Department of Electronic and Telecommunication Engineering

Internet of Things

EN3250

Currency Converter Service
Project Report

University of Moratuwa

170258L	: R.H.R. Jayarathne
170543G	: M.K.T. Sampath
170698J	: L.T.A. Wijayaratne
170375R	: D.R. Marasinghe

This report is submitted in partial fulfilment of the requirements for the module
EN3250 - Internet of Things

Contents

1 Introduction

1.1 Overview

Foreign exchange market is a global market, where people around the world buy and sell different foreign currencies everyday for various purposes. One purpose of the traders in the market is engaging in day-trading activities. Day trading is performed by normal people who intend to obtain profits and earn out of small price mismatches that are available for a short amount of time. Day traders' activity is influenced by information generated from past foreign currency price information. This information is generated by performing various kinds of mathematical transformations on time series price charts . These people find a lot of value in this information generated and they usually buy required market information from their trusted services.
This project mainly focuses on providing foreign exchange market day-traders, a selected set of useful information. This is achieved by using a currency API that provides live currency prices of various currencies around the world. Usually, hourly past currency price data are not available easily, because past currency data is usually provided on a daily basis. Due to this reason, storing the live currency data in the database is useful for making predictions. Users will also find the notification system that warns when the prices are beginning to exceed their expected boundaries, valuable.
ESP8266 is used for controlling the display as well as connecting to the user's mobile phone and Node-Red through WiFi technology. Node-Red to Node-MCU communication happens through the MQTT communication protocol. Node-MCU's ability to enter into sleep mode is also utilized. Node-MCU acts as both Wifi access point and http server. The user may sign-in using a username and a password, when the Node-MCU is functioning as a server. The access point mode is required when it is fetching and submitting data.

1.2 Objectives

- To provide several technical analysis charts such as Simple Moving Average
- to the day-trader so that they can make useful predictions
- To provide the useful information in a user friendly manner through the Node-Red dashboard as well as mobile interface.
- To notify and alarm the day-trader clients when their set upper and lower boundaries are exceeded so that they can take quick actions.
- To provide the user flexibility to choose the interested currencies for observation

1.3 Scope

The scope of the project includes an application using IOT concepts, tools and standards to provide timely information to Day Traders. This provides the users with the following.

1. Exchange rate of 6 currencies - updated every 15 seconds
2. Buzzer and Email notifications when set thresholds for exchange rates are exceeded
3. A dashboard displaying the variations of exchange rate and Technical indicators over time (updated hourly).

The basic components of which the application comprises are given below.

1. **A realtime database** (using Google Firestore)
2. **An open-source API** providing currency exchange rates
3. **A Node-Red Dashboard** to display variations of the 4 Technical Indicators
4. **A Buzzer** to notify the user when the set thresholds for exchange rates have been exceeded
5. **An LED Display** to show the exchange rate
6. **A Node-MCU** for accepting requests of the human users as well as to send control signals to a Buzzer and the LED Display

7. **A client mobile phone** as the user interface

The MQTT protocol is used to facilitate communication between Node MCU and the Node-red applications.

