***Internet of Things Course***

**Final Project**

**Wearable Activity Tracker**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc36129233)

[2 Requirements 1](#_Toc36129234)

[3 Tasks 2](#_Toc36129235)

[3.1 Overview 2](#_Toc36129236)

[3.2 Embedded System Development 2](#_Toc36129237)

[3.3 Machine Learning Models 3](#_Toc36129238)

[3.4 Back-end Design 4](#_Toc36129239)

[3.5 Mobile Development 4](#_Toc36129240)

# Introduction

In this final lab, you will design a full-stack Internet of Things (IoT) system for real-time classification of user activities. You will use an IoT board with accelerometer and gyroscope sensors, a cloud platform where you can instantiate a virtual machine (e.g., Google Cloud Platform), and an Android mobile phone. The project involves a range of independent tasks across multiple platforms, including

* Developing firmware for an embedded system
* Collecting readings for different activities (walking, running, etc.)
* Training and deploying appropriate machine learning algorithms for motion detection
* Developing back-end web interfaces and associated databases
* Designing robust network communication schemes
* Mobile app development

At the end of this lab, you should have a working prototype of a system that can classify different activities with negligible latency and display the results on the mobile user interface.

# Requirements

In this lab, you will be using the following hardware and platforms:

* **A DISCO-L475VG-IOT01A board with embedded gyroscope, accelerometer, and Wi-Fi transceiver**
* **A device management platform**
* **A cloud platform (e.g., the Google Cloud Platform - GCP)**
* **An Android smartphone (e.g., Motorola Moto G7)**

There are no specific restrictions in terms of the software (web servers, databases, machine learning frameworks, and communication schemes) you should use, so as to allow sufficient flexibility in the project design, as well as room for creativity. However, we list below several popular options that you may wish to consider:

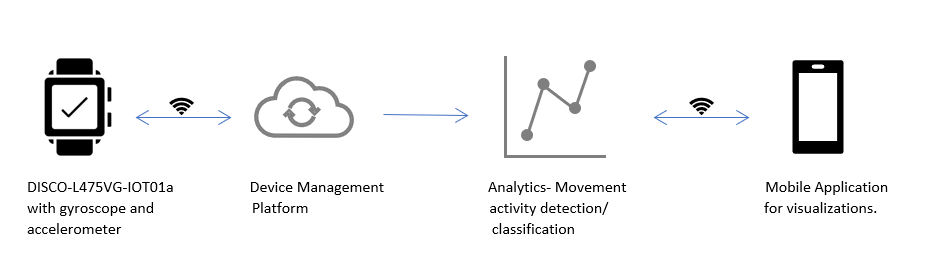
* Web servers: Django, Flask, Node.js, PHP
* Databases: MySQL, SQLite, SQL server, Firebase
* Machine learning framework: TensorFlow, PyTorch, MXNet

Please be aware that you are free to choose other software solutions as long as they can help you achieve the tasks described below.

# Tasks

## Overview

The Figure 1 below gives a high-level illustration of the IoT system you will develop. The DISCO-L475VG-IOT01A board functions as a wearable device, which can acquire sensor readings and periodically send them to the device management platform, via a Wi-Fi connection. The cloud service acquires the reading updates and infers the current user activity based on a pre-trained machine learning model. Finally, the user can view corresponding statistics in a mobile app, by querying web APIs.



*Figure 1. An illustration of the main components of the IoT system*

The assignment is broken down into several tasks, as follows:

1. Embedded system development and communications with a device management platform.
2. Machine learning model selection, training, and deployment
3. Back-end design
4. Mobile app development

The four tasks are described in detail in the rest of this section.

## Embedded System Development

In this assignment, we are using again the DISCO-L475VG-IOT01A board, which is compatible with Mbed OS.

The example project in your online compiler includes a Wi-Fi library that allows your IoT board to connect to a personal Wi-Fi hotspot. However, in order to obtain readings from the sensors, you need to identify the chip module on your board and import the associated libraries into your compiler. You can find the libraries relevant to your sensors at <https://os.mbed.com/code/>.

Similarly, the example project shows how to connect and send data from the board to the device management platform:

M2MObjectList m2m\_obj\_list;

// GET resource 3000/0/5701

m2m\_get\_res = M2MInterfaceFactory::create\_resource(m2m\_obj\_list, 3000, 0, 5701, M2MResourceInstance::INTEGER, M2MBase::GET\_ALLOWED);

cloud\_client->add\_objects(m2m\_obj\_list);

m2m\_get\_res->set\_value();

This code snippet binds a resource (any value) with a web path /3000/0/5701. The path can be specified randomly. Any update of the resource can be pushed to the device management platform through M2M protocols, by calling the function set\_value(), and you can view the updates simultaneously

In this project, the resources of which you will make use are readings from the gyroscope and accelerometer. You should be careful about how many resources you need to create, from the perspective of both communication overhead and API usage.

