Sk8 Training Monitor

Architectural Association
ETS4 Digital Ecologies
Ningwei Liu
2023-24

Olafur Eliasson @TATE Modern

•Immersive interactive experience;
•Co-authorship: user & artist

Assignment #1
Persona

L

General info

26 years old
Uni student
Sitting more than 8 hours a day
Trying to work out regularly

Goals

Practicing every other day
Not picking up her phone during the exercise session

Personality

Eager to learn
Passionate and dedicated
Resilient and adaptable

Frustrations

Getting distracted
No habit of exercise

Motivations

Living happier & healthier
Better time management

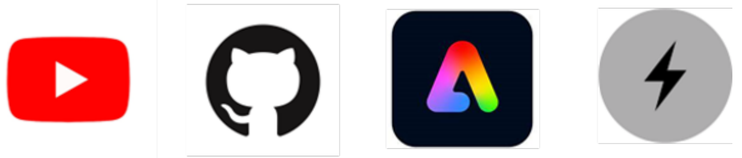
Preferred channels

Instagram, Discord

Bio

L is a university student living in London. She often sits for long hours and stays late for work, and feels the urgent need to work out. She always enjoys watching sports, but is never good as at any sports. She recently started to skate, and goes skateboarding twice a week.

Brand logos



Assignment #1
User stories

As a sport-enthusiast,
L **wants to** know how many time she practices the ollie,
so that She can keep track on her skateboarding progress.”

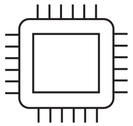
Assignment #1
Storyboard

Assignment #2

Components

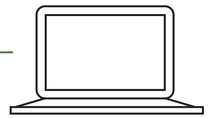
Sensor: Wemos LSM303
Accelerometers/Magnetometer

Control Signal



esp8266

Control Signal



laptop
to storage analyze data

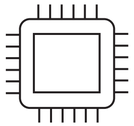
Line Voltage

Control Signal

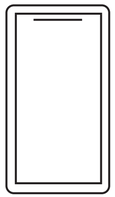
lights
to indicate rating of the motion

Control Signal

Control Signal
& Line Voltage



esp8266



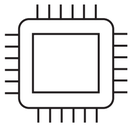
user's phone
to monitor the training &
rating

