|  |  |  |
| --- | --- | --- |
|  | MANIPAL INSTITUTE OF TECHNOLOGY  (A constituent Institute of MANIPAL UNIVERSITY)  MANIPAL - 576 104, KARNATAKA, INDIA |  |

Industrial Training

on

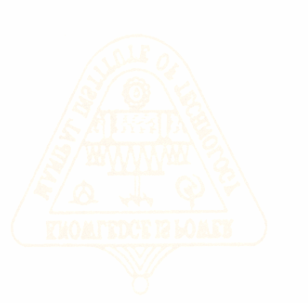
Cloud Based Predictive Model for Machine Health Monitoring on Factory Floor

SUBMITTED

BY

Rajat Kumar Singh 140905538 krajat1995@gmail.com

7406580777



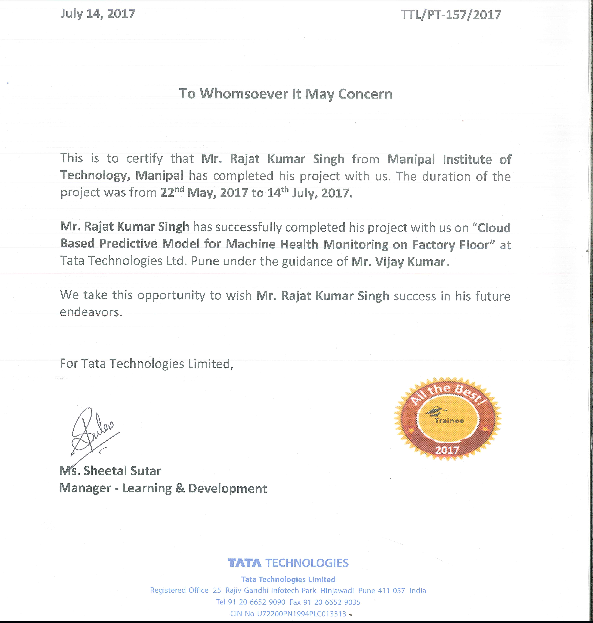
Under the Guidance of:

Mr. Vijay Kumar

Program Manager

Connected Enterprise IT(CEIT)

Tata Technologies



**CONTENTS**

1. ***Abstract***
2. ***About Tata Technologies***
3. ***Introduction***

* *3.1 General*
* *3.2 Time Series*
* *3.3 Internet of Things(IoT)*
* *3.4 Cloud Communication Models*

1. ***Simulation Model***
2. ***Data Acquisition(DAQ) hardware***
3. ***MQTT Protocol***

1. ***Pattern Recognition***

* *7.1 Pre-Processing*
* *7.2 Modelling the Hidden Markov Model*
* *7.3 Training the Hidden Markov Model*
* *7.4 Probability Computation for HMM Model*
* *7.5 Hidden State Prediction*

1. ***Results***
2. ***Conclusion and Future work***
3. ***References***
4. ABSTRACT

Internet of Things (IoT) is an ecosystem of connected physical objects that are accessible through the internet. Advancement in the field of Computer Networks and Artificial Intelligence has garnered a lot of attention and interests towards incorporating them in IoT systems. The basic premise is to have smart sensors collaborate directly without or with minimal human involvement. Some smart applications built around home automation, health monitoring and smart sensing have already been implemented and shown feasible at a small scale. In the coming years, IoT is expected to bridge diverse technologies to enable new applications by connecting physical objects together which in turn would support intelligent decision making.

The aim of this project is to implement an Internet of Things(IoT) system comprising of actuators, effectors, network connectivity and software for detecting patterns, monitor readings and generating alerts. The system is primarily divided into the data acquisition system and the data analytics system. Data acquisition is the process of measuring and analyzing various physical and electrical entities like voltage, current, temperature, pressure etc. Our data acquisition system consists of sensors, interfacing circuitry, an Analog to Digital Converter and application software. Such a system forms the backbone of our much larger IoT system as it is responsible for the sole interface to our remote location. Our project aims at acquiring data from an accelerometer and a temperature sensor deployed in an industrial environment. This data is sent to a cloud storage via the Ethernet, using the lightweight, Machine to Machine MQTT protocol, for analysis. A Raspberry Pi 2 is used for data collection and it is interfaced to the sensors using an Arduino Uno microcontroller.

Once sent to the cloud, the data is analyzed to check for any abnormalities in the functioning of the processes in the industrial environment. The data analytics system is responsible for analyzing patterns in the data at regular time intervals and alert the user when abnormalities in the readings are observed. This system allows us to monitor the health of the machines in an industry without any human intervention. To achieve this goal, machine learning was employed, specifically the concepts of Hidden Markov Models. The learning model is supervised in nature, taking into consideration that defects on the factory floor will have different patterns, each of which is detectable and distinguishable by out sensors.

1. About Tata Technologies

Tata Technologies is a global company in the Tata Group that provides services in engineering and design, product lifecycle management, manufacturing, product development, and IT service management to automotive and aerospace original equipment manufacturers and their suppliers.

