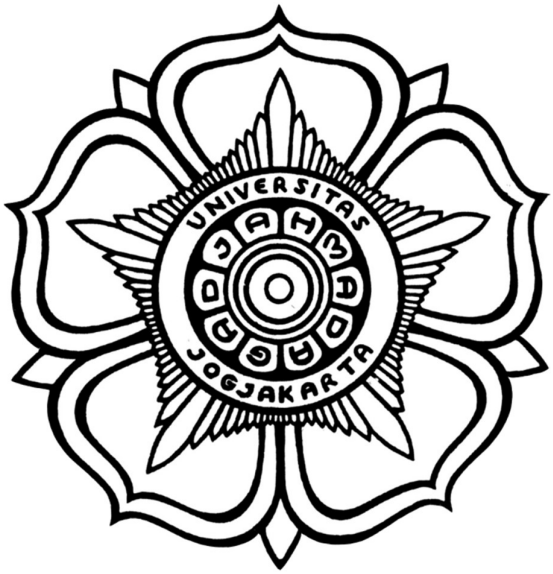
PERANCANGAN KOMPONEN PENUNJANG SISTEM PENGENDALIAN DAN PEMONITORAN TEMPERATUR DAN KELEMBABAN UDARA PADA GUDANG PENYIMPANAN BERBASIS IOT

SKRIPSI



Disusun oleh:

CORNELIUS STEVEN SANJAYA
12 / 333590 / TK / 39938

PROGRAM STUDI TEKNIK ELEKTRO

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA

YOGYAKARTA

2016

HALAMAN PENGESAHAN

PERANCANGAN KOMPONEN PENUNJANG SISTEM PENGENDALIAN DAN PEMONITORAN TEMPERATUR DAN KELEMBABAN UDARA PADA GUDANG PENYIMPANAN BERBASIS IOT

SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh

Gelar Sarjana Teknik Program S-1

Pada Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik

Universitas Gadjah Mada

Disusun oleh:

CORNELIUS STEVEN SANJAYA
12 / 333590 / TK / 39938

Telah disetujui dan disahkan

pada tanggal 27 Juli 2016

Dosen Pembimbing I	Dosen Pembimbing II
I Wayan Mustika, S.T., M.Eng., Ph.D.	**Iswandi, S.T., M.Eng.**
NIP: 1981 0921 2014 04 1 001	**NIP: 1976 0415 2002 12 10 04**

HALAMAN PERSEMBAHAN

JADILAH TERANG DAN GARAM DI TEMPAT YANG MEMBUTUHKAN.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan, atas Berkat-Nya lah skripsi yang berjudul "Perancangan Komponen Penunjang Sistem Pengendalian dan Pemonitoran Temperatur dan Kelembaban Udara pada Gudang Penyimpanan Berbasis IOT" dapat diselesaikan dengan baik. Skripsi ini merupakan syarat wajib untuk menyelesaikan studi S1 di Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada.

Dalam menyelesaikan tugas akhir ini, penulis mendapatkan banyak sekali bantuan dan dukungan dari berbagai pihak. Maka dari itu, penulis ingin mengucapkan terima kasih kepada:

1. Bapak Suharyanto, Dr.Eng., S.T., M.Eng. selaku Ketua Jurusan Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada.
2. Bapak Dr. I Wayan Mustika, S.T., M.Eng., selaku dosen pembimbing pertama yang telah memberikan motivasi, dukungan, bimbingan, kritik, dan saran kepada penulis dalam menyelesaikan tugas akhir ini.
3. Bapak Iswandi, S.T., M.Eng., selaku dosen pembimbing kedua yang telah memberikan arahan, ide, kritik, dan saran kepada penulis dalam menyelesaikan tugas akhir ini.
4. Kedua orang tua tercinta, Budijanto dan Sherly Juliani. Serta Adik, Theresia Sharron Sanjaya. Terimakasih untuk dukungan dan doanya yang tiada henti.

5. Seluruh dosen dan karyawan DTETI UGM. Terima kasih kepada dosen untuk seluruh ilmu yang telah diberikan. Terima kasih pula untuk karyawan yang telah membantu mahasiswa baik di bidang akademik maupun non-akademik.

6. Wieka Jurika Preseilla yang telah memberikan semangat, doa, dan dukungan. Terima kasih atas kesabaran dalam menemani penulis hingga tugas akhir ini berhasil diselesaikan.

7. Seluruh teman – teman grup *Smart* System di Laboratorium Sistem Elektronis. Terima kasih telah banyak membantu dan menemani perjuangan tugas akhir ini.

8. Teman-teman Teknik Elektro dan Teknologi Informasi Fakultas Teknik UGM angkatan 2012. Terima kasih telah menemani dan memberikan momen-momen yang indah bagi penulis.

9. Teman-teman StudentXCEOs Yogyakarta Chapter Batch 3 yang telah memberikan pengalaman berkesan pada masa perkuliahan penulis.

10. Kepada Reinardus Larry, Johanes Handjono, Adrianus Rio, Lukas, dan Prayoga Dharma serta teman-teman KKN JTM-08 yang telah menjadi pengalaman pertemanan yang memberikan banyak makna selama penulis menjalani perkuliahan.

11. Semua pihak yang tidak bisa disebutkan satu per satu, yang telah membantu penulis.

Yogyakarta, 27 Juli 2016

Penulis

DAFTAR ISI

HALAMAN PENGESAHAN ii

HALAMAN PERSEMBAHAN iii

KATA PENGANTAR ii

DAFTAR ISI iv

DAFTAR TABEL viii

DAFTAR GAMBAR ix

DAFTAR SINGKATAN xiii

Intisari xv

Abstract xvi

BAB I PENDAHULUAN 1

1.1. Latar Belakang 1

1.2. Rumusan Masalah 4

1.3. Batasan Masalah 4

1.4. Tujuan Penelitian 5

1.5. Manfaat Penelitian 6

1.6. Sistematika Penulisan 6

BAB II DASAR TEORI 8

2.1. Tinjauan Pustaka 8

2.2. Landasan Teori 13

2.2.1. Kondisi temperatur dan kelembaban udara hama gudang 13

2.2.2. Internet *of Things* (IoT) 17

2.2.3. Arduino 18

2.2.4. Arduino IDE 20

2.2.5. Sensor DHT22 22

2.2.6. Zigbee 25
2.2.7. 2 *Channel relay* modul 30
2.2.8. XCTU 31
2.2.9. Raspberry Pi 31
2.2.10. Python 35
2.2.11. PuTTY 36
2.2.12. *Web server* 37
2.2.13. *Twitter* API 38
2.2.14. Metode POST 39
BAB III METODE PENELITIAN 40
3.1. Bahan Penelitian 40
3.2. Alat yang Digunakan 40
3.2.1. Perangkat keras 40
3.2.2. Perangkat lunak 41
3.3. Alur Penelitian 41
3.4. Analisis Kebutuhan Sistem 44
3.4.1. Konsep keseluruhan sistem 44
3.4.2. Konsep pengiriman data sensor n*ode* 47
3.4.3. Konsep penerimaan data dari n*ode* dan pengiriman data ke s*erver* pada *gateway* 48
3.4.4. Konsep pengolahan data pada s*erver* 50
3.4.5. Konsep penerimaan data status pada *gateway* dan pengiriman data status ke *node* 53
3.4.6. Sistem peringatan *twitter* 54
3.4.7. Konsep penerimaan data status dan pengolahan aktuasi pada *node* 56
3.4.8. Format data komunikasi 56
3.5. Perancangan Perangkat Keras 58
3.5.1. Perancangan perangkat keras pada *node* 58
3.5.2. Perancangan perangkat keras pada *gateway* 61
3.6. Perancangan Perangkat Lunak *Node* 63

3.6.1. Pemanggilan *library* 64
3.6.2. Inisiasi *mode* pin 65
3.6.3. Pembacaan sensor DHT22 dan persiapan pengiriman data 66
3.6.4. Pengiriman data ke *gateway* 68
3.6.5. Penerimaan status dari *gateway* dan proses aktuasi 69
3.7. Perancangan Perangkat Lunak pada *Gateway* 72
3.7.1. Pemanggilan *library* 73
3.7.2. Penentuan PORT, *URL*, dan *baud rate* 74
3.7.3. Penerimaan dan pemrosesan data dari *node* 75
3.7.4. Pengiriman data ke *server* dan penerimaan status 77
3.7.5. Pengiriman status ke *node* 78
3.7.6. Sistem peringatan *twitter* 78
3.8. Perancangan Perangkat Lunak *Server* 81
3.8.1. Pengambilan data *trigger* 82
3.8.2. Proses pengujian data 84
3.9. Konfigurasi XBee 85
3.10. Persiapan *Database Server* 88
BAB IV HASIL DAN PEMBAHASAN 90
4.1. Persiapan Pengujian Sistem 90
4.2. Pengujian Sistem 92
4.3. Pengujian *Node* 94
4.3.1. Pengujian pengiriman data 94
4.3.2. Penerimaan data status 95
4.3.3. Penerjemahan data status dan aktuasi sistem 97
4.4. Pengujian *Gateway* 99
4.4.1. Pengujian pengiriman data ke *server* 99
4.4.2. Pengujian penerimaan status dari *server* dan sistem peringatan *twitter* 103
4.5. Pengujian *Server* 104

4.5.1. Pengujian pengambilan data batas pengguna 105
4.5.2. Pengujian data 106
4.5.3. Penampilan sederhana tabel dan grafik pada *web* 107
4.6. Pengujian Simulasi 109
4.7. Kelebihan dan Kekurangan Sistem 111
BAB V KESIMPULAN DAN SARAN 112
5.1. Kesimpulan 112
5.2. Saran 113
DAFTAR PUSTAKA 114
LAMPIRAN 117

DAFTAR TABEL

Tabel 2.1 Pengaruh temperatur terhadap lama perkembangbiakan T*ribolium castaneum* dengan kelembaban udara 70% (Singh & Prakash, 2015) . 14

Tabel 2.2 Spesifikasi sensor DHT22 (Adafruit.com, 2016)..... 24

Tabel 2.3 Konsumsi daya DHT22 (Adafruit.com, 2016) 25

Tabel 2.4 Karakeristik elektronis XBee (Product Manual v1.xEx - 802.15.4 Protocol, 2016) 28

Tabel 2.5 Perbandingan Python dengan bahasa pemrograman lain (EasyLearning.Guru, 2016)..... 35

Tabel 4.1 Hasil pengamatan penerimaan status 96

Tabel 4.2 Perbandingan data pada *gateway* dan *server* 101

DAFTAR GAMBAR

Gambar 2.1 Gudang BULOG (beritadaerah.co.id, 2016) 10

Gambar 2.2 Gudang silo (pixabay.com, 2016) 11

Gambar 2.3 Grafik proses *survival rate Rhyzopertha dominica* (Nadia Baldassari, 2005) 15

Gambar 2.4 Grafik jumlah telur yang diletakan oleh masing-masing *Trogoderma granarium* perempuan dewasa (Tanseela Riaz, 2014) 17

Gambar 2.5 Jaringan Internet *of Things*(electronicsdesign.com, 2016) 18

Gambar 2.6 Modul Arduino Uno (Arduino.cc, 2016) 19

Gambar 2.7 Tampilan Arduino IDE 22

Gambar 2.8 Sensor DHT22 (electroschematics.com, 2016) 23

Gambar 2.9 Modul XBee PRO (5hertz.com, 2016) 28

Gambar 2.10 2 *Channel relay* module (artofcircuits.com, 2016) 30

Gambar 2.11 Tampilan XCTU 31

Gambar 2.12 Raspberry Pi 3B (thehackernews.com, 2016) 33

Gambar 2.13 Pin GPIO Raspberry Pi 3B (raspberrypi.org, 2016) 34

Gambar 2.14 Tampilan Raspbian Jessie Lite operating system 34

Gambar 2.15 Tampilan PuTTY 37

Gambar 3.1 Alur penelitian 42

Gambar 3.2 Konsep keseluruhan sistem 45

Gambar 3.3 *Flowchart* sistem pengiriman data sensor *node* 48

Gambar 3.4 *Flowchart* penerimaan data dan pengiriman data 49

Gambar 3.5 *Flowchart* permintaan batas temperatur dan kelembaban udara pengguna 51

Gambar 3.6 Proses pengujian data batas temperatur dan kelembaban udara 52

Gambar 3.7 OAuth *Twitter* 55

Gambar 3.8 Format data status 56

Gambar 3.9 Format data yang digunakan komunikasi *node* ke *gateway* 57

Gambar 3.10 Format *frame* yang digunakan untuk *broadcast* status 57

Gambar 3.11 Perancangan perangkat keras pada *node* 61

Gambar 3.12 Perancangan perangkat keras pada *gateway* 63

Gambar 3.13 *Flowchart* perangkat lunak *node* 64

Gambar 3.14 Pemanggilan *library node* 65

Gambar 3.15 Inisiasi *Mode* Pin 66

Gambar 3.16 Membaca sensor dan persiapan pengiriman data 67

Gambar 3.17 Pengalamatan pengiriman data 68

Gambar 3.18 Fungsi pengiriman pada *node* 69

Gambar 3.19 Pemrosesan data status pada *node* 71

Gambar 3.20 Interval penerimaan status 72

Gambar 3.21 Pengaturan awal penerimaan status 72

Gambar 3.22 *Library gateway* 73

Gambar 3.23 Pengaturan PORT, Url, dan *baud rate* 75

Gambar 3.24 Akses serial XBee 75

Gambar 3.25 Penerimaan dan pemrosesan data pada *gateway* 76

Gambar 3.26 Mengirimkan data ke *server* 77

Gambar 3.27 Menerima status dari *server* 78

Gambar 3.28 Mengirim status ke *node* 78

Gambar 3.29 Penerjemahan alamat *node* 79

Gambar 3.30 Peringatan *twitter gateway* 80

Gambar 3.31 Peringatan setelah 1 jam 81

Gambar 3.32 Program pembaruan batas 82

Gambar 3.33 Program halaman web *input trigger* 83

Gambar 3.34 Pengujian pada *server* 85

Gambar 3.35 Pengaturan *baud rate* dan *data bits* XCTU 86

Gambar 3.36 Pemilihan XBee pada XCTU 87

Gambar 3.37 Pengaturan XBee tipe *end device* 87

Gambar 3.38 Pengaturan XBee tipe *coordinator* 87

Gambar 3.39 Pengaturan PAN ID 88

Gambar 3.40 *Database* smart_ricestock 89

Gambar 3.41 Tabel treshold 89

Gambar 3.42 Tabel smart 89

Gambar 4.1 Tampak luar dan dalam *node* 90

Gambar 4.2 Penampakan luar dan dalam *gateway* 90

Gambar 4.3 Denah tempat pengujian 91

Gambar 4.4 Pengiriman data dari *node* 95

Gambar 4.5 Penerimaan status pada *node* 97

Gambar 4.6 Status AC menyala 98

Gambar 4.7 Status keduanya non-aktif 98

Gambar 4.8 Status *exhaust fan* menyala 98

Gambar 4.9 Status AC dan e*xhaust fan* menyala 99

Gambar 4.10 Pemeriksaan data 100

Gambar 4.11 Proses pengiriman data ke *server* 100

Gambar 4.12 Data *log* pengiriman gagal 102

Gambar 4.13 Data *log* pengiriman berhasil 102

Gambar 4.14 Penerimaan status dari *server* 103

Gambar 4.15 *Tweet* peringatan..... 104

Gambar 4.16 Tampilan Memasukkan Batas 105

Gambar 4.17 Batas pada *database*..... 106

Gambar 4.18 Hasil pengujian data pada *server*..... 107

Gambar 4.19 Tampilan tabel pada web 108

Gambar 4.20 Tampilan grafik web..... 108

Gambar 4.21 Tampilan web memasukkan batas..... 108

Gambar 4.22 Grafik penurunan kelembaban udara..... 109

Gambar 4.23 Grafik perbandingan temperatur..... 110

DAFTAR SINGKATAN

A	
AC	*Air Conditioner*
API	*Application Programming Inte*RF*ace*
AS	Amerika Serikat
C	
CDMA	*Code Division Multiple Access*
G	
GPIO	*General Purpose Input/Output*
H	
HDD	*Hard Disk Drive*
I	
IEEE	*Institure of Electrical and Electronics Engineers*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
L	
LAN	*Local Area Network*
LED	*Light Emitting Diode*
O	
OAuth	*Open Standar for Authorization*
R	
RAM	*Random Access Memory*
RF	*Radio Frequency*

U

UART	*Universal Asynchronus Receiver Transmitter*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*

W

WPAN	*Wireless Personal Area Network*

Intisari

Internet of *Things* (IoT) adalah hubungan objek fisik yang terhubung dengan elektronik, *software,* sensor, dan hubungan jaringan yang menjadikan objek-objek tersebut dapat saling mengumpulkan dan mengirim data. IoT memperbaharui sistem industri pertanian dengan sangat signifikan, konsep pengendalian dan pemonitoran yang disediakan oleh IoT sangat bermanfaat untuk meningkatkan efisiensi dunia pertanian, salah satunya adalah dalam pemonitoran dan pengendalian kondisi temperatur dan kelembaban udara gudang penyimpanan pascapanen khususnya beras. Kehilangan hasil pada tahapan pascapanen disebabkan oleh banyak faktor, tetapi faktor hama adalah yang terutama. Keadaan temperatur dan kelembaban udara yang mendukung perkembang biakan hama akan menyebabkan kerugian yang semakin besar pada hasil pascapanen.

Dengan konsep IoT, *node* yang menggunakan Arduino diletakkan pada gudang penyimpanan pascapanen berfungsi sebagai sensor temperatur dan kelembaban udara dan terhubung dengan AC dan *exhaust fan* melalui *relay* untuk melakukan pengendalian. *Gateway* yang menggunakan Raspberry Pi menerima data dari *node* dan mengirimkan data tersebut ke *server* untuk melakukan pengujian nilai temperatur dan kelembaban udara dengan nilai yang diinginkan oleh pengguna berdasarkan informasi karakteristik hama gudang yang dimiliki. *Gateway* juga memiliki fungsi untuk memberikan peringatan kepada pengguna secara personal ketika terjadi potensi ledakan hama pada gudang. Penelitian yang dihasilkan bahwa komunikasi jaringan IoT dapat membantu mengendalikan dan memonitor temperatur dan kelembaban udara pada gudang penyimpanan pascapanen dengan keberhasilan 99% sesuai dengan keinginan pengguna serta dapat memberikan peringatan ketika terjadi potensi ledakan hama berdasarkan jangka waktu kondisi temperatur dan kelembaban udara.

