

IoT project using Matlab sensors and ThingSpeak platform

1. Introduction

MATLAB is a programming and numerical computing platform, used by millions of engineers and scientists, in order to analyze data, develop algorithms, and create models and applications.

The platform ThingSpeak is an IoT analytics service that allow us to collect, visualize, and analyze real-time data streams in the cloud. When we send data to ThingSpeak from our devices, ThingSpeak instantly visualizes it. Using both MATLAB code and ThingSpeak, we can perform online analysis and process data as it comes in. Besides, from any internet-connected device directly to ThingSpeak we can send data using a Rest API or MQTT.

The term "Internet of Things" (IoT) refers to a new trend in which a significant number of embedded devices (or "things") are linked to the Internet. These interconnected gadgets exchange data with other devices and people, and frequently send sensor data to cloud storage and cloud computing resources for processing and analysis to produce significant insights.

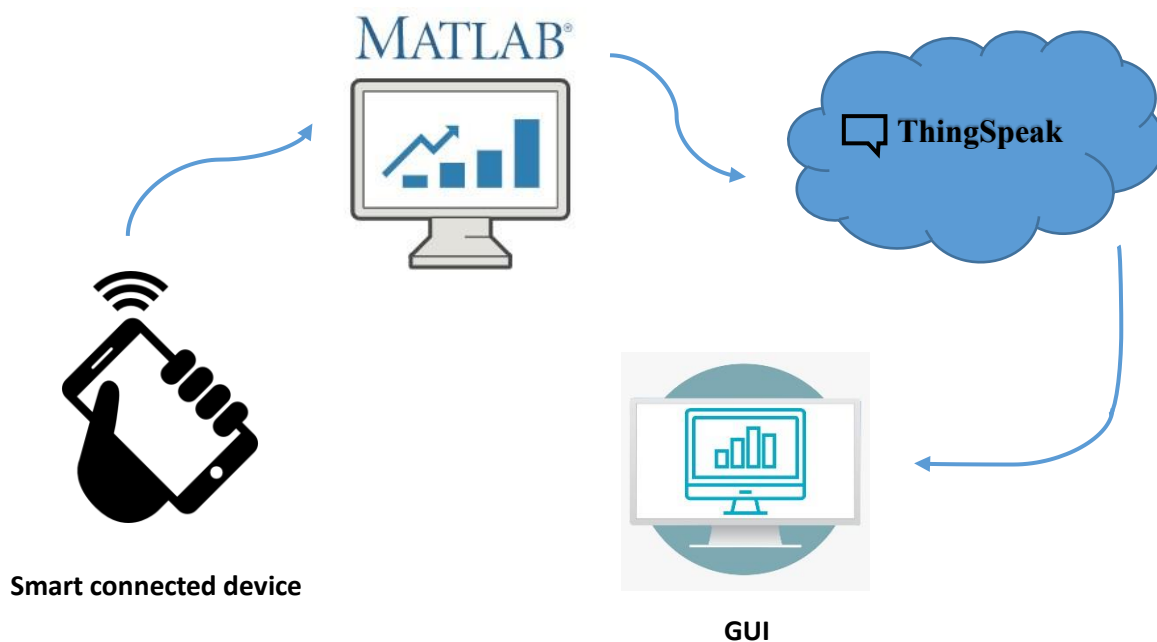


Fig.1 Block scheme of the project

Basically, our project retrieves information from Matlab Mobile Sensors, upload the acquired data from MATLAB Mobile to ThingSpeak Channel, then stream the data from the cloud to our Graphical User Interface.

On the left, we have the smart device (the “things” in IoT) that live at the edge of the network. This device collects data and includes the sensors used by us.

In the middle, we have the MATLAB application and to connect to sensors on the device and collect data, we create a *mobiledev* object in MATLAB. Also, here we have the cloud where data from the source is aggregated and analyzed in real time, by ThingSpeak.

Finally, the data is pulled from the IoT platform into a desktop software environment, in our case, the interface created in AppDesigner.

An IoT system includes all these elements. ThingSpeak fits in the cloud part of the diagram and provides a platform to quickly collect and analyze data from internet connected sensors.

2. Implementation

2.1 Sensor data acquisition

We stream data from our mobile device sensors to a MATLAB session on a computer.

