

Student name: Yasmin Pokia

Student ID: s222245206

SIT225: Data Capture Technologies

Activity 2.1: Working with sensor - DHT22

DHT22 is a temperature and humidity sensor.

Hardware Required

Arduino Board

DHT22 sensor

USB cable

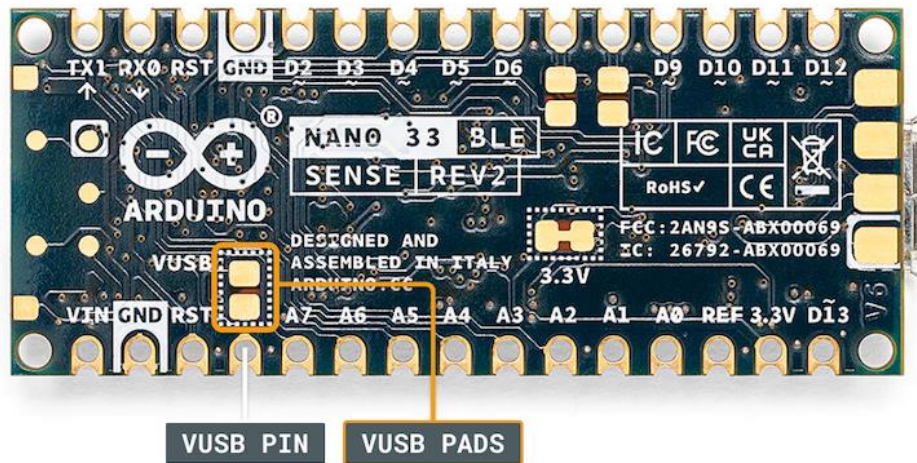
Software Required

Arduino programming environment

Known issue, action required

Arduino Nano 33 IoT board operates on 3.3 V, it needs to be arranged to make it 5 V. The Arduino board has a pin called VUSB or VBUS and there are 2 pads next to the pin. **These two pads must be shorted to enable the pin** (see detail here

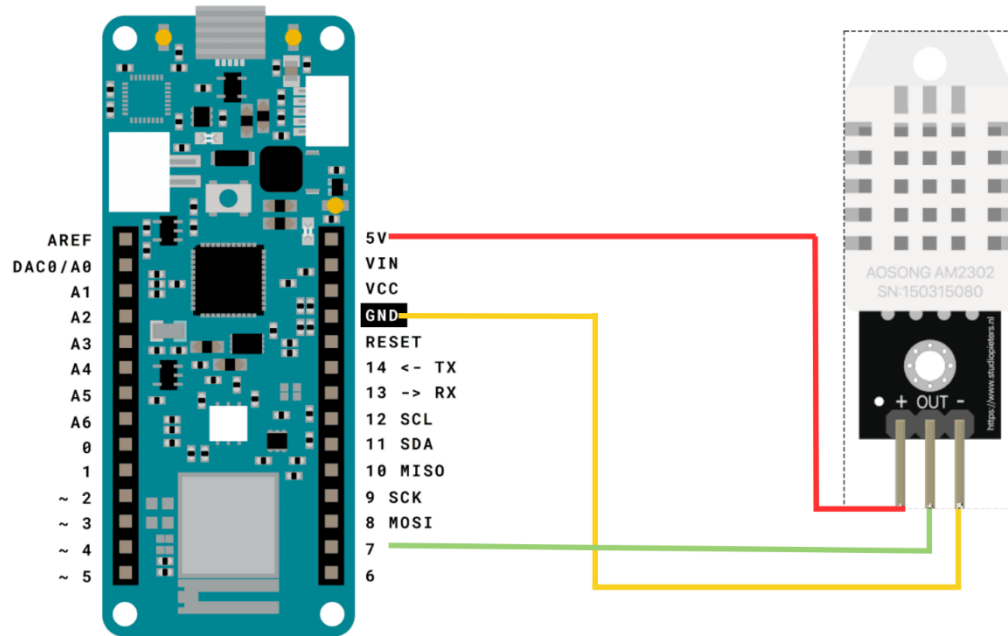
<https://support.arduino.cc/hc/en-us/articles/360014779679-Enable-5-V-power-on-the-VUSB-or-VBUS-pin-on-Nano-boards>).



To test, you can use a wire and connect these 2 pads manually by hand to see if data is coming through DHT22 sensor. For long data collection, you will need to solder the wire permanently to connect the pads. Seeking help from tutors there is an on-campus facility called Maker Space where you can do it.

