

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Cybernetics



# Master's Thesis

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Topic name: IoT based Condition Monitoring System

Student: Deepak Koranga

Supervisor: doc. Ing. Radislav Šmíd, Ph.D.

Prague,

May 2017



**Czech Technical University in Prague  
Faculty of Electrical Engineering**

**Department of Cybernetics**

## **DIPLOMA THESIS ASSIGNMENT**

**Student:** Deepak K o r a n g a

**Study programme:** Cybernetics and Robotics

**Specialisation:** Robotics

**Title of Diploma Thesis:** IoT based Condition Monitoring System

### **Guidelines:**

1. Get familiar with condition monitoring techniques based on signals (vibrations, temperatures, etc.) measured on typical industrial objects (e.g. motors, fans, compressors, valves, dampers)
2. Design and realize a condition monitoring system composed of sensors, microcontroller and the communication interface for wireless connection (preferably via WiFi) to MQTT broker.
3. Connect the monitoring system to IoT cloud and create a demo application for visualization and analytics.

### **Bibliography/Sources:**

- [1] Šmíd, R.; Hanuš, O.: Condition monitoring using the Internet of Things (IoT) - In: CM 2016 and MFPT 2016. Northampton: British Institute of Non-Destructive Testing, 2016.
- [2] Randall, R. B.: Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications, Wiley 2011.
- [3] Ganssle, J. G. : The Art of Designing Embedded Systems, Newness 2008.

**Diploma Thesis Supervisor:** doc. Ing. Radislav Šmíd, Ph.D.

**Valid until:** the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic  
**Head of Department**

prof. Ing. Pavel Ripka, CSc.  
**Dean**

Prague, January 19, 2017



## Acknowledgement

I would like to express my special appreciation and thanks to doc. Ing. Radislav Šmíd Ph.D. for being my thesis supervisor and for all his kind contribution by defining a starting point for this project, the constant helpful supervision and providing valuable input throughout the course of this thesis.

Also a big thank you to the Texas Instruments' MSP430 Forum where I received great help and recommendations multiple times by MSP430 team member on making the best use of the MSP430 low power microcontroller used in this application.

### Declaration of Authenticity:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodological instructions for observing the ethical principles in the preparation of university theses.

Prague, date .....

signature

## **Abstract**

Using the Internet of Things for Condition Monitoring has many advantages like interoperability between machines, sensors and actuators, at the same time adding features such as cloud data storage and analytics. In the age of Industry 4.0, maintenance strategies are evolving by making use of the advancements and convergence in other technologies such as Internet of things and Cloud Computing.

This thesis deals with demonstrating the use of Internet of Things for Condition Monitoring using Vibration Analysis. Utilizing the hardware, software and communication protocol well suited for IoT applications, an IoT chain has been designed and tested for successful implementation. The hardware with low power features, lightweight communication protocol MQTT, large bandwidth sensor, as well as an IoT platform ThingSpeak with configurable analytics and visualization of data, is used to make an efficient Internet of Things chain.

## **Keywords**

Condition Monitoring, Vibration Analysis, Internet of Things, Industry 4.0, MQTT Protocol, Sensors, Machine health monitoring

## Table of Contents

|  |    |
|--|----|
| <b>1. Introduction</b>   | 1  |
| 1.1. Condition Monitoring  | 2  |
| 1.2. Applying IoT principles in Condition Monitoring - specifically for Vibration Analysis | 2  |
| <b>2. Objective</b>  | 4  |
| 2.1. Requirements  | 4  |
| <b>3. Concept Design</b>   | 5  |
| 3.1. Sensors   | 5  |
| 3.2. Data Acquisition, Processing and Publishing Unit                                      | 6  |
| 3.3. Access Point  | 6  |
| 3.4. MQTT Broker   | 6  |
| 3.5. Cloud   | 7  |
| <b>4. Design</b>   | 8  |
| 4.1. The Monitored object  | 9  |
| 4.2. Accelerometer   | 11 |
| 4.3. Data Acquisition, Processing and Publishing Unit                                      | 11 |
| 4.3.1. MSP430F5529 LaunchPad Development Kit   | 12 |
| 4.3.2. SimpleLink™ Wi-Fi® CC3100 BoosterPack   | 16 |
| 4.4. Access Point  | 17 |
| 4.5. MQTT  | 18 |
| 4.6. Raspberry Pi  | 20 |
| 4.7. Wi-Pi™ WLAN USB Module  | 20 |
| 4.8. ThingSpeak™   | 21 |
| <b>5. Implementation</b>   | 22 |
| 5.1. Hardware  | 22 |
| 5.2. Software  | 25 |
| 5.2.1. Data Acquisition, Processing and Publishing   | 26 |
| 5.2.2. Broker  | 29 |
| <b>6. Results</b>  | 31 |
| 6.1. Experimental Testbench Vibration device Setup   | 31 |
| 6.2. ThingSpeak Results  | 33 |
| 6.2.1. Multiple Publishers Simultaneous Operation Test                                     | 34 |
| 6.2.2. Experimental Testbench Results  | 36 |

|                                    |           |
|------------------------------------|-----------|
| 6.2.3. Additional Experiments..... | 45        |
| 6.3. Analysis .....                | 46        |
| <b>7. Conclusion .....</b>         | <b>48</b> |
| <b>Bibliography.....</b>           | <b>i</b>  |



## List of Figures

|  |    |
|--|----|
| <b>Figure 1:</b> Basic Block diagram of multiple parallel chains.....                          | 5  |
| <b>Figure 2:</b> Block Diagram .....   | 9  |
| <b>Figure 3:</b> Flowchart of the process .....  | 10 |
| <b>Figure 4:</b> Functional Block Diagram, MSP430F5529. Image source [21]. .....               | 13 |
| <b>Figure 5:</b> ADC block diagram. Image source [21]. .....                                   | 15 |
| <b>Figure 6:</b> Functional Block Diagram, CC3100. Figure reproduced from [22]. .....          | 17 |
| <b>Figure 7:</b> MQTT Architecture .....   | 19 |
| <b>Figure 8:</b> MSP-EXP430F5529LP LaunchPad Development kit header. Reproduced from [34]..... | 22 |
| <b>Figure 9:</b> MSP430F5529 and CC3100 Pin Compatibility. Source [35] .....                   | 23 |
| <b>Figure 10:</b> MSP430 and Accelerometer 810M1-0025X connection .....                        | 24 |
| <b>Figure 11:</b> Internet Connectivity .....  | 25 |
| <b>Figure 12:</b> Configure CC3100 and Connect to AP .....                                     | 28 |
| <b>Figure 13:</b> Subscribed to Topic and publishing to ThingSpeak .....                       | 30 |
| <b>Figure 14:</b> Test Setup for Vibration Measurement.....                                    | 32 |
| <b>Figure 15:</b> ThingSpeak out for Accelerometer 810M1-0025X .....                           | 34 |
| <b>Figure 16:</b> ThingSpeak out for Accelerometer 805-0050.....                               | 35 |
| <b>Figure 17:</b> Acceleration output for 20Hz .....   | 37 |
| <b>Figure 18:</b> Crest Factor output for 20 Hz .....  | 38 |
| <b>Figure 19:</b> Acceleration output for 50Hz .....   | 39 |
| <b>Figure 20:</b> Crest Factor output for 50 Hz .....  | 40 |
| <b>Figure 21:</b> Acceleration output for 100Hz .....  | 41 |
| <b>Figure 22:</b> Crest Factor output for 100 Hz .....   | 42 |
| <b>Figure 23:</b> Acceleration output for 1 kHz.....   | 43 |
| <b>Figure 24:</b> Crest Factor output for 1 kHz .....  | 44 |
| <b>Figure 25:</b> Uncontrolled Vibration output 1 .....  | 45 |
| <b>Figure 26:</b> Uncontrolled Vibration output 2 .....  | 45 |
| <b>Figure 27:</b> Uncontrolled Vibration output 3 .....  | 46 |

## List of Tables (if needed)

|  |    |
|--|----|
| <b>Table 1:</b> Low Power Modes in MSP430F5529. Source [21]. ..... | 16 |
| <b>Table 2:</b> Table of Parameters used for Experiment .....      | 32 |

## Abbreviations

| Abbreviation | Explanation                       |
|--------------|-----------------------------------|
| ADC          | Analog-to-Digital Conversion      |
| DMA          | Direct Memory Access              |
| GPIO         | General Purpose Input Output      |
| IoT          | Internet of Things                |
| MQTT         | Message Queue Telemetry Transport |
| QoS          | Quality of Service                |
| RMS          | Root Mean Square                  |

# 1. Introduction

This Master's thesis project was conducted during summer 2017 at Czech Technical University in Prague, under the supervision of Dr. Radislav Smid.

With the advent of Industry 4.0, the automation at every possible process has become rather a necessity. The technologies such as Internet of Things and Cloud Computing are converging to make this possible. The Internet of Things is an indispensable part of Industry 4.0 revolution.

Maintenance is one of the most critical process in any production or manufacturing. Machine failures don't happen all of a sudden, but some faults when not detected and treated on time build up to cause the failure eventually. The consequences sometimes are catastrophic and can cause the production to stop which leads to big repairing losses or even shutdown, depending on whether the failure is repairable or not. There are maintenance strategies to provide solutions to this problem. Instead of letting the machines operate until they fail, the state of the machine is observed constantly for any faults that might lead to failure.

The scope of this thesis is one of the many Intelligent Maintenance Systems, called Condition Monitoring. This chapter explains what Condition Monitoring is and how we can use Internet of Things to do the same. In the succeeding chapters, the aim of this project as well as the implementation would be described in detail.

The maintenance strategies are majorly classified into two, viz.

- Preventive Maintenance or Time Based Maintenance
- Predictive Maintenance or Condition Based Maintenance (CBM)

The preventive maintenance is about diagnosing and replacing parts at regular intervals, while predictive maintenance is about monitoring the condition of the machine, also known as Condition based maintenance.

## 1.1. Condition Monitoring

Condition monitoring is a type of Condition-based maintenance in which some machine parameter or condition in machinery is monitored for a significant change that could be indicative of a developing failure. This allows maintenance to be scheduled or other actions to be taken to avoid the consequences of failure, before the failure actually occurs. It is not done at regular time intervals but rather depending on the “condition”, which is a term for defining one or more parameters that could give an insight into the machine’s current health. Condition Monitoring relies on trend analysis which means that it is the change in the observed parameter that is monitored, not the parameter itself.

There are several methods to monitor the performance of machinery or a component. Some of them are listed below

- Vibration analysis [4, 5, 6, 7, 8]
- Oil analysis [9, 10]
- Thermography [11]
- Acoustic emission testing [12, 13]

Of all these methods, Vibration analysis is the most successful technique [14, 15, 16, 17] and has several advantages over others and will be our method for this thesis. With proper analysis of vibration signals, one can narrow down to the specific component which is at fault.

## 1.2. Applying IoT principles in Condition Monitoring - specifically for Vibration Analysis

Condition monitoring can be made even more effective when the sensor data is directly stored in a cloud environment, where it can be further analyzed to take necessary actions when required. This can be achieved by using Internet of Things (IoT).

As the name suggests, IoT is about expanding the scope of Internet from merely connecting humans to the network. It provides a platform to add the physical “things” or appliances or machines or devices to the inter-connected network. It is expected that in the next five to ten years, there will be more devices and machines connected to the internet than the current whole world population of 7.5 Billion. IoT is one step above M2M Machine to machine connectivity in that it is not a closed environment but connects to the Internet, as we know it. Also, the Internet of Things is indispensable to Industry 4.0 [18].

This thesis is primarily concerned with condition monitoring of a machine for detecting any faults that could lead to future failures. IoT requires hardware, software and communication infrastructure (such as internet, local area network, devices ... et al) for its implementation. The IoT chain, as described in the later chapters, consists of one of the various sensors at one end (source), and a cloud application at the other end (sink). The resulting sensor-to-cloud chain has a capability to put the information, about the specific parameter in question, to the IoT platform for further analysis either by a human or some machine learning software. The parameter depends on the choice of the sensor. For an instance, the sensors could be any of accelerometer, temperature sensor, voltage sensor etc. and the parameter would simply be the quantity that they measure (namely acceleration and its derivatives such as velocity, position and orientation; temperature and voltage respectively).

## 2. Objective

The objective of this thesis is to demonstrate a proof of concept for vibration analysis of a vibrating device being monitored using Internet of Things (IoT) chain. The guidelines for the approach are as follows:

- To get familiarized with condition monitoring techniques based on signals (vibrations, temperatures, etc.) measured on typical industrial objects (e.g. motors, fans, compressors, valves, dampers)
- To design and realize a condition monitoring system composed of sensors, microcontroller and the communication interface for wireless connection (preferably via WiFi) to MQTT broker.
- To connect the monitoring system to IoT cloud and create a demo application for visualization and analytics.

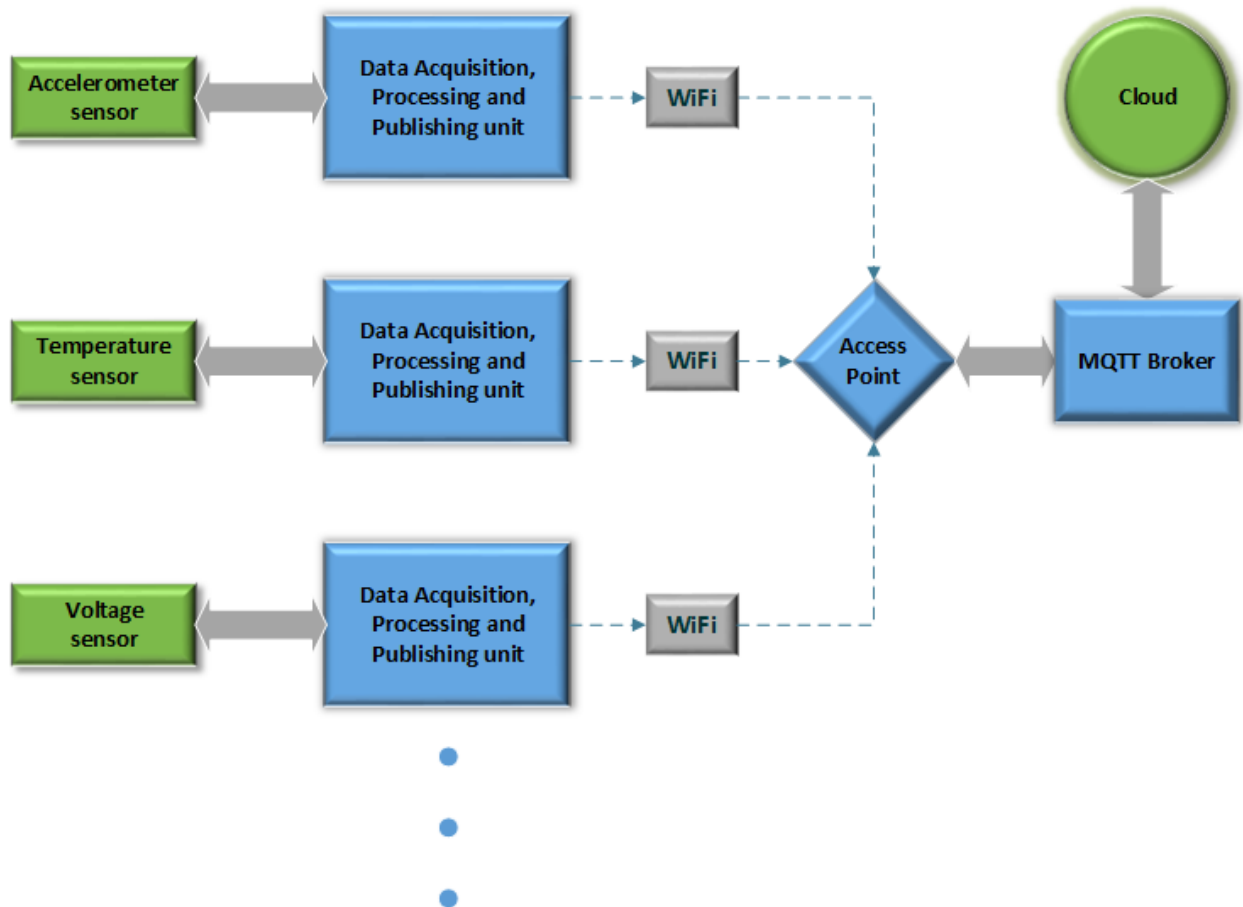
### 2.1. Requirements

Since this a specialized application having many dependencies, a specific strategy and list of pre-requisites should be met in order to have a successful implementation.

- The data from all accelerometers should be sent separately to the cloud
- The implementation assumes that raw sensor data is already pre-processed into a specific format that is compatible with the cloud interface so as to pass it on to the next phase.
- The system should be scalable i.e. new sensor-to-cloud chains should be added without affecting the existing chains
- The end-users should only receive updates on the sensor of their interest

### 3. Concept Design

The idea is to create a system consisting of several parallel sensor-to-cloud chains as shown in the *Figure 1*. The Basic Block diagram shows how we can just add several chains without affecting the existing ones. This is possible because of the choice of communication protocol. Any two sensors are basically isolated from each other and the data is dealt with individually.



**Figure 1:** Basic Block diagram of multiple parallel chains

#### 3.1. Sensors

On the lowest end is a sensor, like accelerometer, temperature sensor, voltage sensor etc., that is used to connect to the object being monitored in the physical world. It could be a variety of sensors on the same object or the same kind of sensors at different positions on the object based on what is being measured. Also, we have a choice between digital and analog types.

### 3.2. Data Acquisition, Processing and Publishing Unit

The sensors are directly connected to Data Acquisition, Processing and Publishing Unit (DAPPU) which is programmed according to the specific sensors. The accelerometer in itself cannot communicate with the other units and hence DAPPU can be considered the first state of the IOT chain.