Kata kunci : Internet of *Things*, temperatur, kelembaban udara, Arduino, Raspberry Pi

Abstract

Internet of things (IoT) is a physical objeng network which connected with electronics, software, sensor, and network that make those objects gather and transfer data. IoT renewed the agricultural industry system significantly, its controlling and monitoring concept is very useful to increase the efficiency of agricultural world, as example is for the monitoring and controlling temperature and relative humidity of the grain storage. Loss of grains in the storage caused by several factors, but the optimal of temperature and relative humidity for pest fecundity affect a greater loss of grains.

With the concept of IoT, node which using Arduino set on the grains storage as a temperature and relative humidity sensor and connected to AC and exhaust fan through relay to control the temperature and relative humidity. Gateway which using Raspberry Pi receive the sensor data from node and send it to server. The data will be tested in the server with the temperature and relative humidity value desired by the user based on the optimal temperature and relative humidity of pest fecundity information. Gateway warn the user personally if there is a possibility of pest fecundity. The research's result show that communication of IoT network could help to control and monitor the temperature and relative humidity of grains storage with 99% reliability according to the user desire and warn if there is a possibility of pest fecundity based on the temperature and relative humidity for a period of time.

Keywords : Internet of things, temperature, relative humidity, Arduino, Raspberry Pi

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dewasa ini, konsep *Internet of Things* (IoT) semakin marak dibahas seiring dengan perkembangan dunia di era informasi ini. *Internet of Things* adalah hubungan objek fisik yang terhubung dengan elektronik, *software*, sensor, dan hubungan jaringan yang menjadikan objek-objek tersebut dapat saling mengumpulkan dan mengirimkan data antar objek tersebut. Konsep IoT ini mewadahi objek-objek tersebut dapat terhubung dan dapat dikontrol oleh *remote control* yang terhubung sehingga diharapkan dapat meningkatkan peluang integrasi yang semakin baik antara dunia fisik dengan dunia komputer agar menghasilkan suatu pekerjaan atau proses menjadi lebih efisien baik dalam aspek energi, akurasi, dan ekonomi.

Perangkat IoT dapat digunakan sebagai alat monitor dan kontrol untuk berbagai macam hal, dan dalam prakteknya sistem ini harus secara umum memenuhi beberapa fungsi antara lain *sensing*, *actuation*, dan *control* (Zafalon, 2013). Ketiga hal tersebut harus terintegrasi untuk mengolah data sehingga mengeluarkan suatu *output* yang prediktif maupun adaptif. Perangkat IoT juga harus terhubung dalam suatu jaringan ataupun internet sehingga dapat diakses dan memberikan informasi secara kontinyu kepada pengguna. Sistem IoT ini juga harus didukung dengan keamanan dan kenyamanan dari sistem tersebut agar data yang

diberikan tidak disalah gunakan oleh personal selain pengguna yang ditujukan. Seiring berjalannya waktu, penerapan sistem IoT ini menjadi sangat luas sehingga diprediksikan menjadi infrastruktur utama yang baru bagi seluruh bidang yang ada seperti agrikultur, ekonomi, transportasi, dan lainnya.

IoT memperbaharui sistem industri pertanian dengan sangat signifikan, konsep pengendalian dan pemonitoran yang disediakan oleh IoT sangat bermanfaat untuk meningkatkan efisiensi dunia pertanian, salah satunya adalah dalam pemonitoran kondisi temperatur dan kelembaban udara gudang penyimpanan pascapanen khususnya beras untuk mencegah ledakan hama. Beras merupakan salah satu padi-padian paling penting di dunia untuk dikonsumsi manusia. Di negara-negara Asia yang penduduknya padat, khususnya Bangladesh, Myanmar, Kamboja, Cina, Indonesia, Korea, Laos, Filipina, Sri Lanka, Thailand dan Vietnam, beras merupakan pangan pokok. Sebanyak 75% masukan kalori harian masyarakat di negara-negara Asia tersebut berasal dari beras. Lebih dari 50% penduduk dunia tergantung pada beras sebagai sumber kalori utama (Haryadi, 2006).

Kehilangan hasil pada tahapan pascapanen dapat diakibatkan oleh banyak faktor, tetapi faktor hama adalah yang utama. Di negara berkembang termasuk di indonesia kerusakan bahan hasil pertanian diperkirakan rata-rata mencapai 25-50 % dari total produksi. Di negara maju, kerusakan yang terjadi berkisar antara 5-15% (Sitinjak,1986). Kondisi gudang penyimpanan pascapanen yang mayoritas saat ini masih konvensional membuat kondisi hama sangat nyaman dengan temperatur dan kelembaban udara yang membiarkan hama tersebut berkembang untuk memakan stok beras yang ada pada gudang penyimpanan. Fenomena tersebut menyebabkan

keadaan stok komoditas pascapanen pada gudang tidak memiliki nilai tambah lagi untuk di distribusikan. Banyaknya kasus serangan hama pada gudang menyebabkan kerugian cukup besar, di Amerika Serikat dengan fasilitas penyimpanan yang *mode*rn, nilai kerugian yang ditimbulkan oleh hama gudang mencapai 5 milyar dolar AS per tahun, terlebih di negara-negara berkembang yang fasilitas penyimpanannya masih sedehana (Agus W. dan Sudarmaji, 2010). Hal tersebut menyadarkan kita bahwa industri pertanian membutuhkan bantuan teknologi agar kualitas dari hasil panen tetap terjaga, sehingga diperlukan pengendalian dan pemonitoran agar ledakan hama pada gudang beras tersebut dapat dideteksi dan diantisipasi.

Oleh karena itu, diharapkan sistem ini dapat memberikan informasi yang menunjukan peringatan dan perlakuan tertentu ketika temperatur dan kelembaban udara mendukung untuk terjadinya ledakan hama pada gudang penyimpanan, sehingga ledakan hama pada gudang penyimpanan pascapanen tersebut dapat dihindari. Untuk memenuhi kebutuhan pemonitoran fenomena tersebut, maka dibutuhkan sensor kelembaban udara dan temperatur yang diletakan di beberapa bagian gudang. Sensor ini memperoleh informasi keadaan *realtime* yang akurat di berbagai titik gudang dan mengirimkan informasi tersebut menggunakan komunikasi nirkabel kepada *gateway*. *Gateway* disini berfungsi sebagai jalur komunikasi informasi antara *node* dengan *server*, setelah *server* menerima informasi, maka *server* menentukan temperatur dan kelembaban udara pada gudang tersebut melewati suatu titik nilai yang telah ditentukan pengguna melalui jaringan internet dan memberikan informasi yang harus dilakukan baik kepada pengguna,

maupun kepada *node* melalui *gateway*. Kumpulan informasi ini nantinya disimpan sebagai sebuah arsip bagi pengelola gudang sebagai bahan untuk mengevaluasi keadaan gudang selama jangka waktu tertentu. Dengan adanya penelitian ini, diharapkan dapat meningkatkan sistem pengendalian dan pemonitoran gudang penyimpanan pascapanen menjadi lebih *mode*rn dan lebih tahan terhadap ledakan hama gudang pascapanen tersebut.

1.2. Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan, disusun rumusan masalah sebagai berikut:

1. Sistem pemonitoran dan pengendalian temperatur dan kelembaban udara gudang pascapanen di Indonesia masih menggunakan sistem manual.
2. Belum ada sistem peringatan dan aktuasi ketika temperatur dan kelembaban udara melebihi batas yang diinginkan pada gudang pascapanen.

1.3. Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Penelitian berfokus pada proses dan sistem pengiriman dan penerimaan data, serta peringatan dan aktuasi yang dilakukan setelah pengiriman dan penerimaan data tersebut.
2. Pengambilan data tidak selama waktu perkembangbiakan hama gudang.

3. Gudang penyimpanan diasumsikan sebagai kotak berbahan dasar akrilik dengan ukuran 25 cm x 25 cm x 16 cm dan ketebalan 3 mm dan AC dengan AC mini *portable.*
4. Penelitian menggunakan acuan batas temperatur dan kelembaban udara khusus hama gudang beras.
5. Keamanan data tidak dibahas pada penelitian ini.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Merancang sensor *node* yang dapat mengirimkan data temperatur dan kelembaban udara serta melakukan aktuasi pada gudang penyimpanan pascapanen.
2. Merancang *gateway* yang dapat mengirim perintah aktuasi ke *node*, mengirimkan data ke *server*, dan melakukan peringatan kepada pengguna untuk melakukan pengendalian dan pemonitoran gudang penyimpanan pascapanen.
3. Merancang *server* yang dapat menerima data dari sistem dan pengguna, untuk menguji temperatur dan kelembaban udara pada gudang penyimpanan pascapanen terhadap keinginan pengguna.
4. Menguji keandalan *node*, *gateway*, dan *server* dalam menjalankan sistem yang dirancang.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah :

1. Memahami konsep penggunaan sensor kelembaban udara dan temperatur dalam pemonitoran suatu keadaan.
2. Memahami konsep komunikasi nirkabel antara *node* sensor dan *gateway*.
3. Memudahkan pemonitoran dan pengendalian gudang penyimpanan pascapanen.
4. Meminimalkan terjadinya ledakan hama gudang pada gudang penyimpanan pascapanen.
5. Menjadi bahan acuan untuk penelitian tentang konsep IoT selanjutnya dalam bidang agrikultur.

1.6. Sistematika Penulisan

Sistematika penulisan pada penelitian ini dibagi menjadi lima bab dengan rincian sebagai berikut.

1. BAB I: PENDAHULUAN

   Pada bab ini dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

2. BAB II: DASAR TEORI

   Pada bab ini dijelaskan tentang penelitian-penelitian dan teori-teori terkait yang digunakan sebagai acuan dan dasar dalam penelitian ini.

3. BAB III: METODE PENELITIAN

   Pada bab ini dijelaskan metode yang digunakan, arsitektur dari sistem yang digunakan dan proses perancangan sistem dalam peneilitian ini.

4. BAB IV: HASIL DAN PEMBAHASAN

   Pada bab ini dijelaskan hasil penelitian serta pembahasannya.

5. BAB V: KESIMPULAN DAN SARAN

   Bab ini berisi kesimpulan akhir penelitian dan saran untuk pengembangan penelitian selanjutnya.

BAB II
DASAR TEORI

2.1. Tinjauan Pustaka

Penyimpanan beras dan bahan pangan lain, merupakan salah satu mata rantai kegiatan pascapanen sebelum komoditas didistribusikan. Kehilangan komoditas berupa menurunnya mutu, bertambahnya kadar air, kotoran benda asing, kerusakan bentuk, warna, bau, rasa, dan kehilangan kualitas berupa penyusutan berat harus diperhatikan selama penyimpanan (Sitinjak, 1986). Perkembangbiakan hama gudang adalah faktor yang kuat dari beberapa faktor lain untuk mempengaruhi penurunan mutu komoditas. Beberapa unsur-unsur iklim mikro yang sangat berpengaruh pada perkembangan hama gudang, yaitu: temperatur, kelembaban, kadar air dan aerasi. Unsur-unsur ini dapat mengembangkan, melumpuhkan, menghambat perkembangbiakan atau memusnahkan populasi hama pascapanen. Temperatur lingkungan dan kadar air bahan simpan merupakan faktor utama yang mempengaruhi masa perkembangan (Setyolaksono, 2013). Masa perkembangan serangga yang hidup pada temperatur tinggi lebih singkat daripada temperatur fluktuatif walaupun dengan rata-rata temperatur yang sama tinggi. Sementara itu pada temperatur rendah, masa perkembangannya lebih lama dibandingkan temperatur fluktuatif dengan rata-rata sama rendah. Kadar air bahan simpan mempengaruhi lama stadium larva. Kadar air bahan simpan yang rendah memperlama stadium larva, tetapi stadium telur dan pupa tidak terpengaruh.

Berdasarkan penelitian Agus W. Anggara dan Sudarmaji, jenis hama gudang yang dominan menyerang gabah meliputi *Ryzhopherta dominica, Sitotroga cerealella, Tribolium castaneum,* dan *T. confusum.* Sementara itu jenis *Sitophilus oryzae, S. zeamais, Trogoderma granarium, Corcyra cephalonica, Plodia interpunctella,* dan *Ephestia elutella* tergolong spesies yang dominan merusak beras (Anggara & Sudarmaji, 2008). Menurut penelitian Adelia Luhjingga Pitaloka yang dilakukan pada Gudang Bulog 103 Kabupaten Demak Sub Dolog Wilayah I Semarang menunjukkan bahwa serangga hama gudang yang ditemukan adalah *Sitophilus oryzae*, *Tribolium castaneum,* dan *Rhyzopertha dominica* dengan kategori tingkat serangan berat (Pitaloka, 2012). Kedua penelitian tersebut menjadi dasar untuk acuan temperatur dan kelembaban udara yang dibutuhkan gudang penyimpanan untuk hama tersebut berada dalam kondisi nyaman untuk berkembang biak. Penulis memilih 5 dari 10 jenis hama gudang yang dominan dalam faktor kerusakan beras pada hama gudang untuk dijadikan informasi acuan temperatur dan kelembaban udara kepada pengguna. Jenis hama gudang yang dipilih yaitu, *Sitophilus oryzae, Tribolium castaneum, Rhyzopertha dominica, Trogoderma granarium,* dan *Corcyra cephalonica.*

Proses pemonitoran dan pengendalian gudang penyimpanan pascapanen yang ada di Indonesia pada saat ini masih menggunakan sistem yang konvensional dengan termometer dan hygrometer. Badan urusan logistik (BULOG) adalah badan yang bertanggung jawab atas kelancaran logistik beras di Indonesia secara nasional, pada gudang penyimpanan pascapanen yang dimiliki oleh BULOG sistem pengendalian dan pemonitoran keadaan gudang masih menggunakan sistem

pengendalian hama gudang terpadu (PHGT). Prinsip (PHGT) merupakan prinsip utama dalam perawatan komoditas di lingkungan Perum BULOG. PHGT mengedepankan kebersihan gudang, kemudian pemonitoran pelaksanaan perawatan komoditas dan gudang, lalu kegiatan preventif (*spraying*) dan kegiatan kuratif pengendalian hama seperti fumigasi apabila terjadi serangan hama. Pada Gudang Bulog 103 Kabupaten Demak Sub Dolog Wilayah I Semarang seperti pada Gambar 2.1, proses perhitungan jumlah hama yang terdapat dalam gudang menggunakan metode sampel dengan sampel diambil secara acak sebanyak 3 kg dengan masing-masing 1 kg dari tiap bagian tumpukan. Analisis data menggunakan metode analisis *univariate* (Pitaloka, 2012).

Gambar 2.1 Gudang BULOG (beritadaerah.co.id, 2016)

Penelitian yang berjudul "*A Real-time Monitoring and Controlling system for Grain Storage with Zigbee Sensor Network* "oleh Huiling Zhou menjelaskan sistem komunikasi antar jaringan sensor Zigbee untuk sistem pemonitoran dan pengendalian pada gudang penyimpanan menggunakan 3 faktor, yaitu temperatur, kelembaban udara, dan tekanan udara (Zhou, Zhang, Liu, & Zhang, 2009). Gudang yang digunakan pada penelitian ini adalah gudang berjenis silo untuk penyimpanan hasil pascapanen seperti yang ada pada Gambar 2.2.

Gambar 2.2 Gudang silo (pixabay.com, 2016)

Dalam penelitian tersebut, telah dirancang sebuah sistem yang berhasil mengkomunikasikan sejumlah 21 *node* sensor untuk mengetahui keadaan di dalam suatu gudang. Sensor temperatur yang digunakan adalah DB18B20, SHT71 sebagai sensor kelembaban udara dan temperatur, dan SDP1000 sebagai sensor tekanan angin. Penelitian ini menguji berapa jarak yang efektif agar komunikasi data antara *coordinator* Zigbee dan *node*. Penelitian ini juga menggunakan *node* yang berbeda untuk pemonitoran gudang dan pengendalian gudang. *Software* yang digunakan untuk pengguna *inte*RF*ace* proses pengendalian dan pemonitoran juga menggunakan Windows CVI. Pada penelitian, sistem belum mendukung untuk terhubung melalui internet, sehingga proses manajemen pengendalian dan pemonitoran pun masih harus secara fisik datang menuju ruang pengendalian dari gudang tersebut.

Li-Hua Wu juga melakukan penelitian yang berjudul "*Design of an Intelegent Granary Monitoring System"* yang berbasis *code division multiple access* (CDMA) pada frekuensi 800 Hz untuk komunikasi pengendalian dan

pemonitorannya. Sistem yang dibangun menggunakan prinsip *host computer* dan *slave computer* dimana *host computer* berfungsi untuk kontrol dari keseluruhan sistem dan *slave computer* berguna untuk meneruskan data dan menerima perintah dari dan ke *host computer*. Komunikasi dengan sistem ini sudah cukup baik untuk melakukan proses pengendalian dan pemonitoran pada gudang penyimpanan pascapanen, namun pada sistem ini keberhasilan sistem sangat ditentukan oleh modul komunikasi CDMA yang terhubung pada *slave computer* karena modul tersebut yang berperan untuk menghubungkan sistem ke jaringan internet (Wu, Yao, Zhang & Lin, 2014).