Assignment #2

System Diagram and Protocols

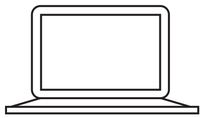
LSM303

Accelerometers/Magnetometer



esp8266

programmable controller
to communicate

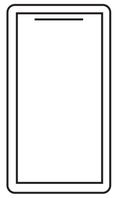


laptop

data storage & analysis

lights

assign criteria to lights
instantly show the rating

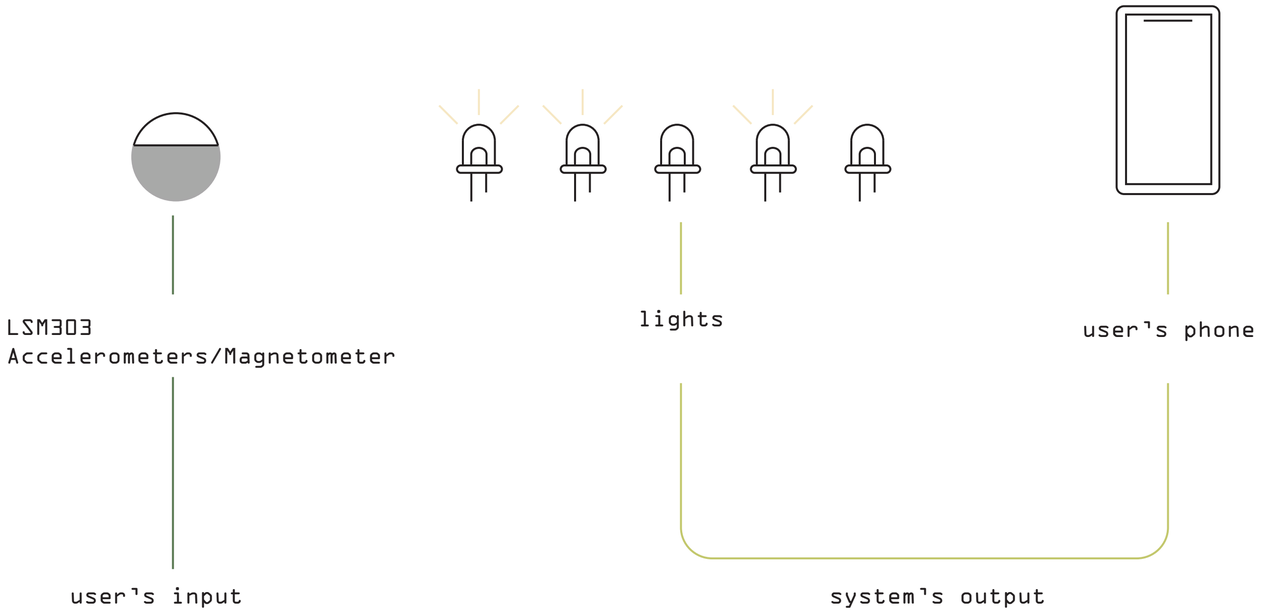


user's phone

monitor the training progress

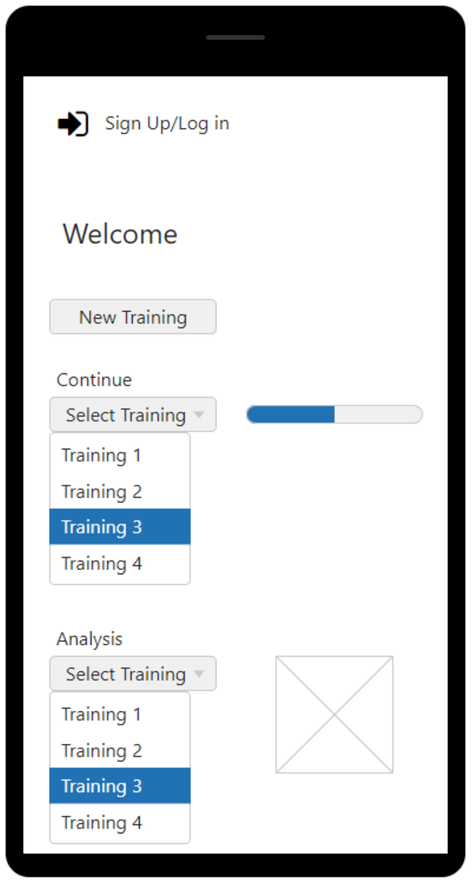
Assignment #2

Touch Points and Modes of Interaction

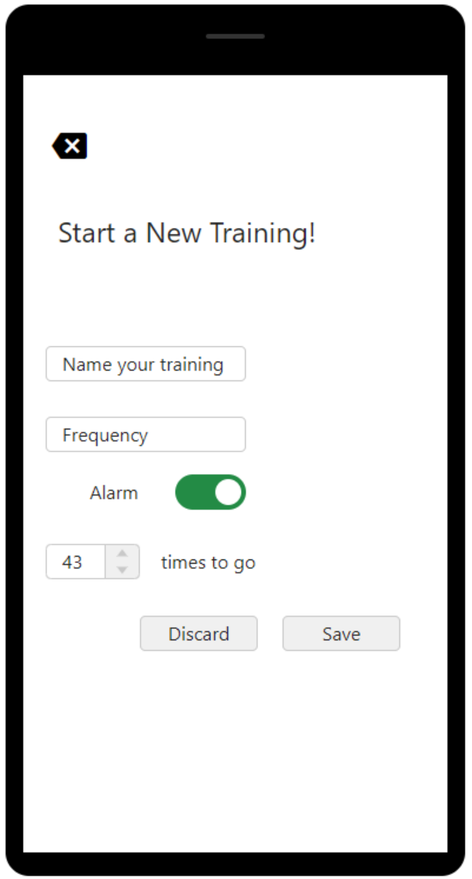


Assignment #3

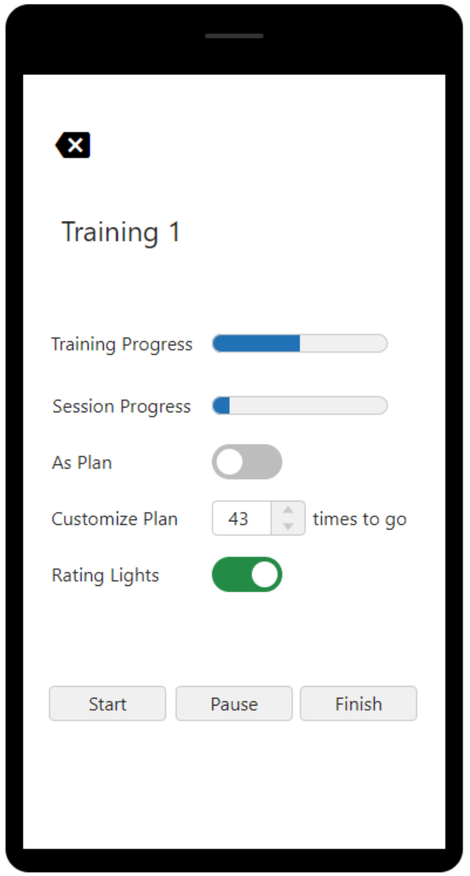