Steps:

| Step | Action |
|------|---|
| 1 | <p>Connect your DHT22 Temperature and Humidity Sensor to the Arduino board. Note that the pin layout in the image below may look different than the board you may have.</p> |

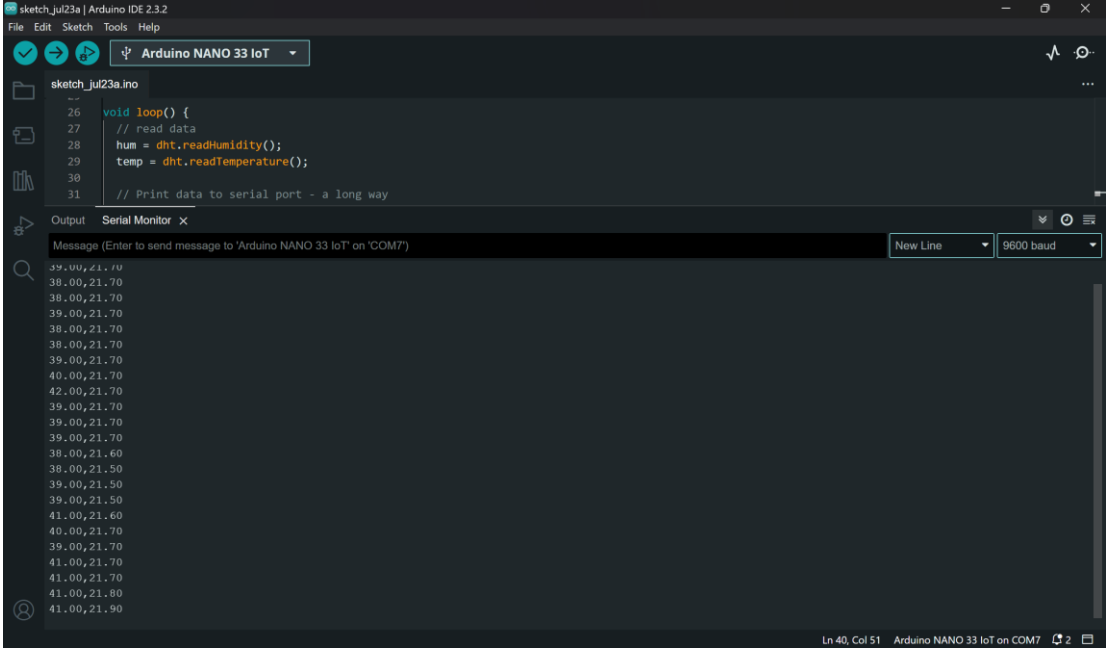


- Pick a red male-female jumper wire and attach the female end to pin 1 (VCC pin) on the sensor. Plug the male end into the Arduino board's 5V power pin.
- Pick a blue male-female jumper wire and attach the female end to pin 2 (DATA pin) on the sensor. Plug the male end into the Arduino board's digital data pin 2.
- Pick a black male-female jumper wire and attach the female end to pin 4 (GND) on the sensor. Plug the male end into the Arduino board's GND pin.
- Sensor's pin 3 is not used.

2 Connect your Arduino board to your computer using the USB cable.

3 Write an Arduino sketch (or download it from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/sketch_dht22.ino) which looks like below. Compile the code in Arduino IDE, deploy to the board and observe output in the Arduino IDE serial monitor.

| | |
|---|--|
| | <pre> 6 7 // Add "DHT sensor library" 8 #include <DHT.h> 9 10 #define DHTPIN 2 // digital pin number 11 #define DHTTYPE DHT22 // DHT type 11 or 22 12 DHT dht(DHTPIN, DHTTYPE); 13 14 // variables to store data. 15 float hum, temp; 16 17 void setup() { 18 // Set baud rate for serial communication 19 Serial.begin(9600); 20 21 // initialise DHT library 22 dht.begin(); 23 } 24 25 void loop() { 26 // read data 27 hum = dht.readHumidity(); 28 temp = dht.readTemperature(); 29 30 // Print data to serial port - a long way 31 // 32 // Serial.print("Humid: "); 33 // Serial.print(hum); 34 // Serial.print(" %, Temp: "); 35 // Serial.print(temp); 36 // Serial.println(" Celsius"); 37 38 // Print data to serial port - a compact way 39 Serial.println(String(hum) + "," + String(temp)); 40 41 // wait a while 42 delay(15*1000); 43 } </pre> |
| 4 | <p>Question: A spec of the DHT22 sensor is given in the link below. It mentions that the sampling rate is 0.5 Hz.</p> <p>https://lastminuteengineers.com/dht11-dht22-arduino-tutorial</p> <ul style="list-style-type: none"> i) What does the sampling rate mean? ii) Where is this used in the Arduino code? <p>Answer: i) The frequency at which a sensor may acquire a new measurement is known as its sampling rate. Given that 0.5 Hz is half of 1 Hz, which is one reading per second, the DHT22 sensor may obtain a new measurement every two seconds at a sampling rate of 0.5 Hz. The DHT22 sensor has a 2-second time frame to update its humidity and temperature measurements. This determines the highest frequency at which you can get fresh data from the sensor.</p> |

