Development of an Arduino-Based Data Acquisition System for a Mid-Sized Wind Tunnel

ABSTRACT

Aims: To develop a low-cost instrumentation system using Arduino-based components for measuring thermophysical properties in a mid-sized wind tunnel.

Study design: The existing research aims to design and implement an Arduino-based Data Acquisition System (DAQ) making use of a Pitot assembly and temperature sensors.

Place and Duration of Study: Department of Mechanical Engineering, University of Nigeria, Nsukka (UNN), between June and August 2023.

Methodology: In this work, we have designed a new model using an Arduino Mega 2560, a MPXV7002DP Pressure Sensor with Pitot Tube, several DS18B20 temperature sensors, K-type thermocouples with MAX6675 Module, and an LCD display to develop a data acquisition system (DAQ) system. The system was calibrated and installed in the wind tunnel to measure air velocity and temperature under varying operational conditions. Experimental procedures involved operating the wind tunnel at different wind speeds, with data collected and analyzed for heat transfer behavior.

Results: A Data Acquisition System based on Arduino have been produced, which provides instrumentation that was lacking to the wind tunnel. The DAQ system enabled real-time measurement of air velocity, pressure, and temperature. After testing the implemented system, results showed reliable and consistent data collection. Heat transfer analysis confirmed the system's effectiveness with calculated heat transfer coefficients across different conditions.

Conclusion: In this research, has been developed and implemented a low-cost Arduino-based instrumentation system for a wind tunnel, restoring and enhancing its usage. This system provides an accessible platform for educational experimentation.

Keywords: Arduino, wind tunnel, instrumentation, airflow velocity, heat transfer, data acquisition system (DAQ).

1. INTRODUCTION

Although the wind tunnel is commonly recognized as a valuable tool for aerodynamic testing nowadays, it was not the initial means of conducting such experiments. Early pioneers of aeronautics understood the need for a device to provide a constant and manageable flow of air instead of relying on the inconsistent and unpredictable winds of nature. Leonardo da Vinci and Isaac Newton had previously recognized this concept. In the early days of aeronautics, both moving a test model through the air at a specific velocity and blowing air past a stationary model were employed. Initially, researchers sought out natural sources of steady wind, such as windswept ridges and blowing caves, but eventually, mechanical devices were developed to achieve consistent airflow. For example, the whirling arm, a type of aeronautical centrifuge, was one of the early devices developed and had the most simple and cost-effective method for achieving high speeds in moving the models [1].

Wind tunnels are extensively used to test the aerodynamic properties of various models such as aircraft, automobiles, and ships. They provide an environment in which the aerodynamic performance of a model can be evaluated before actual construction. Wind-tunnel testing on full or model-scale components is a common method that assists in making precise design decisions for thermal-fluid systems and facilitates fundamental research of fluid phenomena. In situations where theoretical or computational techniques are insufficient, either due to problem complexity or inadequate computing resources, wind-tunnel testing is often the most cost-effective option. In addition to the high costs associated with many forms of full-scale testing, wind tunnel testing can also be linked to such tests through appropriate parameter matching [2].

Wind tunnels can also be classified based on size, such as low-speed and high-speed wind tunnels. The former can vary from small tunnels to large ones capable of testing full-scale automobiles, trucks, and aircraft components. The latter, due to power requirements, are relatively smaller compared to their low-speed counterparts. Wind tunnels can also be classified based on their applications, such as automotive testing, studying the effects of ice formation on aircraft wings, simulating various environmental conditions, flow visualization, aircraft engine evaluation, spin recovery of an aircraft, and studying flight dynamics [3].

Despite the value of wind tunnels in aerodynamic testing, traditional data acquisition systems are often expensive especially for academic or low-budget setups. Microcontroller platforms like Arduino offer a low-cost, customizable alternative, making it easier to measure key parameters such as pressure and velocity without the high cost of commercial systems.

Arduino is primarily used to process signals received from both digital and analog sensors, transmit the processed data to a computer, and control the connected devices. These capabilities allows for the development of affordable tools for educational and analytical purposes [4].