UI Wireframe + GitHub username



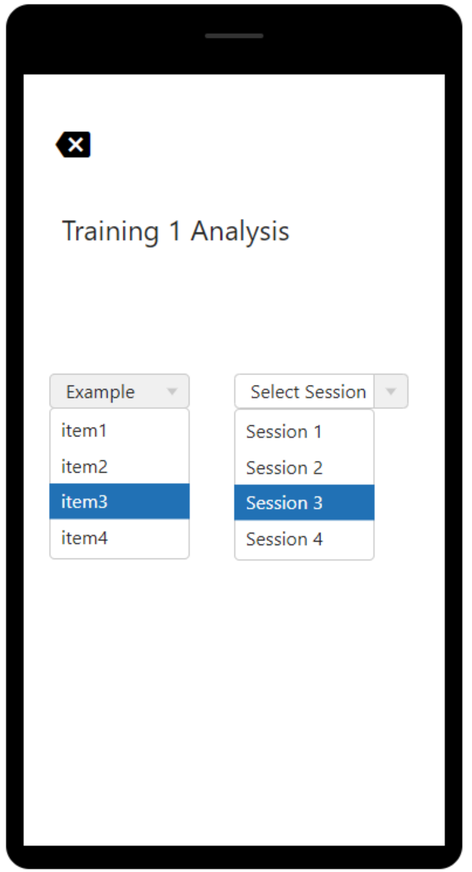
Home Page



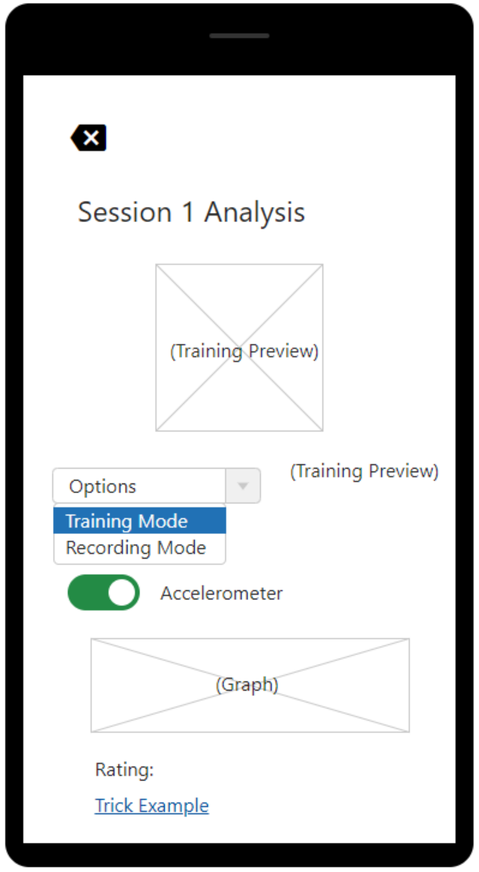
New Training



Continue Training



Navigate to analysis



Analysis Result

GitHub username: nliu12333

Assignment #4

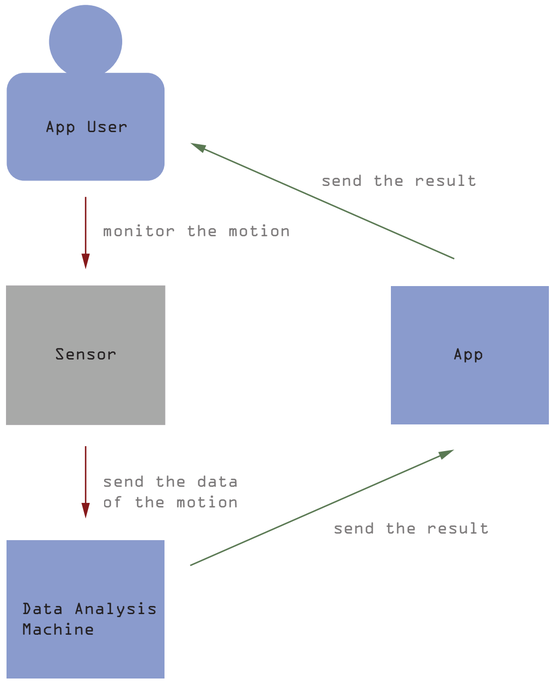
Bill of Materials

Assembly Name :	Product name
Assembly Number :	1
Assembly Revision :	1.0
Approval Date :	
Part Count :	
Conversion EUR-GBP :	0.9
Total Cost GBP	£10.74

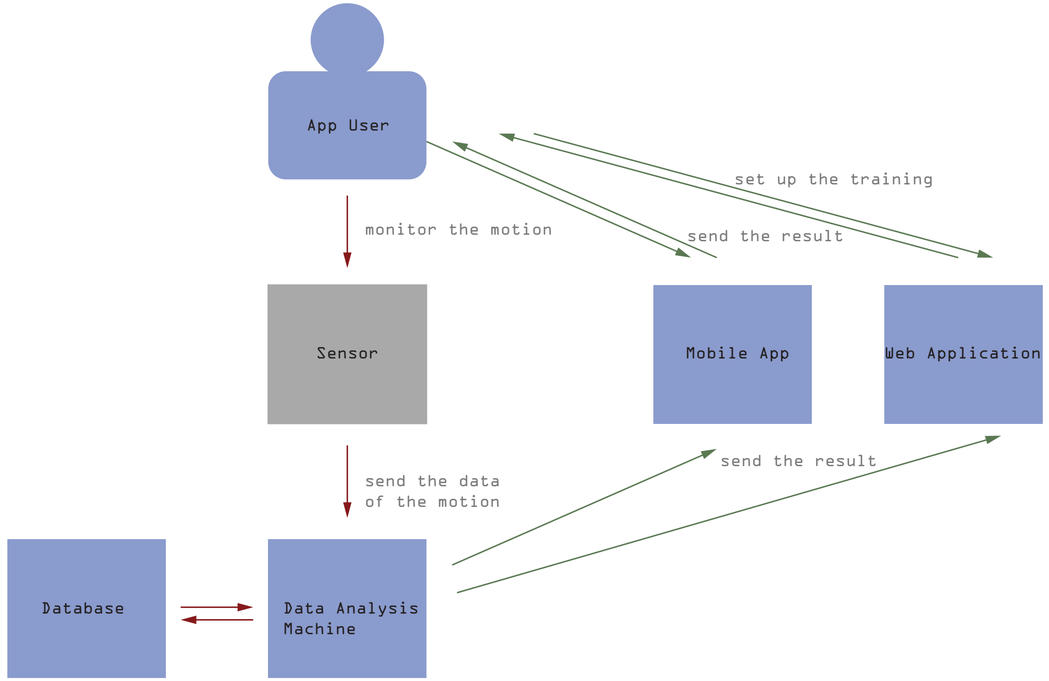
Category	Part description	Product name	Qty	Note	Picture	Supplier	Link	Unit Cost GBP	Total Cost GBP	Unit Cost EUI	Notes
Physical	a box attaching to the skate	laser skate box	1								
Wiring											
connectivity											
Fixings											
Controls	Development Board	ESP 8266	1		#VALUE!	ebay	ESP8266 for sale | eBay	£4.79	£4.79		
Wirings											
Sensor	3D digital linear acceleration sensor; 3D digital magnetic sensor	LSM303 3D Accelerometers	1		#VALUE!	ebay	https://www.ebay.co.uk/sch/i.html?_from=R40&_trksid=p2334524.m570.l1311&_nkw=adxl345&_sacat=0&_odkw=LSM303&_osacat=0	£5.95	£5.95		Connecting cables included
								Total	£10.74		