| | |
|---|---|
| | <p>ii) By making sure that the code doesn't try to read new data from the sensor more frequently than the sensor's designated sampling rate, the Arduino code follows the sampling rate. Usually, the loop() function is used to do this by adding a delay between each subsequent reading.</p> |
| 5 | <p>Question: Take a screenshot of your Serial Monitor displaying temperature & humidity sensor data logs. Add the image here.</p> <p>Answer:</p>  <p>The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar shows icons for running, uploading, and saving. The sketch is named 'sketch_jul23a.ino' and is being compiled for an 'Arduino NANO 33 IoT'. The code in the sketch is as follows:</p> <pre> 26 void loop() { 27 // read data 28 hum = dht.readHumidity(); 29 temp = dht.readTemperature(); 30 31 // Print data to serial port - a long way </pre> <p>The Serial Monitor window is open, showing the output of the sketch. The output consists of a series of data points, each representing a humidity and temperature reading, separated by a comma. The data points are as follows:</p> <pre> 39.00,21.70 38.00,21.70 38.00,21.70 39.00,21.70 38.00,21.70 38.00,21.70 39.00,21.70 40.00,21.70 42.00,21.70 39.00,21.70 39.00,21.70 39.00,21.70 38.00,21.60 38.00,21.50 39.00,21.50 39.00,21.50 41.00,21.60 40.00,21.70 39.00,21.70 41.00,21.70 41.00,21.70 41.00,21.80 41.00,21.90 </pre> <p>The Serial Monitor window also shows the baud rate set to 9600 and the port set to COM7. The status bar at the bottom indicates the current line and column in the sketch.</p> |

Activity 2.2: Working with sensor - HC-SR04

HC-SR04 is an Ultrasonic sensor.

Hardware Required

Arduino Board

HC-SR04 Ultrasonic sensor

USB cable

Software Required

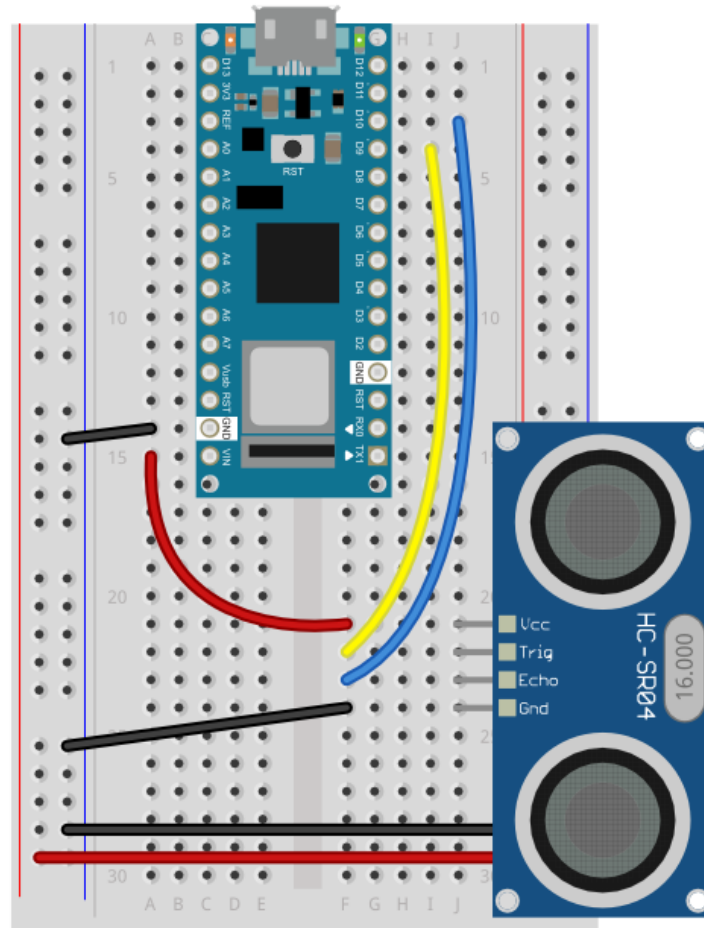
Arduino programming environment

Known issue, action required

The same known issue applies to SR04 which operates at 5 V source while the Arduino Nano 33 IoT supplies 3.3 V. It requires 2 VUSB (or VBUS) pads to be shorted to enable the pin.

