AN ENHANCED CROSS WIND ESTIMATION IN AVIATION

ABSTRACT

Crosswind estimation plays a vital role in ensuring the stability, control, and safety of systems in aviation, autonomous vehicles, and meteorological applications. This project focuses on developing a DSP-based crosswind estimation system utilizing the GY-521 (MPU6050) sensor, Arduino, and an OLED display. The MPU6050 sensor provides raw acceleration and angular velocity data from its built-in 3-axis accelerometer and 3-axis gyroscope. These measurements are collected in real time by an Arduino microcontroller and are used to infer the motion characteristics of a moving platform affected by crosswinds. The acquired sensor data is used to estimate wind-induced motion by applying mathematical models that convert linear acceleration and angular velocity into an estimated wind speed acting on the system. This wind speed data is then processed using a series of advanced digital signal processing techniques in MATLAB. These include **signal filtering** (such as low-pass Butterworth filtering), **Fourier analysis** to examine the frequency components, **spectral analysis** using spectrograms to understand time-varying frequency behavior, and **adaptive filtering** (e.g., LMS algorithm) to eliminate dynamic interference and noise sources in real time.The refined, noise-reduced wind speed estimates are then displayed on an OLED screen, offering a clear, real-time visualization of crosswind effects.

INDEX

INTRODUCTION:

Crosswind disturbances introduce lateral dynamic forces that significantly affect the trajectory and stability of aerial and autonomous systems. Traditional anemometric solutions are often unsuitable for mobile platforms due to their dependency on external references and hardware complexity. Our project proposes a compact, real-time crosswind estimation approach using inertial measurements obtained from a GY-521 (MPU6050). By leveraging embedded data acquisition and applying digital signal processing (DSP) techniques—including low-pass filtering, spectral analysis, and adaptive filtering algorithms such as LMS—we extract wind-induced motion signatures from raw acceleration and angular velocity data. The processed signal enables reliable wind speed estimation and real-time visualization, supporting improved control strategies in safety-critical applications.

PROPOSED WORK & BLOCK DIAGRAM:

1.Sensor Integration and Data Acquisition:

- The MPU6050 module is used to capture raw acceleration and angular velocity data from a moving platform.
- This data is collected by an Arduino microcontroller in real time and represents the physical effects of wind on the system's motion.

2.Wind Speed Estimation via Mathematical Modeling:

- Acceleration and angular velocity values are mathematically converted into estimated wind speed values using physical relationships and assumptions about system dynamics.
- This initial estimation provides a raw approximation of crosswind effects.

3.Digital Signal Processing Using MATLAB:

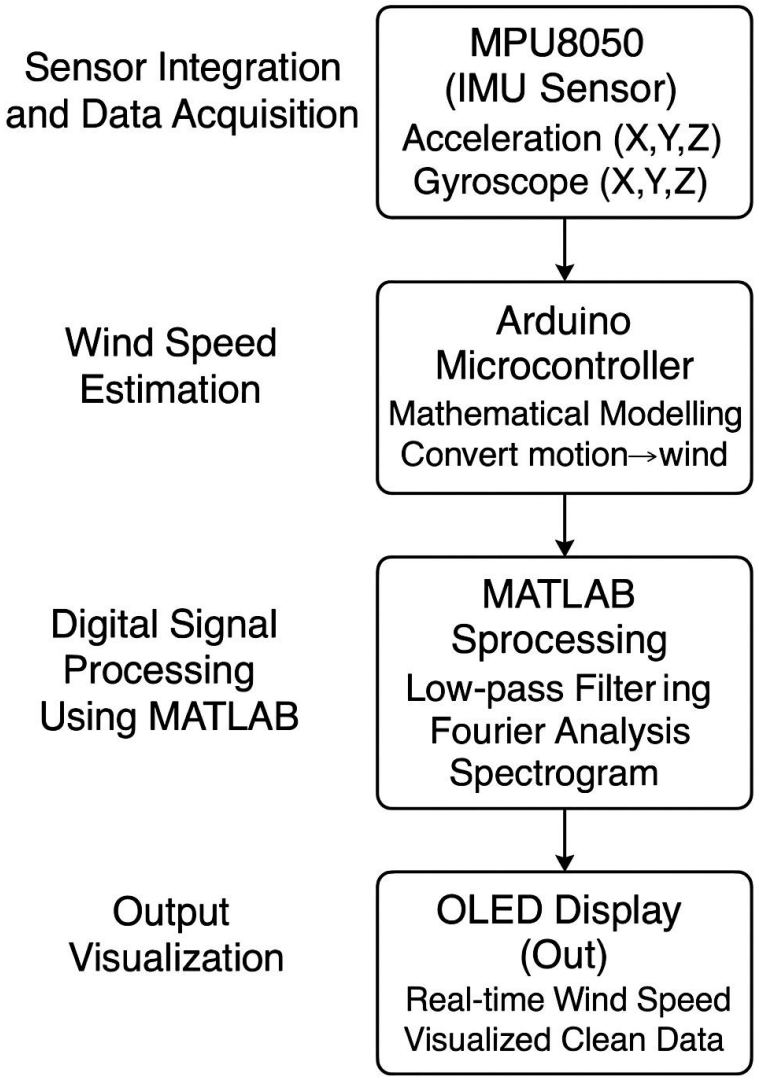
- The raw wind speed data is exported to MATLAB for further processing and enhancement.
- Signal filtering techniques, such as low-pass Butterworth filters, are applied to eliminate high-frequency sensor noise.
- Fourier analysis is used to examine the frequency components of the wind signal, helping identify and isolate periodic