2 Methodology

2.1 Architecture

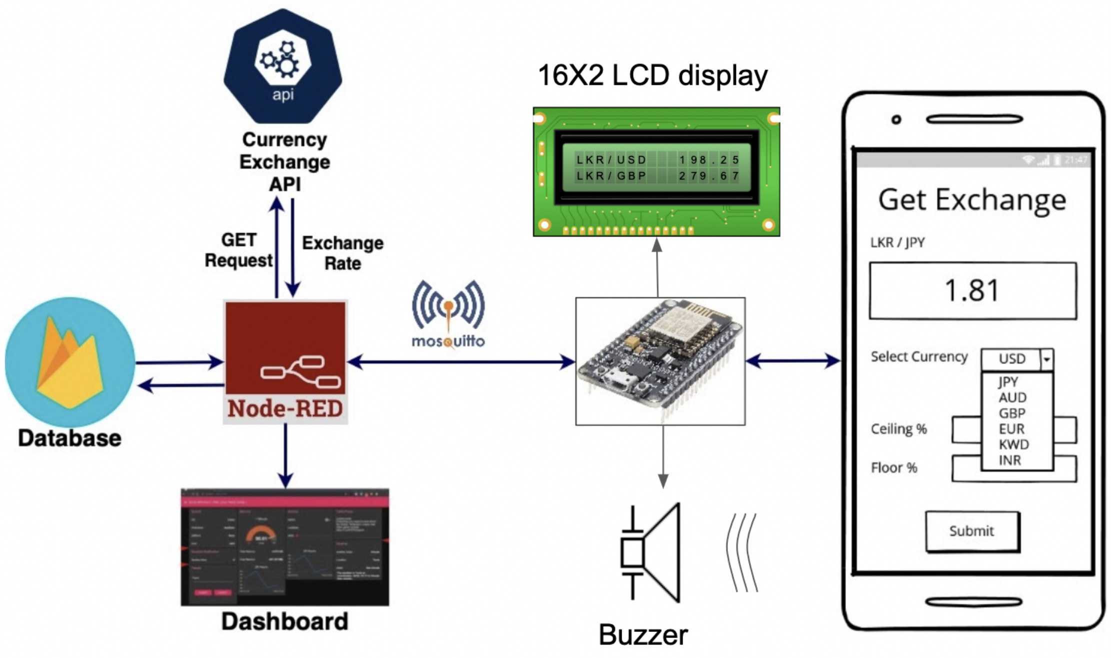

Figure 1: Architecture of the entire solution

2.2 Exchange API

2.3 Integrating Firebase

2.4 Technical analysis charts

The hourly price data collected and stored in the real-time database can be sent through popular technical analysis functions to generate useful charts. These charts can be used to make valuable inferences. To process the price data available, an external library named TechnicalIndicators was used. Using the library, following indicators are calculated for hourly historical currency rates. Each currency will have four technical analysis charts.

1. **Moving average lines** are frequently used to smooth the fluctuations in a price chart (or a chart of any time series). A moving average is simply the mean of the last n closing prices.
2. **Rate of Change oscillator** (ROC) or momentum oscillator is calculated as 100 times the difference between the latest closing price and the closing price n periods earlier. Thus, it oscillates around zero.
3. **Moving average convergence/divergence** (MACD) oscillators are drawn using exponentially smoothed moving averages, which place greater weight on more recent observations. The “MACD line” is the difference between two exponentially smoothed moving averages of the price.
4. **Relative Strength Index** (RSI) is based on the ratio of total price increases to total price decreases over a selected number of periods. This ratio is then scaled to oscillate between 0 and 100.

2.5 Creating the Node-Red dashboard

2.6 Configuring ESP8266

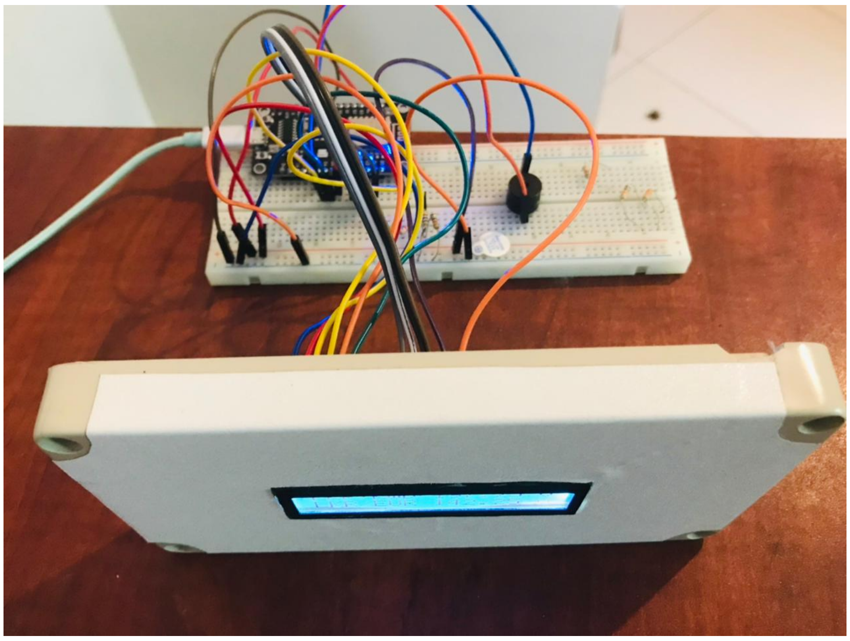

Figure 2: Setting up ESP8266 with an LCD display and a buzzer

The ESP8266 used in the IOT system performs multiple tasks to facilitate the expected functionality.

- Receiving data from Node-Red through MQTT protocol
- Decoding the compact data received to use them for the subsequent activity
- Actively sensing alerts issued by Node-Red
- Notifying the user with a buzzer sound when an alert is received
- Supporting and updating the display with currency price data sent from Node-Red
- Obtaining time and date from a NTP server to keep accurate time
- Checking whether the received notifications are expired or not
- Saving authentication results from the initial login in the EEPROM
- Acting as a server to provide a login page and a web interface to select ceiling/floor limits and view current prices.
- Sending authentication data and the data from the user interface to Node-Red through MQTT.