[5] developed and used an open-source USB data acquisition system using 16-bit acquisition resolution simple electronic components and Arduino to demonstrate an analytical experiment using gas chromatography and a capillary electrophoresis-UV separation on an instrument used for capillary purposes. [6] proposed a PID control scheme integrated using an Arduino microcontroller and LabVIEW which is an instrumentation software, to heat and control the temperature of a heating element.

2. MATERIAL AND METHODS

2.1 Design Analysis

The hardware design is for the electronics unit which consist of modules such as the temperature sensor, the microcontroller, and LCD display. A proper circuit design and simulation was adopted in other to achieve a desired output for the project, the circuit diagram serves as guide to a proper circuit design in which the Arduino controls the other components which performs the desired operation in accordance with user preference.

The circuit diagram used is shown in Fig. 1, which was designed using fritzing. The design uses an Arduino mega 2560 microcontroller as a control center for acquiring and processing sensor data. The Arduino is connected to various temperature sensors and a pressure sensor. The sensors include DS18B20 digital temperature sensors, a K-type thermocouple connected via the MAX6675 amplifier module, and an MPXV7002DP differential pressure sensor. Each sensor is powered appropriately using 5V output from the Arduino board, which is based on their operational requirements.

DS18B20 sensors, which require three terminals (VCC, GND, and Data) has their data line connected to a digital I/O pin with a 4.7kΩ resistor to ensure stable one wire communication. The MAX6675 thermocouple amplifier is connected with the CS, SCK, and SO pins linked to digital pins on the Arduino for temperature data acquisition.

The MPXV7002DP pressure sensor, which provides an analog voltage output, is connected to one of the analog input pins on the Arduino. It receives 5V power from the Arduino and outputs a voltage that is interpreted to calculate air velocity in the wind tunnel. An I2C-compatible LCD is connected to the SDA and SCL pins of the Arduino for displaying temperature, velocity, and pressure data in real-time. The LCD is powered through the 5V output of the Arduino.

All sensors and modules are wired using a breadboard and jumper cables to ensure flexibility and ease of modification. The Arduino itself is powered via USB from the computer system, which also serves as the serial interface for monitoring and recording real-time data. Proper grounding is maintained throughout the system by connecting the GND pin of the Arduino to the common ground rail on the breadboard. This setup allows for a fully integrated, low-cost instrumentation system capable of real-time data acquisition and visualization, tailored specifically for aerodynamic testing in a mid-sized wind tunnel.

The microcontroller on the board is programmed using Arduino, which uses C++. Programming is done through the Arduino Integrated Environment (IDE) which provides tools for writing code and debugging errors.