The company is a strategic partner for developing complete vehicles, engineering subsystems and components, managing the new product introduction (NPI) process through collaborative engineering tools, such as Product Lifecycle Management (PLM) and tying together information created and used throughout the extended manufacturing enterprise.

The company was founded in 1989 and acquired INCAT, a European-based company, in 2005. Tata Technologies is headquartered in Singapore, with regional headquarters offices in the United States (Novi, Michigan), India (Pune) and the UK (Coventry) with a combined global work force of more than 8,500 employees serving clients worldwide from facilities in North America, Europe and the Asia-Pacific region.

***Purpose***  
To make product development dreams become reality.

***Mission***  
To transform product development through deep industry knowledge combined with intelligently different approaches to technology, process, innovation and execution. To help the world to drive, fly, build and farm by enabling our clients to realize better products through our offerings across our focus verticals (automotive, aerospace and Industrial machinery).

***Vision***  
To engineer a better world by helping our clients realize better products and improve the quality of lives of those that are exposed to those products.

The services offered by Tata Technologies include Engineering, Research and Development(ER&D), Product Lifecycle Management(PLM) and Connected Enterprise IT(CEIT). Some of the products are IProducts, PTC Software and Dassault Systemes. Tata Technologies also provides consultancy services across a wide range of industries.

The Tata name has been synonymous with corporate social responsibility (CSR) for more than a century. As a Tata company, Tata Technologies proudly champions sustainable, socially responsible business practices and undertakes many philanthropic and community development initiatives throughout the year. Moreover, sustainability is managed as a core pillar of the company’s strategy, with oversight by a dedicated subcommittee of the Board of Directors. Every year, this subcommittee conducts a materiality assessment of the environmental, social and governance factors that surround the company with a view to identifying issues that represent high concern to external stakeholders and have the potential to significantly impact the company’s performance. The output of the materiality assessment is a set of priorities that the company has embraced in the areas of education, diversity, green workplaces, and ensuring the products that we help our clients engineer positively impact safety, health and the environment.

1. INTRODUCTION

**3.1 GENERAL**

Automated tools have made analysis and recognition much easier than they were earlier. With the rise of big data, manual analysis of raw data has become impossible. The need of real time data analytics requires complex tools which have a low latency and high accuracy for computation of results. Internet of Things(IoT) requires a real time analytics system which accumulate data from remote locations to a central system, which is usually a database. Analytics is performed at this location and the output is delivered to the user which could be in the form of remote monitoring, notifications or weekly updates. Every system has different requirements and utilities. Each system requires a different model, but all of them have one thing in common – deployment of software incorporating advanced algorithms and optimizations to automate the analytics.

Machine Learning is a set of application interfaces and tools of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without external explicitly programming. The main focus of Machine learning is on the development of programs that can access, analyze and deduce observations from data and use it learn for themselves(Unsupervised) or with the help of external models(Supervised). Most of them can be easily solved by algorithms such as Support Vector Machines, Regression Models, Clustering Models and Bayesian Classifiers. Many tools employ these algorithms to solve real world problems. This paper deals with data which is present as a continuous distribution in real time. A possible way to treat sequential data would be simply to ignore the past observations and sequential history and treat every observation as an independent occurrence. Such an approach, in any case, would neglect to misuse the successive patterns in the information. Suppose, for instance, that we observe a variable denoting the color of the traffic lights. Assuming we are given a time series of recent observations of this variable, we wish to predict the next color of the traffic light. If we treat the data as independent occurrences, then the only information we can obtain is guessing the probability of a particular color based on the prior probabilities. However, we know in practice that colors of the traffic lights follow a fixed pattern. If the current color is green, then the next state would be yellow followed by red. Observing the past history is therefore necessary in this case. This sequential model can be modelled by Hidden Markov Models which take into consideration the history of data, just preceding the next observation. The problem is modelled as a stochastic process which contains N output states and M hidden states as per the requirements of the model. A set of observations cause the model to change states and finally converge at a final state. Many applications such as speech recognition, motion genre classification and pattern recognition employ HMM to model the continuous input data.

* 1. **Time Series**

Much of the data which we store are in the form of time series. Time series data forms a crucial part in the everyday life. Time series data conceal important information in the form of patterns or aberrations which can be detected and used in applications. In the past, traders usually took their investment decisions after looking at the prices graphs, critically analyzing the data and recognizing some trading patterns. We can replicate the behavior of a discretionary trader, by training a model which specifically detects the patterns in the financial model. Hidden Markov Models are statistical models widely applied in speech recognition and they can easily handle these patterns' characteristics.

Since our sensors are stored in a predefined location which records readings and transmits the readings via an IoT Gateway, we can remotely access the time series data and run the analytics algorithm on a powerful processor. By prediction patterns we can let the user know of the patterns encountered till date. A further enhancement which employs event detection to check for ‘critical’ conditions is deployed. Thus the user can receive an alert which tells him of a abnormalities in the machine functioning, thus helping to mitigate damages. Event Detection is particularly useful as it can be used to predict patterns beforehand, thus providing time to take corrective measures.

