Aerodynamic Pressure Measurement System Setup Guide

Hardware Requirements

Arduino Setup

- **Arduino Uno/Nano** or similar (with at least 6 analog inputs)
- **Pressure Sensors**: 6 differential pressure sensors (recommended: Honeywell SSCDRRN series)
- **Power Supply**: 5V regulated supply for sensors
- **Connectors**: For pitot tube connections

Recommended Pressure Sensors

1. **Static Pressure**: ±2.5 kPa differential pressure sensor
2. **Dynamic Pressure**: ±2.5 kPa differential pressure sensor
3. **Drag Measurements**: 4× ±1.25 kPa differential pressure sensors

Pitot Tube Setup

- **Main Pitot Tube**: For velocity measurement (connects to dynamic pressure sensor)
- **Static Port**: For reference pressure
- **Model Pressure Ports**: Multiple taps on the test model connected to drag sensors

Wiring Diagram

```
Arduino Pin Connections:
├── A0: Static Pressure Sensor (Vout)
├── A1: Dynamic Pressure Sensor (Vout)
├── A2: Drag Sensor 1 (Vout)
├── A3: Drag Sensor 2 (Vout)
├── A4: Drag Sensor 3 (Vout)
├── A5: Drag Sensor 4 (Vout)
├── 5V: All sensor Vcc
├── GND: All sensor GND
└── USB: Computer connection

Pressure Sensor Wiring (per sensor):
├── Pin 1: Vcc (5V)
├── Pin 2: GND
├── Pin 3: Vout → Arduino Analog Pin
```

```
├── Pin 4: P1 (High pressure side)
└── Pin 5: P2 (Low pressure side)
```

Software Installation

Arduino IDE Setup

1. Install Arduino IDE
2. Install required library: `PID_v1` (Tools → Manage Libraries → Search "PID")
3. Upload the Arduino code to your board

Python Environment

```bash
# Install required packages
pip install pyserial pandas numpy matplotlib scipy

# For Jupyter notebook support (optional)
pip install jupyter ipywidgets
```

Calibration Procedure

1. Zero Pressure Calibration

```python
# Connect to system
daq = PressureDataAcquisition(port='COM3')
daq.connect()

# Ensure all pressure ports are at atmospheric pressure
# (disconnect from pitot tubes, open to atmosphere)
daq.calibrate_sensors()
```

2. Sensor Sensitivity Calibration

You'll need to determine the sensitivity (Pa/V) for each sensor type:

1. **Apply Known Pressure**: Use a manometer or pressure calibrator
2. **Record Voltage**: Note the sensor output voltage
3. **Calculate Sensitivity**: sensitivity = pressure_change / voltage_change

4. **Update Arduino Code**: Modify calibration constants

```cpp
// Example calibration values in Arduino code
PressureCalibration static_cal = {2.5, 1000.0, 5.0};   // offset_V, Pa/V, Vcc
PressureCalibration dynamic_cal = {2.5, 1000.0, 5.0};
PressureCalibration drag_cal = {2.5, 500.0, 5.0};
```

Usage Examples

Basic Single Measurement

```python
from pressure_analysis import PressureDataAcquisition

daq = PressureDataAcquisition(port='COM3')
daq.connect()

# Take single measurement
result = daq.take_single_measurement()
print(f"Velocity: {result['velocity_mph']:.2f} mph")
print(f"Lift Coefficient: {result['lift_coefficient']:.4f}")
```

Continuous Data Recording

```python
# Record for 60 seconds
daq.start_recording(duration=60)
time.sleep(61)  # Wait for completion

# Save and analyze data
daq.save_data("test_run_1")
daq.analyze_data()
daq.plot_data()
```

Complete Aerodynamic Experiment

```python
```

```python
from pressure_analysis import AerodynamicExperiment

# Setup experiment
experiment = AerodynamicExperiment(port='COM3')
experiment.setup_experiment("NACA_0012", {
    'air_density': 1.225,     # kg/m³
    'reference_area': 0.01,   # m²
    'chord_length': 0.1       # m
})

# Run velocity sweep
velocities = [10, 15, 20, 25, 30]  # mph
results = experiment.run_velocity_sweep(velocities, duration_per_point=30)

# Plot results
experiment.plot_experiment_results()
```

Live Data Monitoring

```python
# Start live plotting
daq.start_recording()
animation = daq.plot_live_data()
plt.show()  # Keep window open for live updates
```

Command Line Interface

Basic Operations

```bash
```

```
# Single measurement
python pressure_analysis.py --mode single --port COM3

# Record data for 60 seconds
python pressure_analysis.py --mode record --duration 60

# Live plotting
python pressure_analysis.py --mode live

# Full experiment mode (interactive)
python pressure_analysis.py --mode experiment
```

Data Analysis

```bash
# Analyze saved data file
python pressure_analysis.py --file pressure_data_20241201_143022.csv
```

Model Integration

Pressure Port Placement

For accurate measurements, place pressure taps at:

1. **Leading Edge**: 5-10% chord from nose
2. **Quarter Chord**: 25% chord position
3. **Mid Chord**: 50% chord position
4. **Trailing Edge**: 85-90% chord from nose

Tubing Considerations

- Use small internal diameter tubing (1-2mm ID)
- Keep tube lengths short and equal
- Avoid sharp bends
- Use manifolds for multiple measurements

Data Interpretation

Pressure Coefficients

```
Cp = (P_local - P_static) / (0.5 * ρ * V²)

Where:
- P_local: Local pressure at measurement point
- P_static: Free stream static pressure
- ρ: Air density
- V: Free stream velocity
```

Force Coefficients

```
Lift Coefficient (Cl) = Lift Force / (0.5 * ρ * V² * S)
Drag Coefficient (Cd) = Drag Force / (0.5 * ρ * V² * S)

Where S = reference area
```

Troubleshooting

Common Issues

1. **Noisy Data**
    - Check electrical connections
    - Add filtering (adjust FILTER_SIZE in Arduino code)
    - Shield sensor wires
2. **Incorrect Readings**
    - Verify sensor calibration
    - Check for air leaks in tubing
    - Ensure proper pressure port orientation
3. **Communication Problems**
    - Verify COM port selection
    - Check baud rate (115200)
    - Ensure Arduino is properly connected

Debugging Commands

```python
# Check system status
daq.send_command("STATUS")

# Recalibrate sensors
daq.calibrate_sensors()

# Zero all readings
daq.zero_sensors()
```

Advanced Features

Custom Analysis Functions

```python
# Calculate pressure distribution
cp_data = daq.calculate_pressure_distribution(data)
daq.plot_pressure_distribution(data)

# Custom filtering
from scipy import signal
filtered_data = signal.butter(N=4, Wn=0.1, btype='low')
```

Data Export Options

```python
# Save as CSV
daq.save_data("experiment_1", format='csv')

# Save as JSON (includes metadata)
daq.save_data("experiment_1", format='json')
```

Safety Considerations

1. **Pressure Limits**: Don't exceed sensor pressure ratings
2. **Electrical Safety**: Use proper grounding for all equipment
3. **Tube Connections**: Ensure secure connections to prevent disconnection

4. **Calibration**: Regular calibration ensures measurement accuracy

Performance Specifications

- **Sampling Rate**: Up to 100 Hz per channel
- **Pressure Range**: ±2.5 kPa (typical)
- **Accuracy**: ±1% full scale (after calibration)
- **Resolution**: 12-bit (Arduino ADC)
- **Response Time**: <100ms (limited by filtering)