PROGRAM_NAME 0.1

User manual

David Buckingham
Eric Tytell
Cassandra Donatelli
Margo

Contents

0.1 Copyright and License

0.2 Introduction

0.3 Hardware

PROGRAM_NAME interconnects three pieses of computational hardware:

1. a PC
2. an Arduino
3. between 1 and 3 Inertial Measurement Units (IMUs)

0.3.1 PC

PROGRAM_NAME has been tested on PCs running Linux Mint 18 Sarah, OS X El Capitan 10.11, and Windows 10. It is expected to work with other versions of these operating systems and with other Linux distributions.

0.3.2 Arduino

PROGRAM_NAME has been tested with an Arduino UNO. It may work on other models having a 16MHz clock speed and sufficient storage.

0.3.3 IMUs

The mpu9250 by InvenSense is a nine-axis (gyroscope, accelerometer, compass) motion tracking device. For testing purposes, or if the added size and weight are acceptable for your application, a mpu9250 mounted on a circuit-board can be used with PROGRAM_NAME.

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

For applications requiring minimum package size and weight, it is possible to wire the mpu9250 directly to the arduino. A separate manual, **name**, documents the procedure we have used to prepare the mpu9250 for use with PROGRAM_NAME.

0.3.4 Wiring the Arduino

Wire each IMU to the Arduino. It may be necessary to use a breadboard, especially if multiple IMUs are used. Table 0.3.4 summarizes the wiring process. The Arduino communicates with the IMUs using the Serial Peripheral Interface bus (SPI) protocol. Each IMU needs a separate *chip select* line, but all IMUs share the other lines. Pins 8, 9, and 10 are used for *chip select* lines for up to three IMUs. Pin 11 is for data traveling from the Arduino to the IMUs, i.e. Master-Out, Slave-In (MOSI). Pin 12 is for data traveling from the IMUs to the Arduino, i.e. Master-In, Slave Out (MISO). Pin 13 is for a clock signal, used to regulate timing of the communication protocol.

If using an optional trigger, wire it to Pin 4.

In addition, each IMU should be connected to 3.3V power (available on the Arduino) and each IMU and the trigger should be connected to Ground.

Connect the arduino to the PC with a USB cable.

To ensure adequate power, especially if using multiple IMUs, it is recommended to power the Arduino with an external power source instead of relying on the USB port.

0.4 Installation

0.4.1 Arduino code

Arduino IDE

While there are many tools for installing program code onto the Arduino, we have tested PROGRAM_NAME using the Arduino IDE.

The software can be downloaded from: urlhttps://www.arduino.cc/en/Main/Software

If you'r operating system has a package management system, it might be able to install the Arduino IDE. For example, on Debian systems, use:

```
# apt-get install arduino-core
```

Install PROGRAM_NAME on the Arduino

Install the file `am_tx/am_tx.ino` on the Arduino. This can be accomplished using the Arduino IDE graphical user interface.

Alternatively, using Arduino IDE version 1.5.0 or later, the Arduino can be programmed directly from the command line:

```
# arduino --upload am_tx/am_tx.ino
```

For more information about the Arduino command line interface, see: https://github.com/arduino/Arduino/blob/master/build/shared/manpage.adoc

0.4.2 PC

PROGRAM_NAME requires Python3.

We recommend using pip to install PROGRAM_NAME: https://pypi.python.org/pypi/pip

```
# pip install --upgrade pip
# pip install \name
```

Pip will automatically install any of the following dependencies if needed:

- h5py
- numpy
- pyqtgraph
- pyserial
- pyqt5

After install PROGRAM_NAME, it can be started from the command line:

```
# PROGRAM_NAME
```

0.5 The data buffer

PROGRAM_NAME uses a single data buffer to hold IMU data. When data is recorded from the IMUs, the data buffer is first errased. Then each sample is added to the buffer as it is recorded. The buffer can be saved to a file, or a file can be loaded into the buffer, erasing any previous contents. Data processing operations can be performed on the data buffer, altering it irreversibley. Therefore, any time the data buffer contains valuable data it is recommended to save it to file before collecting new data, loading another file, or executing data processing operations.

0.6 Collecting data

To begin collecting data, press the "record" button. The PC will connect to the Arduino and instruct it to begin collecting data from the IMUs at 200Hz.

0.6.1 Data buffer

The "data buffer length (# samples)" slider adjusts the size of the data buffer. When each new sample is received, it is added to the data buffer. If the buffer was already full (the number of samples in the buffer was 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 size.

