.** They can be attached to the Joy-Con grip to be used as a single controller, or held separately in each hand and controlled freely using the motion control features in both the Joy-Con (L) and Joy-Con (R). You can give one to another player to enjoy shared play together, or you can attach both of them to the system and play while viewing the system screen in handheld mode. Choose the playstyle that fits your play location, the game content, and your own preferences. In addition to the features of past game controllers, **the Joy-Con controllers have incredible expressive power** due to new features such as the IR Motion Camera that can sense the distance, shape, and motion of nearby objects, and the HD rumble feature that can convey detailed tactile sensations, such as the feeling of ice cubes tumbling against each other in a glass. https://www.nintendo.co.jp/corporate/release/en/2017/170113.html

Claim 2	Accused Products
	 https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263
[2a] a housing;	Each Accused Product comprises a housing. For example, the Switch handheld controller comprises a linear resonant actuator housing that contains the moveable component. *See* limitations below. *See also, e.g.*:

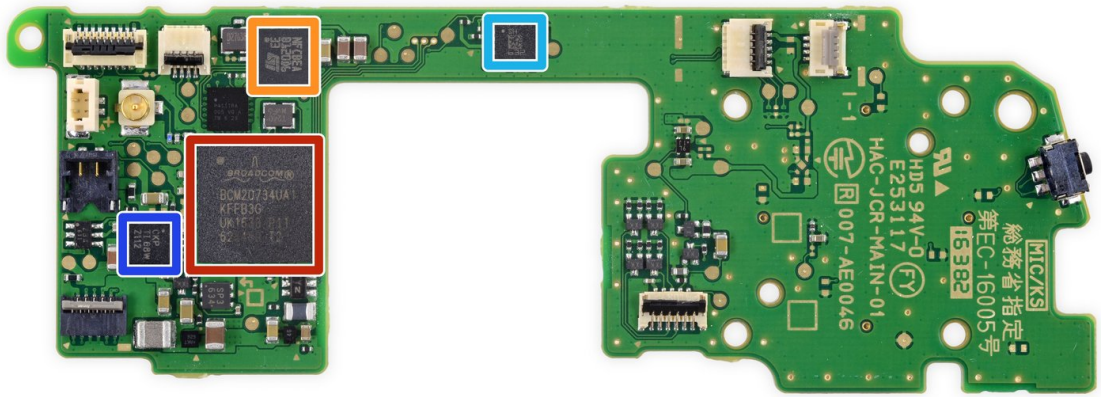
Claim 2	Accused Products
	 Photograph of Switch handheld controller showing LRA at left (https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263).

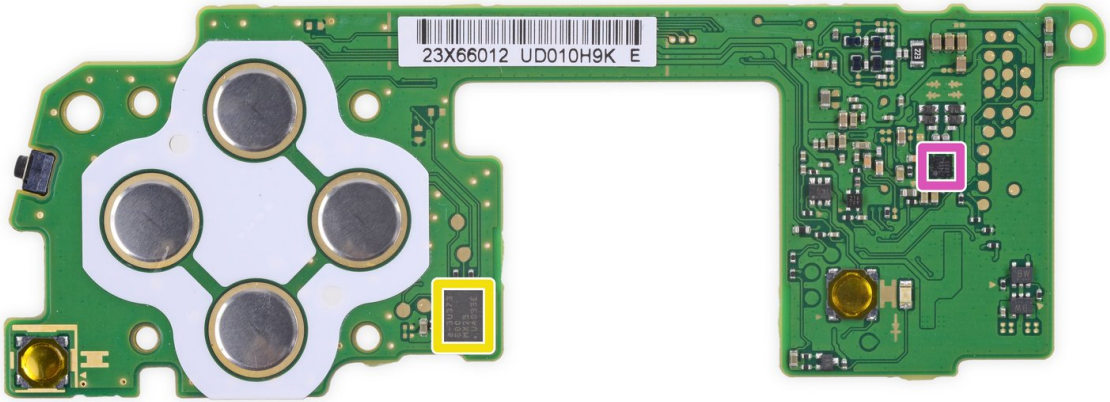
Claim 2	Accused Products
	 Photograph of internals of Switch LRA (https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263)
[2b] a moveable component;	Each Accused Product comprises a moveable component.

