A Testbed for Adaptive Microphones in Ultra-Low-Power Systems

Evan Fallis*, Mark Lipski*, Andrew Mackey*, Marc Jayson Baucas*, Matthew James**,
Petros Spachos*, and Stefano Gregori*
*School of Engineering, University of Guelph, Guelph, Ontario, Canada
**Kapik Integration, Toronto, Ontario, Canada

***Abstract*—Smart cities bring new technological advances that help improve everyday life. One such improvement is the ability to map out a city based on a characteristic. The amount of acoustic noise in an environment has many impacts on human life and has the potential to be collected wirelessly. Unfortunately, systems made today would not have the battery life capabilities to handle such a high demand if continuous transmission was used. In this paper, the design of a testbed for a smart microphone system is presented. In order to promote power savings, an analog-to-digital converter (ADC) which dynamically switches between high and low power modes in response to environmental noise is presented. Specifically, the high power ADC mode is triggered from a spike in the acoustic noise level. Ideally, this would be configurable and allow for detection of different types of sound such as human voice or music. A framework for basic environmental sound collection is presented along with preliminary results of the testbed.**

I. Introduction

In an era where information and communication is the main driving force for progress, many cities attempt to stay on top of the technological ladder. These cities that enhance their internal services based on technological advances are called smart cities. Smart cities enable the introduction of new technology to the public through different services that optimize resource usage, transportation, utilities and more. In a smart city, large amounts of data is collected and used for improvements to its infrastructure [1].

One example of a data type that can be utilized by smart cities is audio data. Noise levels can determine the amount human comfort in different areas within a city. By creating a tool that can visualize the levels of acoustic noise around each street and corner, the city and its people are given the ability to react and adjust accordingly. Not only will noise visualization indicate the loudness of an area, but it can also show the type of sounds present. An example is the mapping of varied sounds such as human voice, traffic, construction, or air transport that is currently present in any school or hospital. Mapping the amount of sound classified as human voice within an area grants the ability of the public to observe the amount of human presence at the time. This can also help citizens to determine if a place within the area is comfortable to go to at a specific time.

Analyzing the audio data in an urban environment can become complicated depending on the location within the city [2]. There are a variety of challenges such as privacy and placement. Privacy has always been a big issue in regards to the collection of data in a public setting. An invasion of privacy could be claimed by anyone who feels uncomfortable about the idea of the voices being recorded.

Another challenge is the reliability of the data, which depends on its accuracy in modelling the noise movement within an area. Data is considered reliable if the extraneous variables that could skew the noise model of the area is minimal. A major concern is the power consumption of various microphones and analog-to-digital converters (ADCs) when collecting audio data [3]. This is especially true when the microphone is set to sample throughout the day, meaning it will have minimum downtime.

In this paper, a technique for reducing power consumption in MEMS microphone ADCs for an acoustic environmental monitoring system is presented, along with a testbed to measure the power savings.

The rest of the paper is organized as follows; Section II reviews the related works. Section III discusses the proposed system followed by Section IV which presents the results found during analysis. The conclusion is in Section V.

II. Related Work

This paper addresses the power consumption when processing data through the use of an ADC. The work in [4] outlines future challenges in terms of designing ultra-low power ADCs. It specifically discusses voltage margins, noise, linearity, and oversampling. In [5], the authors present an approach to ADCs that involves predicting the next value. This may not only increase the conversion speed of already in place ADC topologies, but also reduce the power consumption as well. Authors in [6] describe a successive approximation ADC which targets low power wireless sensor systems. They claim the ADC only takes 2 μW and achieves a signal to noise and distortion ratio of 64.42 dB using 0.35 μm complementary metal-oxide-semiconductor technology. The work in [7] presents a design for an ADC that focuses more on minimizing power consumption rather than the resolution of the ADC. The resolution is limited to 6 bits and it spends less than 8 pJ of energy per bit.