Assignment #5

C4 Diagram



Level 1: Context

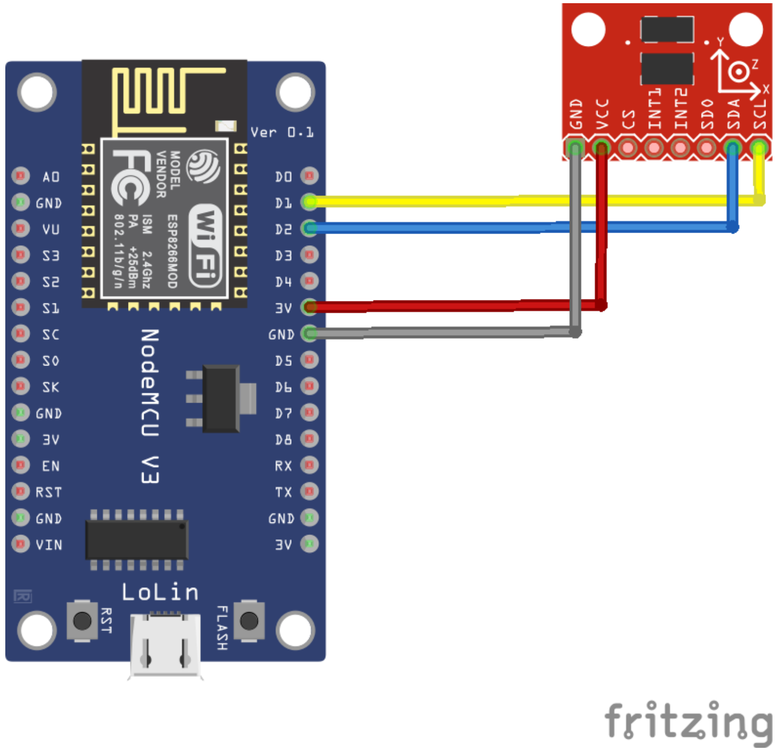


Level 2: Containers

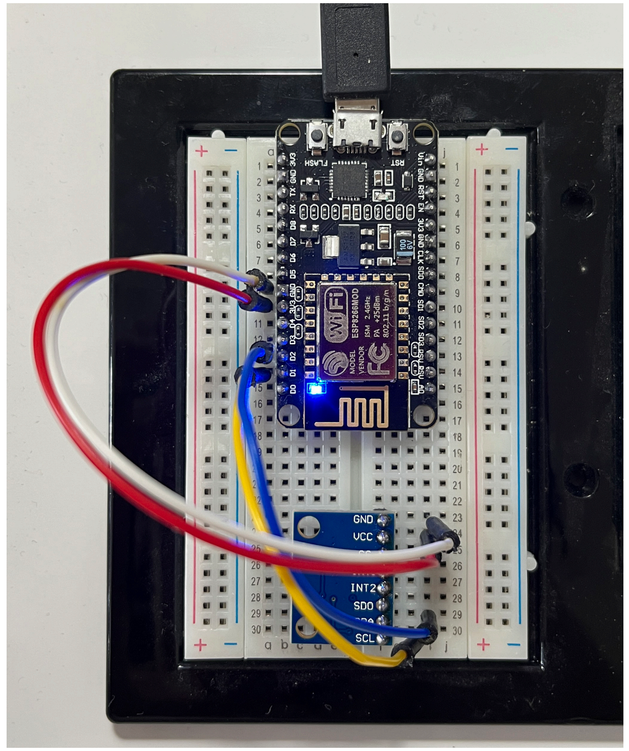
Skate Training Monitor

Wiring Diagram & Photo

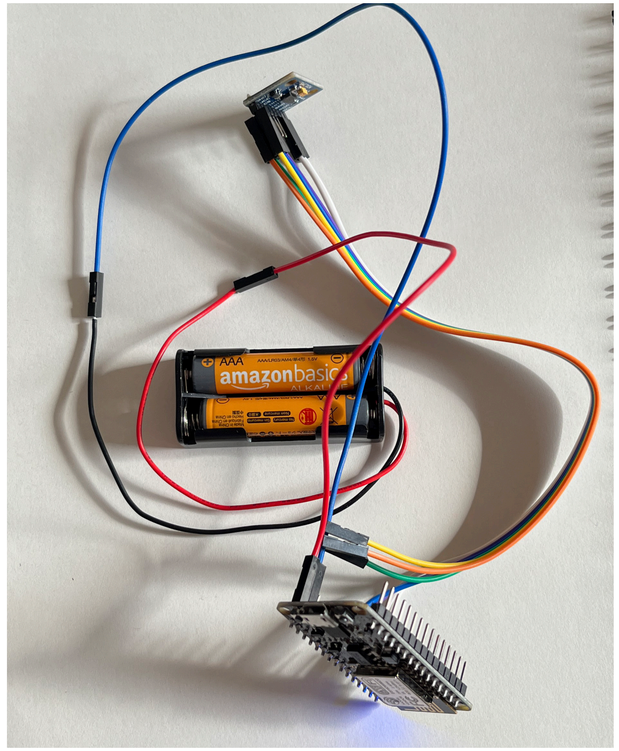
The goal here is to attach the device on the skateboard to draw real-time acceleration data while riding.



Wiring Diagram



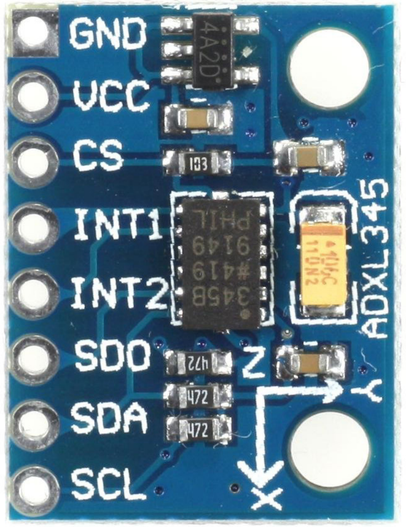
Testing Device
(connecting to & powered by computor)



Complete Device with power supply

Skate Training Monitor
Testing #1: ADXL 345 accelerometer

The sensor



ADXL345 accelerometer

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16g.

- VCC: Power supply pin connects in the range of 3 to 5.5V DC.
- CS: chip select
- SDO: Serial Data Out
- SDA: serial data pin
- SCL: serial clock pin

I2C protocol: 2 lines, a serial clock pin (SCL) that the Arduino Controller board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices.

Library used: Adafruit_ADXL345

Output

```
Output
. Variables and constants in RAM (global, static), used 29048 / 80192 bytes (36%)
|   SEGMENT  BYTES    DESCRIPTION
|-- DATA     1504     initialized variables
|-- RODATA   1288     constants
`-- BSS      26256    zeroed variables
. Instruction RAM (IRAM_ATTR, ICACHE_RAM_ATTR), used 60931 / 65536 bytes (92%)
|   SEGMENT  BYTES    DESCRIPTION
|-- ICACHE   32768    reserved space for flash instruction cache
`-- IRAM     28163    code in IRAM
. Code in flash (default, ICACHE_FLASH_ATTR), used 245560 / 1048576 bytes (23%)
|   SEGMENT  BYTES    DESCRIPTION
`-- IROM     245560   code in flash
esptool.py v3.0
Serial port COM3
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 84:f3:eb:b6:83:04
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 280672 bytes to 205549...
Writing at 0x00000000... (7 %)
Writing at 0x00004000... (15 %)
Writing at 0x00008000... (23 %)
Writing at 0x0000c000... (30 %)
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 280672 bytes (205549 compressed) at 0x00000000 in 18.2 seconds (effective 123.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

result of verifying

result of successful uploading

Test #1 ADXL345

Understand the Code

The Code in GitHub: 1_ADXL345_sensortest
based on the sensortest example from ADXL345 library

The libraries

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
```

Assign ID to the sensor

```
/* Assign a unique ID to this sensor at the same time */
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);
```

displaySensorDetails(void): print the data of the sensor

```
void displaySensorDetails(void)
{
  sensor_t sensor;
  accel.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" m/s^2");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" m/s^2");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" m/s^2");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}
```

