CSCI502 Project 1

Azat Utessov
Alar Akilbekov
Bauyrzhan Askarov
Aidarkhan Toktargazin
(All team members contributed across Tasks 1-3)

Task 1 Raspberry Pi Configuration

1. Safety rules:

- `sudo shutdown now -h`
- don't place on metal surfaces
- connect a HDMI before powering up the board

2. Install Raspberry Pi OS (32-bit) using RPi Imager program

We uploaded Raspberry Pi OS using imager on SD-card. We installed SD-card into Raspberry Pi.

3. Connect Raspberry to monitor, mouse and keyboard

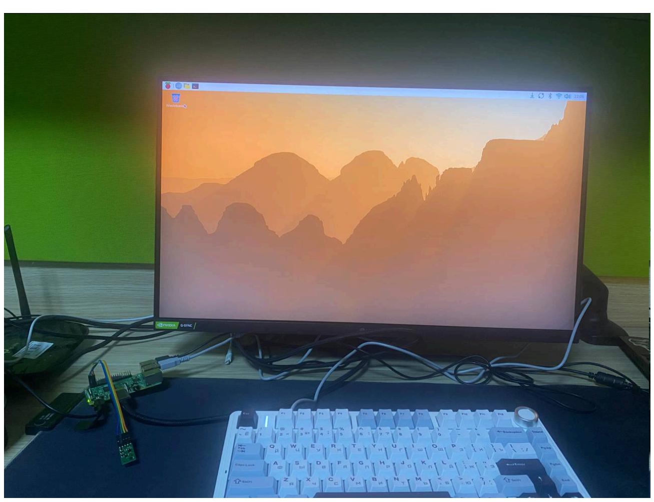


Figure 1. Monitor, mouse and keyboard

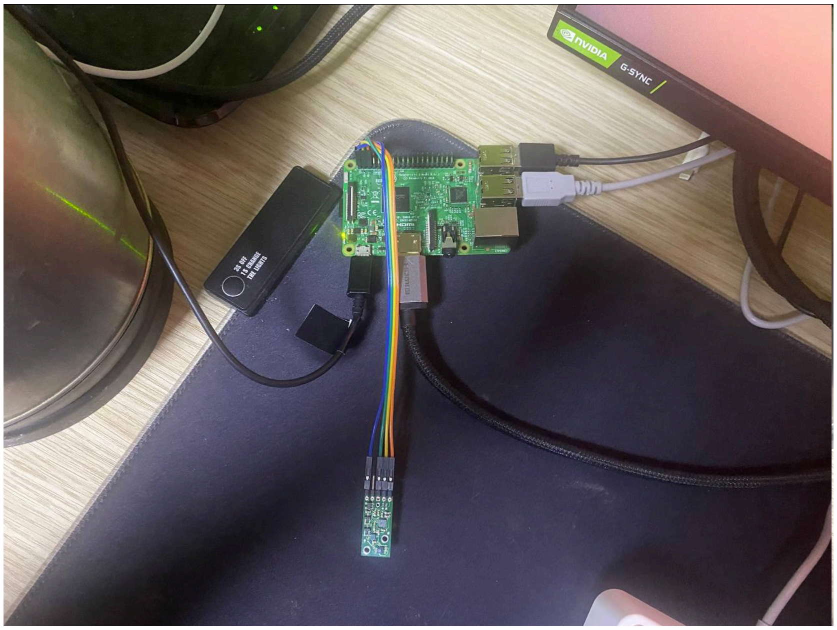


Figure 2. Raspberry Pi connected via HDMI and 2 USB cables

4. Boot the board

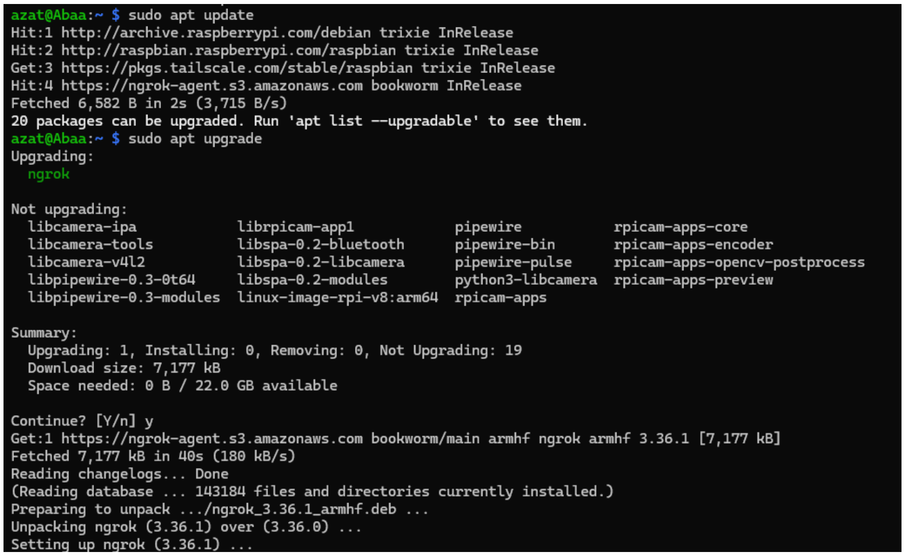
We started the Raspberry Pi and booted the board. The board was connected to Wi-Fi.

```
azat@Abaa:~ $ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether b8:27:eb:94:04:ee brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether b8:27:eb:c1:51:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.179/24 brd 192.168.0.255 scope global dynamic noprefixroute wlan0
       valid_lft 6959sec preferred_lft 6959sec
