

Smart Data Centre Monitoring System Based On Internet of Things (IoT) (Study Case : Pustipanda UIN Jakarta)

¹⁾Feri Fahrianto, ²⁾Nenny Anggraini, ³⁾Hendra Bayu Suseno, ⁴⁾ Almas Shabrina, ⁵⁾ Alfatta Reza

Universitas Islam Negeri Syarif Hidayatullah Jakarta
Tel : (021) 7493315 Fax : (021) 7493315

ABSTRACT

State Islamic University Syarif Hidayatullah Jakarta as rapidly growing university toward world-class university placed in the edge of Jakarta has a data center run by its unit of organization called Pustipanda (The Center of Information Technology and Database). The team has made observations during the last 1 year and found that these data center lately has an incident occurred where its hardware is found defected caused by instability of voltage and current from power supply, several uninterrupted power supply installed that should cover this problem is also unable to answer cause the current provided is insufficient for power up computers in period of time and it was verified by Mr. Nasrul Hakiem as the head of the data center through the interview team has done.

Internet of Things, based on ITU-T 2015, some objects are able to transmit data among object by using Internet connection. It means by this technology the Internet used has been widely changed, from human to machine communication now also become machine to machine communications. As the number of Internet-connected devices grows, the amount of traffic they generate is expected to rise significantly. Cisco estimates that Internet traffic generated by non-PC devices will rise from 40% in 2014 to just under 70% in 2019. Cisco also forecasts that the number of "Machine to Machine" ("M2M") connections (including in industrial, home, healthcare, automotive, and other IoT verticals) will rise from 24% of all connected devices in 2014 to 43% in 2019.[13] By using this technology a small object or device is able to implement into electrical system in data center in order to monitor abnormality of voltage and current from electrical supply and also implemented in the surroundings in order to monitor the temperature condition. Object implemented in the data center system are able to monitor and send the data into monitoring system that researchers would build with scrum methodology

Keyword :

Internet of Thing, data center

1. Background

Data center by definition is a place where collection group of computer connected build a logical server or several physical servers for servicing a request from Internet users. Many issues or problems are faced by the data center. One of the issues is electrical problem, voltage and current stability from electric power supply should constantly power up the power consumption of computers, when the power is discontinued. It causes a big problem for providing services and caused hardware damage. The other problem is a temperature issue; the air

condition in data center should be kept cool in order to freeze the processor of computer after heating up by the electric current because of data processing.

State Islamic University Syarif Hidayatullah Jakarta as rapidly growing university toward world-class university placed in the edge of Jakarta has a data center run by its unit of organization called Pustipanda (The Center of Information Technology and Database).

The team has made observations during the last 1 year and found that these data center lately has an incident occurred where its hardware is found

defected caused by instability of voltage and current from power supply, several uninterrupted power supply installed that should cover this problem is also unable to answer cause the current provided is insufficient for power up computers in period of time and it was verified by Mr. Nasrul Hakiem as the head of the data center through the interview team has done.

Internet of Things, based on ITU-T 2015, some objects are able to transmit data among object by using Internet connection. It means by this technology the Internet used has been widely changed, from human to machine communication now also become machine to machine communications.

By using this technology a small object or device is able to implement into electrical system in data center in order to monitor abnormality of voltage and current from electrical supply and also implemented in the surroundings in order to monitor the temperature condition. Object implemented in the data center system are able to monitor and send the data into monitoring system that researchers would build.

This monitoring system should run twenty four hours a day, seven days a week, and three hundreds sixty five a year, this system should record all the abnormality and send a alarm to several authorize person in real time so the potential defective hardware could be avoided.

1.1. Objectives

Objectives in this research are :

1. Build a monitoring system in order to record all abnormality in electrics supply and air temperature in data center.
2. Build early warning system to inform/alarm the authorize person instantly when the abnormality occurred.

1.2. Limitation

1. This system should record electric current consumption, humidity, and temperature issue because of data processing and send an alarm to several authorize person in real time.
2. This system able to prevent a serious damage in sever by give an early warning to administrator and authorize person to state further corrective action.
3. Monitoring reportall the abnormality is viewed through web.

2. Review of Related Literature

2.1. Interactive Device

All of the objects are built using microcontroller follow a very simple pattern that as known the

“Interactive Device”. The Interactive Device is an electronic circuit that is able to sense the environment using sensors (electronic components that convert real-world measurements into electrical signals). The device processes the information it gets from the sensors with behaviour that’s implemented as software. The device will then be able to interact with the world using actuators, electronic components that can convert an electric signal into a physical action.^[8]

2.2. Scrum Methodology

Scrum is one of software and hardware engineering methods by using AGILE approach which has strong points in team collaboration and iteration incremental product review for final results.^[5]

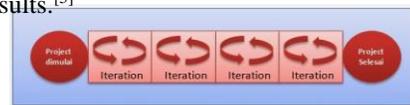


Fig 2. Scrum Methods

3. Research Methodology

Methods are used in this research are as follow:

3.1. Literature Reviews

At this step, literature review is done by collecting all books and articles that are relevant to this research. Several textbooks are used in this research in example books that contain theory about UML, Embedded System, Electronics Circuits, and Database System. Articles that has similarity with this research also is collected, some article found in Journal and some articles found in proceeding of international conference.

3.2. Observation and interview

Observation is done to research object PUSTIPANDA (a data center unit in UIN Jakarta), specially its power supply and server room from May 2016-October 2016 and in this research, interview is done by asking the chief of PUSTIPANDA Nashrul Hakiem, Ph.D as the man who responsible running the operation of Data Center.

4. Analysis and Design

system development methodology that reseracher used is SCRUM. The SCRUM methodology has several steps and iteration. The first step in this methodology is collecting all data and facts about Pustipanda condition, and then analyze the facts, observe the environment, problem identification, design and programming, testing the system, and deployment in site.

4.1. Collecting All Data

The researcher tries to collect all data and facts about pustipanda 's data center, starting with reading the log system and history provided. Several incidents are found in the log system. Few months ago the power of electricity is black out about an hour and causes the storage server dysfunction. Three from eight hard drive from storage server found to be defect. The service level agreement from pustipanda is disrupted. From here the officials from pustipanda decided to monitor the electricity power supplies from the electricity company in order to avoid this kind of incident is occurred again

4.2. Analyze the Facts.

4.2.1. Scope

The first thing should be done in determining the all elements needed by the system is define the scope. The scope that the researcher do in this research is in Pustipanda data center. The researcher need to monitor all the electricity power from the electric company. This electricity monitoring system use sensor that captured data and send the data to a server automatically and periodically updated. The monitoring system should be real-time and stored to the server. Afterwards the system able to show and visualize the graph of power used remotely and locally.