Here are the major functions that this unit broadly serves:

- Receive data from the sensor using Analog to Digital Conversion
- Process the data to extract parameters used for Vibration Analysis
- Connect to the WLAN access point
- Prepare data packets containing the vibration data to be sent over MQTT protocol
- Send the data packets to the MQTT broker

### 3.3. Access Point

To provide the necessary communication infrastructure, an access point enables the wireless connectivity. The DAPPU is directly connected to the access point through Wi-Fi. This access point is common to all the DAPPUs connected to various sensors. It is just a wireless interface and nothing happens at this stage. However it still plays an important role as this access point is in turn connected to a broker through a wired interface.

### 3.4. MQTT Broker

The MQTT Broker is a very useful feature of MQTT protocol which also has several advantages over other protocols such as HTTP [19]. The broker basically segregates data from different chains and sends it to the cloud platform. In the whole chain, only the broker is connected to the Internet, while other devices are just connected through WLAN connection to the access point. Even though there could be scenarios where two chains can exchange information between each other, in this case they are isolated and thus the data needs to be propagated separately.



### 3.5. Cloud

The cloud is the terminal state of this chain. However, it is not only for storage of the data (record keeping), but also for analytics. In further sophisticated setups it can be used for machine learning as well that can enable appropriately planned actions prior to failures.

There could be just one sensor being monitored or hundreds of them. So the need for analysis varies from pattern recognition of just one data channel to hundreds of sensors uploading different patterns at the same time. Hence the presentation of the data is a crucial step that is taken care by what is called an IoT platform. . **An IoT platform does the tedious work of gathering, analyzing and acting on data.**

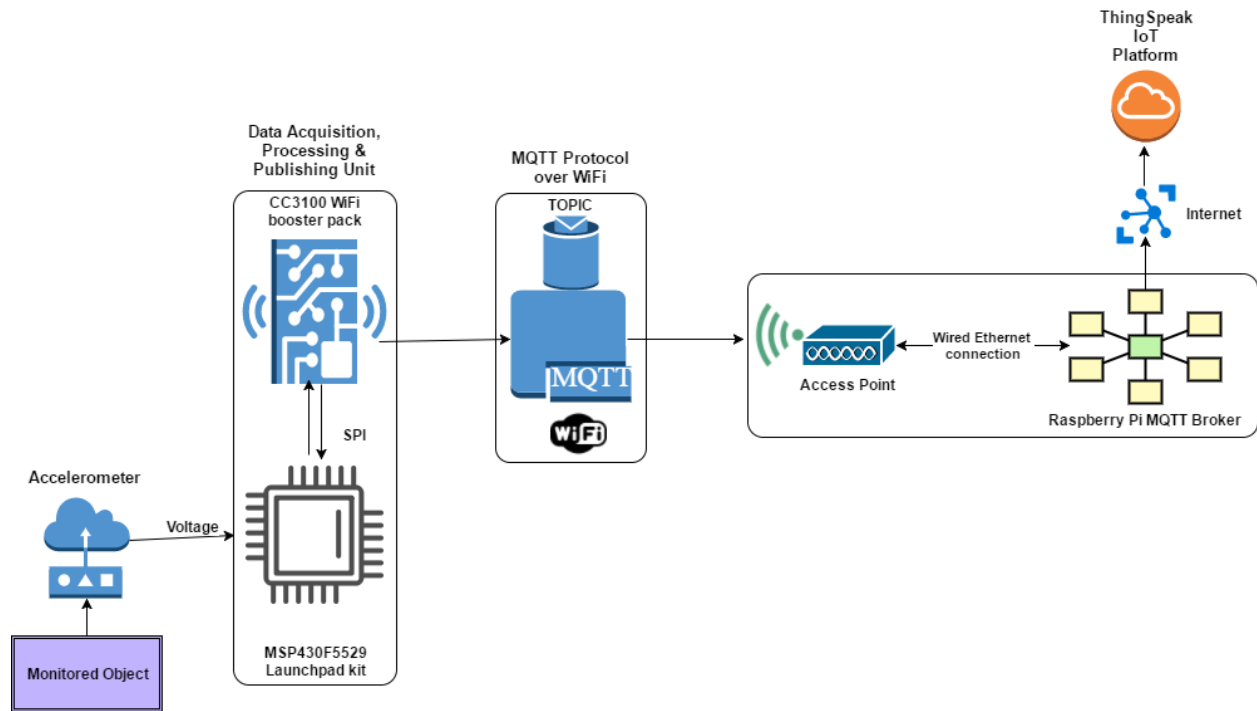
## 4. Design

We will go through the design of a single sensor-to-cloud chain first and then later add new ones in parallel. A lower level block diagram of the system is shown in the *Figure 2*. It has details on the constituents of the various units present in the system. More information on the hardware and software components used and their features will be presented in this section.

Essentially, the data goes through several stages starting from its capture at the sensor level to the IoT platform. Here are the steps

- The accelerometer senses the vibration and sends a corresponding voltage to the Microcontroller it is connected to
- The Microcontroller, using ADC, samples the analog values and store 1k samples
- Data is processed and then, using a Wi-Fi booster pack, establishes a connection to the access point and sends this data over MQTT Protocol, which is above TCP/IP
- The MQTT broker, connected to the access point, receives this data and separates it based on a topic assigned to the sensor
- A code running on the broker constantly updates this data to the cloud separately for each sensor
- The IoT platform performs analysis on the data and presents it to the end-user

*Figure 3* shows a flowchart describing how the data from accelerometer goes through various stages and is updated to the corresponding channel of the IoT platform. It begins with Analog to Digital on the accelerometer data and saving the result of 1k samples to the memory. Then the data is processed in the Data Acquisition, Processing and Publishing Unit. A WLAN connection is made between this unit and the access point in the next step. Once the data has been processed and the connection has been made, the data is sent over MQTT protocol to the broker that is directly connected to the access point. The broker, which is connected to the internet uploads the same data to the IoT platform where it is displayed.

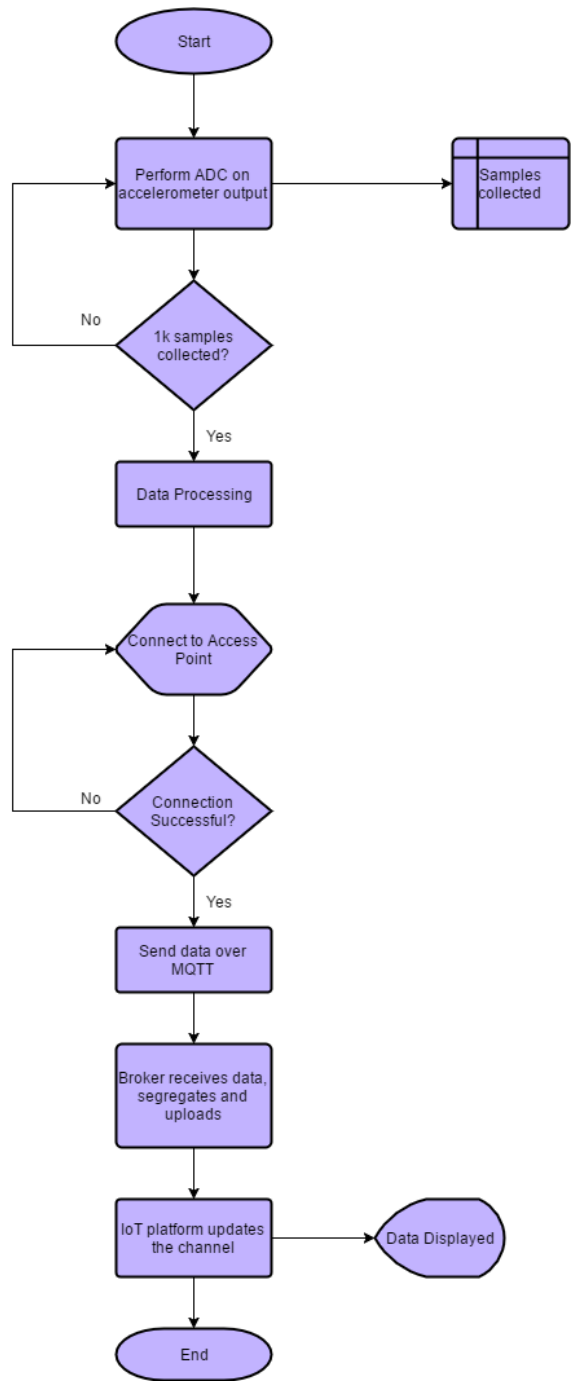


**Figure 2: Block Diagram**

#### 4.1. The Monitored object

The monitored object considered in this project is a vibration actuator or a shaker that generates vibrations corresponding to the output of the waveform generator that is connected to it. The accelerometer is directly mounted on the shaker so that the vibrations are sent through the IoT chain to its terminal which is the cloud. Every machine, healthy or faulty, generates some vibrations while it is running. The vibration frequencies that are of interest for diagnostics purpose are usually not human perceptible, so an accelerometer is used for this purpose. Based on the type and range of accelerometer, a band of frequencies are captured and sent for further processing to the subsequent units in this system.

Multiple accelerometers can be mounted on the same machine for more details. For example, one of them can be mounted horizontally, while the other in the vertical direction. This redundancy could ensure that the position of the accelerometer is not responsible for any missing data.



**Figure 3:** Flowchart of the process

## 4.2. Accelerometer

Two accelerometers are used in this study, with part numbers 810M1-0025X and 805-0050 respectively. Both of them are analog type and have various applications, one of which is Machine Health Monitoring.

Here are some of the technical specifications 810M1-0025X [20] as found on the datasheet:

- Stable piezo-ceramic crystals in shear mode
- X-axis Measurement
- Range of  $\pm 25g$
- Flat frequency response up to  $>6$  kHz
- Requires 3.3 to 5.5Vdc Excitation Voltage
- Full Scale Output Voltage of  $\pm 1.25V$
- Sensitivity (mV/g) 50.0

For 805-0050 the datasheet lists the following specifications:

- stable piezo-ceramic crystal
- Range of  $\pm 50g$
- Flat frequency response to 15 kHz. Typical being 1-8000 Hz ( $\pm 1dB$ ) and 0.3-10000 Hz ( $\pm 3dB$ )
- Requires 2 to 10 mA of Excitation current
- Full Scale Output Voltage of  $\pm 5V$
- Sensitivity (mV/g) 100.0

The accelerometers get their power from MSP430F5529 LaunchPad Development Kit and their data output is connected to the pin P6.0 of the same. The MSP430F5529 LaunchPad Development Kit performs ADC operations on the analog output of the accelerometer to change it into digital and then use the same for further computations.

## 4.3. Data Acquisition, Processing and Publishing Unit

This is the most important stage of the IoT chain. The Data Acquisition, Processing and Publishing Unit consists of the following two hardware evaluation boards

- Texas Instruments MSP430F5529 LaunchPad Development Kit
- SimpleLink™ Wi-Fi® CC3100 wireless network processor BoosterPack™ plug-in module.

These two boards stacked together create an interface for the accelerometer to connect to the WLAN network which is created by an access point. This unit serves the most important purpose in the entire chain as it gathers the accelerometer data and does some processing on the data and at the same time it is also responsible for finally sending the data further through the WLAN network.

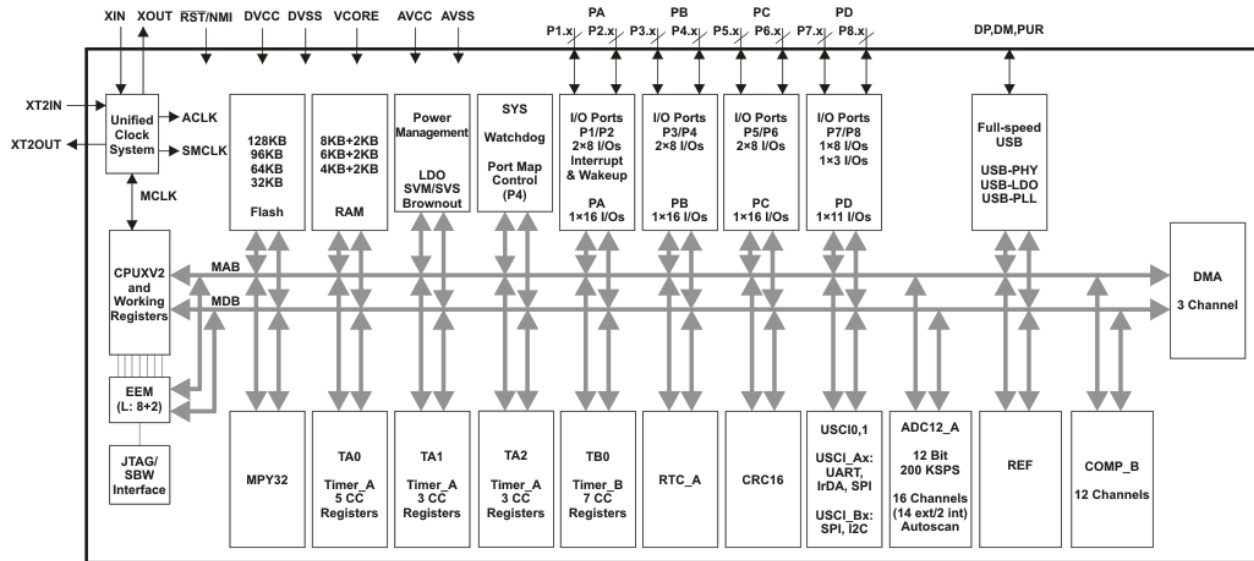
#### 4.3.1. MSP430F5529 LaunchPad Development Kit

The MSP-EXP430F5529LP [21] LaunchPad is an inexpensive, simple microcontroller development kit for the MSP430F5529 USB microcontroller. It includes the MSP430F5529 16-bit MCU, some of its features being

- 16-bit RISC CPU
- 128KB Flash and 8KB RAM
- Up to 25MHz CPU speed
- 12-bit analog-to-digital converter (ADC), timers
- Serial communication (UART, I2C, SPI)
- USB or battery powered. 5V and 3.3V through a high efficiency DC/DC converter
- Typical applications include analog and digital sensor systems

Additionally, it has a 40-pin BoosterPack expansion header, which can be used to add any of the various BoosterPack plug-in development board modules available. For example, the Simplelink Wi-Fi CC3100 BoosterPack is stacked over this header to provide Wi-Fi functionality as described later.

The *Figure 4* shows the functional block diagram of MSP430F5529. It contains the essential functions like Flash, RAM, Timers, and Universal Serial Communication Interface (USCI) etc. The functions that have been used in this project are described below.



**Figure 4:** Functional Block Diagram, MSP430F5529. Image source [21].

### Unified Clock System

The Unified Clock System (USC) module allows for the best use of the MSP430's low power capabilities while maintaining the performance at the same time. It has up to five clock sources, namely XT1CLK, VLOCLK, REFOCLK, DCOCLK, and XT2CLK.

The USC module provides three clock signals, namely ACLK, MCLK, and SMCLK. All of these are software selectable and can be sourced from any of the available clock sources (XT1CLK, VLOCLK, REFOCLK, DCOCLK, DCOCLKDIV, or XT2CLK). Both the ACLK and SMCLK are software selectable by individual peripheral modules, while MCLK is used by the CPU and system. It is also possible to divide any of these clock signals by 1, 2, 4, 8, 16, or 32.

The selection of a clock signal is done based on the requirement. A low clock frequency is a good choice for power saving, while for fast response time and processing, a high clock frequency would be more suitable. The clock accuracy is also a consideration at times, while in other cases it is acceptable not to have high accuracy clock. ACLK is 32786 Hz, and other clocks can be also sourced from the same source as ACLK.

### Analog-to-Digital Conversion

We are going to use the Analog-to-Digital Converter (ADC) functionality for processing the Accelerometer data. Here are some of the main features

- 16 channel, 12-Bit SAR core Analog-to-Digital Converter (ADC)
- Software-selectable on-chip reference voltage generation
- Sample-and-Hold (200k samples per second max)
- Autoscan available
- 16 conversion-result storage registers

The *Figure 5* shows a block diagram of ADC functionality of MSP430F5529 microcontroller. In the heart of the ADC, there is a 12-bit ADC core responsible for the conversion. The input to ADC core is fed through the Sample-and-Hold circuit which, as the name suggests, samples and holds the analog values from one of the 16 channels present.

The ADC, when enabled, does the conversion to these values with reference to the clock and the reference voltage. The clock ADC12CLK used for conversion as well as sampling (in Pulse mode) is software selectable and the clock source can be anything from SMCLK, MCLK, ACLK, and the ADC12OSC. The availability of pre-divider as well as divider further gives more options for modifying the clock frequency.

The start of sampling and conversion can be done by setting ADC12SC or the timer sources can be configured to handle the same. The final result of the ADC conversion is stored in one of the 16 available ADC12MEMx memory register. Accessing the values ends the conversion process by setting a flag.

#### *Direct Memory Access*

Another important feature for our use is the Direct Memory Access (DMA) module. The 3-channel internal DMA has the following features:

- Requires only two MCLK clock cycles per transfer
- Four different addressing modes
- Several transfer modes such as single, block, burst as well as repeated

The DMA controller can be used to move data from the ADC conversion memory to RAM. This happens without CPU intervention, so the CPU can be off during the whole transfer process.