Claim 2	Accused Products
	For example, the linear motor in the Switch handheld controller includes a movable component comprising permanent magnet(s) and a mass. *See, e.g.*:

Claim 2	Accused Products
	 Photograph of internals of Switch LRA showing moveable component (https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263)
[2c] a power supply;	Each Accused Product comprises a power supply.

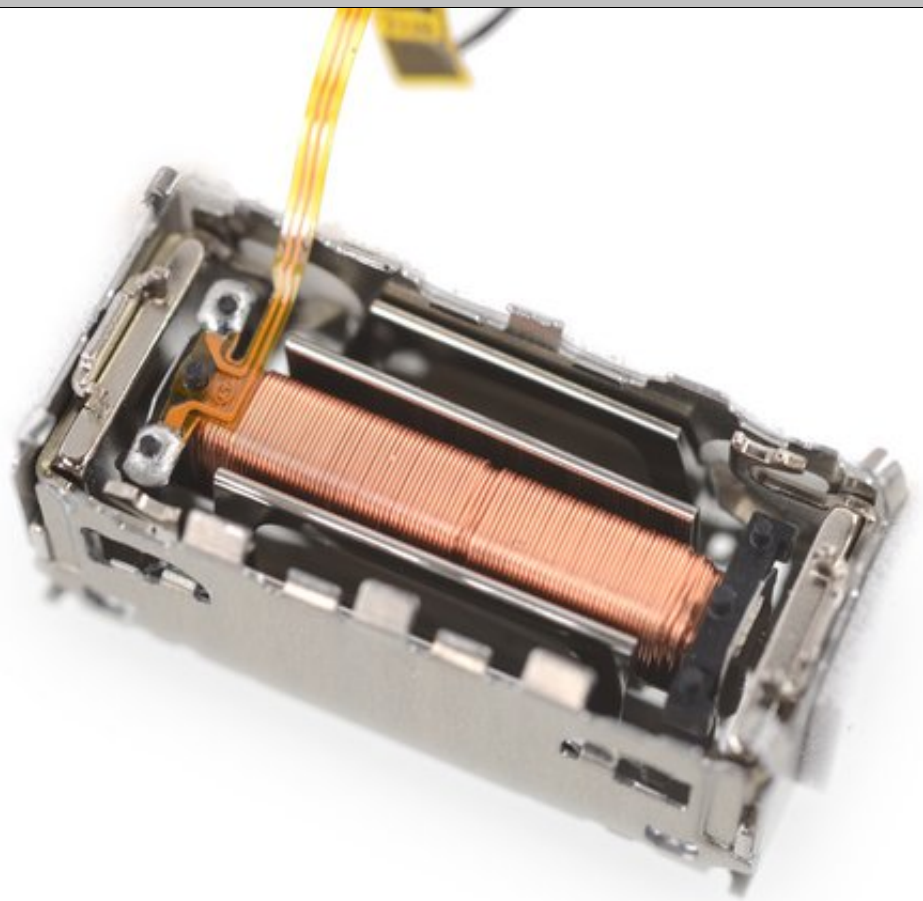
Claim 2	Accused Products
	For example, the linear motor coil driver (described below) in the handheld controller of the Switch receives power from a battery, a USB connection, one or more voltage regulators on or near the Accused Product's system board, and one or more amplifier or haptic control ICs. *See, e.g.*:

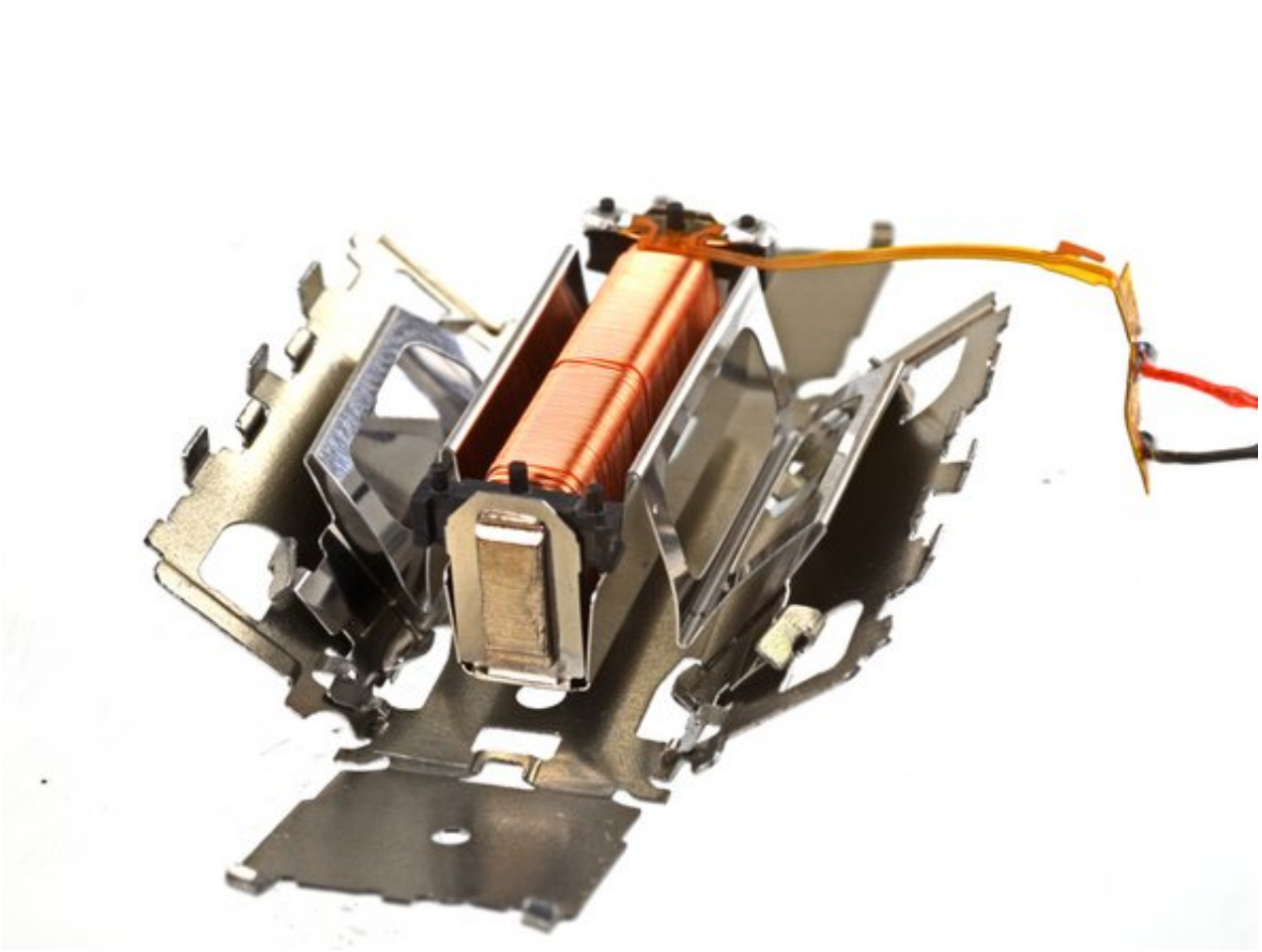
Claim 2	Accused Products
	 Photograph of Switch handheld controller system board, with power supply and control ICs highlighted (https://guide-images.cdn.ifixit.com/igi/NYPdejcKUDMhLnLr.huge)

Claim 2	Accused Products
	 Photograph of Switch handheld controller system board, with class-D audio amplifier highlighted in magenta at right (https://guide-images.cdn.ifixit.com/igi/hwXfQaAA2FH1RLkl.huge) *See also* claim elements below.

