Application of Internet of Things for Equipment Maintenance in Manufacturing System

Tejaswini S Sharadhi[1], R S Ananda Murthy[2], Dr M S Shashikala[3]
[1]*MTech, Energy Systems and Management, Department of Electrical and Electronics Engineering, JSS Science and Technology University, Mysuru, Karnataka, India*
[2]*Associate Professor, Department of Electrical and Electronics Engineering, Sri Jayachamarajendra College of Engineering, Mysuru, Karnataka, India*
[3]*Professor, Department of Electrical and Electronics Engineering, Sri Jayachamarajendra College of Engineering Mysuru, Karnataka, India*

Abstract- **Internet of Things (IoT) has provided an encouraging scope to build potent industrial systems and applications by making use of wireless, mobile, and sensor devices. Time to come is of Internet of Things, which will remodel the real world things into rational virtual things. Maintenance is one of the major aspects of industries and power plants. Manual maintenance consumes more energy, time, labour etc. This paper proposes an IoT model for maintenance system in industries and power plants. The paper proposes a system which helps in automating the maintenance system thereby reducing the human intervention to a greater extent. The system saves a large amount of time, money, energy, labour, increases safety level, quality of production and improves the efficiency of manufacturing industry.**

Index Terms- Raspberry Pi, MQTT Protocol, MQTT Dash, Sensor Kit.

I. INTRODUCTION

Industrial Revolution of 18th Century is a boon to the mankind. Machines are replacing men in each and every aspect. Usage of machines helps in reduction of time consumption, accuracy and similarity in manufactured products, reduced human intervention etc. But, improper maintenance of machines results in breakdowns, failures and increase in the wastage of manufactured products. Hence maintenance is one of the important aspects of any manufacturing industry. Maintenance can be done either manually or by using automated systems. One of the ways of doing this is by using Internet of Things.

The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators and connectivity which enables these objects to connect and exchange data. Each thing is uniquely identifiable through its embedded computing system but it is able to interoperate within the existing Internet infrastructure [1].

II. ARCHITECTURE OF IOT

Several models of IoT have been proposed by several researchers, practitioners and authors. Most simple architecture is shown in fig 1 and is described below from bottom to top [2].

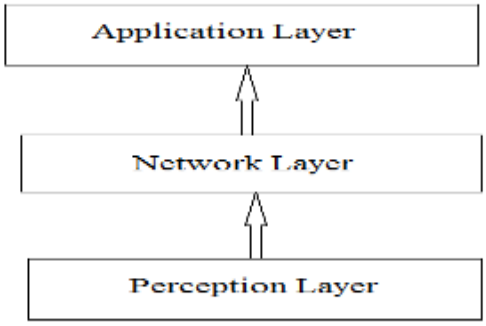

Fig 1 Architecture of IoT

A. Perception Layer

Also known as the sensor layer, perception layer is placed at the bottom of the architecture of IoT. This layer communicates with physical components and devices through sensors, actuators etc. Its main purposes are to connect devices into IoT network and to measure, collect and process the state information through smart devices, passing on the processed information into upper layer.

B. Network Layer

Also known as transmission layer, network layer is located in the middle of the IoT architecture. This layer receives the processed information from the perception layer. It then passes on the information and data to the IoT hub, appliances and devices through integrated networks. Different communication technologies like Bluetooth, Wi-Fi etc. are integrated in this layer.

C. Application Layer

Also known as the business layer, application layer is implemented as the top layer of the IoT architecture. This layer obtains the data conveyed from the network layer and uses the data to furnish needed service or operations. Many applications exist in this layer which includes smart grid, smart cities, smart transportation etc.

Several applications of IoT include industries, agriculture, food processing industry, environmental monitoring, security surveillance, healthcare service industry, safer mining production, transportation and logistics, fire fighting [3]. Other emerging area of IoT is industrial IoT [4]. With this, the current paper presents an industrial IoT model for maintenance system of an industry or a power plant.

III. BLOCK DIAGRAM OF PROPOSED MODEL

The proposed IoT model consists of Raspberry Pi, a sensor kit and some other hardware components like monitor, keyboard and mouse. Its block diagram is shown in fig 2.