4.2.2. The Hardware Requirement Analysis

Designing a prototype power monitoring system, researcher used several hardware and software for build a complete system. The Hardware uses microcontroller components, sensors for read current, temperature, and humidity. The specification for each component choose is also important to make sure the system working perfectly. Following is list of component used in this power monitoring system :

Table 1 Hardware Analysis

| No. | Components | Qty | Function |
|-----|----------------------------|-----|--|
| 1. | Raspberry Pi 3 | 8 | As center of computation and processing in the system. |
| 2. | Arduino Uno | 8 | Acting as Analogue digital Converter from current sensor |
| 3. | SCT 013 100A | 4 | Current Sensor |
| 4. | DHT-22 | 4 | Temperature and Humidity Sensor |
| 5. | Power Source USB 5 V 2.0 A | 8 | Giving Power for Raspberry Pi |

4.3. Current System Analysis

The power system that uses in pustipanda is using UPS and GENSET for interruption in power network from electric company. The monitoring system that Pustipanda uses only

using a panel with led monitoring system. More advanced and reliable system for monitoring a electricity problem is needed for preventing a further damage and defection in the server working in pustipanda data centre. By this condition the researcher design and build a power monitoring system based IoT. This system able to capture current, temperature, and humidity and stored in database, and show the graphic as visualisation for further analysis. By this system, abnormalities in power also detected completely by the system.

4.3.1. Requirement Analysis

Requirement will be divided into two section, the first section is the fucntional requirements, the definition of functional requirements is activity and services that the system should provide in order to make the system working. The second section is Non-functional requirements, by the definition the non-functional requirements is featerus that the system has in order to make the system more easy to used, more advanced to operate.

4.3.1.1. Functional Requirements

The system build should have fucntional requirements as follow :

1. Able to monitor and capture the current, temperature and humidity from the environment.
2. Stored the data into database.
3. Visual the data into graph and digital meter

4.3.1.2. Nonfunctional Requirements

Nonfunctional Requirements from the system build as follow as shown in Table 3

Table 2 Nonfunctional requirement

| Jenis kebutuhan | Penjelasan |
|-----------------------------|--|
| 1. Graphical User Interface | a. Graphical user interface using graphic and digital meter. |
| 2. Database | b. Giving relational and high capacity of data |
| 3. Historical Data | c. Able to giving a historical data |

4.3.2. Design system



Fig 3. Design System

At figure 3 explained about the system power monitoring system based on IoT. Those monitoring system monitor the power abnormality from PLN by using SCT 013 and also the deviation of chamber temperature following is the explanation from the system:

1. Raspberry Pi3 use as processing unit and wifi connection from arduino and database server.
2. SCT 013 sensor, current from the PLN cable is measured and digitalize by using arduino.
3. Arduino using as analogue digital converter and send the data as serial data.
4. DHT-22 temperature and humidity sensor is used for capturing the temperature in the server room.
5. Raspberry pi send the data into database server.
6. Database server send the data to web server and visualize the data.
7. User able to monitor the power and the temperature.

From data forwarded to database server and database server will push to web server. By using webbrowser the data are viewed to user in realtime.

4.4. Workshop Design

Design or designing a system could be define as task that focus on detail specification of solution based on computer.

4.4.1. Application Design

4.4.1.1. Identification of User

The actors is something that act outside the system. For detail explanation will be given in the table 4

Table 4 Requirement Actor dan Use Case

| Requirement | Actor | Use Case |
|--|----------------|--|
| 1. Microcontroller send data to server | Mikrokontroler | Send data to database server |
| 2. User/Staff monitorize the graph | User | Monitor and has information from the graph |

4.4.1.4. Class Diagram

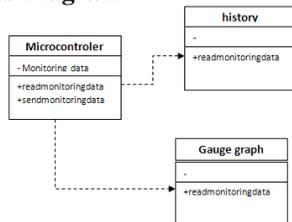


Figure 9. Application Class Diagram

4.4.1.7. Design of Database

The system needs database to store the information captured by sensor. The Desgin of database should support the information system wich based on web. Database is obtained by integrated all data which is connecting each other, mapping and entity class.

1. Information Table Current, Power, Temperature and Humidity

Namatabel : info

Primary key: id

Foreign key : -

| No | Nama Field | Tipe/lebar | keterangan |
|----|------------|-------------|--------------------|
| 1 | Id | Int(10) | Id server |
| 2 | Server | Varchar(30) | IP Server |
| 3 | Arus | Float | Data arus |
| 4 | Suhu | Float | Data suhu |
| 5 | Kelembaban | Float | Data kelembaban |
| 6 | Tanggal | Date | Tanggal monitoring |
| 7 | waktu | Time | Jam monitoring |

4.4.2. Interface Design

In this step, interface is made by rules of human computer interation. By using this rule, the outlook and interface are operated easily by user for monitoring the server with graphic visualization