MATLAB Mobile sends the data in real-time with a WiFi connection to MATLAB running in the MathWorks® Cloud.

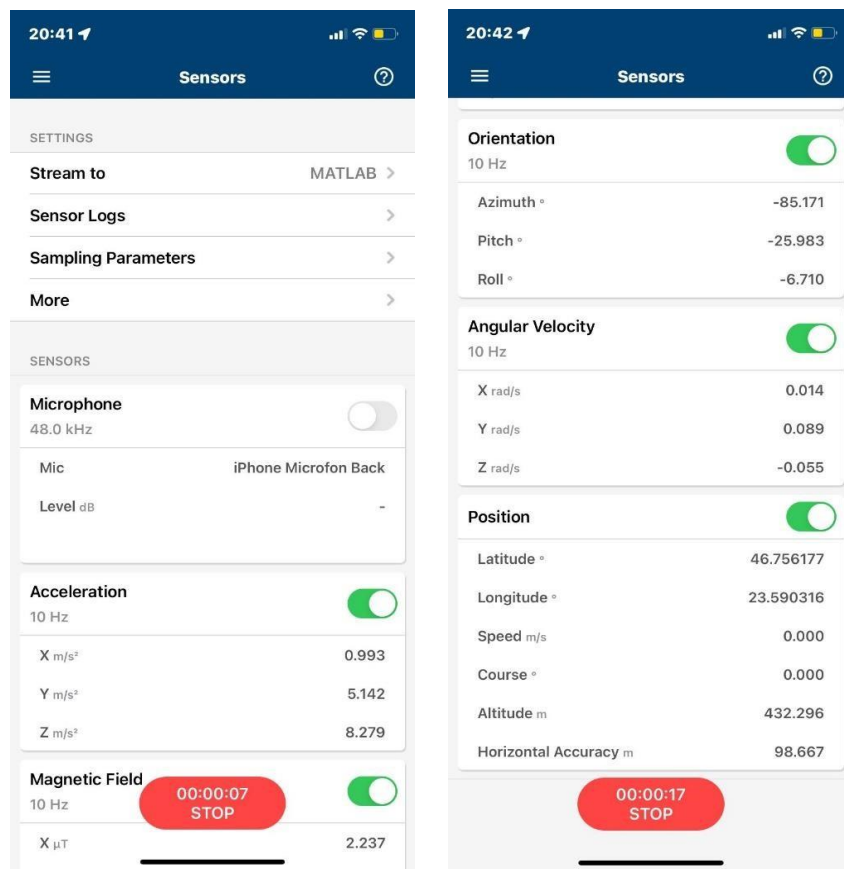
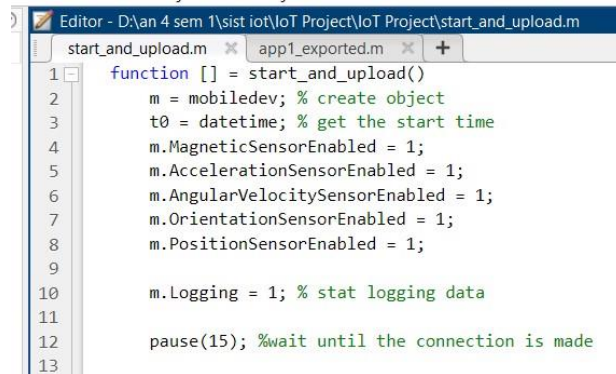


Fig.2 Matlab Mobile Sensors

To start the data acquisition, we create an object, and we enable the sensors:



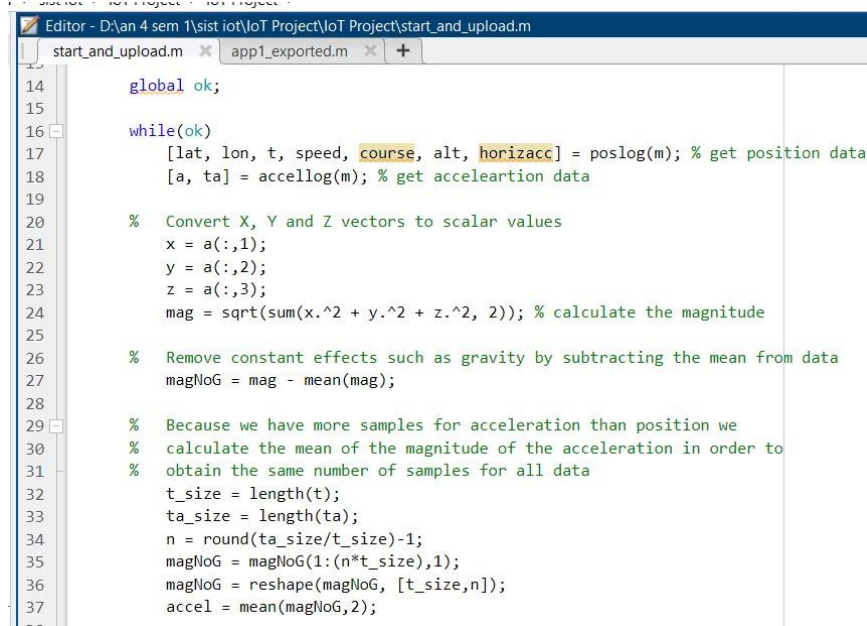
```

1 function [] = start_and_upload()
2     m = mobiledev; % create object
3     t0 = datetime; % get the start time
4     m.MagneticSensorEnabled = 1;
5     m.AccelerationSensorEnabled = 1;
6     m.AngularVelocitySensorEnabled = 1;
7     m.OrientationSensorEnabled = 1;
8     m.PositionSensorEnabled = 1;
9
10    m.Logging = 1; % stat logging data
11
12    pause(15); %wait until the connection is made
13

```

Fig.3 Matlab code.1

To read continuously data from the sensors we use a while loop:



```

14     global ok;
15
16     while(ok)
17         [lat, lon, t, speed, course, alt, horizacc] = poslog(m); % get position data
18         [a, ta] = accellog(m); % get acceleration data
19
20         % Convert X, Y and Z vectors to scalar values
21         x = a(:,1);
22         y = a(:,2);
23         z = a(:,3);
24         mag = sqrt(sum(x.^2 + y.^2 + z.^2, 2)); % calculate the magnitude
25
26         % Remove constant effects such as gravity by subtracting the mean from data
27         magNoG = mag - mean(mag);
28
29         % Because we have more samples for acceleration than position we
30         % calculate the mean of the magnitude of the acceleration in order to
31         % obtain the same number of samples for all data
32         t_size = length(t);
33         ta_size = length(ta);
34         n = round(ta_size/t_size)-1;
35         magNoG = magNoG(1:(n*t_size)+1);
36         magNoG = reshape(magNoG, [t_size,n]);
37         accel = mean(magNoG,2);
38

```

Fig.4 Matlab code.2

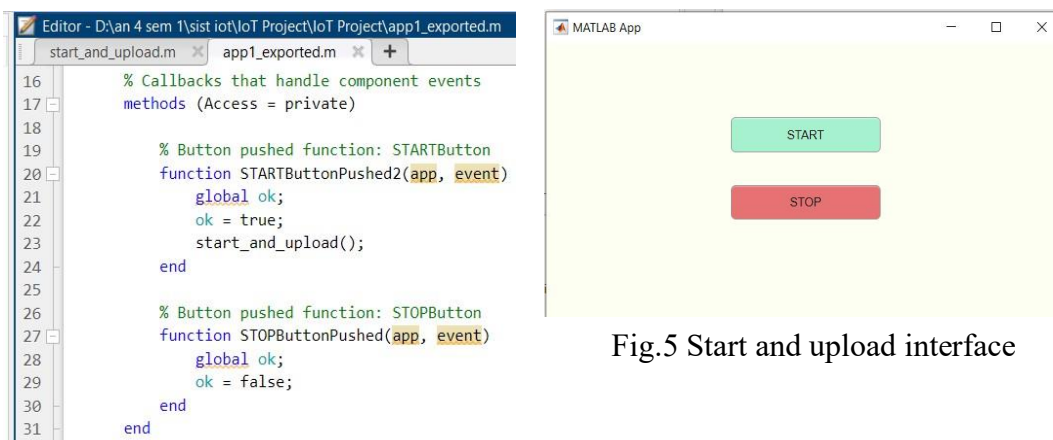


Fig.5 Start and upload interface

2.2 Transmit the acquired sensor data to cloud platform

In order to make the sensors start up more efficient, we created a small interface:

We transmit the acquired sensor data using the Matlab function: *thingSpeakWrite* (*channelID*, *data*, *'WriteKey'*, *'channel write API key'*), which writes the data to the specified channel. The Write API key is specified as a comma-separated pair consisting of *'WriteKey'* and a character vector or string representing the channel write key.