* 1. **Internet of Things(IoT)**

The term “Internet of Things” (IoT) was first used in 1999 by British technology pioneer Kevin Ashton to describe a system in which objects in the physical world could be connected to the Internet by sensors. The term Internet of Things generally refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention. More specifically, the Internet of Things aims at offering new applications and services bridging the physical and virtual worlds in which Machine-to- Machine (M2M) communications represent the baseline communications that enables the interactions between Things and applications in the cloud.

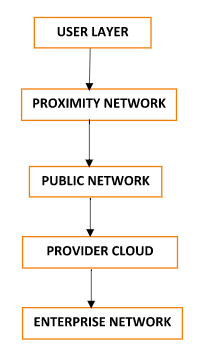
Internet of Things has a wide range of applications such as manufacturing, medical as well as security.

* Smart homes which provide autonomous management of domestic premises (heating control, consumer appliance management and monitoring, and security) thus making human interaction minimal and redundant.
* Medical applications such as remote monitoring and treatment of patients are ground- breaking innovations made possible by the advent and progress of IoT.
* Connected vehicles, which are becoming closer to reality require communications between the different vehicles for successful navigation.
* Industrial and manufacturing applications use smart production lines for monitoring quality and quantity. Furthermore, machine monitoring can be done through the use of IoT by supplying suitable sensors for anomaly detection.
* Various consumer applications are also made possible by the growing smartphones and wearables .
* Recently for the Champion’s Trophy, it was announced that the players will use electronic chips in their bats to monitor their technique. This is also an application of IoT.

IoT is a transformational technology which has led to the building of ‘smart’ technologies capable of active and proactive actions without human interaction.

The building blocks which make IoT possible are : -

* Application and User Interaction:- The needs of the user and business are addressed and hence new technologies are proposed for improved functionality. There is a collaboration involving people , business and applications.
* Cloud Server:- The ability of the cloud servers to store large amounts of real time data, support real time processing, analytics , content delivery and application hosting have made IoT possible.
* Network:- Internet access , Bluetooth , WiFi, cellular 2G/3G/4G
* Gateway:- Addressing of devices has been made possible by communication standards and protocols. Thus devices can now use this gateway to communicate with other devices or transmit data.
* Physical Objects and Devices:- These comprise of the objects which are to be made ‘smart’. This is to be made possible through the use of Sensors and Actuators. Sensors are used to encode physical information in an environment in the form of a signal for computational processing. Actuators act on the signal to produce the desired output.



***Fig 1: Domains of an IoT System***

*1. User Domain*:- This layer is independent of any specific network domain. The IoT user and the end user application are a part of this layer.

*2. Proximity Network*:- This layer comprises the Sensor, Actuator and Firmware. It basically comprises of the physical entities which are fundamental in IoT. This domain has networking capabilities. The device communicate through an IoT gateway . The IoT Gateway acts as a means to connect one or more devices to the public network. It also has other capabilities like the ability to filter and intelligently react to data.

*3. Public Network*:- It comprises of the various data sources that feed the architecture. Data sources include the traditional systems of record from the enterprise. It primarily contains the peer cloud systems and the edge services. A peer cloud is basically a 3 rd party cloud system which is used for providing functionality to the IoT service. The edge services provide a safe passage for data to flow trom internet into the provide cloud. It consists of the Domain Name System Server, Content Delivery Networks , Firewall and Load Balancers.

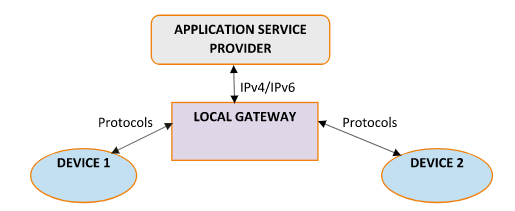
*4. Provider Cloud*:- Perhaps the backbone of the IoT framework, this layer hosts the Device Data Store, API Management, Application Logic, Device Management and Analytics. Application Logic pertains to the logic which handles the chunk of data being received from the devices and directing them to other services thus establishing a control flow. Analytics refers to the application which finds out interesting or important patterns in the data. This part uses various algorithms to discover such patterns. Various machine learning algorithms are deployed. It can use transform, filter and analyze data in real time along with storing it in repositories. The control logic for triggering actuators which could be in the form of an alert, message , siren, etc.

*5. Enterprise Network*:- This layer hosts the enterprise data, enterprise application and enterprise user directory. The data contains metadata of the various applications which are in use. The directory stores user information to support authorization and authentication. The results of the system are transformed in a suitable form and output to the user via secure messaging to and from the enterprise systems.

* 1. **Cloud Communication Models**

The various types of communications involved in an IoT system are:-

*1. Device to Cloud* :- The device communicates directly with the cloud to exchange data and other control messages. This communication allows for the data to reside on the cloud service and various operations and analysis can be performed on the data. This form of communication requires an application software on the local gateway to provide satisfaction of security and other protocol requirements.



***Fig 2: Device to Cloud Model***

The local gateway can be any device capable of interacting with the cloud. In many cases a

smartphone is the local gateway. The smartphone interacts with the device to get the data and

then acts as a local gateway for transmitting the data to the cloud.