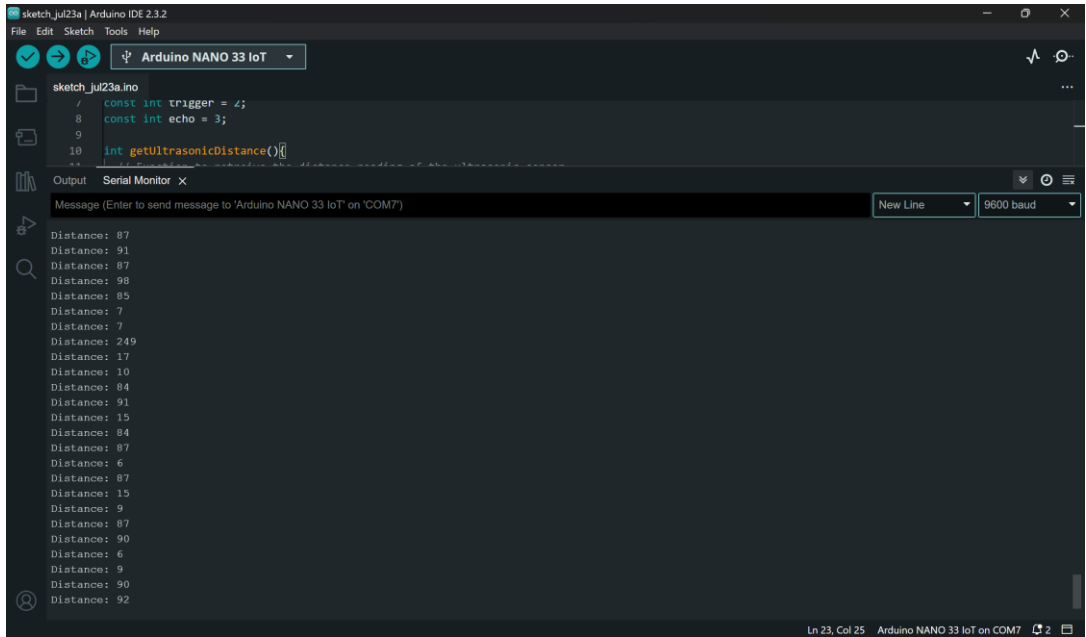
Steps:

| Step | Action |
|------|---|
| 1 | Connect your HC-SR04 sensor to the Arduino board. Note that the pin layout in the image below may look different than the board you may have. |



| | |
|---|---|
| 2 | Connect your Arduino board to your computer using the USB cable. |
| 3 | Write an Arduino sketch (or download it from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/sketch_hcsr04_distance.ino) which looks like below. Compile the code in Arduino IDE, deploy to the board and observe output in the Arduino IDE serial monitor. |

| | |
|---|--|
| | <pre> 7 const int trigger = 2; 8 const int echo = 3; 9 10 int getUltrasonicDistance(){ 11 // Function to retrieve the distance reading of the ultrasonic 12 long duration; 13 int distance; 14 15 // Assure the trigger pin is LOW: 16 digitalWrite(trigger, LOW); 17 // Brief pause: 18 delayMicroseconds(5); 19 20 // Trigger the sensor by setting the trigger to HIGH: 21 digitalWrite(trigger, HIGH); 22 // Wait a moment before turning off the trigger: 23 delayMicroseconds(10); 24 // Turn off the trigger: 25 digitalWrite(trigger, LOW); 26 27 // Read the echo pin: 28 duration = pulseIn(echo, HIGH); 29 // Calculate the distance in centimeter (CM): 30 distance = duration * 0.034 / 2; 31 32 // Uncomment this line to return value in IN instead of CM: 33 //distance = distance * 0.3937008 34 35 // Return the distance read from the sensor: 36 return distance; 37 } 38 39 void setup() { 40 // Define inputs and outputs: 41 pinMode(trigger, OUTPUT); 42 pinMode(echo, INPUT); 43 44 // Start the serial monitor: 45 Serial.begin(9600); 46 } 47 48 void loop() { 49 // Print the distance to the serial monitor: 50 Serial.print("Distance: "); 51 Serial.println(getUltrasonicDistance()); 52 53 // Wait one second before continuing: 54 delay(1000); 55 } </pre> |
| 4 | <p>Question: Spec of SR04 is available here (https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf). Identify 2 critical aspects you should be careful about this sensor operation.</p> <p>Answer: One critical aspect you should be careful about this sensor operation is the power supply voltage. The HC-SR04 operates on a power supply of 5V. Lower than 5V voltage applied to the sensor may cause it to malfunction or give false signals. For the sensor to work correctly, a steady 5V power source is needed.</p> |