## Machine Learning Models

To achieve activity recognition, you will leverage the power of machine learning. You may use the following dataset on Kaggle: <https://www.kaggle.com/vmalyi/run-or-walk>, which contains previously collected sensor data from accelerometers and gyroscopes. The documentation therein explains how the data was collected and what model was used for training.

You are strongly encouraged to read through the posts, in case you might hope to collect additional data later, but also if you wish to enhance the robustness of your model or diversify the detection scenarios. Even if you do not want to do so, the original posts contain a key hint about the measurement unit used in representing sensor data, which is not recorded explicitly in the dataset.

However, you are not advised to use the same model as proposed by the respective author. You may find that his method did/could not make use of readings from all axes. Here comes one important point: you should do your best to make sure that your model can generalize well in multiple rigorously testing scenarios. To achieve this, you need to make your model learn as much useful information as possible.

In terms of model design, you may consider the following aspect, to ensure your model achieves good performance:

1. Make use of temporal information. Clearly, motion detection is a pattern recognition task on time-series multi-dimensional data. In other words, the task is time-sensitive, so you need to ensure that the model can learn temporal correlations from the dataset. For some time-sensitive tasks (even for this one), it might be possible that a time-insensitive model can exhibit reasonable performance. However, be aware that if your model does not extract/learn temporal correlations from the dataset, you may only gain limited insights.
2. Keep things lightweight. Activity tracking is not a difficult learning task that would require thousands of parameters and days of training. When designing a model, you should start from the simplest solution and gradually add blocks to it, rather than going directly for a sophisticated deep learning model. Even though all the computation can be achieved in the cloud, you should not use more than you need. Different models by default would use different resources. You can think of the following questions before you start: Is it better to choose a sophisticated model that should achieve excellent performance, but has many parameters and requires a lot of data for training, or choose a model with fewer parameters that can be training in one batch? Do you value more the computation resources or memory resources? Do you want your model to give probabilistic interpretations or not?

## Back-end Design

For the communications between the device and device management platform, it is important to decide an appropriate frequency as well as the sampling rate. Once every 20 seconds would incur large latency and you may potentially lose important temporal information while reading 100 times per second will generate lots of traffic and maybe prove unnecessary. Thus, you would need to do some exploratory experiments first.

Once you have trained your machine learning model, you need to develop a web server that

1. Gives predictions periodically, based on the readings from the device management platform.
2. Stores predictions in the Cloud (in a database).
3. Create a web API to be called when the user wishes to see the real-time predictions.
4. Create a web API to be called when the user wishes to see their historical activity records.

Of course, you can add more functionality to both your server logic and mobile design for better usability. As explained above, the web server needs to work on at least three tasks simultaneously:

1. Listening for and replying to HTTP requests;
2. Waiting for new readings and making predictions based on these; and
3. Performing database operations.

You need to carefully design the logic of the server to prevent unexpected errors, such as data overwriting and deadlocks, to name just a few.

## Mobile App Development

You should use an Android smartphone on which you should develop a mobile app to display relevant information retrieved from the cloud. Android Studio is the most accessible tool for developing an app and you are free to choose Java or Kotlin as the programming language. If you are more familiar with JavaScript, there are a number of JS mobile frameworks that you can also opt for, such as Cordova and NativeScript.

If you are using Android Studio, the recommendation is to use Volley, an HTTP library developed by Google, in order to communicate with the cloud service. More information about Volley can be found at <https://developer.android.com/training/volley>. Android strictly discourages I/O-intensive operations, such as networking, from running in the UI thread (that is the main thread). Normally, HTTP requests are wrapped in an Asyntask class and a callback function needs to be defined, which hugely affects productivity. Volley provides convenient APIs for various HTTP requests that can substantially reduce the amount of code.

There are no specific requirements in terms of the design of the mobile app, but in general it should have the following functionalities:

1. Display the real-time predictions retrieved from the cloud;
2. Display a history of activity records stored in the database in the cloud;
3. Ensure the user can interact with the app smoothly and the code is bug-free.

Attractive UIs and animations are also highly encouraged.