IMU-Capture 0.2.2

User manual

November 10, 2023

David Buckingham

Contents

1 Copyright and License 3

2 Introduction 3

3 Hardware 3
- 3.1 PC . . . 3
- 3.2 Arduino . . . 3
- 3.3 IMUs . . . 3
- 3.4 SPI . . . 4
- 3.5 Wiring the Arduino . . . 4

4 Installation 4
- 4.1 Arduino . . . 4
- 4.2 PC . . . 6

5 Main control panel 6
- 5.1 Starting the program . . . 6
- 5.2 Record . . . 7
- 5.3 Load . . . 7
- 5.4 Batch transform . . . 8
- 5.5 Quit . . . 8
- 5.6 Trigger . . . 8

6 Data window 8
- 6.1 Data buffer . . . 8
- 6.2 Plots . . . 8
- 6.3 Save . . . 9
- 6.4 Close . . . 9
- 6.5 Calibrate . . . 9

7 Transform 10
- 7.1 Integration algorithm . . . 11

8 Troubleshooting 11
- 8.1 Installation . . . 11
- 8.2 General error messages . . . 11
- 8.3 Calibration error messages . . . 13

1 Copyright and License

2 Introduction

IMU-Capture is an application for recording and processing data from inertial measurement units (IMUs). The software has two parts. Microcontroller code running on an Arduino board collects data from the IMUs and transmits it to a PC. A Python program running on the PC receives data from the Arduino and provides users with a graphical interface. This interface allows users to interact with the IMU via the Arduino and to view and manipulate IMU data.

3 Hardware

IMU-Capture interconnects three pieces of computational hardware: a PC, an Arduino, and between 1 and 3 IMUs.

3.1 PC

IMU-Capture has been tested on PCs running Linux Mint 18 Sarah, OS X El Capitan 10.11, and Windows 10.

3.2 Arduino

IMU-Capture has been tested with an Arduino UNO. It may work on other models that use a 16MHz clock speed and have sufficient storage.

3.3 IMUs

The mpu9250 by InvenSense is a nine-axis (gyroscope, accelerometer, compass) motion tracking device.

For applications requiring minimum package size and weight, the mpu9250 can be wired directly to the Arduino. A separate manual documents the procedure we have used to prepare the mpu9250 for use with IMU-Capture: `http://www.url_for_cassandras_manual.com`

For testing purposes, or if the added size and weight are acceptable, an mpu9250 mounted on a circuit-board can be used.

3.4 SPI

The Arduino communicates with the IMUs using the Serial Peripheral Interface bus (SPI) protocol. With SPI, one or more slave devices (IMUs in this application) exchange data with a master device (the Arduino) over a single bus consisting of 2 data lines and a clock line. In addition, each slave device has a separate *chip select* line to mediate access to the shared bus.

3.5 Wiring the Arduino

Table 1 summarizes the process of connecting the IMUs to the Arduino. It may be necessary to use a breadboard, especially if multiple IMUs are used. Figure 3.5 shows an Arduino wired to a single IMU. Pins 8, 9, and 10 are used for *chip select* lines for up to three IMUs. Pin 11 carries data traveling from the Arduino to the IMUs, i.e. Master-Out, Slave-In (MOSI). Pin 12 carries data traveling from the IMUs to the Arduino, i.e. Master-In, Slave Out (MISO). Pin 13 carries a clock signal which regulates timing of the communication protocol. Each IMU should be connected to 3.3V power (available on the Arduino Uno) and to ground.

If using a trigger, connect it to Pin 4 and to ground.

Connect the Arduino to the PC with a USB cable. To ensure adequate power, especially if using multiple IMUs, it may be necessary to connect the Arduino to an external power source instead of relying on the USB port for power.

4 Installation

4.1 Arduino

The portion of IMU-Capture that runs on an Arduino is called `ic_arduino.ino`. It is located inside the directory `ic_arduino` and is distributed with the rest of IMU-Capture. We have succesfully installed `ic_arduino.ino` on an Arduino Uno using the Arduino IDE version 1.8.1. Earlier versions of the Arduino IDE may lack some of library definitions used by IMU-Capture. Use the Arduino IDE 1.8.1 or some other means to install `ic_arduino.ino` on the Arduino.

