**Design Report: Versatile Sensor Board Design with STM32WL, Analog Front-End, LoRa Wireless, and External SMA Sensor Connector**

**By: Caoilte Donohoe**

## Table of Contents

1. Introduction
2. Block Diagram and Subcircuit Descriptions  
     2.1 3D Render and PCB using Altium Designer  
     2.2 Subcircuit Schematics from Altium Designer  
      2.2.1 ADC & Analog Front End  
      2.2.2 Power and Input Filtering  
      2.2.3 BME280 and MPU-6050 Sensors

2.2.4 STM32WL55 with SWD and I2C and UART Breakout Connector

  2.2.5 LoRa Output Stage  
   2.3 Input π Filter Frequency Response Calculation

 2.4 3rd Order Sallen-Key Anti-Aliasing Filter and Pseudo-Differential Output

1. Power Delivery System  
     3.1 Overview  
     3.2 Switching Mode Power Supply (SMPS) for Digital Circuitry  
     3.3 Low-Dropout Regulator (LDO) for Analog Circuitry
2. Component Selection Criteria  
     4.1 Low-Pass Filter Components  
     4.2 Operational Amplifier (MCP6001)  
     4.3 Sallen-Key Filter Design  
     4.4 Power Supply Selection  
     4.5 Sensor Selection
3. 3rd Order Sallen-Key Anti-Aliasing Filter Simulations  
     5.1 Salen-Key Anti Aliasing Circuit in LTSpice  
      5.1.1 AC Sweep & Bode Plot Analysis  
      5.1.2 Frequency Response (Bode Plot)  
      5.1.3 Transient Analysis: 10 kHz Sine Wave Response  
      5.1.4 Pulse Response Analysis  
      5.1.5 Pseudo-Differential Output  
     5.2 Python Analysis of 3rd-Order Sallen-Key Filter Summary  
      5.2.1 Bode Plots/Frequency Response (Python)  
      5.2.2 Time-Domain Analysis (Python)  
      5.2.3 Noise Filtering in 3rd-Order Sallen-Key Anti Aliasing Filter (Python)
4. System Integration and Peripheral Interfaces  
     6.1 Reference Voltage Considerations  
     6.2 Microcontroller and LoRa Communication  
      6.2.1 LoRa Matching and Filtering Network  
      6.2.2 Sensor Integration  
      6.2.3 Connectivity and Breakout Pins
5. Conclusion

**1. Introduction and System Overview**

I’ve designed this system as a **versatile sensor board** built around the STM32WL microcontroller. It has a **BME280** for temperature, humidity, and pressure, plus an **MPU-6050** IMU for motion sensing. There’s also a dedicated **SMA connector** for plugging in an external analog sensor—this could be a microphone for audio monitoring, a vibration sensor for industrial applications, or even air quality and turbidity sensors for environmental data.

To make sure all signals are captured cleanly, I’ve added a low-noise analog front-end (AFE) with a carefully tuned low-pass filter and a pseudo-differential output for precise ADC conversion. The system’s power delivery is rock-solid, with a switching mode power supply (SMPS) and low-dropout regulator (LDO) to keep both analog and digital domains quiet and stable.

On the **connectivity side**, I’ve included a breakout connector with **I2C, UART, and GPIO** pins. This makes it easy to hook up additional peripherals, sensors, or even control modules—whatever the application calls for. For the layout and simulation work, I used **Altium Designer** for schematic capture and PCB layout, **LTspice** for analog simulations, and **Python** in the **Anaconda** environment (with **Jupyter Notebook**) to analyze and visualize the data after simulation.

With **LoRa wireless built in**, this system is perfect for environmental sensing, motion tracking, or remote data logging—wherever accurate, low-noise analog data and reliable communication are needed.

# Use Cases for the STM32WL Sensor Board

Environmental Monitoring  
- Capture real-time temperature, humidity, and pressure data for weather stations, smart farming, or building automation.  
- External analog sensors (like air quality or turbidity probes) expand these applications further.

Vibration and Motion Monitoring  
- Use the onboard IMU to track vibrations or motion in industrial equipment, helping with preventive maintenance and machinery health monitoring.

Flexible External Sensor Integration  
- SMA connector allows for easy connection of sensors like microphones (for acoustic analysis), analog chemical sensors, or photodiodes (for light intensity measurements).

Noise-Free Analog Signal Acquisition  
- The low-noise analog front-end ensures accurate data acquisition in noisy environments—ideal for precision sensing applications.

