

ELEC230 Semester 2 – Laboratory 1

More Sensor Interfacing

Introduction

Broadly speaking, any robot needs sensors, actuators and a 'brain' to interpret and set actions based on the available information. This week you will be interfacing the second of two sensor units to your Raspberry Pi Zero, which is required to finish Assignment 2. This second sensor is an inertial measurement unit (IMU): a combined accelerometer, rate gyro and magnetometer. The first two are part of the essential definition of an IMU and measure the force and angular rate in three axes X, Y and Z; the magnetometer is an optional extra sometimes included in an IMU, and provides a measure of its orientation using magnetic force.

Note about the interface

Whilst both sensor units interface via the I2C bus, there is a slight issue with the pull-up resistors used on the I2C bus in that the Raspberry Pi Zero W has pull-up resistors of 1.8 k Ω and the accelerometer unit has pull-up resistors of 4.7 k Ω to give a combined resistance of 1.3 k Ω . While the value of 1.3 k Ω is within specification (anything between 1.0 k Ω and 10 k Ω should be acceptable) prior tests have shown that it does not work reliably using the cables provided, therefore another I2C bus will be configured for this work.

More about IMUs

You can find more information out about IMUs in [Lecture 11a \(Part 2\)](#) in particular. (Make sure you have watched the other Week 9-11 Sensors lectures on Canvas for broader principles and overview, e.g. the relevance of sensors for mobile robots.) You can also find out more in one of the recommended e-books for sensors: [Siegwart et al \(2011\)](#); see in particular Sections 4.1.5 to 4.1.7.

Note about Assignment 2

Assignment 2 is mainly based around this lab work, plus the lab work (and work at home) you did to interface and experiment with the VL53L1X laser rangefinder. (There are also some supplementary questions to research and answer in the assignment script.) Part of the requirement of the assignment is that a demonstrator has 'signed off' your lab work i.e. you are able to show to them that you have successfully interfaced both the laser rangefinder and the IMU with your Pi-Zero, with all four measurement functions operating simultaneously.

You will probably want to experiment with your Pi-Zero and sensors at home while writing up your Report. Don't forget that the folder on Canvas recently retitled to '**Assignment 2 Initial Guidance – plus Assignment 2 script**' has useful information which will help you with aspects of this.

Interfacing the Grove - IMU 9DOF (lcm20600+AK09918)

You are going to interface the Seed Grove IMU 9DOF sensor to your Raspberry Pi. You can find details of this sensor by following this link:

http://wiki.seeedstudio.com/Grove-IMU_9DOF-lcm20600%2BAK09918/

As indicated above there are problems connecting this to the same I2C bus used for the VL53L1X sensor therefore the next stage is to enable a second I2C bus on the Raspberry Pi Zero W.

Enable the I2C-3 bus

The “config.txt” file stored in the “/boot” directory of your SD Card contains information that the “bootloader” uses to configure the Raspberry Pi Zero.

To enable a second I2C bus add the lines shown in Figure 11 to your “/boot/config.txt” file. You can do this using your raspberry pi editor of choice e.g. “gedit”, “nano” or “geany”. However as this is a system file you need to use the “sudo” command to give the editor permission to modify the “/boot/config.txt” file. For example you could “sudo nano /boot/config.txt”

```
# create second i2c bus
dtoverlay=i2c-gpio,i2c_gpio_sda=5,i2c_gpio_scl=6,bus=3
```

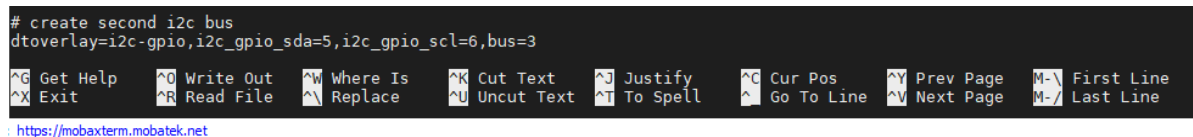


Figure 11 - /boot/config.txt

The line “dtoverlay=i2c-gpio,i2c_gpio_sda=5,i2c_gpio_scl=6,bus=3” creates a new I2C bus (Bus 3) that uses GPIO Connection 5 (Pin 29) as the SDA signal and GPIO Connection 6 (Pin 31) as the SCL signal. Once you have modified your “/boot/config.txt” reboot your Pi and examine the “/dev” directory to check that your new I2C bus has been created as shown in Figure 12.