The other concern this paper attempts to address is finding an effective way to collect and handle the sound data while being able to integrate it into the design. The authors in [8] discuss a mobile system dedicated to the sensing of noise pollution in urban environments. It focuses on determining the level of attenuation as part of the

sound collection process in addition to gathering the actual audio levels. The work in [9] specializes in the tracking of audio data with an emphasis on vehicles. The data being sent is based on a certain threshold value which will restrict any data below it. In [10], a factory was examined and profiled based on the acoustic levels. Their goal is to reduce the amount of noise pollution by redesigning the air compressor. The work presented in [11] highlights an improved way to collect audio data for voice activity detection. It uses both audio and visual signals with a supervised learning algorithm to detect what audio frames correspond to human voice. The authors in [12] use a Gaussian mixture model based classification approach to speech detection. It's trained using spectral flow direction, a novel feature for this type of classification. In [13], a multistage system is developed to analyze environmental sound and classify it. The authors claim to achieve an accuracy higher than 90% for audio concept identification.

There have been a few other papers that involve a wireless sensor network for the purposes of audio collection. A similar implementation has been developed in [14]. The work focuses on developing an ultra low power sensor node that is designed to wake up periodically to collect data, it was programmed to wake for ten minutes every hour. Although this may be sufficient to get the overall average noise level during a day, it can easily miss many important events since it is only awake less than 20% of the time. The authors in [15] developed an Android application for the use of audio data collection. It was able to collect many data points and plot them on a map to create a visual dashboard. There is promise in collecting audio data by utilizing a large user base, but this method limits the amount of data by requiring human interaction to characterize an area. The work in [16] presents a node that collects various multiple types of data. It is designed to wake up based on a timer interrupt or when the accelerometer is triggered. This implementation is intended to focus more on the individual using the sensor rather than mapping a location itself.

In comparison with these works, this paper introduces a testbed design that is fully portable, wireless, and conserves power by using a novel technique. Specifically, this testbed utilizes a wake up circuit to conserve power when minimal audio activity is present. In addition, this design is scalable and allows for audio data collection through simple electronic components that costs less compared to most sound data designs that were cited.

III. System Overview

A. System Framework

The system being built is designed to wirelessly send audio data through Wi-Fi to a remote server while minimizing the power needed by switching components to low power mode during operation. This section introduces the components of the proposed system. This includes the hardware, the sensing devices, the software management, and the application interface. The initial design of the testbed can be seen in Fig. 1, with seven major components:

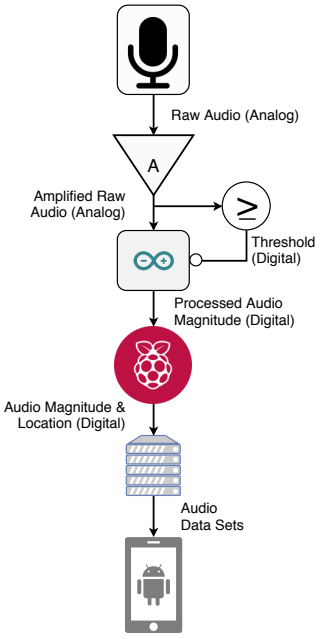

Fig. 1: System framework.

- MEMS Microphone
- Variable Gain Amplifier
- Threshold Detector
- Arduino
- Raspberry Pi
- Server
- Android Phone

The MEMS microphone is used to collect raw audio data. Typically, the output of a MEMS microphone is too small to record with most ADCs. There is usually an amplification process to allow for accurate readings of each sample. A variable gain amplifier was chosen to find the optimal mix of gain and Signal-to-Noise Ratio (SNR). The threshold circuit is used to control the status of the ADC based on an interrupt pin. The Arduino is the chosen microcontroller used for its ADC to quantize the analog signal. It is also capable of performing light signal processing on the data being read. This is then connected to the Raspberry Pi, which has built-in Wi-Fi capabilities. The Pi also allows for scripting when sending data online, as well as location services. The server collects data from the Pi and stores it. This can then be visualized both on a page hosted on the server, and an Android application.

B. MEMS Microphone

For the purposes of the project, a MEMS microphone was selected to keep cost, area and power low. The MEMS microphone chosen was the PUI PMM-3738-VM1010-R, for its low power and sound threshold detection [17]. The microphone can be set to low power mode digitally, it then wakes from sleep mode when the audio levels exceed a certain sound level. The current consumption of the microphone is approximately 80 μA while active, and approximately 10 μA in low power mode. The supply voltage of the microphone is rated between 1.6 V and 3.6

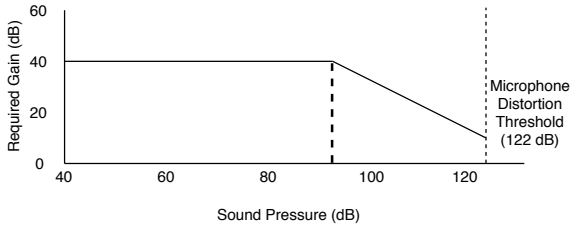

Fig. 2: Graph of desired amplifier gain vs sound pressure.