Penelitian tentang sistem pemonitoran gudang pascapanen dimulai di China pada tahun 2009 dan sudah menggunakan sistem *Wireless Sensor Network*, namun penelitian tersebut menggunakan mikrokontroler sederhana seperti ATmega 128L dan belum terhubung secara *mobile* kepada internet seperti penelitian Jianguang Jia pada tahun 2009 yang berjudul "*Design of Monitor System for Grain Depots Based on* Nirkabel *Sensor Network"* (Jia, Kuang, He & Mu, 2009). Dalam penelitian kali ini disediakan sebuah sistem yang memberikan pengguna kewenangan melakukan proses pengendalian dan pemonitoran keadaan temperatur dan kelembaban gudang pascapanen melalui jaringan internet. Sensor *node* mengirimkan data temperatur dan kelembaban udara gudang penyimpanan pascapanen yang diukur oleh sensor DHT22 yang terhubung dengan Arduino. Data temperatur dan kelembaban udara pada gudang tersebut dikirimkan kepada *gateway* yang menggunakan perangkat Raspberry Pi melalui komunikasi XBee dengan protokol jaringan Zigbee dan diteruskan kepada *server* melalui jaringan Ethernet LAN. Data tersebut

dibandingkan dengan nilai batas temperatur dan kelembaban udara yang diinginkan oleh pengguna yang di-*input* melalui *website* terlebih dahulu. Setelah dibandingkan dengan batas tersebut, *server* dapat menentukan perintah pengendalian yang harus dilakukan oleh *node* yang dikirim melalui *gateway*. *Gateway* juga memiliki peranan penting untuk memberikan peringatan kepada pengguna jika temperatur dan kelembaban udara pada gudang penyimpanan pascapanen melebihi dari batas yang ditentukan. Data temperatur, kelembaban udara, dan status dari *node* tersebut disimpan di *database* dan ditampilkan melalui *website*.

2.2. Landasan Teori

2.2.1. Kondisi temperatur dan kelembaban udara hama gudang

2.2.1.1. Sitophilus oryzae

Sitophilus oryzae adalah kumbang bubuk beras yang tergolong hama primer dan paling dominan menimbulkan kerusakan beras dalam penyimpanan. Menurut penelitian Agus W. Anggara dan Sudarmaji, *Sitophilus oryzae* mengalami metamorphosis sempurna dengan perkembangan dari telur hingga imago selama 35 hari di daerah tropis, dan 110 hari di daerah beriklim dingin. Lingkungan yang paling sesuai bagi perkembangan hama ini adalah pada temperatur 25 – 27º C dan kelembaban udara 70% (Anggara & Sudarmaji, 2008).

2.2.1.2. Tribolium Castaneum

Menurut penelitian Dr. Shweta Singh dan Prof. Sant Prakas, *Tribolius castaneum* memiliki 3 fase perkembangbiakan mulai dari larva, pupa, dan dewasa.

Penelitian tersebut membagi pengaruh temperatur dan kelembaban terhadap masing-masing fase kehidupan dari *Tribolium castaneum.* Tabel 2.1 memperlihatkan pengaruh temperatur terhadap lama perkembangbiakan *Tribolium castaneum* dengan kelembaban udara 70%.

Tabel 2.1 Pengaruh temperatur terhadap lama perkembangbiakan T*ribolium castaneum* dengan kelembaban udara 70% (Singh & Prakash, 2015)

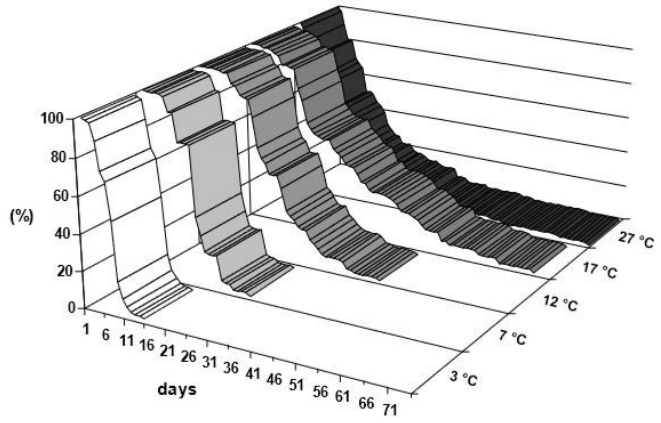
Temperatur / Fase	25º C	30 º C	35 º C
Telur	4 - 5 Hari	3 - 4 Hari	11 -12 Hari
Larva	40 - 41 Hari	27 - 28 Hari	25 - 27 Hari
Pupa	12 -14 Hari	6 -8 Hari	5 - 6 Hari
Imago	2 - 3 Hari	3 - 4 Hari	5 - 6 Hari
Total	63 Hari	44 Hari	51 Hari

Dari kondisi Tabel 2.1, disimpulkan oleh Dr. Shweta Singh bahwa keadaan yang paling nyaman untuk *Tribolium castaneum* berkembang biak berada pada 30º C – 35º C dan dengan kelembaban udara 70%, sehingga dibutuhkan kondisi di bawah dari kedua angka tersebut untuk mengurangi perkembangbiakan hama yang terjadi di gudang penyimpanan pascapanen (Singh & Prakash, 2015).

2.2.1.3. Rhyzopertha dominica

Nadia Baldassari, Antonio Martini, Sandro Cavicchi, dan Piero Baronio melakukan penelitian pada tahun 2005 untuk mengetahui karakteristik perkembangbiakan hama *Rhyzopertha dominica* terhadap temperatur dan kelembaban udara di Italia, dari penelitian tersebut dilakukan pengujian terhadap 5 temperatur dari 3º C, 7º C, 12º C, 17º C, dan 27º C dengan kelembaban udara sebesar 70%.

Sesuai dengan Gambar 2.3., menurut penelitian tersebut, untuk mematikan *Rhyzopertha dominica* secara keseluruhan dibutuhkan penurunan temperatur yang sungguh signifikan sampai pada temperatur 3º C, namun pada temperatur terkendali sebesar 27º C sudah cukup untuk menurunkan jumlah perkembangbiakan *Rhyzopertha dominica* pada gudang penyimpanan pascapanen. Kelembaban udara harus dalam keadaan terkendali dengan tidak mencapai batasnya pada angka 70%. Dengan keadaan temperatur diatas 27º C dan kelembaban udara mencapai 70%, kemungkinan *Rhyzopertha dominica* berkembang biak secara masif hanya membutuhkan waktu 50 hari (Baldassari, Martini, Cavicchi & Baronio, 2005).



Gambar 2.3 Grafik proses *survival rate Rhyzopertha dominica*
(Nadia Baldassari, 2005)

2.2.1.4. Corcyra cephalonica

Corcyra cephalonica atau yang sering disebut dengan *rice moth* merupakan hama gudang yang memiliki peran cukup penting dalam merusak komoditas hasil pascapanen. Penelitian yang dilakukan oleh N. B. Osman, V. F. Wright, dan R. B. Mills di Kansas State University melakukan pengaturan

temperatur pada ruang tertutup dan melihat pengaruhnya terhadap setiap fase pertumbuhan dari *Corcyra cephalonica.*

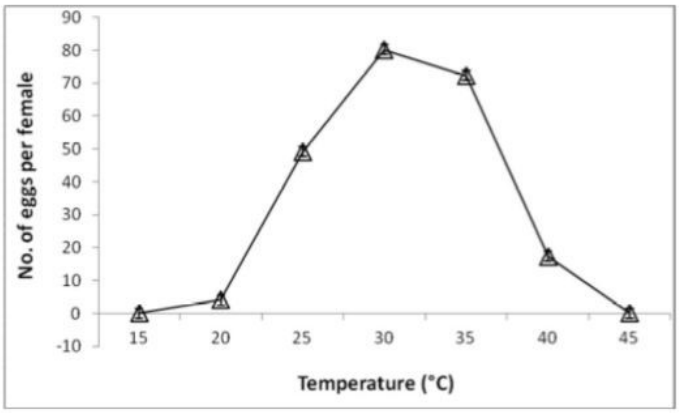
Dari penelitian kali ini, dihasilkan bahwa pada temperatur 32º C populasi dari *Corcyra cephalonica* mencapai 40 kali lebih banyak dibandingkan dengan pada saat temperatur 28º C selama waktu 105 hari percobaan. Setelah 120 hari percobaan, perkembangbiakan *Corcyra cephalonica* memiliki jumlah 30 kali lebih banyak pada saat temperatur 30º C dibandingkan dengan pada saat temperatur 28º C. Penelitian tersebut menyimpulkan bahwa pada temperatur 30º C sampai 32º C dan kelembaban udara 70% merupakan keadaan yang paling optimal untuk proses perkembangbiakan *Corcyra cephalonica* dengan kebutuhan hari selama 27 – 31 hari proses perkembangbiakan dari telur menjadi dewasa (Osman, Wright & Mills, 1983).

2.2.1.5. Trogoderma granarium

Penelitian yang berjudul *"Effect of Temperatur on the Development, Survival, Fecundity and Longevity of Stored Grains Pest, Trogoderma granarium"* oleh Tanseela Riaz, Farah Rauf, dan Syed Shahid menghasilkan sebuah karakteristik khusus dari *Trogoderma granarium* mulai dari setiap fasenya.

Penelitian tersebut menyatakan bahwa tidak ada telur yang diletakan pada beras ketika temperatur ruangan 15 dan 45°C dan nilai jumlah telur maksimum ditemukan pada temperatur 30°C dengan jumlah 80,24±1,31 buah dan terus menurun pada temperatur 20°C, 25°C, 35°C, dan 40°C sesuai dengan Gambar 2.4. *Trogoderma granarium* berkembang biak secara optimal pada temperatur 30 –

35°C dengan kelembaban udara 65%, pada keadaan ini *Trogoderma granarium* hanya membutuhkan waktu selama 38 hari dari telur menjadi dewasa (Riaz, Shakoori & Ali, 2014).



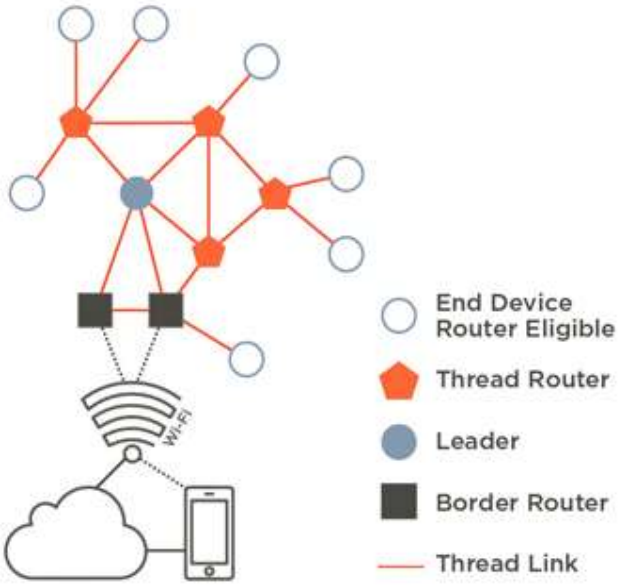
Gambar 2.4 Grafik jumlah telur yang diletakan oleh masing-masing *Trogoderma granarium* perempuan dewasa (Tanseela Riaz, 2014)

2.2.2. Internet *of Things* (IoT)

Internet of Things adalah objek jaringan fisik yang terdiri dari elektronik, *software*, sensor, hubungan jaringan, yang memperbolehkan objek-objek tersebut untuk memperoleh data dan saling memberikan data. *Internet of Things* memperbolehkan objek-objek tersebut untuk dikontrol oleh suatu *remote* yang terhubung oleh infrastruktur jaringan yang sama, sehingga menjadikan adanya integrasi antara dunia fisik dan sistem berbasis komputer untuk meningkatkan keuntungan efisiensi, akurasi, dan ekonomi. Jaringan dari Internet *of Things* dapat diamati pada Gambar 2.5.

Konsep IoT ini mulai marak diperbincangkan sejak penggunaan internet menjadi hal yang umum bagi mayoritas masyarakat. Penggunaan internet dalam jumlah yang masif mengizinkan banyaknya pribadi di dunia yang dapat terhubung

satu sama lain, maupun terhubung dengan perangkat lain yang juga terhubung kepada internet. Konsep IoT ini memberikan batasan yang sangat luas dalam membuat sesuatu yang mempermudah proses bekerja bagi para manusia, seperti halnya menghidupkan lampu melalui *smartphone*, maupun juga mengontrol temperatur dan kelembaban udara dari suatu ruangan dari belahan dunia lain. Semua hal tersebut dapat dilakukan ketika *device* terhubung pada internet.



Gambar 2.5 Jaringan Internet *of Things*(electronicsdesign.com, 2016)

2.2.3. Arduino

Arduino adalah *platform* pembuatan prototipe elektronik yang bersifat *open-source hardware* yang berdasarkan pada perangkat keras dan perangkat lunak yang fleksibel dan mudah digunakan. Arduino pada awalnya dikembangkan di Ivrea, Italia. *Platform* Arduino terdiri dari Arduino *board*, *shield*, bahasa pemrograman Arduino, dan Arduino *development environment* (Arduino.cc, 2016).

Arduino *board* memiliki terdapat lebih dari 20 jenis *board* yang resmi keluaran Arduino. Tiap jenis *board* memiliki karakteristik dan spesifikasi dan harga yang berbeda, sehingga pengguna dapat memilih jenis *board* sesuai kebutuhan. Akan tetapi, meskipun terdapat lebih dari 20 jenis *board*, semuanya dapat diprogram dengan bahasa pemrograman dan IDE yang sama.

Salah satu dari produk Arduino yaitu Arduino Uno Rev 3 (Arduino Uno R3). Arduino Uno R3 adalah mikrokontroler dengan menggunakan ATmega328. Arduino mempunyai 14 pin digital *input* dan *output*, 6 analog *input*, 16 MHz osilator kristal, koneksi USB, *power jack*, *ICSP header*, dan tombol *reset*. Arduino dapat dijalankan hanya dengan menghubungkan Arduino Uno melalu koneksi USB dengan komputer. Bila tidak terhubung dengan komputer, dapat menggunakan catu daya melalui baterai atau sumber DC lainnya. Gambar 2.6 menunjukkan *board* dari Arduino Uno.

Gambar 2.6 Modul Arduino Uno (Arduino.cc, 2016)

Menurut Ritvaldi, Arduino memiliki beberapa keunggulan dibandingkan dengan *platform – platform* mikrokontroler lainnya, yakni:

- Harga yang terjangkau dibandingkan *platform* lain
- Arduino IDE yang kompatibel dengan berbagai sistem operasi yaitu Windows, Mac, dan Linux.
- *Open source* yang dapat digunakan dan dikembangkan oleh semua orang baik peningkatan dari sisi perangkat lunak maupun perangkat keras (Ritvaldi, 2016).

Arduino memiliki fasilitas untuk berkomunikasi dengan mikrokontroler lain dengan tambahan modul lain seperti XBee, dengan metode *serial universal asynchronous receiver/transmitter (UART), inter integrated communication (I2C),* dan *serial peripheral inte*RF*ace (SPI)*. Komunikasi UART dapat menggunakan satu kabel transmisi. Dalam komunikasi UART, saat pengiriman data, *clock* antara pengirim dan penerima harus sama karena paket data dikirim tiap bit mengandalkan *clock* tersebut. Paket data UART dikirimkan bergantung dari nilai *baud rate*. Karena protokol ini bersifat universal, maka *baud rate* yang ada nilai tetap dan tidak berubah. Untuk komunikasi *serial monitor*, *baud rate* yang disediakan 300, 600, 1200, 2400, 14400, 19200, 28800, 38400, 57600, dan 115200. Semakin cepat *clock* yang digunakan sebuah mikrokontroler maka komunikasi semakin cepat juga. Kelemahan UART terletak pada pesat pengiriman dan jarak transmisi. Jika semakin cepat dan jarak semakin jauh maka paket bit dapat terganggu sehingga data yang dikirim akan mengalami galat.

2.2.4. Arduino IDE

Arduino *integrated development environment* (IDE) merupakan sebuah *platform open source* yang disediakan oleh Arduino untuk melakukan penulisan

program yang dimasukan ke dalam *hardware* Arduino yang kemudian akan menjalankan program tersebut. Arduino IDE terdiri dari *text editor* untuk menulis program, *text console*, dan *toolbar* dengan menu dari beberapa fungsi (Arduino.cc). Program yang ditulis dengan Arduino IDE disimpan dengan file yang memiliki format .ino, Arduino- IDE memiliki prosedur penulisan program yang harus selalu diikuti yang terdiri dari minimal 3 bagian, yaitu:

- Inisiasi awal program yang terletak sebelum *Void Setup* (). Inisiasi ini berguna untuk menginisiasi variabel – variabel yang dibutuhkan pada keseluruhan program dan juga sebagai penyertaan *library* yang digunakan dalam keseluruhan eksekusi program.

- *Void Setup* ():

  Void Setup () adalah bagian program yang ingin dijalankan hanya sekali ketika Arduino mulai dinyalakan. Pada bagian ini dituliskan karakteristik dari program seperti *baud rate* dari komunikasi Arduino dengan *serial monitor* ataupun dengan mikrokontroler lain. *Void* juga digunakan untuk inisiasi variabel yang dibutuhkan dan juga permulaan dari penggunaan *library* yang telah diikutsertakan pada awal program.

- *Void Loop* ():

  Void Loop () adalah *void* yang dijalankan secara berulang kali. Dalam *void* ini terdapat logika-logika yang digunakan untuk

menjalankan fungsi Arduino sesuai dengan keinginan dari pengguna dan sesuai dengan kebutuhannya. Dalam kebutuhannya, pengguna diperkenankan untuk membuat *void* lain yang berguna untuk dipanggil pada *void loop* ketika menemui suatu logika tertentu.

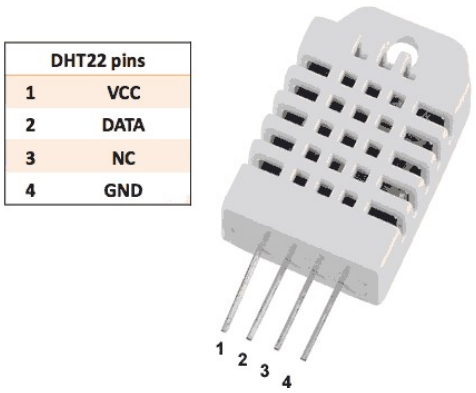
Mengunduhkan program ke dalam Arduino yang kita miliki memiliki beberapa langkah yang harus diikuti seperti menentukan *board* yang sesuai dan *port* dari Arduino yang sesuai. Arduino IDE memiliki karakteristik untuk mendeteksi adanya *error* dalam program ketika tidak memenuhi standar penulisan tertentu. Program yang ditulis dalam Arduino IDE harus mengikuti standar yang telah ditentukan terlebih dahulu untuk dapat diunduh ke dalam Arduino. Gambar 2.7 menunjukkan tampilan dari Arduino IDE.

