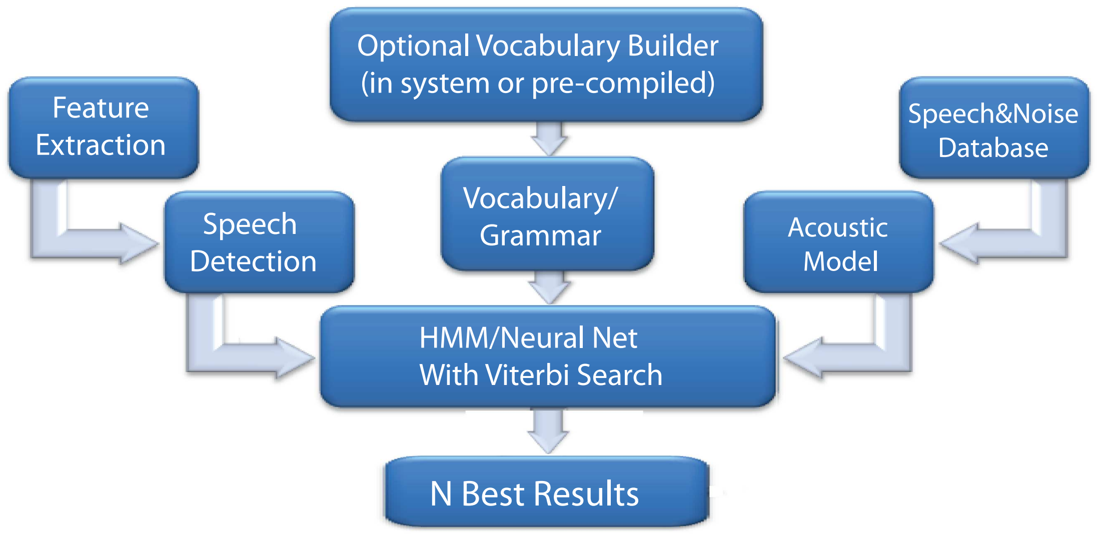
Fluent Soft Data Sheet



Sensory's FluentSoft technology and SDK's enable small footprint speaker independent speech recognition on embedded platforms. Small to medium sized vocabularies (under 1000 words) can be pre-compiled outside of the system for the smallest possible memory footprint, or a dynamic vocabulary build can occur within the system, so words can be added on the fly.

Included in the FluentSoft libraries is the revolutionary and award winning TrulyHandsfree™ Voice Control, a phrase spotting technology known for fast response, low current consumption, and excellent performance even from a distance or in noisy environments. This technology is an excellent solution for full featured voice control including the device being always on and listening for a "wake up" trigger so no button pressing is required, as well as for highly noise robust and reliable command sets that follow to provide product control. TrulyHandsfree™ Voice Control enables a long sought after level of functionality, safety, and convenience for a broad variety of devices that can be operated by voice alone.

With the FluentSoft SDK, TrulyHandsfree™ Voice Control may be fully custom speaker independent triggers and product control commands, or provide user defined triggers via a simple user enrollment procedure on the application processor (ARM-Android, x86-Windows, etc.). User defined triggers can also be used with Sensory's voice biometric technology to implement Speaker Verification (SV) for a single user, or Speaker Identification (SID) to select from a group of enrolled users. Triggers, product control commands, SV or SID can be executed on either the applications processor or a low power DSP. Sensory's voice biometric technology demonstrates best in class Equal Error Rates (EER) in quiet and in noise.

Product command and control with TrulyHandsfree™ Voice Control in the FluentSoft SDK creates great user experiences with natural language interface grammars utilizing word and phrase spotting to allow the user to say commands flexibly and in the natural way the user would like to speak. Applications can also utilize voice prompts supported bythe FluentSoft SDK to further engage the user and add to the experience.

Software Development Kits & OS Support

FluentSoft SDK's sell for $2500 and include:

- Speech technology library
- Reference code samples
- Documentation for reference and speech tuning samples
- Technology API
- 5 hours of support

The SDK functions include abilities to:

- Compile Vocabularies. Text or grammar based vocabularies can be defined in advance or created on the fly from incoming text streams with dynamic databases and building of vocabularies. FluentSoft works with or without a file system so vocabularies can be built and stored in files or in memory.
- Perform Speech detection. Reduces resource utilization by filtering when to activate the recognizer
- Perform Acoustic Model ("AM") selection. In addition to language selection the FluentSoft SDK allows the user to make critical footprint vs speed tradeoff to determine the size of the neural net (the neural net is used as a probabilistic means to replace a Gaussian Mixture Model and thereby reduce size from the traditional Hidden Markov Modeling approach).
- Configure the recognizer. Recognition parameters can be tuned to effect accuracy, speed, and confidence levels. Configurable recognition and detection parameters allow the user to improve performance under a variety of conditions.

