Proceedings of the International Conference on Industrial Engineering and Operations Management

Publisher: IEOM Society International, USA DOI: 10.46254/NA10.20250071
Published: June 17, 2025

Microprocessor-Based Simulation for the Study of Wind Speeds as a Function of Ventilator Position in a Model Building

Muhammad Ashhub Ali
Software Engineering Department
National University of Science and Technology
Islamabad, Pakistan
maali.bese21seecs@seecs.edu.pk

Anjum Ali
Former Professor of EE, FAST-NU, Lahore, Pakistan
Former Professor, ECE, KICS, UET, Lahore, Pakistan
Former Associate Professor of EE, Mercer University, MACON, GA, USA
Former Associate Professor of CSE, LUMS, Lahore, Pakistan
anjumatlanta@gmail.com, anjum.ali@nu.edu.pk

Abstract

This paper presents a microprocessor-based simulator used in the study of wind speeds in a model building. A 7mx5mx5m room has been used as a model building to study the wind speed as the 70cmx1m chimney (with a 30cmx1m ventilator) is placed at different positions on the roof. The size of the window in the room is 2mx3m. The height of the chimney is 12m from the top of the roof, or 17m from the floor. The position of the ventilator is changed, and the resulting wind speed is displayed on seven-segment displays in decimal format. Detailed schematics along with relevant flowcharts and the code for operating the simulator are also included as part of the paper.

Keywords
Wind Energy, Renewable Energy, Computer-Aided Design and Simulation.

1. Introduction
The "Venturi Effect" is the increase in the speed of a fluid when it passes through a narrow area (Cadence CFD). Many natural as well as man-made structures exhibit this phenomenon. Figure 1 shows the pictures from a real house, in which the speed of the wind, out of the gate, is naturally increased because of the geometry of the alley. This has served as a motivation to further explore different geometries where this phenomenon can be utilized for natural wind-speed amplification.

1.1 Objectives
The objective of this research is to develop a microprocessor-based interface and simulator for studying and displaying the wind speed corresponding to the position of the ventilator in a model building. This will help us to determine the optimum geometry of the ventilator along with the position of the chimney on which this ventilator is located, for the best wind speed amplification.

Figure 1. Pictures from a Real House where the Wind Speed in naturally increased at the Entrance of the Gate

2. Literature Review

A lot of work has been done to study the benefits and uses of renewable energy, including, but not limited to wind energy. The "Venturi Effect" is one of the phenomena that causes the increase in wind speed based on the shapes through which the wind is made to pass. INVELOX™ from SheerWind™ (2022) claims to implement this approach in their product. Also see Gohar GA, Manzoor T, Ahmad A, et al. (2019). However, there have also been objections to their claims. See Paul Gipe as an example. Another example is the UN Climate Change Conference (UNFCCC COP 29). COP, which stands for Conference of the Parties, is a series of formal meetings where governments assess global efforts to advance the Paris Agreement and the Convention, also limit global warming to 1.5°C as informed by the latest science. At COPs, world leaders come together to measure progress and negotiate the best ways to address climate change. There are now 198 Parties (197 countries plus the European Union) to the Convention, constituting near universal membership. COP29 was held in Baku, Azerbaijan, in November 2024.

3. Methods

A 7mx5mx5m room has been used as a model building to study the wind speed and wind patterns as the 70cmx1m chimney (with a 30cmx1m ventilator) is placed at different positions on the roof. The size of the window is 2mx3m. The height of the chimney is 12m from the top of the roof, or 17m from the floor. This is shown in Figure 2 below:

