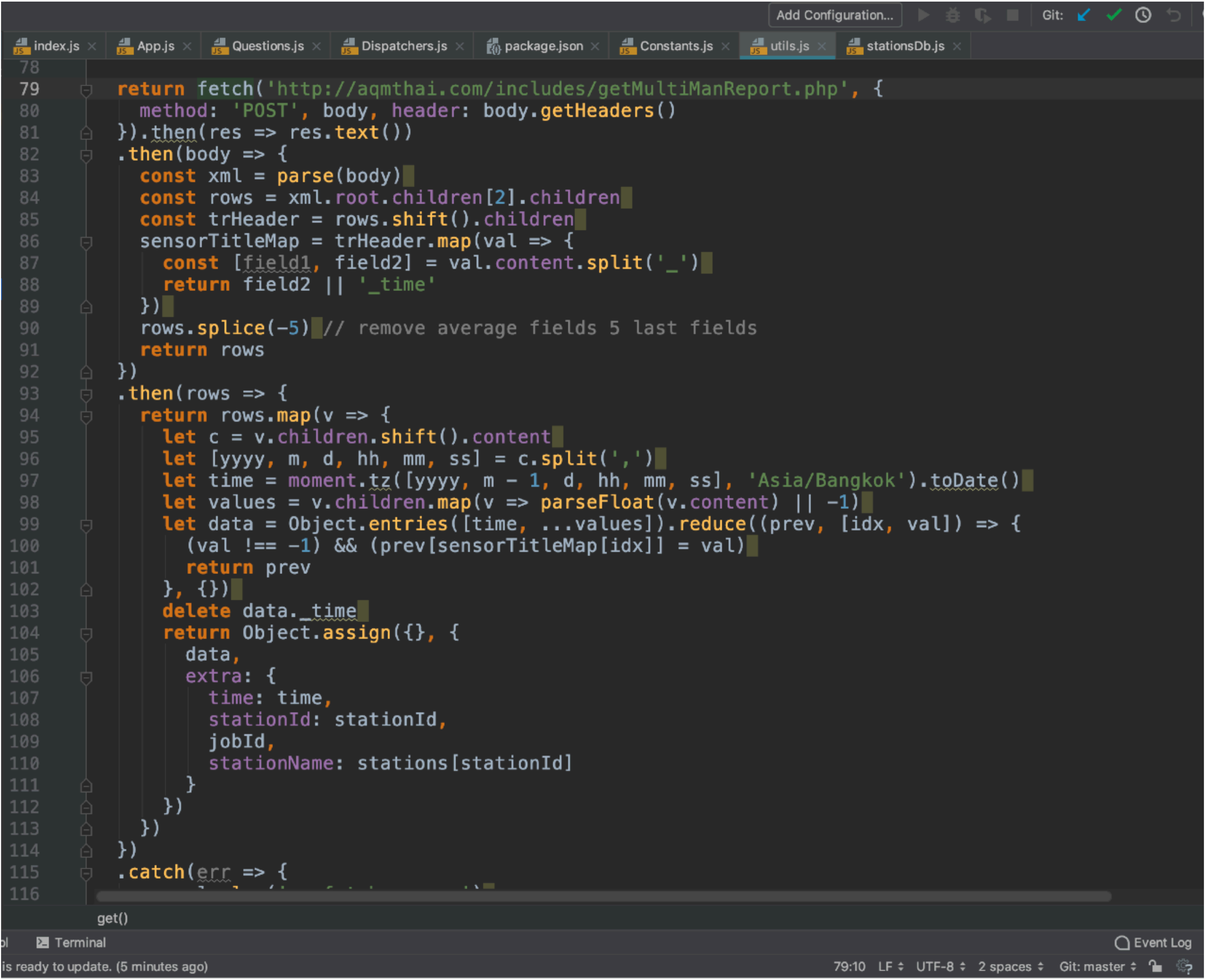
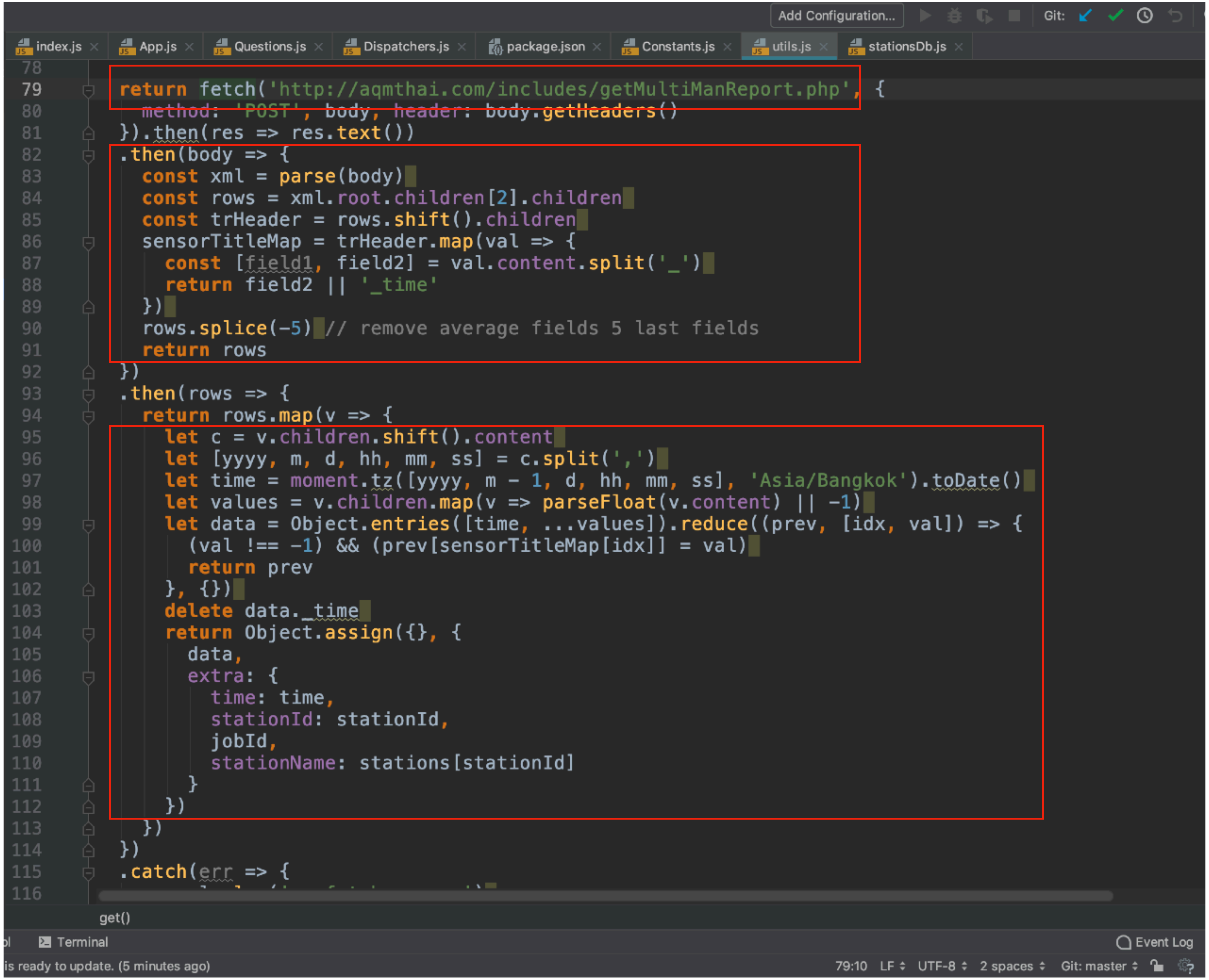


```
return fetch('http://aqmthai.com/includes/getMultiManReport.php', {
  method: 'POST', body, header: body.getHeaders()
}).then(res => res.text())
.then(body => {
  const xml = parse(body)
  const rows = xml.root.children[2].children
  const trHeader = rows.shift().children
  sensorTitleMap = trHeader.map(val => {
    const [field1, field2] = val.content.split('_')
    return field2 || '_time'
  })
  rows.splice(-5) // remove average fields 5 last fields
  return rows
})
.then(rows => {
  return rows.map(v => {
    let c = v.children.shift().content
    let [yyyy, m, d, hh, mm, ss] = c.split(',')
    let time = moment.tz([yyyy, m - 1, d, hh, mm, ss], 'Asia/Bangkok').toDate()
    let values = v.children.map(v => parseFloat(v.content) || -1)
    let data = Object.entries([time, ...values]).reduce((prev, [idx, val]) => {
      (val !== -1) && (prev[sensorTitleMap[idx]] = val)
      return prev
    }, {})
    delete data._time
    return Object.assign({}, {
      data,
      extra: {
        time: time,
        stationId: stationId,
        jobId,
        stationName: stations[stationId]
      }