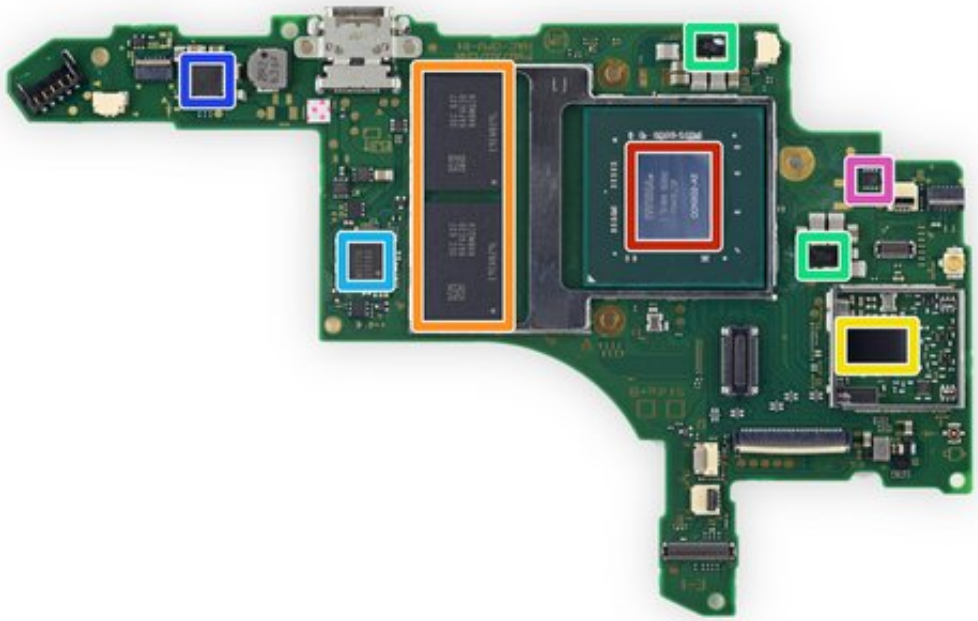
Claim 2	Accused Products
[2d] user-input features;	Each Accused Product comprises user-input features. For example, the handheld controller of the Switch comprises buttons, a joystick, and a trigger. *See, e.g.*:  https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263
[2e] a driving component that drives the moveable component in each of two	Each Accused Product includes a driving component that drives the moveable component to oscillate within the housing.

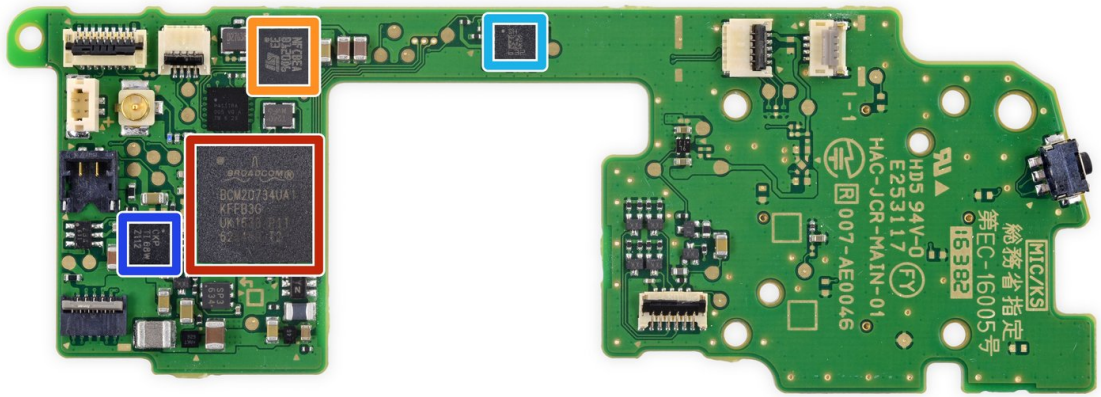
Claim 2	Accused Products
opposite directions within the housing; and	For example, the linear motor in the Switch includes one or more coils to form electromagnetic fields for driving the moveable component in two directions. *See, e.g.*:

Claim 2	Accused Products
	 Photograph of internals of Switch LRA with driving component (https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263)

Claim 2	Accused Products
	 Photograph of internals of Switch LRA with driving component (https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263)

Claim 2	Accused Products
[2f] a control component that controls supply of power from the power supply to the driving component to cause the moveable component to oscillate at a frequency and an amplitude specified by user input received from the user-input features,	Each Accused Product comprises a control component that controls supply of power from the power supply to the driving component to cause the moveable component to oscillate at a frequency and an amplitude specified by one or more stored values. For example, the Switch includes a Tegra X1-based SoC that instructs the power supply in the handheld controller to provide power to the driving component to drive the moveable component to oscilate at a frequency and amplitude specified by user input received from the user-input features. For another example, the handheld controller of the Switch includes one or more integrated circuits that control the supply of power to the driving component to drive the moveable component to oscilate at a frequency and amplitude specified by user input received from the user-input features. Although Nintendo's APIs are proprietary and confidential, published reverse-engineering efforts by third parties show that the Switch is configured to communicate specific frequencies and amplitudes to the handheld controller over a serial and/or Bluetooth connection. These frequencies and amplitudes are specified by user input received from the user-input features. *See* claim limitations above. *See also, e.g.*:

Claim 2	Accused Products
	 Photograph of Switch SoC, highlighted in red (https://www.ifixit.com/Teardown/Nintendo+Switch+Teardown/78263)

Claim 2	Accused Products
	 Photograph of Switch handheld controller system board, with power supply and control ICs highlighted (https://guide-images.cdn.ifixit.com/igi/NYPdejcKUDMhLnLr.huge)

Claim 2	Accused Products

	# Bluetooth HID Information
	
	## Output reports
	
	### OUTPUT 0x01
	
	Rumble and subcommand.
	
	The OUTPUT 1 report is how all normal subcommands are sent. It also includes rumble data.
	
	Sample C code for sending a subcommand:
	
	``
	uint8_t buf[0x40]; bzero(buf, 0x40);
	buf[0] = 1; // 0x10 for rumble only
	buf[1] = GlobalPacketNumber; // Increment by 1 for each packet sent. It loops in 0x0 - 0xF range.
	memcpy(buf + 2, rumbleData, 8);
	buf[10] = subcommandID;
	memcpy(buf + 11, subcommandData, subcommandDataLen);
	hid_write(handle, buf, 0x40);
	``
	
	You can send rumble data and subcommand with `x01` command, otherwise only rumble with `x10` command.
	
	See "Rumble data" below.
	
	https://github.com/dekuNukem/Nintendo_Switch_Reverse_Engineering/blob/ac8093c84194b3232acb675ac1accce9bcb456a3/bluetooth_hid_notes.md

Claim 2	Accused Products
	Rumble data A timing byte, then 4 bytes of rumble data for left Joy-Con, followed by 4 bytes for right Joy-Con. [00 01 40 40 00 01 40 40] (320Hz 0.0f 160Hz 0.0f) is neutral. The rumble data structure contains 2 bytes High Band data, 2 byte Low Band data. The values for HF Band frequency and LF amplitude are encoded. **Byte #** / **Range** / **Remarks** 0, 4 / `x04` - `xFC` (81.75Hz - 313.14Hz) / High Band Lower Frequency. Steps `+0x0004`. 0-1, 4-5 / `x00 01` - `xFC 01` (320.00Hz - 1252.57Hz) / Byte `1`, `5` LSB enables High Band Higher Frequency. Steps `+0x0400`. 1, 5 / `x00 00` - `xC8 00` (0.0f - 1.0f) / High Band Amplitude. Steps `+0x0200`. Real max: `FE`. 2, 6 / `x01` - `x7F` (40.87Hz - 626.28Hz) / Low Band Frequency. 3, 7 / `x40` - `x72` (0.0f - 1.0f) / Low Band Amplitude. Safe max: `00 72`. 2-3, 6-7 / `x80 40` - `x80 71` (0.01f - 0.98f) / Byte `2`, `6` +0x80 enables intermediate LF amplitude. Real max: `80 FF`. https://github.com/dekuNukem/Nintendo_Switch_Reverse_Engineering/blob/ac8093c84194b3232acb675ac1accce9bcb456a3/bluetooth_hid_notes.md

Claim 2	Accused Products

	The high frequency and low amplitude are encoded and must always add the "control" byte to the HA/LF byte. An example is the following:
	
	``
	//Left linear actuator
	uint16_t hf = 0x01a8; //Set H.Frequency
	uint8_t hf_amp = 0x88; //Set H.Frequency amplitude
	//Byte swapping
	byte[0] = hf & 0xFF;
	byte[1] = hf_amp + ((hf >> 8) & 0xFF); //Add amp + 1st byte of frequency to amplitude
	
	uint8_t lf = 0x63; //Set L.Frequency
	uint16_t lf_amp = 0x804d; //Set L.Frequency amplitude
	//Byte swapping
	byte[2] = lf + ((lf_amp >> 8) & 0xFF); //Add freq + 1st byte of LF amplitude to the f
	byte[3] = lf_amp & 0xFF;
	``
	https://github.com/dekuNukem/Nintendo_Switch_Reverse_Engineering/blob/ac8093c84194b3232acb675ac1accce9bcb456a3/rumble_data_table.md