    inet6 fe80::ab6f:1261:31a7:e7/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Figure 2. Raspberry Pi connected to Wi-Fi (It has IP)

5. Update RPI OS

We updated RPI OS using update and upgrade commands in terminal.

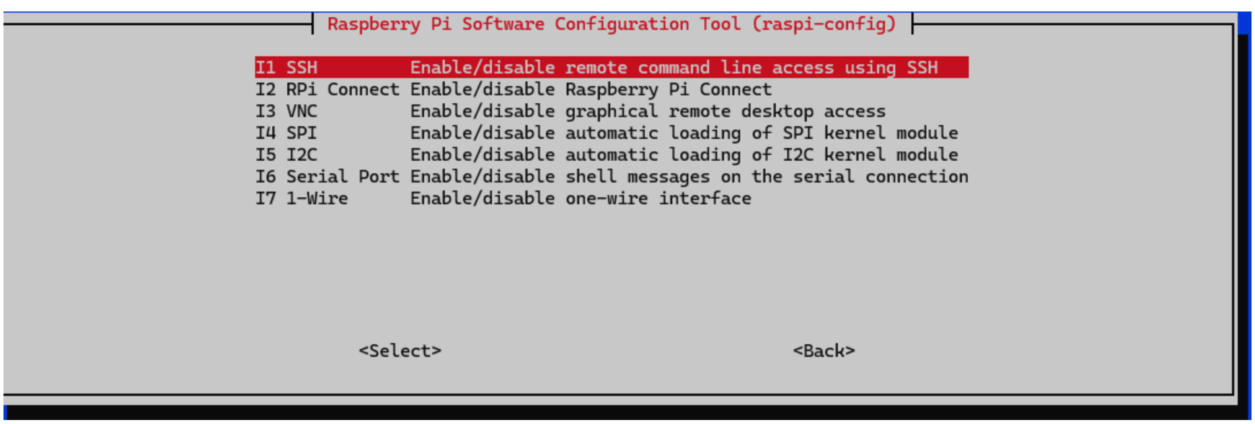


```
azat@Abaa:~ $ sudo apt update
Hit:1 http://archive.raspberrypi.com/debian trixie InRelease
Hit:2 http://raspbian.raspberrypi.com/raspbian trixie InRelease
Get:3 https://pkgs.tailscale.com/stable/raspbian trixie InRelease
Hit:4 https://ngrok-agent.s3.amazonaws.com bookworm InRelease
Fetched 6,582 B in 2s (3,715 B/s)
20 packages can be upgraded. Run 'apt list --upgradable' to see them.
azat@Abaa:~ $ sudo apt upgrade
Upgrading:
  ngrok

Not upgrading:
  libcamera-ipa           librpicam-app1           pipewire          rpicam-apps-core
  libcamera-tools         libspa-0.2-bluetooth     pipewire-bin      rpicam-apps-encoder
  libcamera-v4l2          libspa-0.2-libcamera     pipewire-pulse    rpicam-apps-opencv-postprocess
  libpipewire-0.3-0t64    libspa-0.2-modules       python3-libcamera rpicam-apps-preview
  libpipewire-0.3-modules linux-image-rpi-v8:arm64 rpicam-apps

Summary:
  Upgrading: 1, Installing: 0, Removing: 0, Not Upgrading: 19
  Download size: 7,177 kB
  Space needed: 0 B / 22.0 GB available

