The Integration of Sensor Data with Raspberry Pi Microprocessor

Name: Yirui Wu

Data: 11/13/2015

Abstract

To implement various sensors on an integrated robotic system, a programming environment and valid communication methods that is capable of inputting digitized sensor data, and use those data in the control loop, is needed.

Keywords: I2C communication, UART communication, Adafruit Ultimate GPS Breakout, Sparkfun MPU-9150 Breakout, Raspberry Pi Setup

Introduction

Sensor data acquisition and association are critical parts for digital control system. Since an embedded digital controller is employed to process all the sensor data inputs, the communication interface, which makes those data readable and usable on the microprocessor, is the first priority at the programming level. An autonomous navigation system, for the team project, requires a continuous tracking on gps location and consistent measurement on course angle. Therefore, the group decides to use a gps module for sensing the current location (in latitude and longitude) of the robot and to use a magnetometer for measuring the course. Data acquired by both modules would be viewable on a remote user interface and can also be used in control software.

Objective

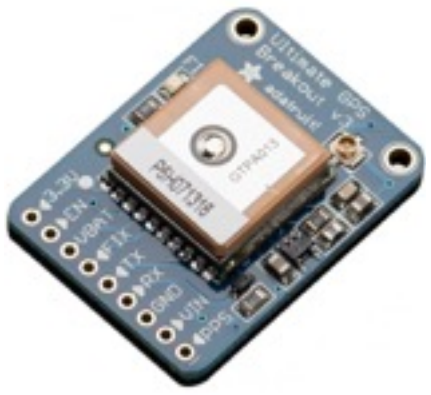
This is an advanced tutorial that assumes the user knows some basic manipulations in Raspbian operating system, for example, install a library or create a script file using Unix terminal in bash commands in the command line. The objective of this application note is to guide a method that user can read accelerometer data and gps data from different communication interface with the Raspberry Pi in real. In addition to the command line instructions, this tutorial also covers the details in wiring sensors to the Pi and the hardware configuration for the sensors.

Information

- Hardware
  - Sparkfun 9 Degrees of Freedom Breakout MPU9150

The inertial measurement sensor used for this project is the Invensense MPU-9150 with breakout board designed by sparkfun. The MPU-9150 is a multi-chip module consisting of two dies integrated into a single package. One die is MPU-6050 with a 3-axis gyroscope and a 3-axis accelerometer. The other die houses the AK8975 3-axis digital compass/magnetometer from Asahi Kasei Microdevices Corporation. This chip was designed for the low power, low cost, and high performance and currently being using in many places including smartphones and tablets.

  - Adafruit Ultimate GPS Breakout



The breakout is built around the third generation MTK3339-based module which has external antenna support and Pulse-Per-Second output. It has an 10 Hz update rate and can track up to 22 satellites on 66 channels. This gps module is built for a embedded system that offers a low power input (3.3 - 5V) and ENABLE pin can be used to turn off the module using any microcontroller pin. There is also a tiny red LED which is capable of indicating the signal status. The LED blinks at 1 Hz while it is searching for satellites and blinks once every 15 seconds when a fix is found. This gps only works at an open field when there is no building around. Whenever it acquires a FIX signal, it will have the course, longitude, latitude and height data updated at user defined frequency.

- Communication standards
  - I2C communication
    I2C stands for Inter-Integrated Circuit Bus. I2C uses microcontroller as a master and connecting with multiple slaves with unique addresses via a single-ended computer bus. It uses only two bidirectional open-drain lines: Serial Data Line (SDA), and a Serial Clock (SCL). SDA sets the transferred bit while SCL is low and the data is received when SCL is high.

  - UART communication
    UART stands for universal asynchronous receiver/transmitter. It transmits/receives data serially from a byte (5-8 bits) of data written/stored to a register. All the data will read at certain baud rate predefined by the programer. The UART transmission require three signal types: a Transmit Data (TxD), a Receive Data (RxD), and Signal Ground (SG). Since this kind of transmission is not bidirectional, two separate receive and transmit lines are needed.