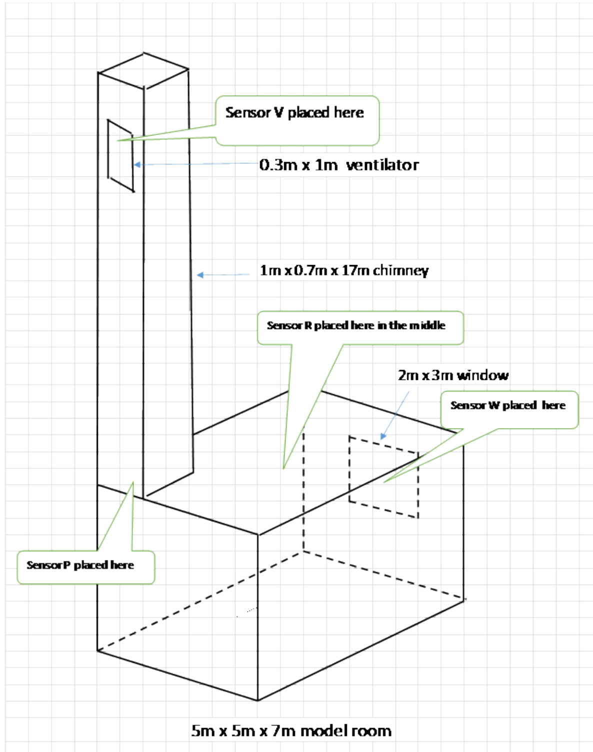


Figure 2. A 5m x 5m x 7m Room with a Chimney and a Ventilator

An 89C51 microcontroller has been used to calculate, simulate and display the resulting wind speed when the ventilator is placed at different positions.
A sensor, called V, is placed at the input of the ventilator, and a sensor, called W, is placed at the window. The sensor, called R, is placed in the middle of the room, and monitors the wind speed in the room. The position of the chimney is called P. The value of R is studied as P is varied, which corresponds to different placements of the chimney.
Nine discrete positions have been used for the chimney (and hence the ventilator) in this simulation, as shown in Figure 3 below.

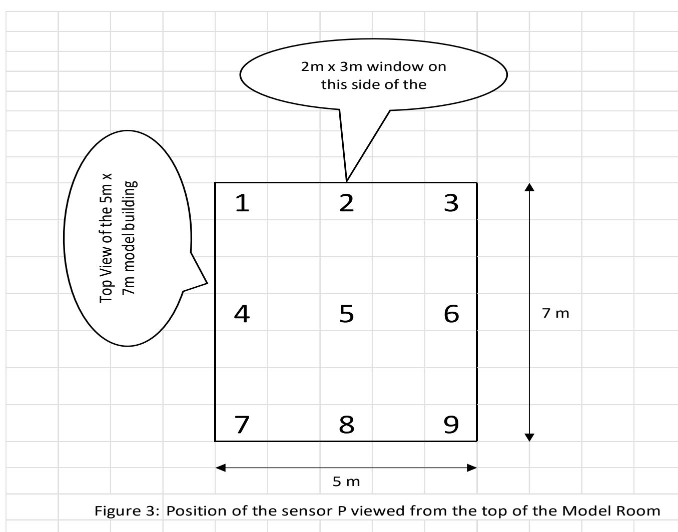


Figure 3. Position of the Sensor P viewed from the Top of the Model Room

These values are reported by sensor P, as a 4-bit BCD (Binary Coded Decimal) value between 1 and 9. In the initial phase, these values are entered manually at runtime, using the keypad in Proteus. (*, and # are used to select the sensor V and W respectively, and 0 is used to select sensor R for display). As an extension, an analog input for P can be used for continuous values using analog input(s) and the Cartesian coordinate system, with its origin assumed on the top-left of the model building. The values of V and W are injected by using anemometers, as shown in Figure y. These values correspond to the values used by Ali. A (2024), and in the initial phase, they are obtained from data tables stored in the memory of the microcontroller. Table z shows a summary of the results.
The observed value of R is measured and displayed on the two right-hand displays in BCD format, along with the position P, on the left side display.

4. Data Collection

Figure 4 shows the schematic used for simulation using the Proteus platform.