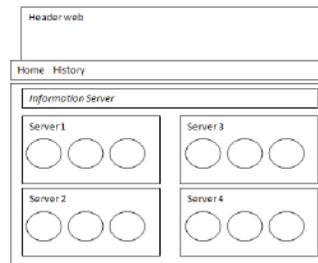


Figure 10. Application Main Page

4.4.3. Coding

```

1 import urllib, urllib
2 import sys
3 from time import sleep
4 import Adafruit_DHT
5
6 sensor = Adafruit_DHT.DHT22
7 pin = 2
8
9 data = {}
10 server_ip = "192.168.1.1"
11 server_url = "192.168.1.1/"
12
13 #!/usr/bin/python
14
15 def sendData(suhu, kelembaban):
16     url = server_url + "sendData.php"
17     data = {'suhu': suhu, 'kelembaban': kelembaban}
18     req = urllib.urlencode(data)
19     url = server_url + req
20     response = urllib.urlopen(url)
21     data = response.read()
22     print "Data: %s" % data
23
24 #!/usr/bin/python
25
26 def sendData(suhu, kelembaban):
27     url = server_url + "sendData.php"
28     data = {'suhu': suhu, 'kelembaban': kelembaban}
29     req = urllib.urlencode(data)
30     url = server_url + req
31     response = urllib.urlopen(url)
32     data = response.read()
33     print "Data: %s" % data
34
35 #!/usr/bin/python
36
37 def sendData(suhu, kelembaban):
38     url = server_url + "sendData.php"
39     data = {'suhu': suhu, 'kelembaban': kelembaban}
40     req = urllib.urlencode(data)
41     url = server_url + req
42     response = urllib.urlopen(url)
43     data = response.read()
44     print "Data: %s" % data
45
46 #!/usr/bin/python
47
48 def sendData(suhu, kelembaban):
49     url = server_url + "sendData.php"
50     data = {'suhu': suhu, 'kelembaban': kelembaban}
51     req = urllib.urlencode(data)
52     url = server_url + req
53     response = urllib.urlopen(url)
54     data = response.read()
55     print "Data: %s" % data
56
57 #!/usr/bin/python
58
59 def sendData(suhu, kelembaban):
60     url = server_url + "sendData.php"
61     data = {'suhu': suhu, 'kelembaban': kelembaban}
62     req = urllib.urlencode(data)
63     url = server_url + req
64     response = urllib.urlopen(url)
65     data = response.read()
66     print "Data: %s" % data
67
68 #!/usr/bin/python
69
70 def sendData(suhu, kelembaban):
71     url = server_url + "sendData.php"
72     data = {'suhu': suhu, 'kelembaban': kelembaban}
73     req = urllib.urlencode(data)
74     url = server_url + req
75     response = urllib.urlopen(url)
76     data = response.read()
77     print "Data: %s" % data
78
79 #!/usr/bin/python
80
81 def sendData(suhu, kelembaban):
82     url = server_url + "sendData.php"
83     data = {'suhu': suhu, 'kelembaban': kelembaban}
84     req = urllib.urlencode(data)
85     url = server_url + req
86     response = urllib.urlopen(url)
87     data = response.read()
88     print "Data: %s" % data
89
90 #!/usr/bin/python
91
92 def sendData(suhu, kelembaban):
93     url = server_url + "sendData.php"
94     data = {'suhu': suhu, 'kelembaban': kelembaban}
95     req = urllib.urlencode(data)
96     url = server_url + req
97     response = urllib.urlopen(url)
98     data = response.read()
99     print "Data: %s" % data
100
101 #!/usr/bin/python
102
103 def sendData(suhu, kelembaban):
104     url = server_url + "sendData.php"
105     data = {'suhu': suhu, 'kelembaban': kelembaban}
106     req = urllib.urlencode(data)
107     url = server_url + req
108     response = urllib.urlopen(url)
109     data = response.read()
110     print "Data: %s" % data
111
112 #!/usr/bin/python
113
114 def sendData(suhu, kelembaban):
115     url = server_url + "sendData.php"
116     data = {'suhu': suhu, 'kelembaban': kelembaban}
117     req = urllib.urlencode(data)
118     url = server_url + req
119     response = urllib.urlopen(url)
120     data = response.read()
121     print "Data: %s" % data
122
123 #!/usr/bin/python
124
125 def sendData(suhu, kelembaban):
126     url = server_url + "sendData.php"
127     data = {'suhu': suhu, 'kelembaban': kelembaban}
128     req = urllib.urlencode(data)
129     url = server_url + req
130     response = urllib.urlopen(url)
131     data = response.read()
132     print "Data: %s" % data
133
134 #!/usr/bin/python
135
136 def sendData(suhu, kelembaban):
137     url = server_url + "sendData.php"
138     data = {'suhu': suhu, 'kelembaban': kelembaban}
139     req = urllib.urlencode(data)
140     url = server_url + req
141     response = urllib.urlopen(url)
142     data = response.read()
143     print "Data: %s" % data
144
145 #!/usr/bin/python
146
147 def sendData(suhu, kelembaban):
148     url = server_url + "sendData.php"
149     data = {'suhu': suhu, 'kelembaban': kelembaban}
150     req = urllib.urlencode(data)
151     url = server_url + req
152     response = urllib.urlopen(url)
153     data = response.read()
154     print "Data: %s" % data
155
156 #!/usr/bin/python
157
158 def sendData(suhu, kelembaban):
159     url = server_url + "sendData.php"
160     data = {'suhu': suhu, 'kelembaban': kelembaban}
161     req = urllib.urlencode(data)
162     url = server_url + req
163     response = urllib.urlopen(url)
164     data = response.read()
165     print "Data: %s" % data
166
167 #!/usr/bin/python
168
169 def sendData(suhu, kelembaban):
170     url = server_url + "sendData.php"
171     data = {'suhu': suhu, 'kelembaban': kelembaban}
172     req = urllib.urlencode(data)
173     url = server_url + req
174     response = urllib.urlopen(url)
175     data = response.read()
176     print "Data: %s" % data
177
178 #!/usr/bin/python
179
180 def sendData(suhu, kelembaban):
181     url = server_url + "sendData.php"
182     data = {'suhu': suhu, 'kelembaban': kelembaban}
183     req = urllib.urlencode(data)
184     url = server_url + req
185     response = urllib.urlopen(url)
186     data = response.read()
187     print "Data: %s" % data
188
189 #!/usr/bin/python
190
191 def sendData(suhu, kelembaban):
192     url = server_url + "sendData.php"
193     data = {'suhu': suhu, 'kelembaban': kelembaban}
194     req = urllib.urlencode(data)
195     url = server_url + req
196     response = urllib.urlopen(url)
197     data = response.read()
198     print "Data: %s" % data
199
200 #!/usr/bin/python
201
202 def sendData(suhu, kelembaban):
203     url = server_url + "sendData.php"
204     data = {'suhu': suhu, 'kelembaban': kelembaban}
205     req = urllib.urlencode(data)
206     url = server_url + req
207     response = urllib.urlopen(url)
208     data = response.read()
209     print "Data: %s" % data
210
211 #!/usr/bin/python
212
213 def sendData(suhu, kelembaban):
214     url = server_url + "sendData.php"
215     data = {'suhu': suhu, 'kelembaban': kelembaban}
216     req = urllib.urlencode(data)
217     url = server_url + req
218     response = urllib.urlopen(url)
219     data = response.read()
220     print "Data: %s" % data
221
222 #!/usr/bin/python
223
224 def sendData(suhu, kelembaban):
225     url = server_url + "sendData.php"
226     data = {'suhu': suhu, 'kelembaban': kelembaban}
227     req = urllib.urlencode(data)
228     url = server_url + req
229     response = urllib.urlopen(url)
230     data = response.read()
231     print "Data: %s" % data
232
233 #!/usr/bin/python
234
235 def sendData(suhu, kelembaban):