A. Sensing the values of parameters using sensor kit

This block corresponds to the perception layer of the IoT architecture. In this block, different sensors like potentiometer, LDR, temperature sensor are used to fetch the values of different parameters.

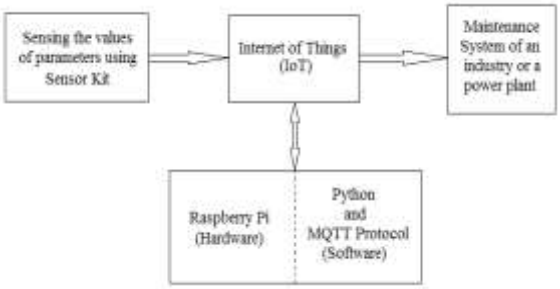

Fig 2 Block diagram of proposed IoT model

B. Internet of Things (IoT)

This block represents the network layer of IoT architecture. This block consists of Raspberry Pi with other devices which compose the major hardware part. Python is used as the programming language to interact with the system. MQTT protocol is used to collect measured data. An android application called MQTT Dash is used for establishing the connection.

C. Maintenance system of an industry or a power plant

This block is analogous to application layer of IoT architecture. This block represents applications like smart grid, smart transportation etc. come under this layer. Since maintenance system is the area to which application of IoT is being proposed, maintenance system comes under application layer of IoT.

IV. SYSTEM DESCRIPTION

Different boards used for making systems related to IoT include Arduino Uno, Raspberry Pi, Tessel, Adafruit Flora, C.H.I.P., Particle Photon, etc. Among these, Raspberry Pi is selected because of its advantages like lesser cost and low energy consumption, simple to use nature, versatility and compact size [5].

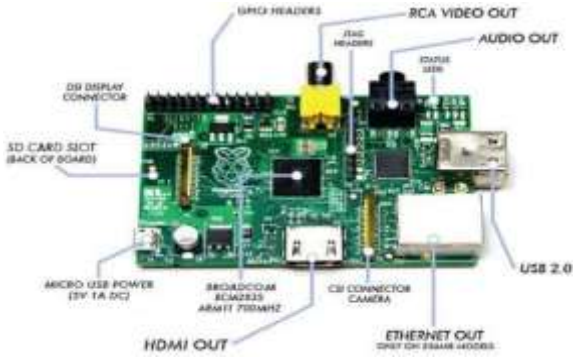

Fig 3 Overview of parts of Raspberry Pi 3 Model B

Several generations of Raspberry Pi's have been released. The major releases of Raspberry Pi include Model A and Model B. For the current model of IoT, Raspberry Pi 3 Model B has been used. The overview of different parts of Raspberry Pi 3 Model B is shown in fig 3.

As discussed earlier, perception layer interacts with physical devices and components through sensors, actuators etc. It is used to connect things into IoT network, and to measure, collect and process the

information. A kit equipped with required sensors and converters is used for this purpose. This kit consists of different sensors to fetch the values of critical parameters. This kit has a board which consists of two LEDs, a buzzer, two push-buttons, temperature sensor, light sensor and a potentiometer with an analog to digital convertor chipset (MCP3008). Fig 4 shows the image of Raspberry Pi with sensor kit.

Fig 4 Raspberry Pi with sensor kit

Various programming languages used to communicate with Raspberry Pi include, Python, C, C++, Java, Ruby, Scratch etc. Among these, Python is selected for programming because of its advantages like, easy syntax, readability, free availability, safety, extensible nature, Graphical User Interface (GUI) programming support, availability of large standard library, dynamic nature, portability, integrated nature etc [6].

As discussed in the architecture of IoT, various communication technologies are integrated in the network layer. Voice over Internet Protocol, Session Initiation Protocol, Message Queuing Telemetry Transport Protocol, ZigBee, Extensible Messaging and Presence Protocol are such communication technologies. Among these, MQTT Protocol is selected because of its advantages like more efficient information distribution, increased scalability, reduction in network bandwidth consumption, well suitability for remote sensing and control, ability to maximize available bandwidth and for having a robust ecosystem.

MQTT is a protocol that specifically sends data from devices of IoT and is supported by most microcontrollers and systems. This is an ISO standard, publish-subscribe based messaging protocol. It works on the top of Transmission Control Protocol (TCP)/Internet Protocol (IP). MQTT was invented by Dr. Andy Stanford-Clark of IBM and Arien Nipper of Arcom (now Eurotech) in 1999 [7].