V. The SNR of the microphone at 94 dB is 60 dB, with a bandwidth of roughly 20kHz. If frequencies above 5kHz are not desired, they could be filtered to improve the SNR of the microphone by 3 dB, raising the SNR to 63 dB at audio levels of 94 dB. The existing VGA used has a pole at 4 kHz, filtering out some of the higher frequency noise.

C. Variable Gain Amplifier

The PUI audio sensitivity is -38 dBV @ 94 dB, indicating that an audio level of 94dB would have an output voltage of 12.5 mV. Given that the Arduino ADC operates at 10bits over a 5 V range, the smallest measurable voltage difference is 5 mV, meaning that the microphone output cannot be measured with more than 2 bits of precision. The output of the microphone must then be amplified to make it possible to read in the data on the ADC. The gain required is based on the desired precision of the audio, in addition to the noise level of the audio level itself. The audio level will clip at a minimum of 4 dBV, and should not exceed that (a safe margin would be 0 dBV). The desired amplifier gain can be seen in Fig. 2, indicating a maximum required gain of 40 dB and a minimum of 5 dB. The amplifier should be linear enough to avoid distortion, and have a high enough gain to make the signal measurable. The amplifier was designed using a low power, low noise op-amp, with a digital potentiometer in the feedback network, seen in Fig. 3 as R_{pot}. In Fig. 3, the C_1 and C_2 are coupling capacitors, R_1 and R_2 are used to bias the input, while the ratio between R_{FB} and R_{pot} determines the gain of the amplifier. To provide additional filtering, a capacitor can be added in parallel to R_{FB}, and a capacitor can be added to the output node to add an additional pole to the op-amp. The op-amp used was the AD8603 from Analog Devices, and the potentiometer used was the Microchip MCP4013 [18] [19]. The resulting amplifier has a programmable gain between 5 V/V and 300 V/V. The input referred noise of the amplifier was found to be negligible relative to the noise from the microphone.

D. Threshold Circuit

The threshold circuit was designed by exploiting the non-linearity of amplifiers. The threshold circuit is comprised of a transistor biased in the near sub-threshold region, or saturation region with a low overdrive voltage. These bias conditions create a large non-linear component with respect to the current through the transistor.

$$I_{DS} = \frac{k'(V_{OV} - V_{IN})}{2} \tag{1}$$

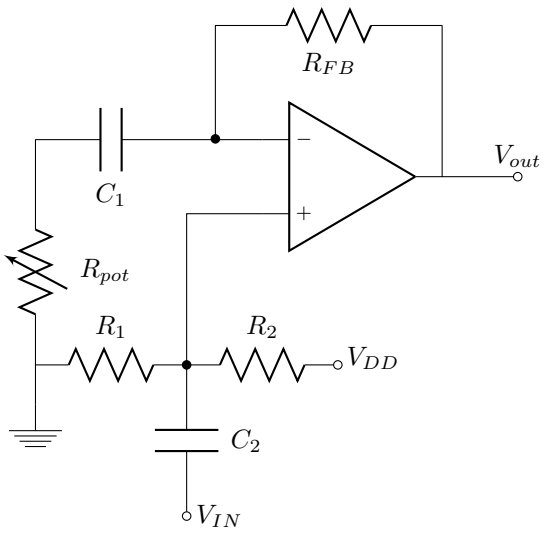

Fig. 3: Variable gain amplifier schematic.

Equation 1 shows the current behavior based on the overdrive and input voltages. Given the square term, the amplitude of V_{IN} gets converted into a DC component decreasing the average output value of the circuit. Provided the AC components are filtered out, the average value of the circuit in question will decrease corresponding to the average power of the input signal. Once the amplitude of the input signal exceeds a threshold, the output of the amplifier will drop below the output from a logic high to a logic low.