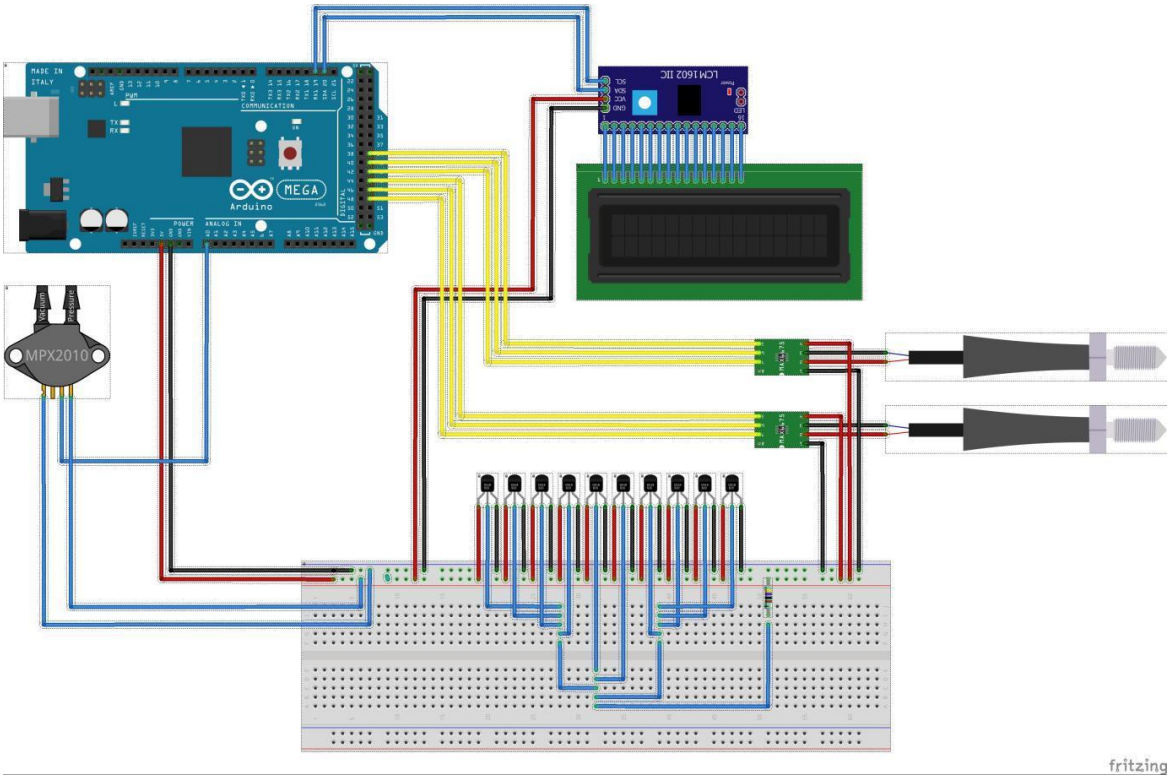

Fig. 1 Circuit Diagram

2.2 Materials Description

2.2.1 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board with 54 digital input/output pins, 16 analog pins, a 16 MHz crystal oscillator, a USB interface, a power connector, and a reset button [7]. This is a very cheap and widely available product.

2.2.2 MPXV7002DP Pressure Sensor with Pitot

The MPXV7002DP is a precision differential pressure sensor designed for applications involving microcontrollers with analog-to-digital conversion capabilities, such as Arduino platforms. It measures pressure ranges from –2 kPa to +2 kPa (approx. ±0.29 PSI) and provides an analog voltage output between 0.5 V and 4.5 V when powered at 5 V. The sensor incorporates on-chip signal conditioning, temperature compensation, and factory calibration to ensure reliable performance. It is often paired with a pitot tube to measure the airspeed of drones and small aircraft. It can also be used to measure local pressures in pipes and enclosures where fluid is flowing, without the need for a pitot tube [8].

2.2.3 DS18B20 temperature sensor

The DS18B20 temperature sensor is a one-wire digital temperature sensor. Each DS18B20 temperature sensor has a unique 64-bit serial code. This allows you to wire multiple sensors to the same data wire [9].

2.2.4 MAX6675 Module and K-Type Thermocouple

The K-Type Thermocouple is capable of measuring temperatures from -200 °C to +1350 °C. It's widely used in high-temperature applications like ovens, engines, and scientific experiments. The MAX6675 converts the signal from the K-Type Thermocouple to a digital output. It can read temperatures ranging

from 0 °C to 1024 °C. The module communicates with the Arduino using SPI (Serial Peripheral Interface) and operates at 3.3 V or 5 V, making it compatible with most Arduino boards. This sensor-amplifier combination was used in this project to accurately measure the temperature of heated water in the copper coil as well as ambient temperatures at various wind tunnel locations [10].

2.3 CALIBRATION OF PRESSURE SENSOR

The sensor calibration for the MPXV7002DP Pressure Sensor was derived from the calculation presented in [11]. Rather than using an external calibrated instrument, the sensor output was interpreted using [11] demonstration of the relationship between analog voltage and differential pressure.

According to [11] the MPXV7002DP outputs voltages between 0.5 V and 4.5 V corresponding to differential pressure from –2 kPa to +2 kPa, with 2.5 V representing zero pressure differential. The conversion from voltage to pressure follows a linear relationship sourced from the manufacturer's datasheet, using the formula derived via point-slope method:

$$\triangle P = \frac{V_{out} - 2.5}{0.2}$$

Where $\triangle P$ is in kilopascals and V_{out} is the measured voltage.