236     url = server_url + "sendData.php"
237     data = {'suhu': suhu, 'kelembaban': kelembaban}
238     req = urllib.urlencode(data)
239     url = server_url + req
240     response = urllib.urlopen(url)
241     data = response.read()
242     print "Data: %s" % data
243
244 #!/usr/bin/python
245
246 def sendData(suhu, kelembaban):
247     url = server_url + "sendData.php"
248     data = {'suhu': suhu, 'kelembaban': kelembaban}
249     req = urllib.urlencode(data)
250     url = server_url + req
251     response = urllib.urlopen(url)
252     data = response.read()
253     print "Data: %s" % data
254
255 #!/usr/bin/python
256
257 def sendData(suhu, kelembaban):
258     url = server_url + "sendData.php"
259     data = {'suhu': suhu, 'kelembaban': kelembaban}
260     req = urllib.urlencode(data)
261     url = server_url + req
262     response = urllib.urlopen(url)
263     data = response.read()
264     print "Data: %s" % data
265
266 #!/usr/bin/python
267
268 def sendData(suhu, kelembaban):
269     url = server_url + "sendData.php"
270     data = {'suhu': suhu, 'kelembaban': kelembaban}
271     req = urllib.urlencode(data)
272     url = server_url + req
273     response = urllib.urlopen(url)
274     data = response.read()
275     print "Data: %s" % data
276
277 #!/usr/bin/python
278
279 def sendData(suhu, kelembaban):
280     url = server_url + "sendData.php"
281     data = {'suhu': suhu, 'kelembaban': kelembaban}
282     req = urllib.urlencode(data)
283     url = server_url + req
284     response = urllib.urlopen(url)
285     data = response.read()
286     print "Data: %s" % data
287
288 #!/usr/bin/python
289
290 def sendData(suhu, kelembaban):
291     url = server_url + "sendData.php"
292     data = {'suhu': suhu, 'kelembaban': kelembaban}
293     req = urllib.urlencode(data)
294     url = server_url + req
295     response = urllib.urlopen(url)
296     data = response.read()
297     print "Data: %s" % data
298
299 #!/usr/bin/python
300
301 def sendData(suhu, kelembaban):
302     url = server_url + "sendData.php"
303     data = {'suhu': suhu, 'kelembaban': kelembaban}
304     req = urllib.urlencode(data)
305     url = server_url + req
306     response = urllib.urlopen(url)
307     data = response.read()
308     print "Data: %s" % data
309
310 #!/usr/bin/python
311
312 def sendData(suhu, kelembaban):
313     url = server_url + "sendData.php"
314     data = {'suhu': suhu, 'kelembaban': kelembaban}
315     req = urllib.urlencode(data)
316     url = server_url + req
317     response = urllib.urlopen(url)
318     data = response.read()
319     print "Data: %s" % data
320
321 #!/usr/bin/python
322
323 def sendData(suhu, kelembaban):
324     url = server_url + "sendData.php"
325     data = {'suhu': suhu, 'kelembaban': kelembaban}
326     req = urllib.urlencode(data)
327     url = server_url + req
328     response = urllib.urlopen(url)
329     data = response.read()
330     print "Data: %s" % data
331
332 #!/usr/bin/python
333
334 def sendData(suhu, kelembaban):
335     url = server_url + "sendData.php"
336     data = {'suhu': suhu, 'kelembaban': kelembaban}
337     req = urllib.urlencode(data)
338     url = server_url + req
339     response = urllib.urlopen(url)
340     data = response.read()
341     print "Data: %s" % data
342
343 #!/usr/bin/python
344
345 def sendData(suhu, kelembaban):
346     url = server_url + "sendData.php"
347     data = {'suhu': suhu, 'kelembaban': kelembaban}
348     req = urllib.urlencode(data)
349     url = server_url + req
350     response = urllib.urlopen(url)
351     data = response.read()
352     print "Data: %s" % data
353
354 #!/usr/bin/python
355
356 def sendData(suhu, kelembaban):
357     url = server_url + "sendData.php"
358     data = {'suhu': suhu, 'kelembaban': kelembaban}
359     req = urllib.urlencode(data)
360     url = server_url + req
361     response = urllib.urlopen(url)
362     data = response.read()
363     print "Data: %s" % data
364
365 #!/usr/bin/python
366
367 def sendData(suhu, kelembaban):
368     url = server_url + "sendData.php"
369     data = {'suhu': suhu, 'kelembaban': kelembaban}
370     req = urllib.urlencode(data)
371     url = server_url + req
372     response = urllib.urlopen(url)
373     data = response.read()
374     print "Data: %s" % data
375
376 #!/usr/bin/python
377
378 def sendData(suhu, kelembaban):
379     url = server_url + "sendData.php"
380     data = {'suhu': suhu, 'kelembaban': kelembaban}
381     req = urllib.urlencode(data)
382     url = server_url + req
383     response = urllib.urlopen(url)
384     data = response.read()
385     print "Data: %s" % data
386
387 #!/usr/bin/python
388
389 def sendData(suhu, kelembaban):
390     url = server_url + "sendData.php"
391     data = {'suhu': suhu, 'kelembaban': kelembaban}
392     req = urllib.urlencode(data)
393     url = server_url + req
394     response = urllib.urlopen(url)
395     data = response.read()
396     print "Data: %s" % data
397
398 #!/usr/bin/python
399
400 def sendData(suhu, kelembaban):
401     url = server_url + "sendData.php"
402     data = {'suhu': suhu, 'kelembaban': kelembaban}
403     req = urllib.urlencode(data)
404     url = server_url + req
405     response = urllib.urlopen(url)
406     data = response.read()
407     print "Data: %s" % data
408
409 #!/usr/bin/python
410
411 def sendData(suhu, kelembaban):
412     url = server_url + "sendData.php"
413     data = {'suhu': suhu, 'kelembaban': kelembaban}
414     req = urllib.urlencode(data)
415     url = server_url + req
416     response = urllib.urlopen(url)
417     data = response.read()
418     print "Data: %s" % data
419
420 #!/usr/bin/python
421
422 def sendData(suhu, kelembaban):
423     url = server_url + "sendData.php"
424     data = {'suhu': suhu, 'kelembaban': kelembaban}
425     req = urllib.urlencode(data)
426     url = server_url + req
427     response = urllib.urlopen(url)
428     data = response.read()
429     print "Data: %s" % data
430
431 #!/usr/bin/python
432
433 def sendData(suhu, kelembaban):
434     url = server_url + "sendData.php"
435     data = {'suhu': suhu, 'kelembaban': kelembaban}
436     req = urllib.urlencode(data)
437     url = server_url + req
438     response = urllib.urlopen(url)
439     data = response.read()
440     print "Data: %s" % data
441
442 #!/usr/bin/python
443
444 def sendData(suhu, kelembaban):
445     url = server_url + "sendData.php"
446     data = {'suhu': suhu, 'kelembaban': kelembaban}
447     req = urllib.urlencode(data)
448     url = server_url + req
449     response = urllib.urlopen(url)
450     data = response.read()
451     print "Data: %s" % data
452
453 #!/usr/bin/python
454
455 def sendData(suhu, kelembaban):
456     url = server_url + "sendData.php"
457     data = {'suhu': suhu, 'kelembaban': kelembaban}
458     req = urllib.urlencode(data)
459     url = server_url + req
460     response = urllib.urlopen(url)
461     data = response.read()
462     print "Data: %s" % data
463