| | |
|---|---|
| | <p>Make that the HC-SR04's VCC pin is connected to either the Arduino's 5V pin or the proper power supply.</p> <p>Another critical aspect you should be careful about this sensor operation is minimum and maximum distance. The HC-SR04 sensor has a minimum measurable distance of approximately 2cm and a maximum measurable distance of up to 400cm. Accurate measurement of distances longer than 400 cm or less than 2 cm is not possible with this sensor. Make sure the object you are aiming for is inside this range. When an object is too close (less than 2 cm) or too far away (more than 400 cm), the sensor may give false or zero distance readings. Additionally, keep the sensor away from areas with obstacles in its path or where it could pick up false echoes.</p> |
| 5 | <p>Question: Take a screenshot of your Serial Monitor displaying distance values while you try to generate a periodic motion by moving your hand gradually back and forth towards the sensor. Add the image here.</p> <p>Answer:</p>  <p>The screenshot shows the Arduino IDE interface with the 'Serial Monitor' window open. The monitor displays a series of distance readings in centimeters, such as 'Distance: 87', 'Distance: 91', 'Distance: 87', 'Distance: 98', 'Distance: 85', 'Distance: 7', 'Distance: 7', 'Distance: 249', 'Distance: 17', 'Distance: 10', 'Distance: 84', 'Distance: 91', 'Distance: 15', 'Distance: 84', 'Distance: 87', 'Distance: 6', 'Distance: 87', 'Distance: 15', 'Distance: 9', 'Distance: 87', 'Distance: 90', 'Distance: 6', 'Distance: 9', 'Distance: 90', and 'Distance: 92'. The code in the background includes constants for trigger and echo pins, and a function to get ultrasonic distance.</p> |

Activity 2.3: Working with sensor - Accelerometer

LSM6DS3 module on the Arduino Nano 33 IoT is an accelerometer and gyroscope sensor.

Hardware Required

Arduino Nano 33 IoT Board (has inbuilt LSM6DS3 module)

USB cable

Software Required

Arduino programming environment

Steps:

| Step | Action |
|------|--|
| 1 | Write Arduino sketch (or download code from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/sketch_accelero.ino) which looks like below. Compile the code in Arduino IDE, deploy to the board and observe output in the Arduino IDE serial monitor. |

```

6
7 // Add Arduino_LSM6DS3 library
8 // from Arduino IDE Library Manager.
9 #include <Arduino_LSM6DS3.h>
10
11 float x, y, z;
12
13 void setup() {
14     Serial.begin(9600); // set baud rate
15     while (!Serial); // wait for port to init
16     Serial.println("Started");
17
18     if (!IMU.begin()) {
19         Serial.println("Failed to initialize IMU!");
20         while (1);
21     }
22
23     Serial.println(
24         "Accelerometer sample rate = "
25         + String(IMU.accelerationSampleRate()) + " Hz");
26 }
27
28 void loop() {
29     // read accelero data
30     if (IMU.accelerationAvailable()) {
31         IMU.readAcceleration(x, y, z);
32     }
33
34     Serial.println(
35         String(x) + ", " + String(y) + ", " + String(z));
36
37     delay(1000);
38 }

```

2

Question: Spec of LSM6DS3 is available here (https://content.arduino.cc/assets/st_imu_lsm6ds3_datasheet.pdf). Identify at least 3 attributes of this sensor you think important to work with.

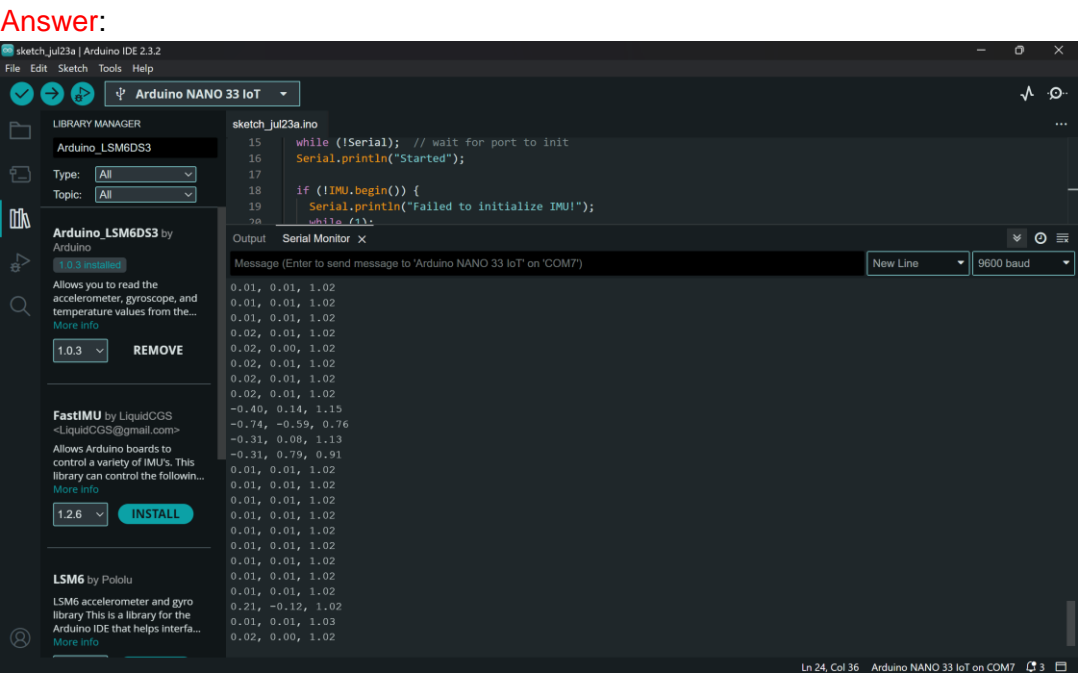
Answer: One attribute of this sensor that I think is important to work with is the **accelerometer range**. The LSM6DS3 accelerometer can be configured with ranges of +/- 2g, +/- 4g, +/- 8g, and +/- 16g. The maximum acceleration that the accelerometer is capable of measuring is determined by its range. To accurately capture the intended level of acceleration in your application, you must choose a suitable range. For small motions, for instance, a range of ± 2 g is appropriate, but for high-impact applications, ± 16 g is required.