0.6.2 Trigger

Optionally, a trigger attached to the Arduino (Section 0.3.4) 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 you attempt to begin recording while the "use trigger" checkbox is checked and the trigger is "active", the recording end as soon as it starts 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 there is nothing setting the current on the Arduino pin associated with the trigger (i.e. if no trigger is connected), then the value of the pin is undefined. Thus, to ensure reliable behavior, the "use trigger" checkbox should not be checked unless there is a trigger connected to the Arduino.

0.7 Processing data

It is possible to apply data processing operations to the data buffer. Clicking the *process* button opens the data processing dialog.

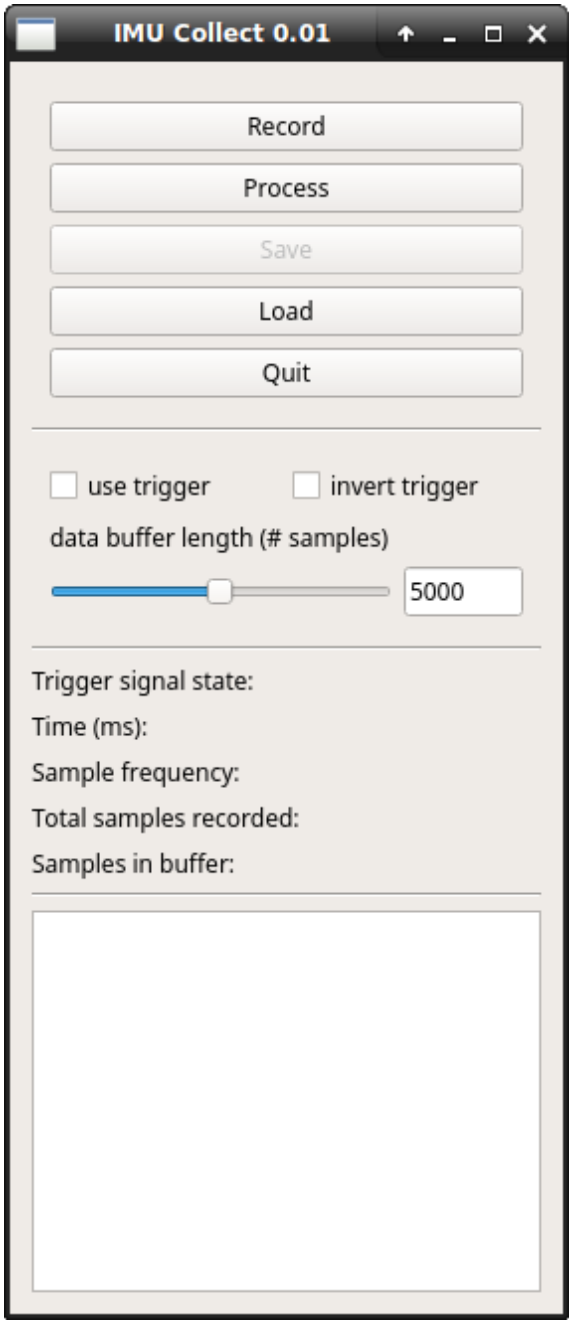

Figure 1: Control panel

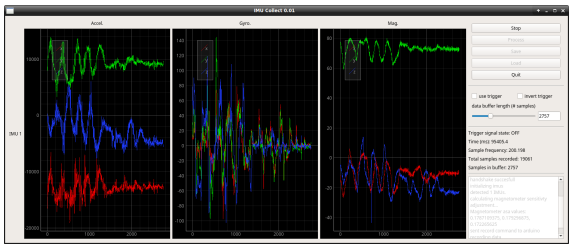

Figure 2: Collecting data from a single IMU

0.8 Saving and loading

0.8.1 Saving

When you press the *save* button, a dialog will appear allowing you to select a filesystem location, a filename, and a filetype (either .csv or .hdf5). The the data buffer is saved to file according to the selections made in the dialog.

0.8.2 .csv file format

PROGRAM_NAME uses the CSV module in the Python Standard Library with default formatting parameters: https://docs.python.org/3/library/csv.html

0.9 Settings

0.10 Troubleshooting

0.10.1 Error messages

invalid csv file

The program attempted to read a .csv file (Section 0.8.2), but the format of the data in the file was not valid.

invalid file type: ...

rx failed, no data read from serial

no Arduino found

failed to create connection

handshake failed

no connection, aborting

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 pins (Section 0.3.4). It then sends a message to the PC reporting the number of IMUs detected. This error is reported if the PC program sends a command to the Arduino to initialize, but does not receive a message reporting the number of IMUs detected. Make sure that any IMUs are correctly wired to the Arduino (Section 0.3.4) and that the correct code is installed on the Arduino (Section 0.4.1), and try resetting the Arduino.

no IMUs detected, aborting

ASA read failed, using 1 adjustment

Sample packet too short: ...