The Arduino IDE can be downloaded from: `https://www.arduino.cc/en/Main/Software`

description	label	color	pin
IMU 1 chip select	NCS / CS / SS	white	8
IMU 2 chip select	NCS / CS / SS	white	9
IMU 3 chip select	NCS / CS / SS	white	10
data from Arduino to IMU	MOSI / SDI / SDA	green	11
data from IMU to Arduino	MISO / SDO / ADO	blue	12
clock	SCL / CLK / SCK	yellow	13
power	VCC	red	3.3V
ground	GND	black	GND
trigger			4

Table 1: Instructions for wiring the IMUs to the Arduino. The first column describes the purpose of each line. The second column provides names commonly used to describe each line. The third column lists the colors specified in the IMU wiring manual. The fourth column specifies pins on the Arduino Uno.

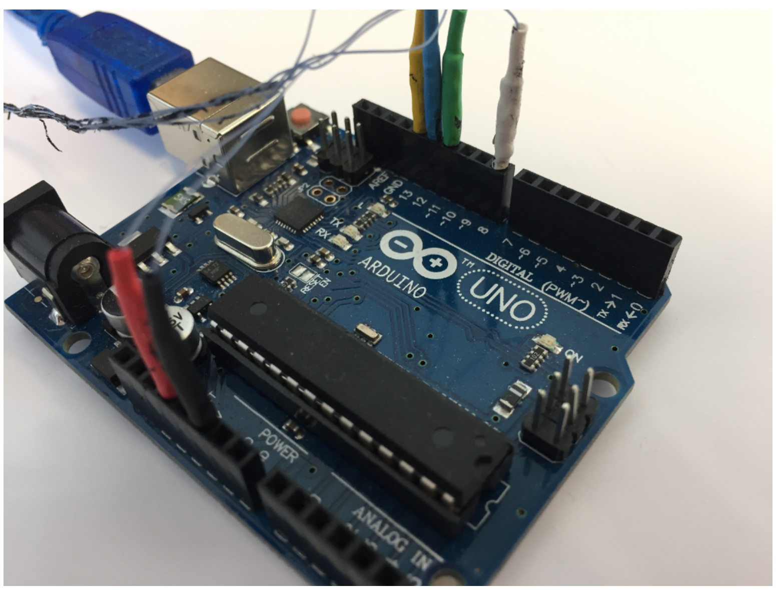

Figure 1: Wiring a single IMU to the Arduino

4.2 PC

MSI

An MSI installation file is provided for Windows users. It includes the python interpreter and all required libraries. It has been tested on Windows 10. To install IMU-Capture, simply download and run the installer.

Pip

IMU-Capture requires Python 3. It has been tested with Python version 3.5. Thus, it is recommended to use Python version 3.5 or later. With Python 3 installed, you can use pip (the Python Package Index) to install IMU-Capture:

```
# pip install --upgrade pip
# cd /path/to/imucapture/
# pip install ./
```

Pip should automatically install any of the following dependencies, and any of their sub-dependencies, if needed:

- H5py
- PyQtGraph
- pySerial
- PyQt5
- pyquaternion

5 Main control panel

5.1 Starting the program

If IMU-Capture was installed with the Windows installer, it can be started by double-clicking the executable (.exe) icon or desktop shortcut.

If IMU-Capture was installed with pip, it can be started from the command line:

```
# IMU-Capture
```

On Windows, if IMU-Capture is installed with pip and invoked from the command line, informational and error messages will not be displayed unless IMU-Capture is invoked by the Python interpreter explicitly:

```
# python -m IMU-Capture
```

When IMU-Capture starts, the main control panel will be visible (Figure 2).

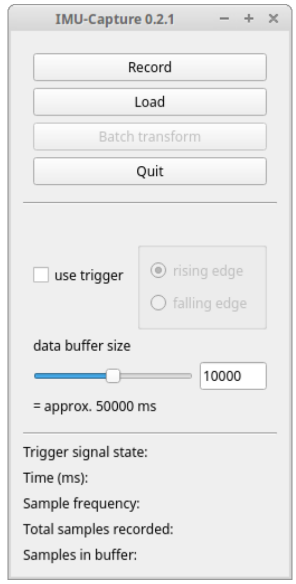

Figure 2: Main control panel

5.2 Record

To begin collecting data, press the *Record* button. The *Record* button changes into the *Stop* button. The PC establishes communication with to the Arduino and instructs it to begin collecting data from the IMUs at 200Hz. This sample rate is hard-coded, and was determined to be near the maximum possible with the Arduino Uno.