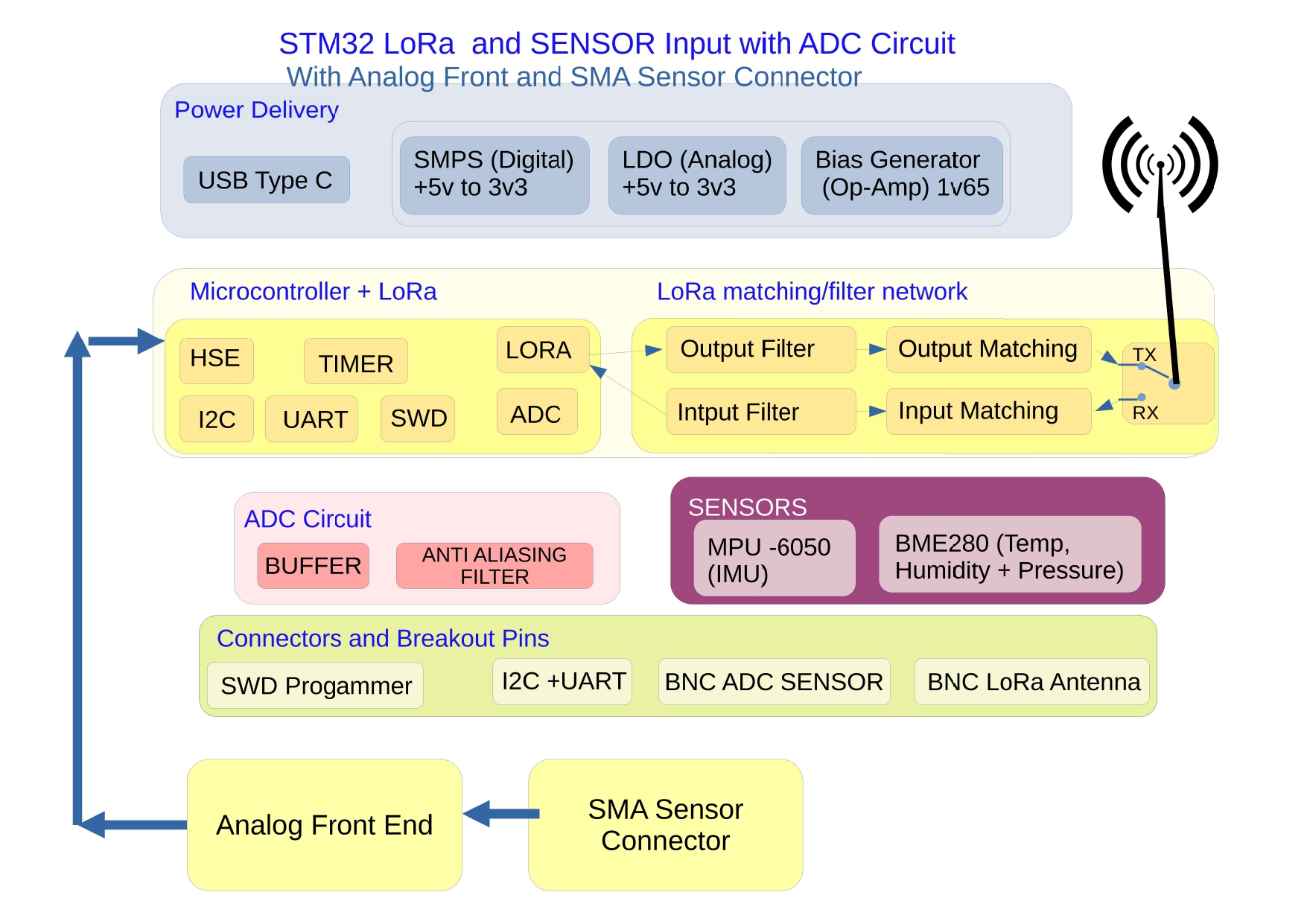
Wireless Remote Monitoring  
- Integrated LoRa wireless communication enables long-range, low-power data transmission—perfect for remote environmental sensing, smart city infrastructure, or industrial monitoring.

Data Analysis and Visualization  
- The use of Python, Anaconda, and Jupyter Notebook for post-processing means this system can also serve as a data collection node for machine learning or real-time analytics.

Smart IoT Nodes  
- With I2C, UART, and GPIO breakouts, this board can serve as a versatile IoT edge device, aggregating data from multiple sensors and transmitting it wirelessly.

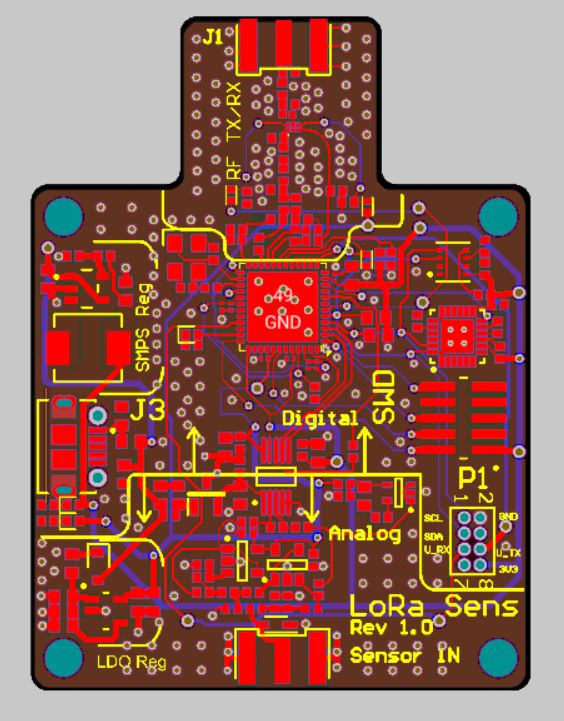
# 2. Block Diagram and Subcircuit Descriptions