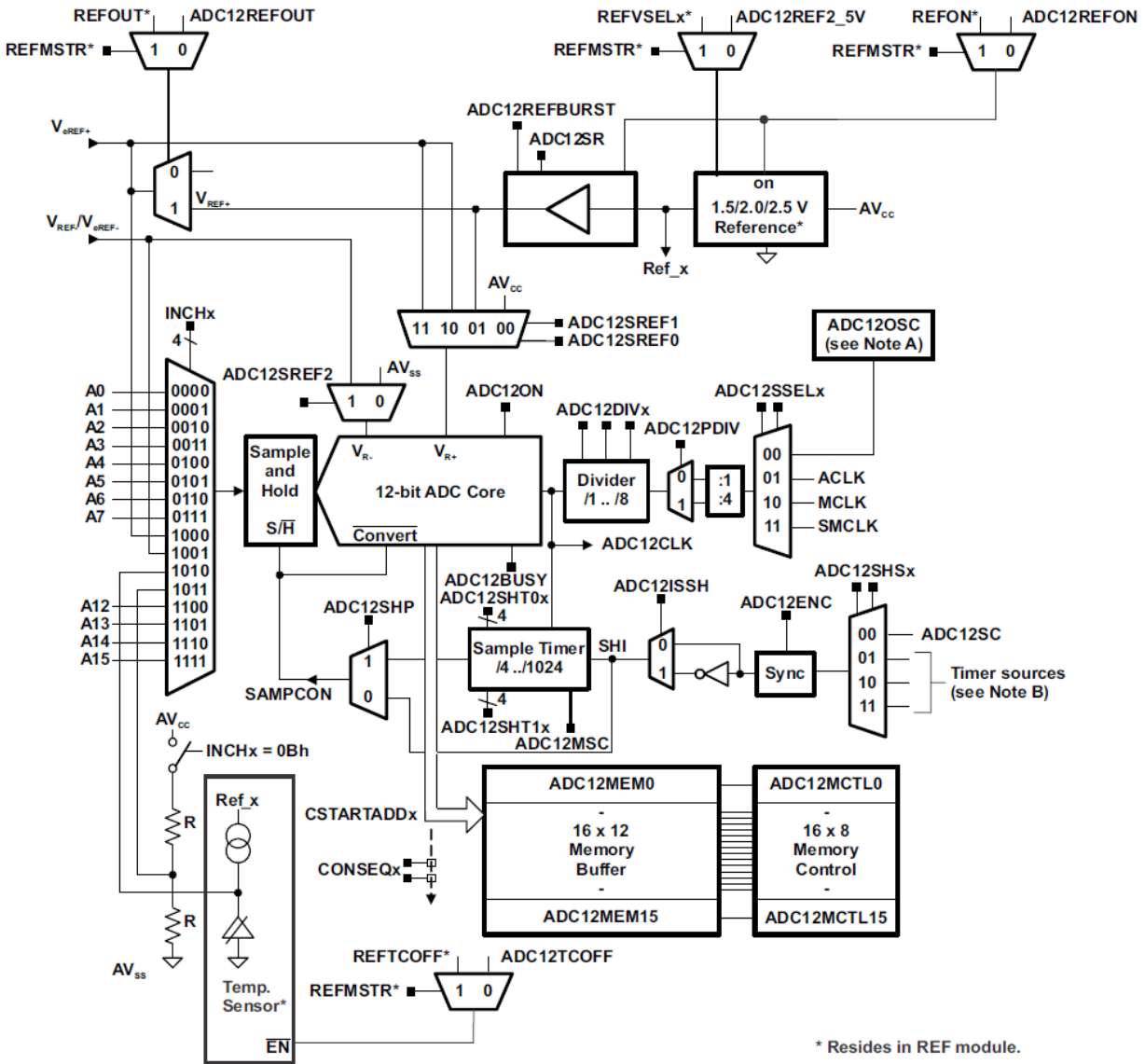


Figure 5: ADC block diagram. Image source [21].

### Low Power Modes

The “LP” in MSP-EXP430F5529LP stands for Low Power which is most special feature. The low power modes LPM0 through LPM4 have CPU disables along with different combinations of disabled clocks and clock sources. The *Table 1* shows a table of the low power modes available along with the state of clocks and clock sources. The choice of the most suitable low power mode depends on the needs of the application.

**Table 1:** Low Power Modes in MSP430F5529. Source [21].

| <b>Low Power Mode</b> | <b>CPU</b> | <b>MCLK</b> | <b>SMCLK</b>      | <b>ACLK</b> | <b>DCO</b>                       | <b>FLL</b>                |
|-----------------------|------------|-------------|-------------------|-------------|----------------------------------|---------------------------|
| <i>LPM0</i>           | Disabled   | Disabled    | Optionally active | Active      | Enabled if sources ACLK or SMCLK | Enabled if DCO is enabled |
| <i>LPM1</i>           | Disabled   | Disabled    | Optionally active | Active      | Enabled if sources ACLK or SMCLK | Disabled                  |
| <i>LPM2</i>           | Disabled   | Disabled    | Disabled          | Active      | Enabled if sources ACLK          | Disabled                  |
| <i>LPM3</i>           | Disabled   | Disabled    | Disabled          | Active      | Enabled if sources ACLK          | Disabled                  |
| <i>LPM4</i>           | Disabled   | Disabled    | Disabled          | Disabled    | Disabled                         | Disabled                  |

#### 4.3.2. SimpleLink™ Wi-Fi® CC3100 BoosterPack

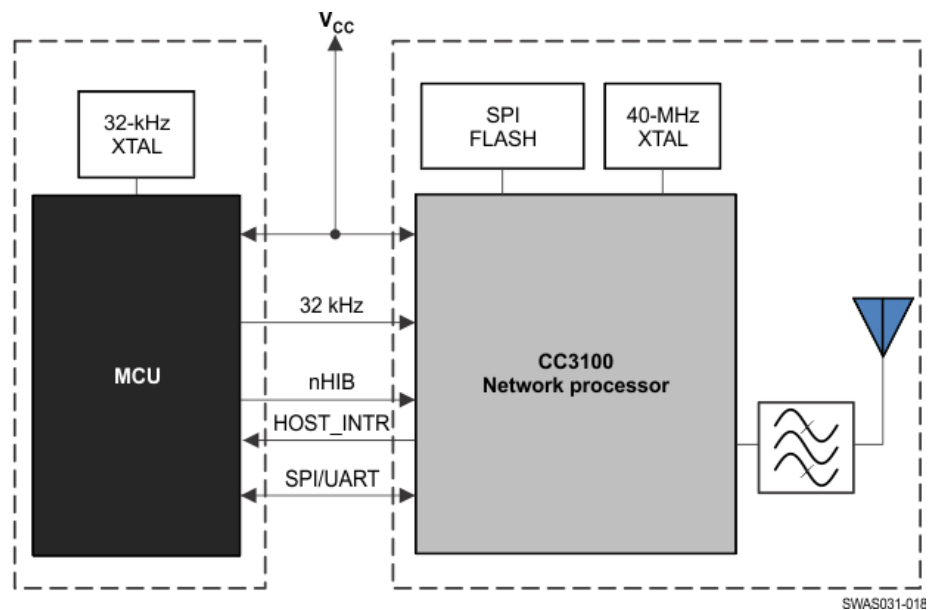
The SimpleLink™ Wi-Fi® CC3100 BoosterPack adds Wi-Fi capabilities to the MSP-EXP430F5529LP LaunchPad board, thereby enabling the communication infrastructure for the IoT chain. It contains CC3100 Wi-Fi network processor which is the industry’s first Wi-Fi CERTIFIED chip used in the wireless networking solution [22]. The Dedicated ARM MCU takes care of Wi-Fi connectivity as well as Internet Protocols, completely offloading the host microcontroller, which is MSP-EXP430F5529LP in this case.

The hardware includes an 802.11 b/g/n radio, baseband, and MAC with a powerful crypto engine for fast, secure Internet connections with 256-bit encryption. The 20-pin connector allows it to be stacked easily on the MSP-EXP430F5529LP LaunchPad.

Figure 6 shows the functional block diagram of CC3100 SimpleLink Wi-Fi solution. The communication with the host microcontroller can happen over either SPI or UART. The 4-wire serial peripheral interface (SPI) allows for integration with any microcontroller at a clock speed of 20 MHz. The CC3100 device has Station, AP, and Wi-Fi Direct® Modes for wireless communication with other devices in the chain.

The booster pack can be powered from on-board LDO using USB or directly connecting to the 3.3 V pin of LaunchPad.

The CC3100 also provides the most important feature which is security. The network engine of the SimpleLink Wi-Fi solutions encapsulates both the link layer and transport layer capabilities. The user is only required to deliver the information needed in order to establish the secure channel.



**Figure 6:** Functional Block Diagram, CC3100. Figure reproduced from [22].

#### 4.4. Access Point

For creating an Access point, the TP-Link Wireless Nano router Model no. TL-WR802N is used. It has the following features that makes it suitable for this application:

- 300Mbps wireless data rates
- IEEE 802.11n-2009
- Supports Router, Repeater, Client, AP and hotspot operation modes
- Supports Router, Repeater, Client, AP and hotspot operation modes

## 4.5. MQTT

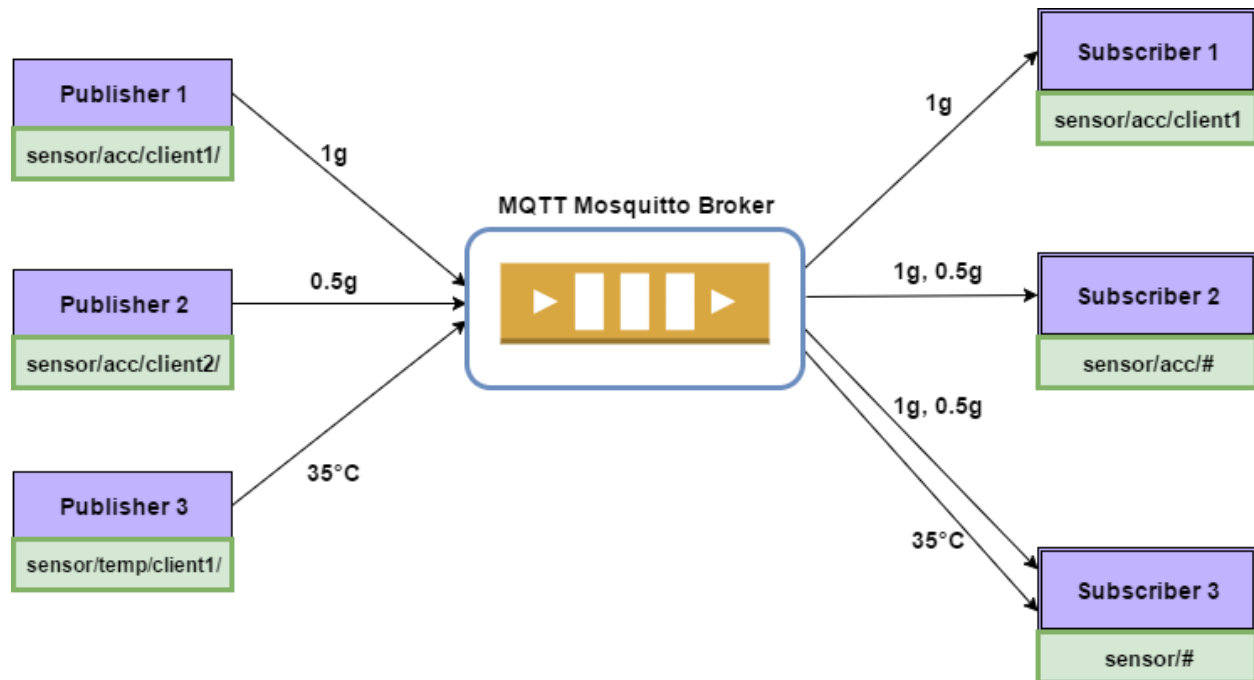
MQTT [23] stands for MQ Telemetry Transport or Message Queue Telemetry Transport and is an ISO standard (ISO/IEC PRF 20922) [24]. It is a machine-to-machine (M2M), Client Server protocol that is used on top of the TCP/IP protocol and has a publish-subscribe messaging pattern. MQTT is light weight, open, simple, and designed so as to be easy to implement [25]. It is designed for sending small data packets with high latency, low bandwidth links. MQTT control packet headers are kept as small as possible. A small header overhead lowers the amount of data transmitted over constrained networks. This makes it suitable for IoT applications such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

As an example of its popularity, Facebook has used aspects of MQTT in Facebook Messenger for online chat [26] and the native iOS Facebook app also credits the libmosquitto library and their blog post mentions that they are using MQTT extensively for notifications and updates [27].

### *MQTT Architecture*

There are three main components of MQTT viz. Subscriber, Publisher and Broker. The Publisher is the source of data, while the Subscriber is a receiver of data. The Broker keeps track of the data or message coming from the publisher(s) and redirects it to the correct subscriber(s). This is made possible by associating a “Topic” with every data block. The communication begins when the subscriber(s) connects to the broker and subscribes to one or more topics. The publisher publishes the data block containing a topic. The Broker receives the data from the publisher, checks the topic, prioritizes, and finally sends the data to the correct subscriber(s).

Figure 7 shows the MQTT architecture describing the flow of data. As can be seen, there are three different publishers that publish data on different topics. The first two are accelerometers, while the last one is a temperature sensor. The subscribers can either subscribe a single client among the accelerometers, as is the case with Subscriber1, or they can subscribe to all the accelerometers’ activity by using “#” sign, as done by Subscriber2. The Subscriber3 goes even further and subscribes to all the sensors, regardless of their type. Based on the topic subscribed, the relevant data is sent to all the subscriber by the broker.



**Figure 7: MQTT Architecture**

### *Quality of Service*

One of the best features of MQTT is Quality of service (QoS). The QoS levels must be specified for every message sent through MQTT and they determine how each MQTT message is delivered. There are three QoS options for message delivery [28]:

- QoS 0 (At most once) - Messages are delivered according to the best efforts of the operating environment. However losses can occur.
- QoS 1 (At least once) - Messages are assured to arrive but duplicates can occur. An acknowledgement is sent on receiving every message
- QoS 2 (Exactly once) - Message are assured to arrive exactly once.

### *Mosquitto*

This thesis uses Eclipse Mosquitto™, which is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1 and 3.1.1. Mosquitto is an [iot.eclipse.org](http://iot.eclipse.org) project [29].

It is easily installed on Raspberry pi that uses Debian.

### *Advantages of MQTT over HTTP*

For Internet of Things application, the MQTT protocol has several advantages over HTTP protocol [30].

- Allows pushing a change to many clients owing to the publish-subscribe model
- Has binary format, unlike HTTP that has a text based format requiring more bandwidth
- Has built-in support for QoS
- HTTP is request-response based, but MQTT, being publish-subscribe, is event-driven
- For HTTP, the device that acts as a host needs web server installed which takes battery. This is not the case in MQTT as it uses the publish-subscribe pattern

### 4.6. Raspberry Pi

Raspberry Pi is a series of small, relatively cheap, single-board computers. It is approximately credit-card sized, representing the standard mainline form-factor, but at the same time, has a 32-bit ARM processor and uses a Debian distribution of Linux for its default Operating System (OS). It can be programmed with python or any other language that will compile for ARMv7.

The setup used for this thesis utilizes Raspberry Pi 2 Model B V1.1 loaded with Raspbian Jessie image. It is essentially a system-on-chip (SoC) with a 900MHz quad-core ARM Cortex-A7 CPU and 1GB RAM. The connectivity comes through ports like USB, Ethernet, etc. Furthermore, a USB Wi-Fi dongle can be used to connect it to the local network or Internet.

An MQTT broker can handle up to thousands connected MQTT client at the same time [31]. Keeping in consideration Raspberry Pi's processing capabilities, scalability would not be an issue in this particular setup.

### 4.7. Wi-Pi™ WLAN USB Module

There are two WLAN operating in this IoT chain. The first one is created by the TP Link Access Point to connect the CC3100 to Raspberry Pi. The second is created by the Wireless router that is directly connected to the Internet. Since the Ethernet port of Raspberry Pi is already connected to the Access Point, the connection to router is achieved by using a special WLAN USB module called WiPi™. Wi-Pi is a high performance, cost effective USB module that connects Raspberry Pi to a WLAN. Here are some of its special features:

- It uses the latest international wireless CCA air channel detection technology, enhancing wireless performance.
- It is compatible with Raspberry Pi Model A/B/B+
- Natively supported by the Raspbian Wheezy Linux operating system distribution onwards. Since we are using Raspbian Jessie, it is supported.
- It can be plugged into USB 2.0 interface and supports 64bit/128bit/152bit WEP encryption.

#### 4.8. ThingSpeak™

ThingSpeak is an IoT platform that enables you to collect, store, analyze, visualize, and act on data from sensors or actuators, such as Arduino®, Raspberry Pi™, BeagleBone Black, and other hardware. For example, with ThingSpeak™ you can create sensor-logging applications, location-tracking applications, and a social network of things with status updates, so that you could have your home thermostat control itself based on your current location. [32]

The data collection and analytics features serves as a bridge, collecting data from low level devices such as temperature and pressure sensors and converting it into a form that is more convenient for a human to do further analysis. ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks. [33] Allowing ThingSpeak users to analyze and visualize uploaded data using Matlab without requiring the purchase of a Matlab license from Mathworks. However, it is not just confined to providing data for analysis. If configured, ThingSpeak can also act on the data.