2.6.1 Setting up MQTT

When communicating through MQTT, ESP8266 subscribes to three MQTT topics.

```
client.subscribe("IOT_6B/G05/BuzzerNotification");
client.subscribe("IOT_6B/G05/CommonData");
client.subscribe("IOT_6B/G05/AuthResponse");
```

IOT_6B/G05/BuzzerNotification sends alerts whether the ceiling price (Upper limit of acceptable prices set by the user) or the floor price (Lower limit of acceptable prices set by the user). Given below is the format we adopted.

timestamp$user_email$currency$ceil_bit$floor_bit

Eg: 1623957326$first.lastname@gmail.com$USD$1$0

IOT_6B/G05/CommonData sends the regular currency value updates of the six currencies we have selected. Along with the message, the information whether these prices increased or decreased is also included. Given below is the format we adopted.

timestamp$USD$GBP$JPY$AUD$KWD$EUR$100101

Eg: 1623921792$200.25$155.25$1.82$143.25$197.25$142.25$101000

timestamp	: Unix epoch time when the data is sent
ceil_bit	: This is set to 1 when the ceiling price is crossed, otherwise set to 0
floor_bit	: This is set to 1 when the floor price is crossed, otherwise set to 0
001001	: Each bit represents the six currencies. 1 represents a recent rise in price, while 0 represents a recent reduction in price.

Combining the data like this was necessary, because a set of data should be available at a certain instance. Then the LCD display can be updated with new values at once. In addition, validating the timestamp of the data for expiration is easier when the data is combined. MQTT packets are received at random times, so receiving them together makes it easier to receive the data in an organized fashion.

2.6.2 Decoding compact received data

The goal here is to decode this large string and update each of them in suitable variables in the ESP8266 as given below. Each data item is separated with '$' marks.

1623921792$200.25$155.25$1.82$143.25$197.25$142.25$101000

timestamp = 1623921792, USD = 200.25 , GBP = 155.25, JPY = 1.82, AUD = 143.25, KWD = 197.25 , EUR = 142.25

When a subscribed topic receives a data item, the callback function is called. The function treats the data from two topics differently by calling additional functions process_notification and process_data.

```
void callback(char* topic, byte* payload, unsigned int length) {
  if (String(topic) == "IOT_6B/G05/BuzzerNotification") {
   process_notification(payload, length, 50, 5);
  }
  if (String(topic) == "IOT_6B/G05/CommonData") {
   process_data(payload, length, 70, 8);
  }
  :
  :
}
```

Let's take the process_notification function for example. It splits the byte array into character arrays and then saves them in the variables treating them as separate cases. Each time the callback function is called.

```
void process_notification(byte* payload, unsigned int length, int charlen, int
   numitem) {

    int digit;
    payloadstr = "";
    Serial.println();
    for (int i = 0; i < length; i++) {
      payloadstr += (char)payload[i];
    }

    char payloadstr_array[charlen];
    payloadstr.toCharArray(payloadstr_array, charlen);

    char * token = strtok(payloadstr_array, "$");

    for (int i = 1; i < numitem+1; i++) {
       switch (i) {
        case 1:
           timestamp = atol(token);
           Serial.print(timestamp);
           Serial.println();
           break;
  :
  :
        }
        token = strtok(NULL, "$");
    }
}
```

2.6.3 Sensing alerts and notifying the user

When an alert is sent through the IOT_6B/G05/BuzzerNotification topic, the callback function instantly identifies a crossing in ceiling or floor prices to activate the buzzer.
Checking whether the notifications are expired is essential to make sure the user is not alerted based on old information. ESP8266 was configured to have updated time and date. The timestamp attached to the notification is cross-checked against the unix epoch time available in the ESP8266. An example is given below.

```
if (timestamp > unix_epoch - 19820) {
    current_user = String(token);
}
```

2.6.4 Supporting the LCD 16x2 display

LCD display is used to display the live currency prices, along with the date and time. These values are displayed one by one sequentially.
The time and date was obtained using a readily available library for ESP8266 that enables the ESP8266 to communicate with an NTP (Network Time Protocol) Server.

3 Conclusion

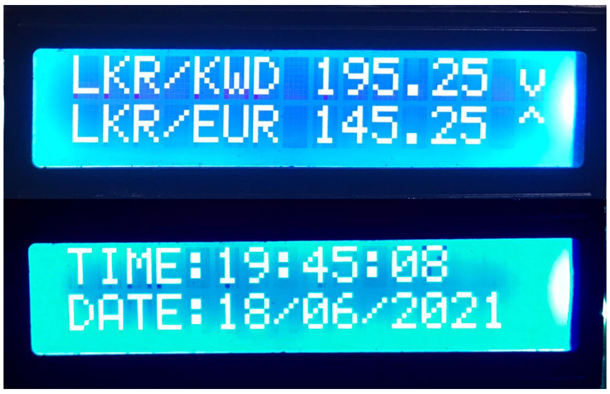

Figure 3: LCD display showing currency values, time and date