APPENDIX 1: SETUP A THINGSPEAK ACCOUNT AND CREATE "EMPTY" APP

- Setup your own ThingSpeak account, it is free for you! (at least for one year)
- Link to ThingsSpeak setup: thingspeak.com/login
- Create a new app under apps/visualisations
- Use the custom template
- Provide name to the template

> Now an "empty" app is created, you will need it with the next step

APPENDIX 2/1: INSERT CODE INTO APPS

- For each algorithm you have to setup a particular app within your personal free ThingSpeak account
- Use the created template (Appendix 1)
- Open editor,
- MATLAB language proficiency is required
- Insert code from
  - APPENDIX 3 (dew point analysis) or
  - APPENDIX 4 (barometric pressure vs. rain analysis) or
  - APPENDIX 5 (filtering algorithm)

APPENDIX 2/2: EDIT APPS CODE & RUN

- Data are automatically collected from channel 12397
- Setup own output channel, if necessary
- When necessary, substitute template output channel id and write key by your own
- Save and **run**
- After successful running you should see the resulting graphs for each one of the algorithms
- In case of MATLAB syntax errors modify code until it runs properly

APPENDIX 3/1: MATLAB CODE FOR DEW POINT ANALYSES (FIRST PART)

```
% extended Analyses: Calculate Dew Point, Data source: A public weather station, USA
% In: temperature,barometric pressure, from channel 12397
readChId = 12397;
% Out: dew point over time to channel 462321, you might setup your own channel!
writeChId =  462321;  %use your own channel id
writeKey = 'SU3VGI4DATPXIGK4'; %use your own write key
% fetch last 3000 minutes of data, more does not work online
[d,t,ci] = thingSpeakRead(12397,'NumPoints',3000);
% fetch and convert temperature from deg F to deg C
tempF = d(:,4); % field 4 is temperature in deg F
tempC = (5/9)*(tempF-32); % convert to deg Celsius
% set channel output
thingSpeakWrite(writeChId,[tempF,tempC],'Fields',[1,2,],...
'TimeStamps',t,'Writekey',writeKey);
% Set necessary constants
b = 17.62;
c = 243.5;
humidity = d(:,3); % field 3 is relative humidity in percent
gamma = log(humidity/100) + b*tempC ./ (c+tempC);
tdp = c*gamma ./ (b-gamma);
% continued on appendix 3/2
```

APPENDIX 3/2: MATLAB CODE FOR DEW POINT ANALYSES (CONTINUED)

```
% continuation from appendix 3/1
% Plot Temperature, Humidity and Dew Point versus Time
availableFields = ci.FieldDescriptions'
figure(1)
[ax, h1, h2] = plotyy(t,[tempC tdp],t,humidity);
set(ax(2),'XTick',[])
set(ax(2),'YColor','k')
set(ax(1),'YTick',[0,5,10,15,20,25])
set(ax(2),'YTick',[0,20,40,60,80,100])
datetick(ax(1),'x','keeplimits','keepticks')
set(get(ax(2),'YLabel'),'String',availableFields(3))
set(get(ax(1),'YLabel'),'String','Temperature (C)')
title('IoT4SME/Demo2: Dew point vs. temperature and humidity')
xlabel('Local Daytime')
grid on
legend('Location','West','Temperature','Dew point', 'Humidity')
```

APPENDIX 4/1: BAROMETRIC PRESSURE VS. RAIN ANALYSIS (FIRST PART)

```
% extended Analyses: barometric pressure vs. rain analysis, Data source: A public weather station, USA
% In: baro pressure, rainfall
% Out: bar diagram rain, pressure, trendline vs. time
% the date 4./5.6. was selected for demonstration reasons (this date was suitable)
readChId = 12397;
% Analytic question to be answered: does the barometric pressure really fall as a rain storm approaches/rainy day ?
[d,t,ci] = thingSpeakRead(12397,'DateRange',[datetime('Jun 04, 2014'),...
   datetime('Jun 06, 2014')]);
availableFields = ci.FieldDescriptions'
% Fetch pressure
baro = d(:,6); % % field 6 is pressure
% Number of one-minute samples
length(baro);
% compute excess points beyond the hour: extraData= number of data points excessing the last hour
extraData = rem(length(baro),60);
rain = d(:,5); % field 5 is rainfall from sensor in inches per minute
% Init at beginning of vector
rain(1:extraData) = [];
t(1:extraData) = [];
% continued on appendix 4/2
```

