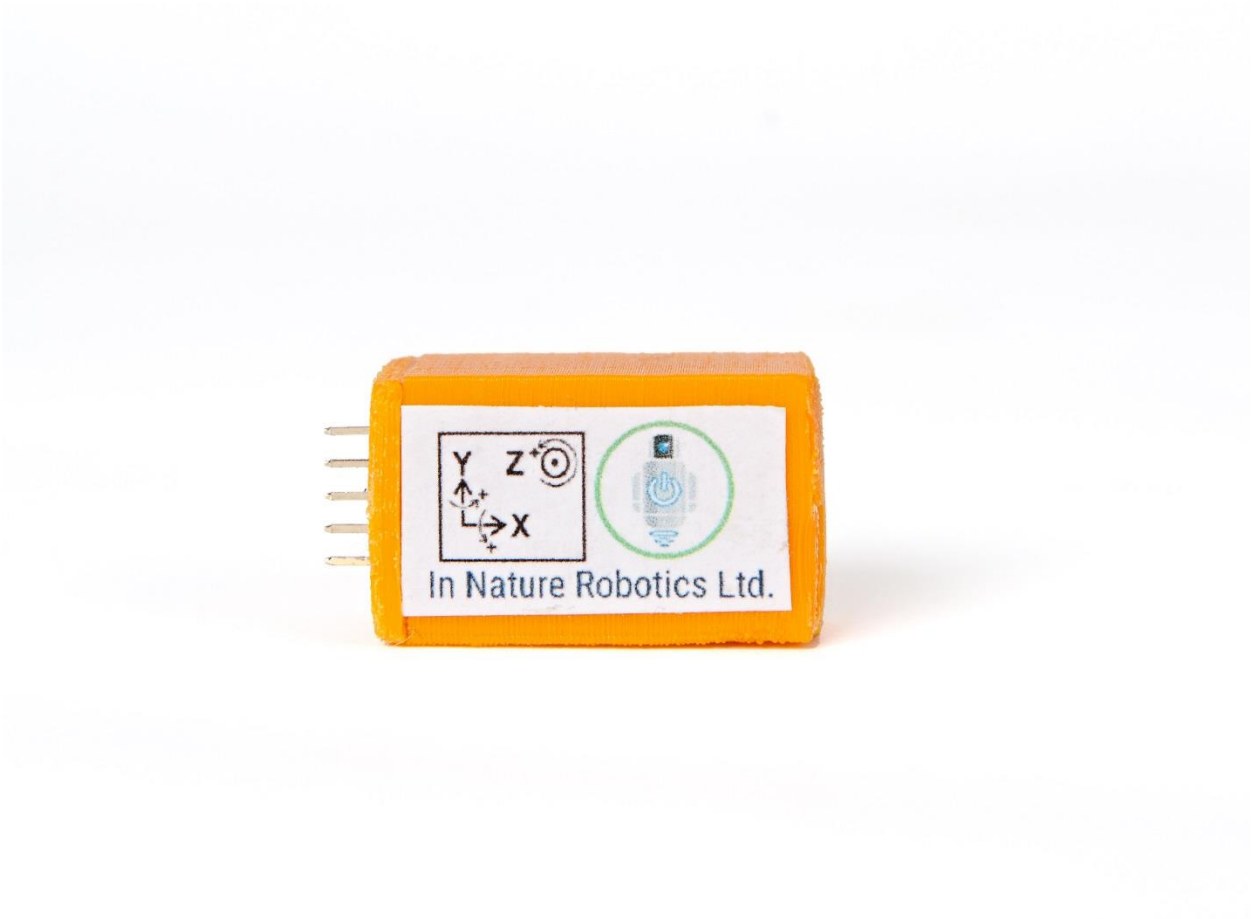
In Nature Robotics Ltd. www.innaturerobotics.com

AMOS-IMU User Guide



Introduction

The AMOS-IMU is an inertial measurement unit that can be used for precisely measuring the orientation of a device in a dynamic environment. It includes triaxial accelerometers, magnetometers, and angular rate (gyroscope) sensors that allow for the computation of 3-D orientation angles in real-time. In Nature Robotics Ltd. provides software libraries and individual calibration data for each AMOS-IMU device.

Please refer to the following additional documents concerning the individual component pieces of the AMOS-IMU:

altimu-10-v5-dimension-diagram.pdf: Dimension diagram of the circuit board used.

altimu-10-v5-schematic.pdf: Schematic diagram of the circuit board used.

LIS3MDL-AN4602.pdf: Application note for the 3-axis digital output magnetic sensor.

LIS3MDL.pdf: 3-axis ultra-low power, high performance digital output magnetic sensor.

