# IOT Application Development CA 1 – Report

## Introduction

The purpose of this CA was to create an IOT application that would support multi-threaded bi-directional message passing. The Raspberry PI(s) which would host these application(s) should read from at least four sensors and use the message passing to send the sensor readings to a message service (Dweepy, HiveMQ, WAI, etc). The IOT system should also have a client application which should receive the sensor readings, display them and be able to send a message back to the application on the RaspberryPI(s).

## Systems Architecture Diagram

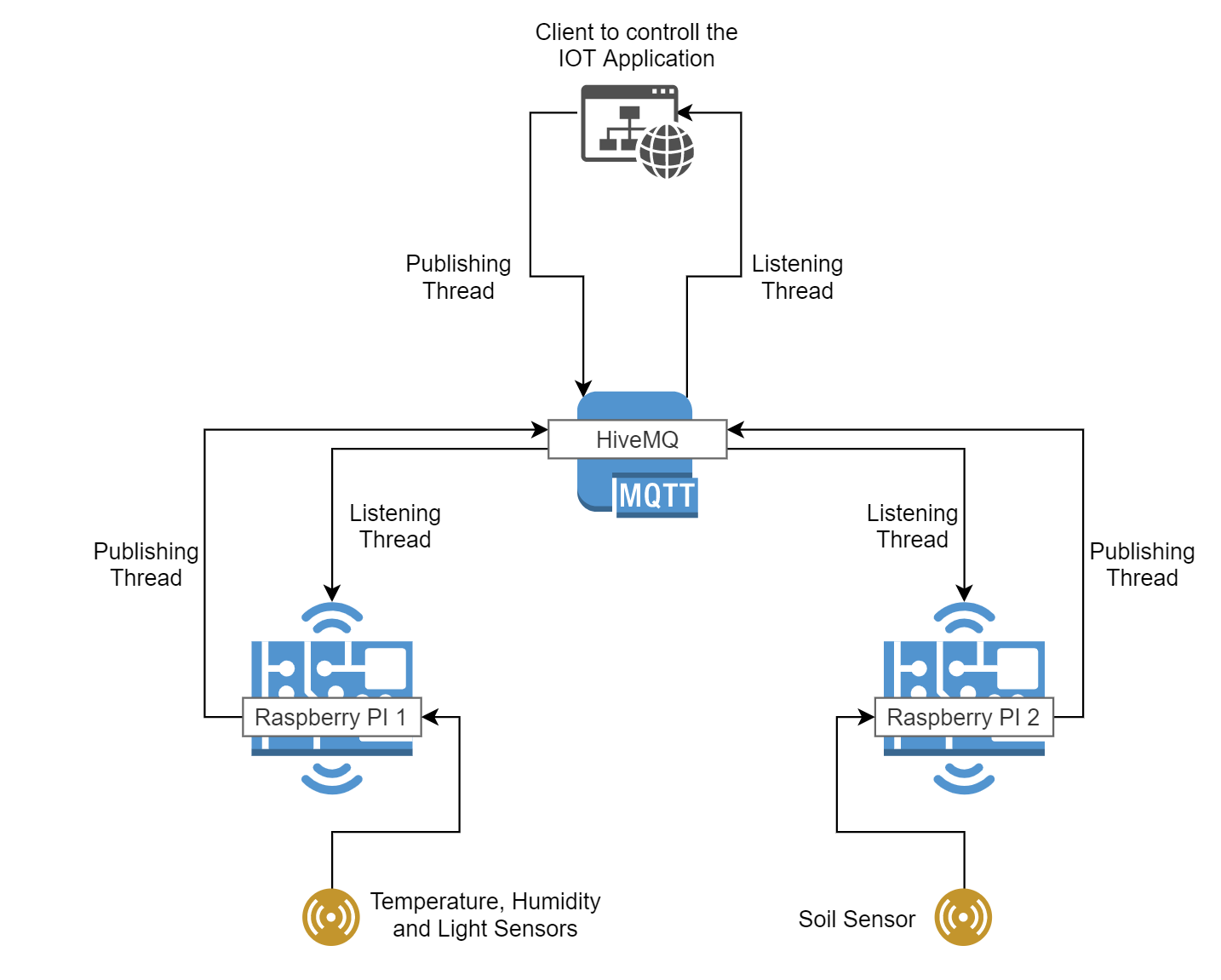


fig.1

## Implementation

### IOT Hardware

As depicted in fig.1 the project uses two separate Raspberry PIs which host each their own application. The first Raspberry PI (3 Model B) uses a GrovePI hat and compatible Temperature, Humidity and Light sensors to read data from its environment. The second Raspberry PI uses a soil probe that is directly connected to the RaspberryPI’s (3 Model B) GPIO on pin 21.

## RaspberryPI Software

The two Python (3.7.3) applications running on the Raspberry PIs use threading to create two Threads: one for listening, one for publishing.