A new data window containing an empty data buffer is opened. As data samples are received from the Arduino, each sample is added to data buffer. The *data buffer length (# samples)* slider adjusts the size of the data buffer to which recorded data is currently being written. When each new sample is received, it is added to the data buffer. If the buffer is already full (the number of samples in the buffer is equal to the size of the buffer), the oldest sample in the buffer is deleted. If the size of the buffer is adjust to be smaller than the number of samples in the buffer, the oldest samples in the buffer are deleted until the number of samples in the buffer is equal to the buffer length.

The *Stop* button (or the trigger, as described in Section 5.6), causes the PC first to instruct the Arduino to stop collecting data and then to halt communication with the Arduino.

5.3 Load

The *Load* button opens a dialog allowing the user to select a file to load. A new data window will be created and the file data will be loaded into its data buffer. The *Load* button is disabled while data is being recorded.

5.4 Batch transform

This feature is not enabled.

5.5 Quit

Halt any activity, close connections, and quit the program.

5.6 Trigger

Optionally, a trigger attached to the Arduino (Section 3.5) can be used to stop recording. If the *use trigger* checkbox is not checked, the trigger is ignored. Otherwise, if an active trigger is detected, recording will stop, i.e. the same effect as pressing the *Stop* button during recording. If the *Record* button is pressed while the *use trigger* checkbox is checked and the trigger is active, the recording immediately ends with zero data stored.

If the *invert trigger* checkbox is not checked, the trigger is considered active when the associated Arduino pin is set high. If the *invert trigger* checkbox is checked, the trigger is considered active when the associated Arduino pin is set low.

If no trigger is connected, the value of the pin is undefined. Thus, to ensure reliable behavior, the *use trigger* checkbox should be unchecked unless there is a trigger connected to the Arduino.

The UTC timestamp of the PC when the trigger is detected is recorded in the data. The "trigger delay" control determines how long data will continue being recorded after the trigger is pressed.

6 Data window

6.1 Data buffer

Each data window has an associated data buffer.

As samples received from the Arduino are stored in a data buffer, they become visible in the data visualization plots.

6.2 Plots

For each IMU, one row of three plots is displayed. The left-most plot shows accelerometer data, the center plot shows gyroscope data, and the right-most plot shows magnetometer data. For each plot, red, green, and blue lines show x, y, and z axis measurements, respectively.

Each plot will scale automatically as the data buffer is updated. Plots can also be adjusted manually using the mouse.

Table 2 lists the units used for each sensor modality.

