# **xIMU\_V527 Testing Procedure (English)**

## **Introduction**

This document provides a detailed, step-by-step method to test the **xIMU\_V527** module, which includes a 3-axis accelerometer, 3-axis gyroscope (LSM6DSO), and 3-axis magnetometer (LIS3MDL). The test uses an **Arduino Uno** to read sensor data over the I²C interface and display it via the Arduino Serial Monitor. This procedure ensures the module’s sensors are operational, making it suitable for quality control in a PCB manufacturing environment.

## **Materials Required**

* **xIMU\_V527 module** (the device under test)
* **Arduino Uno** (or compatible microcontroller with I²C support)
* **Jumper wires** (male-to-male or male-to-female, at least 4 wires)
* **Breadboard** (optional, for stable connections)
* **USB A-to-B cable** (to connect Arduino Uno to a computer)
* **Computer** with Arduino IDE installed (Windows, macOS, or Linux)
* **Stable power source** (provided via USB or Arduino’s 5V pin)

## **Safety Precautions**

* Ensure all connections are made with the Arduino powered off to avoid short circuits.
* Verify the xIMU\_V527 is not exposed to static electricity, as it contains sensitive components. Use a grounded workstation if possible.
* Do not apply voltages higher than 5.5V to the xIMU\_V527 VIN pin to prevent damage.
* Handle the module carefully to avoid bending pins or damaging solder joints.

## 

## 

## 

## 

## **Step-by-Step Testing Procedure**

### **Step 1: Set Up the Hardware**

1. **Power off the Arduino Uno** to ensure safe connections.
2. Connect the **xIMU\_V527** to the **Arduino Uno** using jumper wires as follows:
   * **VIN** (xIMU\_V527) to **5V** (Arduino Uno): Supplies power to the module.
   * **GND** (xIMU\_V527) to **GND** (Arduino Uno): Establishes a common ground.
   * **SDA** (xIMU\_V527) to **A4** (Arduino Uno): I²C data line.
   * **SCL** (xIMU\_V527) to **A5** (Arduino Uno): I²C clock line.
3. **Optional**: Use a breadboard to secure connections and reduce the risk of loose wires.
4. Double-check the connections against the pin labels on the xIMU\_V527 board to avoid errors.
5. Ensure no other devices are connected to the Arduino to prevent I²C bus conflicts.

### **Step 2: Install the Arduino IDE**

1. Download and install the **Arduino IDE** from [arduino.cc](https://www.arduino.cc/en/software) if not already installed.
2. Connect the **Arduino Uno** to the computer using the **USB A-to-B cable**.
3. Verify the Arduino is recognized by the computer (check **Device Manager** on Windows or **System Information** on macOS/Linux for a COM port).

### **Step 3: Install Required Libraries**

1. Open the **Arduino IDE**.
2. Navigate to **Sketch > Include Library > Manage Libraries**.
3. Search for and install the following Pololu libraries:
   * **LSM6**: For the LSM6DSO (gyroscope and accelerometer).
   * **LIS3MDL**: For the LIS3MDL (magnetometer).
4. Alternatively, download the libraries from Pololu’s GitHub:
   * [LSM6 library](https://github.com/pololu/lsm6-arduino)
   * [LIS3MDL library](https://github.com/pololu/lis3mdl-arduino)
   * Install them via **Sketch > Include Library > Add .ZIP Library**.
5. Confirm the libraries are installed by checking **Sketch > Include Library** for LSM6 and LIS3MDL entries.

### 

### 

### 

### 

### **Step 4: Upload the Test Sketch**

Copy and paste the following test sketch into a new Arduino IDE window:  
  
  
#include <Wire.h>

#include <LSM6.h>

#include <LIS3MDL.h>M6 imu

void setup() {

Serial.begin(9600);

Wire.begin();

*// Initialize LSM6DSO (gyro and accelerometer)*

if (!imu.init()) {

Serial.println("Failed to initialize LSM6DSO!");

while (1);

}

imu.enableDefault();

*// Initialize LIS3MDL (magnetometer)*

if (!mag.init()) {

Serial.println("Failed to initialize LIS3MDL!");

while (1);

}

mag.enableDefault();

}

void loop() {

*// Read accelerometer and gyro data*

imu.read();

Serial.print("Accel (g): ");

Serial.print(imu.a.x \* 0.000061); *// Convert raw to g (±2g range)*

Serial.print(", ");

Serial.print(imu.a.y \* 0.000061);

Serial.print(", ");