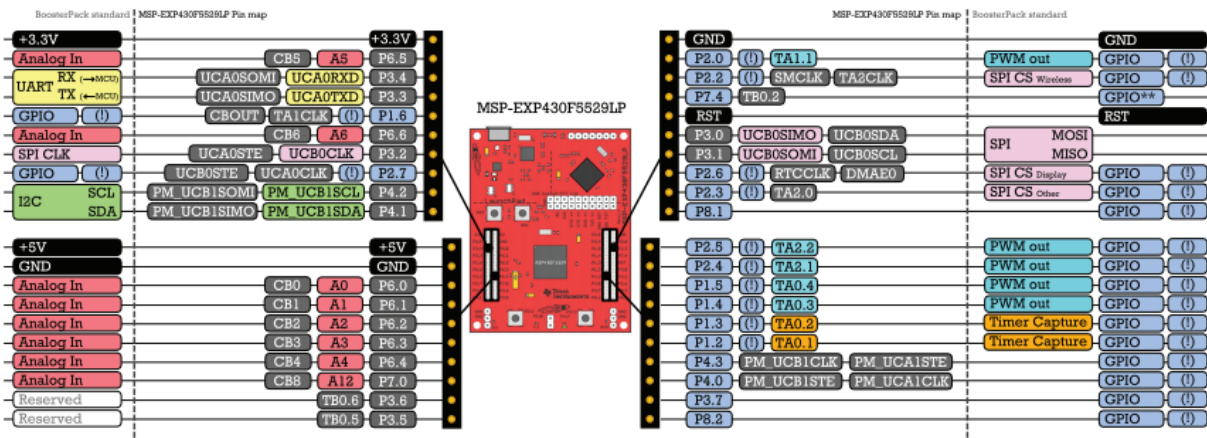
It provides the user with APIs to store and retrieve data from things using HTTP over the Internet or via a Local Area Network. The data is organized in *channels* which contains data fields, location fields, and a status field.

## 5. Implementation

Making use of the hardware and software components as well as the communication devices described above, the IoT chain has been implemented. This section describes the circuit design as well as the software implementation.

### 5.1. Hardware

The hardware implementation consists of both wired as well as wireless connections between all the devices. The first thing to consider is the pin availability of the microcontroller and allocate the pins for connections in the design. The MSP-EXP430F5529LP LaunchPad Development kit is equipped with 40-pin BoosterPack expansion headers. *Figure 8* shows the pins contained in the headers as well as the functionality associated with them.



**Figure 8:** MSP-EXP430F5529LP LaunchPad Development kit header. Reproduced from [34]

The SimpleLink™ Wi-Fi® CC3100 BoosterPack can be stacked on the expansion header on either top or the bottom of the MSP-EXP430F5529LP LaunchPad Development kit. For connecting accelerometer using female jumper wires, it is more convenient to have the booster pack stacked below the Launchpad kit.

Texas Instruments provides a wonderful tool [35] to check pin compatibility between the vast variety of available Launchpad and BoosterPack.



Compatible? **YES**

Reason: Selected combination is **Compatible** (with warnings). [5]

[Buy Now](#)
[Cloud Tools](#)
[Share My Combo](#)

- My Selections | All | LP Only

BP1

SimpleLink Wi-Fi CC3100 BoosterPack

+

LP

MSP-EXP430F5529LP

▼ Connector #1 (40 Pins)

|    |    |    |    |
|----|----|----|----|
| 1  | 21 | 40 | 20 |
| 2  | 22 | 39 | 19 |
| 3  | 23 | 38 | 18 |
| 4  | 24 | 37 | 17 |
| 5  | 25 | 36 | 16 |
| 6  | 26 | 35 | 15 |
| 7  | 27 | 34 | 14 |
| 8  | 28 | 33 | 13 |
| 9  | 29 | 32 | 12 |
| 10 | 30 | 31 | 11 |

**Pin Summary**

Pin: **#28** > Function: **analog** > Instance: **A12** > GPIO: **P7.0**

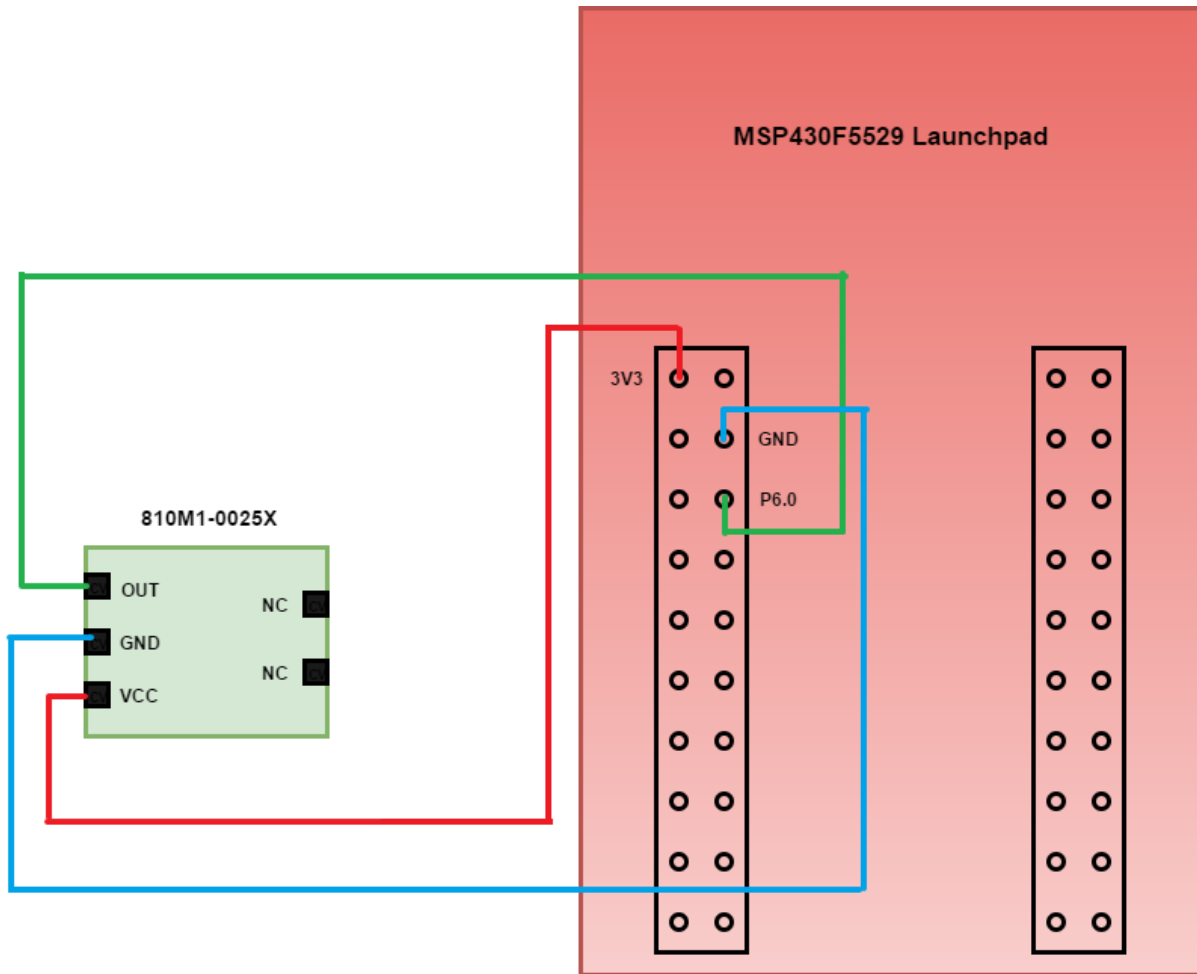
**Pin Status**

Available

**Figure 9:** MSP430F5529 and CC3100 Pin Compatibility. Source [35]

The *Figure 9* shows that the SimpleLink™ Wi-Fi® CC3100 BoosterPack is compatible with MSP-EXP430F5529LP LaunchPad Development kit. It also shows more details about the pin function, instance and whether it is a General Purpose Input Output (GPIO) pin or not. The green color on the pin denote that they are used for the communication between the two boards. The remaining pins can be used for connecting to other components.

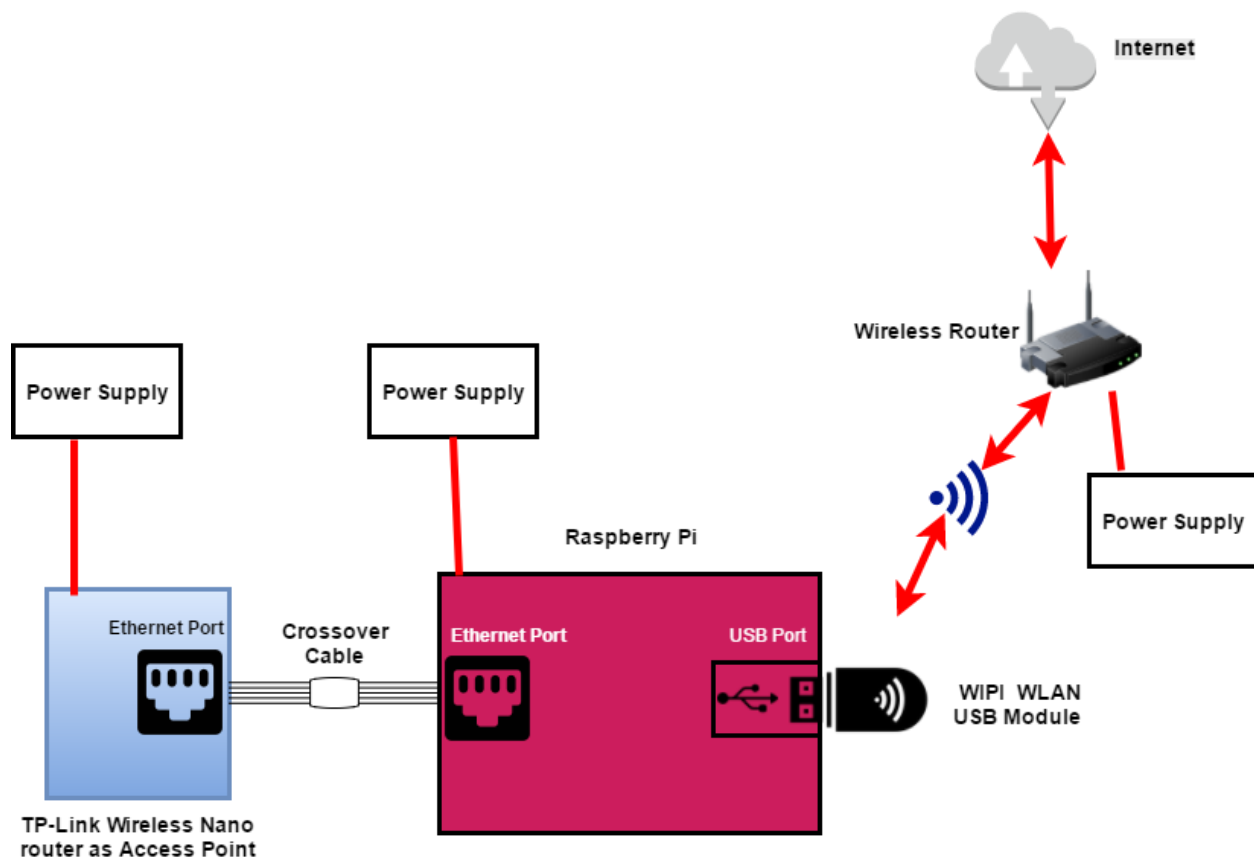
Now the connection between the accelerometers and the MSP-EXP430F5529LP LaunchPad Development kit is done. Since both the accelerometers have different parameters and electrical requirements, different circuits were designed. The first accelerometer 810M1-0025X has built-in amplifier and thus doesn't need any external components for interfacing. It was directly connected to the MSP430 pins. *Figure 10* shows the wired connections for the same.



**Figure 10:** MSP430 and Accelerometer 810M1-0025X connection

The second accelerometer 805-0050 started giving erroneous readings, most likely due to some faults, so the first one was used for both the cases.

The next step is to connect the Access point to Raspberry Pi through wired connection and Raspberry Pi to the Wireless Router using WIPI WLAN USB Module. This is the most important link as it provides Internet connectivity to the MQTT broker. *Figure 11* shows the devices connections and how the IoT chain is finally connected to its terminal state, i.e. Internet.



**Figure 11: Internet Connectivity**

## 5.2. Software

The software is the backbone of this application and it accounts for the majority of the time spent on designing this IoT chain. There are more than one units as well as components on different stages running different code, but they act together to ensure the sensor data reaches the cloud platform.

Before beginning any development, there are some pre-requisites:

- Update firmware of all CC3100 booster packs using a software called UniFlash. An additional hardware called CC31XXEMUBOOST is required for this purpose
- Install CC3x00ServicePack-1.0.1.6-2.7.0.0 on CC3100 using the same setup as in the previous step

- Install Raspbian Jessie on Raspberry Pi which is a Debian-based OS
- Install Mosquitto MQTT broker on Raspberry Pi
- Install Paho MQTT Python client on Raspberry Pi

After doing the above mentioned steps, the development was done stage by stage. The application can be divided into two parts according to the stages in the chain:

- Data Acquisition, Processing and Publishing
- Broker

### 5.2.1. Data Acquisition, Processing and Publishing

The firmware for this specific application is written in C and resides on the MSP-EXP430F5529LP. The MSP-EXP430F5529LP LaunchPad Development kit provides easy programming and debug capabilities by simply plugging the device to the USB port of a host computer.

The development is done on Texas Instruments' Code Composer Studio™ version 6. Code Composer Studio (CCStudio or CCS) is an integrated development environment (IDE) to develop applications for Texas Instruments (TI) embedded processors. Code Composer Studio is primarily designed as for embedded project design and low-level (baremetal) JTAG based debugging. However, the latest releases are based on unmodified versions of the Eclipse open source IDE, which can be easily extended to include support for OS level application debug (Linux, Android, and Windows Embedded) and open source compiler suites such as GCC. [36] Code Composer Studio comprises a suite of tools used to develop and debug embedded applications. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features. [37]

As described in the earlier chapters, this stage has the following four functions and performed in the same order:

1. Configure the CC3100 and connect to the Access Point through Wi-Fi
2. Read Accelerometer data and perform Analog to Digital Conversion
3. Process the data by removing the offset, find the RMS value, Acceleration(g) and Crest factor
4. Send the data over MQTT Protocol to the broker connected to the Access Point

Here is a description of the constituents of the software.

This software is developed on top of the Getting Started with WLAN Station example project [38] provided with CC3100 SDK so that the DAPPU is connected to the Access point before any further processing or communication begins. It contains the basic configuration like turning off watchdog timer, setting the system clock etc. The code is structure so that the `configure_and_connect()` function contains what is provided by the Getting Started with WLAN Station to configure the CC3100 device and connect to the access point.

#### *The Main() function*

This is the entry point for the application and contains function call for the following:

- basic setup of MSP430F5529
- configuring CC3100 and connect to the access point
- setup Analog to Digital Conversion (ADC) and Direct Memory Access (DMA)
- inside an infinite loop do the ADC conversion, process data and send it over MQTT

The microcontroller stays in the Low Power mode LPM0 until 1k samples are samples are collected by the DMA.

#### *Configure CC3100 and Connect to Access Point*

The configuration and connection to Access point (AP) was made possible by the CC3100 Getting Started with WLAN Station application. It configures the CC3100 in WLAN station mode and then connects to the access point using the SSID, security and password specified in the `sl_common.h` file as `SSID_NAME`, `SEC_TYPE` and `PASSKEY` respectively. Getting Started with WLAN Station communicates w/ CC3100 over SPI, which is its default.

*Figure 12* shows the serial port output of the `configure_and_connect()` function. The step in the original example, in which it attempts to connect to the Internet has been removed as in this case only the connection to the access point is required.

```
Getting started with station application - Version 1.2.0
*****
Device is configured in default state
Device started as STATION
Connection established w/ AP and IP is acquired
Pinging...!
Device successfully connected to the LAN
```

**Figure 12:** Configure CC3100 and Connect to AP

### *Setting up ADC and DMA*

Once the connection has been made, the ADC setup is done. It does the following steps:

- Enable the 12-bit ADC core
- Select multiple sample and conversion
- Set Internal ref as 1.5V
- Set sampling time
- Use sampling timer and select Pulse Sample Mode
- Select ACLK (32.768 kHz) as the clock source
- Divide the clock by 2 so that the effective clock is now  $ACLK/2$ , i.e. around 16kHz. This is based on the Nyquist criterion
- Select ADC option on P6.0 on which the accelerometer is connected

The Enable conversion and Start conversion is done inside the infinite loop of the main function.

The results of ADC are moved from the memory to an array using Direct Memory Access (DMA). The DMA is configured to work in Repeated single transfer incrementally until 1k samples have been moved to the array. During this period, the controller is in the Low power mode LPM0. On reaching the size, which is 1k, it sets a flag causing its Interrupt Service Routine to disable the ADC and exit the low power mode. The ADC is re-enabled in the next cycle of the infinite loop.

## *Data Processing*

In this function, the mean of the collected 1k samples is calculated and then subtracted from each value in the array so that the offset is removed. After that the peak, root mean square (RMS) value is calculated. The Acceleration and Crest Factor is calculated using these values.