Continue? [Y/n] y
Get:1 https://ngrok-agent.s3.amazonaws.com bookworm/main armhf ngrok armhf 3.36.1 [7,177 kB]
Fetched 7,177 kB in 40s (180 kB/s)
Reading changelogs... Done
(Reading database ... 143184 files and directories currently installed.)
Preparing to unpack .../ngrok_3.36.1_armhf.deb ...
Unpacking ngrok (3.36.1) over (3.36.0) ...
Setting up ngrok (3.36.1) ...
```

Figure 3. Update of the board

6. Configuring the board

We configured the board through sudo raspi-confi. We enabled SSH and I2C connections, and Expanded File Systems.



```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 SSH         Enable/disable remote command line access using SSH
I2 RPi Connect Enable/disable Raspberry Pi Connect
I3 VNC         Enable/disable graphical remote desktop access
I4 SPI         Enable/disable automatic loading of SPI kernel module
I5 I2C         Enable/disable automatic loading of I2C kernel module
I6 Serial Port Enable/disable shell messages on the serial connection
I7 1-Wire      Enable/disable one-wire interface

<Select>                    <Back>
```

Figure 4. Enabling SSH and I2C

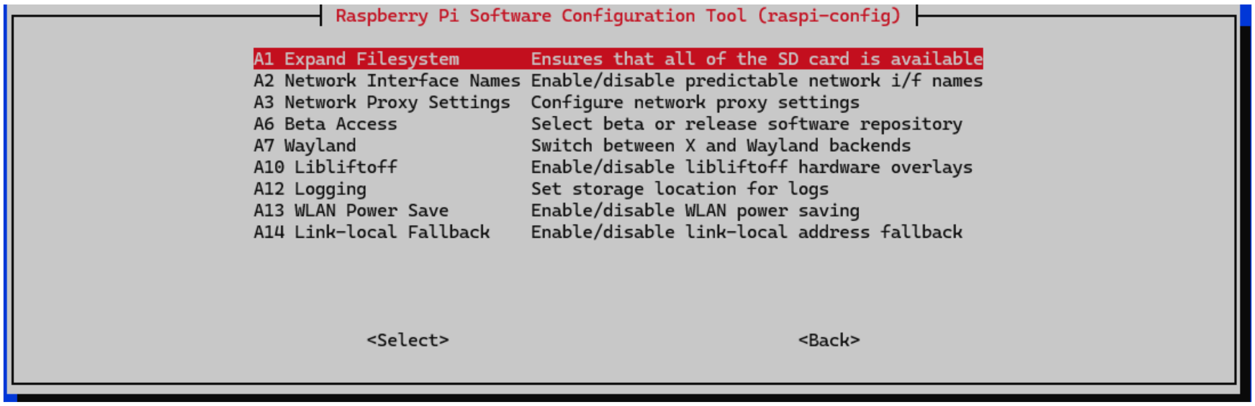


Figure 5. Expanding Filesystem

7. Establishing SSH connection