Serial.print(imu.a.z \* 0.000061);

Serial.print(" | Gyro (dps): ");

Serial.print(imu.g.x \* 0.00875); *// Convert raw to dps (±250dps range)*

Serial.print(", ");

Serial.print(imu.g.y \* 0.00875);

Serial.print(", ");

Serial.print(imu.g.z \* 0.00875);

*// Read magnetometer data*

mag.read();

Serial.print(" | Mag (gauss): ");

Serial.print(mag.m.x \* 0.00014); *// Convert raw to gauss (±4 gauss range)*

Serial.print(", ");

Serial.print(mag.m.y \* 0.00014);

Serial.print(", ");

Serial.print(mag.m.z \* 0.00014);

Serial.println();

delay(500); *// Update every 500ms*

}

**Explanation**:

* + Initializes I²C communication and the Serial Monitor.
  + Checks if the LSM6DSO and LIS3MDL initialize correctly.
  + Reads raw data from the accelerometer, gyroscope, and magnetometer.
  + Converts raw values to physical units (g, degrees per second, gauss) using default scaling factors.
  + Outputs data to the Serial Monitor every 500ms.

### **Step 5: Upload and Run the Test**

1. In the Arduino IDE, select **Tools > Board > Arduino Uno**.
2. Select the correct port under **Tools > Port** (e.g., COM3 on Windows or /dev/ttyACM0 on Linux).
3. Click **Upload** to compile and upload the sketch to the Arduino Uno.
4. Open the **Serial Monitor** (**Tools > Serial Monitor**) and set the baud rate to **9600**.
5. Observe the output, which should look like:  
     
   Accel (g): 0.01, -0.02, 0.98 | Gyro (dps): 0.1, -0.2, 0.0 | Mag (gauss): 0.15, -0.10, 0.30

### 

### 

### 

### **Step 6: Verify Sensor Functionality**

1. **Accelerometer Test**:
   * Hold the module flat (Z-axis up). The Z-axis should read approximately 1g (due to gravity).
   * Tilt the module along the X and Y axes. Values should change smoothly (e.g., X increases when tilted right).
2. **Gyroscope Test**:
   * Rotate the module around each axis (X, Y, Z). Values should spike during rotation and return to near zero when stationary.
3. **Magnetometer Test**:
   * Move the module in different directions or near a weak magnet. Values should change based on orientation or magnetic fields.
   * Note: Magnetometer readings may vary indoors due to interference.
4. If all sensors show responsive values, the module is functional.

### **Step 7: Record Results**

1. Log the Serial Monitor output for at least 10 seconds to confirm consistent data.
2. Note any anomalies (e.g., unchanging values, initialization errors).
3. If the module passes all tests, mark it as **functional**. If not, proceed to troubleshooting.

## **Troubleshooting**

* **No Serial Output**:
  + Verify the Serial Monitor baud rate is set to 9600.
  + Check the USB cable and port selection.
  + Ensure the Arduino Uno is powered (LED on).
* **Initialization Errors** (e.g., “Failed to initialize LSM6DSO!”):
  + Double-check wiring (VIN to 5V, GND to GND, SDA to A4, SCL to A5).
  + Inspect solder joints on the xIMU\_V527 for defects.
  + Test with a different xIMU\_V527 to isolate module vs. setup issues.
* **Erratic Values**:
  + Secure all connections to eliminate loose wires.
  + Move the module away from metal objects or magnetic interference.
  + Verify the Arduino’s 5V pin provides stable voltage (use a multimeter if needed).
* **Library Issues**:
  + Reinstall the LSM6 and LIS3MDL libraries.
  + Ensure no conflicting libraries are installed (e.g., generic LSM6DSO libraries).

## 

## **Pass/Fail Criteria**

* **Pass**: All three sensors (accelerometer, gyroscope, magnetometer) produce responsive, reasonable values in the Serial Monitor, changing appropriately with movement.
* **Fail**: Any sensor fails to initialize, produces no data, or outputs unresponsive/erratic values after troubleshooting.

## **Additional Notes**

* This test confirms basic functionality. For advanced applications (e.g., AHRS), further calibration or testing may be required.
* Refer to the [LSM6DSO datasheet](https://www.st.com/resource/en/datasheet/lsm6dso.pdf) and [LIS3MDL datasheet](https://www.st.com/resource/en/datasheet/lis3mdl.pdf) for detailed sensor specifications.
* If multiple modules are tested, ensure each is tested individually to avoid I²C address conflicts.