# **Implementation of the Wake Word for Smart Home Automation System**

the wake word module should continuously listen and process the sounds from the environment, in order to detect the pre-defined wake word.

Once the wake word is pronounced by the user, wake word module should trigger the actual voice command processing, which will result in the action within the home automation system.

*Keywords*—*wake word; home automation; voice command*

## I. INTRODUCTION

Voice control module should convert the spoken command to text first, and then map it to the appropriate action in the HA system.

Namely, the goal is to minimize the number of cases in which the action in the HA system is triggered without the actual user intent.

This is impossible if the voice command module continuously listens and processes the sounds from the environment

The more comfortable way for a user to address the HA voice control is to pronounce a pre-defined wake word, as a signal that the actual voice command is to follow.

In this paper, we will focus on the challenges of implementing the wake word detection mechanism as part of the HA voice control module.

On the other hand, Pocketsphinx is a voice recognition engine designed for embedded devices [5], and is available as a C library.

Also, Snowboy [6] is a highly customizable wake word detection engine that is compatible with embedded devices with low memory.

The two, Pocketsphinx and Snowboy are used and compare these two implementations in terms of accuracy and performance.

**W**

**AKE**

**W**

**ORD**

**D**

**ETECTION**

**C**

**OMMAND**

**R**

**ECORDING**

**C**

**OMMAND**

**E**

**XECUTION**

Keyword found

Stop Decoding

PCM data

**S**

**PEECH**

**TO**

**T**

**EXT**

**E**

**NGINE**

Command (Text)

Fig 1. Voice command processing module

### II. VOICE CONTROL SYSTEM ARCHITECTURE

The voice control module is represented in Fig. 1. The wake word detection component continuously records and processes the sounds from the environment.

Once the predefined wake word is pronounced by the user, it triggers the actual command recording. The recorded command is then processed by the speech-to-text engine. Textual output of the

engine is finally converted into MQTT protocol [8] commands for the HA system by command execution module.

### III. IMPLEMENTATION OF WAKE WORD DETECTION

Continuous recording of user’s voice is implemented through the background thread.

The recorded audio from environment is stored in the PCM buffer, and a window of ~2s of duration is analyzed by the wake word recognition engine.

When the wake word is detected within the analyzed data, the background thread triggers further command processing.

At the same time, this background thread stops processing the recorded PCM buffer data, and waits for command processing to end.

Once the command is executed, the background wake word detection thread is started again, and PCM buffer data is analyzed.

On the other hand, Snowboy is designed specifically for the scenario of wake word detection.

### V. CONCLUSION

In this paper, we analyzed two implementations of the wake word detection module, based on Pocketsphinx and Snowboy voice recognition engines. Experiments have shown that Pocketsphinx has better accuracy than Snowboy, but its performance is limited on RPI2. In the cases when the voice command processing module is performed on a more powerful device, Pocketsphinx implementation is preferred, due to the better accuracy. However, for the embedded device scenario, Snowboy is a better solution, due to its satisfactory performance.