Another attribute is **sampling rate**, where the LSM6DS3 supports various sampling rates for its accelerometer, with a maximum rate of 6.66 kHz. The frequency at which the sensor changes its readings is dependent on the sampling rate. For applications needing high precision or fast reaction times, a

higher sample rate makes updates possible more quickly and in-depth. To accurately catch changes in quick movements or vibrations, for example, a greater sample rate will be required.

The last attribute is **power consumption**, where the LSM6DS3 features low power consumption with different operating modes, including a low-power mode. In battery-operated or energy-sensitive applications, power consumption is crucial. Switching to low-power modes is beneficial for portable devices or systems where power efficiency is critical since it helps prolong battery life. Effective power mode management can have a big impact on the device's lifespan and overall performance.

3 **Question:** Take a screenshot of your Serial Monitor displaying sensor readings. Add the image here.




4 **Question:** Identify the max sampling rate and consider reducing the delay (line 37 in the sketch) to increase the number of samples. Summarise your findings here.

Answer: Summary of Findings

The LSM6DS3 sensor supports a maximum sampling rate of 6.66 kHz or 6660 Hz for the accelerometer. The current delay in the sketch is set to 1000 ms which is 1 second, which limits the rate at which new samples are taken and printed to the serial monitor to 1 sample per second. Given the maximum sampling rate of 6.66 kHz, the sensor can provide up to 6660 samples per second. By reducing the

delay, the number of samples increase through changing the delay to 100 ms or 0.1 second, would output approximately 10 samples per second, and then further reducing it to 10 ms, this would give around 100 samples per second. You can obtain more data points per second and make greater use of the maximum sampling rate of the sensor by adjusting these settings.



The screenshot shows the Arduino IDE interface. At the top, the title bar reads "sketch_jul23a | Arduino IDE 2.3.4". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for checking, running, and uploading, along with a dropdown menu currently set to "Arduino NANO 33 IoT".