Gambar 2.7 Tampilan Arduino IDE

2.2.5. Sensor DHT22

Sensor DHT22 adalah sensor temperatur dan kelembaban udara dengan *output* digital yang terkalibrasi. DHT22 mengaplikasikan *exclusive digital-signal-*

collecting-technique dan *humidity sensing technology* untuk mengedepankan realibilitas dan stabilitas sensor. Elemen sensor dari DHT 22 terhubung dengan *8-bit single-chip computer.* Setiap *mode*l dari sensor *mode*l ini telah terkompensasi temperatur dan terkalibrasi dengan akurat, koefisien kalibrasi tersebut disimpan dalam beberapa tipe program pada *OTP memory.* Ketika sensor mendeteksi, sensor tersebut mengambil koefisien tersebut dari memori. Gambar 2.8 menunjukkan tampilan dari sensor DHT22 serta karakter dari setiap pinnya.



Gambar 2.8 Sensor DHT22 (electroschematics.com, 2016)

Sensor DHT22 adalah sensor yang memiliki akurasi lebih tinggi dibandingkan dengan sensor DHT11. Dengan kebutuhan daya yang rendah dan akurasi yang tinggi pada temperatur dan kelembaban udara, sensor DHT22 merupakan sensor yang sesuai untuk segala macam aplikasi pengukuran temperatur dan kelembaban udara dengan tersedianya 4 pin yang memudahkan koneksi dari setiap komponen pentingnya. Tabel 2.3 merupakan spesifikasi dari sensor DHT22.

Tabel 2.2 Spesifikasi sensor DHT22 (Adafruit.com, 2016)

Model	**AM2302**
Power Supply	3.3-5.5V DC
Output Signal	Digital Signal via 1-wire bus
Sensing Element	Polymer Humidity Capacitor
Operating Range	Humidity 0-100% RH; Temperatur -40˚C~80˚C
Accuracy	Humidity ±2%RH (Max ±5%RH); Temperatur ±0.5 ˚C
Resolution Sensitivity	Humidity 0.2%RH; Temperatur 0.1 ˚C
Repeatability	Humidity ±1%RH
Humidity Hysterisis	±0.3%RH
Long-term Stability	±0.5%RH/Year
Interchangeability	Fully Interchangeable

DHT 22 mengemas data temperatur dan kelembaban udara dengan sebuah *data set* sebesar 40-bit dengan ketentuan 16-bit awal merupakan data dari kelembaban udara, 16-bit selanjutnya adalah data dari temperatur, dan 8-bit terakhir adalah *check sum*. Pada 16-bit temperatur, jika biner pertama bernilai satu maka nilai dari temperatur saat itu adalah *negative* atau minus. 16-bit biner tersebut diterjemahkan menjadi desimal sehingga menghasilkan nilai aktual dari keadaan saat itu. Mikrokontroler yang terhubung dengan DHT22 akan mengirimkan sinyal *start* untuk memulai pengukuran temperatur dan kelembaban udara dan DHT22 mengirimkan data 40-bit kepada mikrokontroler yang kemudian diterjemahkan menjadi sebuah nilai desimal. Jika sinyal *start* tidak diberikan kepada DHT22 dari mikrokontroler, maka DHT22 tidak mengirimkan sinyal. Konsumsi daya yang dibutuhkan oleh DHT22 baik saat mengukur maupun keadaan *stand by* dapat diamati pada Tabel 2.3.

Tabel 2.3 Konsumsi daya DHT22 (Adafruit.com, 2016)

Items	Condition	Min	Typical	Max	Unit
Power Supply	DC	3.3	5	6	V
Current Supply	Meassuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting Period			2		Second

2.2.6. Zigbee

Zigbee merupakan protokol komunikasi nirkabel, *Zig* yang diambil dari kata *zig- zag* yang berarti sistem komunikasi ini memiliki sifat jaringan komunikasi yang *zig-zag*. *Zig-zag* pada konteks ini adalah dapat berkomunikasi secara tak menentu dan bersifat seperti lebah yang dapat menyampaikan informasi tempat adanya madu. Zigbee juga memiliki desain dengan konsumsi daya yang rendah dan bekerja untuk jaringan personal tingkat rendah. Perangkat Zigbee biasa digunakan untuk mengendalikan sebuah alat lain maupun sebagai sebuah sensor yang nirkabel. Zigbee memliki karakteristik untuk mampu mengatur jaringan sendiri, maupun mengatur pertukaran data pada jaringan. Kelebihan lain dari Zigbee adalah membutuhkan daya rendah, sehingga bisa digunakan sebagai alat pengatur secara nirkabel yang pemasangan hanya perlu dilakukan sekali. Selain itu Zigbee juga memiliki topologi jaringan "*mesh*" sehingga mampu membentuk jaringan yang lebih luas dan data yang lebih diandalkan. Zigbee adalah spesifikasi untuk *protokol* komunikasi tingkat tinggi yang mengacu pada standart IEEE 802.15.4 yang berhubungan dengan *Wireless Personel Area Networks* (WPANs).

Teknologi dari Zigbee sendiri dimaksudkan untuk penggunaan pengiriman data secara nirkabel yang membutuhkan transmisi data rendah dan juga konsumsi daya rendah, dan juga tidak lebih mahal dibandingkan dengan WPANs lain seperti Bluetooth. Standar Zigbee sendiri lebih banyak diaplikasikan kepada sistem tertanam (*embedded application*) seperti pengendalian industri atau pengendali alat lain secara nirkabel*, data logging*, dan juga *sensor* nirkabel dan lain-lain. Zigbee memilki pengiriman *rate* sekitar 250 Kbps, yang lebih rendah dibandingkan dengan WPANs lain seperti Bluetooth yang mempunyai *transfer rate* dengan 1 Mbps. Sedangkan jarak atau *range* kerja dari Zigbee sendiri sekitar 76 m, yang jaraknya lebih jauh dibandingkan dengan Bluetooth.

Dalam melakukan tugas sebagai alat pengiriman data secara nirkabel, Zigbee memiliki 3 jenis peran yang berbeda dalam menjalankan tugasnya, yaitu (Digi.com, 2016):

- Zigbee *Coordinator*:

  Coordinator pada jaringan Zigbee adalah peran yang memiliki otoritas tertinggi dan hanya terdapat 1 *coordinator* dari setiap jaringan Zigbee. *Coordinator* memiliki fungsi untuk menyimpan informasi tentang jaringan termasuk *security key* dari jaringan tersebut. *Coordinator* juga memiliki tugas untuk mengkondisikan jaringan sesuai dengan keinginan pengguna. Tidak jarang bahwa *coordinator* disebut sebagai *gateway* karena seringkali *coordinator*

berhubungan langsung dengan internet dan pengguna untuk proses pengendalian dan pemonitoran data.

- Zigbee *Router:*

  Router berguna sebagai jembatan penghubung pada jaringan antara *end-devices* dan *coordinator. Router* memegang peranan penting untuk meneruskan informasi kepada *coordinator* begitu juga sebaliknya, *router* sangat berguna jika jarak komunikasi jaringan antara *coordinator* dan *router* kurang terjangkau sehingga tidak ada galat terjadi dalam pengiriman informasi.

- Zigbee *End device:*

  End device adalah *device* yang melakukan *sensing* dan *actuating* dari keseluruhan jaringan sistem. *Device* ini bertindak sebagai point terakhir yang memberikan informasi memperoleh informasi dari *coordinator*, namun *end device* tidak mendapat otoritas sebagai *router* atau *coordinator*.

XBee adalah modul yang digunakan untuk komunikasi nirkabel menggunakan Zigbee Protokol dengan standar 802.15.4 protokol yang sudah mendukung *mesh network*. XBee dapat menghantar data sehingga 30 m (*indoor*) dan 100 m (*outdoor*) manakala XBee Pro dapat menghantar data sehingga 100 m (*indoor*) dan 1500 m (*outdoor*). Modul ini membebani arus sebesar 270 mA saat mengirim data dan 55 mA untuk penerimaan data. XBee memiliki kemampuan

untuk mengirimkan data secara *Unicast* maupun secara *Broadcast*. *Unicast* mengirimkan data pada salah satu alamat XBee tertentu, sedangkan *broadcast* mengirimkan data ke seluruh unit XBee yang memiliki komunikasi pada jaringan yang sama. Gambar 2.9 adalah tampilan dari modul XBee RO, dan Tabel 2.4 merupakan informasi karakteristik elektronis dari XBee.

Gambar 2.9 Modul XBee PRO (5hertz.com, 2016)

Tabel 2.4 Karakeristik elektronis XBee (Product Manual v1.xEx - 802.15.4 Protocol, 2016)

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
V_{IL}	Input Low Voltage	All Digital Inputs	-	-	0.35 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.7 * VCC	-	-	V
V_{OL}	Output Low Voltage	I_{OL}=2mA VCC>=2.7 V		-	0.5	V
V_{OH}	Output High Voltage	I_{OL}=-2mA VCC>=2.7 V	VCC – 0.5	-	-	V
I_{IN}	Input Leakage Current	V_{IN}=VCC or GND, all inputs, per pin	-	0.025	1	uA
I_{OZ}	High Impedance Leakage Current	V_{IN}=VCC or GND, all I/O	-	0.025	1	uA

		High-Z, per pin					
TX	Transmit Current	VCC = 3.3 V	-	45 (XBee)	215,140 (PRO, Int)		mA
RX	Receive Current	VCC = 3.3 V	-	50 (XBee)	55 (PRO)		mA
PWR-DWN	Power-down Current	SM parameter = 1	-	<10			uA

XBee dapat menggunakan data *universal asynchronous receiver-transmitter* (UART) dan menggunakan tegangan 3.3 V. Oleh karena itu, *voltage regulator* untuk 3.3 V diperlukan. XBee memiliki banyak sekali keunggulan dari sisi efisiensi maupun fungsionalnya, berikut ini adalah keunggulan dari penggunaan XBee dalam sistem komunikasi nirkabel:

- Konsumsi energi yang sangat rendah. Dengan 3.6 V 600 mA *Lithium battery* dapat mencatu XBee komunikasi nirkabel dengan jarak 1 mil selama 6 – 12 bulan.
- Mendukung IPv6
- Kecepatan pengiriman data yang bisa mencapai 250 Kbps.
- *Point to multipoint* sampai 60.000 titik (.

XBee memiliki 2 jenis operasi pengiriman data, yaitu:

- *Transparent Operation Mode*:

Operasi ini adalah *setting* operasi *default* dari XBee pada pengiriman data, pada operasi *mode* ini, modul XBee menjadi seperti pengganti *serial line*. Data yang

diterima dibariskan untuk transmisi RF data dan ketika RF data berhasil diterima, data dikirim melalui pin DO.

- *Application Programing Interface* (API) *Operation Mode*:

Mode API adalah operasi pengiriman alternatif yang menggunakan *frame* dalam proses pengiriman datanya. Ketika dalam *mode* API, data yang dikirimkan dan diterima selalu berbentuk suatu *frame* tertentu yang memiliki konten RF data dan *command frame*.

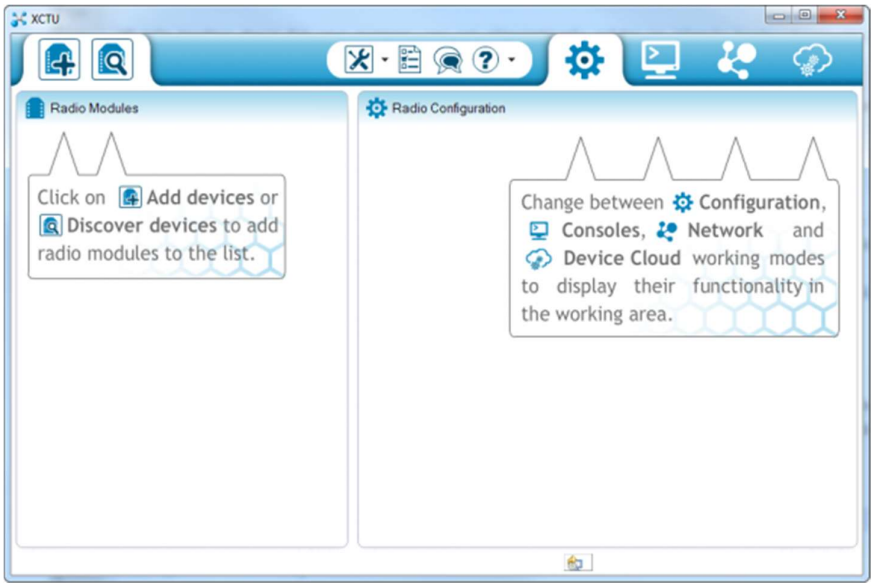
2.2.7. 2 *Channel relay* modul

Kegunaannya sebagai saklar otomatis yang dapat dikendalikan oleh pengguna melalui Arduino- memberikan kemudahan bagi proses pengendalian suatu alat elektronis. *Relay* modul ini dapat menghidupkan dan mematikan sebuah rangkaian yang memiliki voltase hingga 250 VAC dan memiliki arus sebesar 10 A hanya dengan menggunakan. tegangan sebesar 5 V. Setiap *channel* dari modul *relay* memiliki 3 koneksi, yaitu *normally clossed (NC), normally open (NO),* dan COM. Sesuai dengan trigger sinyal *input*, rangkaian akan tersambung ketika logika sinyal *close* pada NO dan sebaliknya. Gambar 2.10 adalah tampilan dari 2 *channel relay modul.*

Gambar 2.10 2 *Channel relay* module
(artofcircuits.com, 2016)

2.2.8. XCTU

XCTU adalah *software* yang telah disediakan oleh Digi sebagai produsen XBee dengan tujuan untuk melakukan konfigurasi dan mengatur peran masing-masing XBee dan melakukan pengujian pada jaringan XBee. XCTU juga memiliki otoritas untuk menentukan XBee sebagai *end device, router,* dan *coordinator.* XCTU menentukan konfigurasi dari setiap XBee sehingga XBee dapat berada di dalam suatu jaringan yang saling terhubung, dengan pengaturan XBee PAN ID maka XBee yang diatur oleh XCTU ditentukan berada pada jaringan komunikasi dengan PAN ID tertentu. Gambar 2.11 merupakan tampilan awal dari XCTU.



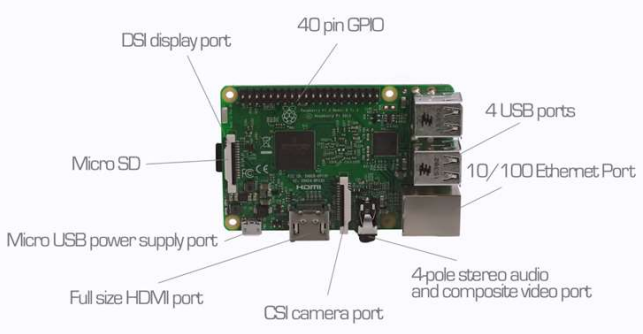
Gambar 2.11 Tampilan XCTU

2.2.9. Raspberry Pi

Raspberry Pi adalah komputer kecil berukuran sebesar kartu kredit yang memiliki harga yang relatif rendah, dan sangat mudah diakses dengan berbagai

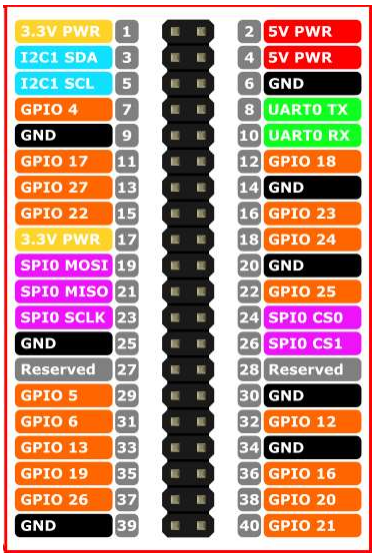
macam bahasa pemrograman seperti Python dan Scratch. Dikembangkan oleh para relawan dan akademisi teknologi di Inggris, Raspberry pi ini memiliki fungsi yang sangat banyak mulai dari berhubungan dengan jaringan internet, terhubung dengan monitor *high definition*, serta dapat mengeluarkan *output* audio dari proses yang diinginkan. Dengan adanya port USB pada Raspberry Pi, perangkat ini diperbolehkan untuk terhubung dengan *mouse* ataupun *keyboard* untuk memudahkan fungsi dari komputer mini tersebut. Gambar 2.12 menunjukan wujud fisik dari Raspberry Pi.

Raspberry Pi terbagi menjadi 2 tipe, yaitu tipe A dan tipe B dan masing-masing dari tipe tersebut yang membedakan adalah RAM dan LAN. Pada tipe A tidak memiliki port LAN, sedangkan pada tipe B memiliki port LAN. Raspberry Pi tipe B memiliki keuntungan untuk aplikasi pada konsep IoT dengan port LAN yang digunakan untuk berhubungan langsung dengan internet. Dengan karakteristik yang dimiliki oleh Raspberry Pi yang dapat terhubung langsung dengan internet melalui LAN, Raspberry Pi tepat untuk dijadikan *gateway* dalam jaringan IoT untuk berhubungan langsung dengan pengguna melalui *Web Server*. Raspberry Pi juga dilengkapi dengan GPIO yang merupakan sederet pin yang terdiri dari 40 pin dengan berbagai fungsi yang dapat diamati pada Gambar 2.13.