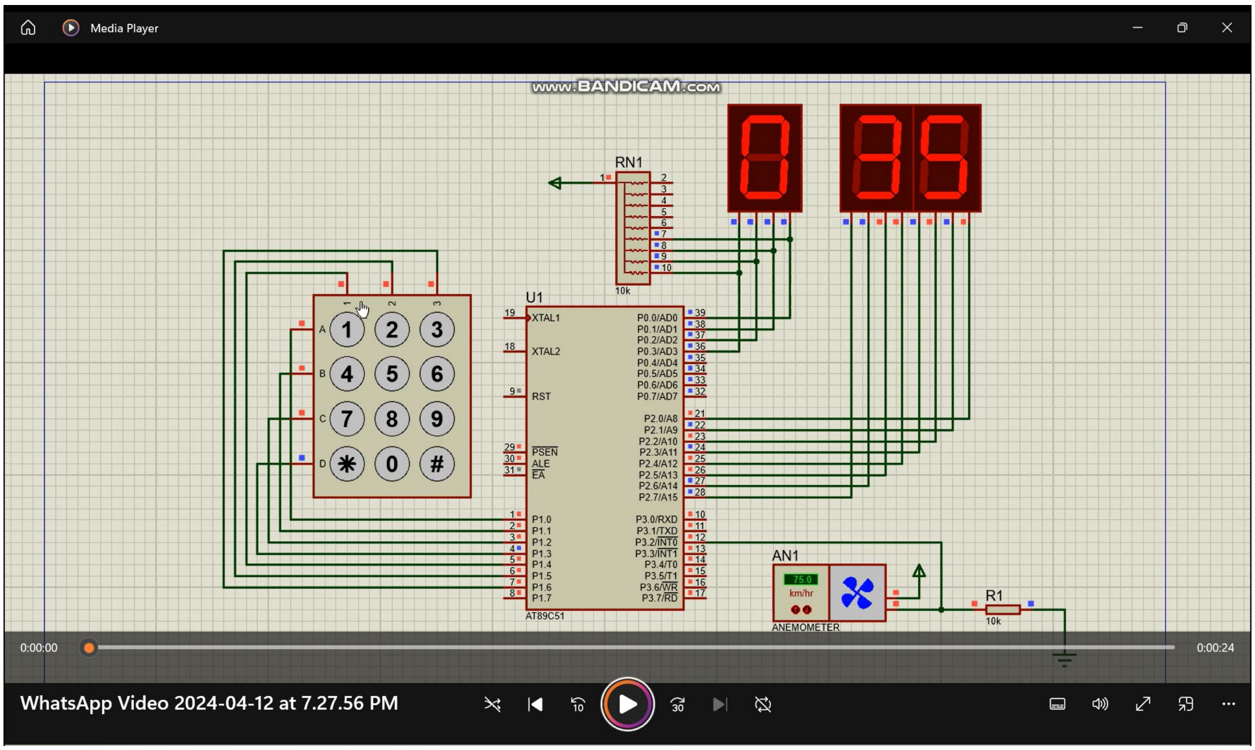


Figure 4. Simulation Setup using Proteus Platform

5. Results and Discussion

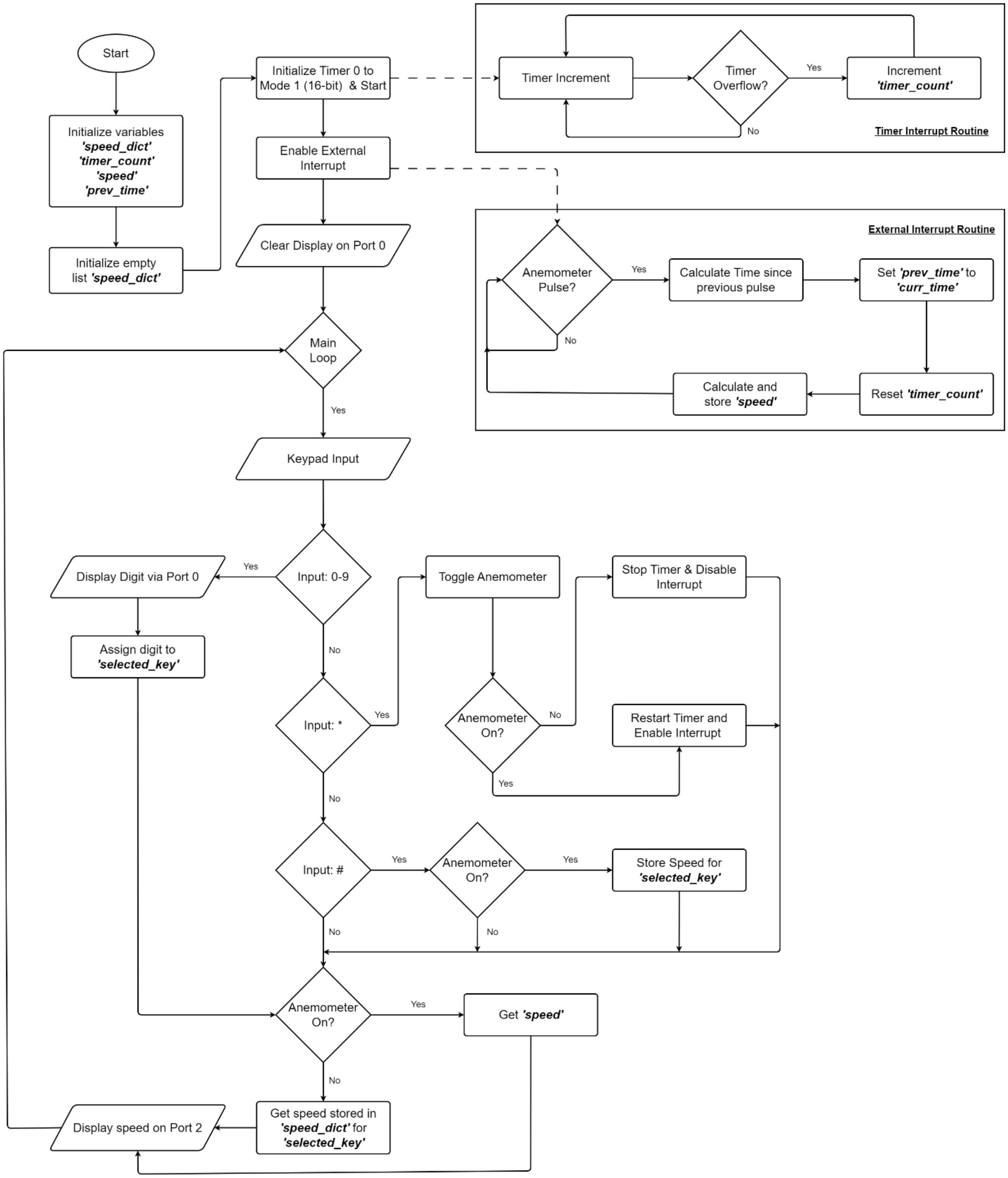
Table 1, shown below, gives a summary of the results, with the following fixed parameters.
Temperature: 293.20 K
Inlet Pressure: (From Window): 101325 Pa
Outlet Pressure: (From Chimney): 100325 Pa
Room Size: 7m x 5m x 5m
Window Size: 3m*3m
Chimney Size: 0.7m*1m
Vent Size: 0.3m*1m
Vent Position: 0.5 m from top

Table 1. Summary of Results

Max wind speed (m/s) corresponging to chimney position		
Chimney Position	Inlet from ventilator	Inlet from window
P	R	R
1	39	38
2	37	35
3	38	34
4	38	35
5	38	34
6	38	34
7	39	36
8	38	35
9	38	35

Please note that the wind speeds have been rounded up rounded up to the nearest whole number in this table