interference.

- A spectrogram is generated for time-frequency analysis, offering insight into how frequency characteristics vary over time.
- Adaptive filtering, specifically the LMS algorithm, is employed to suppress dynamic and reference noise signals without distorting the true wind signature.

4.Output Visualization:

- The final, noise-reduced wind speed estimates are displayed on an OLED screen, providing real-time visualization.
- This enables users to intuitively monitor crosswind effects and observe wind variations as they occur.



RESULTS AND DISCUSSION :

1. Sensor Integration and Data Acquisition

Accurate real-time data acquisition from onboard sensors is vital for detecting sudden wind disturbances or shifts in aircraft dynamics. The MPU6050 enables such detection in lightweight aircraft or drones, supporting flight stability and control.

2. Wind Speed Estimation via Mathematical Modeling

Using kinematic equations and assumptions about the vehicle's motion in free air, the acquired accelerometer and gyroscope data were converted into an estimated wind speed. This estimation considered both linear acceleration and rotational dynamics to approximate the effect of lateral (cross) winds.

This method provides onboard wind estimation without the need for external references, which is particularly valuable for UAVs or in GPS-denied environments.

3. Digital Signal Processing Using MATLAB

- Low-pass Butterworth Filtering

  This removed high-frequency noise introduced by mechanical vibrations or sensor instability, preserving only meaningful wind-induced signals.
- Fourier Analysis (FFT)

  Helped visualize and identify dominant frequency components, such as motor interference or environmental disturbances, allowing for targeted filtering.
- Spectrogram Analysis

  Offered a time-frequency visualization, showing how different frequencies evolved over time — useful for detecting wind gusts and turbulence.
- Adaptive Filtering (LMS Algorithm)

  Adaptively suppressed known or reference noise (e.g., motor hum or structural vibration) without affecting the wind signal itself.
- All these DSP techniques simulate how real-world aviation systems

clean and interpret sensor data to prevent erroneous navigation, ensure autopilot accuracy, and enhance flight performance under varying wind conditions.

4. Output Visualization

An OLED display was integrated to present the real-time, noise-reduced wind speed. This visual feedback system makes wind monitoring intuitive and immediate, useful for both operators and autonomous systems.