Parts list

For hardware setup and programming on Raspberry Pi, the following parts are required:

- A Raspberry Pi
- A Power supply
- A Wifi dongle
- A Sparkfun MPU9150 Gyroscope and Accelerometer
- An Adafruit Ultimate GPS Breakout
- An USB to TTL Adapter cable (optional)
- A Breadboard
- M-F Jumper wires

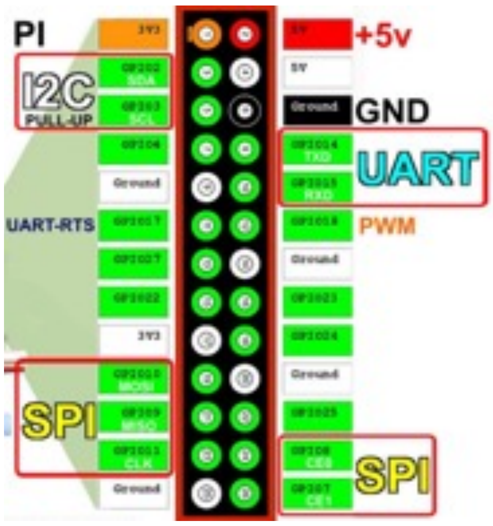
Assumptions

Before starting this tutorial, there are several assumptions that need to be made. Since this application notes will only focus on a particular facet of the whole project, the basic setup procedure for new-out-of-box Raspberry Pi and wireless network setup is going to be ignored here. The following are some underlying assumptions this application notes based on:

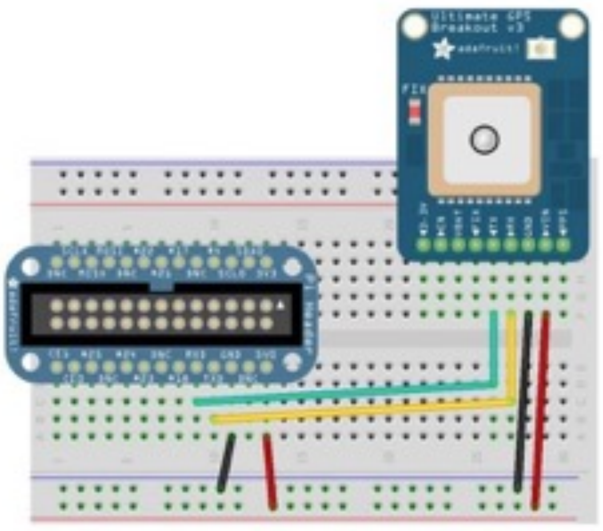
- a valid internet access
- a installed Raspian operating system and the boot behavior for the Pi has been set to desktop mode
- a mouse and a keyboard is connecting to the usb ports of the raspberry pi, and a monitor is connecting via HDMI adapter to ensure the graphic display.

Procedures:

1. Knowing the Raspberry Pi GPIO configuration for serial communication.
   The serial pins using in later part of this tutorial is GPIO 14&15 for UART, and GPIO 2&3 for I2C.

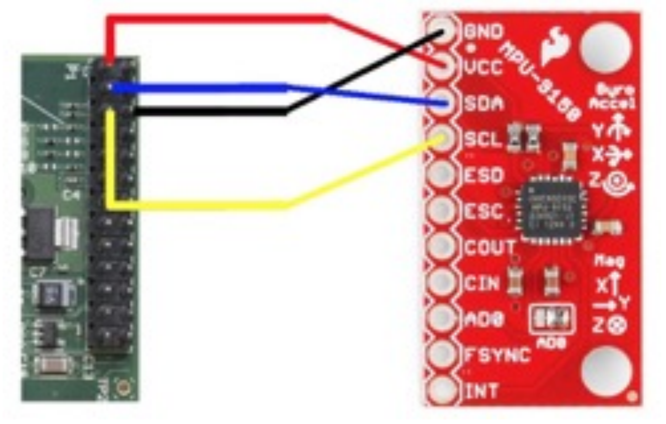


2. Wiring the Adafruit GPS to the Raspberry Pi
   Find the Tx/Rx, Vin and gnd pins on your adafruit gps module. Cross-connecting its Tx and Rx pin to Rx and Tx pins on Raspberry Pi. Then supply either 5V or 3.3V from the Pi to the Vin pin and short the GND pin with the Pi's ground.