displayDataRate(void) & setDataRate(datarate)
select the data rate from slowest to fastest which is 10Hz to 3200Hz

```
void displayDataRate(void)
{
  Serial.print  ("Data Rate:    ");

  switch(accel.getDataRate())
  {
    case ADXL345_DATARATE_3200_HZ:
      Serial.print  ("3200 ");
      break;
    case ADXL345_DATARATE_1600_HZ:
      Serial.print  ("1600 ");
      break;
    case ADXL345_DATARATE_800_HZ:
      Serial.print  ("800 ");
      break;
    case ADXL345_DATARATE_400_HZ:
      Serial.print  ("400 ");
      break;
    case ADXL345_DATARATE_200_HZ:
      Serial.print  ("200 ");
      break;
    case ADXL345_DATARATE_100_HZ:
      Serial.print  ("100 ");
      break;
    case ADXL345_DATARATE_50_HZ:
      Serial.print  ("50 ");
      break;
    case ADXL345_DATARATE_25_HZ:
      Serial.print  ("25 ");
      break;
    case ADXL345_DATARATE_12_5_HZ:
      Serial.print  ("12.5 ");
      break;
    case ADXL345_DATARATE_6_25HZ:
      Serial.print  ("6.25 ");
      break;
    case ADXL345_DATARATE_3_13_HZ:
      Serial.print  ("3.13 ");
      break;
    case ADXL345_DATARATE_1_56_HZ:
      Serial.print  ("1.56 ");
      break;
    case ADXL345_DATARATE_0_78_HZ:
      Serial.print  ("0.78 ");
      break;
    case ADXL345_DATARATE_0_39_HZ:
      Serial.print  ("0.39 ");
      break;
    case ADXL345_DATARATE_0_20_HZ:
      Serial.print  ("0.20 ");
      break;
    case ADXL345_DATARATE_0_10_HZ:
      Serial.print  ("0.10 ");
      break;
    default:
      Serial.print  ("???? ");
      break;
  }
  Serial.println(" Hz");
}
```

displayRange(void)
The default range is ±16g. [±156.8 m/s² (16 * 9.8)]

```
void displayRange(void)
{
  Serial.print  ("Range:         +/- ");

  switch(accel.getRange())
  {
    case ADXL345_RANGE_16_G:
      Serial.print  ("16 ");
      break;
    case ADXL345_RANGE_8_G:
      Serial.print  ("8 ");
      break;
    case ADXL345_RANGE_4_G:
      Serial.print  ("4 ");
      break;
    case ADXL345_RANGE_2_G:
      Serial.print  ("2 ");
      break;
    default:
      Serial.print  ("?? ");
      break;
  }
  Serial.println(" g");
}
```

Test #1 ADXL345

Understand the Code & the Accelerometer

Setup function

initiated the serial commincation with 9600 Baud rate
(because esp8266 is using 115200, later when using the sensor with webser, the serial comminication rate should set to 115200)

```
110  void setup(void)
111  {
112  #ifndef ESP8266
113    while (!Serial); // for Leonardo/Micro/Zero
114  #endif
115    Serial.begin(9600);
116    Serial.println("Accelerometer Test"); Serial.println("");
```

Set the range of the accelerometer

```
126    /* Set the range to whatever is appropriate for your project */
127    accel.setRange(ADXL345_RANGE_16_G);
128    // accel.setRange(ADXL345_RANGE_8_G);
129    // accel.setRange(ADXL345_RANGE_4_G);
130    // accel.setRange(ADXL345_RANGE_2_G);
```

Print out the detail information of the sensor

```
132    /* Display some basic information on this sensor */
133    displaySensorDetails();
134
135    /* Display additional settings (outside the scope of sensor_t) */
136    displayDataRate();
137    displayRange();
138    Serial.println("");
139  }
```

Loop function

create an event object: sensors_event_t is a user-defined datatype (Structures in C) that contains data from the ADXL345 sensor from a specific moment in time.

```
141  void loop(void)
142  {
143    /* Get a new sensor event */
144    sensors_event_t event;
145    accel.getEvent(&event);
```

Print out the results in serial monitor

```
147    /* Display the results (acceleration is measured in m/s^2) */
148    Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");
149    Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");
150    Serial.print("Z: "); Serial.print(event.acceleration.z); Serial.print("  ");Serial.println("m/s^2 ");
151    delay(500);
152  }
153
```

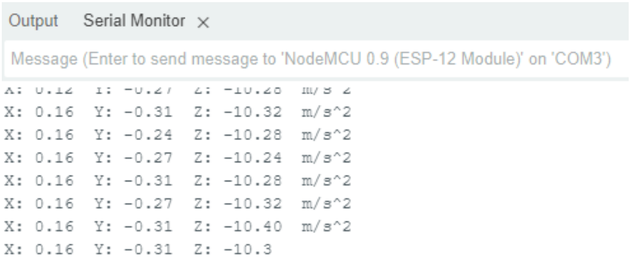
Troubleshoot #1

serial monitor doesn't print out the sensor details after uploading

solution: by default, current version of Arduino IDE (2.3.2) will only run the functions in loop(void). After uploading, print RST on the ESP8266 to run the set-up functions again and print out the sensor details.

Serial Monitor

output from the sensor



Serial Monitor

details of the sensor

```
IOOOO0OOOOOOOOOAccelerometer Test

------------------------------------
Sensor:       ADXL345
Driver Ver:   1
Unique ID:    12345
Max Value:    -156.91 m/s^2
Min Value:    156.91 m/s^2
Resolution:   0.04 m/s^2
------------------------------------

Data Rate:    100  Hz
Range:         +/- 16  g

X: -0.71  Y: -0.71  Z: -10.24  m/s^2
X: -0.75  Y: -0.71  Z: -10.28  m/s^2
X: -0.67  Y: -0.75  Z: -10.24  m/s^2
X: -0.71  Y: -0.75  Z: -10.24  m/s^2
```