*2. Back-End Data-Sharing Model*:- The data from the devices once available from the cloud can be utilized with other data and applications in the cloud itself. This allows for the smart device’s data to be aggregated and analyzed. The back end sharing model relies on a API framework for collection of data from cloud and issuing response to the desired actuators.

*3. Device to Device Model*:- This type of communication involves direct connection and communication between devices and remove the use of an intermediate application server. This model relies on Bluetooth and other forms of communication to transmit data packets. The task performed by these models are usually lightweight and require transmission of small packets. One such example can be a smart home. Whet the door in unlocked from the outside the lights can be switched on automatically. Such setups also cause various challenges because of adherence to a set of protocols.



***Fig 3: Device to device Communication***

The model which we are using is the more common device to cloud model. The model already supports state of the art protocols and eliminates security concerns associated with the other models.

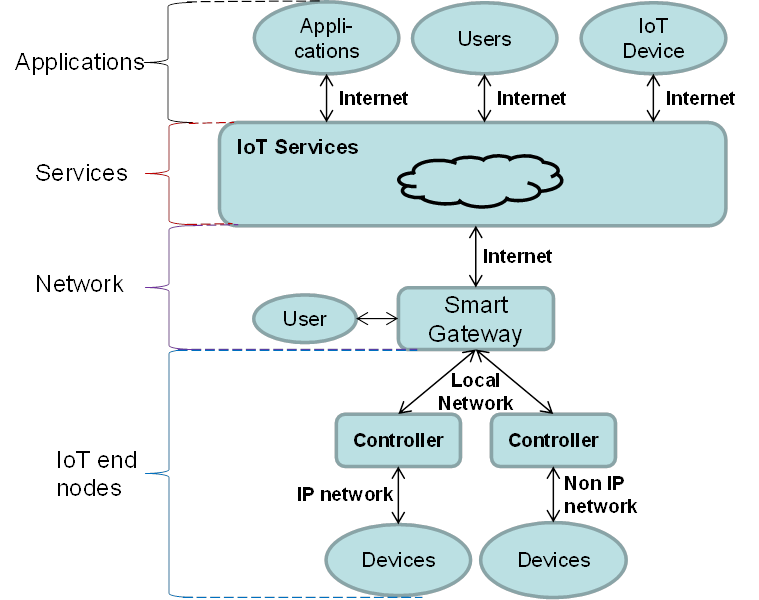
1. SIMULATION MODEL

* The data will be read by the sensors at regular intervals of time.
* The sampling frequency of for the sensors will be set by the programmer.
* The data sampled by the sensors will be in the analog form (in terms of voltages or currents) and will be converted to digital using an Analog to Digital converter like ADC 8088.
* In order to model the above data that will be generated, a code can be written to fit the nature of the data approximately by a mathematical function, and this function can be sampled in order to generate discrete values of data.
* In order to achieve a more practicable system noise will be incorporated in the simulation.
* The noise may be due to shutdown, overheating or any other plausible error.
* Further the noise may be added or multiplied to the system to increase its applicability.
* This data will be processed by the CPU and sent over to the cloud to store the data and send notifications to the user device.
* The data will be sent to the cloud using the MQTT (Message Queue Telemetry Transport) protocol.
* The data will be analyzed in the cloud and accordingly the device will be notified.
* In order to simulate the data that will be sent by the sensors we fit a continuous well behaved mathematical function to give the approximate nature of how that parameter behaves.
* The daily temperature can be modelled by a Gaussian curve which has an offset equal to the mean temperature of that region over the day, and the mean and variance will be decided by the maximum and minimum values attained by the temperature over the day.
* For the temperature we can use the Gaussian curve with offset 20, mean 15 and variance 2.1 to fit the temperature range of Pune, as of May 2017.
* All the other sensor parameters can be modelled by mathematical functions in a similar way.
* Noise: In order to incorporate noise in the system, we consider the following reasons for noise to be generated in the system.

1. One of the sensors is faulty and fails to give a reading, thus creating noise in the system.
2. If there is a sudden shutdown of power in the plant, all the readings will be zero.
3. There may also be a continuous noise in the system, it can be additive or multiplicative. This can be due to manufacturing defects, heat generation etc.

* The data will now be sent over to the cloud using the MQTT protocol.
* In the cloud, the data will be processed and analyzed to give appropriate feedback to the user.

Sensors provide input about its current state (internal state + environment). Actuators are used to make a change in the environment in correspondence to the sensor readings.The functioning of our particular system is spread over 3 layers- Perception Layer, Network Layer and Application Layer.



***Fig 4: IoT Architecture of Model***

1. Data Acquisition(DAQ) Hardware

Transducers sense physical phenomena and provide electrical signals that the DAQ system can measure. The electrical signals produced are proportional to the physical parameters they are monitoring. The analog input specifications can give you information on both the capabilities and the accuracy of the DAQ product. Basic specifications, which are available on most DAQ products, tell you the number of channels, sampling rate, resolution, and input range. The number of analog channel inputs will be specified for both single-ended and differential inputs on boards that have both types of inputs. Single-ended inputs are all referenced to a common ground point. These inputs are typically used when the input signals are high level (greater than 1 V), the leads from the signal source to the analog input hardware are short (less than 15 ft.), and all input signals share a common ground reference. If the signals do not meet these criteria, you should use differential inputs. With differential inputs, each input has its own ground reference. Noise errors are reduced because the common-mode noise picked up by the leads is canceled out.