E. Arduino

Now that the analog audio data is amplified and can be sampled, it should be recorded. To capture and save the audio data as a digital signal in the testbed, an ADC needs to be used. ADCs are used when data from an analog medium needs to be translated to a digital one. They are limited to a finite resolution and sample rate. An Arduino Uno hosts the ADC being used in this testbed, having a resolution of 10 bits, a sample rate of approximately 10 kHz, and using a successive approximation register style ADC [20]. This was chosen due to its ease of use, portability, and dependability.

The Arduino is programmed to sample the data and average a set of maximum values within a certain time frame to form each data point. This will allow the user to see the magnitude of sound in the environment based on a specified sample. As mentioned, there are privacy concerns with any audio collection system. Taking the magnitude helps bypass any privacy concerns as the data being transmitted can not be converted back into human speech. After the sample is processed, it is written to the serial port of the Arduino. The data is then transferred to a Raspberry Pi 3 using a Universal Serial Bus (USB) connection.

One feature of the Arduino Uno is an interrupt based on an input pin. This allows a digital signal to control when the Arduino enters sleep mode and when it is active. For the purposes of this experiment, the interrupt pin will be connected to the wake up circuit.

F. Raspberry Pi

One of the major advantages of this prototype is the ability to be flexible in terms of placement. Wireless technology bypasses the need to be placed inside of a building. By default, the Arduino Uno has no Wi-Fi capabilities unless an external shield is connected. This makes it impossible to store the data wirelessly to a server without the use of external equipment.

To solve this problem, the Arduino is connected to a Raspberry Pi, allowing for internet access through the Pi's Wi-Fi [21]. Raspberry Pis are very useful for applications where the processing demand is low. It allows the testbed to be more robust and portable when compared to a standard computer. As mentioned, an Arduino Uno is connected to the Raspberry Pi 3 using USB. The Raspberry Pi runs on Rasbian 8 and the firmware is version 4.14.79-v7+. It is set to run a Python script which collects data and creates a Uniform Resource Locator (URL) request. This also allows location data to be sent, letting the user see which node has sent the data set.

G. Server

To securely record the data, a private server is needed. This will provide a location to store and analyze the collected data, as well as host code to allow the user to visualize the streaming audio data. The server being used runs Ubuntu 16.04.2 and hosts a PHP (Hypertext Preprocessor) script. This script collects web requests and appends to a Comma Separated Value (CSV) file, representing both the audio level of the environment, and the location of the node in terms of longitude and latitude.

The server hosts a JavaScript page which uses the Google Charts Application Programming Interface (API) to render the data points of each set. This will execute whenever a user navigates to the web page on their local computer. Real time plotting of audio data can be very useful for a live map of the city. Additionally, having the previous data sets available will allow users to view any trends in the audio data such as weekly events.

H. Mobile Application

A critical aspect of the testbed is the interface and visual representation of the audio data. In order to achieve this, an Android mobile application was made. Android applications allow for open source development and simply integrate the testbed into the vast majority of the mobile market. This allows for the presentation of testbed data to be highly scalable and easily reproduced. The testbed utilizes a Nexus 5 running Android 6.0.1 but is capable of supporting any Android device with software version 4.0 or greater.

The application is comprised of 3 visual activities.

1) Audio Level Heat-map
2) Audio Classification
3) Node Management

The audio classification expands on the audio data collected, presenting more useful information to the user of the application, but at the cost of reduced privacy. When the high power ADC is triggered, a more complex detection scheme could differentiate between sound sources and types based on unique wave characteristics. This information could then be presented to the mobile user by associating a time-stamp, location, and category with the measured sound.

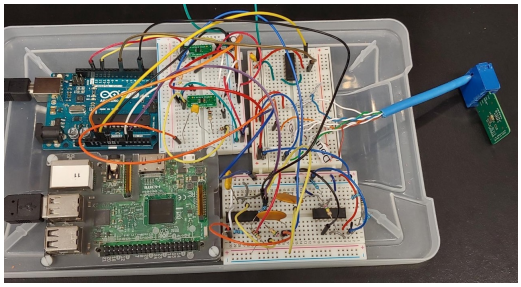

Fig. 4: Initial prototype.

Finally, the testbed application could provide management capabilities for the individual audio nodes of the testbed; reporting on individual location, power consumption, and audio activity rates. Management could be extended to include the control of adding and removing nodes to the testbed.