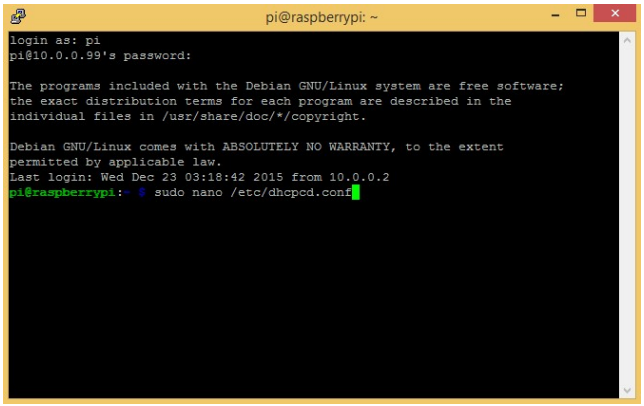


Gambar 2.12 Raspberry Pi 3B (thehackernews.com, 2016)

Selain sebagai *input output* pada beberapa pin GPIO juga berfungsi sebagai komunikasi serial diantaranya I2C, SPI dan serial komunikasi UART. Untuk menggunakan Raspberry pi kita memerlukan *operating system* (contoh OS: Windows, Linux, Mac, dan Unix) yang dijalankan dari SD *card pad board* Rasberrry tidak seperti pada *board microcontroller* AVR yang selama ini dipakai tanpa OS. *Operating system* yang banyak dipakai antara lain Linux distro Raspbian. OS disimpan di SD card dan saat proses *boot* OS hanya bisa dari SD *card*. OS yang bisa dijalankan di Raspberry *board* antara lain: Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Inferno, Raspbian OS, RISC OS, Slackware Linux, dan yang digunakan untuk penelitian ini adalah Jessie. Raspbian Jessie adalah seri terbaru dari sistem operasi Raspbian. Gambar 2.14 menunjukkan tampilan *operating system* Raspbian Jessie lite.



Gambar 2.13 Pin GPIO Raspberry Pi 3B (raspberrypi.org, 2016)



Gambar 2.14 Tampilan Raspbian Jessie Lite *operating system*

Raspbian Jessie dirilis pada tanggal 21 November 2015. Karakteristik yang terdapat pada Raspbian Jessie adalah bawaan aplikasi *office*, Java, dan pengaturan konfigurasi yang lebih lengkap dari seri terdahulu. Selain itu, Raspbian Jessie juga memiliki versi *lite* yang memiliki ukuran data yang lebih kecil.

2.2.10. Python

Python adalah bahasa pemrograman yang bersifat *object-oriented, high level* dengan dinamika bahasa yang *semantic* (Python.org). Bahasa pemrograman Python memiliki tingkat kemudahan yang paling mudah dan paling umum untuk digunakan. Python memiliki keuntungan dalam hal saling berhubungan dengan bahasa pemrograman lain seperti MySQL, PHP, JavaScript, dan lainnya. Python memiliki banyak sekali *library* yang dapat diakses dan di-*install* secara langsung selama proses pemrograman Python terhubung pada koneksi internet. Raspberry Pi adalah perangkat yang sangat lekat hubungannya dengan bahasa pemrograman Python dan dengan saling berhubungnya kedua bahasa tersebut dapat mendukung kegunaaan dari perangkat Raspberry Pi dan bahasa pemrograman Python dalam menjalankan tugas sesuai keinginan pengguna. Tabel 2.5 merupakan tabel perbandingan dari beberapa bahasa pemrograman terhadap bahasa pemrograman Python.

Tabel 2.5 Perbandingan Python dengan bahasa pemrograman lain (EasyLearning.Guru, 2016)

	C	**C++**	**JAVA**	**Python**
Object Oriented	NO	YES	YES	YES
Functional	NO	YES	NO	YES
Safety	UNSAFE	UNSAFE	SAFE	SAFE
Expression	EXPLICIT	EXPLICIT	EXPLICIT	IMPLICIT

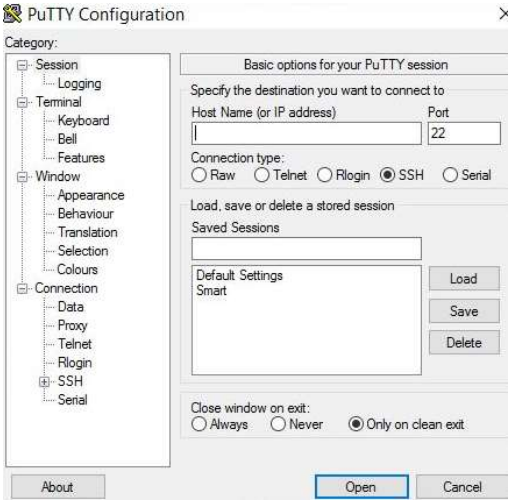
Type Checking	STATIC	STATIC	STATIC	DYNAMIC
Failsafe I/O	NO	NO	YES	YES
Readability	DIFFICULT	DIFFICULT	DIFFICULT	EASY
Learning	DIFFICULT	DIFFICULT	DIFFICULT	EASY
Languange	PROGRAMMING	PROGRAMMING	PROGRAMMING	PROGRAMMING AND SCRIPTING
Length of Code	5-10 GREATER THAN PYTHON	5-10 GREATER THAN PYTHON	3-5 GREATER THAN PYTHON	SMALL AND MANAGEABLE CODES

2.2.11. PuTTY

PuTTY adalah program yang bersifat *free* yang digunakan untuk melakukan SSH dan Telnet untuk *platform* Windows dan Unix yang dibuat oleh Simon Tatham. PuTTY dapat digunakan sebagai *terminal emulator* dan juga sebagai aplikasi pengiriman data pada suatu jaringan. *Platform* ini mendukung beberapa protokol jaringan seperti SCP, SSH, Telnet, rlogin, dan *raw socket connection*.

Platform PuTTY ini seringkali digunakan oleh pengguna komputer untuk melakukan *running test* untuk sebuah program yang telah diunggah kepada suatu *server* dengan melakukan koneksi melalui IP *address* yang dituju dengan memiliki sambungan jaringan yang sama. Penggunaan *port* 22 sebagai *port* akses merupakan *default* dari program PuTTY itu sendiri, id dan *password* dari *server* yang dituju

diminta ketika berhasil melakukan koneksi dengan *server* tersebut yang muncul pada Windows lain. Gambar 2.15 menunjukkan tampilan dari PuTTY.



Gambar 2.15 Tampilan PuTTY

2.2.12. *Web server*

Web server adalah program yang menggunakan HTTP (*Hypertext Transfer Protocol*) yang berfungsi untuk menyediakan sejumlah informasi kepada pengguna melalui *world wide web*. *Web server* memiliki fungsi utama sebagai tempat penyimpanan, pemrosesan, dan penyampaian halaman *web* kepada *client.* Halaman *web* yang ditampilkan kepada *client* selalu dalam bentuk dokumen HTML yang memiliki komposisi gambar maupun *text* sesuai yang ada pada program HTML tersebut. *Client* mengirimkan *request* melalui *web* browser untuk menyediakan informasi yang diinginkan oleh *client* menggunakan HTTP kepada *server* dan *server* tersebut mengirimkan konten dari sumber tersebut atau mengirimkan pesan *error* jika tidak dapat terpenuhi.

Beberapa bahasa pemrograman seperti PHP dan JavaScript didukung oleh *web server* dalam proses pembuatan *website* tersebut. Dengan mendukungnya *platform web server* tersebut, dengan mudah *web service* yang ada di *server* untuk melakukan akses kepada *database* yang berada pada *secondary storage* pada *server*. Dari sistem tersebut, *client* dengan mudah untuk menemukan informasi dan mengakses berbagai macam informasi yang terdapat pada *web server*. *Web server* tidak berguna hanya untuk menyediakan *web service* kepada *client* melalui *server*, tetapi seiring dengan kemajuan teknologi dengan perkembangannya *web server* dapat digunakan sebagai *platform* untuk melakukan pembuatan *embedded system* yang dapat menghubungkan beberapa *device* seperti *printer*, *router*, dan lainnya dengan menggunakan jaringan LAN.

2.2.13. *Twitter* API

Twitter adalah *online social networking service* yang membuat pengguna dapat mengirim atau membaca suatu posting dengan maksimal 140 karakter. *Twitter* menggunakan API *service* sehingga memperbolehkan *web service* lainnya agar terintegrasi dengan *twitter*. *Twitter* s*treaming* API adalah karakteristik yang diberikan oleh *twitter* agar pengguna dapat mengakses *twitter* melalui HTTP *server* lain, yang salah satunya adalah *gateway.*

Dengan menggunakan OAuth, pengguna dapat melakukan akses *twitter* melalui *web server* lain dan melakukan *posting tweet* secara otomatis ketika menemukan suatu kondisi tertentu. Untuk menggunakan karakteristik ini,

diperlukan *API key*, *API secret, consumer key,* dan *consumer secret* yang bersifat seperti id *password* dari aplikasi yang dibuat.

2.2.14. Metode POST

Terdapat 2 cara untuk *client* melakukan pengiriman informasi kepada *web server*, yaitu dengan metode GET, dan POST. Sebelum mengirimkan informasi, informasi tersebut terlebih dahulu di *encode* dengan menggunakan *URL encoding.* Metode POST mengirimkan informasi dengan HTTP headers sehingga metode POST memiliki beberapa keunggulan dibandingkan dengan metode GET, yaitu:

- Metode POST tidak memiliki batasan besar ukuran data untuk dikirimkan
- Metode POST dapat digunakan untuk mengirimkan ASCII dan juga data biner
- Data yang dikirimkan dengan metode POST akan melalui HTTP *header*, sehingga keamanan data berpangku pada HTTP protokol.
- PHP menyediakan $_POST *associative array* untuk mengakses seluruh informasi terkirim.

BAB III
METODE PENELITIAN

3.1. Bahan Penelitian

Bahan penelitian ini didapatkan dari studi literatur mengenai IoT, *gateway*, Arduino, Raspberry Pi, komunikasi data XBee dengan protokol Zigbee, pembacaan data dan *parsing* data menggunakan Python, Keadaan temperatur dan kelembaban udara optimal untuk hama gudang berkembang biak, keadaan gudang penyimpanan pascapanen, dan *Twitter* API.

3.2. Alat yang Digunakan

3.2.1. Perangkat keras

Perangkat keras yang digunakan dalam penelitian ini adalah:

a. Komputer jinjing dengan spesifikasi: Intel core i5, RAM DDR3 4GB, HDD 1TB, Windows 10.
b. Arduino UNO.
c. Sensor Temperatur dan Kelembaban Udara DHT22.
d. Raspberry Pi 3 B.
e. 2 *channel relay.*
f. MicroSD Transcend 8GB.
g. XBee Pro S2B.
h. XBee USB *adapter.*
i. Kipas 12V.

j. AC Mini.

k. Sumber daya 5V, 2A.

3.2.2. Perangkat lunak

Perangkat lunak yang digunakan dalam penelitian ini adalah:

a. Sistem Operasi Windows 10.

b. Sistem Operasi Raspbian Jessie Lite (*Release date 2015-21-11*).

c. Python 2.7.

d. XCTU.

e. PuTTY 0.63.

f. WinSCP.

g. Atom.

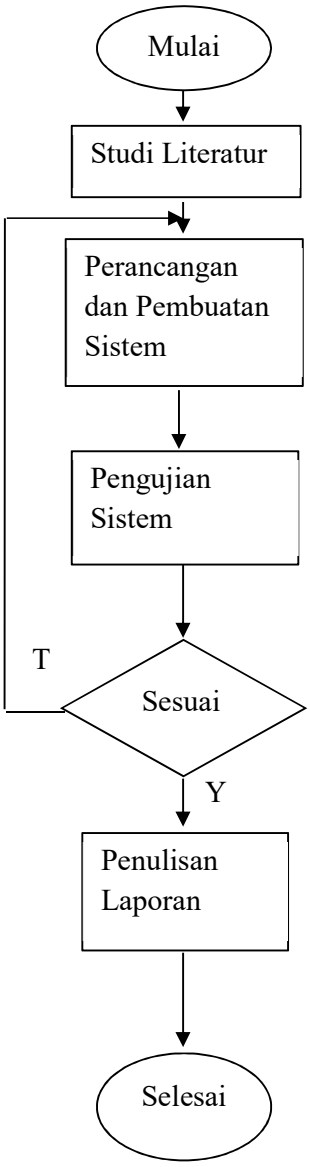
h. Arduino IDE.

i. Google Chrome.

3.3. Alur Penelitian

Alur penelitian yang dilakukan sesuai dengan Gambar 3.1, dimulai dengan studi literatur terlebih dahulu untuk mengetahui hal yang harus dilakukan untuk mencapai keberhasilan perancangan komponen penunjang sistem. Studi literatur dilakukan melalui buku, jurnal, *website*, dan publikasi pada konferensi tertentu. Studi literatur yang dilakukan memiliki jangkauan tema tentang IoT, keadaan temperatur dan kelembaban udara gudang, dan keadaan temperatur dan kelembaban udara optimal untuk hama gudang berkembang biak. Dalam hal teknis, studi literatur dilakukan tentang Arduino, Raspberry Pi, XBee, bahasa pemrograman

Python, dan cara meminta aktuasi dan mengirimkan aktuasi dari *web server* menuju *gateway* dan *node*.



Gambar 3.1 Alur penelitian

Setelah melakukan studi literatur, dilakukan perancangan komponen penunjang sistem yang akan dibuat. Sesuai dengan konsep IoT, perancangan sistem menghasilkan dibutuhkannya *node, gateway,* dan *web server. Node* berguna

sebagai sensor dan pemberian aktuasi pengendalian kondisi di dalam gudang penyimpanan dan *gateway* sebagai penerus informasi kepada pengguna melalui jaringan internet ke *server*. Keseluruhan komponen diharapkan dapat merealisasikan sistem pengendalian dan pemonitoran temperatur dan kelembaban udara pada gudang penyimpanan pascapanen melalui jaringan internet. Setelah melakukan perancangan, dilakukan pembuatan sistem tersebut secara nyata, dengan melakukan penyediaan komponen yang dibutuhkan oleh sistem dilanjutkan dengan merangkai seluruh komponen tersebut sehingga dapat menjalankan sistem sesuai dengan keinginan pengguna. Pembuatan sistem dibagi menjadi 2 bagian, yaitu pembuatan perangkat keras dan perangkat lunak. Perangkat keras menjalankan sistem sesuai dengan perintah dari perangkat lunak yang dibuat oleh pengguna.

Setelah pembuatan komponen penunjang sistem selesai, dilakukan pengujian terhadap komponen tersebut. Pengujian ini bertujuan untuk melakukan *benchmark* antara sistem terhadap keingiinan pengguna dan keadaan sesungguhnya. Ketika pengujian sistem selesai, dilakukan penilaian terhadap sistem tersebut terhadap kriteria yang dibutuhkan. Jika masih belum menemui kriteria yang diinginkan, dilakukan tahap perancangan kembali dan seterusnya sampai menemui titik bahwa sistem siap dan sesuai dengan kebutuhan. Setelah sistem teruji siap dan lengkap, dilakukan penulisan laporan untuk dokumentasi penelitian.

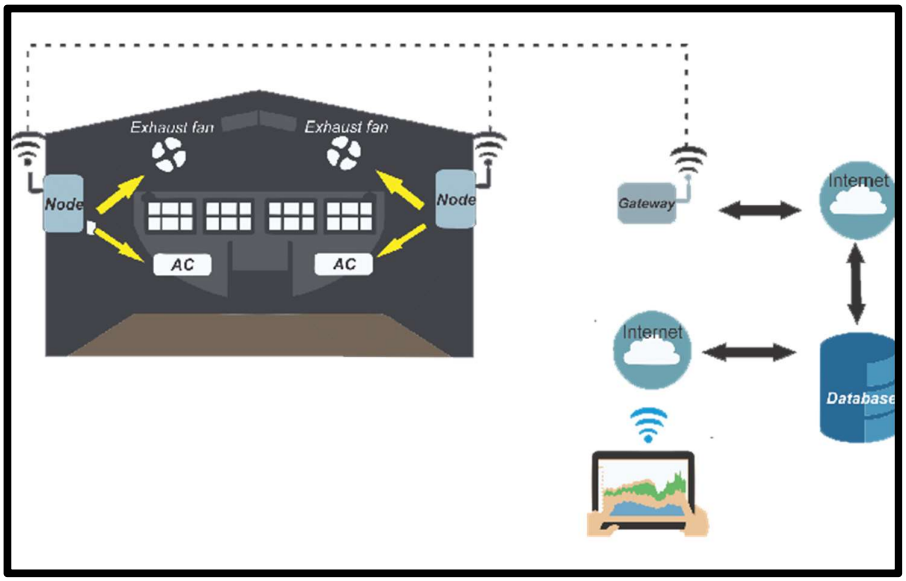
3.4. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan dengan adanya studi literature tentang IoT, dan studi tentang keadaan gudang peyimpanan pasca panen serta temperature dan kelembaban udara optimal untuk hama pasca panen berkembang biak. Analisis kebutuhan sistem memilihi beberapa tahap dalam prosesnya, dimulai dari studi literatur yang sudah dilakukan, lalu mulai membuat konsep seluruh kesatuan bentuk sistem yang dibutuhkan dimulai dari *node, gateway*, dan *web server.* Setelah melakukan analisis kebutuhan sistem melalui studi literatur, dibutuhkan beberapa konsep pada komponen penunjang sistem pengendalian dan pemonitoran gudang, yaitu:

a. Konsep keseluruhan sistem
b. Konsep pengiriman data sensor *node*
c. Konsep penerimaan data dari *node* dan pengiriman data ke *server* pada *gateway*
d. Konsep pengolahan data pada *server*
e. Konsep penerimaan data status pada *gateway* dan pengiriman data status ke *node*
f. Sistem peringatan *twitter*
g. Konsep penerimaan data status dan pengolahan aktuasi pada *node*
h. Format data komunikasi XBee

3.4.1. Konsep keseluruhan sistem

Konsep keseluruhan sistem dibangun berdasarkan konsep IoT, konsep ini dapat memudahkan pemasangan sensor pada gudang penyimpanan pascapanen, dan dapat mempermudah penambahan sensor jika dubutuhkan pada gudang selama masih berada pada jaringan yang sama. Gambaran umum sistem dapat diamati pada Gambar 3.2.



Gambar 3.2 Konsep keseluruhan sistem

Node yang menjadi sensor pada gudang penyimpanan beras terhubung dengan AC dan *exhaust fan* melalui *relay* untuk mengendalikan AC dan *exhaust fan* tersebut. Sensor *node* dilengkapi dengan modul XBee untuk mengirimkan data temperatur dan kelembaban udara gudang penyimpanan pascapanen tersebut ke *gateway*. *Node* tersebut juga melakukan fungsi *receive* secara berulang-ulang setiap 5 detik dalam waktu 5 menit. Proses *receive* tersebut dilakukan secara berkali-kali agar selalu siap ketika terdapat perintah khusus dari pengguna untuk menyalakan AC dan *exhaust fan.* Karakteristik tersebut merupakan karakteristik yang

disediakan pada proses pengembangan sistem saat ini yang berguna untuk pengembangan sistem yang akan datang dengan adanya integrasi secara langsung antara pengguna dan *node* untuk mengendalikan *relay.* Setelah waktu mencapai 5 menit, maka *node* segera melakukan proses *sensing* dan mengirimkan data temperatur dan kelembaban udara pada gudang kepada *gateway*.