5.1 Flowchart for the system



5.2 C-code for the system

```
#include <reg51.h>
volatile unsigned char selected_key = 0;
volatile unsigned long prev_time = 0;
volatile double speed = 0;
volatile int timer_count = 0;
volatile bit toggle_anemometer = 1;
volatile double speed_dict[10] = {0};
// Keypad definitions
sbit r1 = P1 ^ 0;
sbit r2 = P1 ^ 1;
sbit r3 = P1 ^ 2;
sbit r4 = P1 ^ 3;
sbit c1 = P1 ^ 4;
sbit c2 = P1 ^ 5;
sbit c3 = P1 ^ 6;

void dat(unsigned char);
void keypad();
void store_speed();
void toggle_mode();

//---------------------------------------------------------
// Timer0 Overflow ISR (Interrupt 1)
// Increments the overflow counter each time Timer0 overflows.
//---------------------------------------------------------
void Timer0_ISR(void) interrupt 1
{
    timer_count++;
}
//---------------------------------------------------------
// External Interrupt 0 ISR (Interrupt 0)
// This ISR is triggered by each pulse from the anemometer.
// It computes the elapsed time (?) in microseconds between pulses.
// Then calculates wind speed using the formula:
//    speed (km/h) = 2400000 / ?
// Finally, it resets the tick_count and updates the previous time.
//---------------------------------------------------------
void External0_ISR(void) interrupt 0
{
    unsigned int curr_timer;
    unsigned long full_time;

    if (!toggle_anemometer)
        return;

    // Get the current 16-bit timer value
    curr_timer = (TH0 << 8) | TL0;

    // Combine the current timer value with the overflow count to get the full timestamp
(in ?s)
    full_time = (timer_count * 65536UL) + curr_timer - prev_time;

    // Update previous time to current timer value (only the 16-bit part) for the next
measurement
    // and reset the tick_count.
    prev_time = curr_timer;
    timer_count = 0;

    // Prevent division by zero
    if (full_time == 0)
```

```
        return;

    // Convert ? (in ?s) to wind speed in km/h using the specification:
    // 1 Hz (i.e. 1,000,000 ?s between pulses) -> 2.4 km/h.
    // Therefore, wind speed = 2.4 * (1,000,000 / ?) = 2400000 / ?.
    speed = 2400000.0 / (double)full_time;

    if (speed > 99) // Clamp if needed
        speed = 99;
}

//-----------------------------------------------------------
// main()
//-----------------------------------------------------------
void main(void)
{
    unsigned char tens_digit, units_digit;

    // Configure External Interrupt 0 for anemometer pulses.
    INT0 = 0;

    // Initialize Timer0 in Mode 1 (16-bit timer)
    TH0 = 0;
    TL0 = 0;
    TMOD = 0x01; // Timer0 mode 1
    TR0 = 1;     // Start Timer0

    // Enable interrupts:
    EA = 1;  // Global interrupt enable
    EX0 = 1; // Enable external interrupt 0
    IT0 = 1; // Set INT0 to be edge-triggered
    ET0 = 1; // Enable Timer0 overflow interrupt

    dat(0); // Clear display on Port0

    while (1)
    {
        keypad(); // Poll the keypad
        if (toggle_anemometer)
        {
            // Display the wind speed on Port2 as a two-digit BCD value
            tens_digit = (unsigned char)(((int)speed) / 10);
            units_digit = (unsigned char)(((int)speed) % 10);
        }
        else
        {
            // Display the selected speed from the dictionary
            tens_digit = (unsigned char)(((int)speed_dict[selected_key])/10);
            units_digit =(unsigned char)(((int)speed_dict[selected_key])%10);
        }
        P2 = (tens_digit << 4) | units_digit;
    }
}

//-----------------------------------------------------------
// dat()
// Outputs a value to Port0
//-----------------------------------------------------------
void dat(unsigned char y)
{
    P0 = y;
```

```c
    selected_key = y;
}

//-----------------------------------------------------
// keypad()
// Polls the keypad and outputs a digit to Port0 when a key is pressed.
//-----------------------------------------------------
void keypad(void)
{
    c1 = c2 = c3 = 1;
    // First row:
    r1 = 0;
    r2 = 1;
    r3 = 1;
    r4 = 1;
    if (c1 == 0)
    {
        while (c1 == 0);
        dat(1);
    }
    else if (c2 == 0)
    {
        while (c2 == 0);
        dat(2);
    }
    else if (c3 == 0)
    {
        while (c3 == 0);
        dat(3);
    }
    // Second row:
    r1 = 1;
    r2 = 0;
    r3 = 1;
    r4 = 1;
    if (c1 == 0)
    {
        while (c1 == 0);
        dat(4);
    }
    else if (c2 == 0)
    {
        while (c2 == 0);
        dat(5);
    }
    else if (c3 == 0)
    {
        while (c3 == 0);
        dat(6);
    }
    // Third row:
    r1 = 1;
    r2 = 1;
    r3 = 0;
    r4 = 1;
    if (c1 == 0)
    {
        while (c1 == 0);
        dat(7);
    }
    else if (c2 == 0)
```

```
    {
        while (c2 == 0);
        dat(8);
    }
    else if (c3 == 0)
    {
        while (c3 == 0);
        dat(9);
    }

    // Fourth row:
    r1 = 1;
    r2 = 1;
    r3 = 1;
    r4 = 0;
    if (c1 == 0)
    {
        while (c1 == 0);
        toggle_mode();
    }
    else if (c2 == 0)
    {
        while (c2 == 0);
        dat(0);
    }
    else if (c3 == 0)
    {
        while (c3 == 0);
        store_speed();
    }
}

void store_speed()
{
    if (toggle_anemometer && selected_key < 10)
    {
        speed_dict[selected_key] = speed;
    }
}

void toggle_mode()
{
    toggle_anemometer = !toggle_anemometer;
    if (toggle_anemometer)
    {
        prev_time = 0;
        timer_count = 0;
        TH0 = 0; // Reset Timer0 high byte
        TL0 = 0; // Reset Timer0 low byte
        TR0 = 1; // Start Timer0
        EX0 = 1; // Enable external interrupt 0
    }
    else
    {
        TR0 = 0; // Stop Timer0
        EX0 = 0; // Disable external interrupt 0
    }
}
```