    })
  })
})
.catch(err => {
```



```javascript
return fetch('http://aqmthai.com/includes/getMultiManReport.php', {
  method: 'POST', body, header: body.getHeaders()
}).then(res => res.text())
.then(body => {
  const xml = parse(body)
  const rows = xml.root.children[2].children
  const trHeader = rows.shift().children
  sensorTitleMap = trHeader.map(val => {
    const [field1, field2] = val.content.split('_')
    return field2 || '_time'
  })
  rows.splice(-5)// remove average fields 5 last fields
  return rows
})
.then(rows => {
  return rows.map(v => {
    let c = v.children.shift().content
    let [yyyy, m, d, hh, mm, ss] = c.split(',')
    let time = moment.tz([yyyy, m - 1, d, hh, mm, ss], 'Asia/Bangkok').toDate()
    let values = v.children.map(v => parseFloat(v.content) || -1)
    let data = Object.entries([time, ...values]).reduce((prev, [idx, val]) => {
      (val !== -1) && (prev[sensorTitleMap[idx]] = val)
      return prev
    }, {})
    delete data._time
    return Object.assign({}, {
      data,
      extra: {
        time: time,
        stationId: stationId,
        jobId,
        stationName: stations[stationId]
      }
    })
  })
})
.catch(err => {
```

influxdb