The main workspace displays a C++ sketch named "sketch_jul23a.ino". The code is as follows:

```
1 // Variables declaration
2 int x = 0;
3 int y = 0;
4 int z = 0;
5
6 // Variables declaration
7 float x2 = 0.0;
8 float y2 = 0.0;
9 float z2 = 0.0;
10
11 // Variables declaration
12 float x3 = 0.0;
13 float y3 = 0.0;
14 float z3 = 0.0;
15
16 // Variables declaration
17 float x4 = 0.0;
18 float y4 = 0.0;
19 float z4 = 0.0;
20
21 // Variables declaration
22 float x5 = 0.0;
23 float y5 = 0.0;
24 float z5 = 0.0;
25
26 // Variables declaration
27 float x6 = 0.0;
28 float y6 = 0.0;
29 float z6 = 0.0;
30
31 // Variables declaration
32 float x7 = 0.0;
33 float y7 = 0.0;
34 float z7 = 0.0;
35
36 // Variables declaration
37 float x8 = 0.0;
38 float y8 = 0.0;
39 float z8 = 0.0;
40
41 // Variables declaration
42 float x9 = 0.0;
43 float y9 = 0.0;
44 float z9 = 0.0;
45
46 // Variables declaration
47 float x10 = 0.0;
48 float y10 = 0.0;
49 float z10 = 0.0;
50
51 // Variables declaration
52 float x11 = 0.0;
53 float y11 = 0.0;
54 float z11 = 0.0;
55
56 // Variables declaration
57 float x12 = 0.0;
58 float y12 = 0.0;
59 float z12 = 0.0;
60
61 // Variables declaration
62 float x13 = 0.0;
63 float y13 = 0.0;
64 float z13 = 0.0;
65
66 // Variables declaration
67 float x14 = 0.0;
68 float y14 = 0.0;
69 float z14 = 0.0;
70
71 // Variables declaration
72 float x15 = 0.0;
73 float y15 = 0.0;
74 float z15 = 0.0;
75
76 // Variables declaration
77 float x16 = 0.0;
78 float y16 = 0.0;
79 float z16 = 0.0;
80
81 // Variables declaration
82 float x17 = 0.0;
83 float y17 = 0.0;
84 float z17 = 0.0;
85
86 // Variables declaration
87 float x18 = 0.0;
88 float y18 = 0.0;
89 float z18 = 0.0;
90
91 // Variables declaration
92 float x19 = 0.0;
93 float y19 = 0.0;
94 float z19 = 0.0;
95
96 // Variables declaration
97 float x20 = 0.0;
98 float y20 = 0.0;
99 float z20 = 0.0;
100
101 // Variables declaration
102 float x21 = 0.0;
103 float y21 = 0.0;
104 float z21 = 0.0;
105
106 // Variables declaration
107 float x22 = 0.0;
108 float y22 = 0.0;
109 float z22 = 0.0;
110
111 // Variables declaration
112 float x23 = 0.0;
113 float y23 = 0.0;
114 float z23 = 0.0;
115
116 // Variables declaration
117 float x24 = 0.0;
118 float y24 = 0.0;
119 float z24 = 0.0;
120
121 // Variables declaration
122 float x25 = 0.0;
123 float y25 = 0.0;
124 float z25 = 0.0;
125
126 // Variables declaration
127 float x26 = 0.0;
128 float y26 = 0.0;
129 float z26 = 0.0;
130
131 // Variables declaration
132 float x27 = 0.0;
133 float y27 = 0.0;
134 float z27 = 0.0;
135
136 // Variables declaration
137 float x28 = 0.0;
138 float y28 = 0.0;
139 float z28 = 0.0;
140
141 // Variables declaration
142 float x29 = 0.0;
143 float y29 = 0.0;
144 float z29 = 0.0;
145
146 // Variables declaration
147 float x30 = 0.0;
148 float y30 = 0.0;
149 float z30 = 0.0;
150
151 // Variables declaration
152 float x31 = 0.0;
153 float y31 = 0.0;
154 float z31 = 0.0;
155
156 // Variables declaration
157 float x32 = 0.0;
158 float y32 = 0.0;
159 float z32 = 0.0;
160
161 // Variables declaration
162 float x33 = 0.0;
163 float y33 = 0.0;
164 float z33 = 0.0;
165
166 // Variables declaration
167 float x34 = 0.0;
168 float y34 = 0.0;
169 float z34 = 0.0;
170
171 // Variables declaration
172 float x35 = 0.0;
173 float y35 = 0.0;
174 float z35 = 0.0;
175
176 // Variables declaration
177 float x36 = 0.0;
178 float y36 = 0.0;
179 float z36 = 0.0;
180
181 // Variables declaration
182 float x37 = 0.0;
183 float y37 = 0.0;
184 float z37 = 0.0;
185
186 // Variables declaration
187 float x38 = 0.0;
188 float y38 = 0.0;
189 float z38 = 0.0;
190
191 // Variables declaration
192 float x39 = 0.0;
193 float y39 = 0.0;
194 float z39 = 0.0;
195
196 // Variables declaration
197 float x40 = 0.0;
198 float y40 = 0.0;
199 float z40 = 0.0;
200
201 // Variables declaration
202 float x41 = 0.0;
203 float y41 = 0.0;
204 float z41 = 0.0;
205
206 // Variables declaration
207 float x42 = 0.0;
208 float y42 = 0.0;
209 float z42 = 0.0;
210
211 // Variables declaration
212 float x43 = 0.0;
213 float y43 = 0.0;
214 float z43 = 0.0;
215
216 // Variables declaration
217 float x44 = 0.0;
218 float y44 = 0.0;
219 float z44 = 0.0;
220
221 // Variables declaration
222 float x45 = 0.0;
223 float y45 = 0.0;
224 float z45 = 0.0;
225