XBee yang ada pada sisi *gateway* menerima data temperatur dan kelembaban udara dalam format *frame API* dan di terjemahkan sehingga didapatkan data temperatur, kelembaban udara, dan identitas XBee pengirim. Ketiga data tersebut segera dikirimkan ke sisi *server* beserta waktu saat itu pada *gateway* untuk dilakukan sistem pengujian. Setelah *server* menerima data dari *gateway,* data tersebut akan dimasukkan ke masing-masing variabel. Setelah variabel tersebut terisi, maka nilai-nilai dari variabel tersebut dimasukan ke *database* yang disediakan.

Server melakukan pengujian nilai temperatur dan kelembaban udara yang dikirimkan dengan cara mengambil data tersebut dari *database* ditambah mengambil data batas temperatur dan kelembaban udara yang diinginkan oleh pengguna pada tabel *database* yang berbeda. Setelah data tersebut diuji, maka *server* menentukan status dari *node* yang sudah mengirimkan data temperatur dan kelembaban udara tersebut. Status tersebut selanjutnya dikirimkan ke sisi *gateway*

Gateway segera mengirimkan status tersebut ke *node* dengan menggunakan XBee modul dan melakukan peringatan melalui *twitter* jika temperatur dan kelembaban udara melewati batas. Status yang diterima pada *node* melalui fungsi *receive* pada Arduino bersifat unik sesuai dengan identitas yang dimiliki oleh XBee

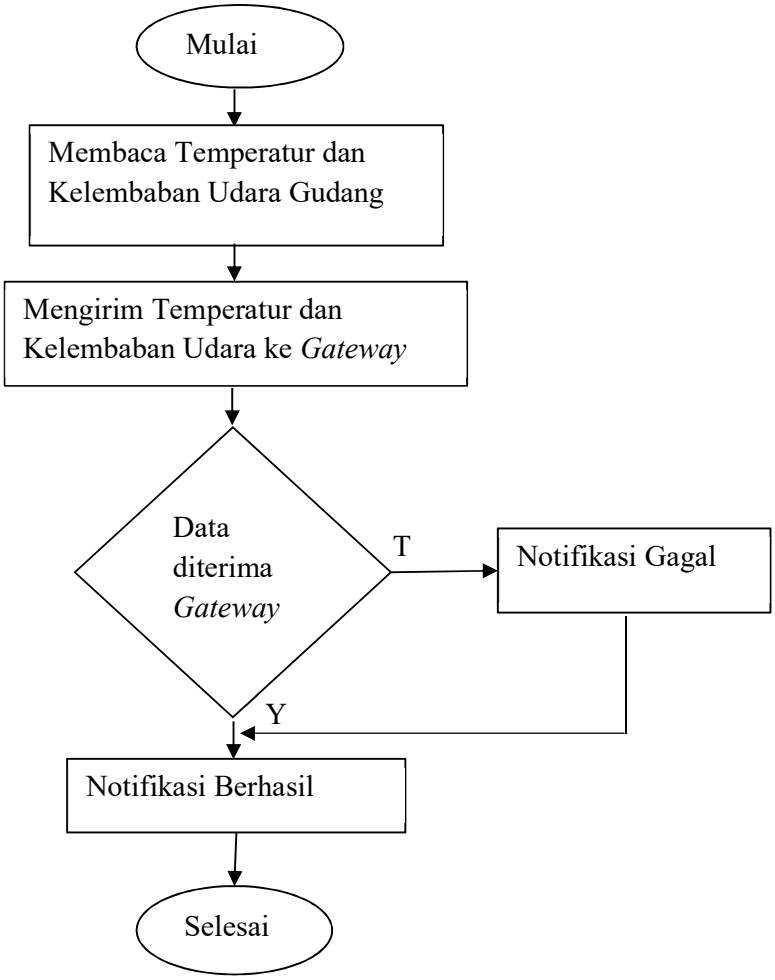
yang terhubung dengan *node.* Jika status yang diterima oleh *node* tidak sesuai dengan identitas XBee yang terhubung, maka sinyal status tersebut tidak diproses secara lanjut.

Topologi *mesh* adalah topologi yang paling proporsional untuk digunakan pada sistem jaringan ini, karena pada jaringan Zigbee terdapat *router* yang dapat meneruskan informasi dari *end device* atau *node* ke *coordinator.* Topologi ini memiliki keunggulan hanya membutuhkan 1 buah *coordinator* pada setiap jaringan yang dimiliki, sehingga lebih mudah dalam melakukan proses pengaturan jaringan tersebut.

3.4.2. Konsep pengiriman data sensor n*ode*

Node yang berada pada gudang membaca temperatur dan kelembaban udara melalui sensor DHT22 dan mengirimkan data tersebut ke *gateway.* Dengan modul XBee, data tersebut dikemas dengan metode *payload.* Metode *payload* digunakan karena komunikasi antar XBee yang digunakan adalah *API mode*, sehingga data harus dikemas dalam bentuk *byte* agar dapat dikirimkan bersama dengan informasi lain dari XBee ke *gateway* dalam 1 *frame* yang sama.

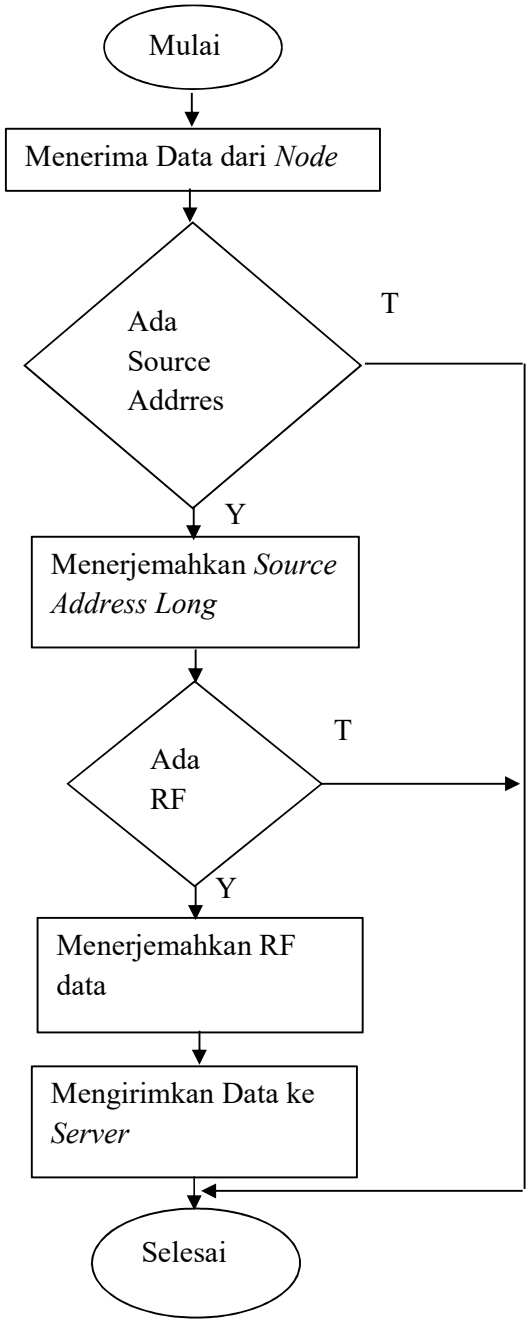
Proses pengiriman data ke *gateway* memberikan notifikasi pada *node* tentang data tersebut diterima dengan baik oleh *gateway* atau tidak. Jika tidak diterima, maka data yang dikirim hilang dan tidak tercatat oleh *gateway*. Hal yang mempengaruhi berhasil atau tidaknya data diterima oleh *gateway* ada pada proses jaringan komunikasi XBee. Diagram alir dari sistem pengiriman data ke *gateway* dapat diamati pada Gambar 3.3.



Gambar 3.3 *Flowchart* sistem pengiriman data sensor *node*

3.4.3. Konsep penerimaan data dari n*ode* dan pengiriman data ke s*erver* pada *gateway*

Frame yang diterima oleh *gateway* tersebut berisi dari bit-bit *hexadecimal* yang diterjemahkan agar data tersebut dapat dikirimkan ke *server* dalam bentuk *string*. Ketika tidak terdapat data yang sesuai pada *frame* yang dikirimkan, perlakuan proses pada *gateway* tidak dilanjutkan, sehingga dilakukan pemeriksaan terlebih dahulu. Diagram alir untuk proses pengiriman data ke *server* dari *gateway* dapat diamati pada Gambar 3.4.



Gambar 3.4 *Flowchart* penerimaan data dan pengiriman data

Setelah pemeriksaan tersebut, maka pengiriman data ke *server* dilakukan. Data yang dikirimkan adalah waktu saat itu pada *gateway,* temperatur, kelembaban udara, dan identitas XBee *node* yang mengirim data. Identitas tersebut berguna

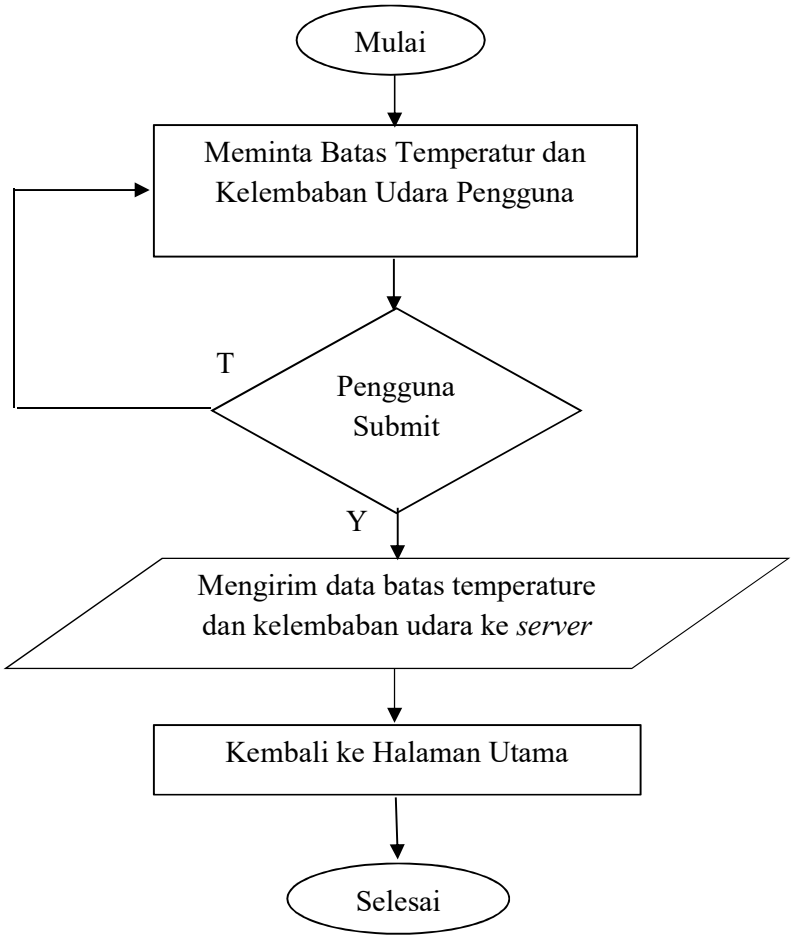
untuk menentukan data status yang diciptakan oleh *server,* sehingga status yang diciptakan bersifat unik untuk *node* yang mengirimkan data tersebut.

Pengiriman data dari *gateway* ke *server* dilakukan dengan menggunakan metode POST, metode POST mengirimkan pesan *request* untuk mengirimkan data ke *URL* tertentu. *URL* yang dituju adalah alamat program PHP *script* yang tersimpan pada *server* untuk diakses. Dalam program tersebut terdapat perintah-perintah yang dibutuhkan dalam memasukan data ke *database* dan melakukan pengujian temperatur dan kelembaban udara yang dibahas lebih lanjut pada sub-bab pengolahan data pada *server*.

3.4.4. Konsep pengolahan data pada s*erver*

Data yang diterima pada *server* berupa sebuah *string* yang dititipkan pada pesan *request.* Data tersebut didefinisikan ke sebuah variabel masing-masing yang berisikan data yang dimaksud, sebagai contoh variabel *Time* pada program PHP *script* diisi dengan variabel waktu yang ada pada paket data *string* dari pesan *request.* Data yang diambil juga dengan menggunakan metode POST sehingga dapat mengambil pecahan data *string* yang diterima. Setelah data berhasil dimasukkan ke variabel masing-masing, variabel tersebut menjadi sebuah data yang dikirimkan ke *database* melalui sebuah pesan *query* sebagai *standard* dari bahasa pemrograman PHP untuk berkomunikasi dengan MySQL yang berfungsi

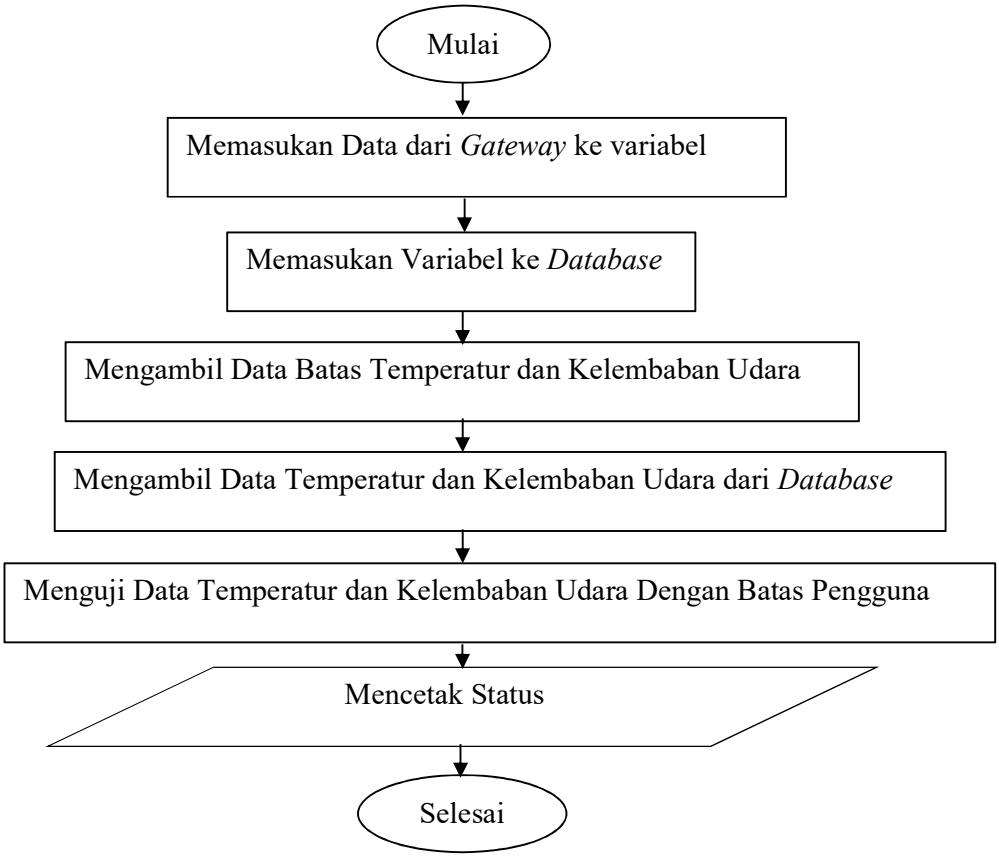
mengakses data ke *database*. Diagram alir permintaan batas temperatur dan kelembaban udara dapat diamati pada Gambar 3.5.



Gambar 3.5 *Flowchart* permintaan batas temperatur dan kelembaban udara pengguna

Terdapat 2 hal yang dilakukan pada *server* untuk melakukan proses pengujian data temperatur dan kelembaban udara, yaitu permintaan data batas temperatur dan kelembaban udara yang diinginkan oleh pengguna dan proses pengambilan data temperatur dan kelembaban udara dari *database* dan pengambilan data batas temperatur dan kelembaban udara yang juga berada pada

database namun tabel yang berbeda untuk selanjutnya diuji. Diagram alir dari proses pengujian data batas temperatur dan kelembaban udara dapat diamati pada Gambar 3.6.



Gambar 3.6 Proses pengujian data batas temperatur dan kelembaban udara

Konsep permintaan batas temperatur dan kelembaban udara digunakan agar pengguna dapat menentukan, batas temperatur dan kelembaban udara yang diinginkan sesuai dengan jenis hama gudang yang ingin diminimalkan perkembangbiakannya. Metode pengambilan data yang digunakan juga menggunakan metode POST, batas temperatur dan kelembaban udara yang dimasukan oleh pengguna hanya akan dimasukan ke *database* ketika tombol *submit*

pada halaman *web* sudah ditekan. Nilai batas temperatur dan kelembaban udara ini disimpan pada *database* sampai pengguna memasukan nilai batas yang baru dan melakukan *submit* kembali.

Untuk melakukan proses awal pengujian, data temperatur dan kelembaban udara diambil dari *database* dengan menggunakan bahasa *query* begitu juga dengan data batas temperatur dan kelembaban udara. Data yang diperoleh dari bahasa *query* tersebut berupa *array* dikarenakan *query* yang berasal dari tabel *database* mengambil keseluruhan data pada tabel tersebut per baris yang dimasukan ke dalam *array*. Seluruh nilai dari data yang dibutuhkan tersebut dimasukan ke sebuah variabel sehingga mempermudah proses pengujian.

Pengujian dilakukan dengan logika data temperatur dan kelembaban udara yang dikirimkan melebihi dari batas temperatur dan kelembaban udara yang diinginkan oleh pengguna yang telah dikirim sebelumnya. Setelah melakukan pengujian tersebut, *server* dapat menentukan status dari *node* yang berisikan informasi identitas XBee pengirim, keadaan AC atau *exhaust fan* dalam keadaan menyala atau tidak. Status tersebut di *update* ke *database* dan di *print* oleh program *script* PHP yang diambil oleh *gateway* sebagai informasi status.