6. Conclusion

A microprocessor-based simulator for monitoring the wind speed in a model building has been presented in this paper. The wind speed is dependent on the position of the ventilator that is mounted on a chimney. The position of the ventilator is changed, and the corresponding wind speed is measured and displayed. The Proteus simulator has been used to model the system, and the control algorithm has been given in the form of a flowchart. The C-code operating the simulator has also been included in the paper. It has been noted that when the ventilator is positioned at the extreme left side of the left wall of the room, opposite to the window. the air inlet is from the ventilator, and the exit is from the window, with a pressure difference of 1KPa between the two ends, we get a maximum wind speed in the room.

References

Cadence CFD: The "Venturi Effect": Available: https://resources.system-analysis.cadence.com/blog/msa2022-explaining-the-venturi-effect-and-wind-flow-analysis-in-structural-design, Accessed on February 9, 2025.

Gohar GA, Manzoor T, Ahmad A, et al. Design and comparative analysis of an INVELOX wind power generation system for multiple wind turbines through computational fluid dynamics. Advances in Mechanical Engineering. 2019;11(4). doi:10.1177/1687814019831475

How SheerWind™ INVELOX™ works (August 16, 2022),

Available: https://adanqwlam.blogspot.com/2022/08/venturi-effect-wind.html, Accessed on February 5, 2025.

Paul Gipe, Available: https://wind-works.org/, Accessed on February 9, 2025. (ISBN 0-471-10924-X)

UNFCCC COP 29, Available: https://cop29.az/en/home

ALI, A., Abu Hurairah M., Hamid Z., and Ali M., A Study of Wind Patterns As a Function of Ventilator Geometry, *Proceedings of the 9th North American Conference on Industrial Engineering and Operations Management*, Washington D.C., United States, June 4, 2024, https://doi.org/10.46254/NA09.20240168.

"GWEC at COP28." *Global Wind Energy Council*, 17 Nov. 2023, gwec.net/cop28/.

[Online]. Available: https://energyeducation.ca/encyclopedia/Venturi_effect. [Accessed 21 4 2024].

"Lumen Learning," [Online]. Available: https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/14-1-fluids-density-and-pressure/. [Accessed 21 4 2024].

J. F. P.Y Huang, "Wall effects on the flow of viscoelastic fluids around a circular cylinder," *Journal of Non-Newtonian Fluid Mechanics,* vol. 60, no. 2-3, pp. 179-195, 1995.

"www.atmo.arizona.edu," [Online]. Available: http://www.atmo.arizona.edu/students/courselinks/fall12/atmo336/lectures/sec1/structure.html. [Accessed 25 4 2024].