Relevance to Aviation: Real-time visualization supports pilot decision-making and autonomous corrections, particularly during landing approaches, turns, or windy navigation scenarios**.**

MATLAB CODING:

```
clc;
clear;
close all;
% ------------------------------------------------
% 1. Given Wind Speed Raw Data (as Matrix)
% ------------------------------------------------
Fs = 50;  % Sampling frequency (Hz)
% wind speed raw data (100 values)
wind_speed_raw = [ ...
    3.42, 3.18, 3.60, 3.45, 3.28, 3.77, 3.41, 3.63, 3.21, 3.54, ...
    3.09, 3.68, 3.84, 3.53, 4.08, 4.02, 4.17, 3.72, 3.95, 3.60, ...
    3.36, 3.64, 3.19, 3.43, 3.12, 2.94, 3.29, 3.06, 3.57, 3.69, ...
    3.82, 3.61, 3.39, 3.26, 3.07, 3.34, 2.91, 3.12, 3.31, 3.48, ...
    3.59, 3.26, 3.79, 3.92, 4.03, 4.19, 4.32, 4.14, 3.71, 3.48, ...
    3.63, 3.24, 2.89, 3.03, 2.72, 2.93, 3.08, 3.29, 3.17, 3.01, ...
    3.45, 3.68, 3.91, 3.47, 3.59, 3.30, 3.52, 3.03, 2.95, 3.10, ...
    3.36, 3.58, 3.70, 3.94, 4.12, 4.23, 3.81, 3.62, 3.50, 3.33, ...
    3.08, 3.02, 3.11, 2.86, 3.05, 3.41, 3.22, 3.52, 3.74, 3.92, ...
    4.11, 4.00, 3.69, 3.51, 3.42, 3.13, 3.01, 2.83, 2.73, 2.65 ...
];
```

```
N = length(wind_speed_raw);   % Number of samples
t = (0:N-1)/Fs;               % Time vector
% ------------------------------------------------
% 2. Signal Filtering: Low-pass Butterworth
% ------------------------------------------------
fc = 2; % Cutoff frequency (Hz)
[b, a] = butter(4, fc/(Fs/2), 'low');
wind_speed_filtered = filtfilt(b, a, wind_speed_raw);
% ------------------------------------------------
% 3. Spectral Analysis (FFT)
% ------------------------------------------------
Y = fft(wind_speed_raw);
f = (0:N-1)*(Fs/N);
magnitude_spectrum = abs(Y)/N;
% Plot FFT
figure;
plot(f, magnitude_spectrum);
title('Frequency Spectrum of Raw Wind Speed Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
grid on;
xlim([0 20]);
% ------------------------------------------------
% 4. Adaptive Filtering (LMS)
% ------------------------------------------------
interference = 0.2 * sin(2*pi*10*t);
x = interference + 0.05 * randn(1, N);
d = wind_speed_filtered;
mu = 0.01;
order = 32;
y = zeros(1, N);
e = zeros(1, N);
w = zeros(order, 1);
for n = order:N
    x_n = x(n:-1:n-order+1)';
    y(n) = w' * x_n;
    e(n) = d(n) - y(n);
    w = w + mu * x_n * e(n);
end
wind_speed_adaptive_filtered = e;
% ------------------------------------------------
```

```
% 5. Plot Results
% ------------------------------------------------
figure;
plot(t, wind_speed_raw, 'r--', 'DisplayName', 'Raw Wind Speed');
hold on;
plot(t, wind_speed_filtered, 'b-', 'DisplayName', 'Low-pass Filtered');
plot(t, wind_speed_adaptive_filtered, 'g-', 'DisplayName', 'Adaptive Filtered');
xlabel('Time (s)');
ylabel('Wind Speed (m/s)');
title('Wind Speed Signal Processing');
legend;
grid on;
% ------------------------------------------------
% 6. Spectrogram (Time-Frequency Analysis) - FIXED
% ------------------------------------------------
figure;
spectrogram(wind_speed_raw, 32, 28, 64, Fs, 'yaxis');
title('Spectrogram of Raw Wind Speed Signal');
% ------------------------------------------------
% 7. Display Processed Values in Command Window
% ------------------------------------------------
disp('------------------------------------------------');
disp('Raw Wind Speed Data:');
disp(wind_speed_raw);
disp('------------------------------------------------');
disp('Low-pass Filtered Wind Speed Data:');
disp(wind_speed_filtered);
disp('------------------------------------------------');
disp('Adaptive Filtered Wind Speed Data:');
disp(wind_speed_adaptive_filtered);
```

