Sensact Functional Document v2

1. **Introduction**

This document is to describe how we plan to implement the next version of the Sensact, not the current functionality. The major changes are: input jacks being configurable, Bluetooth input, trigger events, and Infrared. Things in red are the things that we are uncertain about.

1. **Inputs**

* 3 audio jacks
  + Each is TRRS and can be configurable by software to be standard connection or Tash connection
  + Each will be controlled by a digital output pin from the Arduino
  + Tash
    - T – VCC
    - R – AX
    - R – AX
    - S – Nothing
  + Standard
    - T – VCC
    - R – AX
    - R – AX
    - S – GND
* Bluetooth BLE
  + This will be used to read the Gyroscope, or any other wireless sensor
  + There can be a variable number of sensors sending information over this Bluetooth line
    - Can have different Bluetooth modules connect without any alterations. (ie. One module has a gyroscope and sends data from 3 sensors, another module is just a wireless button and sends one sensor’s data)
  + This line will use Digital Pins 0 and 1 for the standard Serial port.
* I2C
  + This is for an alternative version of the Sensact without Bluetooth input
  + The Code on the Sensact Arduino will be different than the one with Bluetooth
  + The Sensact board will have female jumper pins with the I2C connections (VCC, GND, SCL, SDA)
  + The device address and register addresses will be different for each I2C device, therefore it will need to be hard coded
  + The Gyroscope will be the main user of this functionality
* IR
  + Still unsure if we need this or not
  + Can be used to record signals from a remote and store it, then can be set up to send it back out as a response to a trigger

1. **Outputs**

* Relays (A&B)
  + Digital pulse
  + Audio TRRS jack
* Bluetooth HID
  + Outputs keyboard character chosen by config software
  + This Serial line will use a Software Serial with Digital Pins other than 0 and 1
* Mouse
  + Move left
  + Move right
  + Move up
  + Move down
  + Left Click
* Keyboard
  + Outputs keyboard character chosen by config software to connected computer
  + (We may be able to allow the user to enter special characters in here like Enter, Arrow keys, etc. if this is necessary. We can let this do “Arrow L/R/U/D” if we need to)
* Buzzer
* IR
  + The user will need to enter a code for a function for a specific IR receiving unit
    - The codes can be looked up in an online database
    - (Possibly in Pronto Hex? A description of Pronto Hex can be found [here](http://www.remotecentral.com/features/irdisp2.htm). The benefit is that the Arduino doesn’t need to understand IR protocols, only send raw data. The downside is that the codes are very large)

1. **Trigger and Thresholds**

* All sensors have multiple responses, each with their own threshold (2 Thresholds each should be enough)
* There are different types of triggering events
  + Rising Edge – The response occurs **once** when the signal rises from low to high over the threshold
  + Falling Edge – The response occurs **once** when the signal falls from high to low over the threshold
  + Above Threshold – The response occurs **continually** for as long as the signal is above the threshold
  + Below Threshold – The response occurs **continually** for as long as the signal is below the threshold
  + Held Above – The response occurs **once** when the signal stays continually over the threshold for a given period of time. When the signal falls back down, it will **not trigger any falling edge** responses
  + Held Below – The response occurs **once** when the signal stays continually under the threshold for a given period of time. When the signal rises back up, it will **not trigger any rising edge** responses
* The different triggering events can facilitate a larger number of responses from the limited number of inputs that we have.
* To create a joystick, two sensors are needed. One to move the mouse left and right and one to move it up and down. The LR sensor will have two triggers, one BELOW THRESH and one ABOVE THRESH, the UD will have the same.
* To create a button with a short response and a long response, the sensor will have two triggers, depending on the OFF state of the button.
  + If the button in the OFF state is a digital ‘0’ and the ON is a digital ‘1’, then the triggers will be a HELD ABOVE and a FALLING EDGE.
  + If the OFF state is a digital ‘1’ and the ON state is a digital ‘0’, then the triggers will be HELD BELOW and RISING EDGE

1. **Communication Protocol**

We can use the same communication protocol as in the previous version with some alterations: each sensor’s commands should be all together, the triggers will be different.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **‘0’** | This sets the Sensact in Configuration Mode. The 0 must be followed by the data for the Sensact sensors. All data is sent separated by commas.  Since each sensor contains multiple triggers, we put those triggers back to back. One trigger will consist of 4 groups of data   |  |  |  |  | | --- | --- | --- | --- | | Threshold Level | Trigger Event | Response Type | Detail |   Threshold Level is an integer between 0 and 100.  Trigger Event is an integer between 0 and 5 based on the following table.   |  | | --- | | 0 – Rising Edge | | 1 – Falling Edge | | 2 – Above Threshold | | 3 – Above Threshold | | 4 – Held Above | | 5 – Held Below |   Response Type and Details are based on the following table   |  |  | | --- | --- | | **Response Type** | **Detail** | | 0 - Relay A | 0 | | 1 - Relay B | 0 | | 2 - Bluetooth HID | Keyboard ASCII character | | 3 - Keyboard | Keyboard ASCII character | | 4 - Mouse | 0 – Move Left  1 – Move Right  2 – Move Up  3 – Move Down  4 - Click | | 5 - Buzzer | 0 | | 6 - IR | Function code |   The full configuration package would be arranged as such (for ‘SenX\_Y’, X is the sensor number, Y is the trigger number):  0, Sen0\_0 Threshold, Sens 0\_0 Event, Sen0\_0 Response, Sen0\_0 Detail, Sen0\_1 Threshold, Sens 0\_1 Event, Sen0\_1 Response, Sen0\_1 Detail, … , Sen1\_0 Threshold, Sens 1\_0 Event, Sen1\_0 Response, Sen1\_0 Detail, … , etc. |
| **‘8’** | Displays the current configuration data stored in the Sensact. Also sets the Sensact to Configuration Mode.  The data will be sent over the serial port in the same pattern as described above, but instead of a leading ‘0’, the data will be proceeded by ‘9999’.  9999, Sen0\_0 Threshold, Sens 0\_0 Event, Sen0\_0 Response, Sen0\_0 Detail, Sen0\_1 Threshold, Sens 0\_1 Event, Sen0\_1 Response, Sen0\_1 Detail, … , Sen1\_0 Threshold, Sens 1\_0 Event, Sen1\_0 Response, Sen1\_0 Detail, … , etc. |
| **‘9’** | Sets the Sensact to Run Mode. |

1. **Arduino Code**

* When the Sensact is in Configuration Mode, it will not activate any of the responses to triggers and will report the sensor values (0-100).
* When the Sensact is in Run Mode, it will activate responses and not report any sensor values.