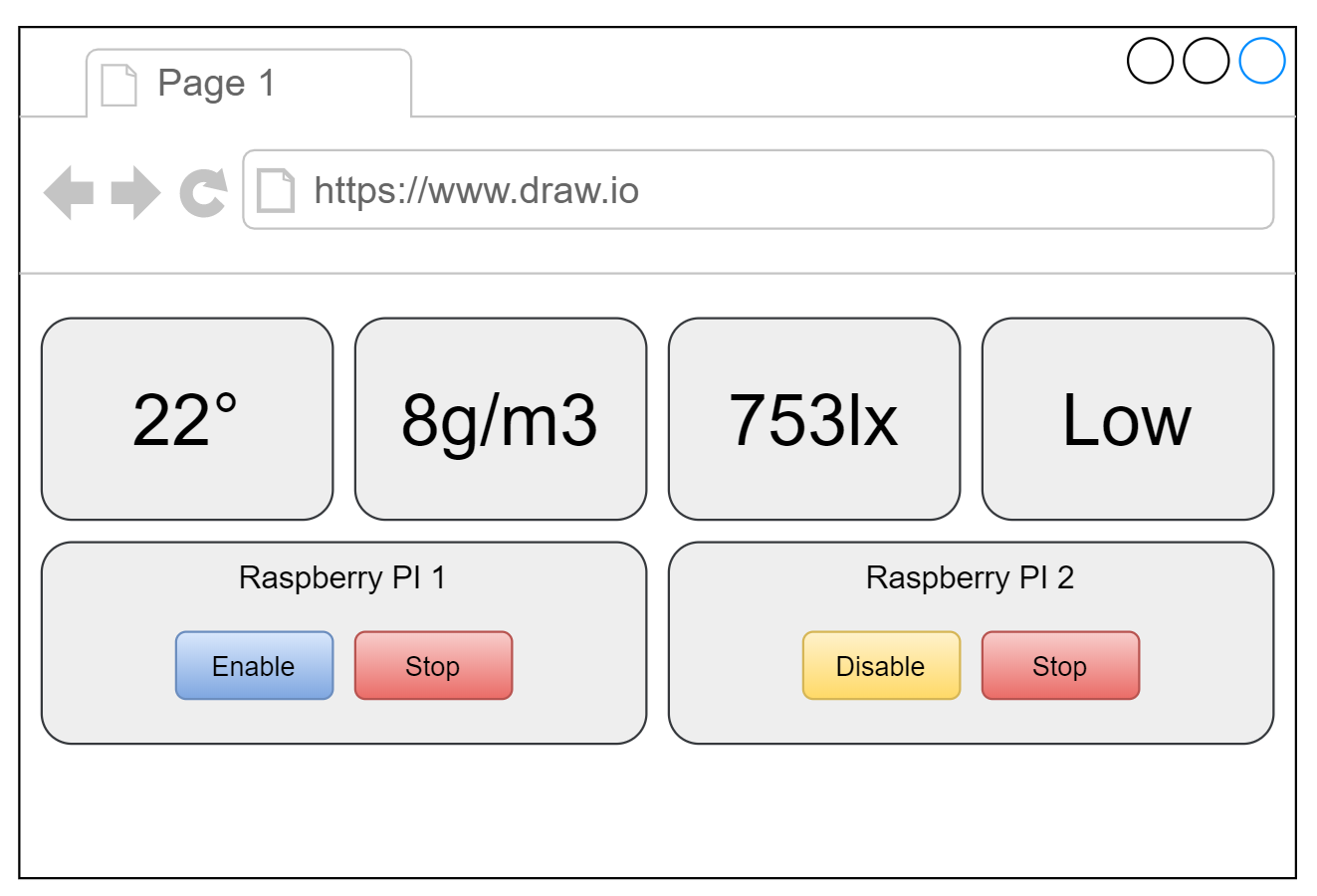
The listening thread starts and listens for a specific topic. Once it receives an input of {“publishing”: true} it starts the second, publishing thread.

The publishing thread, reads the sensor readings (temperature, humidity and light on PI1 and soil on P2), converts them to a dictionary and publishes the information on a separate topic on the Hive MQTT service.

Both Raspberry PIs use separate listening and publishing threads.

The Raspberry PI’s publishing thread can also be paused by sending a false value which will be re-enabled when receiving a true value or receive a value of {“terminate”: true} which will close the application completely.

## Client Application

fig.2

The client application is a simple HTML/CSS based website using Bootstrap, jQuery and the Eclipse Paho Javascript MQTT API

The client application is structured simple with the four readings displayed on the top and the two Raspberry PIs at the bottom. For both of the Raspberry PIs there is two buttons with the functionality to turn publishing off/on and to stop the Raspberry PI software.

The JavaScript file that powers the interactivity uses the Eclipse Paho Javascript MQTT API (<https://www.eclipse.org/paho/clients/js/>). It also stores the information in localStorage for quicker loading.

When the page is loaded the JS looks for the keys in the localstorage and populates the readings from the last available. At the same time it also connects to the broker with two separate clients (one for each publishing thread) and sets onMessageArrived callback functions to update the values on the page.

The JS also sets event listeners on the buttons to send messages to enable/disable the publishing threads and stop the Raspberry PI software.