ARDUINO CODING:

```
#include <Wire.h>
#include <SoftwareWire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <MPU6050.h>
// OLED display setup (uses hardware I2C A4 = SDA, A5 = SCL)
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
// Software I2C for MPU6050 on A1 (SDA), A2 (SCL)
SoftwareWire swWire(A1, A2);     // SDA, SCL
MPU6050 mpu(&swWire);            // Pass SoftwareWire pointer to MPU
// Low-pass filter constant
const float alpha = 0.1;
float filteredX = 0;
void setup() {
  Serial.begin(9600);
  // OLED initialization
  Wire.begin();
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("OLED not found!");
    while (true);
  }
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Initializing MPU...");
  display.display();
  // MPU initialization
  swWire.begin();
  mpu.initialize();
  if (!mpu.testConnection()) {
    Serial.println("MPU6050 init failed!");
    display.setCursor(0, 10);
    display.println("MPU init failed");
    display.display();
    while (true);
  }
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("MPU6050 Ready!");
  display.display();
  delay(1000);
}
void loop() {
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);
  // Convert raw acceleration to 'g' units (assuming ±2g range)
```

```
  float rawX = (float)ax / 16384.0;
  // Apply simple low-pass filter
  filteredX = alpha * rawX + (1 - alpha) * filteredX;
  // Estimate wind speed (tuned factor – adjust as needed)
  float windSpeed = abs(filteredX) * 70.0;  // Approximation
  // Display on OLED
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Crosswind Estimator");
  display.setCursor(0, 16);
  display.print("Accel X: ");
  display.print(rawX, 3);
  display.println(" g");
  display.setCursor(0, 32);
  display.print("Filtered: ");
  display.print(filteredX, 3);
  display.println(" g");
  display.setCursor(0, 48);
  display.print("Est. Wind: ");
  display.print(windSpeed, 1);
  display.println(" m/s");
  display.display();
  delay(200);
}
```

CONCLUSION:

This project presents a smart, compact solution for estimating crosswind using motion data from an MPU6050 sensor. By combining real-time data acquisition with advanced signal processing in MATLAB—such as filtering, spectral analysis, and adaptive algorithms—the system delivers accurate wind speed insights with reduced noise and interference.

The final output, shown on an OLED display, enables real-time wind

monitoring, especially valuable for UAVs, autonomous systems, and weather applications. This work proves how embedded platforms and DSP techniques can work hand-in-hand to solve real-world environmental sensing challenges efficiently.

APPLICATION:

- Crosswind Landing Assistance

  Helps aircraft adjust landing approach by providing onboard crosswind estimation, especially in airports lacking advanced weather infrastructure.

- Autonomous Drone Navigation

  Enables drones to detect and correct for wind-induced drift during flight, improving stability and flight path accuracy.

- In-Flight Stability Monitoring

  Continuously tracks wind disturbances affecting aircraft motion, enhancing autopilot and flight control system performance.

- Light Aircraft & Glider Safety

  Offers a lightweight, low-cost solution for small aircraft to monitor environmental wind effects during manual or assisted flight.

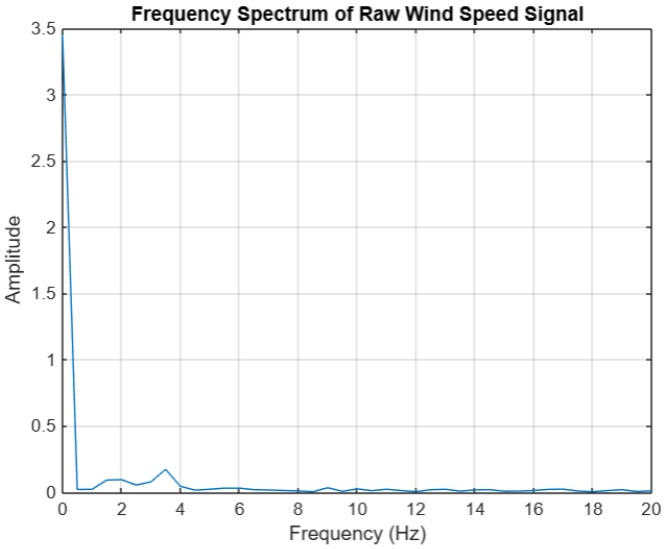
- Aviation Training Simulators

  Simulates real-world wind disturbances for pilot training, improving preparedness for crosswind conditions.