# Block Diagram:

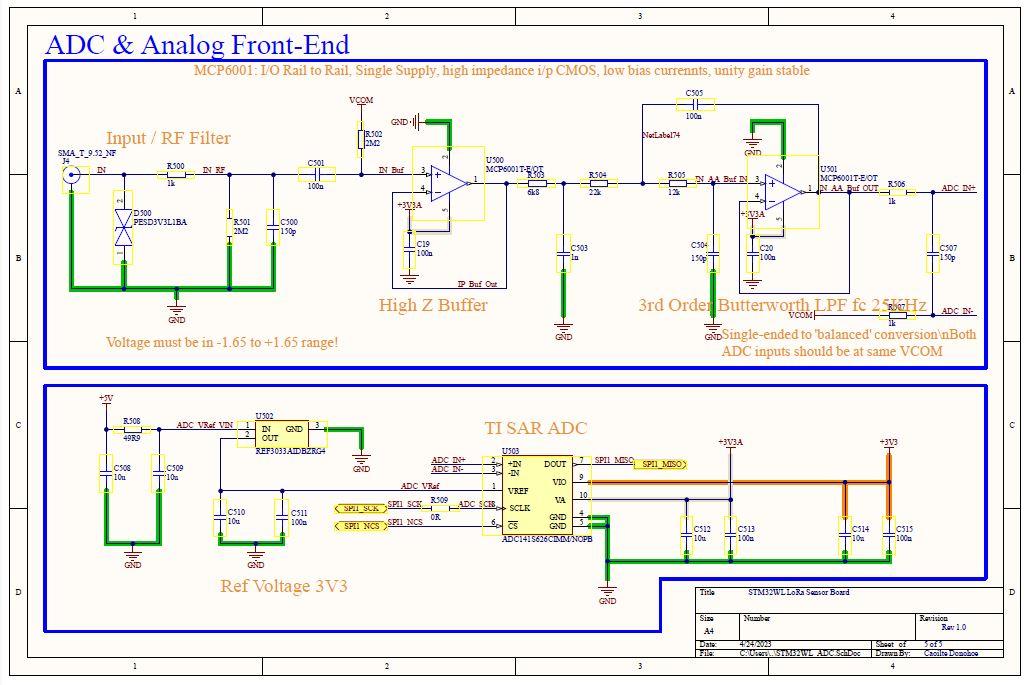


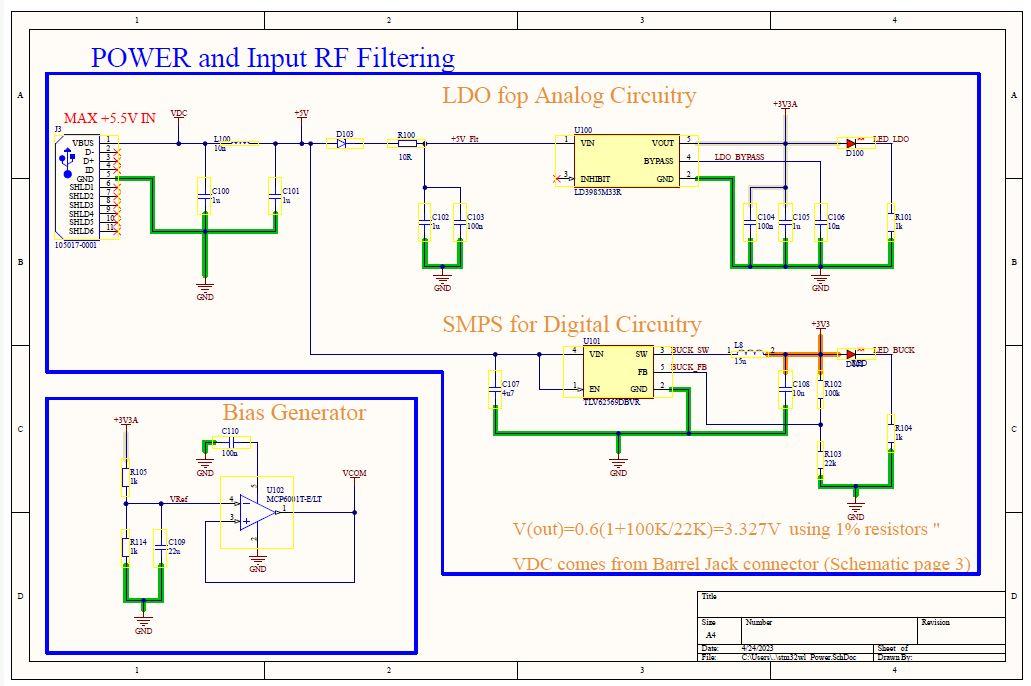
# 2.1. 3D Render and PCB using Altium Designer