Velocity is then calculated using Bernoulli's equation:

$$v = \sqrt{\frac{2 \triangle P}{\rho}}$$

Assuming air density $\rho = 1.2041 kg/m^3$
In the Arduino code, the arrangement is implemented as shown in fig. 2.

```
64
65  void loop() {
66    // Initialize variables
67    float adc_avg = 0;
68    float velocity = 0.0;
69    float pressure_kPa = 0.0;
70
71    // average a few ADC readings for stability
72    for (int ii=0;ii<mean_size;ii++){
73      adc_avg+= analogRead(A0) - offset;
74    }
75    adc_avg/=mean_size;
76
77    // Convert ADC reading to pressure in Pascals (Pa)
78    pressure_kPa = (((adc_avg / 1023.0) * 5) - 2.5);
79
80    // Calculate velocity based on ADC reading and density
81    if (adc_avg>512-zero_span and adc_avg<512+zero_span){
82      // No velocity change within the defined range
83    } else{
84      if (adc_avg<512){
85        velocity = -sqrt((-10000.0*((adc_avg/1023.0)-0.5))/density);
86      } else{
87        velocity = sqrt((10000.0*((adc_avg/1023.0)-0.5))/density);
88      }
89    }
90
```

Fig. 2 Formatted code for MPXV7002DP Pressure Sensor

To improve measurement stability, multiple ADC readings are averaged. The output voltage is centered around 2.5V, which corresponds to 0kPa differential pressure. The averaged voltage is converted to pressure using the equation provided in the sensor datasheet and explained in [11].

$$P = \frac{V_{out} - 2.5}{Sensitivity} \; with\; sensitivity = 0.2V/kPa$$

Velocity is then calculated as:

$$v = \sqrt{\frac{2 \triangle P}{\rho}} = \sqrt{\frac{10000 \cdot (ADC - 0.5)}{\rho}}$$

The Arduino code utilizes five calibration constants to ensure stable and accurate pressure and velocity measurements from the MPXV7002DP sensor. The sensor is powered using the Arduino's 5 V supply (VS = 5.0 V). The offset value of 54 ADC units was established by observing the raw output at zero differential pressure and adjusting to align with the expected mid-scale voltage (2.5 V). A total of 10,000 readings is averaged (mean_size = 10000) during each measurement cycle to suppress high-frequency noise. A zero_span of 2 ADC units is defined around the midpoint (ADC = 512), within which the sensor output is considered to reflect no meaningful airflow. An air density of 1.2041 kg/m³ was used in the Bernoulli-based velocity calculation, which corresponds to dry air at 20 °C and sea level atmospheric pressure. These constants are shown in Fig. 3.

```
45
46  // Define constants for pressure sensor calculations
47  const float VS = 5.0; // supply voltage to the pressure sensor
48  const float density = 1.2041; // density of air
49  int offset = 54; // Offset value for calibration
50  int mean_size = 10000; // Number of samples for calculating mean
51  int zero_span = 2; // Zero span value for calibration
52
```

Fig. 3 Calibration Constants

2.3.1 Limitations in Calibration

Due to the unavailability of a calibrated reference instrument and time constraints, no external device was used to validate the output of the MPXV7002DP sensor. As a result, the calibration process relied solely on the specifications and guidelines provided in the manufacturer's datasheet. While this method ensures a reasonable level of accuracy, it introduces potential uncertainties, especially regarding sensor drift, environmental variation, and manufacturing tolerances.

3. RESULTS AND DISCUSSION

3.1 Experimental Results

The aim of this experiment was to determine the heat transfer coefficient of a hollow copper coil located within the wind tunnel. Airflow was controlled via the rheostat of the wind tunnel at 80% (176 V) power levels. Hot water at temperatures ranging from 61–67 °C was passed through the coil, and real-time temperatures were recorded at several locations within the wind tunnel. Thermocouples captured inlet and outlet water temperatures, while the DAQ system recorded surface and ambient temperatures. Pressure values from the Pitot-static setup were used to compute air velocity.