P. S. S. Z. Florian Koch, "www.researchgate.net," [Online]. Available: https://www.researchgate.net/figure/Distance-dependent-air-velocity-measured-for-an-air-pressure-of-3-bars-with-a-Prandtls_fig2_303287243. [Accessed 25 4 2024].

K. Cieslicki, "Investigations of the effect of inertia on flow of air through porous bearing sleeves," *WEAR,* vol. 172, no. 1, pp. 73-78, 1994.

Biographies

Muhammad Ashhub Ali is an undergraduate software engineering student in the final semester at the National University of Science and Technology (NUST), Islamabad, Pakistan. Throughout his studies, he has gained expertise in courses such as Cloud Computing, Operating Systems, Web Engineering, Embedded Systems, Internet of Things (IoT), Software Testing & Quality Assurance, Digital Logic Design, and Database Management Systems. He has professional experience as an SDET Intern at Motive for 6 months, where he worked with Playwright and RestAssured for test automation. His projects include working with ESP32 and Blynk IoT (WiFi Controlled Car), 8051 (Serial Bluetooth Controlled Car), and building REST APIs using FastAPI and Node.js for backend development. Additionally, he has experience in frontend development with React, creating full-stack applications that integrate seamlessly with backend services. For his Final Year Project (FYP), Muhammad Ashhub Ali is working on a SaaS application, Augmentize, which includes a Mobile App, Backend Server, and a Photogrammetry Pipeline. This project focuses on 3D model generation and processing to create an efficient and scalable solution.

Dr. Anjum Ali completed his Ph.D. degree in August 1988 from the University of Alabama, Huntsville, Alabama, U.S.A. He has been teaching Electrical and Computer Engineering subjects since March 1978. His first teaching appointment, as a lecturer of Electrical Engineering, was at the University of Engineering and Technology (UET), Lahore, Pakistan, after winning gold medals in each of the last three years of his undergraduate engineering education.

His teaching experience includes twelve years at Mercer University, Macon, Georgia, USA, and about nine years at three different universities in Saudi Arabia. As a tenured ECE faculty member at Mercer University (1988-1999), he developed and taught a number of computer engineering courses, starting from the first undergraduate course in the area to various advanced MS level electives. He has also worked, as an associate professor, at the Lahore University of Management Sciences (LUMS), Lahore, Pakistan, from 1996 to 1998 (on leave from Mercer University). During his stay at LUMS, he developed the computer engineering portion of the CS curriculum and helped the university transition from the quarter system to the semester system. He served as the chairman of the Electronics Engineering and Instrumentation Department at the Hail Community College (now University of Hail), Hail, Saudi Arabia, from February 2000 to June 2002. During his stay there, he developed a four-year degree program in Electrical Engineering for the University of Hail. Dr. Anjum Ali moved to Pakistan in July 2002, and joined Al-Khawarizmi Institute of Computer Science (KICS) at the University of Engineering and Technology, Lahore, as a professor in December 2002. During his stay at KICS, he initiated many research and development projects and won research grants. He also developed teaching materials related to courses in computer architecture for the Virtual University of Pakistan.

He has been a professor of Electrical Engineering at the National University of Computer and Emerging Sciences, (FAST-NU), Lahore, from May 2005 to May 2018. He was the Head of Electrical Engineering from March 2007 to September 2013, and during this time he developed multiple long-range policies and procedures for the university, which are still in place. He was also the Acting Director, Lahore Campus, at different occasions during his stay at FAST-NU, Lahore. Dr. Anjum Ali served as the convener of the National HEC Computer Engineering curriculum development committee. The HEC committee (NCRC) developed and finalized the 2009 HEC Computer Engineering Curriculum for all Pakistani universities. Dr. Anjum Ali has given lectures as a "Distinguished Speaker" at the Global Engineering Education forum during four international IEOM conferences. Dr. Anjum Ali has taught many EE, CE and CS courses and supervised numerous graduate as well as undergraduate students during his 40 years of teaching career. He has over 30 conference and journal publications. He is also the founding editor of the FAST-NU Research Journal. His areas of current research interest include embedded control systems and computer architecture.