## 

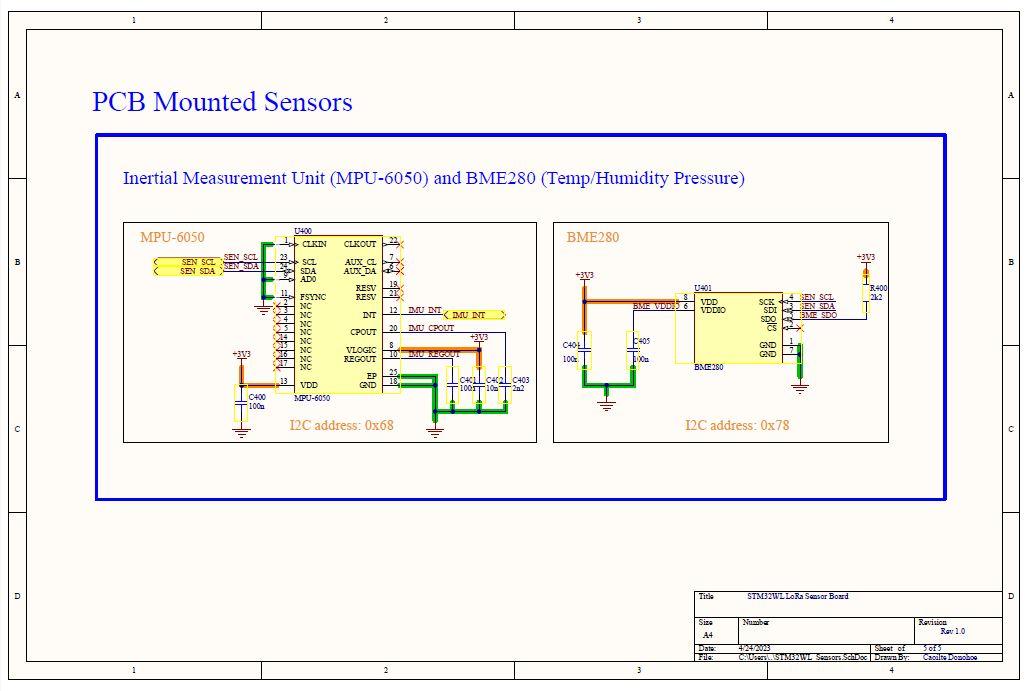


# 2.2 Subcircuit Descriptions

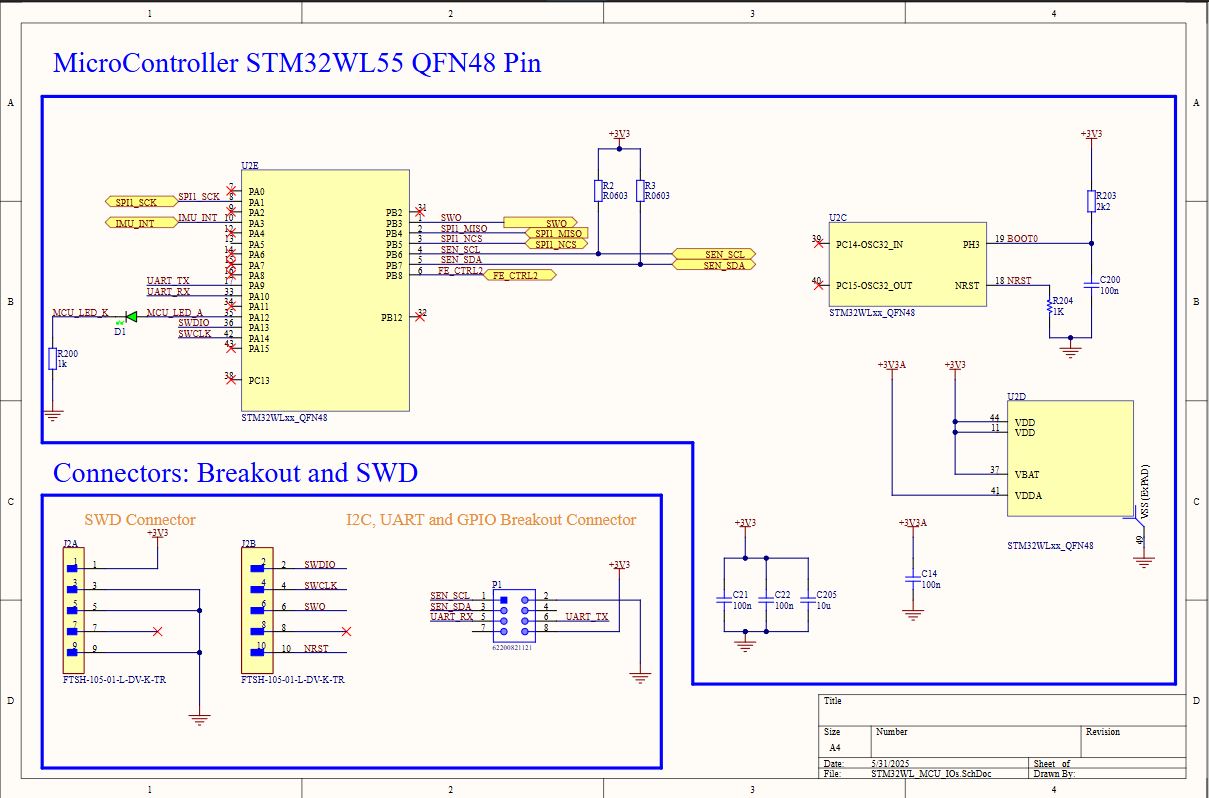
**2.2.1 ADC & Analog Front End**

**2.2.2 Power and Input Filtering**

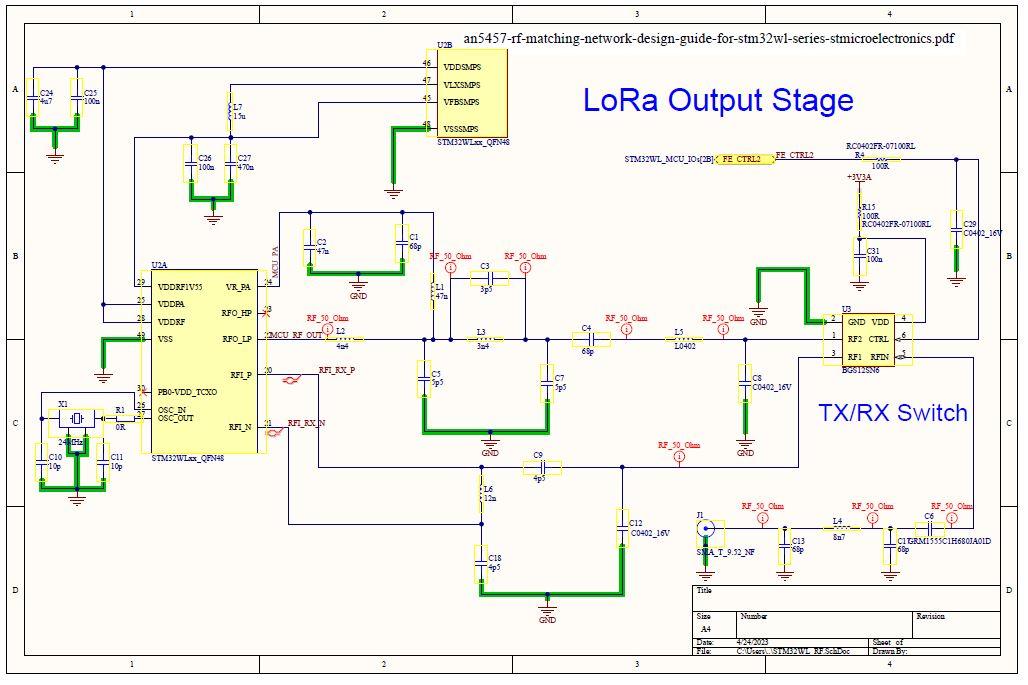
**2.2.3 BME280 and MPU-6050 Sensors**



**2.2.4 STM32WL55 with SWD and I2C and UART Breakout Connector**



**2.2.5 LoRa Output Stage**



# 2.3 Input π Filter Frequency Response Calculation

The input of the power and RF filtering circuit uses a pi filter to cut down on high-frequency noise and EMI from the power line. This filter has two 100 nF capacitors (C100 and C101) and a 1 µH inductor (L100) in between.

The cutoff frequency can be worked out with:

Assuming typical values for these components:

- C100 = C101 = 100 nF

- L100 = 1 µH

The cutoff frequency of the π filter is calculated using the formula:

f\_c = 1 / (2π √(L·C))

where:

- L is the series inductor (1 µH),

- C is the capacitance of either shunt capacitor (100 nF).

Substituting these values:

√(L·C) = √(1 × 10⁻⁶ × 1 × 10⁻⁷) = 3.16 × 10⁻⁷

f\_c = 1 / (2π × 3.16 × 10⁻⁷) ≈ 500 kHz

So, it filters out most noise above 500 kHz while letting low-frequency signals through, helping keep the power supply clean for the rest of the circuit.

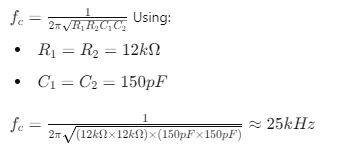
# 2.4 3rd Order Sallen Key Anti Aliasing Filter and Pseudo-Differential Output

I implemented a **3rd Order Sallen Key Anti Aliasing** (Butterworth topology) low-pass filter as a cascaded combination of:

* First-order RC low-pass filter (previously calculated at 156 kHz)
* Second-order Sallen-Key (Butterworth topology) filter, designed for a cutoff frequency of 25 kHz

The Butterworth topology ensures a maximally flat frequency response with minimal ripple in the passband, which is crucial for maintaining signal integrity before ADC conversion.