The screenshot shows the ThingSpeak IoT project page for 'IoT project'. It displays the Channel ID: 1951496, Author: mwa000025790675, and Access: Private. Below this, there are tabs for Private View, Public View, Channel Settings, Sharing, API Keys, and Data. The 'API Keys' tab is selected, showing the 'Write API Key' section with a key: 11KP850VPM3KC2II and a 'Generate New Write API Key' button. Below that, the 'Read API Keys' section shows a key: 3DT0QXNG1B2NVZD1.

Fig.6 ThingSpeak Channel ID and API Key

```
38
39 % Send data to thingSpeak
40 thingSpeakWrite(1951496,[lat, lon, alt, speed, accel], 'WriteKey', '11KP850VPM3KC2II', 'Timestamp', t0+t/24/60/60)
41 % Because of limitations of thingSpeak we cannot write continuously
42 % => set a pause of 15 seconds
43 pause(15);
44 end
```

Fig.7 Transmit the acquired sensor data to ThingSpeak

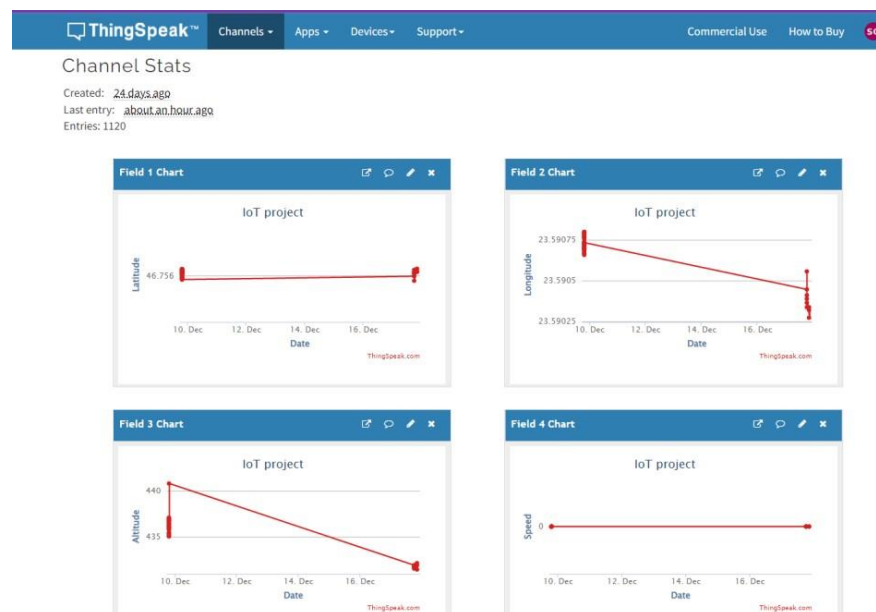


Fig.8 ThingSpeak Channel

2.3 Realize graphical interface

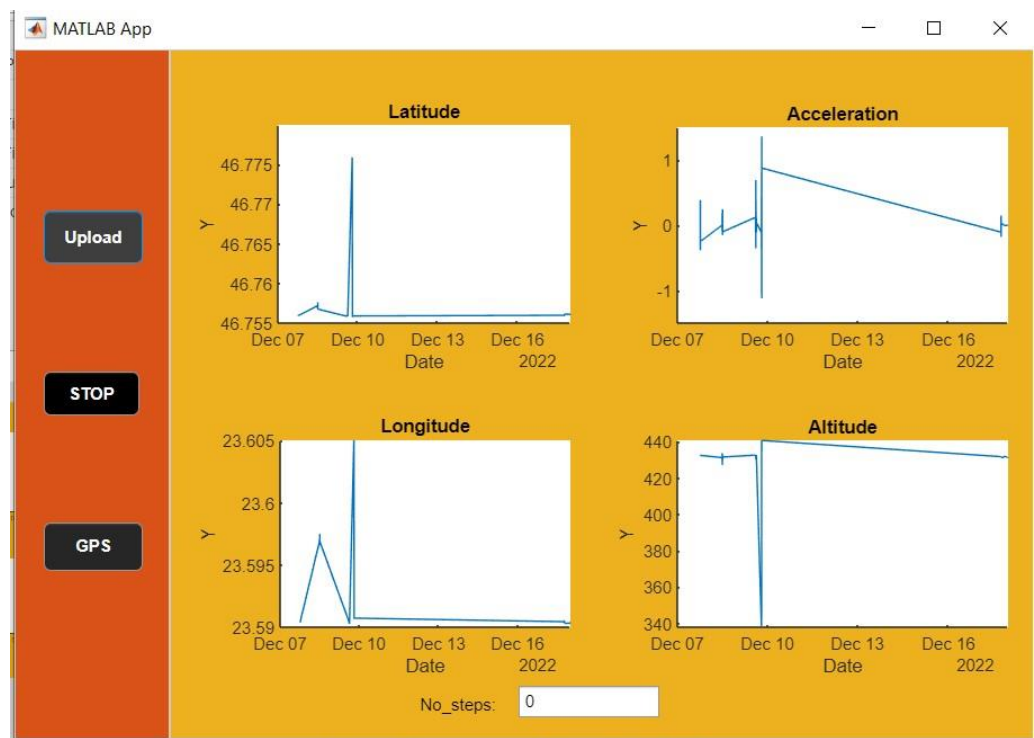


Fig.9 Graphical interface

```
Editor - D:\an 4 sem 1\sisst iot\IoT Project\IoT Project\IoT_GUI_mod\IoT_GUI_mod\read.m *
start_and_upload.m x app1_exported.m x read.m x plot_gps.m x +
1 function [lat,lon,alt,acc,time]=read()
2 %return read values as matrices