The Android application has no effect on the energy efficiency or function of the adaptive ADC, but remains essential for simple, real-time visual representations of the testbed data.

IV. Preliminary Results

The fully realized prototype for sound level collection can be seen in Fig. 4. This includes the microphone, amplifier circuit, threshold circuit, Arduino, and Raspberry Pi. Figure 5 shows a sample graph that is plotting the audio data after being transferred wirelessly to the server. This allows a user to easily observe and analyze the data. Although the data has not been normalized, it can still be used to determine the relative acoustic noise level based on the minimum and maximum values of the arbitrary units.

To measure the power consumption of the testbed, a Monsoon Power Monitor was used. Unlike most power measurement devices, the Monsoon is able to get very accurate readings as it will supply power to the entire prototype [22]. Figure 6 shows a visualization of the data collected from the Monsoon for the Arduino. One important note is that this graph includes all components being powered by the Arduino. The transition to active mode happens almost immediately, seen by the spike in power.

Based on initial measurements, the prototype used approximately 202.82 mW of power when collecting audio data. A summary of power usage can be seen in Table I. This includes the power of the microphone, amplifier, and Arduino. The Arduino in particular is the main source of power savings currently. This is done by utilizing a pin interrupt to wake up the Arduino when the audio threshold is broken. When the Arduino is in sleep mode, it uses a considerably smaller amount of power. It lowers the power

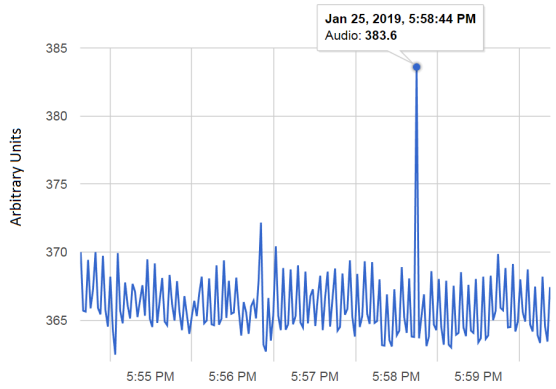

Fig. 5: Sample audio graph data.

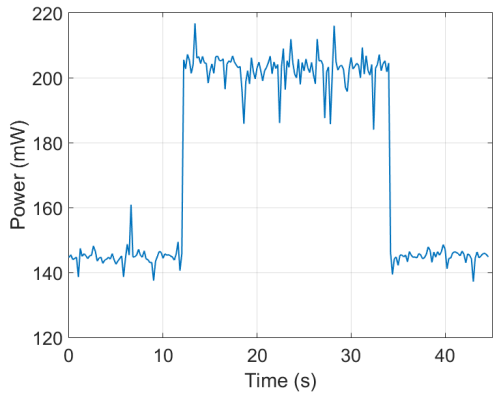

Fig. 6: Audio acquisition power usage.

Component	Microphone	Amplifier	Threshold	Arduino	Total
Power [mW] (Active)	1.41	0.13	0.08	201.20	202.82
Power [mW] (Sleep)	1.41	0.13	0.08	143.32	144.94

TABLE I: Testbed power consumption.

usage of the Arduino from 202.82 mW to 144.94 mW. It is estimated to reduce the power consumption by much more when implemented on an Application Specific Integrated Circuit (ASIC) or a more custom microcontroller.

An Android application named Phyphox was used to validate the results being shown. Phyphox allows the user to record audio data and export to a usable data format. The smartphone used to run the application was a Galaxy Note 3 (Android 5.1.1) and an LG G7 (Android 8.0.0) was used to play the audio file. Throughout the experiment, a distance of 10 cm was maintained between the smartphone speaker and the microphone. The volume of the speaker was held constant at a fraction of $\frac{9}{15}$ with respect to the maximum volume for each test. To convert the magnitude of the audio to decibels, the prototype was calibrated. The calibration resulted in the following quadratic expression:

$$decibels = -0.00376124x^2 + 3.43177x - 693.237 \quad (2)$$

where the variable x corresponds to the value seen by the ADC, no intermediate conversion to voltage is necessary.