APPENDIX 4/2: BAROMETRIC PRESSURE VS. RAIN ANALYSIS (CONTINUED)

```
% continuation from appendix 4/1
RainHourly = sum(reshape(rain,60,[]))'; % reshape a vector into a matrix, sum adds column-wise
maxRainPerMinute = max(rain)
june5rainfall = sum(rainHourly(25:end)) % 24 hours of measurements from June 5
baroHourly = downsample(baro,60); % hourly samples
timestamps = downsample(t,60); % hourly samples
figure(2)
subplot(2,1,1)
% creating a bar graph
bar(timestamps,rainHourly) % plot rain
xlabel('Date and Time')
ylabel('Rainfall (inches /per hour)')
grid on
datetick('x','dd-mmm HH:MM','keeplimits','keepticks')
title('MyDEMO/Demo:Rainfall on June 4 and 5')
% continued on appendix 4/3
```

APPENDIX 4/3: BAROMETRIC PRESSURE VS. RAIN ANALYSIS (CONTINUED)

```
% continuation from appendix 4/2
subplot(2,1,2)
hold on
plot(timestamps,baroHourly) % plot barometer
xlabel('Date and Time')
ylabel(availableFields(6))
grid on
datetick('x','dd-mmm HH:MM','keeplimits','keepticks')
% detrend computes the least-squares fit of a straight line  to the data and subtracts the resulting function from the data
detrended_Baro = detrend(baroHourly);
baroTrend = baroHourly - detrended_Baro;
plot(timestamps,baroTrend,'r') % plot trend
hold off
legend('Barometric Pressure','Pressure Trend')
title('MyDemo/Demo2:Barometric Pressure on June 4 and 5')
```

APPENDIX 5/1: FILTERING OF OUTLIERS (FIRST PART)

```
% extended Analyses: Clean-up Data , Data source: A public weather station, USA
% In: temperature on a particular day
% Out: cleaned temperature on a particular day
% Question: how to clean up the data?
readChId = 12397;
[d,t,ci] = thingSpeakRead(readChId,'DateRange',[datetime('May 30, 2014'),datetime('May 31, 2014')]);
availableFields = ci.FieldDescriptions'
%fetch temperature in F
rawTemperatureData = d(:,4);
% convert to Celsius
rawTemperatureDataC = (5/9)*(rawTemperatureData-32);
newTemperatureData = rawTemperatureDataC;
% get the minimum, result: unreasonable value
minTemp = min(rawTemperatureDataC)
%% Use a Threshold Filter to Remove Outside the range (Outliers)
% data limit: temperatures that are > 0C or < 30C (Springtime)
%Horizontally transponed timescale
tnew = t';
outlierIndices = [find(rawTemperatureDataC < 0); find(rawTemperatureDataC > 30)]
% delete data outside the range
tnew(outlierIndices) = [];
newTemperatureData(outlierIndices) = [];
% continued on appendix 5/2
```

APPENDIX 5/2: FILTERING OF OUTLIERS (CONTD.)

```
% continuation from appendix 5/1
% Plot the cleaned up data and the original data
figure(1)
subplot(3,1,2)
plot(tnew,newTemperatureData,'-og')
%Date formatted tick labels
datetick
xlabel('Time of Day')
ylabel('Temperature (C)')
title('MyDemo/Demo:Filtered Data - outliers deleted')
grid on
subplot(3,1,1)
plot(t,rawTemperatureDataC,'-.r')
datetick
xlabel('Time of Day')
ylabel('Temperature (C)')
% continued on appendix 5/3
```

APPENDIX 5/3 : FILTERING OF OUTLIERS (CONTD.)

```
% continuation from appendix 5/2
title('MyDemo/Demo:Original Data')
grid on
%% Use Median Filter to Remove Outlier Data
n = 5; % this value determines the number of total points used in the filter
f = medfilt1(rawTemperatureDataC,n);
subplot(3,1,3)
plot(t,f,'-o')
datetick
xlabel('Time of Day')
ylabel('Temperature (C)')
title('MyDemo/Demo: Filtered Data -Median filter')
grid on
```