## *Configure MQTT and send the data*

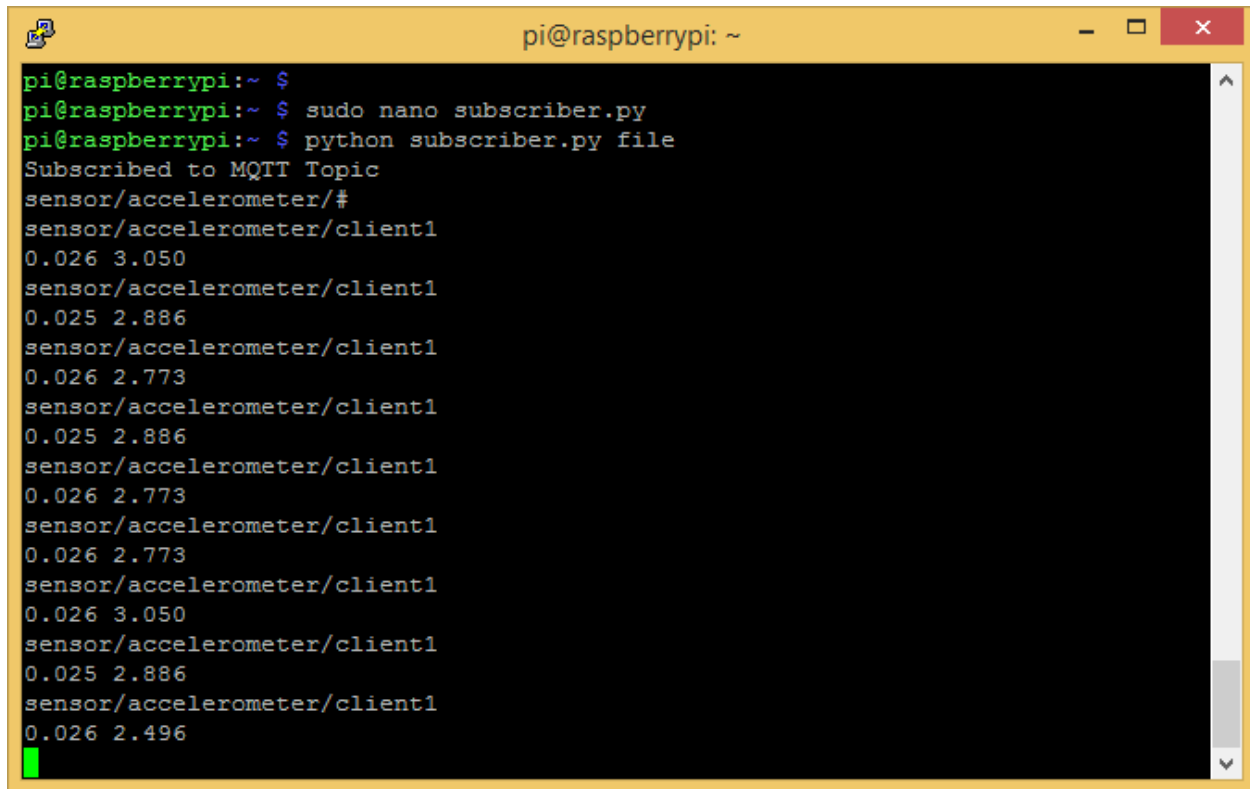
The MQTT component is added to the software by using Paho Embedded MQTT C/C++ Client Libraries [39]. This function makes the MQTT packet, creates the socket connection and then finally sends it to the MQTT Broker [40]. The library allows for setting QoS level, which in this application has been 0.

### 5.2.2. Broker

The Broker, which is Raspberry Pi and runs Raspbian Jessie, has Mosquitto installed and running on it. The software is one python file named `subscribe_and_update_ThingSpeak.py` uses Eclipse Paho MQTT Python client library, which implements versions 3.1 and 3.1.1 of the MQTT protocol.

The Python file, when run, connects to the MQTT broker using the IP address and port number specified, and subscribes to a topic. On arrival of a message from the publisher, it connects to the ThingSpeak channel and updates the fields. This is done by using the Write API Key for the channel for the particular channel as described in its settings.

Figure 13 shows the `subscribe_and_update_ThingSpeak.py` in working. The data being printed on the console is being uploaded to the ThingSpeak channel at the same time.

A terminal window titled "pi@raspberrypi: ~" with a yellow title bar. The terminal output shows the execution of a Python script that subscribes to an MQTT topic and publishes accelerometer data. The data consists of timestamps and two numerical values.