[2g] wherein the control component drives simultaneous oscillation of the moveable component at two or more frequencies to generate complex vibration modes.	In the Accused Products, the control component drives simultaneous oscillation of the moveable component at two or more frequencies to generate complex vibration modes. For example, the Switch is configured to create “HD Rumble” effects. Although Nintendo’s APIs are proprietary and confidential, published reverse-engineering efforts by third parties show that the Switch is configured to communicate two specific, simultaneous frequencies to the handheld controller. The controller responds to these commands by simultaneously oscillating at two frequencies, generating complex vibration modes. *See, e.g.*:

Claim 2	Accused Products

	# Bluetooth HID Information
	
	## Output reports
	
	### OUTPUT 0x01
	
	Rumble and subcommand.
	
	The OUTPUT 1 report is how all normal subcommands are sent. It also includes rumble data.
	
	Sample C code for sending a subcommand:
	
	``
	uint8_t buf[0x40]; bzero(buf, 0x40);
	buf[0] = 1; // 0x10 for rumble only
	buf[1] = GlobalPacketNumber; // Increment by 1 for each packet sent. It loops in 0x0 - 0xF range.
	memcpy(buf + 2, rumbleData, 8);
	buf[10] = subcommandID;
	memcpy(buf + 11, subcommandData, subcommandDataLen);
	hid_write(handle, buf, 0x40);
	``
	
	You can send rumble data and subcommand with `x01` command, otherwise only rumble with `x10` command.
	
	See "Rumble data" below.
	
	https://github.com/dekuNukem/Nintendo_Switch_Reverse_Engineering/blob/ac8093c84194b3232acb675ac1accce9bcb456a3/bluetooth_hid_notes.md

Claim 2	Accused Products
	Rumble data A timing byte, then 4 bytes of rumble data for left Joy-Con, followed by 4 bytes for right Joy-Con. [00 01 40 40 00 01 40 40] (320Hz 0.0f 160Hz 0.0f) is neutral. The rumble data structure contains 2 bytes High Band data, 2 byte Low Band data. The values for HF Band frequency and LF amplitude are encoded.

Byte #	Range	Remarks
0, 4	`x04` - `xFC` (81.75Hz - 313.14Hz)	High Band Lower Frequency. Steps `+0x0004`.
0-1, 4-5	`x00 01` - `xFC 01` (320.00Hz - 1252.57Hz)	Byte `1`, `5` LSB enables High Band Higher Frequency. Steps `+0x0400`.
1, 5	`x00 00` - `xC8 00` (0.0f - 1.0f)	High Band Amplitude. Steps `+0x0200`. Real max: `FE`.
2, 6	`x01` - `x7F` (40.87Hz - 626.28Hz)	Low Band Frequency.
3, 7	`x40` - `x72` (0.0f - 1.0f)	Low Band Amplitude. Safe max: `00 72`.
2-3, 6-7	`x80 40` - `x80 71` (0.01f - 0.98f)	Byte `2`, `6` +0x80 enables intermediate LF amplitude. Real max: `80 FF`.

https://github.com/dekuNukem/Nintendo_Switch_Reverse_Engineering/blob/ac8093c84194b3232acb675ac1accce9bcb456a3/bluetooth_hid_notes.md

Claim 2	Accused Products

	The high frequency and low amplitude are encoded and must always add the "control" byte to the HA/LF byte. An example is the following:
	
	`//Left linear actuator`
	`uint16_t hf = 0x01a8; //Set H.Frequency`
	`uint8_t hf_amp = 0x88; //Set H.Frequency amplitude`
	`//Byte swapping`
	`byte[0] = hf & 0xFF;`
	`byte[1] = hf_amp + ((hf >> 8) & 0xFF); //Add amp + 1st byte of frequency to amplitude`
	
	`uint8_t lf = 0x63; //Set L.Frequency`
	`uint16_t lf_amp = 0x804d; //Set L.Frequency amplitude`
	`//Byte swapping`
	`byte[2] = lf + ((lf_amp >> 8) & 0xFF); //Add freq + 1st byte of LF amplitude to the f`
	`byte[3] = lf_amp & 0xFF;`
	
	https://github.com/dekuNukem/Nintendo_Switch_Reverse_Engineering/blob/ac8093c84194b3232acb675ac1accce9bcb456a3/rumble_data_table.md