|  |  |
| --- | --- |
| Project | **Specification of Sensor Interface for Cyber and Physical World**  <https://sagroups.ieee.org/2888/ > |
| Title | **Sensor data formats for microphones and color cameras** |
| DCN | **2888-20-0009-01-0001** |
| Date Submitted | **Feb. 23, 2020** |
| Source(s) | Sang-Kyun Kim, [goldmunt@gmail.com](mailto:goldmunt@gmail.com) (Myongji University)  Kyoungro Yoon, [yoonk@konkuk.ac.kr](mailto:yoonk@konkuk.ac.kr) (Konkuk University)  Sangkwon Jeong, [ceo@joyfun.kr](mailto:ceo@joyfun.kr) (Joyfun) |
| Re: |  |
| Abstract | Microphones and cameras are the most basic sensors for acquiring audio and visual information of the physical world, and the data through them are important information for determining the characteristics and motion of cyber objects in the cyber world. This contribution proposes a schema for defining sensor data input from microphones and color camera sensors. |
| Purpose | To start discussion on purpose of the standard |
| Notice | This document has been prepared to assist the IEEE 2888 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. |
| Release | The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE’s name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE’s sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that IEEE 3079 may make this contribution public. |
| Patent Policy | The contributor is familiar with IEEE patent policy, as stated in [Section 6 of the IEEE-SA Standards Board bylaws](http://standards.ieee.org/guides/opman/sect6.html#6.3) <[http://standards.ieee.org/guides/bylaws/sect6-7.html#6](http://127.0.0.1:4664/cache?event_id=757737&schema_id=1&s=5X0vID10lu_E6yrIkWkNd4Wz2H8&q=hancock)> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/faq.pdf> |

# Introduction

Microphones and cameras are the most basic sensors for acquiring audio and visual information of the physical world, and the data through them are important information for determining the characteristics and motion of cyber objects in the cyber world. This contribution proposes a schema for defining sensor data input from microphones and color camera sensors.

# Schema for A/V sensor data

This section describes the data types of AV sensors, which collect audio and visual information from the physical world.

## Microphone sensor

### Syntax of microphone sensor data type

    "microphoneSensorType": {

      "type": "object",

      "properties": {

        "orientation": {

          "$ref": "#/definitions/float3DVectorType"

        },

        "location": {

          "$ref": "#/definitions/float3DVectorType"

        },

        "audioData": {

          "$ref": "#/definitions/rawAudioType"

        }

      },

      "additionalProperties": false

    },

    "rawAudioType": {

      "type": "object",

      "properties": {

        "sample\_rate": {

          "$ref": "#/definitions/sampleRateType"

        },

        "byte\_order": {

          "$ref": "#/definitions/byteOrderType"

        },

        "sign": {

          "$ref": "#/definitions/signType"

        },

        "resolution": {

          "$ref": "#/definitions/resolutionType"

        }

      },

      "additionalProperties": false

    },

    "sampleRateType": {

      "type": "number",

      "enum": [

        8.0,

        11.025,

        16.0,

        22.05,

        32.0,

        44.056,

        44.1,

        47.25,

        48.0,

        50.0,

        50.4,

        88.2,

        96.0,

        176.4,

        192.0,

        352.8,

        2822.4,

        5644.8

      ]

    },

    "byteOrderType": {

      "type": "string",

      "enum": [

        "littleEndian",

        "bigEndian"

      ]

    },

    "signType": {

      "type": "string",

      "enum": [

        "signed",

        "unsigned"

      ]

    },

    "resolutionType": {

      "type": "integer",

      "enum": [

        4,

        8,

        12,

        16,

        20,

        24,

        32,

        48,

        64

      ]

    },

figure 1. Schema for microphone sensor

Figure 1 shows the schema of a microphone sensor. Microphone sensor data includes *orientation*, *location* and *audioData*.

The *orientation* and *location* indicate the microphone's orientation and location, respectively. The *orientation* indicates the direction through yaw, pitch, and roll. The *location* represents x, y, z coordinates. The *audioData* refers to *rawAudioType* with *audioData16*, *audioData64*, *sample*\_rate, *byte*\_*order*, *sign*, and *resolution*.

The *audioData16* is audio binary data in string format encoded in base-16 format. The *audioData64* is audio binary data in string format encoded in base-64 format. You must choose either *audioData16* or *audioData64*.

The *sample\_rate* represents a sampling rate of recorded audio, and a unit is kHz. It uses predefined values ​​from 8.0 to 5644.8. The *byte*\_*order* is used to specify whether the byte representing the audio data follows a big or little-endian. The *sign* indicates whether audio data is a signed value or an unsigned value. The *resolution* indicates how many bits the sample uses. Typically, 8 bits are used.

### Example of microphone sensor data instance

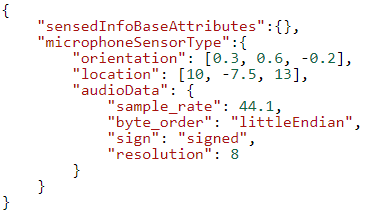


figure 2. Example of microphone sensor data JSON instance

Figure 2 shows a JSON instance of microphone sensor data. The *orientation* has [0.3, 0.6, -0.2], the *altitude* is 123.21 and the *location* is [10, -7.5, 13]. The *sample\_rate* is 44.1kHz and *byte\_order* follows the little-endian method. It is signed data and its *resolution* is 8 bits.