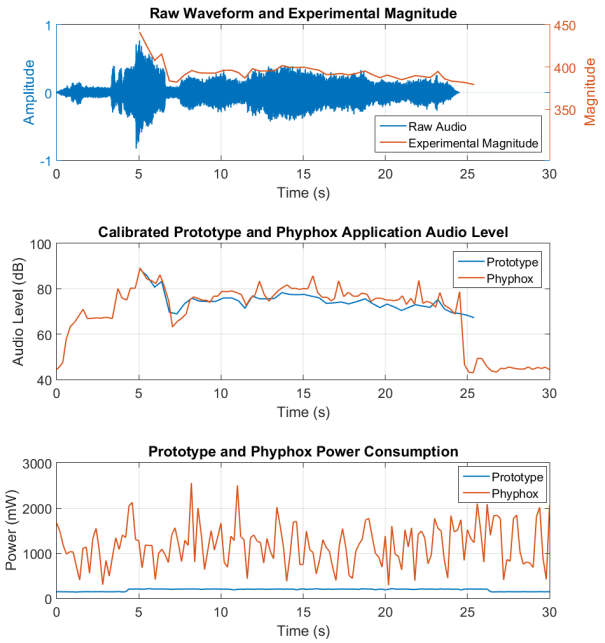

Fig. 7: Comparison of raw data (top), sound level (middle), and power (bottom).

To get accurate power readings on the Note 3 smartphone, the battery was bypassed and connected to the Monsoon instead. A voltage of 4.2 V was being utilized to power the smartphone during the test. The Note 3 was seen to consume 0.4 W of power under the standard conditions of operation. The standard configuration was defined as the following:

1) Screen on
2) Brightness at minimum
3) Wi-Fi, Bluetooth, and LTE turned off
4) Idle display on the Phyphox application

The audio recording test involved starting the Phyphox application and displaying the data being collected. This took an average of 1.59 W when streaming the audio level continuously. This shows that the audio acquisition took approximately 1.19 W of power when accounting for the power consumption of the phone without collecting data. When compared to the prototype, this is a much higher power way to collect audio data and therefore is not feasible to deploy on a large scale.

Figure 7 gives details of the audio data and how closely it captures traits of the signal. The experimental magnitude determined from the prototype traces the audio data fairly accurately.

The comparison between the prototype and the Phyphox Android application sound levels can also be seen. This was after the prototype data had been normalized to represent the audio level in decibels. The prototype woke at the time of 5 seconds after starting the experiment based on the audio threshold detector, this explains the initial gaps in data when compared to Phyphox. The prototype went into sleep mode shortly after the audio clip was finished playing.

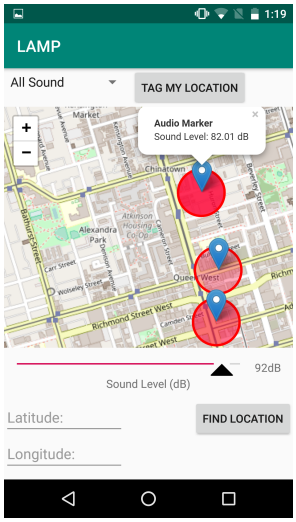

Fig. 8: Android application: audio level heat-map.

It can be seen that the prototype follows the app relatively well although it has a slightly slower response to any audio impulses. This was deliberate since the target application is audio data collection for long periods of time. The figure also shows the relative power consumption, the act of sampling audio using an Android smartphone uses excessive power when compared to the prototype.

The main visual the mobile application provides is an up to date heat-map of location specific audio spikes/levels, shown in Fig. 8. When the high power ADC is triggered and begins to measure the audio levels, that data will be passed to the server along with its geographical location. The mobile application pulls this data from the server and plots a polygon on a map in its correct location. Moreover, a circle of some colour corresponding to an audio level range will plotted.

V. Conclusion

The end-to-end system presented in this paper demonstrates a novel way to wirelessly collect and visualize audio information. It uses a MEMS microphone to collect the sound pressure level at the first stage. The analog signal is then fed through a low noise amplifier and quantized using the ADC of an Arduino Uno. The Arduino is only ever active once a threshold has been reached based on the related circuit. The Arduino then sends the data through USB to a Raspberry Pi 3 which creates a web request to the server, allowing for data to be pushed and saved securely. From the server, an Android smartphone can be used to view the data and visualize the different sound levels across different areas on a map.