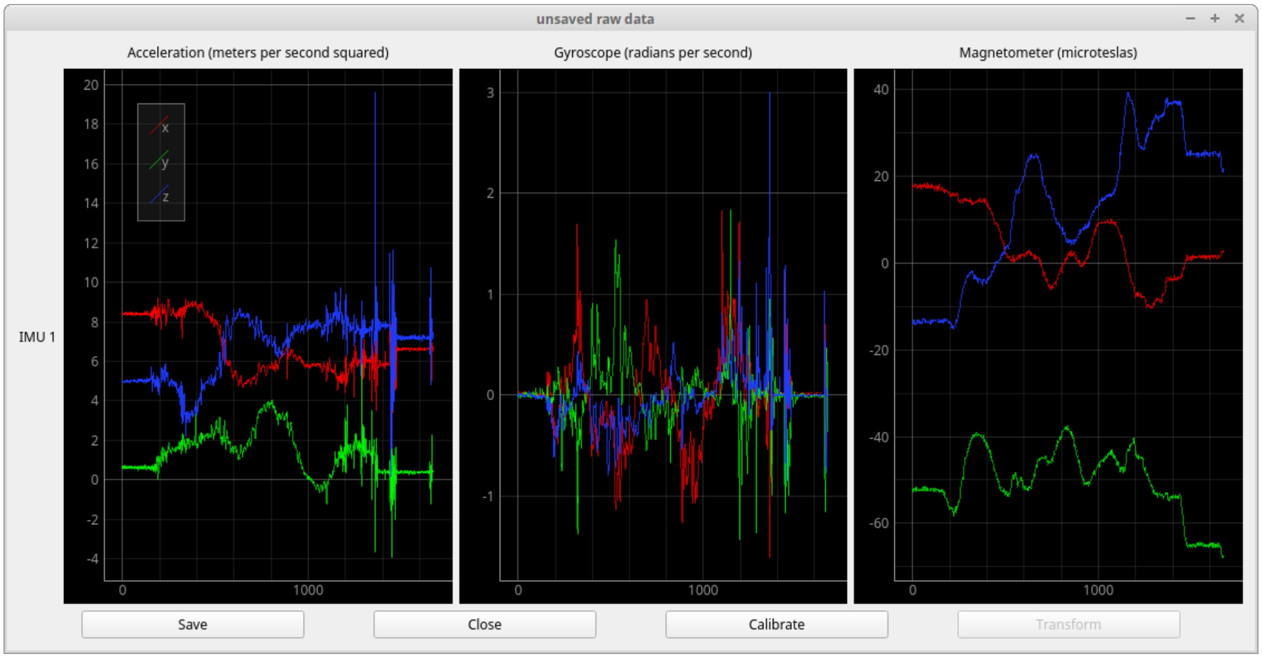

Figure 3: Collecting data from a single IMU

instrument	modality	units
accelerometer	acceleration	meters per second squared
gyroscope	rotational speed	radians per second
magnetometer	magnetic flux density	microteslas

Table 2: Units used by IMU-Capture

6.3 Save

The *Save* button opens a dialog allowing the user to select a file system location and a file name. The data buffer is saved to file according to the selections made in the dialog. The *Save* button is disabled if the data buffer is empty or while data is being recorded.

6.4 Close

Close the data window. Any unsaved data in the data buffer will be lost.

6.5 Calibrate

Before any of the transformation algorithms can be applied, calibration data must be used to calibrate the sytem. Three main pieces of information are extracted from the calibration data:

1. Three orthogonal orientations are used to construct a set of basis vectors
2. The first of the three orientations is used to determine the initial gravity vector
3. The first of the three orientations is used to determine noise covariance matrices

Follow these steps to create a calibration file.

1. Wire up the IMU as you will use it to collect data
2. Hold the IMU in an orientation to be considered “upright”
3. Begin recording
4. Hold the IMU upright, in the first orthogonal orientation, for about 5 seconds
5. Rotate the IMU 90 degrees so that the new orientation is orthogonal to the first
6. Hold the IMU in the second orthogonal orientation for about 5 seconds
7. Rotate the IMU 90 degrees again so that the new orientation is orthogonal to both previous orientations
8. Hold the IMU in the third orthogonal orientation for about 5 seconds
9. Stop recording
10. Press the *Calibrate* button.

If calibration fails, error messages explaining why can help determine if there are problems with the calibration data and help to record correct calibration data. See Section 8.3 for more discussion about these messages.

7 Transform

Due to factors such as the effect of gravity upon the accelerometer, sensor drift, and noise, the raw data collected by IMU-Capture will not reflect the actual dynamics of the IMUs. IMU-Capture includes several algorithms that can be applied to IMU data after it has been recorded. Currently, it is not possible to apply these algorithms in real time as the data is recorded. Running one of these algorithms will have to main effects upon recorded data: 1) The effect of gravity is removed from the acceleration data, leaving only *dynamic* acceleration. 2) The rotational speed data is converted into rotational angle data, that is, the rotation of the IMUs is integrated over time.

Currently, filtering algorithms only apply to a single IMU.

It is only possible to transform data after a calibration has been set. Transformations should not be applied to callibration data, even though it is possible to do so.

7.1 Integration algorithm

The *Integration algorithm* radio buttons select the data filtering algorithm.

DSF The Dynamic Snap Free algorithm developed by Vikas: (cite)

Madgwick
: An implementation of the filter developed by Madgwick: `http://x-io.co.uk/res/doc/madgwick_internal_report.pdf`

Simple integration
: Use the Madgwick algorithm with the beta parameter set to 0.

8 Troubleshooting

8.1 Installation

If you encounter problems during installation using pip, make sure you are using Python 3.5 or later and that pip is working with the correct Python version.

8.2 General error messages

This sections discusses some of the error messages that may be reported by IMU-Capture.

This section provides explanation and troubleshooting tips for each error message produced by IMU-Capture.