3.4.5. Konsep penerimaan data status pada *gateway* dan pengiriman data status ke *node*

Penerimaan status pada *gateway* merupakan lanjutan dari metode POST yang digunakan. Metode yang menggunakan fungsi *URL*lib pada Python tersebut memiliki fungsi *URLlib2.URLopen* yang berfungsi untuk menangkap hasil proses

URL yang diakses ketika *gateway* mengirimkan pesan *request* untuk meminta *URL* tersebut melakukan proses pengujian data. Data yang berisikan status hasil pengujian *server* tersebut dimasukkan ke variabel status pada *gateway.*

Variabel yang berisikan status tersebut selanjutnya segera dikirimkan kembali ke *node* untuk memberikan informasi aktuasi pada sisi *node.* Informasi ini dikirimkan secara *broadcast* sehingga seluruh *router* ataupun *end device* yang terhubung dalam jaringan yang sama mendapatkan informasi ini. Informasi ini disediakan secara unik sesuai dengan identitas *node* pengirim sehingga hanya diproses pada *node* yang sama.

3.4.6. Sistem peringatan *twitter*

Sistem peringatan *twitter* dilakukan untuk memberikan peringatan langsung kepada pengguna ketika temperatur dan kelembaban udara melebihi batas yang diinginkan. Peringatan menggunakan *twitter* karena *twitter* dapat melakukan *live tweet* yang berfungsi untuk memberikan informasi secara *realtime,* sehingga pengguna dapat mendapatkan informasi tersebut hanya dengan melihat pada *website twitter*. Selain menyediakan informasi secara *realtime, twitter* juga memiliki kemampuan untuk melakukan notifikasi personal, ketika terjadi potensi ledakan hama *twitter* memberikan peringatan secara personal kepada pengguna. Kelebihan yang dimiliki oleh *twitter* adalah dengan dua karakteristik tersebut, karakteristik tersebut dapat dilakukan secara bebas

Sistem peringatan melalui *twitter* ini membutuhkan suatu *library* khusus pada Python agar dapat berfungsi dan terhubung dengan *twitter* dengan API *service* yang

disediakan. Untuk menggunakan *twitter* API *service* ini Python harus terlebih dahulu memasang *library tweepy.* Dengan menggunakan API *service* ini, *twitter* dapat melakukan *tweet* sesuai dengan keinginan pengguna ketika terjadi sesuatu, contohnya sebagai sebuah karakteristik peringatan pada pengguna secara *mobile.*

Selain memasang *library* tweepy pada Python, akses *twitter* dengan API *service* ini juga memiliki keamanan yang tinggi dengan diwajibkannya memasukan *consumer key, consumer secret, access token,* dan *access token secret.* Keempat hal tersebut wajib ada untuk mengakses akun *twitter* tertentu dengan karakteristik OAuth (*open standar for authorization*), dengan *access token, consumer secret,* dan *access token secret* sepanjang 48 karakter hal tersebut membuktikan bahwa akses ke akun *twitter* dengan API *service* ini memiliki keamanan yang sangat tinggi sehingga hanya pengguna yang memiliki keempat hal tersebut yang boleh menggunakan akun *twitter* sebagai salah satu *service* dari suatu sistem. Gambar 3.7 menunjukkan *consumer key* dan *access tokens* dari OAuth *twitter*.

```
# Consumer keys and access tokens, used for OAuth

consumer_key = 'pKXDsBkMfDcCtOAYaLEXwso6y'

consumer_secret =
'unlezwTlBG0lqof61Kvm8UTvTC6e8dQWHMglxp6xv3xy8vc1JY'

access_token = '734619274728312832-
71RoUmfvXzKZbb5d2NZSGvOWslzV5Ze'

access_token_secret =
'P1dRz5npRNQr6tkrBZRFtXzUPxqHkcOAV813ECsR4fHqO'
```

Gambar 3.7 OAuth *Twitter*

3.4.7. Konsep penerimaan data status dan pengolahan aktuasi pada *node*

Node menerima data hasil *broadcast* dari *coordinator* atau *gateway* yang berisikan status dari hasil pengujian data temperatur dan kelembaban udara gudang. Status yang dikirimkan dari *gateway* berupa 12 *byte* yang dimasukan ke paket data *frame API* untuk pengiriman *broadcast*. Panjangnya sama dengan panjang banyaknya karakter pada data yang dikirimkan oleh *gateway.* Gambar 3.8 menunjukan RF data status yang diterima oleh *node*.

0	1	2	3	4	5	6	7	8	9	10	11
Identitas XBee								**,**	**Status AC**	**,**	**Status *Exhaust Fan***

Gambar 3.8 Format data status

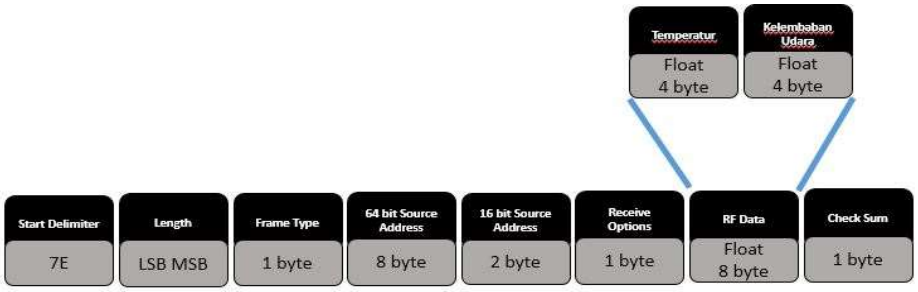
Setelah menerima RF data tersebut, *node* akan memecah setiap byte tersebut menjadi tipe data *array* sesuai dengan urutannya. *Array* nomor 0 sampai 8 menunjukan identitas XBee, *range array* ini menentukan status yang dikirimkan oleh *gateway* tertuju untuk *node* tersebut. *Array* nomor 8 dan 10 adalah pelengkap sesuai dengan format data status yang dikirimkan dari *gateway* dan pada *array* nomor 9 menunjukan status dari AC *array* nomor 11 menunjukan status dari *exhaust fan*. Logika dari status yang diberikan bernilai 1 atau 0, nilai 1 menyalakan dan 0 mematikan AC atau *exhaust fan* melalui 2 *channel relay* modul.

3.4.8. Format data komunikasi

Format data sebagai faktor penting dalam alur komunikasi data suatu jaringan baik nirkabel ataupun melalui kabel. Format data ini menentukan kebenaran informasi yang diterima. Sesuai dengan ketentuan komunikasi API pada XBee, data yang dikirimkan dari *node* dikirimkan dalam satu bentuk kesatuan

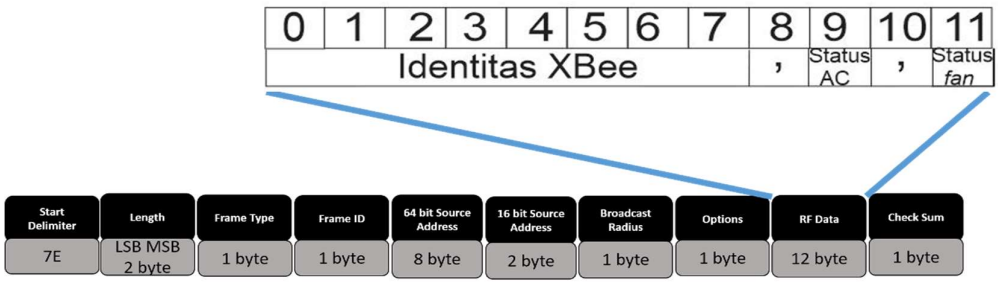
frame yang berisi *start bit*, *length, frame type, 65-bit source address, 16-bit source address, receive option,* RF *data,* dan *check sum* seperti pada Gambar 3.9.

RF data pada *frame* komunikasi XBee diisi oleh data temperatur dan kelembaban udara yang diperoleh dari sensor. *Frame* ini dikirimkan melalui jaringan XBee dan diterima oleh *coordinator* atau *gateway* dan diambil informasi RF data dan 16-bit *source address.*



Gambar 3.9 Format data yang digunakan komunikasi *node* ke *gateway*

Selanjutnya kedua informasi tersebut dikirimkan melalui jaringan internet dan dimasukan ke *database*. Setelah dimasukan ke *database*, *server* segera melakukan pengujian dan memberikan status yang diambil oleh *gateway. Gateway* mengirimkan pesan *broadcast* pada jaringan Zigbee dengan format data 12 *byte* pada RF data. Keseluruhan *frame* data yang dikirimkan *broadcast* melalui *gateway* dapat diamati pada Gambar 3.10.



Gambar 3.10 Format *frame* yang digunakan untuk *broadcast* status

3.5. Perancangan Perangkat Keras

3.5.1. Perancangan perangkat keras pada *node*

Perangkat keras yang dirancang pada *node* berupa mikrokontroler yang memiliki beberapa fungsi yang meliputi membaca temperatur dan kelembaban udara, komunikasi data nirkabel dengan *gateway,* indikator, dan aktuasi. Perangkat keras pada *node* juga harus memiliki sumber daya yang tepat, sehingga pada *node* perangkat keras yang digunakan adalah:

a. Arduino UNO

Arduino UNO berfungsi sebagai otak dari *node*. Arduino UNO melakukan pembacaan sensor, menampilkan indikator, mengirimkan data dan menerima data menggunakan XBee terhadap *gateway*, dan melakukan proses aktuasi.

b. XBee PRO S2B

XBee Pro S2B berfungsi untuk menghubungkan *node* dengan *gateway.* XBee mengirimkan data dengan format *frame* API untuk melakukan komunikasi pengiriman data sensor dan juga pengiriman status dari *node* masing-masing.

c. Sensor DHT22

Sensor DHT22 adalah sensor temperatur dan kelembaban udara yang berfungsi sebagai pengambil data dari *node.* Informasi yang diambil dari

sensor ini merupakan informasi utama dari komunikasi sistem dan sebagai informasi yang diuji di *server.*

d. RGB LED

RGB LED memiliki 3 warna yang pada sistem ini befungsi sebagai indikator dari sistem ini. Dengan warna hijau, merah, dan biru LED ini dapat membantu sebagai benda fisik yang memberikan informasi ketika sistem berjalan dan juga memberikan informasi keadaan gudang melalui status yang diterima.

e. 2 *Channel Relay* Modul

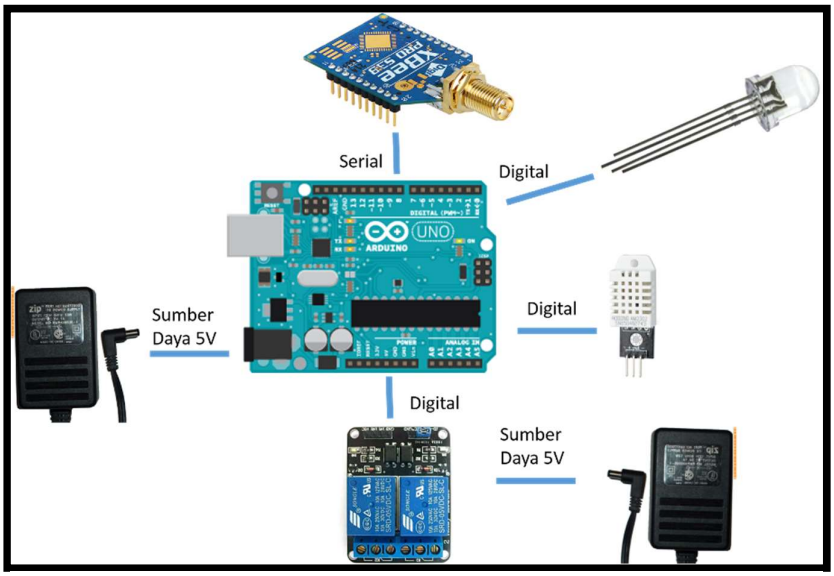
2 *channel relay* modul berfungsi sebagai saklar otomatis dari keseluruhan sistem. Dengan pengendalian yang dilakukan oleh mikrokontroler, *relay* ini dapat menyalakan dan mematikan AC dan *exhaust fan* secara otomatis berdasarkan dari status yang dikirimkan oleh *server* melalui *gateway.*

f. Sumber Daya 5 V

Sumber daya 5V adalah sumber daya *standard* yang digunakan oleh Arduino UNO dan 2 *channel relay* modul. Sumber daya yang dibutuhkan digunakan untuk memproses keseluruhan sistem pada *node.*

Perancangan perangkat keras *node* dapat diamati pada Gambar 3.11. Perancangan perangkat keras pada *node* menghubungkan 2 buah LED RGB yang berfungsi menjadi indikator dari masing-masing *channel relay* modul. LED RGB yang memiliki masing-masin 4 pin yang terdiri dari *ground*, merah, hijau, dan biru yang sesuai dengan namanya untuk menghidupkan LED tersebut. Pin LED hijau menjadi indikasi bahwa *channel relay* terputus hubungan sedangkan pin LED merah mengindikasikan *channel relay* terhubung dan menghidupkan AC atau *exhaust fan*. Pin hijau dari *channel relay* 1 terhubung dengan pin digital 7 dan *channel relay* 2 terhubung dengan pin digital 10 pada Arduino. Pin merah *channel relay* 1 terhubung dengan pin digital 5 dan *channel relay* 2 terhubung dengan pin digital 9, sedangkan untuk pin *ground* dari kedua LED tersebut terhubung dengan *Vin* dikarenakan jenis LED yang digunakan adalah jenis *common anode*, yaitu *ground* harus diberi tegangan lebih tinggi untuk menyalakan LED.

Untuk 2 *channel relay* modul sendiri, masing-masing *channel* dihubungkan secara NO atau *normally open*. *Channel relay* 1 terhubung dengan pin digital 11 pada Arduino dan *channel relay* 2 dihubungkan dengan pin digital 12 pada Arduino. Untuk sensor DHT22 terdapat 3 pin, yaitu *Vin*, *ground,* dan *data.* Pin *Vin* dan *ground* terhubung dengan *Vin* dan *ground* dari Arduino, sedangkan pin *data* terhubung dengan pin digital 4 pada Arduino.



Gambar 3.11 Perancangan perangkat keras pada *node*

3.5.2. Perancangan perangkat keras pada *gateway*

Perangkat keras pada *gateway* harus dapat menerima data yang dikirim dari *node* di gudang dan mampu mengolahnya. Data yang telah diolah tadi kemudian harus dikirimkan ke *server* melalu metode POST lewat perantara jaringan internet. Fungsi lainnya juga membutuhkan penerimaan status dari *server* dan memberikan peringatan serta mengirimkan status ke *node.* Gambar 3.12 memperlihatkan perancangan perangkat keras dari *gateway.* Dengan spesifikasi tersebut perangkat keras yang diperlukan adalah:

a. Raspberry Pi 3B

Raspberry Pi memiliki modul ethernet sehingga dapat berkomunikasi dengan jaringan internet. Raspberry pi juga memiliki banyak *library* yang dapat di-*install* salah satunya *twitter* sehingga dapat mengirimkan pesan

peringatan kepada pengguna. Raspberry pi juga memiliki kemampuan mengolah data yang telah dikirimkan dari *node* untuk dikirimkan ke *server.*

b. XBee PRO S2B

XBee pada *gateway* menjadi *coordinator* dalam suatu jaringan Zigbee sehingga berfungsi sebagai penerima informasi dari *node.* XBee pada juga bertanggung jawab dalam memberikan pesan *broadcast* yang berisikan status dari *node* pada gudang.

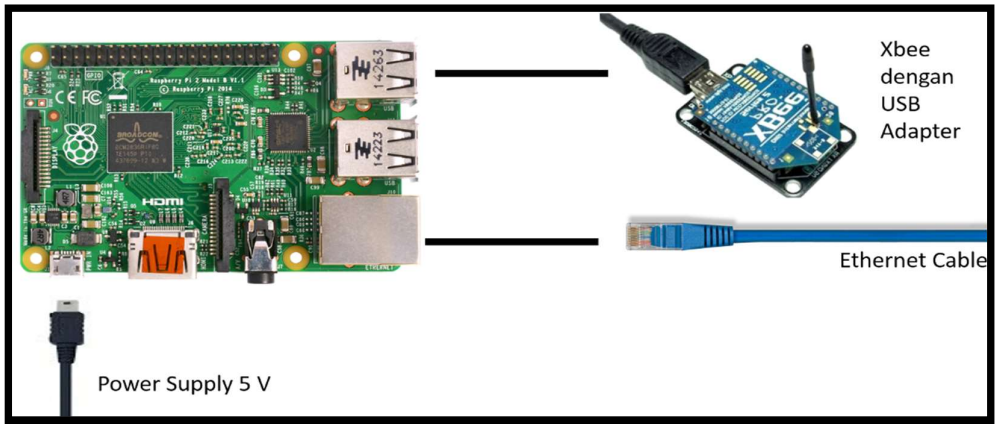
c. XBee USB Adaptor

Untuk memudahkan menghubungkan XBee dengan Rasberry Pi secara serial dibutuhkan suatu perangkat tambahan yang berupa XBee adaptor. Dengan adanya alat ini XBee dapat dihubungkan dengan Raspberry Pi melalui koneksi USB.

d. Sumber Daya 5 V

Untuk dapat membuat Raspberry Pi ini bekerja memerlukan sebuah sumber daya. Sesuai dengan ketentuan yang ada pada spesifikasi Raspberry Pi, dibutuhkan sumber daya 5V.

Sesuai dengan kebutuhan perangkat keras yang telah dirancang, perancangan perangkat keras dari *gateway* dapat diamati pada Gambar 3.12.