LSM6DS33.pdf: iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope

LPS25H.pdf: MEMS pressure sensor: 260-1260 hPa absolute digital output barometer

UM10204.pdf: I^2C-bus specification and user manual

The AMOS-IMU uses an I^2C communications interface, and functions as an I^2C slave device. The software libraries and examples provided are for the Arduino and Raspberry Pi platforms.

Absolute Maximum Ratings(1)

Rating	Maximum Limit	Units
VDD to GND	-0.3 to +5.5	V
Analog input current	100, momentary	mA
Analog input current	10, continuous	mA
Analog input voltage to GND	-0.3 to VDD + 0.3	V
SDA, SCL, ADDR, ALERT/RDY voltage to GND	-0.5 to +5.5	V
Maximum junction temperature	+150	°C
Storage temperature range	-60 to +150	°C

(1) Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. Exposure to absolute maximum conditions for extended periods may affect device reliability.

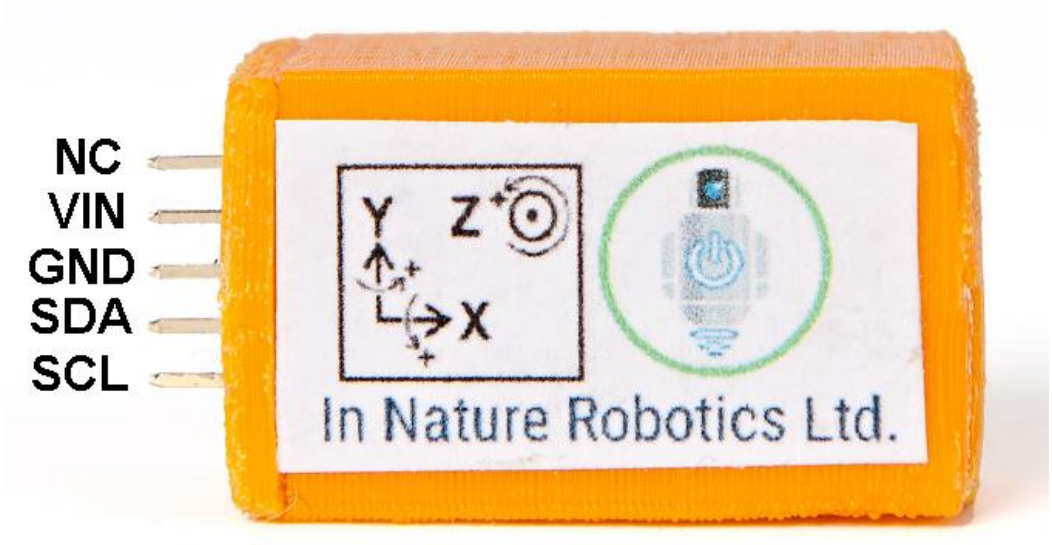
Dimensions

Size	38 mm x 20 mm x 16 mm
Weight	9.1 g

General Specifications

For detailed specifications, please refer to the documents described above.

Interface	I^2C
Minimum operating voltage	2.5V
Maximum operating voltage	5.5V
Axes	Pitch (x), roll (y), and yaw (z)
Measurement ranges	±125, ±245, ±500, ±1000, or ±2000°/s (gyro) ±2, ±4, ±8, or ±16 g (accelerometer) ±4, ±8, ±12, or ±16 gauss (magnetometer) 26 kPa to 126 kPa (barometer)
Supply current	5 mA



Pin names / functions for AMOS-IMU:

NC	No connect, leave open
VIN	Input voltage, 2.5V to 5.5V
GND	Ground
SDA	I^2C data line
SCL	I^2C clock line

Sample Code

Arduino Uno Sample Code

The following program collects data from the accelerometers, magnetometers, gyros, and temperature sensors. It requires some modified LIS3MDL and LSM6 library files to be installed. These files are available from the In Nature Robotics support page: https://www.innaturerobotics.com/support

```
//program for testing the output of the AMOS-IMU module as a function of
temperature
//outputs lines of data out serial port (i.e. does not save any data locally)
#include <Wire.h>
#include <LIS3MDL.h>
#include <LSM6.h>

LIS3MDL mag;
```

```
LSM6 imu;
LIS3MDL::vector<int16_t> magData;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Wire.begin();

  if (!mag.init())
  {
    Serial.println("Failed to detect and initialize magnetometer!");
    while (1);
  }
  if (!imu.init()) {
    Serial.println("Failed to detect and initialize IMU!");
    while (1);
  }
  mag.enableDefault();
  imu.enableDefault();
}

void loop() {
  mag.read();
  imu.read();
  Serial.print(mag.m.x);
  Serial.print(", ");
  Serial.print(mag.m.y);
  Serial.print(", ");
  Serial.print(mag.m.z);
  Serial.print(", ");
  Serial.print(mag.m_fTempDegC);
  Serial.print(", ");
  Serial.print(imu.a.x);
  Serial.print(", ");
  Serial.print(imu.a.y);
  Serial.print(", ");
  Serial.print(imu.a.z);
  Serial.print(", ");
  Serial.print(imu.g.x);
  Serial.print(", ");
   Serial.print(imu.g.y);
  Serial.print(", ");
   Serial.print(imu.g.z);
  Serial.print(", ");
```

```
  Serial.print(imu.m_fTempDegC);
  Serial.println();
  delay(1000);
}
```