Sallen-Key Filter Calculation The cutoff frequency for a Sallen-Key filter is:



This stage provides additional attenuation of unwanted frequencies before digitisation.

**3. 3rd Order Salen-Key Anti-Aliasing Filter Simulations:**

# 3.1 Salen-Key Anti Aliasing Circuit in LTSpice:

An anti-aliasing filter is a low-pass filter used before the analog-to-digital converter (ADC).  
Its job is to remove high-frequency noise and signals above the ADC’s Nyquist frequency (half the sampling rate).

This prevents “aliasing”—a distortion where high-frequency signals fold back into the lower frequencies during digital conversion.

**Aside**:**The Nyquist-Shannon Sampling Theorem** says:

To accurately capture (and then reconstruct) a continuous signal without losing information, the **sampling rate** (the number of samples per second) must be at least **twice the highest frequency component** present in the signal.

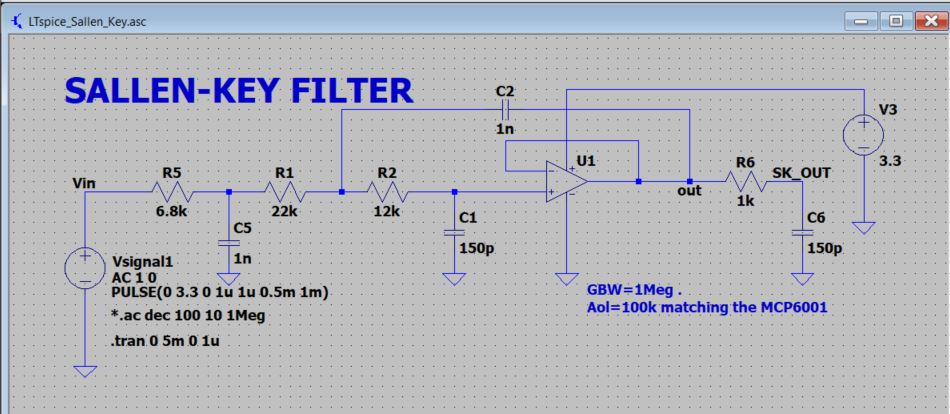
Mathematically, if the highest frequency in your signal is **fₘₐₓ**, then your sampling rate **fₛ** must satisfy:

fs > 2 \* fmax

where:

* **fₛ** = sampling rate (in Hz)
* **fₘₐₓ** = highest frequency component (in Hz)

This critical minimum sampling rate is called the **Nyquist rate**.

****

# 3.1.1 AC Sweep & Bode Plot Analysis

Filter Order: 3rd-order

Cascaded Behavior: 1st-order RC stage followed by 2nd-order Sallen-Key stage

* 3 dB Cutoff Frequency: ~24 kHz
* Roll-off Slope: ~–52 dB/decade to –60 dB/decade, consistent with 3rd-order low-pass behavior
* Input Voltage: ±0.5 V to 1 V pp (centered at 1.65 V DC bias)

Conclusion: Filter meets design expectations for a low-pass anti-aliasing filter centered at 24 kHz.

3.1.2 Frequency Response (Bode Plot)

### 

# 3.1.3 Transient Analysis: 10 kHz Sine Wave Response

Input Frequency: 10 kHz sine wave

Filter’s Low-Pass Behavior:

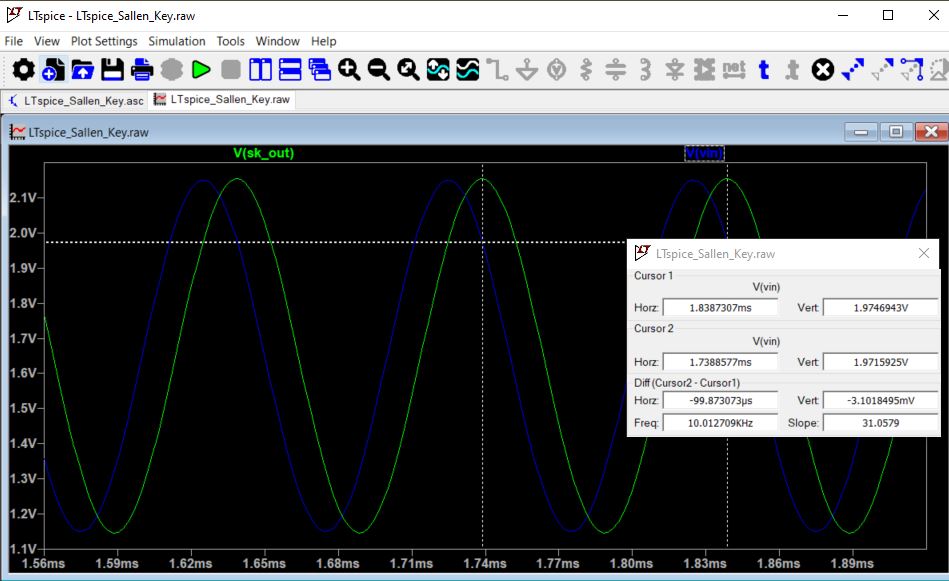
* Output sine wave has slightly reduced amplitude (attenuation) at this frequency
* No significant phase shift (consistent with low-frequency performance below
* 24 kHz

cutoff)

* Confirms excellent filter pass-through for lower frequency signals

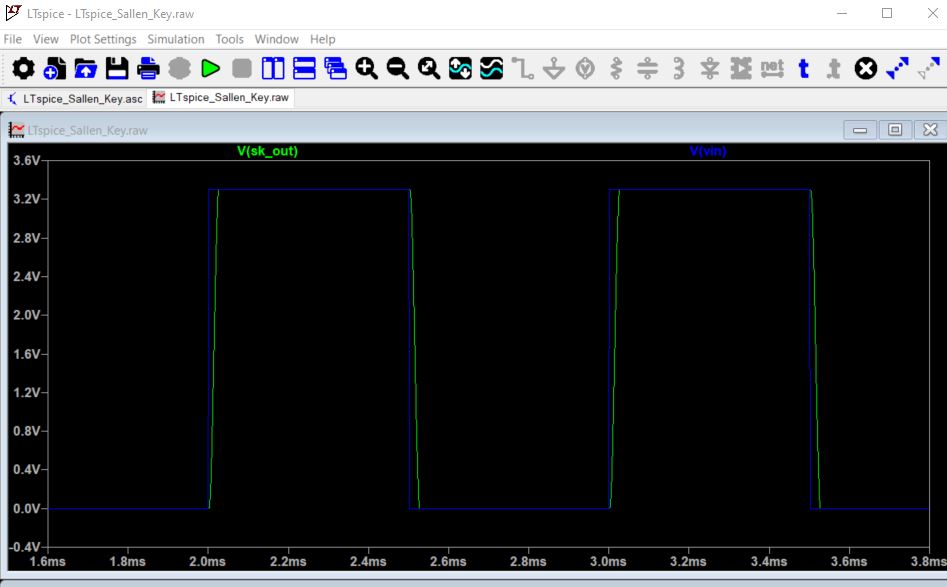
No Distortion or Overshoot:

* Clean waveform shapes for both input and output
* Indicates that the filter design is stable and performing as expected

****

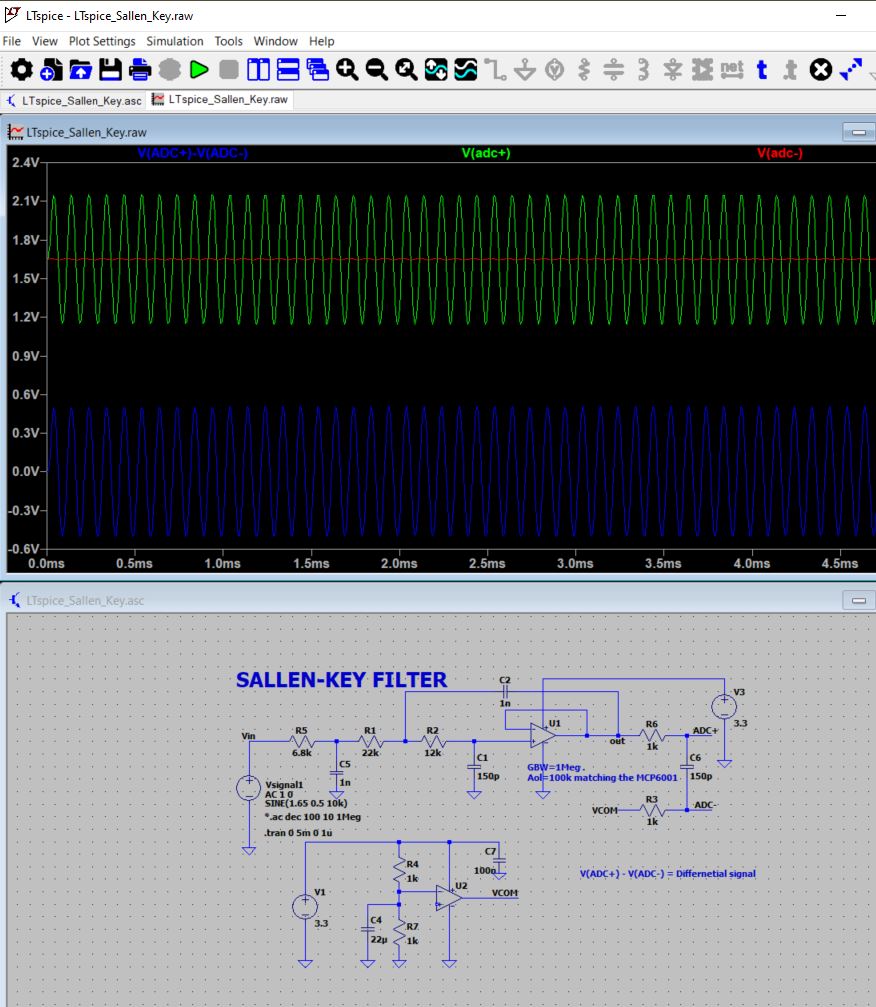
# 3.1.4 Pulse Response Analysis

* Input: 0 V to 3.3 V square wave
* Output (V(sk\_out)):  
    
  Preserves low-frequency square wave shape  
    
  Rounded edges due to filter’s bandwidth limitation  
    
  No overshoot or ringing—indicative of a stable low-pass filter
* Edge Smoothing:  
    
  Confirms 3rd-order filter limits high-frequency harmonics of the square wave  
    
  Transition slope matches expectations for a 24 kHz low-pass filter

****

**3.1.5 Pseudo-Differential Output**

Simulation Setup

* ADC+: Comes from the filter’s output.
* ADC–: Connected to a 1.65 V DC reference (VCOM) generated by a low-impedance op-amp buffer.
* R6 (ADC+ side): 1 kΩ
* R3 (ADC– side): 1 kΩ
* VCOM generation: Generated using an MCP6001 Bias generator
* Differential output (V(ADC+) – V(ADC–)):
* Clean, centered at 0 V (pure AC signal only).
* 1 Vpp sine wave (as expected).
* ****

# 4 Python Analysis of 3rd-Order Sallen-Key Filter Summary

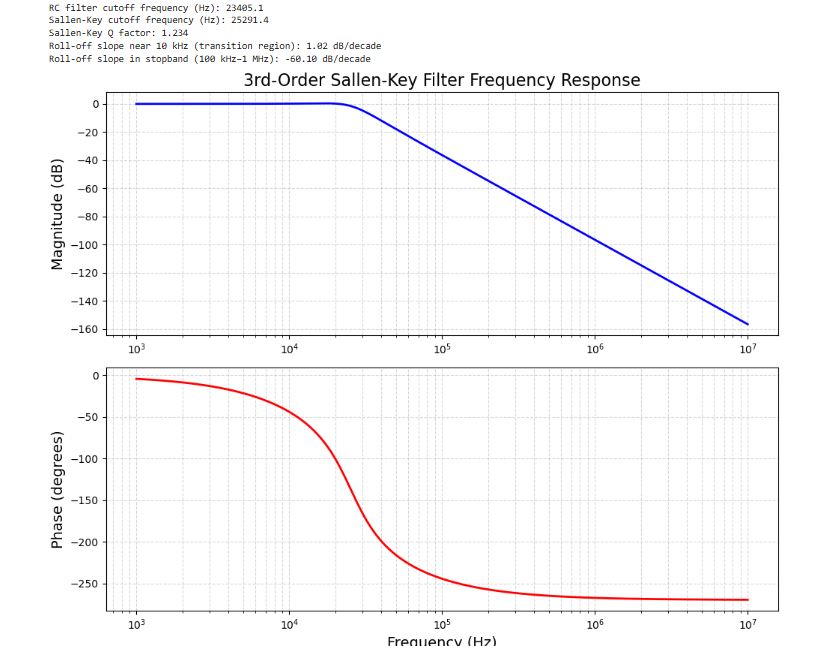
Before starting the analysis, I used Python in the Anaconda environment with Jupyter Notebook. This setup made it easy to run scripts and visualize results interactively.  
 Key libraries I used include:

* **numpy** for numerical operations,
* **matplotlib** for plotting,
* **scipy** for signal processing.