Memory requirements

- RAM memory is used for calculations and storage during feature extraction, speech detection, and running the HMM recognition with Viterbi search. A single TrulyHandsfree trigger can use as little as 4KB for a single trigger phrase or larger vocabularies can use hundreds of KB. If the vocabulary build function is done on the system then a minimum of 30KB RAM is required.
- Flash or ROM for code can be used to store vocabulary data and grammars, acoustic models, and the speech recognizer. Code requirements for the recognizer can be as little as 40KB. Constant data for acoustic models, vocabularies and grammars are proportional to the number of words and languages. A minimal size could be 20KB for a single phrase, 10 phrases could be 40KB or 500 words could be ~500KB.

	Standard Embedded	Deeply Embedded		
		Speaker Independent Trigger/Commands	Single User Trigger+SV	Additional Users SV or SID
OS	iOS, Android, Linux,	customized		
	X86, WinPhone, QNX			
RAM Memory				
1 phrase (trigger)	26KB	8KB	8KB	+1KB/user
10 phrases	52KB	22KB	n/a	n/a
500 phrases	325KB	100KB	n/a	n/a
ROM/Flash				
Recognizer Code	100KB	20KB	25KB	
Recognizer Const. Table	n/a	10KB	15KB	
Constants - AM & Vocab				
1 phrase (trigger)	60KB	20KB	20KB	+10KB/user
10 phrases	110KB	40KB	n/a	n/a
500 phrases	700KB	500KB	n/a	n/a
MIPS				
1 phrase (trigger)	40	9	10	+2/user
10 phrases	90	35	n/a	n/a
500 phrases	200	75	n/a	n/a

SENSORY®

Note: Footprint estimates are based on a 5 syllable phrase.
Acoustic models (AM) and vocabulary data are required for each language. Most languages require slightly less data than English, but this can vary.

For further power reduction, deeply embedded applications may use Sensory's sound detector as a front end to the recognizer. The sound detector will monitor the environment and only wake the recognizer when there is some chance of ambient speech. The sound detector requires 500 KIPS, 1KB of code, and a 10KB RAM buffer of audio history to initialize the recognizer upon waking.

MIPS requirements and power consumption

MIPS required can range from 9 to over 100 depending on the complexity of the task. For low power TrulyHandsfree™ key word spotting, where the goal is minimizing power consumption, a system can be designed to use just 9 MIPS, and with a good system design with minimal hardware activated and a digital microphone a <3mA current is realistic. Sensory is working on algorithmic and hardware based approaches to lower this to ~1mA.

- **Audio capture SNR (mic + ADC)**
  * 68dB SNR

Language Availability for Speech Acoustic Model

US/UK/AU English	EU Italian	Mandarin	Turkish
NA/EU Spanish	EU/BR Portuguese	Japanese	
EU/CA French	Dutch	Korean	
EU German	Russian	Arabic	

*Other languages to be introduced

Standard & Deeply Embedded Implementations

Sensory has SDK's available for the popular standard OS's including Linux, Android, iOS, and more. Sensory can also run deeply embedded implementations to lower the MIPS, memory and current consumption or just to fit on a more cost sensitive platform. Current deeply embedded environments include Sensory's NLP and RSC chips, CSR's BC05 and 8670 Bluetooth chips, and Tensilica's DSP cores, as well as several popular, mobile platform audio codec DSP chips. Contact Sensory Sales for the current list of ports.

Accuracy

Accuracy can vary widely with vocabulary size, vocabulary words, grammar specifications, noise conditions, accents of users, distance from mic to speaker, and technology used. The TrulyHandsfree approach is best for rejecting the wrong words and performing well in higher noise or far field situations.
A sample accuracy curve for a single TrulyHandsfree trigger showing false accept/false reject(FA/FR) trade off in different noise conditions for the phrase "Hello BlueGenie" is shown on the following page.

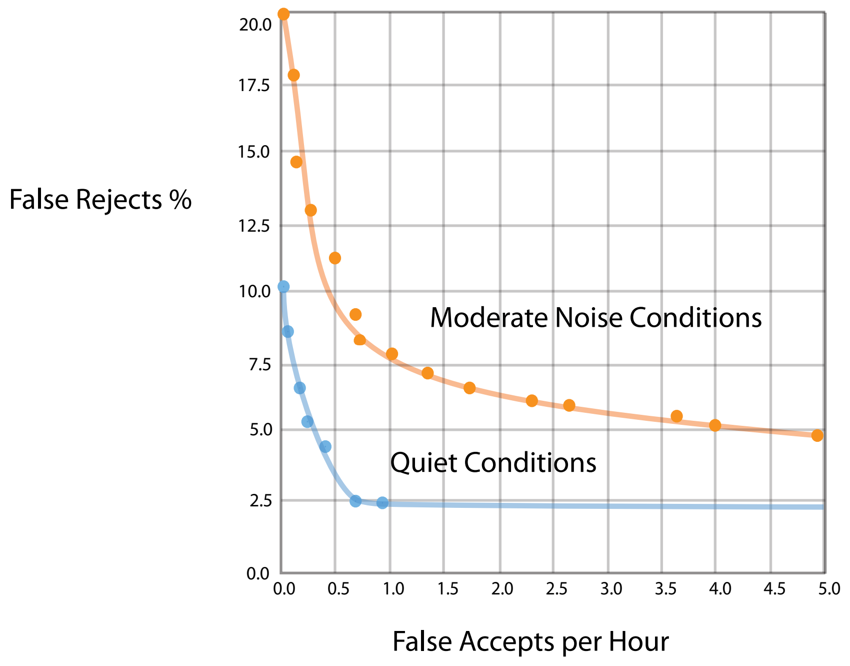
Test data comprises of two sets:

1. In-vocabulary data: ~250 recordings of "hello blue genie" used to calculate the false reject (FR) rate.
2. Out-of-vocabulary data: ~23 hours of conversational speech data used to calculate the false accept (FA) rate. TO BE CLEAR, TESTS ARE DONE WITH ON GOING AND NON STOP CONVERSATIONAL SPEECH DATA TO CALCULATE FA. In the real world we might expect less speech in the background. Obviously if the background was "quiet" there would never be false alarms.

Two test cases are shown :

1. 'Quiet Conditions' means no noise is added to the "hello blue genie" recordings
2. 'Moderate Noise Conditions' means out-of-vocabulary noise is added to the "hello blue genie" recordings at a controlled signal-to-noise ratio (SNR). This case simulates mixed noise conditions by combining results for SNR=10 and SNR=infinity (i.e., quiet), where SNR is defined as the ratio of 'peak' speech level over average 'noise' level.

PLEASE NOTE: Sensory's definition of SNR uses 'peak' speech level for the signal rather than 'average' speech level making it a more severe noise condition (i.e., more difficult) than avg/avg or peak/peak.



Business Model and Costs

Sensory's FluentSoft SDK's cost $2500 and are valid for a 6 month development window; SDK purchasers need to sign a simple 1 page development agreement. Customers with plans to go to market would also sign a License Agreement within this 6 month window. Custom deeply embedded ports can be done by Sensory with an NRE fee. Sensory typically receives a per unit royalty but we are open to considering other business models.

SENSORY®

Answers to Frequently Asked Questions

- **Preferred DSP Architecture(s)**

We prefer a DSP that supports bit-field manipulation, 8 and 16-bit addressability of data to avoid packing and unpacking. Otherwise we have to tradeoff MIPS and memory. We also prefer intrinsics and/or library modules that efficiently process FFT and Signed Vector Inner Product functions.
It's very important that the DSP and it's support memory be designed for lowest power, which means the max clock rate in the design should not be more than ~50MHz. This reduces device sizes and therefore power. Preferably the chip on which the DSP resides will not have any blocks extraneous for DSP operation, and the DSP should be able to power down all unnecessary blocks to very low quiescent current when running the TrulyHandsfree™ algorithms. The mic+pre-amp+ADC should be very low power designs. Ideally the entire system current consumed when running the TrulyHandsfree trigger (including audio front end, power regulation, DSP and it's memory) will be ~1-2mA.

- **Precision (Fixed/Float/Integer)**

Integer

- **Block Size**

We prefer 15ms blocks of 16KHz sample rate, 16 bit audio data. We can accept blocks of other sizes, and buffer to create the block size we need, but that increases the memory requirement. The time slice architecture for VR algorithmic processing works most efficient when we have a 15ms time slice.

A couple of other questions:

1) Sample rate: What are the trade-offs of running it at 8 kHz vs. 16 kHz?

8KHz sample rate input will result in ~10% increase in false rejects versus 16kHz; potentially more for females due to the impact on higher frequencies

2) Model sizes for phrase spotting and speaker verification: Assume that we would run Speaker Verification and potentially allow user-defined phrases (for preference or to support different languages). We would presumably have to pass down the appropriate models for the user and those phrases. Although potentially covered in the memory requirements above, please itemize the memory assumptions for the various models and including those for speaker verification.

The const data for trigger and trigger+SV+SID documented in the table above are for the acoustic model and vocabulary grammars, regardless of pre-programmed or user-defined.

However, during the user-defined enrollment process using a GUI at the applications level, the user-specific const data build process requires ~65Mbytes background noise data for automatically testing and tuning the trigger const, and ~15Mbytes anti-speaker data for tuning the SV+SID const. This is unique to a language at this point, but we are working on a language independent version and early results are encouraging. Note that the result of this build process are acoustic models and grammars of the same size as in the table.

3) Audio buffer: do the memory requirements in the previous table include an allocation for an audio buffer? If so, how much audio is buffered?

A buffer is not required if the deeply embedded application interface provides 15ms blocks.
However, a circular buffer may be desired to allow passing the audio up to a recognizer at the application/cloud level so that a seamless trigger-followed-by-search process can be supported. The size of that buffer depends mostly on the latency to wake up the host processor, pass the buffer up to host processor, and if necessary pass to the clouds. Sensory's technology is virtually instantaneous. This is an application decision, and is not required by Sensory's trigger.