We used 2 methods to connect to board: SSH (for local network connection) and ngrok (for internet connection) to the board.

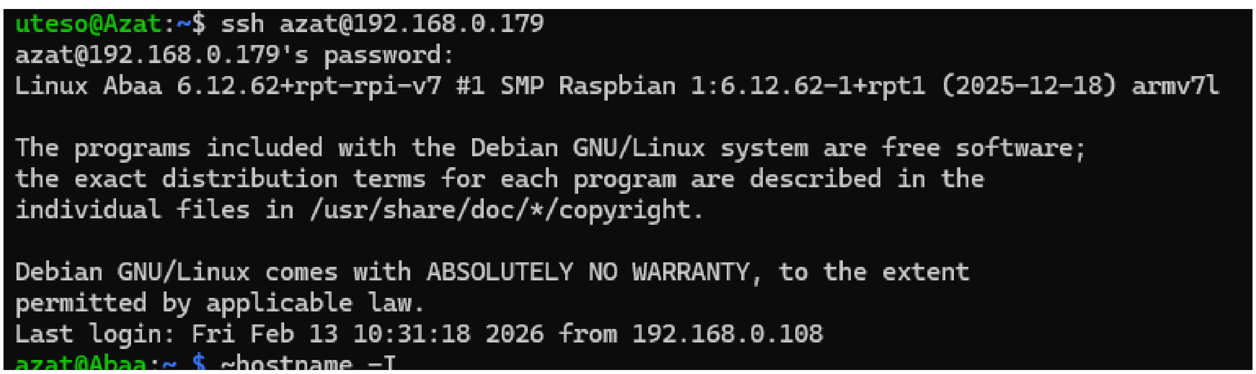


Figure 6. Connecting via SSH

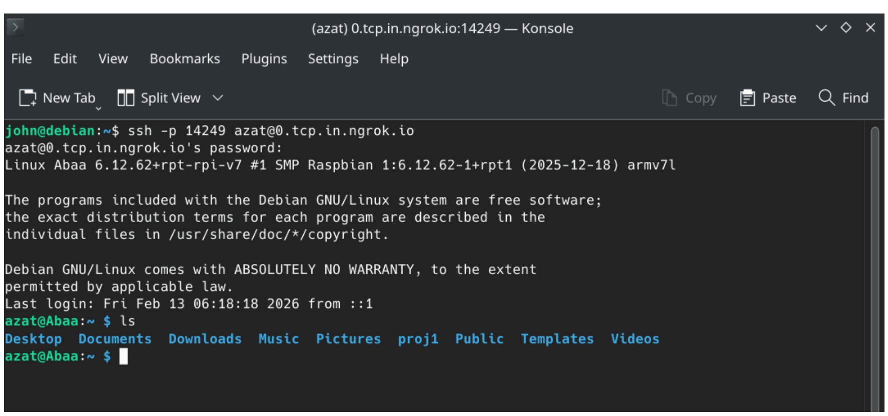


Figure 7. Connecting via ngrok

Task 2 Github Repository

GitHub Classroom, ALARIS-NU
https://github.com/ALARIS-NU/project-1-team_baaa

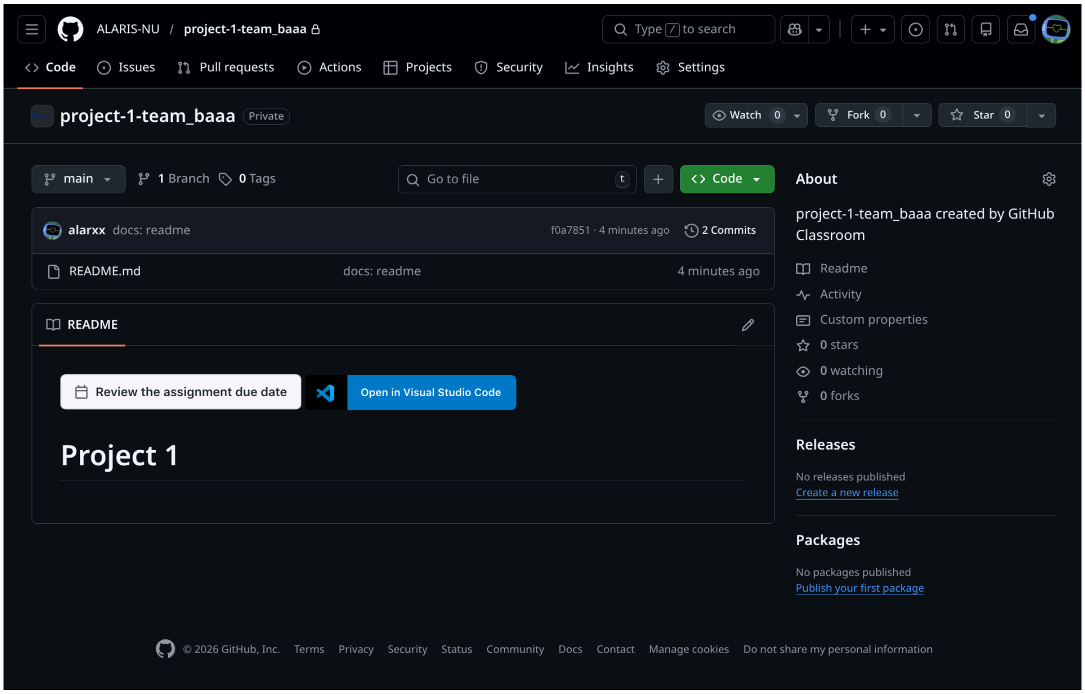


Figure 8. Github Repository

We created repository for the project. You can access and download it through:
$ git clone https://github.com/ALARIS-NU/project-1-team_baaa

Task 3 IMU Sensor Interfacing

1. Verify the I2C connected

We verified the I2C connection is enabled. Here are the buses available for the board.

Figure 9. I2C available buses

2. GPIO study

Pin 3 → SDA
Pin 5 → SCL

GPIO2 → SDA
GPIO3 → SCL

In this project, physical pin numbering was used for hardware connection

3. Study I2C connection

The I²C protocol was used because it allows multiple devices to communicate using only two wires (SDA and SCL). The Raspberry Pi acts as the master device, and the IMU sensor chips act as slave devices with unique addresses (0x6B and 0x1E).

4. Test i2cdetect and i2cdump

We have mentioned i2cdetected above. Here is i2cdump for LSM6DS33 (Gyro + Accelerometer)