Using the mass of water (0.116 kg) and copper coil area (0.055 m²), the heat transfer *QQQ* was calculated using:

$$Q = m \cdot c \cdot (T_{inlet} - T_{outlet})$$

The convection heat transfer coefficient *h* was then calculated as:

$$h = \frac{Q}{A \cdot (T_{coil} - T_6)}$$

Where T_6 represents the ambient air temperature near the coil.

Table 1. Data for 80% capacity of the rheostat (176V)

Inlet (°C)	Outlet (°C)	Pressure (P)	Velocity (m/s)	Copper Temperature (°C)	Ambient Temperature (sensor 6) (°C)
67.00	52.00	138.92	15.19	33.75	28.25
66.00	46.00	144.42	15.49	33.06	28.31
65.00	43.00	144.97	15.52	32.44	28.97
64.00	42.00	142.09	15.36	31.87	28.44
63.00	41.00	143.11	15.42	31.62	28.44
62.00	39.00	143.34	15.43	31.12	28.37
61.00	38.00	145.10	15.52	31.00	28.44

Table 2. Table for Heat Transfer and Heat Transfer Coefficient

Inlet $(℃)$	Heat Transferred $Q(KJ)$	Heat Transfer Coefficient(KJ/m^2K)
67.00	-7.28	138.92
66.00	-9.71	144.42
65.00	-10.68	144.97
64.00	-10.68	142.09
63.00	-10.68	143.11
62.00	-11.16	143.34
61.00	-11.16	145.10

In Table 2, the heat transferred (*Q*) is presented as a negative value to reflect the loss of thermal energy from the water as it flows through the copper coil. This aligns with the physical interpretation that heat is being released by the hot water and transferred to the surrounding airflow via convection. Thus, negative values of *Q* indicate heat dissipation, not absorption.

Fig. 4 shows a direct correlation between the temperature difference (ΔT) and the heat transfer coefficient (h). As ΔT increased, h also increased, indicating enhanced convective heat transfer at higher thermal

gradients. This validates the effectiveness of the copper coil as a heat exchange medium under controlled airflow conditions. It shows the variation of the heat transfer coefficient with temperature difference at 176 V. The graph confirms that greater thermal gradients promote higher convective heat transfer under stable airflow conditions.