## Color camera sensor

### Syntax of color camera sensor data type

    "colorCameraSensorType": {

      "type": "object",

      "properties": {

        "cameraSensor": {

          "$ref": "#/definitions/cameraSensorType"

        },

        "rawVideo": {

          "$ref": "#/definitions/rawVideoType"

        }

      }

    },

    "rawVideoType": {

      "type": "object",

      "properties": {

        "videoData16": {

          "type": "string"

        },

        "videoData64": {

          "type": "string"

        },

        "width": {

          "type": "integer"

        },

        "height": {

          "type": "integer"

        },

        "bit\_depth": {

          "type": "integer"

        },

        "stride": {

          "type": "integer"

        },

        "coding4CC": {

          "type": "integer"

        },

        "fps": {

          "type": "integer"

        },

        "use\_frame\_packing": {

          "type": "integer"

        },

        "frame\_packing": {

          "type": "integer"

        }

      }

    },

    "cameraSensorType": {

      "type": "object",

      "properties": {

        "cameraOrientation": {

          "$ref": "#/definitions/orientationSensorType"

        },

        "cameraLocation": {

          "$ref": "#/definitions/globalPositionSensorType"

        },

        "cameraAltitude": {

          "$ref": "#/definitions/altitudeSensorType"

        },

        "focalLength": {

          "type": "number"

        },

        "aperture": {

          "type": "number"

        },

        "shutterSpeed": {

          "type": "number"

        },

        "filter": {

          "type": "string",

          "enum": [

            "UV",

            "Polarizing",

            "NB",

            "Diffusion",

            "Star"

          ]

        }

      },

      "additionalProperties": false

    },

    "globalPositionSensorType": {

      "type": "object",

      "properties": {

        "longitude": {

          "type": "number",

          "minimum": -180.0,

          "maximum": 180.0

        },

        "latitude": {

          "type": "number",

          "minimum": -90.0,

          "maximum": 90.0

        }

      },

      "required": [

        "longitude",

        "latitude"

      ],

      "additionalProperties": false

    },

figure 3. Schema for color camera sensor

Figure 3 shows the schema of a color camera sensor. The color camera sensor includes a basic camera *cameraSensor* and *rawVideo* representing video data.

The *cameraSensor* has *cameraOrientation*, *cameraAltitude*, *cameraLocation*, *focalLength*, *aperture*, *shutterSpeed*, and *filter*. The *cameraOrientation* refers to the orientation sensor, which is a location sensor, to indicate the orientation of the camera. The *cameraLocation* indicates the global position of the camera by referring to the global position sensor, which is a location sensor. The *focalLength* describes the distance in millimeters (mm) between the lens and the image sensor when the subject is in focus. The *aperture* represents the diameter of the aperture. For example, if the diaphragm is f/2.8, it is written as 2.8. The *shutterSpeed* ​​represents the camera shutter speed. The shutter speed is expressed in seconds. The *filter* represents a camera filter and can only be used among the specified filters.

The *rawVideo* includes *videoData16*, *videoData64*, *width*, *height*, *bit\_depth*, *stride*, *use\_frame\_packing*, and *frame\_packing*. The *videoData16* is video binary data in string format encoded in base-16 format. The *videoData64* is video binary data in string format encoded in base-64 format. You must choose either *videoData16* or *videoData64*. The *width* and *height* represent the width and height of the video in pixels. The *bit\_depth* represents the number of bits of each channel sample in the set of allowed values ​​defined by *coding4CC.* The *stride* represents the size of the horizontal byte. The *coding4CC* is a four letter code representing the raw data parameter specified by MP4RA. The *fps* represents frames per second. The *use\_frame\_packing* Indicates whether two or more views are included in the frame. The *frame\_packing* represents a frame packing value.

### Example of color camera sensor data instance

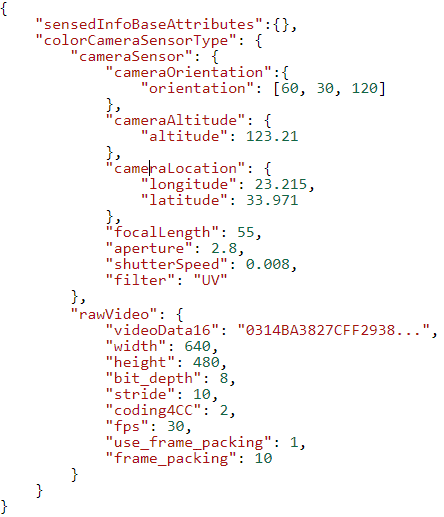


figure 4. Example of color camera sensor data JSON instance

Figure 4 shows a JSON instance of color camera sensor data. The *cameraOrientation* values are yaw, pitch, and roll values of 60, 30, and 120, respectively. The *cameraAltitude* is 123.21 and the *cameraLocation* has longitude 23.215 and latitude 33.971. The *flocalLength* is 55mm and the *aperture* is f / 2.8. The *shutterSpeed* is 0.008sec and the *filter* indicates that a UV filter is used.

The *videoData16* is base-16 encoded data starting with "0314BA3827CFF2938 ..". The *width* and *height* of the image are 640 and 480 pixels, respectively. The *bit\_depth* is 8 and stride is 10. The *coding4CC* is 2 and the *fps* of the image is 30 frames/sec. The *use\_frame\_packing* is 1 and the *frame\_packing* is 10.

# Conclusion

It is recommended to adopt the JSON-based microphone and color camera sensor data schema proposed in this contribution as the audio/visual sensor data schema of IEEE 2888.1. In addition, it is recommended to improve the data schema in the future by adding AV sensor data information actually used in the industry.