226 // Variables declaration
227 float x46 = 0.0;
228 float y46 = 0.0;
229 float z46 = 0.0;
230
231 // Variables declaration
232 float x47 = 0.0;
233 float y47 = 0.0;
234 float z47 = 0.0;
235
236 // Variables declaration
237 float x48 = 0.0;
238 float y48 = 0.0;
239 float z48 = 0.0;
240
241 // Variables declaration
242 float x49 = 0.0;
243 float y49 = 0.0;
244 float z49 = 0.0;
245
246 // Variables declaration
247 float x50 = 0.0;
248 float y50 = 0.0;
249 float z50 = 0.0;
250
251 // Variables declaration
252 float x51 = 0.0;
253 float y51 = 0.0;
254 float z51 = 0.0;
255
256 // Variables declaration
257 float x52 = 0.0;
258 float y52 = 0.0;
259 float z52 = 0.0;
260
261 // Variables declaration
262 float x53 = 0.0;
263 float y53 = 0.0;
264 float z53 = 0.0;
265
266 // Variables declaration
267 float x54 = 0.0;
268 float y54 = 0.0;
269 float z54 = 0.0;
270
271 // Variables declaration
272 float x55 = 0.0;
273 float y55 = 0.0;
274 float z55 = 0.0;
275
276 // Variables declaration
277 float x56 = 0.0;
278 float y56 = 0.0;
279 float z56 = 0.0;
280
281 // Variables declaration
282 float x57 = 0.0;
283 float y57 = 0.0;
284 float z57 = 0.0;
285
286 // Variables declaration
287 float x58 = 0.0;
288 float y58 = 0.0;
289 float z58 = 0.0;
290
291 // Variables declaration
292 float x59 = 0.0;
293 float y59 = 0.0;
294 float z59 = 0.0;
295
296 // Variables declaration
297 float x60 = 0.0;
298 float y60 = 0.0;
299 float z60 = 0.0;
300
301 // Variables declaration
302 float x61 = 0.0;
303 float y61 = 0.0;
304 float z61 = 0.0;
305
306 // Variables declaration
307 float x62 = 0.0;
308 float y62 = 0.0;
309 float z62 = 0.0;
310
311 // Variables declaration
312 float x63 = 0.0;
313 float y63 = 0.0;
314 float z63 = 0.0;
315
316 // Variables declaration
317 float x64 = 0.0;
318 float y64 = 0.0;
319 float z64 = 0.0;
320
321 // Variables declaration
322 float x65 = 0.0;
323 float y65 = 0.0;
324 float z65 = 0.0;
325
326 // Variables declaration
327 float x66 = 0.0;
328 float y66 = 0.0;
329 float z66 = 0.0;
330
331 // Variables declaration
332 float x67 = 0.0;
333 float y67 = 0.0;
334 float z67 = 0.0;
335
336 // Variables declaration
337 float x68 = 0.0;
338 float y68 = 0.0;
339 float z68 = 0.0;
340
341 // Variables declaration
342 float x69 = 0.0;
343 float y69 = 0.0;
344 float z69 = 0.0;
345
346 // Variables declaration
347 float x70 = 0.0;
348 float y70 = 0.0;
349 float z70 = 0.0;
350
351 // Variables declaration
352 float x71 = 0.0;
353 float y71 = 0.0;
354 float z71 = 0.0;
355
356 // Variables declaration
357 float x72 = 0.0;
358 float y72 = 0.0;
359 float z72 = 0.0;
360
361 // Variables declaration
362 float x73 = 0.0;
363 float y73 = 0.0;
364 float z73 = 0.0;
365
366 // Variables declaration
367 float x74 = 0.0;
368 float y74 = 0.0;
369 float z74 = 0.0;
370
371 // Variables declaration
372 float x75 = 0.0;
373 float y75 = 0.0;
374 float z75 = 0.0;
375
376 // Variables declaration
377 float x76 = 0.0;
378 float y76 = 0.0;
379 float z76 = 0.0;
```

The screenshot shows the Arduino IDE 2.3.2 interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for opening files, saving, compiling, and uploading. The current sketch is named "sketch_jul23a.ino" and is targeted to an "Arduino NANO 33 IoT" board. The sketch code is as follows:

```

32 }
33
34 Serial.println(
35   String(x) + ", " + String(y) + ", " + String(z));
36
37   delay(10);
38 }

```

The Serial Monitor is open, showing the output of the sketch. The output consists of multiple lines of comma-separated values, such as "-0.03, -0.02, 1.01". The Serial Monitor settings are set to "New Line" and "9600 baud". A status bar at the bottom indicates "Done uploading."

Activity 2.4: Plot data using Python Notebook

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. You can find detail in official website (<https://matplotlib.org>).

Hardware Required

No hardware required

Software Required

Python Jupyter notebook

Steps:

| Step | Action |
|------|--|
| 1 | Download the Jupyter notebook week2_notebook.ipynb from here (https://github.com/deakin-deep-dreamer/sit225/blob/main/week_2/week2_notebook.ipynb). Follow the instructions in the notebook to carry out instructions and finally convert the notebook to PDF so you can combine it with this activity sheet PDF. |