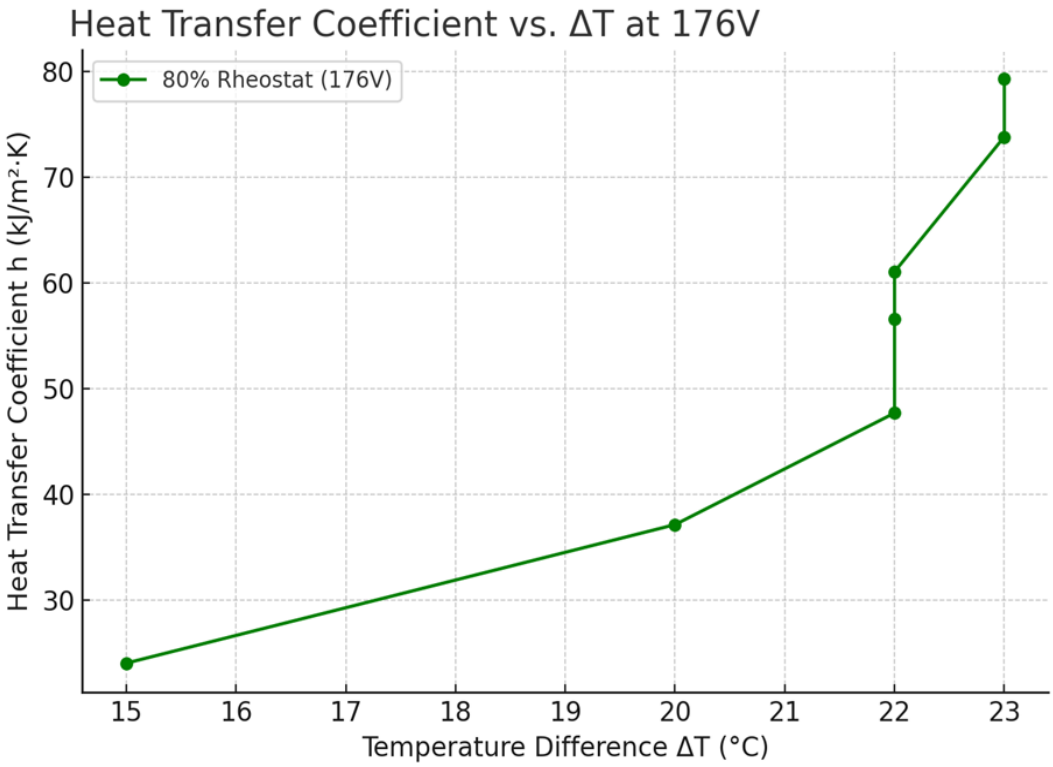

Fig. 4. Heat Transfer Coefficient vs $\triangle T$

While inlet water temperature varied, the calculated heat transfer coefficient was more strongly influenced by the temperature difference and airflow velocity, rather than the inlet temperature alone.

Throughout the 176 V tests, airflow velocity measured by the MPXV7002DP sensor remained relatively constant, ranging from 15.19 m/s to 15.52 m/s. This ensured consistent convective conditions across the heat transfer measurements. Another test was conducted with the rheostat at 60% and the airflow velocity ranged between 11.01m/s and 11.16m/s

4. CONCLUSION

Through the careful design and implementation of various sensors, and a data acquisition module, real-time measurements of the airspeed and temperature provided valuable insights into the aerodynamic behaviors within the tunnel.

The utilization of the Arduino platform offered several advantages, including cost effectiveness, ease of programming, and a wide range of compatible sensors and modules. This ensured that the solution was not only effective but also sustainable and adaptable for future upgrades or modifications.

REFERENCES

1. Baals, D. D., & Corliss, W. R. (1981). Wind tunnels of NASA / Donald D. Baals and William R. Corliss. Scientific and Technical Information Branch, National Aeronautics and Space Administration.
2. Cattafesta, L. N., Bahr, C. J., & Mathew, J. (2010.). Fundamentals of Wind-Tunnel Design. Encyclopedia of Aerospace Engineering. https://doi.org/10.1002/9780470686652.EAE532
3. Barlow, J. B., Rae, W. H., & Pope, A. (1999). Low-speed wind tunnel testing / Jewel B. Barlow, William H. Rae, Jr., Alan Pope. (3rd ed.). Wiley.
4. Gubsky, S. (2023). Development of Low-Cost Arduino-Based Equipment for Analytical and Educational Applications. https://doi.org/10.3390/CSAC2023-14893.
5. Grinias, J.P., Whitfield, J.T., Guetschow, E. D., & Kennedy, R. T. (2016). An Inexpensive, Open-Source USB Arduino Data Acquisition Device for Chemical Instrumentation. J. Chem. Educ. 2016, 93, 7, 1316-1319.
6. Asraf, H. M., Dalila, K. N., Hakim, A. M., & Hon, R. M. F. (2017). Development of experimental simulator via Arduino-based PID temperature control system using LabVIEW. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9(1-5), 53-57.
7. Arduino. (n.d.). Arduino Mega 2560. Retrieved July 2023, from https://docs.arduino.cc/hardware/mega-2560/
8. NXP USA Inc. (n.d.). MPXV7002DP Differential Pressure Sensor (±2 kPa, 0.5 V–4.5 V out, on-chip signal conditioned). Retrieved July 2023, from https://vector4engineering.com/product/differential-pressure-sensor-mpxv7002dp/
9. Random Nerd Tutorials. (n.d.). *Guide for DS18B20 temperature sensor with Arduino*. Retrieved July 2023, from https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/
10. Random Nerd Tutorials. (n.d.). *Arduino: K-Type Thermocouple with MAX6675 Amplifier*. Retrieved July 2023, from https://randomnerdtutorials.com/arduino-k-type-thermocouple-max6675/
11. Maker Portal LLC. (2019, February 8). *Arduino Pitot Tube Wind Speed and Airspeed Indicator – Theory and Experiments*. Retrieved July 2023, from https://makersportal.com/blog/2019/02/06/arduino-pitot-tube-wind-speed-theory-and-experiment