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ sudo nano subscriber.py
pi@raspberrypi:~ $ python subscriber.py file
Subscribed to MQTT Topic
sensor/accelerometer/#
sensor/accelerometer/client1
0.026 3.050
sensor/accelerometer/client1
0.025 2.886
sensor/accelerometer/client1
0.026 2.773
sensor/accelerometer/client1
0.025 2.886
sensor/accelerometer/client1
0.026 2.773
sensor/accelerometer/client1
0.026 2.773
sensor/accelerometer/client1
0.026 3.050
sensor/accelerometer/client1
0.025 2.886
sensor/accelerometer/client1
0.026 2.496
```

**Figure 13:** Subscribed to Topic and publishing to ThingSpeak



## 6. Results

To verify that the IoT chain is functioning as intended, an experimental setup was used as described in the following section. This chapter summarizes the test environment, the output of the experiment, as well as analysis of the data displayed.

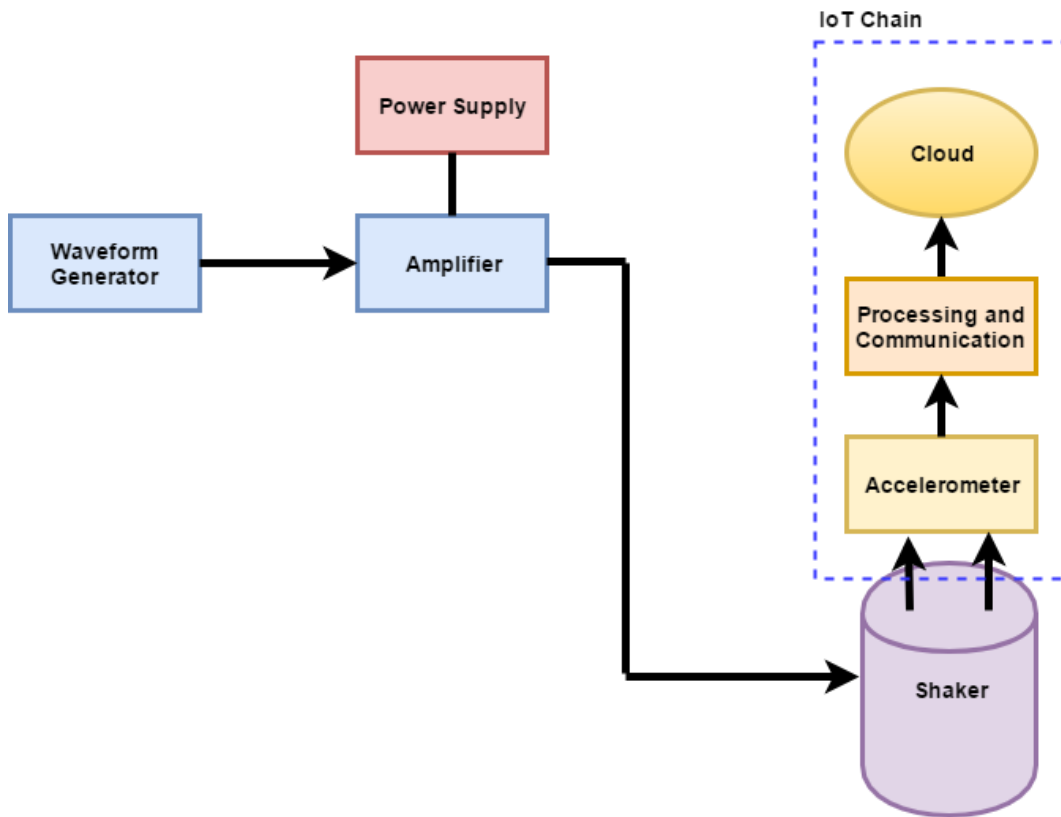
The initial plan was to utilize two accelerometers, 810M1-0025X and 805-0050, and thus two separate IoT chains running in parallel. But since the second accelerometer, 805-0050, eventually became faulty, the tests were done using only one IoT chain.

However this doesn't affect the test and validation as before the failure of the second accelerometer, some tests were already performed to see if the data is uploaded to Thingspeak by two separate chains operating at the same time. The results are described later in this chapter.

### 6.1. Experimental Testbench Vibration device Setup

The experimental setup consists of a waveform generator, an amplifier which is designed specifically for inductive loads and a special vibrating device called a Shaker. *Figure 13* shows the block diagram of the experimental setup. A generic representation of the IoT is shown in this block diagram as it has already been described in details in the previous chapters. The Accelerometer is mounted using nuts and screws on top of the Shaker with the acceleration axis facing downwards. This is to ensure that the accelerometer vibrates along its own axis.

The experiment consists of providing different waveforms to the Shaker and observing the output at the associated ThingSpeak channel. The variations in the waveforms is with respect to frequency, amplitude and waveform type. *Table 2* shows these parameters and their individual values for which the output is observed. All the combinations of the values shows in this table were analyzed and the impact of parameters on the output is described in the next section.



**Figure 14:** Test Setup for Vibration Measurement

The waveform generator used in this setup is Agilent 33220A which is a 20 MHz function generator with built-in arbitrary waveform and pulse capabilities. It is connected to amplifier which is designed specifically for the inductive loads. The resulting signal is then fed to the Shaker, which is LDS V406 by Brüel & Kjær.

**Table 2:** Table of Parameters used for Experiment

| Frequency(Hz) | Amplitude(mVpp) | Waveform Type |
|---------------|-----------------|---------------|
| 20            | 100             | Sine          |
| 50            | 150             | Square        |
| 100           | 200             | Ramp          |
| 1k            |                 |               |

mVpp = milli Volts peak-to-peak

The LDS V406 is a permanent magnetic shaker ideal for vibration testing of components, small assemblies or modal and structural analysis. The shakers' efficient armature design enables it to deliver impressive peak forces and accelerations over a wide frequency range. It can impart maximum forces up to 196 N (44 lbf) [41]. It has the following features that makes it well suited for vibration analysis application:

- Wide frequency band combined with high peak forces
- Low mass, high performance armature construction
- Powered by compact, quiet and energy efficient amplifiers
- Robust, lightweight suspension system provides excellent torsional and traverse stiffness with minimal impact on system acceleration

Among its various applications, the most relevant are as a velocity transducers or high-speed actuators for accelerometer calibration, and machinery condition monitoring and vibration measurement.

## 6.2. ThingSpeak Results

The results are presented in a form so as to observe the impact of amplitude, frequency and the shape of the waveform on the Acceleration (g) and Crest Factor of the Accelerometer data.

The Acceleration, measured in 'g', is calculated using the RMS value of 1000 samples of Accelerometer data after the Analog to Digital Conversion. An offset is first subtracted from all the raw values before calculating the RMS. Acceleration is directly proportional to the RMS value.

Crest factor is calculated by dividing the peak value by the RMS value. Both the RMS value and Crest factor, as calculated for the 1k samples are sent over the MQTT to the Broker and eventually to the ThingSpeak cloud.

The output of the IoT chain is the data displayed on the channel of the ThingSpeak platform. It is organized in the form of charts for different fields which is basically a graph between the parameter and time. Since there are two parameters there are two graphs, between Acceleration and Time, and between Crest Factor and Time respectively.

### 6.2.1. Multiple Publishers Simultaneous Operation Test

The first test was done to verify that data from two different accelerometers, 810M1-0025X and 805-0050, is uploaded to ThingSpeak. The accelerometers publish data and the Broker pushes this data to the ThingSpeak channel at the same time.

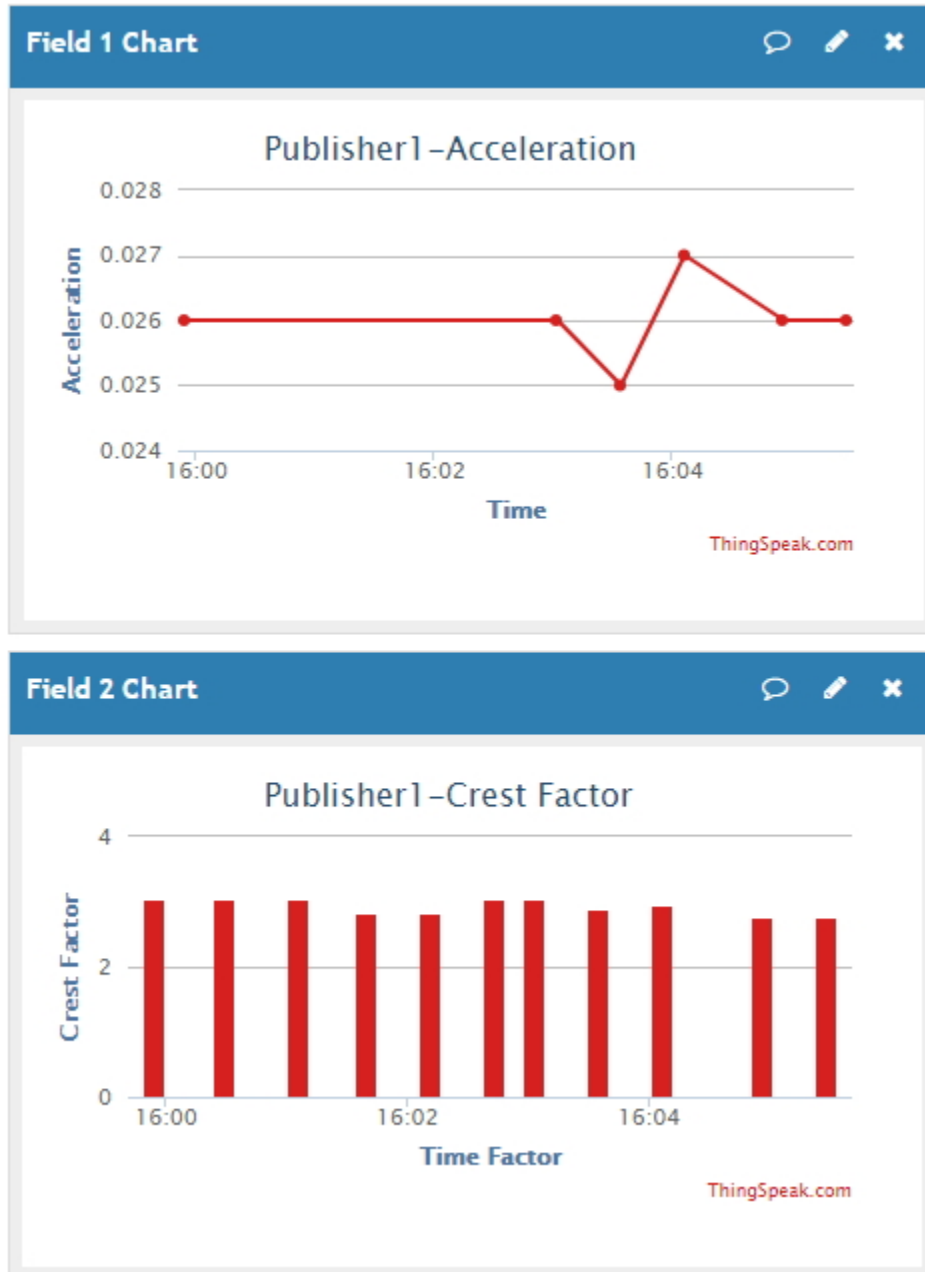
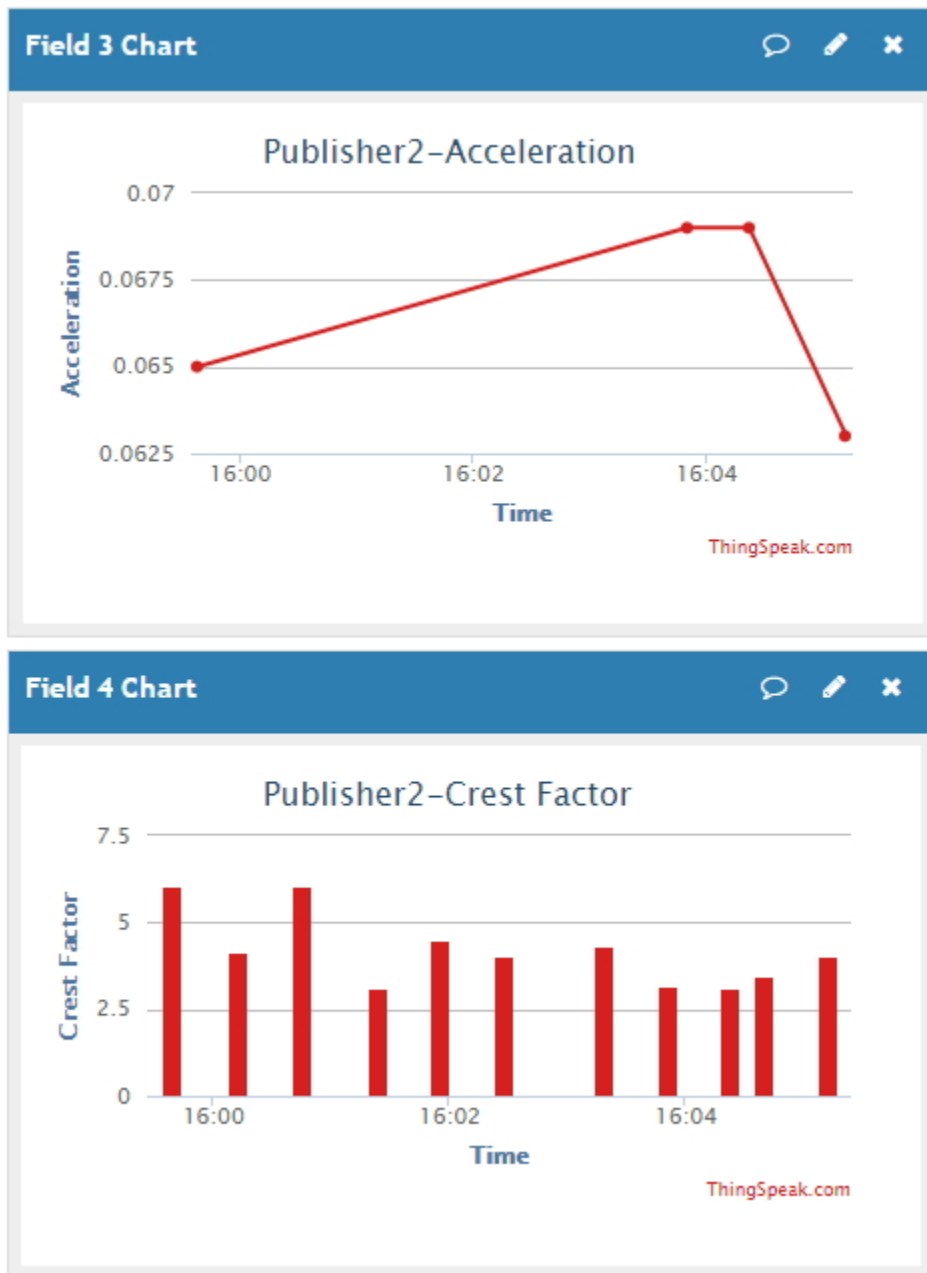


Figure 15: ThingSpeak out for Accelerometer 810M1-0025X

The channel has four fields. The *Figure 14* shows Field 1 and Field 2 for Acceleration and Crest Factor respectively for Publisher 1, which is the accelerometer 1(810M1-0025X). Similarly, *Figure 15* shows Field 3 and Field 4 for Publisher 2 i.e. accelerometer 2(805-0050).



**Figure 16:** ThingSpeak out for Accelerometer 805-0050

### 6.2.2. Experimental Testbench Results

The results for the experimental setup consists of various ThingSpeak charts for all combinations of all Frequency, Amplitude and Waveform shape. The output contains both the Acceleration as well as Crest Factor data.

The readings were taken with increasing amplitudes 100, 150 and 200, while keeping the frequency as well as the shape of the waveform constant. The *Figure 17* to *Figure 24* show the Acceleration and Crest Factor output for all the frequencies and the shapes chosen as sine, square and ramp wave. In the Acceleration output, the readings for different Amplitude levels are shown with a blue marking on the chart itself. The corresponding Amplitude is also mentioned as text.

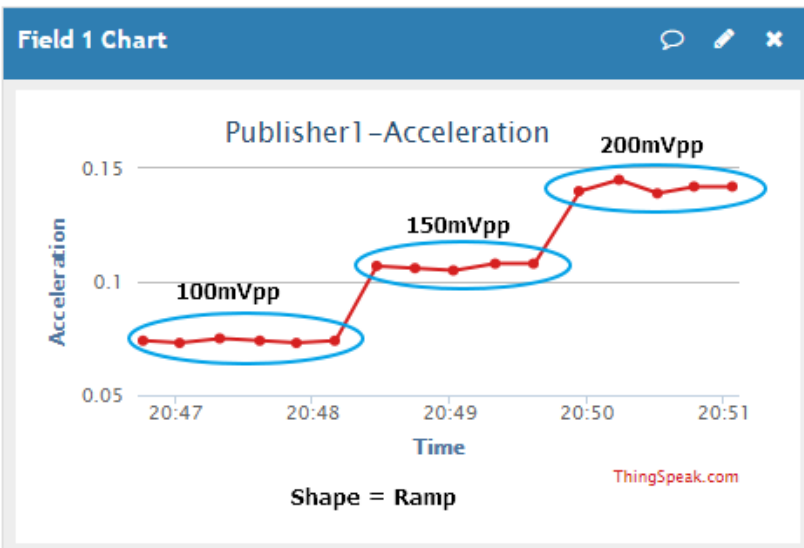
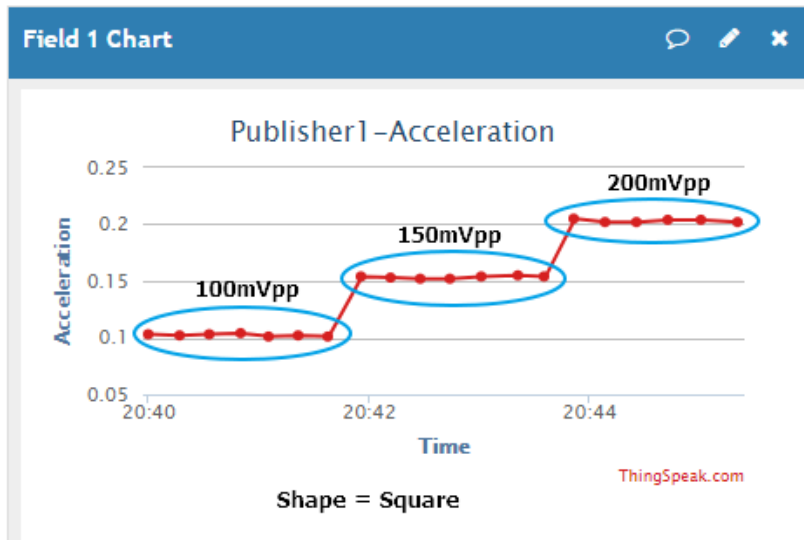
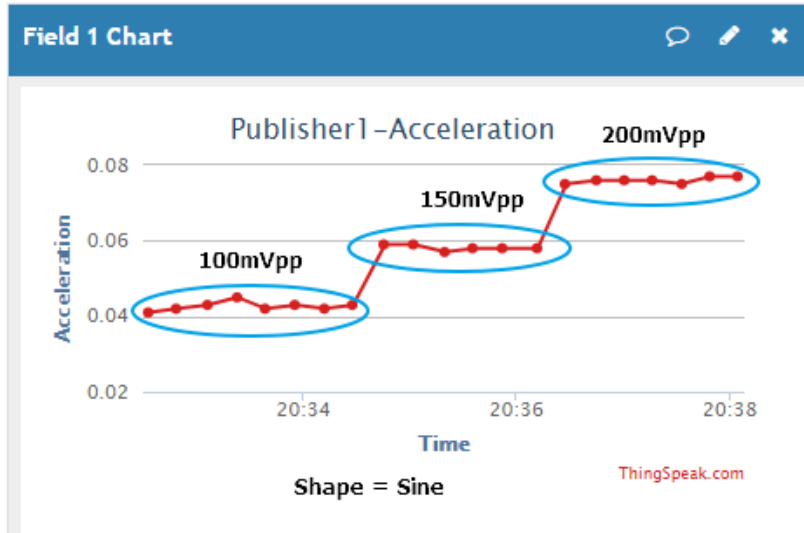
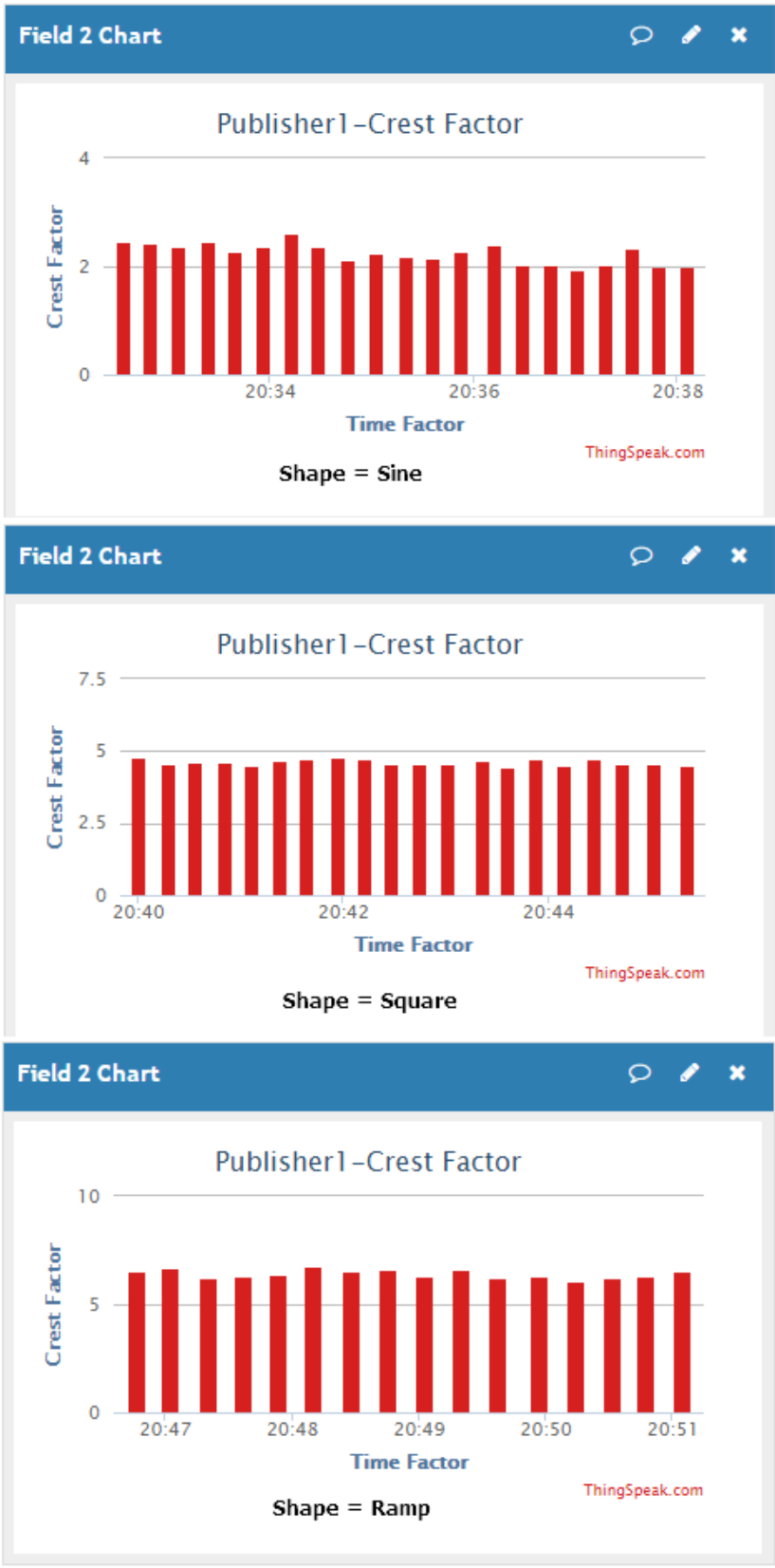


Figure 17: Acceleration output for 20Hz



**Figure 18:** Crest Factor output for 20 Hz



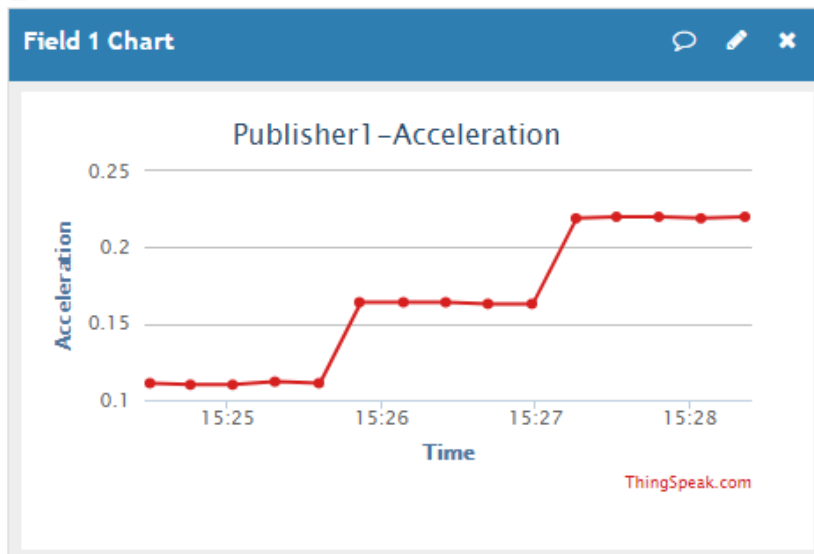
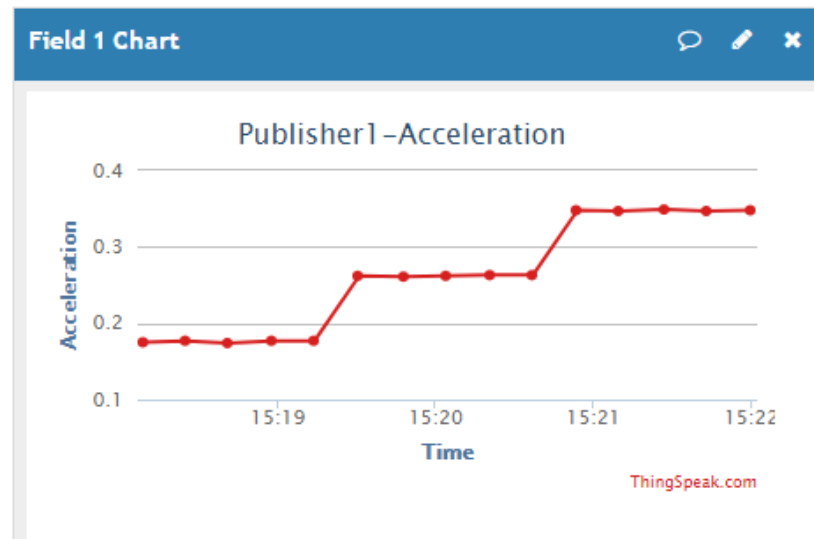
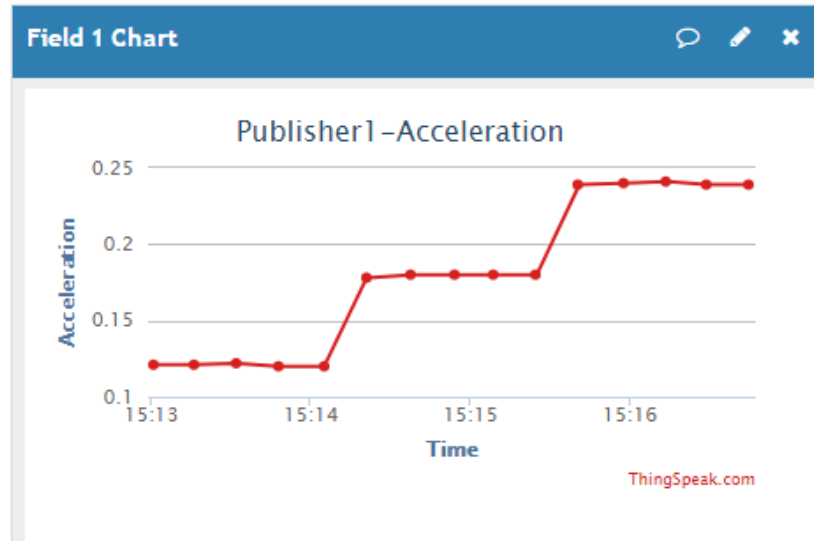
