Lab 2: Sensors

Learning outcome: Understand how to access on-board sensors, and use serial communication to transfer data

# Introduction

## Lab overview

In this lab you will learn how to interface with environmental sensors using the DISCO-L475VG-IOT01A board and Mbed API.

The DISCO-L475VG-IOT01A device comes with on-board sensors, including a pressure sensor (LPS22HB), 3-axis accelerometer and 3-axis gyroscope (LSM6DSL), a 3-axis magnetometer (LIS3MDL), a humidity sensor (HTS221), a capacitive digital sensor for relative humidity and temperature (HTS221), a Time-of-Flight and gesture-detection sensor (VL53L0X), and two digital omnidirectional microphones (MP34DT01).

In this lab, we will read data from the sensors, transmit them to the PC via UART, and display the readings on a terminal emulator.

# Requirements

## Software functions

The functions that may be used in this lab are listed below:

**UART functions**

You will be using the BufferedSerial API to communicate with the PC.

**On-board Sensor functions**

**Initialization of Sensors**

*HTS221* Temperature BSP\_TSENSOR\_Init()

Humidity BSP\_HSENSOR\_Init()

*LPS22HB* Pressure BSP\_PSENSOR\_Init()

*LIS3MDL* Magnetometer BSP\_MAGNETO\_Init()

*LSM6DSL* Accelerometer BSP\_ACCELERO\_Init()

Gyroscope BSP\_GYRO\_Init()

**Reading Data from sensors**

*HTS221* float BSP\_TSENSOR\_ReadTemp(void)

Get the temperature reading from the HTS221 sensor.

float BSP\_HSENSOR\_ReadHumidity(void)

Get the humidity reading from the HTS221 sensor.

*LPS22HB* float BSP\_PSENSOR\_ReadPressure(void)

Get the pressure reading from the LPS25H sensor.

*LIS3MDL* void BSP\_MAGNETO\_GetXYZ(int16\_t \*pDataXYZ)

Get the 3-axis values of the magnetometer.

void BSP\_MAGNETO\_LowPower(uint16\_t status)

Activates low power mode in the magnetometer.

*LSM6DSL* void BSP\_GYRO\_GetXYZ(float\* pfData)

Get the 3-axis values of the gyroscope.

void BSP\_GYRO\_LowPower(uint16\_t status)

Activates low power mode in the gyroscope.

void BSP\_ACCELERO\_AccGetXYZ(int16\_t \*pDataXYZ)

Get the 3-axis values of the accelerometer.

void BSP\_ACCELERO\_LowPower(uint16\_t status)

Activates low power mode in the accelerometer.

# Getting Set-up

You will need to make sure you’ve included the board support files in your project, so that you can access the sensor APIs. You will also need to enable floating-point support in printf output.

The board support files for the board we’re using are here:

<https://os.mbed.com/teams/ST/code/BSP_B-L475E-IOT01/>

To add this to your Mbed Studio project go to File > Add Library to Active Program then paste the URL and select the default/master branch.

To enable floating-point support, you need to enable the minimal-printf-enable-64-bit and minimal-printf-enable-floating-point settings in the mbed\_lib.json file.

# Application Code

In this lab exercise we will write a program that reads the temperature, humidity, pressure, magnetometer, accelerometer, and gyroscope values from the sensors on-board, every three seconds. The temperature is measured in degrees Celsius but we will convert it to both Fahrenheit and Kelvins.

The program will then enter sleep mode and wait for interrupts. At the same time the program will blink a LED every second to show it's still responsive. We will inspect the sensor readings via the serial debug interface.

## Program structure

Here is an overview of how we will structure our program:

1. Global variables
   * A BufferedSerial object for communicating with the PC
   * A DigitalOut object for the LED.
   * Two LowPowerTicker objects for tracking events:
     1. One for flashing the LED every second
     2. One for triggering sensor readings
2. Initialisation
   * Initialize sensors.
   * Print a welcome message.
   * Activate tickers.
3. Ticker Handlers
   * Toggle the LED and update the measurements.
   * Set a flag that indicates that the measurements need to be read and displayed again.
4. Main function
   * In a loop:
     1. Check if the flag is set.
     2. Read from the sensors.
     3. Convert the temperature into Fahrenheit and Kelvins.
     4. Write the output to the UART.
     5. Enter sleep mode.

### Global Variables

Start by creating global variables for the LED and the serial port, and override the console device so we can use printf.

| static DigitalOut led(LED1);  static BufferedSerial serial\_port(USBTX, USBRX, 9600);  FileHandle \*mbed::mbed\_override\_console(int fd)  {  return &serial\_port;  } |
| --- |

### Initialisation

Turn the LED on, and write a welcome message over the serial port. Test your program at this point to make sure it’s working.

The code might look like this:

| int main()  {  led = true;  printf("Hello, world.\n");  return 0;  } |
| --- |

### Tickers

Now you’ve checked that you can compile your program, and can read from the serial port, create the ticker objects and hook them up to functions that will run each time the ticker ticks.

**Global variables:**

| static LowPowerTicker ledTicker;  static LowPowerTicker readoutTicker;  static bool shouldReadSensors; |
| --- |

**Handlers:**

| static void ledTick()  {  led = !led;  }  static void readoutTick()  {  shouldReadSensors = true;  } |
| --- |

**Initialisation code:**

| ledTicker.attach(&ledTick, 1s);  readoutTicker.attach(&readoutTick, 3s); |
| --- |

At this point, you should compile and run your program again. The LED should toggle its state every second.

### Reading from sensors

Now, we can read and display sensor values. To do this, the sensors must be initialised, so create a function to do that, and call it from main:

| static void initialiseSensors()  {  BSP\_TSENSOR\_Init();  BSP\_HSENSOR\_Init();  BSP\_PSENSOR\_Init();  BSP\_MAGNETO\_Init();  BSP\_ACCELERO\_Init();  BSP\_GYRO\_Init();  } |
| --- |

| int main()  {  ...  initialiseSensors();  ...  } |
| --- |

And finally, create a function to read the sensors and print out their values:

| static void readSensors()  {  printf("Sensor values:\n");  float temp = BSP\_TSENSOR\_ReadTemp();  printf("\* Temperature: %f C, %f F, %f K\n",  temp,  convertCelsiusToFahrenheit(temp),  convertCelsiusToKelvin(temp));  ...  } |
| --- |

You will need to implement the temperature conversion functions yourself, and you will need to display sensor data from the other sensors.

Now, in your main loop, check to see if shouldReadSensors is true, and if it is, call your readSensors function, and clear the flag:

| while (true) {  sleep();  if (shouldReadSensors) {  readSensors();  shouldReadSensors = false;  }  } |
| --- |

## Expected Output

When your program is running, you should see output like the following:

| Hello, world.  Sensor values:  \* Temperature: 24.369190 C, 75.864540 F, 297.519196 K  \* Humidity: 53.434387  \* Pressure: 1000.169983  \* Magneto: 80 -433 221  \* Gyro: 0.000000 -1120.000000 770.000000  \* Accelerometer: 5 -1 1011 |
| --- |

(Your sensor values will be different)