464 #!/usr/bin/python
465
466 def sendData(suhu, kelembaban):
467     url = server_url + "sendData.php"
468     data = {'suhu': suhu, 'kelembaban': kelembaban}
469     req = urllib.urlencode(data)
470     url = server_url + req
471     response = urllib.urlopen(url)
472     data = response.read()
473     print "Data: %s" % data
474
475 #!/usr/bin/python
476
477 def sendData(suhu, kelembaban):
478     url = server_url + "sendData.php"
479     data = {'suhu': suhu, 'kelembaban': kelembaban}
480     req = urllib.urlencode(data)
481     url = server_url + req
482     response = urllib.urlopen(url)
483     data = response.read()
484     print "Data: %s" % data
485
486 #!/usr/bin/python
487
488 def sendData(suhu, kelembaban):
489     url = server_url + "sendData.php"
490     data = {'suhu': suhu, 'kelembaban': kelembaban}
491     req = urllib.urlencode(data)
492     url = server_url + req
493     response = urllib.urlopen(url)
494     data = response.read()
495     print "Data: %s" % data
496
497 #!/usr/bin/python
498
499 def sendData(suhu, kelembaban):
500     url = server_url + "sendData.php"
501     data = {'suhu': suhu, 'kelembaban': kelembaban}
502     req = urllib.urlencode(data)
503     url = server_url + req
504     response = urllib.urlopen(url)
505     data = response.read()
506     print "Data: %s" % data
507
508 #!/usr/bin/python
509
510 def sendData(suhu, kelembaban):
511     url = server_url + "sendData.php"
512     data = {'suhu': suhu, 'kelembaban': kelembaban}
513     req = urllib.urlencode(data)
514     url = server_url + req
515     response = urllib.urlopen(url)
516     data = response.read()
517     print "Data: %s" % data
518
519 #!/usr/bin/python
520
521 def sendData(suhu, kelembaban):
522     url = server_url + "sendData.php"
523     data = {'suhu': suhu, 'kelembaban': kelembaban}
524     req = urllib.urlencode(data)
525     url = server_url + req
526     response = urllib.urlopen(url)
527     data = response.read()
528     print "Data: %s" % data
529
530 #!/usr/bin/python
531
532 def sendData(suhu, kelembaban):
533     url = server_url + "sendData.php"
534     data = {'suhu': suhu, 'kelembaban': kelembaban}
535     req = urllib.urlencode(data)
536     url = server_url + req
537     response = urllib.urlopen(url)
538     data = response.read()
539     print "Data: %s" % data
540
541 #!/usr/bin/python
542
543 def sendData(suhu, kelembaban):
544     url = server_url + "sendData.php"
545     data = {'suhu': suhu, 'kelembaban': kelembaban}
546     req = urllib.urlencode(data)
547     url = server_url + req
548     response = urllib.urlopen(url)
549     data = response.read()
550     print "Data: %s" % data
551
552 #!/usr/bin/python
553
554 def sendData(suhu, kelembaban):
555     url = server_url + "sendData.php"
556     data = {'suhu': suhu, 'kelembaban': kelembaban}
557     req = urllib.urlencode(data)
558     url = server_url + req
559     response = urllib.urlopen(url)
560     data = response.read()
561     print "Data: %s" % data
562
563 #!/usr/bin/python
564
565 def sendData(suhu, kelembaban):
566     url = server_url + "sendData.php"
567     data = {'suhu': suhu, 'kelembaban': kelembaban}
568     req = urllib.urlencode(data)
569     url = server_url + req
570     response = urllib.urlopen(url)
571     data = response.read()
572     print "Data: %s" % data
573
574 #!/usr/bin/python
575
576 def sendData(suhu, kelembaban):
577     url = server_url + "sendData.php"
578     data = {'suhu': suhu, 'kelembaban': kelembaban}
579     req = urllib.urlencode(data)
580     url = server_url + req
581     response = urllib.urlopen(url)
582     data = response.read()
583     print "Data: %s" % data
584
585 #!/usr/bin/python
586
587 def sendData(suhu, kelembaban):
588     url = server_url + "sendData.php"
589     data = {'suhu': suhu, 'kelembaban': kelembaban}
590     req = urllib.urlencode(data)
591     url = server_url + req
592     response = urllib.urlopen(url)
593     data = response.read()
594     print "Data: %s" % data
595
596 #!/usr/bin/python
597
598 def sendData(suhu, kelembaban):
599     url = server_url + "sendData.php"
600     data = {'suhu': suhu, 'kelembaban': kelembaban}
601     req = urllib.urlencode(data)
602     url = server_url + req
603     response = urllib.urlopen(url)
604     data = response.read()
605     print "Data: %s" % data
606
607 #!/usr/bin/python
608
609 def sendData(suhu, kelembaban):
610     url = server_url + "sendData.php"
611     data = {'suhu': suhu, 'kelembaban': kelembaban}
612     req = urllib.urlencode(data)
613     url = server_url + req
614     response = urllib.urlopen(url)
615     data = response.read()
616     print "Data: %s" % data
617
618 #!/usr/bin/python
619
620 def sendData(suhu, kelembaban):
621     url = server_url + "sendData.php"
622     data = {'suhu': suhu, 'kelembaban': kelembaban}
623     req = urllib.urlencode(data)
624     url = server_url + req
625     response = urllib.urlopen(url)
626     data = response.read()
627     print "Data: %s" % data
628
629 #!/usr/bin/python
630
631 def sendData(suhu, kelembaban):
632     url = server_url + "sendData.php"
633     data = {'suhu': suhu, 'kelembaban': kelembaban}
634     req = urllib.urlencode(data)
635     url = server_url + req
636     response = urllib.urlopen(url)
637     data = response.read()
638     print "Data: %s" % data
639
640 #!/usr/bin/python
641
642 def sendData(suhu, kelembaban):
643     url = server_url + "sendData.php"
644     data = {'suhu': suhu, 'kelembaban': kelembaban}
645     req = urllib.urlencode(data)
646     url = server_url + req
647     response = urllib.urlopen(url)
648     data = response.read()
649     print "Data: %s" % data
650
651 #!/usr/bin/python
652
653 def sendData(suhu, kelembaban):
654     url = server_url + "sendData.php"
655     data = {'suhu': suhu, 'kelembaban': kelembaban}
656     req = urllib.urlencode(data)
657     url = server_url + req
658     response = urllib.urlopen(url)
659     data = response.read()
660     print "Data: %s" % data
661
662 #!/usr/bin/python
663
664 def sendData(suhu, kelembaban):
665     url = server_url + "sendData.php"
666     data = {'suhu': suhu, 'kelembaban': kelembaban}
667     req = urllib.urlencode(data)
668     url = server_url + req
669     response = urllib.urlopen(url)
670     data = response.read()
671     print "Data: %s" % data
672
673 #!/usr/bin/python
674
675 def sendData(suhu, kelembaban):
676     url = server_url + "sendData.php"
677     data = {'suhu': suhu, 'kelembaban': kelembaban}
678     req = urllib.urlencode(data)
679     url = server_url + req
680     response = urllib.urlopen(url)
681     data = response.read()
682     print "Data: %s" % data
683
684 #!/usr/bin/python
685
686 def sendData(suhu, kelembaban):
687     url = server_url + "sendData.php"
688     data = {'suhu': suhu, 'kelembaban': kelembaban}
689     req = urllib.urlencode(data)
690     url = server_url + req
691     response = urllib.urlopen(url)
692     data = response.read()