*Sampling Rate*: This parameter determines how often conversions can take place. A faster sampling rate acquires more points in a given time and can therefore often form a better representation of the original signal. For example, audio signals converted to electrical signals by a microphone commonly have frequency components up to 20 kHz. To properly digitize this signal for analysis, the Nyquist sampling theorem tells us that we must sample at more than twice the rate of the maximum frequency component we want to detect. So, a board with a sampling rate greater than 40 kS/s is needed to properly acquire this signal.

*Multiplexing*: A common technique for measuring several signals with a single ADC is multiplexing. The ADC samples one channel, switches to the next channel, samples it, switches to the next channel, and so on. Because the same ADC is sampling many channels instead of one, the effective rate of each individual channel is inversely proportional to the number of channels sampled.

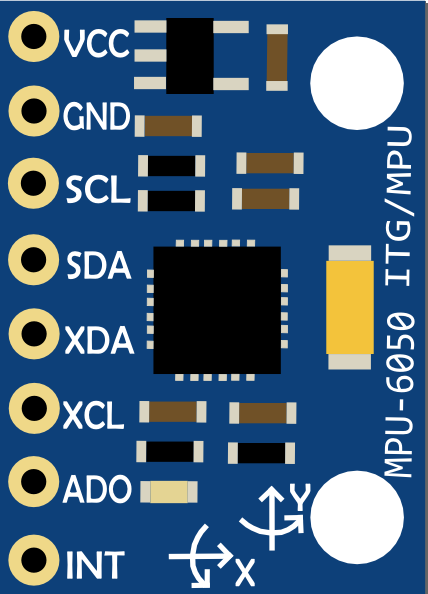
*Range*: Range refers to the minimum and maximum voltage levels that the ADC can quantize. The multifunction DAQ boards offer selectable ranges so that the board is configurable to handle a variety of different voltage levels. With this flexibility, you can match the signal range to that of the ADC to take best advantage of the resolution available to accurately measure the signal.

*Noise*: Any unwanted signal that appears in the digitized signal of the DAQ board is noise. Because the PC is a noisy digital environment, acquiring data on a plug-in board takes a very careful layout on multilayer DAQ boards by skilled analog designers

We used the two sensors **DHT11** and **MPU 6050** as temperature sensor and accelerometer respectively. The DHT 11 sensor uses a thermistor to convert temperature changes to changes in voltage and has three pins. The pins are for VCC, ground and analog data.



***Fig 5: DHT11 Pin Layout***



***Fig 6: MPU6050 Pin Layout***

MPU6050 has an accelerometer and a gyroscope measuring six degrees of freedom which are converted to voltages using the DMP, Digital Motion Processor, built in the chip itself. It has two pins to transmit the data, they are SDA and SCL. The SCL pin maintains a clock frequency and the SDA pin carries the data so that this sensor can be interfaced with an asynchronous line of communication. The MPU 6050 uses the I2C (Inter Integrated Circuit) for transmission of data.

We interfaced these sensors to an Arduino board which was programmed to read the data and give a serial string of data as output for further operation. In order to read the data sent by the accelerometer in the I2C format we used the MPU6050 library by Jeff Rowberg to convert the data into readable values which help in visualizing the movement of the machine and gives the output as the three angles yaw, pitch and roll.

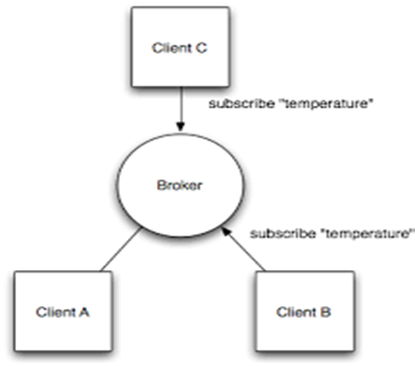
The final step of this Data Acquisition process is for the raspberry pi, which acts as the IoT gateway to read the data from the Arduino via a serial USB port and transmit it over to the cloud for storing in the database and further analysis.

The USB serial port of the Raspberry Pi 2 is used to read the data sent from the Arduino. A baudrate of 115200 is used to receive data via the serial port. The sampling frequency at which the data needs to be read is set on the pi. The PySerial library for Python allows us to read the data on the serial port and process it according to our needs. The data is read from the serial port and is packed in the JSON format to be sent to the MQTT broker in real time.

A Python code slows the pi to read the data from the USB port and transmit it over to the cloud using the MQTT protocol. In order to implement the publish-subscribe method of the MQTT protocol we created our own MQTT broker locally which allows data to transmitted to the cloud. Further the data is sent in the JSON format which is a lightweight medium and allows data to be sent in the form of arrays or lists thus making it highly readable.