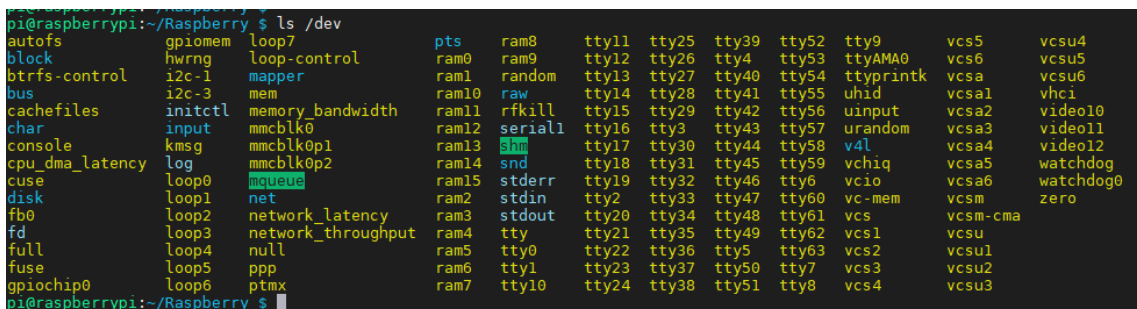


Figure 12 - Contents of /dev showing the i2c-3 bus

Connecting the Grove IMU 9DOF

The Grove IMU 9DOF sensor is connected via the provided cable, the female to female connectors and some short lengths of stripped solid core cable. You need four approx. 15mm long lengths of stripped solid core cable that you insert into the white Grove sensor connector and the female to female connectors. The other end of the female to female connector is connected to the Raspberry Pi zero. You need to connect the Grove black GND line to a GND on the Pi, for example Pin 25. The red VCC line to the other 3.3V connector the Pi, Pin 17. The white SDA line to your I2C-3 SDA (pin 29) and the yellow SCL line to your I2C-3 SDA (pin 31). Once connect power up your board and probe your i2c-3 bus using the command:

```
i2cdetect -y 3
```

You should now see the two sensors connected on the i2c-3 bus as shown in Figure 13. On the IMU board are two ICs an ICM20600 (3 axis accelerometer and rate gyro) at address 0x69 and an AK09918C (3 axis magnetometer) at location 0x0C.

```
pi@raspberrypi:~/Raspberry $
pi@raspberrypi:~/Raspberry $ i2cdetect -y 3
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- 0c -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- 69 -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~/Raspberry $
```

:: <https://mobaxterm.mobatek.net>

Figure 13 - Result of "i2cdetect -y 3"

Testing the Grove IMU 9DOF

Currently there is not a lot of software available for these sensors that run on the Raspberry Pi. However on this page

<https://github.com/turmary/bmi088-python>

there are some C source files that can access the sensors. Download the master branch and copy the "src" directory to you raspberry pi.

On your Raspberry pi "cd" into the "src" directory. Before you compile the examples you need to modify the programs so that they use bus "i2c-3" rather than "i2c-1". To start with modify "test_icm20600.c" so that it accesses "/dev/i2c-3" as shown in Figure 14.

```
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
* THE SOFTWARE.
*/
#include <stdio.h>
#include <unistd.h>
#include "rpi_icm20600.h"
#include "rpi_i2c.h"

int main(int argc, char* argv[]) {
    rpi_icm20600_t icm[1];
    double x, y, z;
    int id;

    icm20600_cfg_t config[1] = {
        {
            RANGE_2K_DPS,
            GYRO_RATE_1K_BW_176,
            GYRO_AVERAGE_1,
            RANGE_16G,
            ACC_RATE_1K_BW_420,
            ACC_AVERAGE_4,
            ICM_6AXIS_LOW_POWER,
            0
        }
    };

    id = rpi_icm20600_init(
        icm,
        "/dev/i2c-3",
        ICM20600_I2C_ADDR1,
        config
    );

    printf("Device ID: 0x%02X\n", id);
}
```