Raspberry Pi Sample Code

The following program collects data from the accelerometers, magnetometers, gyros, and temperature sensors, and uses that information to compute accelerations, a magnetic vector, angular rates, and roll, pitch, and yaw orientation angles. It requires the IMU.cpp and IMU.h library files for the Raspberry Pi to be installed. These files are available from the In Nature Robotics support page: https://www.innaturerobotics.com/support

```
#include "../RemoteControlTest/IMU.h"
#include <pthread.h>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <memory>


//example program that tests out the operation of the AltIMU-10 v5 Gyro,
Accelerometer, Compass, and Altimeter from Pololu Electronics (www.pololu.com)

/**
 * @brief return true if a magnetometer calibration flag (-magcal) was specified
in the program arguments
 *
 * @param argc the number of program arguments
 * @param argv an array of character pointers that corresponds to the program
arguments
 * @return true if a magnetometer calibration flag (-magcal) is present in the
array of program arguments
 * @return false if no magnetometer calibration flag is present in the array of
program arguments.
 */
```

```
bool isMagCalFlagPresent(int argc, char * argv[]) {
  for (int i=0;i<argc;i++) {
    if (strlen(argv[i])<7) continue;
    if (strncmp(argv[i],"-magcal",7)==0) {
      return true;
    }
  }
  return false;
}


/**
 * @brief the main function for the IMUTest program
 *
 * @param argc the number of program arguments specified on the command line to
the program
 * @param argv a character array corresponding to each of the program arguments
 * @return int the return code from the IMUTest program
 */
int main(int argc, char * argv[])
{
  const int NUM_SAMPLES = 100;
  const int NUM_TO_AVG = 1;//number of individual samples to average for each
call to IMU::GetSample
  pthread_mutex_t i2cMutex = PTHREAD_MUTEX_INITIALIZER;;//mutex for controlling
access to i2c bus
  IMU imu(&i2cMutex);
  if (imu.m_bInitError) {
    printf("An error occurred trying to initialized the IMU.\n");
    return -1;
  }
  if (isMagCalFlagPresent(argc,argv)) {
    if (!imu.DoMagCal()) {
      printf("Error doing magnetometer calibration.\n");
      return -2;
    }
  }
  IMU_DATASAMPLE imu_sample;
  for (int i=0;i<NUM_SAMPLES;i++) {
    if (!imu.GetMagSample(&imu_sample, NUM_TO_AVG)) {//collect raw magnetometer
data from the LIS3MDL 3-axis magnetometer device and process it to get the
magnetic vector and temperature
      printf("Error getting magnetometer sample #%d.\n",i+1);
      return -2;
```

```
    }
    if (!imu.GetAccGyroSample(&imu_sample, NUM_TO_AVG)) {//collect accelerometer,
gyro, and temperature data from the LSM6DS33 3-axis acc/gyro device
      printf("Error getting acc/gyro sample #%d.\n",i+1);
      return -3;
    }
    //sample #, accX, accY, accZ, gyroX, gyroY, gyroZ, temperature
    //un-comment the following lines to show mag, acc, and gyro data
     printf("%d (%.3f sec): magX = %.6f, magY = %.6f, magZ = %.6f, accX = %.6f,
accY = %.6f, accZ = %.6f, gyroX = %.3f, gyroY = %.3f, gyroZ = %.3f, temperature =
%.1f deg C\n",

i+1,imu_sample.sample_time_sec,imu_sample.mag_data[0],imu_sample.mag_data[1],imu_
sample.mag_data[2],
       imu_sample.acc_data[0],imu_sample.acc_data[1],imu_sample.acc_data[2],

imu_sample.angular_rate[0],imu_sample.angular_rate[1],imu_sample.angular_rate[2],
imu_sample.acc_gyro_temperature);

    imu.ComputeOrientation(&imu_sample);
    printf("%d (%.3f sec): roll = %.1f deg, pitch = %.1f deg, heading = %.1f
deg\n",i+1,imu_sample.sample_time_sec,imu_sample.roll,imu_sample.pitch,imu_sample
.heading);
  }
  return 0 ;
}
```