1. MQTT Protocol

After the sensors have been interfaced, we need to transmit the data over a network. The messaging protocol used in our project is MQTT. MQTT (MQ Telemetry Transport or Message Queue Telemetry Transport) is an ISO standard publish–subscribe based lightweight messaging protocol used on top of the TCP/IP protocol. The main advantage of using this protocol is the minimal network bandwidth requirements. In contrast to the client-server paradigm where the server runs continually to service the client/s, the publish-subscribe approach uses a message broker for distributing messages to interested clients. Instead of initiating connections to specific nodes, information is published and interested parties may subscribe to it. The publish-subscribe paradigm also aims to achieve a scalable multicast, which greatly increases network’s efficiency for content delivery when the same content is requested by multiple subscribers. The network also makes sure that subscribers only receive the information they are interested in, effectively preventing most of SPAM and DoS attacks.



***Fig 7: Illustration of use of MQTT***

Mosquitto is an open source message broker that implements the MQTT protocol and which is used as a broker for our project. The MQTT connection is always between the client and the broker. Clients cannot directly interact with each other. The connection with the broker is handled by the CONNECT/CONNACK messages. Even if the subscribed client has lost its connection with the broker, it will receive the remaining messages once it reestablishes connection with the broker. Scalability and fault tolerance are key aspects of the MQTT protocol.

In figure 7, the clients ‘A’ and ‘B’ publish their messages on a particular topic name, which in this case is ‘temperature’. Client ‘C’ subscribes to the topic ‘temperature’ and it receives the messages from both the clients. The messages transmitted to the MQTT broker can either be simple strings or floating point values or even complex JSON( JavaScript Object Notation ) objects. The Mosquitto broker is configured to handle communications from only a predefined set of users. The users can publish or subscribe to the particular message broker by using the predefined APIs along with the broker location details(IP address + port address).

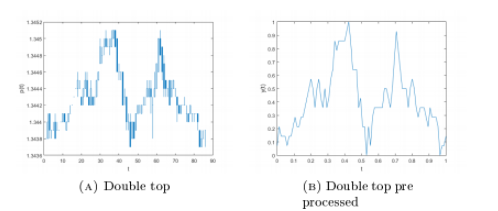
1. Pattern Recognition

**7.1 Pre-Processing**

Datasets can be scraped off the World Wide Web ore sensor readings could be compiled and stored for specific implementations. For our implementation, we compiled readings from the sensor throughout many days and used it for supervised learning .HMM implementations are available in RHmm for R and hmmlearn for python. The next step is to normalize the readings in accordance to each reading set. To normalize the data we use the following formula-

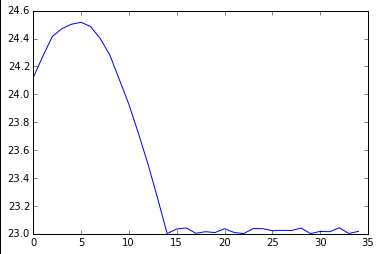


The denominator acts as scaling factor whereas the subtraction of *mins Cs* acts as a linear translator to limit the dataset values in the range [0,1]. The results can be seen in Figure 8.

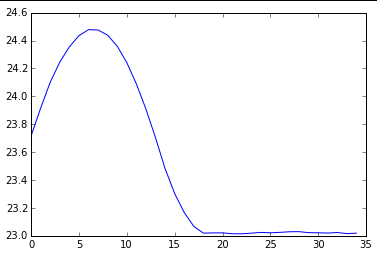


***Fig 8: Effects of pre-processing on time series***

Filtering or Smoothing can be performed on the given data series. This is optional and totally depends on the needs of the model and the accuracy and sensititivity desired. If changes performed on the data are significant then this can lead to loss of valuable data. As long as smoothing is performed to get rid of noise of minute amplitudes, moving average can be performed. Proper care should be taken to define the window size for implementing the moving average. Figure 9 contains the input and result to the moving average algorithm.



*INPUT*



*OUTPUT*

*Fig 9: Effects of moving average smoothing on financial time series data*

**7.2 Modelling the Hidden Markov Model**

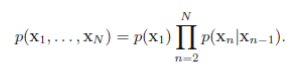
The most important task was to convert the set of continuous values of a signal to a discrete form so that it could be used as a feature set. To do this, we extracted the temporal features of the signal. Temporal features would accurately predict the type of noise introduced in a system without depending on the signal level on which the noise is imposed. After transforming out continuous data to a discrete form, out next aim was to map it to a Hidden Markov Model.

A Markov Model allows for modelling of a state space design in which each space is related to one or more states. A first order Markov Chain, as shown in figure 3, is one in which the next observation can be directly predicted by the observation which preceded the current observations.



***Fig 10: First Order Markov Chain***

The joint distribution of this model is given by



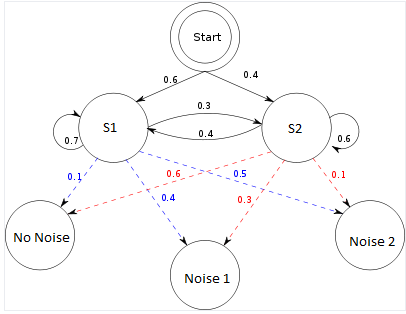
This model is too basic form most application requirements as each state is deterministically associated with another state. A slightly generalized version of this model is the second order Markov chain as shown in Figure 11.