SNAPSHOT OF PROJECT OUTPUT:

Raw Wind Speed Data :

3.4200	3.1800	3.6000	3.4500	3.2800	3.7700	3.4100	3.6300	3.2100
3.5400	3.0900	3.6800	3.8400	3.5300	4.0800	4.0200	4.1700	3.7200
3.9500	3.6000	3.3600	3.6400	3.1900	3.4300	3.1200	2.9400	3.2900
3.0600	3.5700	3.6900	3.8200	3.6100	3.3900	3.2600	3.0700	3.3400
2.9100	3.1200	3.3100	3.4800	3.5900	3.2600	3.7900	3.9200	4.0300
4.1900	4.3200	4.1400	3.7100	3.4800	3.6300	3.2400	2.8900	3.0300
2.7200	2.9300	3.0800	3.2900	3.1700	3.0100	3.4500	3.6800	3.9100
3.4700	3.5900	3.3000	3.5200	3.0300	2.9500	3.1000	3.3600	3.5800
3.7000	3.9400	4.1200	4.2300	3.8100	3.6200	3.5000	3.3300	3.0800
3.0200	3.1100	2.8600	3.0500	3.4100	3.2200	3.5200	3.7400	3.9200
4.1100	4.0000	3.6900	3.5100	3.4200	3.1300	3.0100	2.8300	2.7300
2.6500								

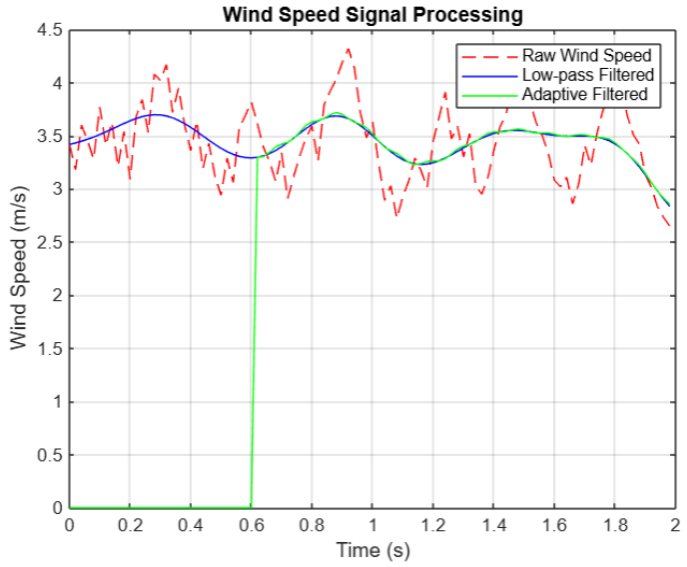


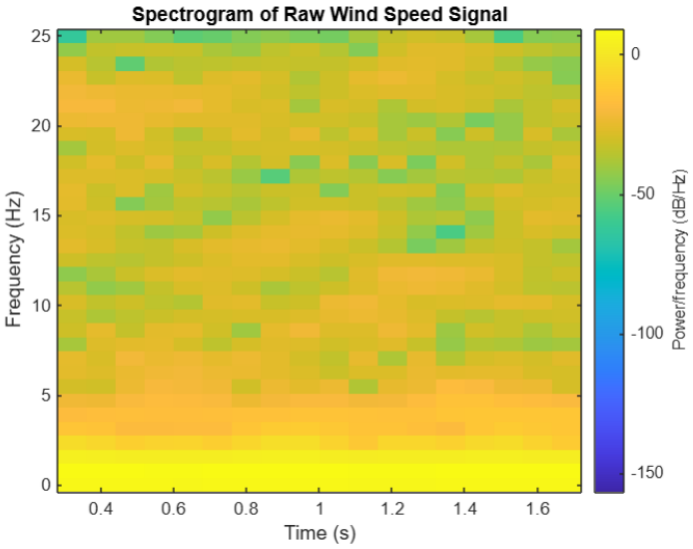
Low-pass Filtered Wind Speed Data:

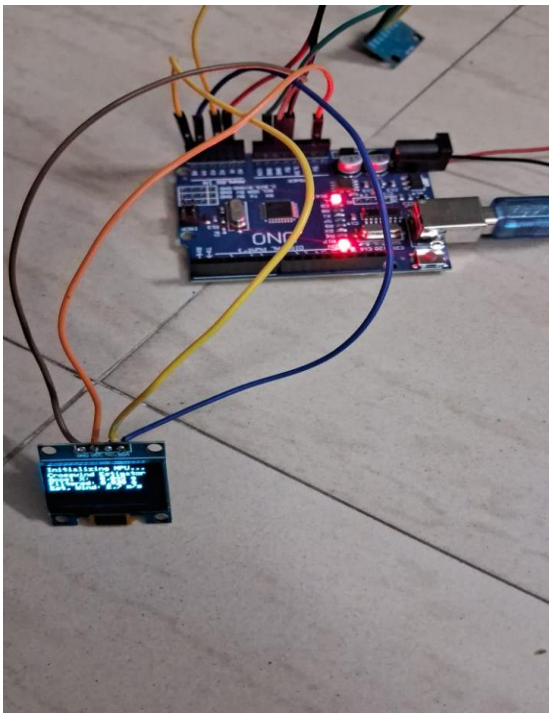
3.4173 3.4321 3.4485 3.4671 3.4878 3.5108 3.5358 3.5621 3.5890
3.6154 3.6402 3.6621 3.6796 3.6917 3.6972 3.6955 3.6861 3.6690
3.6449 3.6144 3.5789 3.5398 3.4989 3.4578 3.4183 3.3820 3.3503
3.3246 3.3057 3.2945 3.2915 3.2968 3.3104 3.3320 3.3609 3.3960
3.4360 3.4791 3.5233 3.5663 3.6058 3.6395 3.6652 3.6811 3.6861
3.6795 3.6614 3.6326 3.5945 3.5493 3.4995 3.4477 3.3969 3.3497
3.3083 3.2746 3.2499 3.2349 3.2296 3.2336 3.2461 3.2657 3.2910
3.3205 3.3526 3.3857 3.4184 3.4495 3.4776 3.5020 3.5218 3.5365
3.5460 3.5503 3.5498 3.5454 3.5381 3.5291 3.5196 3.5109 3.5040
3.4992 3.4969 3.4964 3.4969 3.4967 3.4942 3.4874 3.4742 3.4530
3.4225 3.3821 3.3317 3.2721 3.2047 3.1317 3.0554 2.9786 2.9041
2.8344

Adaptive Filtered wind Speed Data :

0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 3.2968 3.3033 3.3443 3.3918 3.4186 3.4367 3.4738 3.5359
3.5956 3.6257 3.6406 3.6613 3.6938 3.7152 3.6996 3.6631 3.6298
3.6070 3.5774 3.5204 3.4492 3.3928 3.3602 3.3342 3.2927 3.2498
3.2294 3.2400 3.2587 3.2645 3.2652 3.2856 3.3310 3.3770 3.4030
3.4211 3.4472 3.4873 3.5253 3.5408 3.5417 3.5468 3.5622 3.5757
3.5657 3.5423 3.5308 3.5326 3.5353 3.5231 3.5036 3.4971 3.5091
3.5221 3.5164 3.4989 3.4875 3.4849 3.4775 3.4416 3.3851 3.3292
3.2824 3.2282 3.1506 3.0588 2.9774 2.9124 2.8543


Wind Speed Signal Processing
Raw Wind Speed
Low-pass Filtered
Adaptive Filtered
Wind Speed (m/s)
Time (s)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8


Spectrogram of Raw Wind Speed Signal
Frequency (Hz)
Time (s)
Power/frequency (dB/Hz)
0
5
10
15
20
25
0.4
0.6
0.8
1
1.2
1.4
1.6
-50
-100
-150




GND VCC SCL SDA
Initializing MPU...
Crosswind Estimator
Accel X: 0.045 g
Filtered: 0.038 g
Est. Wind: 2.7 m/s