**4.1 Bode Plots/Frequency Response (Python):**

* The **magnitude plot** does what I expect: it’s flat in the passband and then drops off at about –60 dB/decade after the cutoff, confirming the filter is acting like a 3rd-order low-pass filter.
* The **phase plot** starts at 0° at low frequencies, then steadily decreases to around –270° at higher frequencies.  
  This phase shift shows how the filter **delays the waveform as frequency increases**—it’s normal for low-pass filters and helps maintain stability in the system.

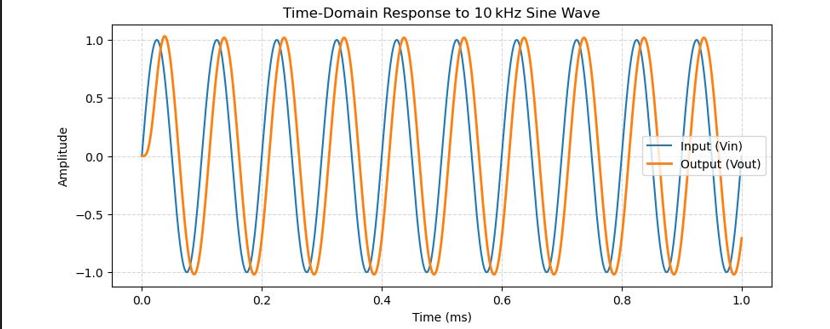
**Why phase response matters:** Phase response tells us how different frequency components of a signal are delayed relative to each other. If the phase shift is too sharp (or non-linear), it can cause waveform distortion—especially for signals with lots of frequency components (like square waves). In most cases for audio or low-frequency sensing, moderate phase shifts (like the –270° here) are expected and not a problem.



**4.2 Time-Domain Analysis (Python):**

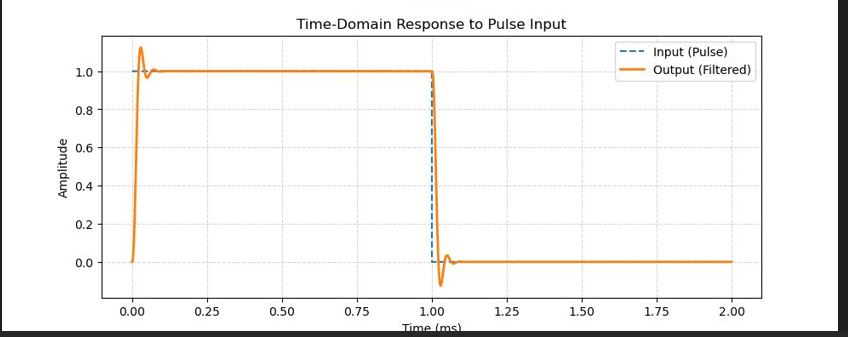
**10 kHz Sine Wave:**

* I saw that the input and output waveforms at 10 kHz are pretty much in phase and amplitude, so the filter does a good job of passing this frequency without much loss.



**Pulse Input Response:**

* The output waveform has a bit of overshoot and some small ringing because of the filter’s peaking near the cutoff. But I don’t see this being an issue because our analog sensor’s frequency is way lower.

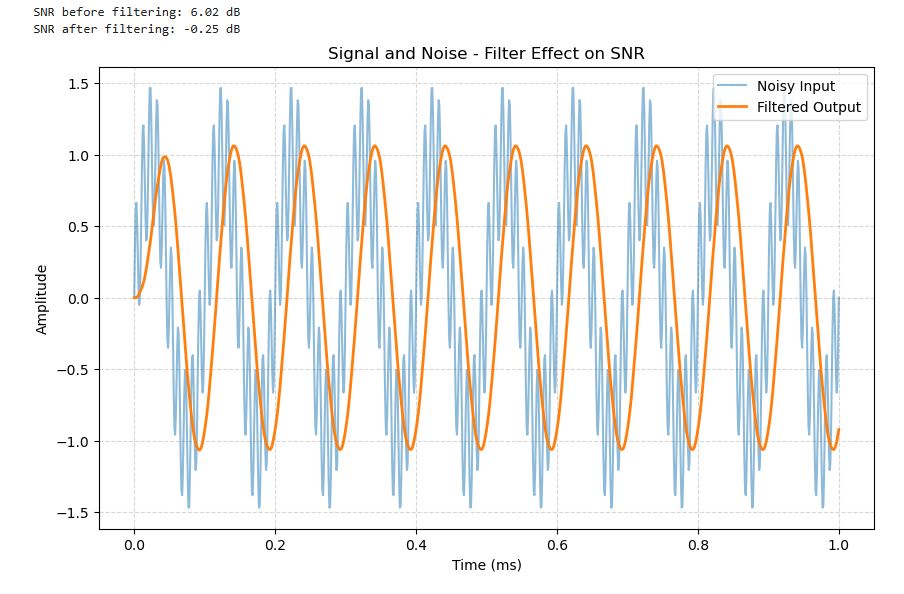


4**.3 Noise Filtering in 3rd-Order Sallen-Key Anti Aliasing Filter (Python):**

I created a test input combining 10 kHz signal and 100 kHz noise

Key results:

* - The filter significantly reduced the 100 kHz noise
* - The 10 kHz signal only lost about –0.52 dB of amplitude – a minor trade-off
* The filtered output looks smooth and clean