***Fig 11: Second Order Markov Chain***

This model is dependent on 2 models and can make a transition to either one of the models. A further modification and enhancement of this model is the Hidden Markov Model. This model is a probabilistic model in which states have a transition probability associated with it and another state. Hence the deterministic nature of the first order Markov chain is eliminated. Each state is known as the observed state as it directly visible. Hidden layer is the output layer corresponding to the output which we want to predict. Each observed state has an emission probability associated with the output layers involved. The transition matrix, emission probabilities and the initial probabilities needs to be specified for predicting the output of a particular state sequence.

Consider for example the Hidden Markov Model depicted in Figure 12.



***Fig 12: Sample HMM consisting of a 2 states modelled to predict noises in a signal***

The model consists is defined by the following parameters :-

*Hidden States = ( 'S1' , 'S2' )*

*Observations = ( 'No Noise' , 'Noise 1' , 'Noise 2')*

*Start\_Probability = { 'S1': 0.6, 'S2': 0.4 }*

*Transition\_Probability = {*

*'S1' : { 'S1': 0.7 , 'S2': 0.3 },*

*'S2' : { 'S1': 0.4 , 'S2': 0.6},*

*}*

*Emission\_probability = {*

*'S1' : { 'No Noise': 0.1 , 'Noise 1': 0.4 , 'Noise 2': 0.5},*

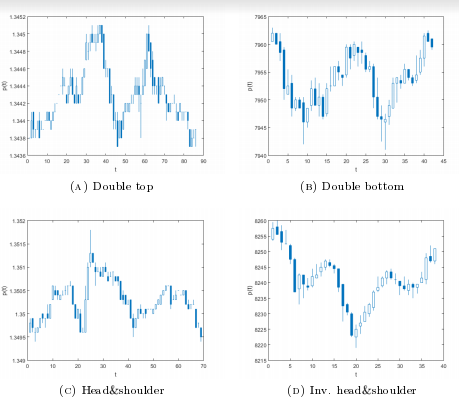
*'S2' : { 'No Noise': 0.6 , 'No Noise': 0.3 , 'Noise 2': 0.1},*

*}*

The observations are directly available to us whereas a random number of hidden states can be created to represent a particular configuration. For our implementation the ‘No Noise’, ‘Noise 1’ and ‘Noise 2’ states would be replaced by the appropriate patterns found in the time series for each of the sensors. Each financial time series pattern can have multiple hidden states.

**7.3 Training the Hidden Markov Model**

Our implementation deals with construction of a software to recognize four patterns: double tops (dtop), double bottoms (dbot), head&shoulders (has) and inverted head&shoulders (invhas) and we will analyze the performance of our recognition algorithms. Figure 14 depicts the appearance of these patterns when the data is plotted against time.



***Fig 13:Patterns for which model is trained***

When you do not have foreknowledge of the HMM parameters:- the state transition matrix, the observation matrix, and the emission probabilities , Baum-Welch Algorithm can be applied to resolve the issue. Baum-Welch EM algorithm uses heuristic functions to arrive at partial solutions corresponding to locally optimum parameters. The steps involved in guessing the hidden state transition includes:-

* Postulate that transition at every single spot in every single observed sequence(separately)
* Check how the probabilities compare to the best probabilities for those observed sequences.
* Use this ratio to update the transition probability.

**7.4 Probability computation for HMM model**

In our implementation, we assume M HMMs for M reference patterns. During our classification of a particular pattern as a seen or unseen pattern. We would like to compute the probabilities of a set of sequences to belong to a particular pattern. Each HMM should return the probability of the sequence to belong to the HMM pattern. The one with the maximum probability is chosen as the pattern. For this implementation, we have to create an HMM to define a pattern which does not resemble any of the 4 reference patterns. This non-reference HMM has to be built to avoid false positives.

Forward Algorithm is used to find the probability of a sequence of observations. A brute force approach would compute all the probabilities and sum them up for a given set of observed states over all possible combinations of hidden states. For a model containing 3 states, a total of 3\*3\*3(=27) probabilities would be computed for each of the 27 combinations. This algorithm can be optimized from O(N!) to an O(N2T) algorithm using basic dynamic programming algorithms. We could compute the probability to reach a particular state from a state ito a state j. To go to the next state, the previous probabilities can be considered and combined with the probability of transition from state i to state j. The final step would be to compute the summation to yield the absolute probability of the observance sequence.

**7.5 Hidden State Prediction**

The algorithms already discussed can train an HMM and predict the next observable state. The Viterbi algorithm is useful to predict the sequence of hidden states which are being followed by a given observation sequence provided we have a trained HMM model.