Test #2 ESP8266, web server displays graph

Testing & troubleshooting

The Code in GitHub: 2_Graph
based on the Gragh example from esp8266WebServer

Set the wifi detail to allow esp8266 connect to the phone

```
57  #define DBG_OUTPUT_PORT Serial
58
59  #ifndef STASSID
60  #define STASSID "NingweisiPhone"
61  #define STAPSK "1A2B3C4D"
62  #endif
```

Troubleshoot #2

esp8266 doesn't connect

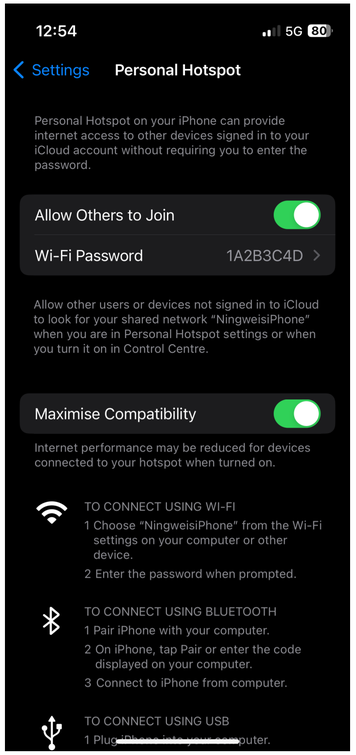
Serial Monitor × Output

Message (Enter to send message to 'NodeMCU 0.9 (ESP-12 Module)' on 'COM3')

```
no NingweisiPhone found, reconnect after 1s
reconnect
......scandone
no NingweisiPhone found, reconnect after 1s
reconnect
......scandone
no NingweisiPhone found, reconnect after 1s
reconnect
......scandone
no NingweisiPhone found, reconnect after 1s
reconnect
.....scandone
no NingweisiPhone found, reconnect after 1s
.reconnect
..
```

Reason & solution: eso8266 doesn't take 5Ghz wifi. With iPhone, the wifi hot-spot is 5GHz by default. Turn on "Maximise Compatibility" to switch to 2.4GHz signal.

Now if run the wifiscan, it will find "NingweisiPhone".



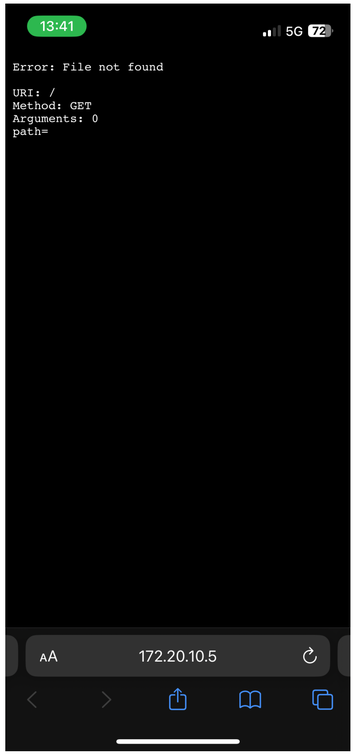

```
Starting WiFi scan...
5 networks found:
  00: [CH 01] [AC:A4:6E:FF:4C:69] -72dBm * V 802.11b/g/n WPS ALHN-6AA7
  01: [CH 01] [18:E8:29:E4:85:2E] -87dBm * V 802.11b/g/n     BEDFORD-WIFI
  02: [CH 06] [2E:46:99:1E:FC:10] -33dBm * V 802.11b/g/n     NingweisiPhone
  03: [CH 13] [06:25:E0:48:58:89] -85dBm * H 802.11b/g/n
  04: [CH 01] [1E:E8:29:E4:85:2E] -83dBm * H 802.11g/n
Starting WiFi scan...
```

esp8266 successfully connected.

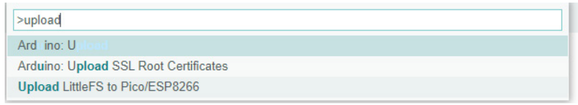
```
connected with NingweisiPhone, channel 6
dhcp client start...
.ip:172.20.10.5,mask:255.255.255.240,gw:172.20.10.1
.
Connected! IP address: 172.20.10.5
Open http://graph.local to open the graph page
HTTP server started
Please pull GPIO4 low (e.g. press button) to switch output mode:
 0 (OFF):    outputs are off and hidden from chart
 1 (AUTO):   outputs are rotated automatically every second
 2 (MANUAL): outputs can be toggled from the web page
pm open,type:2 0
```

Test #2 ESP8266, web server displays graph
Testing & troubleshooting

Troubleshoot #2
Web file missing

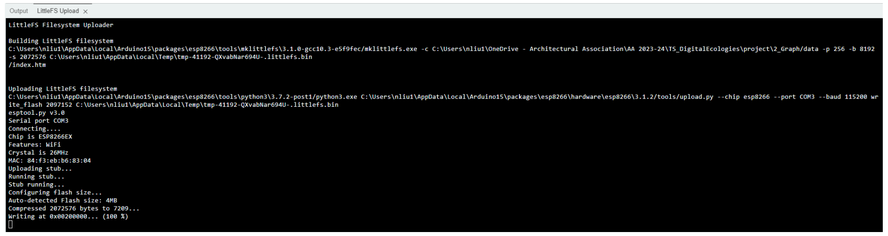



```
27  //#define USE_SPIFFS
28  #define USE_LITTLEFS
29  //#define USE_SDFS
```