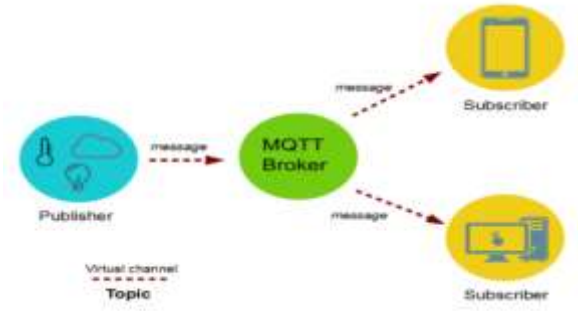

Fig 5 Flow of data between publisher and subscriber in MQTT Protocol

This protocol decouples a client that publishes a message (publisher) to other clients that receive the message (subscriber). This is an asynchronous protocol which does not block the client when it waits for the message. Also, it does not require that the client and the publisher are connected simultaneously. This protocol uses a virtual channel called topic that connects a publisher to its subscribers. MQTT broker handles this topic. Through this virtual channel, the publisher is decoupled from the subscribers and the MQTT clients i.e., publishers or subscribers do not have to know each other in order to exchange the data. This makes this protocol highly scalable without any contact between publisher and subscriber. Figure 5 shows the flow of data between the publisher and the subscriber.

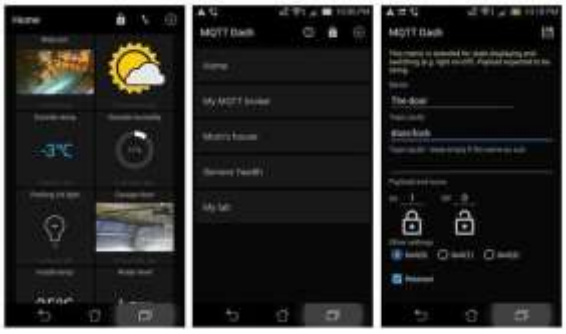

Fig 6 Screenshots of MQTT Dash running in Android mobile phone

Different applications that can be used to establish a connection between Raspberry Pi and a mobile or a computer include MQTT Dash, MQTT Client, IoT MQTT Dashboard, IoT MQTT Panel, MQTT Buddy, Linear MQTT Dashboard, MQTT Alarm Control Panel, MQTT Broker Pro, MQTT Widget and many more. Among these, MQTT Dash is selected because of its high rating and user friendly nature. Fig 6

shows the image of different screenshots of MQTT Dash in an Android mobile phone.

V. PROGRAMMING FOR THE PROPOSED IOT SYSTEM

With the above description about the system, the programming part of the system will be discussed in this section. This programming comes under the network layer of IoT architecture. This includes fetching the values of parameters and converting them into the required forms such as temperature, voltage etc.

At first, few programs which are used to give an indication of breakdown or some failure in the form of blinking of LED, pushing of button etc. will be discussed in the coming section.

A. Code for LED Blinking

This program is used for blinking LED at regular intervals of time. In this program, the RPi.GPIO module is imported. After this, channels 11 and 7 are set as output channels. In the program level, both the channels are alternatively set to HIGH and LOW values which results in blinking of LED. After this, cleanup is done at the end of the program.

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode (GPIO.BOARD)
GPIO.setup (11,GPIO.OUT)
GPIO.setup (7,GPIO.OUT)
while 1:
try:
GPIO.output(11,GPIO.HIGH)
sleep (1)
GPIO.output (11,GPIO.LOW)
sleep (1)
GPIO.output(7,GPIO.HIGH)
sleep (1)
GPIO.output (7,GPIO.LOW)
sleep (1)
except KeyboardInterrupt:
GPIO.cleanup()
```

B. Code for Push Button

This program is used to push ON and OFF button in the sensor kit. In this program, the RPi.GPIO module is imported same as in the previous one. The board numbering scheme is set to follow the pin numbers. And in order to push the button, the 12^{th} channel has been defined as pull down (GPIO.PUD_DOWN). And in the programming level, the button is pushed. And cleanup is done.

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode (GPIO.BOARD)
GPIO.setup(12,GPIO.IN,pull_up_down=GPIO.PUD_
DOWN)
while 1:
try:
if GPIO.input(12)==True:
print “1”
else:
print “0”
except KeyboardInterrupt:
GPIO.cleanup()
```

Now, few other programs which sense the values of parameters such as temperature, illumination etc using sensors such as potentiometer, LDR, temperature sensor are discussed in the coming section.