A straightforward and inefficient approach to finding the hidden state sequence would be to compute the probabilities for all initial states. Viterbi algorithm optimizes the solution by using partial outputs at each state. Consider states A, B and C in which a model is currently. Consider the partial probabilities to reach the states A, B and C are already computed. To reach a state X , we have the following probabilities-

(Sequence of prior states Si),……..,A,X

(Sequence of prior states Sj),……..,B,X

(Sequence of prior states Sk),……..,C,X

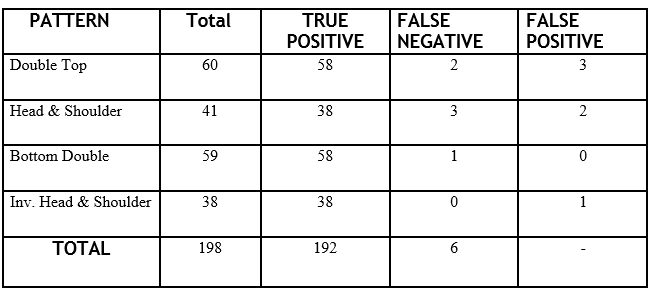
We will choose the path which would have the maximum probability at the state X. Thus the max probability becomes the partial output at state X, which can be used further. Thus, the formula for computing the partial output at stage X is

*Pr(X at time t) = max i=A,B,C ( Pr(i at time t-1) \* Pr(observation at time t | X) \* Pr(X | i))*

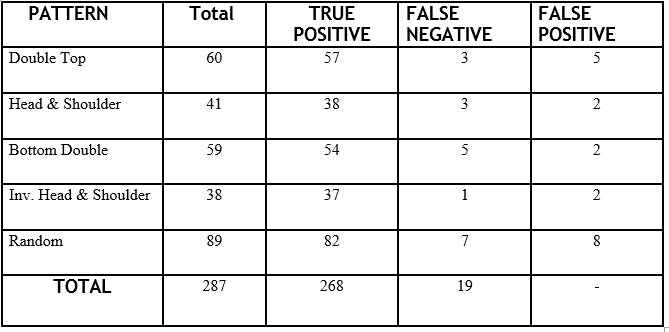
1. RESULTS

The HMM models yield an accuracy of 97% the dataset is limited to reference patterns ,i.e., one of the 4 patterns used in this paper.

*Table 1: Recognition results on sample data without inclusion of non-reference states*



When 89 random patterns were added to simulate real time data, the accuracy deteriorated to 93%. These non-reference observation sequences lead to some false positives which lead to a performance deterioration.

*Table 2: Recognition results on sample data with inclusion of non-reference states* 

Analysis of the false positives could lead to the making of filters which would help get rid of the false positives in the case of non-reference algorithms. Further pre-processing can be applied to filter and normalize the data to yield better results. Proper care should be taken to reduce loss of information.

Event Detection using an HMM based model led to promising results when it was superimposed on carrier waved which would distort the signal to a certain extent thereby simulating noisy environments.

1. CONCLUSION AND FUTURE WORK

This paper adequately classifies the financial time series data into the following 4 patterns with an accuracy of 93%. The Hidden Markov Model approach used in this implementation performs very well on the 4 categories. Future work to be done to build on this implementation would include incorporating more number of sensor series patterns. Incorporating more models would lead to deterioration of accuracy to some extent. Sensor readings are very erratic in nature if the sensitivity is high as in our case, and the readings contain large amounts of noise, which makes it difficult for us to analyze and predict it.

A module needs to be built on this HMM approach which works in real time to predict the next sets of observations by taking past observation sequence into consideration. Since our model takes past observations into consideration only for detecting the current pattern, we fail to utilize the probabilistic nature of certain pattern sequences. The forecast sequence can be fed to out model to classify s one of the pattern. Using a forecast from the newly trained HMM , we can fit the probabilistic observation sequence to our models to find the model it most likely represents.

There are many challenges which need to be addressed if a real time model is considered for implementation. A certain amount of ambiguity will always be involved as patterns could have similarities in its shape. Let us suppose 2 signals as shown in Figure 14.

Signal A Signal B

***Fig 14: Signals which would cause problems in real time analysis***

Both the signals resemble each other. The increasing slope in the beginning would lead to ambiguities. Since both the signals are equally probable if we compare the initial patterns, we cannot predict which model it is even if we know it resembles a particular pattern. The solution in this case would be to wait till the signal shows a variation in the pattern . When this happens, it can then accurately predict the pattern type.

10. References

[1] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, 2009.

[2] “Cloud Customer Architecture for IoT” – 2106 Cloud Standards Customer Council

[3] Sara Rebagliati, Emanuela Sasso “Pattern recognition using hidden Markov models in financial time series”, ACTA ET COMMENTATIONES UNIVERSITATIS TARTUENSIS DE MATHEMATICA Volume 21, Number 1, June 2017

[4] ‘Internet of Things- Converging Technologies for Smart Environments and Integrated Ecosystems’- Ovidiu Vermesan,

[5] L. R. Rabiner, B. H. Juang “An Introduction to Hidden Markov Models”, IEEE ASSP Magazine,January 1986

[6] S. Kwong, C. W. Chan, K. F. Man, and K. S. Tang, “Optimisation of hmm topology and its model parameters by genetic algorithm, Pattern Recognition”