ASA read failed, using 1 adjustment
: For each magnetometer axis, a sensitivity adjustment value (ASA) is stored in ROM by the manufacturer. This error message is reported when the Arduino is unable to read the ASA values from an IMU. It likely indicates that the Arduino is not communicating correctly with the magnetometer. Any magnetometer data recorded after this message should be discarded. Try resetting the Arduino. Make sure that the Arduino is correctly powered and connected to the PC (Section 3.5), and that the correct code is installed on the Arduino (Section 4.1).

failed to create connection, aborting
: The program failed to establish a serial connection with the Arduino. Try resetting the Arduino. Make sure that the Arduino is correctly powered and connected to the PC (Section 3.5), and that the correct code is installed on the Arduino (Section 4.1).

handshake failed
: Even though the PC may have established a valid serial connection to the Arduino, the data exchange

protocol used by IMU-Capture failed to establish a communication handshake with the Arduino. Try resetting the Arduino. Make sure that the Arduino is correctly powered and connected to the PC (Section 3.5), and that the correct code is installed on the Arduino (Section 4.1). It is possible the IMU-Capture incorrectly identified a serial device as the Arduino. Any device that the pySerial library (`https://pythonhosted.org/pyserial/#`) identifies as manufactured either by "Arduino" or by "Microsoft" will be identified as an Arduino by IMU-Capture. Try disconnecting all serial devices and then reconnecting them, starting with the Arduino.

invalid file type:...

There was an attempt either to save or to load a file type other than `.hdf5`.

no Arduino found

The program searched for Arduinos on all serial ports but didn't find any. IMU-Capture uses the pySerial library to scan for Arduinos: `https://pythonhosted.org/pyserial/#`

no IMUs detected, aborting

The Arduino did not detect any attached IMUs. After the Arduino is initialized, it attempts to determine the number of IMUs by sending a WHOAMI request while signaling each of the three legal chip select pins (Section 3.5). It then sends a message to the PC reporting the number of responses received. This error is reported if the Arduino does not receive any WHOAMI responses. Try resetting the Arduino. Make sure that the Arduino is correctly powered and connected to the PC (Section 3.5), and that the correct code is installed on the Arduino (Section 4.1). Make sure at leaste 1 IMU is connected to the arduino, as described in Section 3.5.

rx failed, no data read from serial

The PC expected to receive data from the Arduino but failed. Perhaps no data was transmitted, or perhaps data that violates the communication protocols used by IMU-Capture was received. Try resetting the Arduino. Make sure that the Arduino is correctly powered and connected to the PC (Section 3.5), and that the correct code is installed on the Arduino (Section 4.1).

unknown sample received:...

The PC expected to receive a packet containing a data sample, but the packet either had the wrong type or the wrong length. This message can be ignored if it occurs only briefly at the beginning of recording. Otherwise, try resetting the Arduino.

unable to determine number of IMUs, aborting

The program failed to determine how many IMUs are attached to the Arduino. After the Arduino is initialized, it attempts to determine the number of IMUs by sending a WHOAMI request while signaling each of the three legal chip select lines (Section 3.5). It then sends a message to the PC reporting the number of IMUs detected. This error is reported if the PC sends a command to the Arduino to initialize, but does not receive a message reporting the number of IMUs detected. Try resetting the Arduino. Make sure that the Arduino is correctly powered and connected to the PC (Section 3.5), and that the correct code is installed on the Arduino (Section 4.1). Make sure at leaste 1 IMU is connected to the arduino, as described in Section 3.5.

8.3 Calibration error messages

This section discusses error messages that may be reported when IMU-Capture fails to calibrate.

fewer than 3 steady intervals

A "steady interval" is a contiguous duration of at least 3 seconds when the change in accelerometer measurement remains below a threshold and and the gyroscope measurement remains below a threshold. Possibly the IMU was not held still enough, or the duration of one or more hold was too short.

too many steady intervals, the limit is...

See the discussion for "fewer than 3 steady intervals" for a definition of "steady interval". IMU-Capture can remove some some extra steady intervals if they are not part of a triple of orthogonal vectors, but too many will cause this error. When recording the calibration file, start recording when with the IMU in the first orientation, end recording immediately after 5 seconds of the third orientation, and make smooth, quick, transitions between the orientations.

found more than 1 triple of orthogonal vectors

More than three orientations were recorded, and more than one triple of orthogonal vectors can be identified. This is similar to "too many steady intervals, the limit is...". See the discussion of that message for possible solutions.

could not find 3 orthogonal vectors

Even though at least 3 "steady intervals" (see the discussion of the "fewer than 3 steady intervals" message) were found, no 3 intervals are orthogonal. That is, the difference between 90 degrees and the angles separating the intervals was greater than a threshold. When recording the calibration file, ensure that that three orientations are mutually orthogonal.