The relatively low cost of the prototype helps in situations where placement matters. There is less worry when putting the system in a potentially harsh environment due to the minimum cost of replacing it. The testbed allows for audio collection in urban environments which can be mapped to provide the acoustic profile of a city. The initial tests have shown that a smart microphone system design was able to save 50 mW when testing with an Arduino Uno.

References

[1] C. Xu, X. Huang, J. Zhu, and K. Zhang, "Research on the construction of sanya smart tourism city based on internet and big data," in *2018 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, Jan 2018, pp. 125–128.

[2] F. Rong, "Audio classification method based on machine learning," in *2016 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, Dec 2016, pp. 81–84.

[3] K. Shehzad, H. Kang, D. Verma, Y. J. Park, and K. Lee, "Low-power 10-bit sar adc using class-ab type amplifier for iot applications," in *2017 International SoC Design Conference (ISOCC)*, Nov 2017, pp. 224–225.

[4] N. Narasimman and T. T. Kim, "Design challenges for vco based adcs for ultra-low power operation," in *2013 International SoC Design Conference (ISOCC)*, Nov 2013, pp. 249–252.

[5] N. Wood and N. Sun, "Predicting adc: A new approach for low power adc design," in *2014 IEEE Dallas Circuits and Systems Conference (DCAS)*, Oct 2014, pp. 1–4.

[6] R. H. Gudlavalleti and S. C. Bose, "Ultra low power 12-bit sar adc for wireless sensing applications," in *2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA)*, Jan 2016, pp. 1–4.

[7] A. Agarwal, Y. B. Kim, and S. Sonkusale, "Low power current mode adc for cmos sensor ic," in *2005 IEEE International Symposium on Circuits and Systems*, May 2005, pp. 584–587 Vol. 1.

[8] I. Kirillov and V. Bulkin, "The mobile system of urban area noise pollution monitoring," in *2015 Second International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S T)*, Oct 2015, pp. 200–203.

[9] P. Patil, "Smart iot based system for vehicle noise and pollution monitoring," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, May 2017, pp. 322–326.

[10] J. Zhou, J. Zhao, and P. Li, "Noise pollution and control measures of rubber factory compressor room," in *2009 Third International Symposium on Intelligent Information Technology Application Workshops*, Nov 2009, pp. 241–242.

[11] D. Dov, R. Talmon, and I. Cohen, "Audio-visual voice activity detection using diffusion maps," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 4, pp. 732–745, April 2015.

[12] . Zubari, E. C. Ozan, B. O. Acar, T. Ciloglu, E. Esen, T. K. Ate, and D. O. nr, "Speech detection on broadcast audio," in *2010 18th European Signal Processing Conference*, Aug 2010, pp. 85–89.

[13] I. Feki, A. B. Ammar, and A. M. Alimi, "Audio stream analysis for environmental sound classification," in *2011 International Conference on Multimedia Computing and Systems*, April 2011, pp. 1–6.

[14] C. Peckens, C. Porter, and T. Rink, "Wireless sensor networks for long-term monitoring of urban noise," *Sensors*, vol. 18, no. 9, p. 3161, 2018.

[15] M. Zappatore, A. Longo, and M. A. Bochicchio, "Crowd-sensing our smart cities: A platform for noise monitoring and acoustic urban planning," *Journal of Communications Software and Systems*, 2017.

[16] V. Risojević, R. Rozman, R. Pilipović, R. Češnovar, P. Bulić *et al.*, "Accurate indoor sound level measurement on a low-power and low-cost wireless sensor node," *Sensors*, vol. 18, no. 7, p. 2351, 2018.

[17] Pmm-3738-vm1010-r. (retrieved: 2019-01-29). [Online]. Available: http://www.puiaudio.com/product-detail.aspx?categoryId=4&partnumber=PMM-3738-VM1010-R

[18] Ad8603. (retrieved: 2019-01-29). [Online]. Available: https://www.analog.com/en/products/ad8603.html

[19] Mcp4013. (retrieved: 2019-01-29). [Online]. Available: https://www.microchip.com/wwwproducts/en/MCP4013

[20] Arduino uno adc. (retrieved: 2019-01-29). [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

[21] Raspberry pi 3. (retrieved: 2019-01-29). [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[22] Monsoon Power Monitor. (retrieved: 2019-01-30). [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/