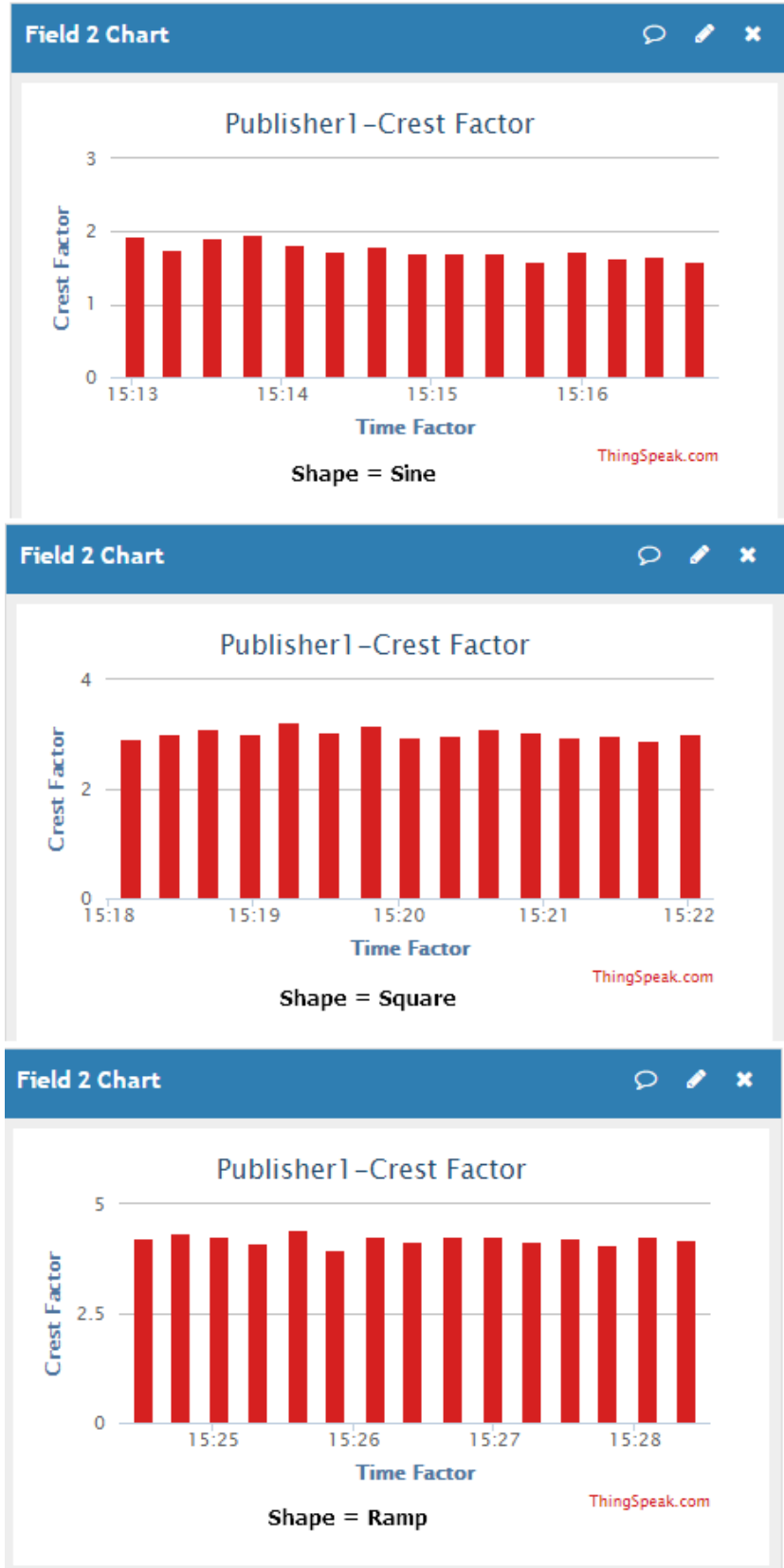


Figure 19: Acceleration output for 50Hz



**Figure 20:** Crest Factor output for 50 Hz

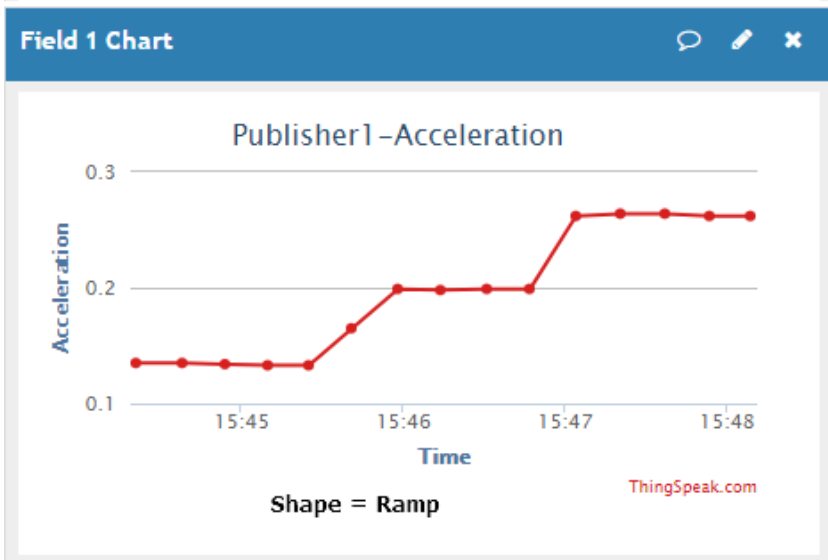
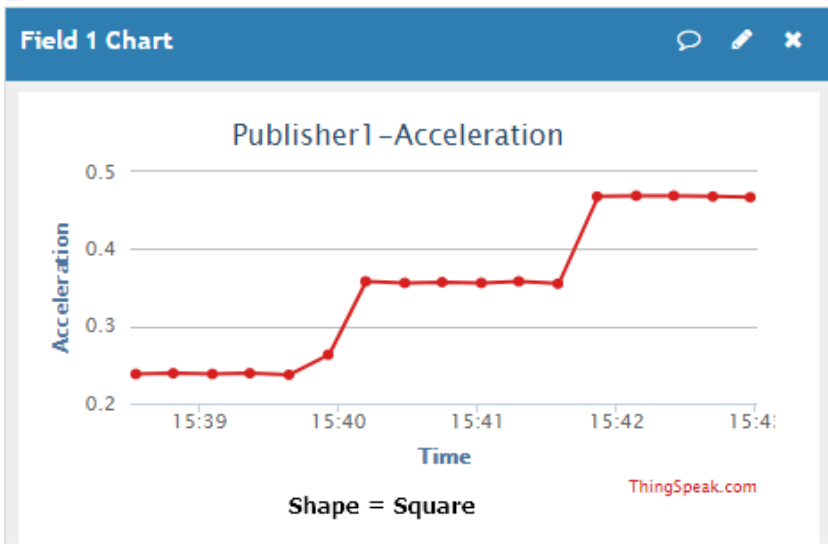
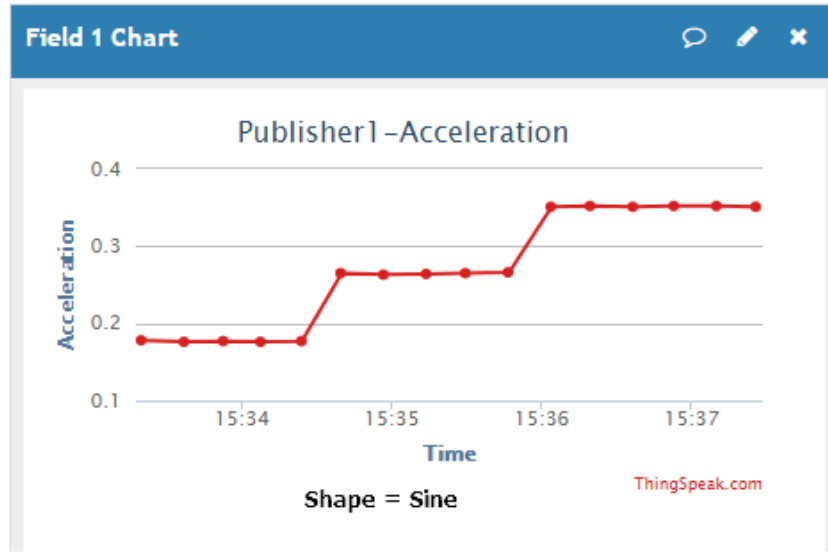
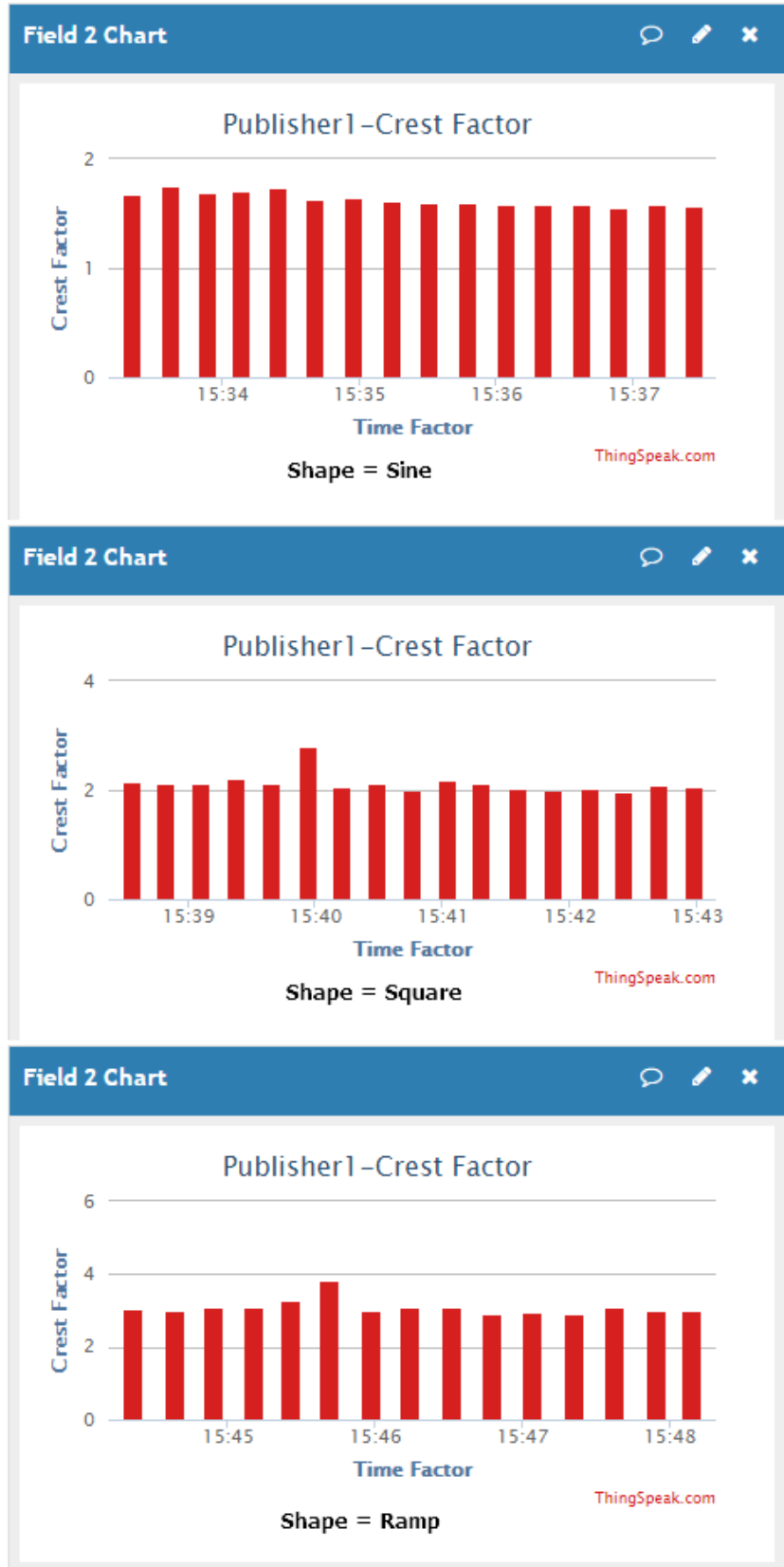


Figure 21: Acceleration output for 100Hz



**Figure 22:** Crest Factor output for 100 Hz

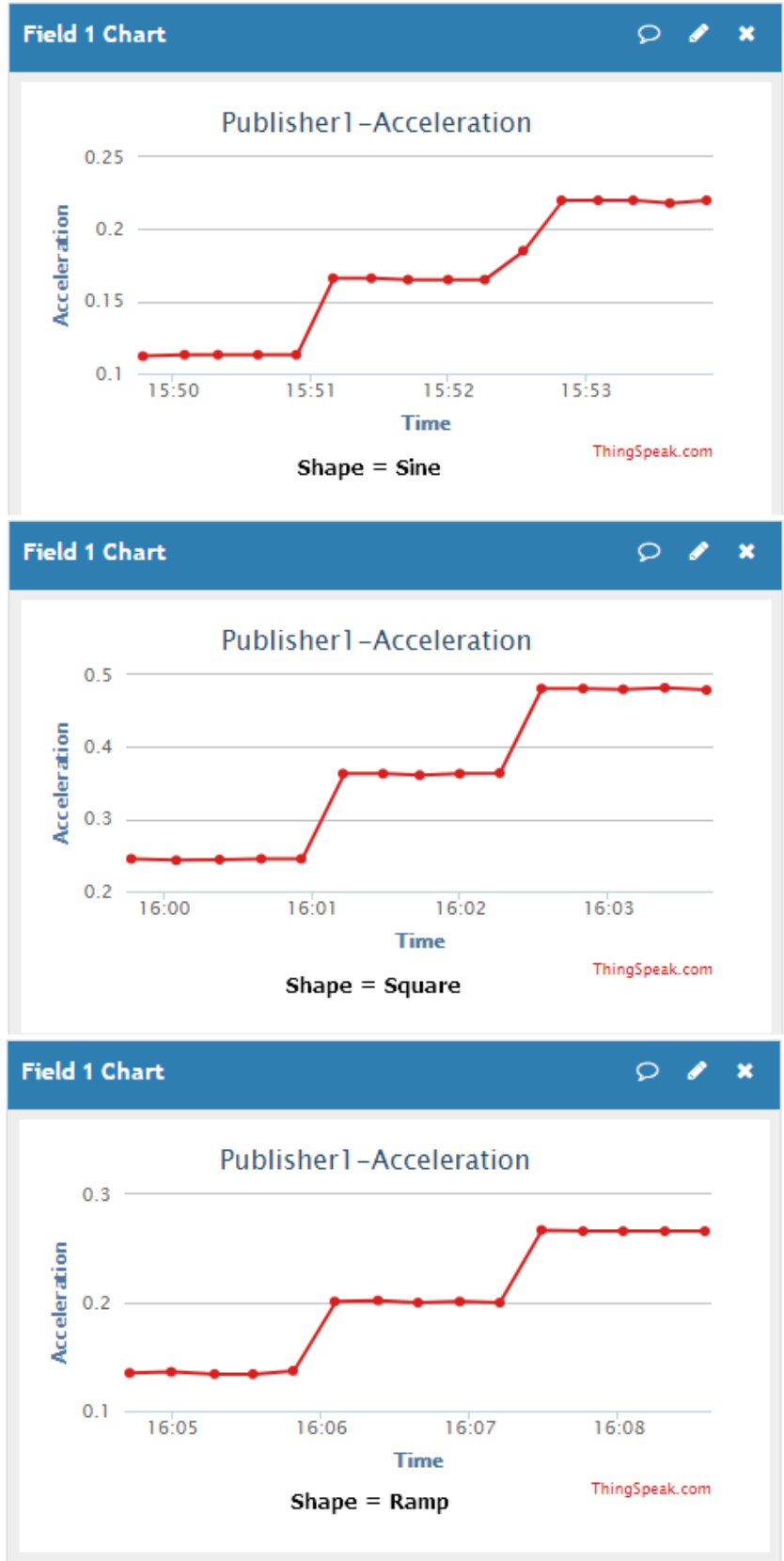


Figure 23: Acceleration output for 1 kHz

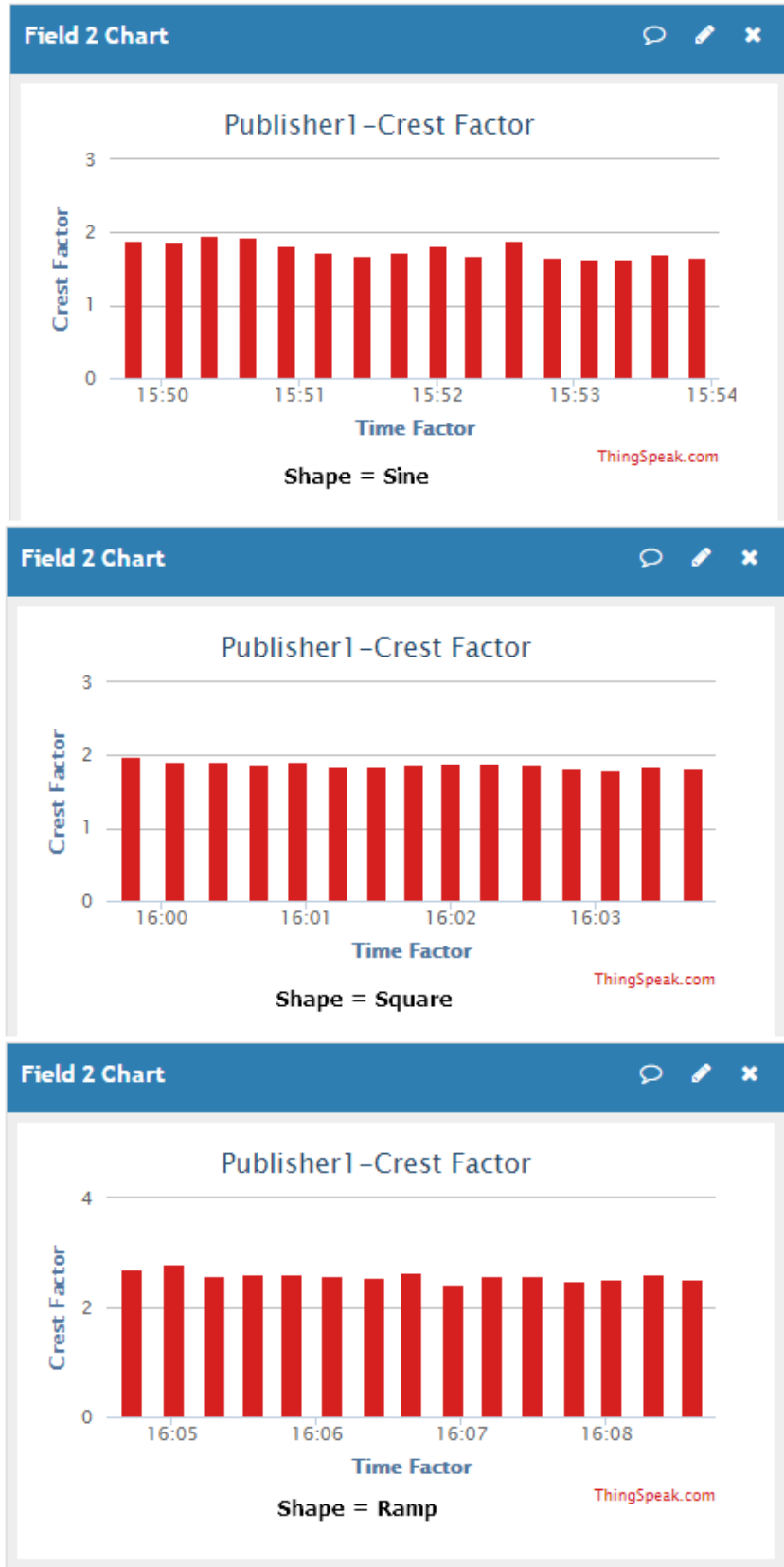


Figure 24: Crest Factor output for 1 kHz

### 6.2.3. Additional Experiments

Another vibration device was tested for measuring uncontrolled vibrations from rotating machinery. The device is Proxxon micromot 50. An artificial imbalance was introduced to the rotary part of the device and the accelerometer was tied to it through a rubber band.

This device is different than the Shaker as it has a DC electrical motor and a rotating machinery. The Shaker, on the other hand, has no rotating part and the vibrations are produced by an electrodynamic linear actuator. However, vibrations from shaker are controllable, while this device generates uncontrolled vibrations.

Here are t

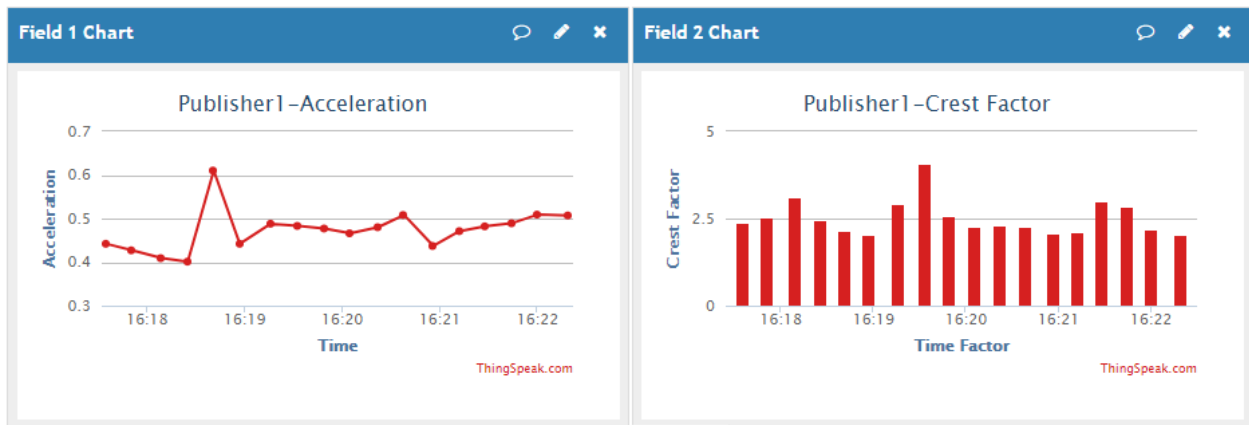


Figure 25: Uncontrolled Vibration output 1

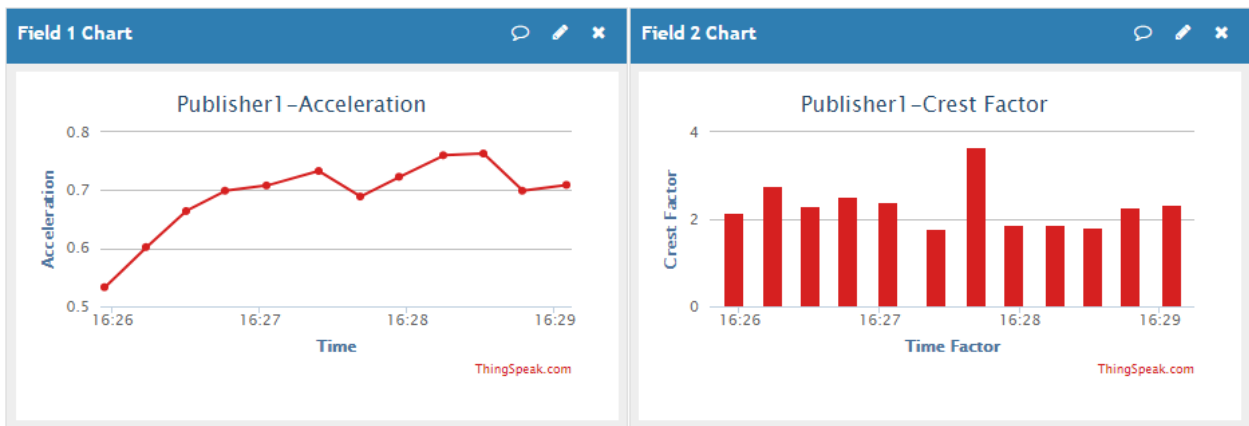
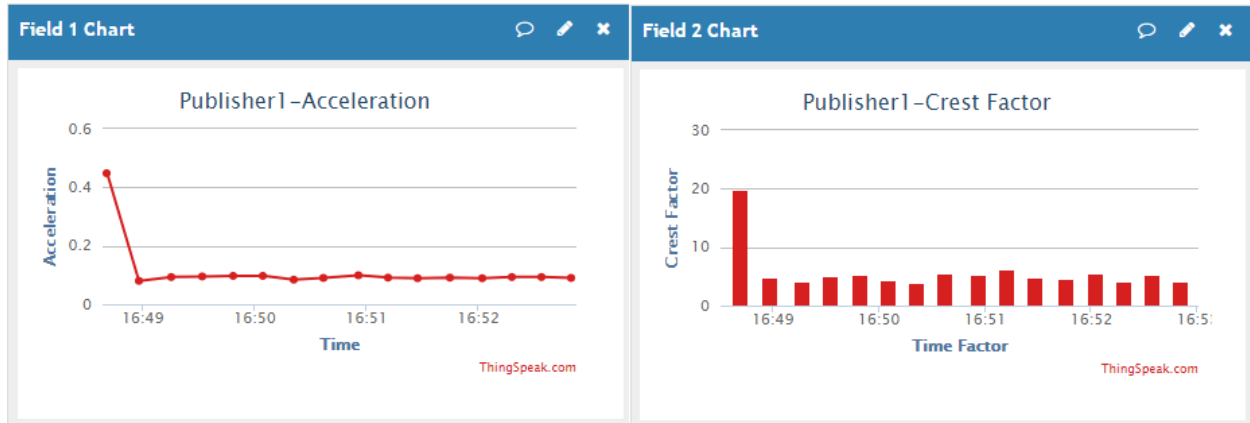


Figure 26: Uncontrolled Vibration output 2



**Figure 27: Uncontrolled Vibration output 3**

### 6.3. Analysis

The analysis is done based on all the three variable parameters.

#### *Effect of Amplitude*

It can be noticed from the graphs that an increase in Amplitude increases the Acceleration, irrespective of the Frequency or Waveform shape. The Acceleration is calculated from RMS value of the signal which represents the energy content of the signal. So increasing the amplitude increases the energy content, thereby increasing the Acceleration.

For a given Frequency and Waveform shape, the Crest Factor doesn't show any correspondence to the change in Amplitude.

#### *Effect of Waveform shape*

For a given Frequency, the Square wave has more Acceleration than both Sine and Ramp throughout the signal.

Sine and Ramp on the other hand have different results for different Frequency.

For 20 Hz, Acceleration of Ramp > Acceleration of Sin

For 50 Hz, Acceleration of Sine slightly higher than Acceleration of Ramp

For 1 kHz, Acceleration of Ramp slightly higher than Acceleration of Sine

The Crest factor shows a consistent pattern in this case. For any given frequency, it has the following

Crest Factor (Ramp) > Crest Factor (Square) > Crest Factor (Sine)



### *Effect of Frequency*

The Acceleration increases when Frequency is increased 20 Hz to 100 Hz. However, there is not much considerable difference between the Acceleration at 100 Hz and at 1 kHz. So it appears that the effect of Frequency on Acceleration is not much beyond a certain frequency limit.

For the Crest Factor it is noticed that it is inversely proportional to the Frequency of the signal. As the frequency increases, Crest Factor reduces.

## 7. Conclusion

This chapter contains the conclusions drawn from working with the concept and the actual results of the thesis. The thesis began with a study of various Condition Monitoring techniques and about why Vibration Analysis is so popular among all of them. Then some research was done in the area of Internet of Things, its various components and applications.

To demonstrate a proof of concept for Condition Monitoring, specifically Vibration Analysis, using Internet of Things, an IoT chain was designed using hardware, software and communication infrastructure. The sensor chosen was an accelerometer with large bandwidth and designed especially for applications like Machine health monitoring. The accelerometer data was processed in a low power microcontroller MSP430F5529 by Texas Instruments. The accelerometer's analog output was sampled and converted using ADC. The data samples were collected and saved in the memory, while making good use of the interrupts and low power modes of the microcontroller whenever possible. After collecting 1k samples, the data was processed and parameters which are useful in Vibration analysis, namely Acceleration (g) and Crest Factor, were extracted. The data after this processing stage was in the format ready to be displayed without any further processing needed.

The Wi-Fi booster pack stacked on to the microcontroller development kit allowed for connectivity with WLAN created by an access point. The communication happened over MQTT protocol which has a publish-subscribe pattern and is well suited for IoT application, especially when there are multiple data publishing nodes as well as multiple subscribers interested in data from one or more publishers. The data was sent as MQTT packets from MSP430F5529 development kit + Wi-Fi Booster pack, to the MQTT broker, which was Raspberry Pi in this case. The MQTT broker took care of the delivery of the data to the correct subscribers, using Topics. Each publisher publishes data on a specific Topic, and the subscribers subscribed to that Topic receive the data from the Broker when it is published.

The final stage was to upload this data to the Cloud for which ThingSpeak has been used. It is an IoT platform that enables you to collect, store, analyze, visualize, and act on data from sensors or actuators. The Python code running on the Raspberry Pi subscribes to a Topic on the Mosquitto MQTT broker installed and running on Raspberry Pi, and updates the ThingSpeak channel with the parameters to be displayed as graphs on different fields.

To test multiple publishers' communication, the two accelerometers published Topics in the form "sensor/accelerometer/Publisher1" and "sensor/accelerometer/Publisher2" respectively. On changing the subscribed Topic on the Python file to "sensor/accelerometer/Publisher1" displayed data from the first accelerometer only. However, when the Topic was set to "sensor/accelerometer/#", the data from both the accelerometers was uploaded to the channel. Hence, the MQTT implementation ensured that the data from different accelerometers is communicated through the chain separately to the cloud platform. An end user, whether a human, an actuator control mechanism or a machine learning software, receives only the data that is has subscribed to using the MQTT Topic.

An experimental Testbench setup was used for further tests performed with one accelerometer mounted on top of a vibrating device/shaker that generates vibrations according to the signal fed by a Waveform generator. The signal parameters, Frequency, Amplitude and Waveform shape, were varied and the resulting ThingSpeak output was analyzed for correspondence.

As future work, this thesis presents several research areas where it can be used as a basis. They include very important aspects such as information security in Internet of Things, machine learning, Big Data analytics for the sensor data.

## Bibliography

- [1] Šmíd, R.; Hanuš, O.: Condition monitoring using the Internet of Things (IoT) - In: CM 2016 and MFPT 2016. Northampton: British Institute of Non-Destructive Testing, 2016.