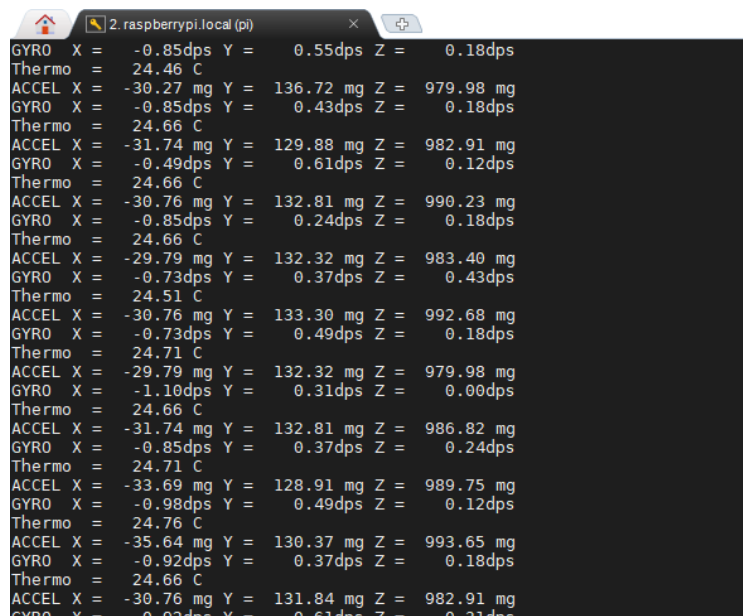
Get Help Write Out Where Is Cut Text Justify Cur Pos Prev Page First Line
Exit Read File Replace Uncut Text To Spell Go To Line Next Page Last Line

Figure 14 - Modified test_icm20600.c

You can compile the “icm20600” example with the command

```
gcc rpi_i2c.c rpi_icm20600.c test_icm20600.c -o test_icm20600
```

and execute it with the “./test_icm20600” command. You should get output similar to that shown in Figure 15. Change the orientation of the sensor so that is flat on the desk. You should see the “Z” axis acceleration close to 1g with the X and Y accelerations close to zero.



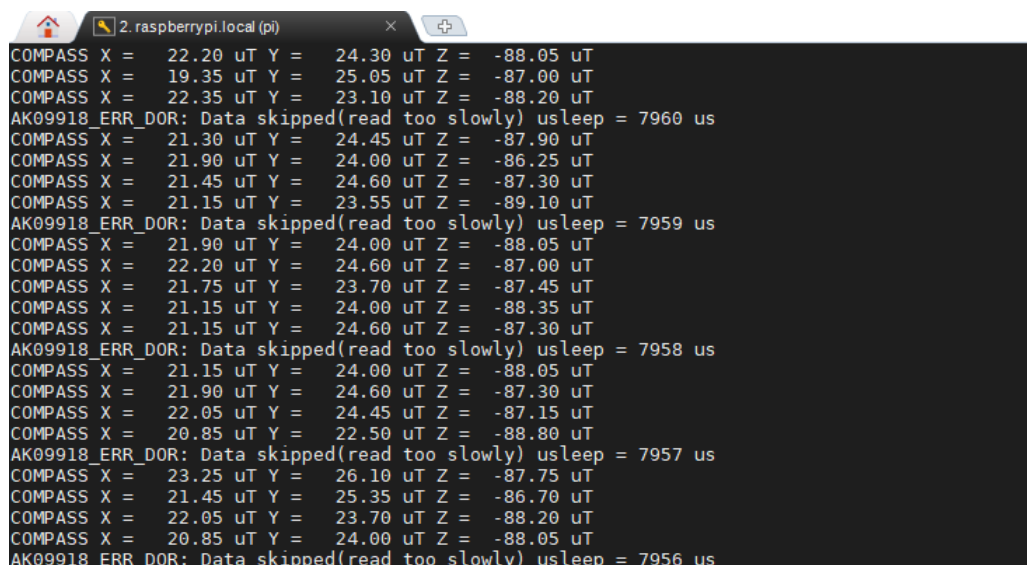
```
2.raspberrypi.local (pi) X
GYRO X = -0.85dps Y = 0.55dps Z = 0.18dps
Thermo = 24.46 C
ACCEL X = -30.27 mg Y = 136.72 mg Z = 979.98 mg
GYRO X = -0.85dps Y = 0.43dps Z = 0.18dps
Thermo = 24.66 C
ACCEL X = -31.74 mg Y = 129.88 mg Z = 982.91 mg
GYRO X = -0.49dps Y = 0.61dps Z = 0.12dps
Thermo = 24.66 C
ACCEL X = -30.76 mg Y = 132.81 mg Z = 990.23 mg
GYRO X = -0.85dps Y = 0.24dps Z = 0.18dps
Thermo = 24.66 C
ACCEL X = -29.79 mg Y = 132.32 mg Z = 983.40 mg
GYRO X = -0.73dps Y = 0.37dps Z = 0.43dps
Thermo = 24.51 C
ACCEL X = -30.76 mg Y = 133.30 mg Z = 992.68 mg
GYRO X = -0.73dps Y = 0.49dps Z = 0.18dps
Thermo = 24.71 C
ACCEL X = -29.79 mg Y = 132.32 mg Z = 979.98 mg
GYRO X = -1.10dps Y = 0.31dps Z = 0.00dps
Thermo = 24.66 C
ACCEL X = -31.74 mg Y = 132.81 mg Z = 986.82 mg
GYRO X = -0.85dps Y = 0.37dps Z = 0.24dps
Thermo = 24.71 C
ACCEL X = -33.69 mg Y = 128.91 mg Z = 989.75 mg
GYRO X = -0.98dps Y = 0.49dps Z = 0.12dps
Thermo = 24.76 C
ACCEL X = -35.64 mg Y = 130.37 mg Z = 993.65 mg
GYRO X = -0.92dps Y = 0.37dps Z = 0.18dps
Thermo = 24.66 C
ACCEL X = -30.76 mg Y = 131.84 mg Z = 982.91 mg
GYRO X = -0.92dps Y = 0.61dps Z = 0.21dps
```