C. Code for measuring voltage with Potentiometer and ADC

As described in the previous program, the SPI interface is put together with RPi. Voltage is measured using potentiometer. By using ADC, it is converted into digital value and finally expressed in terms of Volts.

```
import spidev, time
spi = spidev. SpiDev()
spi. open(0, 0)
def analog_read(channel):
r = spi. xfer2([1,(8+channel<< 4,0])
adc_out = ((r[1] &3) << 8) + r[2]
return adc_out
while True:
reading = analog_read(0)
voltage = reading * 3.3 / 1024
print(“Potentiometer value =%d\t\t Voltage
f=%f" % (reading, voltage))
time. sleep(1)
```

D. Code for measuring resistance with LDR and ADC

Same as the above program, the SPI interface is put with RPi. Resistance is measured using an LDR.

With suitable conversion, the resistance is converted into voltage and expressed in terms of Volts (V).

```
import spidev, time
spi = spidev. SpiDev()
spi. open(0, 0)
def analog_read(channel):
r = spi. xfer2([1,(8+channel)<<4, 0])
adc_out = ((r[1] &3) << 8) + r[2]
return adc_out
while True:
reading = analog_read(0)
voltage = reading * 3.3 / 1024
print("LDR value =%d\t\tVoltage f=%f" % (reading, voltage))
time. sleep(1)
```

With all the hardware and software systems which were discussed in the previous sections, parameters like temperature, illumination etc can be determined either in the form of voltage or resistance. Based on the critical value for each parameter, suitable corrective actions like repair or replacement of worn out part can be done.

VI. RESULTS AND DISCUSSIONS

The current paper proposed an IoT system wherein few parameters which play a vital role in determining the health of a machine in a manufacturing industry or a power generation plant are monitored periodically. As part of this, in previous sections, the block diagram, hardware system and programming part of the proposed IoT system were discussed. With this, the results of the proposed system will be discussed in this section.

A. Result of execution of program for LED Blinking

This program has been written to give visual indication whenever the value of a parameter exceeds the predefined critical value. Fig 7 shows the Result of execution of Program for LED Blinking. The experimental setup consists of Raspberry Pi connected to sensor kit and other accessories like keyboard, mouse and monitor. Here, the two LEDs blink alternatively at regular intervals of time of one second as set in the program level. This time gap between blinking can be modified according to the requirements.

Fig 7 Result of execution of Program for LED Blinking

B. Result of execution of program for Push Button

This program is written to ensure that the workpiece or cutting tool is placed properly in its position inside the machine before starting the machining operation. Here, the program is written such that when the workpiece or tool is placed exactly in its position, button is pushed which indicates that the machining operation can be started. Fig 8 shows the result of execution of program for push button. The experimental setup consists of Raspberry Pi with keyboard and sensor kit.

In all the above programs different methods of actuations such as blinking of LED, pushing of button are used to give indication whenever the value of a parameter exceeds the predefined critical or threshold value. Any of the above methods can be used for getting an alert message.

Fig 8 Result of execution of program for push button

With this, the results of execution of few other programs where sensors of different types are used to fetch the values of different parameters are analysed in subsequent stages.

C. Result of execution of program for measuring voltage with Potentiometer and ADC

In this program also the SPI interface is put together with RPi. Voltage is measured using potentiometer. By using ADC, it is converted into digital value and

finally expressed in terms of Volts. Fig 9 shows the voltage values sensed by potentiometer.