3. Wiring the MPU9150 to the Raspberry Pi
   Connect the top four pins on MPU9150 breakout to the Raspberry Pi. Wire GPIO 2 to SDA (Data line) and GPIO 3 to SCL (Clock line). Supply 3.3V from

the Pi to the Vcc pin of the accelerometer and short the GND pin to the ground.



4. Preparing the Pi for I2C communication
    a. Open a terminal in Raspberry Pi and enter the following bash commands:

```
sudo apt-get update
sudo apt-get install i2c-tools libi2c-dev
```

    b. WiringPi is a very powerful C++ library designed for the RasPi that includes a wide variety of GPIO tools for the PI. To install WiringPi, enter the following commands:

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
sudo nano /etc/modules
```

    c. Add the following three lines to the bottom of the file (/etc/modules) if they aren't there already

```
snd-bcm2835
i2c-bcm2708
i2c-dev
```

    d. Next, modify the boot parameters to turn the i2c on at the start. Typing the bash command:

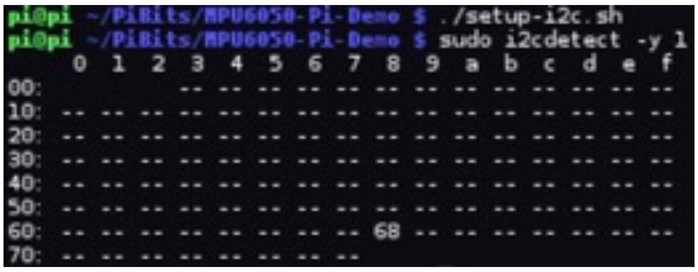
```
sudo nano /boot/config.txt
```

    e. Add the following lines to the bottom of this file:

```
dtparam=i2c1=on
dtparam=i2c_arm=on
```

f. Reboot your Raspberry Pi

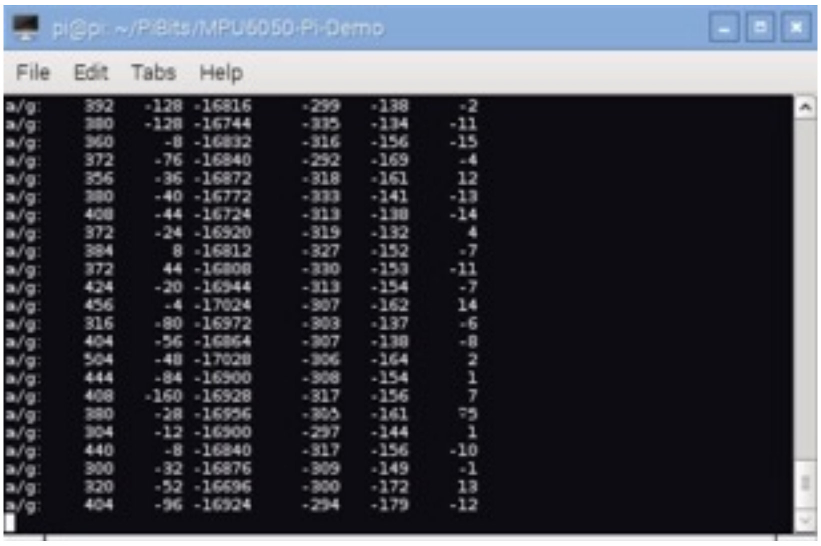
5. Installing and running the MPU-6050-Pi-Demo software
    a. Enter the following bash commands in a terminal window:
```
     git clone git://github.com/richardghirst/
PiBits.git
     cd PiBits/MPU6050-Pi-Demo
     sudo apt-get install libgtkmm-3.0-dev
```
    b. Some of the source files have to be edited to work with the Pi. We need to modify both I2Cdev.cpp and setup-i2c.sh files. Use the bash commands “nano” to edit the file.
    c. Change all the references to “/dev/i2c-0” to read “/dev/i2c-1” in this file and save.
    d. Enter the following bash commands to compile the source:
```
     make
     ./setup-i2c.sh
```
    e. Wait until the sources has finished compiling, then enter the following bash command:
```
     sudo i2cdetect -y 1
```
    f. You should see the following output