Figure 15- test_icm20600 output

Now modify the “test_ak09918.c” file so that it also uses the “/dev/i2c-3” bus. Compile the test application with the command

```
gcc rpi_i2c.c rpi_ak09918.c test_ak09918.c -o test_ak09918
```

and execute the program with the command “./testak09918”. You should see output similar to that shown in Figure 16. Rotate the sensor round to maximise the Compass X output. Is it pointing North?



```
2.raspberrypi.local (pi) X
COMPASS X = 22.20 uT Y = 24.30 uT Z = -88.05 uT
COMPASS X = 19.35 uT Y = 25.05 uT Z = -87.00 uT
COMPASS X = 22.35 uT Y = 23.10 uT Z = -88.20 uT
AK09918_ERR_DOR: Data skipped(read too slowly) usleep = 7960 us
COMPASS X = 21.30 uT Y = 24.45 uT Z = -87.90 uT
COMPASS X = 21.90 uT Y = 24.00 uT Z = -86.25 uT
COMPASS X = 21.45 uT Y = 24.60 uT Z = -87.30 uT
COMPASS X = 21.15 uT Y = 23.55 uT Z = -89.10 uT
AK09918_ERR_DOR: Data skipped(read too slowly) usleep = 7959 us
COMPASS X = 21.90 uT Y = 24.00 uT Z = -88.05 uT
COMPASS X = 22.20 uT Y = 24.60 uT Z = -87.00 uT
COMPASS X = 21.75 uT Y = 23.70 uT Z = -87.45 uT
COMPASS X = 21.15 uT Y = 24.00 uT Z = -88.35 uT
COMPASS X = 21.15 uT Y = 24.60 uT Z = -87.30 uT
AK09918_ERR_DOR: Data skipped(read too slowly) usleep = 7958 us
COMPASS X = 21.15 uT Y = 24.00 uT Z = -88.05 uT
COMPASS X = 21.90 uT Y = 24.60 uT Z = -87.30 uT
COMPASS X = 22.05 uT Y = 24.45 uT Z = -87.15 uT
COMPASS X = 20.85 uT Y = 22.50 uT Z = -88.80 uT
AK09918_ERR_DOR: Data skipped(read too slowly) usleep = 7957 us
COMPASS X = 23.25 uT Y = 26.10 uT Z = -87.75 uT
COMPASS X = 21.45 uT Y = 25.35 uT Z = -86.70 uT
COMPASS X = 22.05 uT Y = 23.70 uT Z = -88.20 uT
COMPASS X = 20.85 uT Y = 24.00 uT Z = -88.05 uT
AK09918_ERR_DOR: Data skipped(read too slowly) usleep = 7956 us
```

Figure 16 - test_ak09918 output