693     print "Data: %s" % data
694
695 #!/usr/bin/python
696
697 def sendData(suhu, kelembaban):
698     url = server_url + "sendData.php"
699     data = {'suhu': suhu, 'kelembaban': kelembaban}
700     req = urllib.urlencode(data)
701     url = server_url + req
702     response = urllib.urlopen(url)
703     data = response.read()
704     print "Data: %s" % data
705
706 #!/usr/bin/python
707
708 def sendData(suhu, kelembaban):
709     url = server_url + "sendData.php"
710     data = {'suhu': suhu, 'kelembaban': kelembaban}
711     req = urllib.urlencode(data)
712     url = server_url + req
713     response = urllib.urlopen(url)
714     data = response.read()
715     print "Data: %s" % data
716
717 #!/usr/bin/python
718
719 def sendData(suhu, kelembaban):
720     url = server_url + "sendData.php"
721     data = {'suhu': suhu, 'kelembaban': kelembaban}
722     req = urllib.urlencode(data)
723     url = server_url + req
724     response = urllib.urlopen(url)
725     data = response.read()
726     print "Data: %s" % data
727
728 #!/usr/bin/python
729
730 def sendData(suhu, kelembaban):
731     url = server_url + "sendData.php"
732     data = {'suhu': suhu, 'kelembaban': kelembaban}
733     req = urllib.urlencode(data)
734     url = server_url + req
735     response = urllib.urlopen(url)
736     data = response.read()
737     print "Data: %s" % data
738
739 #!/usr/bin/python
740
741 def sendData(suhu, kelembaban):
742     url = server_url + "sendData.php"
743     data = {'suhu': suhu, 'kelembaban': kelembaban}
744     req = urllib.urlencode(data)
745     url = server_url + req
746     response = urllib.urlopen(url)
747     data = response.read()
748     print "Data: %s" % data
749
750 #!/usr/bin/python
751
752 def sendData(suhu, kelembaban):
753     url = server_url + "sendData.php"
754     data = {'suhu': suhu, 'kelembaban': kelembaban}
755     req = urllib.urlencode(data)
756     url = server_url + req
757     response = urllib.urlopen(url)
758     data = response.read()
759     print "Data: %s" % data
760
761 #!/usr/bin/python
762
763 def sendData(suhu, kelembaban):
764     url = server_url + "sendData.php"
765     data = {'suhu': suhu, 'kelembaban': kelembaban}
766     req = urllib.urlencode(data)
767     url = server_url + req
768     response = urllib.urlopen(url)
769     data = response.read()
770     print "Data: %s" % data
771
772 #!/usr/bin/python
773
774 def sendData(suhu, kelembaban):
775     url = server_url + "sendData.php"
776     data = {'suhu': suhu, 'kelembaban': kelembaban}
777     req = urllib.urlencode(data)
778     url = server_url + req
779     response = urllib.urlopen(url)
780     data = response.read()
781     print "Data: %s" % data
782
783 #!/usr/bin/python
784
785 def sendData(suhu, kelembaban):
786     url = server_url + "sendData.php"
787     data = {'suhu': suhu, 'kelembaban': kelembaban}
788     req = urllib.urlencode(data)
789     url = server_url + req
790     response = urllib.urlopen(url)
791     data = response.read()
792     print "Data: %s" % data
793
794 #!/usr/bin/python
795
796 def sendData(suhu, kelembaban):
797     url = server_url + "sendData.php"
798     data = {'suhu': suhu, 'kelembaban': kelembaban}
799     req = urllib.urlencode(data)
800     url = server_url + req
801     response = urllib.urlopen(url)
802     data = response.read()
803     print "Data: %s" % data
804
805 #!/usr/bin/python
806
807 def sendData(suhu, kelembaban):
808     url = server_url + "sendData.php"
809     data = {'suhu': suhu, 'kelembaban': kelembaban}
810     req = urllib.urlencode(data)
811     url = server_url + req
812     response = urllib.urlopen(url)
813     data = response.read()
814     print "Data: %s" % data
815
816 #!/usr/bin/python
817
818 def sendData(suhu, kelembaban):
819     url = server_url + "sendData.php"
820     data = {'suhu': suhu, 'kelembaban': kelembaban}
821     req = urllib.urlencode(data)
822     url = server_url + req
823     response = urllib.urlopen(url)
824     data = response.read()
825     print "Data: %s" % data
826
827 #!/usr/bin/python
828
829 def sendData(suhu, kelembaban):
830     url = server_url + "sendData.php"
831     data = {'suhu': suhu, 'kelembaban': kelembaban}
832     req = urllib.urlencode(data)
833     url = server_url + req
834     response = urllib.urlopen(url)
835     data = response.read()
836     print "Data: %s" % data
837
838 #!/usr/bin/python
839
840 def sendData(suhu, kelembaban):
841     url = server_url + "sendData.php"
842     data = {'suhu': suhu, 'kelembaban': kelembaban}
843     req = urllib.urlencode(data)
844     url = server_url + req
845     response = urllib.urlopen(url)
846     data = response.read()
847     print "Data: %s" % data
848
849 #!/usr/bin/python
850
851 def sendData(suhu, kelembaban):
852     url = server_url + "sendData.php"
853     data = {'suhu': suhu, 'kelembaban': kelembaban}
854     req = urllib.urlencode(data)
855     url = server_url + req
856     response = urllib.urlopen(url)
857     data = response.read()
858     print "Data: %s" % data
859
860 #!/usr/bin/python
861
862 def sendData(suhu, kelembaban):
863     url = server_url + "sendData.php"
864     data = {'suhu': suhu, 'kelembaban': kelembaban}
865     req = urllib.urlencode(data)
866     url = server_url + req
867     response = urllib.urlopen(url)
868     data = response.read()
869     print "Data: %s" % data
870
871 #!/usr/bin/python
872
873 def sendData(suhu, kelembaban):
874     url = server_url + "sendData.php"
875     data = {'suhu': suhu, 'kelembaban': kelembaban}
876     req = urllib.urlencode(data)
877     url = server_url + req
878     response = urllib.urlopen(url)
879     data = response.read()
880     print "Data: %s" % data
881
882 #!/usr/bin/python
883
884 def sendData(suhu, kelembaban):
885     url = server_url + "sendData.php"
886     data = {'suhu': suhu, 'kelembaban': kelembaban}
887     req = urllib.urlencode(data)
888     url = server_url + req
889     response = urllib.urlopen(url)
890     data = response.read()
891     print "Data: %s" % data
892
893 #!/usr/bin/python
894
895 def sendData(suhu, kelembaban):
896     url = server_url + "sendData.php"
897     data = {'suhu': suhu, 'kelembaban': kelembaban}
898     req = urllib.urlencode(data)
899     url = server_url + req
900     response = urllib.urlopen(url)
901     data = response.read()
902     print "Data: %s" % data
903
904 #!/usr/bin/python
905
906 def sendData(suhu, kelembaban):
907     url = server_url + "sendData.php"
908     data = {'suhu': suhu, 'kelembaban': kelembaban}
909     req = urllib.urlencode(data)
910     url = server_url + req
911     response = urllib.urlopen(url)
912     data = response.read()
913     print "Data: %s" % data
914
915 #!/usr/bin/python
916
917 def sendData(suhu, kelembaban):
918     url = server_url + "sendData.php"
919     data = {'suhu': suhu, 'kelembaban': kelembaban}
920     req = urllib.urlencode(data)
921     url = server_url +
```