```
pi@pi ~/PiBits/MPU6050-Pi-Demo $ ./setup-i2c.sh
pi@pi ~/PiBits/MPU6050-Pi-Demo $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

    g. Now execute the example program “demo_raw” in this folder by typing the following bash command:
```
     ./demo_raw
```
    h. This demo will display raw gyro and accel values in terminal

```
a/g:  Ax   Ay  Az    Gx   Gy   Gz
```



```
a/g:     392  -128 -16816      -299   -138     -2
a/g:     380  -128 -16744      -335   -134    -11
a/g:     360    -8 -16832      -316   -156    -15
a/g:     372   -76 -16840      -292   -169     -4
a/g:     356   -36 -16872      -318   -161     12
a/g:     380   -40 -16772      -333   -141    -13
a/g:     408   -44 -16724      -313   -138    -14
a/g:     372   -24 -16920      -319   -132      4
a/g:     384     8 -16812      -327   -152     -7
a/g:     372    44 -16808      -330   -153    -11
a/g:     424   -20 -16944      -313   -154     -7
a/g:     456    -4 -17024      -307   -162     14
a/g:     316   -80 -16972      -303   -137     -6
a/g:     404   -56 -16864      -307   -138     -8
a/g:     504   -48 -17028      -306   -164      2
a/g:     444   -84 -16900      -308   -154      1
a/g:     408  -160 -16928      -317   -156      7
a/g:     380   -28 -16956      -305   -161     -5
a/g:     304   -12 -16900      -297   -144      1
a/g:     440    -8 -16840      -317   -156    -10
a/g:     300   -32 -16876      -309   -149     -1
a/g:     320   -52 -16696      -300   -172     13
a/g:     404   -96 -16924      -294   -179    -12
```

6. In order to obtain the data from Adafruit GPS module, Adafruit have their own self-developed software named GPS Daemon (gpsd). The tutorial is available at the link: https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi
7. Since gpsd is not an open-source software, it is better to use a third party software that allows users to modify and create their own file to read from gps. “libgps” is an open-source gps library created for mainly use on Raspberry ARM boards, and been tested with Adafruit Ultimate GPS Breakout. We are using this library in our project.
8. Installing and compiling the libgps package.
    a. To install libgps, typing the following commands:

       ```
       git clone git://github.com/wdalmut/libgps.git
       cd libgps
       ```

    b. After directing to the libgps folder, build the files to obtain libgps.a by typing:

       ```
       make
       sudo make install
       ```

9. You can find a sample code in the example folder called “position_logger.c” to test the connection
    a. Compile it with

       ```
       gcc -o position_logger position_logger.c -lgps
       -lm
       ```

    b. Run it with

       ```
       $ ./position_logger
       ```

c. When there is a valid gps signal (the fix LED is not blinking at 1 Hz), you should see the decimal degrees for latitudes and longitudes directly in the console, as the following:

```
45.071060 7.646363
45.071082 7.646385
45.071078 7.646387
45.071060 7.646373
45.071048 7.646358
45.071052 7.646372
45.071057 7.646392
45.071062 7.646397
45.071062 7.646383
45.071073 7.646395
45.071082 7.646403
```

Conclusion

This application note took two sensors as an example and guided the user how to setup serial communication between the sensors and the Pi from the first step. It also briefly explained the command line method to edit, compile and run a file. However, to write those sensor data into an algorithm needs much more efforts than just viewing them. The ultimate goal, or in other word, the more advanced derivation of this application note should be addressing different sensor libraries from a single script file. This requires a deeper understanding of the coding principles behind each sensor software package.

Reference:

1. Kevin Townsend. Adafruit Ultimate GPS on the Raspberry Pi, 15 July 2014. Web. https://learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps-on-the-raspberry-pi.pdf
2. Walter Dal Mut. UART NMEA GPS library for Raspberry Pi. 08 Sept. 2014. Web. https://github.com/wdalmut/libgps
3. InvenSense. MPU-9150 Datasheet, 18 Sept. 2013. Web. http://www.invensense.com/mems/gyro/documents/PS-MPU-9150A-00v4_3.pdf