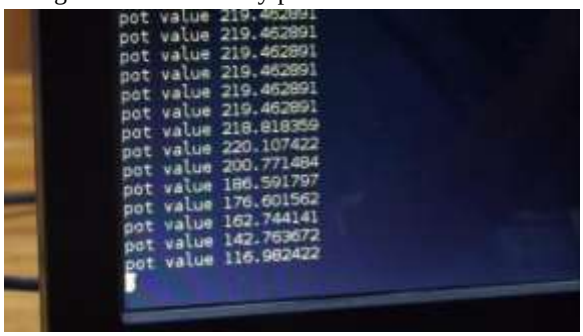

Fig 9 Values of voltage measured using Potentiometer

D. Result of execution of program for measuring resistance with LDR and ADC

Here, the SPI interface is put with RPi. Resistance is measured using an LDR. With suitable conversion, the resistance is converted into voltage and expressed in terms of Volts (V). Fig 10 shows the voltage measured in terms of resistance using LDR.

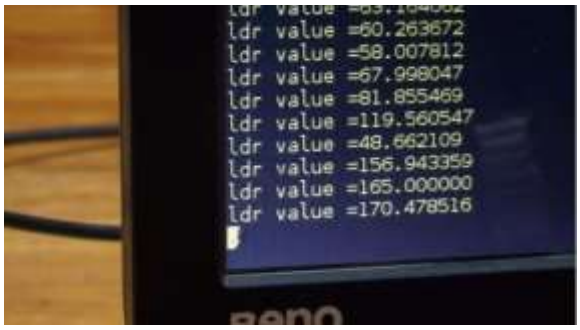

Fig 10 Values of voltage measured in terms of resistance using LDR

Here, the sensors fetch the values of parameters in the form of voltage or resistance. These values can be converted back to their original forms such as temperature, illumination etc. using suitable conversion factors. The converted values are then compared against standard threshold or critical values. Based on the comparison, corrective measures like repairing or replacement of the worn out part or machine can be done.

VII. CONCLUSION

The paper proposed an IoT system with Raspberry Pi and a sensor kit with other accessories like keyboard, mouse, monitor etc. The sensor kit includes different sensors like temperature sensor, potentiometer, LDR with an ADC. These sensors sense the values of the parameters to be monitored at regular intervals of time.

With this, one can automate the maintenance system of any manufacturing industry or a power plant. With the proposed IoT system, maintenance becomes very easy under conditions wherein equipments are subjected to extreme temperature, while handling toxic substances, when installed at locations which are not easily accessible by human beings. Manual maintenance more often consumes large amount of energy, time, money and involves large amount of risk. Added to this, under extreme or critical conditions which are stated above, manual maintenance becomes a very difficult task. Moreover, manual maintenance may not accurate always and may involve overlooking or underestimating the observation. By automating the maintenance system, human intervention can be reduced to a greater extent which increases the accuracy of the maintenance system. This saves a lot of time, money, labour, increases safety level, quality of production and improves the efficiency of the manufacturing industry. Hence the proposed IoT system can become a promising solution to automate the maintenance system.

VIII. FUTURE SCOPE

After discussing the results of the current project and concluding them, the future works and scope of the current system are discussed in this section.

A. To store the fetched values of different parameters in a cloud so that they are available whenever required.
B. To set the critical values for parameters in the program level itself.
C. To modify programs to give an indication or an alarm whenever the fetched values from the sensors exceed the corresponding critical values.
D. To include the corrective measures to be taken in the program itself, based on the comparison.

With the incorporation of the above mentioned improvements with the current system, the proposed project becomes a promising solution to overcome the present limitations of maintenance system of any manufacturing industry or a power plant.

ACKNOWLEDGEMENT

The authors are thankful to JSS Science and Technology University, Mysuru for providing necessary facilities and support.

REFERENCES

[1] https://en.m.wikipedia.org/wiki/Internet_of_things
[2] Jie Lin et al., "A survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy and Applications" IEEE INTERNET OF THINGS JOURNAL, VOL. 4 NO. 5, OCTOBER 2017
[3] Li Da Xu et al., "Internet of Things in Industries: A Survey" IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 10, NO. 4, NOVEMBER 2014
[4] Jiafu Wan et al., "Software-Defined Industrial Internet of Things in the Context of Industry 4.0" IEEE SENSORS JOURNAL, VOL. 16, NO. 20, OCTOBER 15, 2016 7373
[5] https://en.m.wikipedia.org/wiki/raspberry_Pi
[6] https://en.m.wikipedia.org/wiki/Python
[7] https://en.m.wikipedia.org/wiki/MQTT