- [2] Randall, R. B.: Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications, Wiley 2011.
- [3] Ganssle, J. G.: The Art of Designing Embedded Systems, Newness 2008.
- [4] Bruel & Kjaer – Vibration Analysis – Envelope Analysis : <https://www.bksv.com/media/doc/BO0187.pdf>  
Written on November 2011, Accessed on 24<sup>th</sup> May 2017
- [5] Bruel & Kjaer – Diagnostics on Bearing Faults – Envelope Analysis – Vibration Analysis: <https://www.bksv.com/media/doc/bo0501.pdf>  
Written on February 2003, Accessed on 24<sup>th</sup> May 24, 2017
- [6] Bearing Envelope Analysis – Eric Bechhoefer:  
<http://www.mfpt.org/FaultData/MFPT%20Bearing%20Envelope%20Analysis.pdf>  
Written on March 2006, Accessed on 24<sup>th</sup> May 24, 2017
- [7] SKF Reliability Systems - Vibration Monitoring: <http://www.maintenance-engineering.eu/downloads/public/envelope%20bearing.pdf>  
Written on February 2003, Accessed on 24<sup>th</sup> May 24, 2017
- [8] Bruel & Kjaer – Measurement of Vibration : <https://www.bksv.com/media/doc/br0094.pdf>  
Written on September 1982, Accessed on May 24, 2017
- [9] Oil Analysis - [https://en.wikipedia.org/wiki/Oil\\_analysis](https://en.wikipedia.org/wiki/Oil_analysis)  
Written on 8<sup>th</sup> May 2017, Accessed on May 24, 2017
- [10] Analysis of Transformer Oil for Condition Monitoring – Dr B Pahlavanpour & Dr. A Wilson  
IEE Colloquium on An Engineering Review of Liquid Insulation : 7-7 Jan. 1997,  
**INSPEC Accession Number:** 5520876, Date Accessed on : May 24, 2017
- [11] Use of infrared thermography for computation of heating curves and preliminary failure detection in induction motors, 2-5 Sept. 2012, **INSPEC Accession Number:** 13119235,  
DOI: 10.1109/ICEIMach.2012.6349920, Date Accessed on : May 24, 2017
- [12] Tool wear forecast using Dominant Feature Identification of acoustic emissions,  
8-10 Sept. 2010, **INSPEC Accession Number:** 11624677, DOI: 10.1109/CCA.2010.5611259,  
Date Accessed on : 24<sup>th</sup> May 24, 2017
- [13] Tool health monitoring using airborne acoustic emission and a PSO-optimized neural network,  
24-26 June 2015, **INSPEC Accession Number:** 15347486, DOI: 10.1109/CYBConf.2015.7175945,  
Date Accessed on : 24<sup>th</sup> May 24, 2017

- [14] Use of Vibration Analysis Technique – Condition Based Maintenance: <http://www.diva-portal.org/smash/get/diva2:453447/FULLTEXT01.pdf>  
SYED TAFAZZUL MAHMOOD, KTH Royal Institute of Technology, Sweden, Accessed on : 24<sup>th</sup> May 24, 2017
- [15] Condition Monitoring : [https://en.wikipedia.org/wiki/Condition\\_monitoring](https://en.wikipedia.org/wiki/Condition_monitoring)  
Written on 23<sup>rd</sup> May 2017, Accessed on 24<sup>th</sup> May 24, 2017
- [16] Vibration Analysis : Six Benefits – David Manney : <http://www.plantengineering.com/single-article/vibration-analysis-six-benefits/7ffbe862785c1ea6c97abdaf3f274817.html>  
Written On : July 2016, Accessed on : May 24<sup>th</sup> 2017
- [17] Benefits of Vibration Analysis – Fluke : <http://en-us.fluke.com/community/fluke-news-plus/vibration/understanding-the-benefits-of-vibration-monitoring-and-analysis.html>  
Accessed on : 24<sup>th</sup> May 24, 2017
- [18] From Wikipedia, the free encyclopedia – Industry 4.0, Accessed on 21-May at 16:28 URL : [https://en.wikipedia.org/wiki/Industry\\_4.0](https://en.wikipedia.org/wiki/Industry_4.0)
- [19] Comparison with HTTP and MQTT on required network resources for IoT, IEEE <http://ieeexplore.ieee.org.ezproxy.techlib.cz/document/7814989/>
- [20] 810M1-0025X DataSheet : [http://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtv&DocNm=810M1\\_Accelerometer&DocType=Data+Sheet&DocLang=English](http://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtv&DocNm=810M1_Accelerometer&DocType=Data+Sheet&DocLang=English)  
Written on September 2015, Accessed on : May 24<sup>th</sup> 2017
- [21] Datasheet: <http://www.ti.com/product/MSP430F5529/datasheet>  
Written on: November 2015, Accessed on : 24<sup>th</sup> May 24, 2017
- [22] CC3100 SimpleLink™ Wi-Fi® Network Processor, Internet-of-Things Solution for MCU Applications
- [23] "MQTT 3.1.1 specification". OASIS. December 10, 2015. Retrieved May 21, 2017 at 18:16.
- [24] ISO/IEC 20922:2016 Preview. Information technology -- Message Queuing Telemetry Transport (MQTT) v3.1.1. URL: [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=69466](http://www.iso.org/iso/catalogue_detail.htm?csnumber=69466)
- [25] Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1, 2016. <https://www.iso.org/standard/69466.html>
- [26] Lucy Zhang : "Building Facebook Messenger". Retrieved May 23, 2017. URL: <https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920>
- [27] Dann Jonathan: "Under the hood: Rebuilding Facebook for iOS". Retrieved May 24, 2017 URL : <https://www.facebook.com/notes/facebook-engineering/under-the-hood-rebuilding-facebook-for-ios/10151036091753920>

- [28] 5 Things to Know About MQTT – The Protocol for Internet of Things. URL: [https://www.ibm.com/developerworks/community/blogs/5things/entry/5\\_things\\_to\\_know\\_about\\_mqtt\\_the\\_protocol\\_for\\_internet\\_of\\_things?lang=en](https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=en)
- [29] Eclipse Mosquitto – Open Source MQTT v3.1/3.1.1 Broker, Released March 9,2017 URL: <https://mosquitto.org/>
- [30] Comparison with HTTP and MQTT on required network resources for IoT, IEEE URL : <http://ieeexplore.ieee.org.ezproxy.techlib.cz/document/7814989/>
- [31] A Web-Based IoT Solution for Monitoring Data Using MQTT Protocol, 2016 International Conference on Smart Systems and Technologies (SST), 12-14 Oct. 2016 URL : <http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=7765668>
- [32] ThingSpeak , article on MathWorks- Accessed on May 24, 2017 at 8:04PM URL : <https://www.mathworks.com/help/thingspeak/index.html>
- [33] Desktop MATLAB support for ThingSpeak, URL : <https://in.mathworks.com/hardware-support/thingspeak.html>
- [34] MSP-EXP430F5529LP Quick start Guide, LaunchPad : URL : <http://www.ti.com/lit/ml/slau536/slau536.pdf>
- [35] BoosterPack Checker, LaunchPad and BoosterPacks compatibility, URL : <https://dev.ti.com/bpchecker/#/>
- [36] From Wikipedia, the free encyclopedia – Code Composer Studio, Accessed on May 25, 2017 at 2:13 AM URL : [https://en.wikipedia.org/wiki/Code\\_Composer\\_Studio](https://en.wikipedia.org/wiki/Code_Composer_Studio)
- [37] Code Composer Studio Integrated Development Environment (IDE), Texas Instruments URL : <http://www.ti.com/tool/ccstudio>
- [38] Bruel & Kjaer : Permanent Magnet Shaker- LDS V406, URL : <https://www.bksv.com/en/products/shakers-and-exciters/LDS-shaker-systems/permanent-magnet-shakers/V406>
- [36] Embedded MQTT C/C++ Client Libraries. URL: <https://eclipse.org/paho/clients/c/embedded/>
- [37] MQTT ON THE TI CC3200 LAUNCHPAD THANKS TO PAHO EMBEDDED CLIENT  
Written on: AUGUST 26, 2014. Retrieved May 26, 2017. URL: <https://blog.benjamin-cabe.com/2014/08/26/mqtt-on-the-ti-cc3200-launchpad-thanks-to-paho-embedded-client>
- [38] paho-mqtt 1.2.3. MQTT version 3.1.1 client class. Accessed on: May 26, 2017. URL: <https://pypi.python.org/pypi/paho-mqtt>