run as module to convert the result taken by the temperature and humidity sensor DHT – 22. And the next step the sensor need to be initialized by Adafruit code. Next process read the data from sensor DHT 22 and send it to server by using php file.

4.4.4. Implementation

The Power monitoring system used a current sensor and temperature sensor. The current sensor works with capturing the electro magnetic fields exist in the power cable. This electro magnetic field will inducted to ferrit coil inside the sensor. Afterwards the fluks magnet exist inside the sensor will be convert to current that can be measrude by coil. With little help from resistor components the sensor will give the voltage value linearly with the current measured. The schematic diagram at current sensor, arduino and microcontroller is as follow. The microcontroller change the data from analogue to digital.

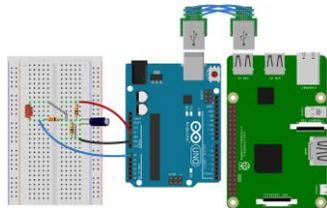


Figure 12. Schematic Diagram

Alternating Current is measured by sensor with maximum range up to 30 Ampere. 30 Ampere is the RMS value from maximum value obtained by sensor. We could calculate th I max with following formula :

$$i(\max) = \sqrt{2} * i(\text{rms_current}) = 1.414 * 30\text{A} = 42.42 \text{ A}$$

I_{\max} current sensor = 42.42 A.

$$i(\text{sensor}) = i(\text{measured}) / \text{nb_turns} = 42.42 \text{ A} / 2000 = 0.02121\text{A}$$

$$\text{calibration_value} = (i(\text{measured}) / i(\text{sensor})) / R(\text{burden})$$

$$\text{calibration_value} = (42.42 \text{ A} / 0.02121\text{A}) / 33\Omega$$

$$\text{calibration_value} = 2000/33\Omega = 60.6$$

From formula we have obtained calibration value to get a normalization value is 60,6. This value used in arduino microcontroller to normalize the current value.

Data sent by arduino will be taken by raspberry pi with serial mode data transfer. And the raspberry will be sent to database server by using wireless connection.

In temperature and humidity measurement using DHT-22 sensor will be more easier because data obtained from the sensore is already

in digital form and without any normalization or calibration applied in this type of sensor.