3 readk='5V5ABZ7A7IMVL16Q';
4
5 data = thingSpeakRead(1944434,'Fields',[1:5],'ReadKey',readk,'NumPoints',1000,OutputFormat='TimeTable');
6 time=data.Timestamps;
7 lat=data.Latitude;
8 lon = data.Longitude;
9 alt = data.Altitude;
10 acc = data.Acceleration_no_G;
11
12
13 end
```

Fig.11 Read data from ThingSpeak

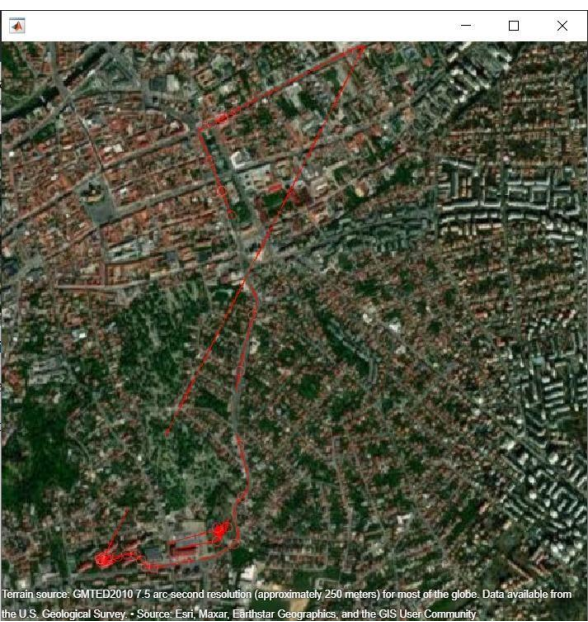


Fig.10 GPS plot

3. Conclusions

In conclusion, we managed to successfully take data from Matlab mobile sensors, send it to the ThingSpeak cloud, and after all we read the data from the platform, plot it on our GUI, count the number of steps using acceleration, make an alarm for exceeding a number of steps and plot the GPS coordinates. This project can be used to monitor (via GPS and number of steps) in real time the route followed, for example jogging route, trekking route, hiking route.

4. Bibliography

- [1] <https://www.mathworks.com/help/matlabmobile/ug/sensor-data-collection-with-matlabmobile.html>;
- [2] <https://www.mathworks.com/help/thingspeak/>;
- [3] <https://www.mathworks.com/help/thingspeak/thingspeakwrite.html>;
- [4] <https://www.mathworks.com/help/matlabmobile/ug/counting-steps-by-capturing-accelerationdata.html>;
- [5] https://www.mathworks.com/help/matlab/ref/thingspeakread.html?s_tid=doc_ta.