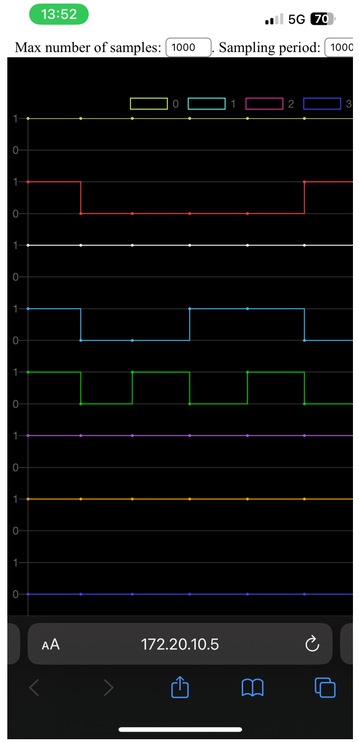


Reason & solution: when using LittleFS, the html file should be placed under the data folder in the sketch folder, then manually upload in Arduino IDE

Close Serial Monitor before uploading LittleFS.



LittleFS uploaded.
Then upload the code to esp8266 again.

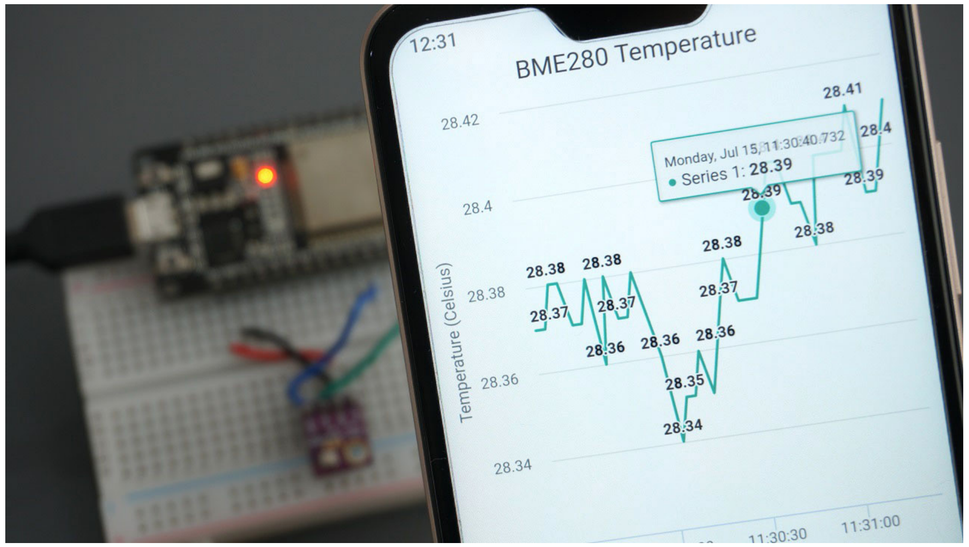


Successfully load the web from the example.
However, I'd like the graph to draw the data from the accelerometer, instead of showing the infomation of the esp8266 (which is what this web page is doing).

Test #3 ESP8266, web server displays graph

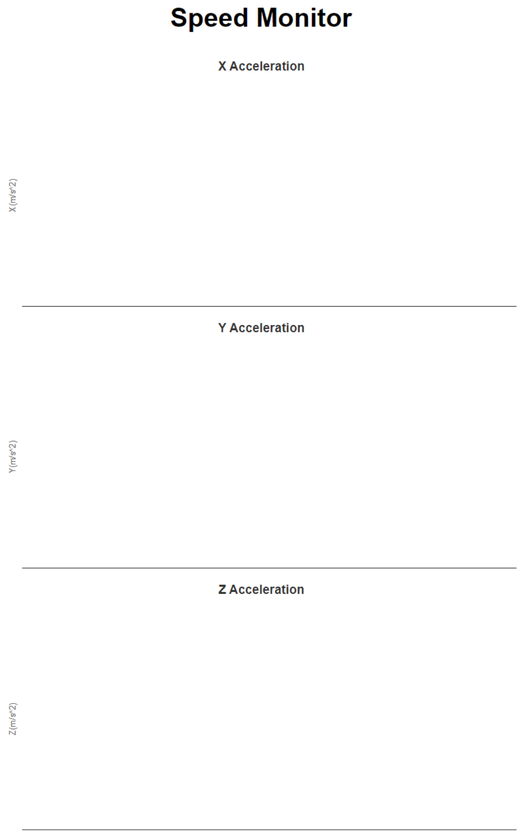
The Code in GitHub: 3_ReadSensorHighcharts
based on the webserver using BME280 and ESP8266, replacing the html file to display the acceleration.
The web is using highcharts library to draw the line charts.

The example using BME280 sensor: https://randomnerdtutorials.com/esp32-esp8266-plot-chart-web-server/
The example was uploaded: 3_Example_BME280



(from the example)
The goal if this test is to display a similar line chart with the accelerometer.

The web page was edited with p5 editor:



Issue here: the webpage doesn't load

The ska8 Monitor, installation