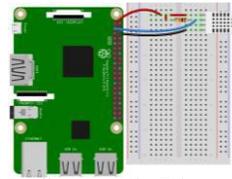


Figure 13. Schematic Diagram of DHT-22

Following is the database design and the database will be automatically will be write by raspberry pi trough php code in order to insert the table.

| + Options | | | | | | | |
|--------------------------|----|------------|------|------|------------|------------|----------|
| | id | server | arus | suhu | kelembaban | tanggal | waktu |
| <input type="checkbox"/> | 1 | 172.27.1.1 | 60.6 | 67.1 | 78.7 | 2016-08-17 | 07:00:00 |
| <input type="checkbox"/> | 2 | 172.27.1.2 | 76.1 | 90.6 | 77.2 | 2016-08-09 | 12:00:00 |
| <input type="checkbox"/> | 3 | 172.27.1.3 | 20.2 | 87.7 | 65.2 | 2016-08-10 | 04:00:00 |
| <input type="checkbox"/> | 4 | 172.27.1.4 | 45.1 | 66.7 | 34.2 | 2016-08-17 | 10:00:00 |
| <input type="checkbox"/> | 2 | 172.27.1.2 | 23.2 | 89.8 | 56.8 | 2016-10-21 | 10:00:00 |

Figure 14. Info Table

Infor Table consist of the condition of all rack server inside data center with information of current, temperature and humidity.



Figure 15. Main Page

The screen at main page use to monitorize the condition of the four server in data center Pustipanda. This condition tells us information about temperature and humidity inside the server room. This information also can be retrieved based on specific date or time or within a certain range of time as beautiful graphic on computer screen.

Historical view/mode tells information about temperature, humidity and current for specific previous date or time. Information system of power monitoring will be implemented by public network. Th topology from network implementation is used based on Local Area Network.

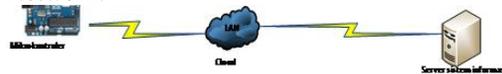


Figure 17. Network Implementation

Figure 17 explained the implementation pllaning will be use a Local Area Network to communicate with a server.

4.4.5. Black Box Testing

At this phase will be conduct a testing for application functionality. This kind of testing is

done to make sure that programm could be running well without any disturbance when it operating. For every test that is running there possibility erorr ocured however by doing kind of testing erorr will be minimized.

Table 8 Black Box Testing

| No | Link | Output | Result |
|----|---|-----------------------------------|---------|
| 1 | Send Data From Microcontroller to Server | Data is sent | Success |
| 2 | Showing a recent condition of rack server with updated data in graphically user interface | New Information in graphical form | Success |
| 3 | Showing Historical data in table form | Historical server in table form | Success |

5. Result

5.1. Monitoring System

The author has developed a system for monitoring the use of current, temperatur and humidity. The author also used the technology so that the security system is becoming increasingly effective and efficient. In its development the author using various technologies, such as Embedded systems, Database Technology and Web.

Authors use ArduinoUno and raspberry pi equipped with wireless as a tool or the brain of the system developed. This tool is in charge to process all the necessary activities in monitoring the flow of electricity. ArduinoUno requires a current sensor for monitoring against the electric current and the data captured by the sensor current is sent through a cloud service. In the manufacture of the system can be accessed online. Power source micro USB with 5V 2.0 Amps needed to provide power at ArduinoUno so it can be turned on and the monitoring of the well as required. Furthermore the data generated by the ArduinoUno developed into the application monitoring system.

5.2. Data Storage

Data is stored into a MySQL-based database of current sensor and processed using the Arduino. Arduino send data to Raspberry Pi using serial data and the Raspberry pi will send tha data and store the data on database server.

REFERENCE

[1] Qiang Liu, Yujun Ma, MusaedAlhousseinc, Limei Peng, Yin Zhang.Green data center with IoT sensing and cloud-assisted smart temperature controlling systembu IEEE access <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7317502>

[2] Xiang Sua, HaoZhangb, JukkaRiekkia, Ari Keranen, Jukka K. Nurminend, Libin Du. Connecting IoT Sensors to Knowledge-Based

Systems by Transforming SenML to RDFby Procedia <http://ubicomp oulu.fi/files/pcs14.pdf>

[3] Martin Fiedler and Stefan Meissner, IoT in Practice: Examples: IoT in Logistics and Health. By Royal institute of technology <http://kth.diva-portal.org/smash/get/diva2:621384/FULLTEXT01.pdf>

[4] ITU-T. ITU Internet Reports The Internet of Things by ITU <https://www.itu.int/net/wsis/tunis/newsroom/stats/The-Internet-of-Things-2005.pdf>

[5] Pressman, Roger S. 2010.software Engineering A Practitioner's Approach seventh Edition. New York : Mc Graw Hill higher Education.

[6] Holt, Huang. 2014. Embedded Operating Systems Practical Approach, Springer, London. UK.

[7] Blum, 2012. Exploring Arduino Tools and Technique For Engineering Wizardy, Wiley, Indiana Polis. USA.

[8] Banzi, Massimo. 2011. Getting Started with Arduino. California : O'Reilly Media,inc

[9] Monk, Simon. 2013. Programming the Raspberry Pi Getting Started with Python. New York : The McGraw-Hill Companies.

[10] Current transformer YHDC SCT-013 <https://openenergymonitor.org/emon/buildingblocks/ct-sensors-interface>

[11] Temperature and humidity sensor DHT22 :<https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>

[12] Pfister, Cuno . 2015. Getting Started with the Internet of Things What Is. USA :O'Reilly Media,inc.

[13] Karen Rose, Scott Eldridge, Lyman Chapin. 2015. The Internet of Things: An Overview Understanding the Issues and Challenges of a More Connected World. The Internet Society (ISOC).

[14] Schwaber,Ken and Sutherland, Jeff. 2013. The Definitive Guide to Scrum: The Rules of the Game .The Scrum Guide™.

[15] Kenneth S. Rubin. 2012.Essential Scrum A Practical Guide To The Most Popular Agile Process. Boston : Addison-Wesley.

[16] Goldstein, Ilan. 2013. Scrum Shortcuts without Cutting Corners Agile Tactics, Tools, & Tips. Boston : Addison-Wesley - Pearson Education, Inc.

[17] Richardson, Matt. Wallace, Shawn. 2012. Getting Started with Raspberry Pi. USA : O'REILLY

[18] The Magpi. THE Official RASPBERRY PI PROJECTS BOOK by Magpi. London : Liz Upton

[19] Vermesan, Ovidium. Friess, Peter. Internet of Things Applications - From Research and Innovation to Market Deployment. Denmark : River Publisher.