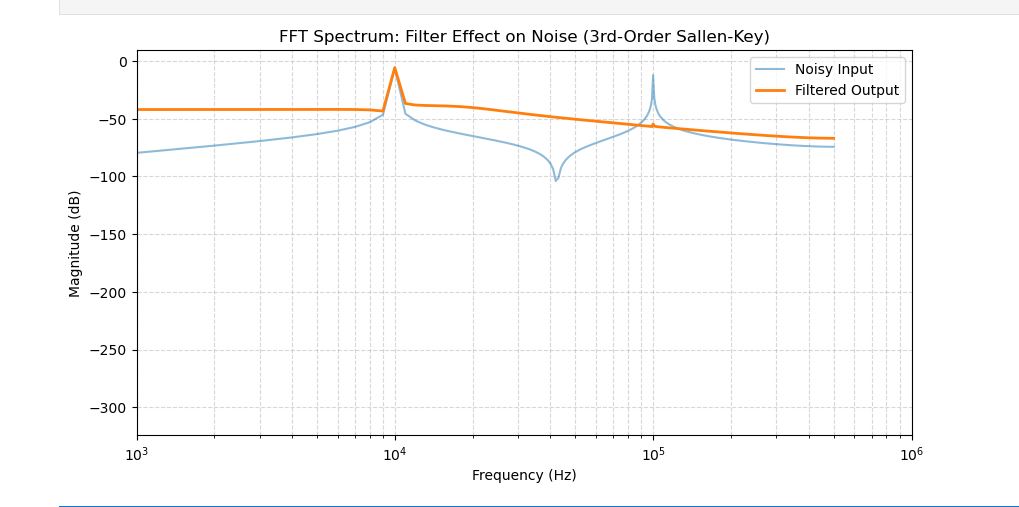


**4.4 FFT in 3rd-Order Sallen-Key Anti Aliasing Filter (Python):**

The plot below shows:

* the noise was removed above 10 kHz
* Also I noticed a dip near 13 kHz in the noisy signal, but it’s just an FFT artifact (not real)

Overall, I’ve shown that this filter keeps my main signal clean, removes high-frequency noise, and significantly improves signal quality – perfect for environmental sensors or similar setups



**5. System Integration and Peripheral Interfaces**

**5.1 Reference Voltage Considerations**

The ADC reference voltage (VREF) was carefully chosen to match the expected input range, ensuring optimal dynamic range and precision in digital conversion.

# 5.4 Microcontroller and LoRa Communication

The STM32WL microcontroller integrates peripherals such as HSE, TIMER, I2C, UART, SWD, ADC, and LoRa communication.

LoRa communication is enabled through an input filter, output filter, and matching network to optimize transmission and reception.

# 5.5 LoRa Matching and Filtering Network

The output filter optimizes the transmitted signal for LoRa RF communication.

The input filter removes unwanted noise before demodulation.

Matching networks ensure impedance compatibility between circuit components and antennas.

# 5.6 Sensor Integration

The system includes an MPU-6050 IMU sensor for motion detection.

A BME280 sensor provides temperature, humidity, and pressure measurements.

These sensors communicate with the microcontroller via I2C.

# 5.7 Connectivity and Breakout Pins

SWD Programmer for debugging and firmware updates.

I2C + UART breakout pins for additional peripheral interfacing.

BNC ADC Sensor input for external analog signals.

BNC LoRa Antenna connection for RF communication.

# 6. Power Delivery System

# 6.1 Overview

The power system consists of an SMPS for digital circuitry and an LDO for analog circuits, ensuring efficient power distribution while maintaining low noise for sensitive analog components.

# 6.2 Switching Mode Power Supply (SMPS) for Digital Circuitry

A buck converter (TLV62569) provides efficient voltage conversion for the STM32WL microcontroller and other digital components.

The high efficiency of the SMPS minimizes heat dissipation, making it suitable for power-sensitive embedded applications.

The output voltage is regulated at 3.3V, providing stable operation for digital logic components.

# 6.3 Low-Dropout Regulator (LDO) for Analog Circuitry

The LD3985M33R LDO is chosen for powering analog circuits to ensure low noise and stable voltage.

LDO regulators are preferred in analog systems due to their superior noise performance compared to switching regulators.

The output voltage is 3.3V, matching the analog voltage requirements of the ADC and op-amp stages.

# 7. Component Selection Criteria

# 7.1 Low-Pass Filter Components

Resistors: Low-noise, high-precision resistors (1% tolerance) to minimize signal distortion.

Capacitors: NP0/C0G ceramic capacitors to ensure stability over temperature variations.

# 7.2 Operational Amplifier (MCP6001)

* Low-input bias current: Ensures minimal loading of the signal source.
* Rail-to-rail I/O: Allows operation across the full ADC input range.
* Low power consumption: Ideal for battery-powered applications.

# 7.3 Sallen-Key Filter Design

* Butterworth topology for maximally flat response.
* Low-value resistors to minimize Johnson noise.
* High-quality capacitors to prevent distortion at high frequencies.

# 7.4 Power Supply Selection

* SMPS for digital circuits due to high efficiency and minimal heat dissipation.
* LDO for analog circuits due to its low noise an
* d stable voltage output.
* Bias generator to ensure proper ADC reference voltage.

# 7.5 Sensor Selection

* MPU-6050: Chosen for its integrated gyroscope and accelerometer for motion tracking.
* BME280: Provides environmental sensing capabilities with high accuracy.

**8. Conclusion**

I’ve shown that this system successfully integrates high-fidelity analog sensing with robust digital and wireless communication, all while keeping noise low and flexibility high. The built-in sensors (BME280 and MPU-6050) cover a wide range of environmental and motion data, and the SMA connector makes it easy to add external sensors—like air quality or turbidity probes—whenever needed. The analog front-end and filter design ensure clean signals get to the ADC, while the pseudo-differential output helps boost accuracy even further.

I’ve backed up the design with thorough simulation and testing—using Altium for layout, LTspice for analog modeling, and Python (with Anaconda and Jupyter) for analysis. The board’s breakout connectors (I2C, UART, GPIO) and LoRa wireless capabilities make it ready for almost any sensor-driven application, from environmental monitoring to industrial tracking. Overall, I’m confident that this design strikes a good balance of performance, flexibility, and practical implementation.