```
azat@Abaa:~ $ i2cdump -y 1 0x6B b
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 69    ...............i
10: 00 00 04 00 00 00 00 00 38 38 00 00 00 00 00 bb    ..?.....88.....?
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
30: 00 00 00 00 00 00 00 00 00 00 00 10 00 00 ff ff    ...........?....
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 ff 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

Figure 10. I2cdump output for Gyro and Accelerometer sensor

5. Checking if the sensors are on or off

The sensors were off by default. We checked the control registers of the sensors using i2cget. For example, for the Gyro sensor the control register is 0x11. We can learn it from the datasheet LSM6DS33

CTRL2_G (11h)

Angular rate sensor control register 2 (r/w).

Table 46. CTRL2_G register

ODR_G3	ODR_G2	ODR_G1	ODR_G0	FS_G1	FS_G0	FS_125	0(1)

1. This bit must be set to '0' for the correct operation of the device.

Table 47. CTRL2_G register description

ODR_G [3:0]	Gyroscope output data rate selection. Default value: 0000 (Refer to *Table 46*)
FS_G [1:0]	Gyroscope full-scale selection. Default value: 00 (00: 245 dps; 01: 500 dps; 10: 1000 dps; 11: 2000 dps)
FS_125	Gyroscope full-scale at 125 dps. Default value: 0 (0: disabled; 1: enabled)

Figure 11. Control register specifications of the Gyro sensor

As we can see here it is currently set to 0x00, which corresponds to off mode. The same is true for the accelerometer and for the magnetometer sensors.

```
azat@Abaa:~ $ i2cget -y 1 0x6b 0x11
0x00
```

Figure 12. Gyro sensor is off

The value 1 in the command: i2cget -y 1 0x6B 0x11 refers to the I²C bus number.

6. Turning the sensors on

To turn the sensors on we used i2cset command.

```
azat@Abaa:~ $ i2cset -y 1 0x6b 0x11 0x24
```

Figure 13. Setting Gyro to 26Hz and ±500 deg/sec

Why we set it to 0x24?

ODR_G3	ODR_G2	ODR_G1	ODR_G0	ODR [Hz] when G_HM_MODE = 1	ODR [Hz] when G_HM_MODE = 0
0	0	0	0	Power down	Power down
0	0	0	1	13 Hz (low power)	13 Hz (high performance)
0	0	1	0	26 Hz (low power)	26 Hz (high performance)
0	0	1	1	52 Hz (low power)	52 Hz (high performance)
0	1	0	0	104 Hz (normal mode)	104 Hz (high performance)

Figure 14. Gyro ODR configuration table

Using abovementioned table we can see that for 26Hz we have to input 0010 in the big endian part of the register, which corresponds to 0x2.

Table 47. CTRL2_G register description

ODR_G [3:0]	Gyroscope output data rate selection. Default value: 0000 (Refer to *Table 46*)
FS_G [1:0]	Gyroscope full-scale selection. Default value: 00 (00: 245 dps; 01: 500 dps; 10: 1000 dps; 11: 2000 dps)
FS_125	Gyroscope full-scale at 125 dps. Default value: 0 (0: disabled; 1: enabled)

Figure 15. Gyro register description

Using this table we can see that small endiad part for 500dps is 0100, which corresponds to 0x4. In the end, connecting these two parts we get 0010 0100, or 0x24. So that is why we set control register of gyroscope to 0x24.

7. Configuration registers

LSM6DS33		LIS3MDL	
Register (address)	Control word	Register (adress)	Control word
CTRL1_XL (0x10)	0x20	CTRL_REG1 (0x20)	0x6C
CTRL2_G (0x11)	0x24	CTRL_REG2 (0x21)	0x00
		CTRL_REG3 (0x22)	0x00

Table 1. Register adresses and control words

We have described above the gyroscope control. Regarding accelerometer, 0x20 means 26 Hz output data rate, ±2g full-scale.

Regarding magnetometer, CTRL_REG1 is set to 0x6C, which is 5 Hz output data rate. CTRL_REG2 is 0x00, which means ±4 gauss full-scale, CTRL_REG3 is set to 0x00, which corresponds to continuous-conversion mode.

9.11 CTRL1_XL (10h)

Linear acceleration sensor control register 1 (r/w).

Table 42. CTRL1_XL register

ODR_XL3	ODR_XL2	ODR_XL1	ODR_XL0	FS_XL1	FS_XL0	BW_XL1	BW_XL0

Table 43. CTRL1_XL register description

ODR_XL [3:0]	Output data rate and power mode selection. Default value: 0000 (see *Table 44*).
FS_XL [1:0]	Accelerometer full-scale selection. Default value: 00. (00: ±2 *g*; 01: ±16 *g*; 10: ±4 *g*; 11: ±8 *g*)
BW_XL [1:0]	Anti-aliasing filter bandwidth selection. Default value: 00 (00: 400 Hz; 01: 200 Hz; 10: 100 Hz; 11: 50 Hz)

Figure 16. Control register for accelerometer

CTRL_REG1 (20h)

Table 18. CTRL_REG1 register

TEMP_EN	OM1	OM0	DO2	DO1	DO0	FAST_ODR	ST

Table 19. CTRL_REG1 description

TEMP_EN	Temperature sensor enable. Default value: 0 (0: temperature sensor disabled; 1: temperature sensor enabled)
OM[1:0]	X and Y axes operative mode selection. Default value: 00 (Refer to *Table 21*)
DO[2:0]	Output data rate selection. Default value: 100 (Refer to *Table 22*)
FAST_ODR	FAST_ODR enables data rates higher than 80 Hz (refer to *Table 20*). Default value: 0 (0: Fast_ODR disabled; 1: FAST_ODR enabled)
ST	Self-test enable. Default value: 0 (0: self-test disabled; 1: self-test enabled)

Figure 17. Control register for magnetometer rate

7.3 CTRL_REG2 (21h)

Table 23. CTRL_REG2 register

0[(1)]	FS1	FS0	0[(1)]	REBOOT	SOFT_RST	0[(1)]	0[(1)]

1. These bits must be set to '0' for correct functioning of the device

Table 24. CTRL_REG2 description

FS[1:0]	Full-scale configuration. Default value: 00 Refer to *Table 25*
REBOOT	Reboot memory content. Default value: 0 (0: normal mode; 1: reboot memory content)
SOFT_RST	Configuration registers and user register reset function. (0: Default value; 1: Reset operation)

Figure 18. Control register for magnetometer deviation

7.4 CTRL_REG3 (22h)

Table 26. CTRL_REG3 register

0[(1)]	0[(1)]	LP	0[(1)]	0[(1)]	SIM	MD1	MD0

1. These bits must be set to '0' for correct functioning of the device

Table 27. CTRL_REG3 description

LP	Low-power mode configuration. Default value: 0 If this bit is '1', DO[2:0] is set to 0.625 Hz and the system performs, for each channel, the minimum number of averages. Once the bit is set to '0', the magnetic data rate is configured by the DO bits in *CTRL_REG1 (20h)* register.
SIM	SPI serial interface mode selection. Default value: 0 (0: 4-wire interface; 1: 3-wire interface).
MD[1:0]	Operating mode selection. Default value: 11 Refer to *Table 28*.

Figure 19. Control register for magnetometer operating mode

Addtional work

We have written and uploaded code that will take information from the IMU sensor and will read the data in real time.

```
azat@Abaa:~/proj1 $ ls
lazat@Abaa:~/proj1 $ vim code.cpp
azat@Abaa:~/proj1 $ vim code.cpp
azat@Abaa:~/proj1 $ g++ code.cpp -o a
azat@Abaa:~/proj1 $ ./a
LSM6DS33 WHO_AM_I = 0x69
LIS3MDL  WHO_AM_I = 0x3d

Configured sensors. Streaming raw data...
Press Ctrl+C to stop.

  Gx      Gy      Gz  |  Ax      Ay      Az  |  Mx      My      Mz
-----------------------------------------------------------------
   229   -306     -87 | -15598    108    4654 |   2469  -1216  -11802
```

Figure 20. Example of the code running

You can acces the code via GitHub by running the following commands:

G++ code.cpp -o a

./a

We used these output registers from the datasheets:

Figure 21. Output registers for accelerometer and gyroscope

Figure 22. Output registers for magnetometer