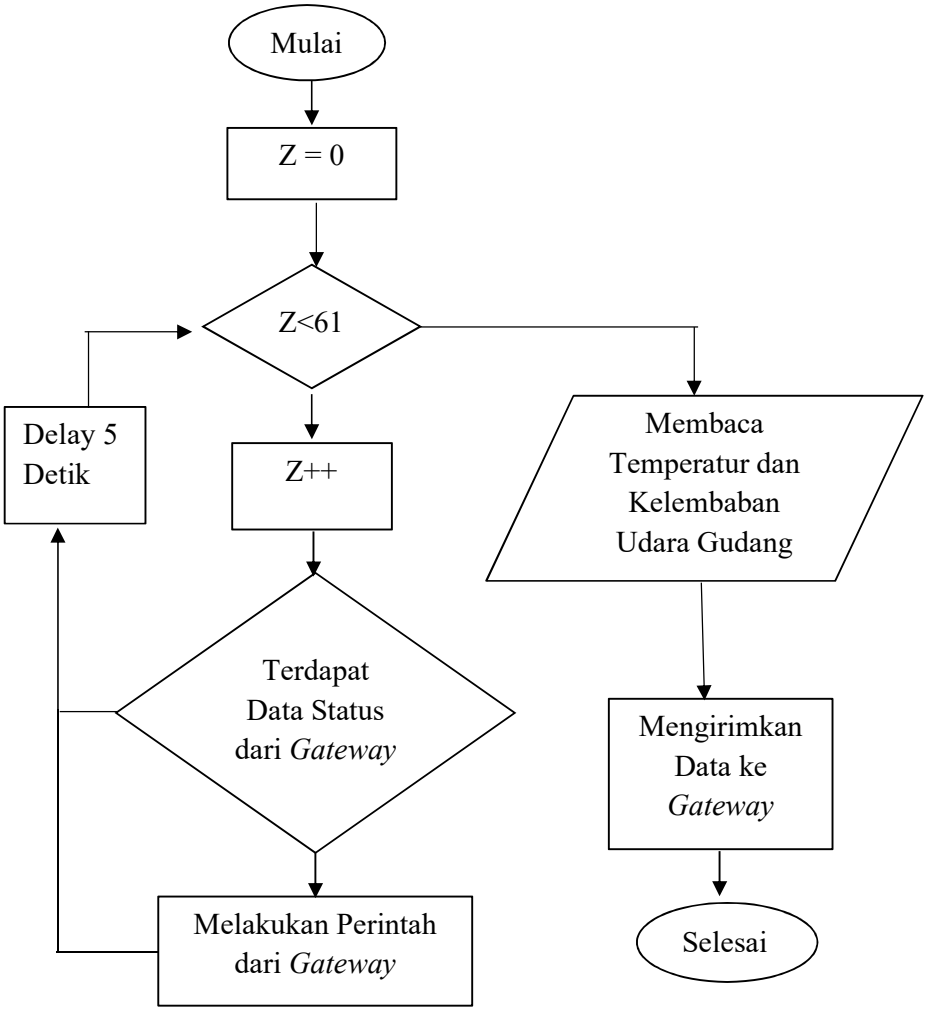


Gambar 3.12 Perancangan perangkat keras pada *gateway*

3.6. Perancangan Perangkat Lunak *Node*

Untuk melakukan perancangan perangkat lunak pada *node*, harus diketahui terlebih dahulu bahwa perangkat lunak *node* memiliki fungsi untuk mengambil data temperatur dan kelembaban udara pada gudang, mengirimkan data melalui XBee, menerima status dari *gateway* dan melakukan aktuasi pada gudang. Bahasa pemrograman yang digunakan dalam pembuatan perangkat lunak pada *node* menggunakan bahas pemrograman C. Diagram alir dari perangkat lunak yang ada pada *node* dapat diamati pada Gambar 3.13.

Nilai z lebih kecil dari 61 dan *delay* 5 detik merupakan keinginan pengguna, dalam penelitian kali ini inisiasi nilai z di *increment* setiap 5 detik sampai nilai z mencapai 60 artinya melakukan pengecekan jika terdapat data status yang dikirimkan dari *gateway* setiap 5 detik sampai mencapai 5 menit, setelah mencapai 5 menit *node* segera mengirimkan data temperatur dan kelembaban udara ke *gateway* untuk selanjutnya diproses seperti yang sudah dijelaskan pada konsep gambaran umum sistem.



Gambar 3.13 *Flowchart* perangkat lunak *node*

3.6.1. Pemanggilan *library*

Pemanggilan *library* pada Arduino IDE menggunakan perintah #*include* diikuti dengan nama *library* yang diawali dan diakhiri dengan tanda petik (“) dan (”) atau (<) dan (>). Cara melakukan pemanggilan *library* pada Arduino dapat diamati pada Gambar 3.14.

```
//Libraries
#include "DHT.h"
#include <XBee.h>;
```

Gambar 3.14 Pemanggilan *library node*

Library yang digunakan pada perangkat lunak adalah sebagai berikut:

a. XBee

Library XBee diperlukan untuk melakukan komunikasi data dengan perangkat XBee. *Library* tersebut berisikan berbagai fungsi untuk berkomunikasi dengan XBee seperti mengirim data dan mengambil data.

b. DHT

Library DHT berguna untuk mengakses DHT sebagai sensor dan memasukan data dari sensor DHT ke pin digital. *Library* tersebut berisikan konversi dari pembacaan temperatur dan kelembaban udara menjadi sebuah data *float* yang dapat di peroleh Arduino secara digital.

3.6.2. Inisiasi *mode* pin

Setiap pin dalam Arduino dapat terdefinisi sebagai *input* maupun *output*, ketika kita membaca sensor, pin yang terhubung dengan sensor memiliki fungsi sebagai *input*. Sedangkan jika kita membutuhkan pin sebagai informasi untuk *relay*, pin tersebut berfungsi sebagai *output*. Pada Gambar 3.15 dapat diamati *source code* dari inisialisasi *mode* pin.

Selain melakukan proses inisialisasi pin *mode*, dilakukan juga pengaturan awal dari masing-masing pin, *relay* 1 dan *relay* 2 HIGH atau terputus hubungannya, LED hijau dari *relay* 1 dan *relay* 2 menyala dan LED merah mati.

```
pinMode(RELAY1, OUTPUT);
pinMode(RELAY2, OUTPUT);
pinMode(LEDPIN1,OUTPUT);
pinMode(LEDPIN11,OUTPUT);
pinMode(LEDPIN2,OUTPUT);
pinMode(LEDPIN22,OUTPUT);
digitalWrite(RELAY2, HIGH);
digitalWrite(RELAY1, HIGH);
digitalWrite(LEDPIN22, HIGH);
digitalWrite(LEDPIN11, HIGH);
digitalWrite(LEDPIN2, LOW);
digitalWrite(LEDPIN1, LOW);
```

Gambar 3.15 Inisiasi *Mode* Pin

3.6.3. Pembacaan sensor DHT22 dan persiapan pengiriman data

Proses pembacaan temperatur dan kelembaban udara pada Arduino sebetulnya memiliki proses yang cukup mudah, karena adanya *library* DHT yang sangat membantu proses pembacaannya. Dengan fungsi *dht.read* Arduino sudah dapat membaca kondisi temperatur dan kelembaban udara dari DHT22. Gambar 3.16 menunjukan cara program membaca nilai temperatur dan kelembaban udara serta mempersiapkan data tersebut untuk dikirim dengan menggunakan metode *payload*.

```
hum = dht.readHumidity();
  temp= dht.readTemperatur();
  if (isnan(hum) || isnan(temp)){
   Serial.println("Failed to read sensor");
   return;
  }
  u.fval = temp;
   for ( i=0; i<4; i++){
     payload[i] = u.b[i];
                }
  u.fval = hum;
   for ( i=4; i<8; i++){
     payload[i] = u.b[i-4];
                 }
```

Gambar 3.16 Membaca sensor dan persiapan pengiriman data

Setelah membaca temperatur dan kelembaban udara, kedua nilai tersebut masih disimpan pada variabel hum untuk kelembaban udara dan temp untuk temperatur. Untuk dapat dikirimkan melalui komunikasi XBee, data harus dikemas dengan *payload. Payload* dibutuhkan karena dalam Arduino, data yang diterima dari sensor dan ditampilkan umumnya memiliki tipe data *string*, jika dalam komunikasi XBee data tersebut dikirim dalam bentuk *string* menjadi sangat panjang. Sebagai contoh nilai temperatur yang diperoleh dari sensor memiliki tipe data *float* dengan nilai 31.600000381469727 dalam bentuk *string* data tersebut membutuhkan 18 *bytes* hanya untuk mengirim temperatur, jika mengirim data temperatur dan kelembaban udara, dibutuhkan 36 *bytes.* Setiap karakter yang ditampilkan menjadi satu *bytes* dalam bentuk *string, payload* menjadikan data tersebut berbentuk *byte* sehingga hanya membutuhkan panjang data 8 *bytes.*

Semakin kecil ukuran *payload* kemungkinan terdapat kesalahan dalam pengiriman data semakin kecil terutama pada suatu jaringan dengan ukuran yang besar.

3.6.4. Pengiriman data ke *gateway*

Pengiriman data ke *gateway* membutuhkan pengalamatan pada awal program sehingga XBee dapat mengetahui alamat pengiriman data yang ingin dituju, pada sisi *node* data yang dikirimkan ke *gateway* memiliki alamat 0x00000000, 0x00000000 yang berarti alamat 64-bit untuk *coordinator* sesuai dengan Gambar 3.17. Alamat tersebut adalah format alamat yang telah ditentukan XBee untuk *coordinator*.

```
XBee XBee = XBee();
XBeeAddress64 addr64 =
XBeeAddress64(0x00000000, 0x00000000);
ZBTxStatusResponse txStatus =
ZBTxStatusResponse();
```

Gambar 3.17 Pengalamatan pengiriman data

Pengiriman data ke *gateway* menggunakan *void Send* (), fungsi *Send* ini menjadi indikator informasi dari proses pengiriman data ke *gateway*. Jika *node* berhasil mengirimkan data ke *gateway* pada *serial monitor* tercetak ”*Success. Time to celebrate*”, sedangkan jika proses pengiriman data ke *gateway* gagal *serial monitor* segera mencetak sebab dari kegagalan pengiriman data tersebut. Proses pengiriman data ke *gateway* dapat diamati pada Gambar 3.18.

```
void Send(){
   ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
   XBee.send(zbTx);
  if (XBee.readPacket(500)) {
     if (XBee.getResponse().isAvailable()) {
    }
    if (XBee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE) {
       XBee.getResponse().getZBTxStatusResponse(txStatus);
      if (txStatus.getDeliveryStatus() == SUCCESS) {
       Serial.println("Success.  time to celebrate");
       z=0;
      }
      else {
        z=0;
       Serial.println("The remote XBee did not receive our packet. Maybe it
powered off");
      }
    }
 }
 else if (XBee.getResponse().isError()) {
    z=0;
  Serial.print("Error reading packet.  Error code: ");
  Serial.println(XBee.getResponse().getErrorCode());
 }
 else {
    z=0;
  Serial.println("local XBee did not provide a timely TX Status Response");
}}
```

Gambar 3.18 Fungsi pengiriman pada *node*

3.6.5. Penerimaan status dari *gateway* dan proses aktuasi

Proses penerimaan status dari *gateway* memiliki prosedur yang tidak jauh berbeda dari proses pengiriman data ke *gateway.* Sesuai dengan *library* yang ada di XBee.h, pengaturan XBee agar bisa menerima tanggapan dari *coordinator*

membutuhkan pengaturan *modem status response* dan *ZB Rx response* sesuai dengan Gambar 3.20. Kedua hal tersebut menjadi indikasi dari XBee menerima paket data dari *coordinator*. Setelah melakukan pengaturan, proses penerimaan paket data dimulai sesuai dengan konsep gambaran umum dengan interval waktu 5 detik sekali dalam waktu 5 menit. Gambar 3.21 memperlihatkan proses interval penerimaan status.

Setelah Proses pengaturan dan penerimaan status, tertampil pada *serial monitor* data status yang berhasil diterima oleh *node* atau tidak, jika tidak ditampilkan pada *serial monitor* ”*Nothing to Do*”. Ketika terdapat sebuah informasi, data tersebut langsung diproses untuk memenuhi kebutuhan aktuasi pada *node* sebagai faktor pengendalian dari gudang. Data yang diterima pada *node* berupa *frame API* dan RF data akan dipecah menjadi tipe data *array* yang memiliki informasi satu per satu karakter dari data *string* yang dikirimkan oleh *gateway.* Gambar 3.19. menunjukkan pemrosesan status pada *node.*

Pertama kali yang dilakukan adalah mencocokan identitas dari XBee yang dimaksud oleh informasi status tersebut, jika identitas XBee yang dimiliki berbeda data tidak diproses dan *serial monitor* mencetak ”*Nothing to Do*”. Ketika identitas yang dimiliki sama, saat itulah proses aktuasi dimulai. Selain menjadi informasi bagi *relay*, 1 dan 0 juga berperan sebagai informasi dari LED indikator pada *node.*

```
void Receive(){
  if(XBee.readPacket(1000)){
    char datareceived[10];
    Serial.println(XBee.getResponse().getApiId());
    XBee.getResponse().getZBRxResponse(rx);
    for(int i=0; i<rx.getDataLength();i++){
     datareceived[i]=(char)rx.getData(i);}
     if(datareceived[0]=='4'){
      if(datareceived[1]=='0'){
       if(datareceived[2]=='e'){
        if(datareceived[3]=='b'){
         if(datareceived[4]=='0'){
          if(datareceived[5]=='b'){
           if(datareceived[6]=='6'){
            if(datareceived[7]=='4'){
             if(datareceived[9]=='1'){
              digitalWrite(RELAY1,LOW);
              digitalWrite(LEDPIN1, HIGH);
              digitalWrite(LEDPIN11,LOW);}
              else if(datareceived[9]=='0'){
               digitalWrite(RELAY1,HIGH);
               digitalWrite(LEDPIN1, LOW);
               digitalWrite(LEDPIN11, HIGH);                    }
              if(datareceived[11]=='1'){
               digitalWrite(RELAY2,LOW);
               digitalWrite(LEDPIN2,HIGH);
               digitalWrite(LEDPIN22,LOW);}
               else if(datareceived[11]=='0'){
                digitalWrite(RELAY2,HIGH);
                digitalWrite(LEDPIN2,LOW);
                digitalWrite(LEDPIN22,HIGH);}          }              }
}          }          }         }       }      }
     Serial.println(datareceived);
     Serial.println("Nice JOB!!");
     delay(5000);       loop();  }
```

Gambar 3.19 Pemrosesan data status pada *node*

```
for(z=0;z<61;z++){
  Receive();
  Serial.println(z);
  delay(5000);
        }
```

Gambar 3.20 Interval penerimaan status

```
XBeeResponse response = XBeeResponse();
// create reusable response objects for responses we expect to handle
ZBRxResponse rx = ZBRxResponse();
ModemStatusResponse msr = ModemStatusResponse();
```

Gambar 3.21 Pengaturan awal penerimaan status

3.7. Perancangan Perangkat Lunak pada *Gateway*

Sesuai dengan kebutuhan yang telah ditentukan pada perancangan perangkat keras pada *gateway,* perangkat lunak *gateway* berfungsi sebagai otak dari kebutuhan fungsi tersebut. Bahasa pemrograman yang digunakan pada perancangan perangkat lunak di *gateway* menggunakan bahasa pemrograman Python. *Gateway* memproses data yang diterima dari *node* untuk memastikan data data dapat diolah. *Gateway* melakukan proses pengecekan isi RF data dan *source address* dari keseluruhan paket data yang diterima. Jika data lolos proses pengecekan ini, selanjutnya data dimasukan ke variabel dan selanjutnya dikirimkan ke *server*. Setelah itu *gateway* mendapatkan data status dari *server* dan dikirimkan ke *node.* Tugas *gateway* tidak berhenti sampai tahap ini, *gateway* harus memenuhi fungsi memberikan peringatan kepada pengguna secara personal dengan karakteristik *twitter.*

3.7.1. Pemanggilan *library*

Pemanggilan *library* pada Python dibutuhkan di *gateway* untuk memenuhi kebutuhan fungsi yang diinginkan. *Library* yang dibutuhkan pada perangkat lunak ini mencakup seperti pada Gambar 3.22.

```
import serial
from XBee import Zigbee
import struct
import urllib
import urllib2
import datetime
import time
import tweepy
```

Gambar 3.22 *Library gateway*

Deskripsi dari *library* yang digunakan pada *gateway* adalah sebagai berikut:

a. Serial

   Library serial memiliki berbagai fungsi untuk melakukan komunikasi data *serial*. *Library* tersebut diperlukan karena Raspberry Pi berkomunikasi dengan perangkat XBee melalui komunikasi data *serial*.

b. XBee

   Library ini diperlukan karena *gateway* yang dirancang berkomunikasi dengan *sensor node* melalui prangkat XBee. XBee bekerja dengan *protokol* Zigbee, sehingga *library* ini berfungsi untuk perangkat lunak menerima data dari *node*.

c. *Struct*

Library Struct diperlukan untuk mengolah data dari *array byte* ke bentuk *float*. Fungsi tersebut digunakan saat melakukan *unpack* paket data yang diterima dari *node*.

d. *URL*lib dan *URL*lib2

Library ini digunakan agar dapat melakukan pesan *request* untuk mengirim data ke *server* melalui jaringan *internet*. *Library* ini memungkinkan program melakukan komunikasi melalui alamat *URL* yang ingin di tuju.

e. Datetime dan time

Library ini diperlukan untuk meng-*generate* waktu pengiriman data ke *server*.

f. Tweepy

Library ini digunakan untuk melakukan akses ke *twitter* melalui API *service* yang telah disediakan oleh *twitter.*

3.7.2. Penentuan PORT, *URL*, dan *baud rate*

Penentuan PORT dan *baud rate* pada *gateway* menjadi acuan dari proses pengiriman dan penerimaan data dari XBee secara *serial*. PORT adalah port serial dari Raspberry pi yang digunakan untuk melakukan komunikasi data. *Baud rate* yang digunakan pada umumnya dalam komunikasi XBee adalah 9600, namun tidak menutup kemungkinan bahwa *baud rate* yang lebih tinggi digunakan apabila data yang dikirimkan membutuhkan waktu pengiriman yang cepat. Untuk *URL*, hal tersebut merupakan akses *server* yang digunakan ketika proses pengiriman pesan

request dari *gateway* ke *server*. *URL* berisikan alamat tujuan dari *server* yang ingin dikirimkan data. Gambar 3.23 menunjukkan pengaturan PORT, *URL*, dan *baud rate*, sedangkan Gambar 3.24 menunjukkan akses serial XBee pada *gateway.*

```
url =
'http://smartcity.wg.ugm.ac.id/webapp/SmartRicestock/insert.php'

PORT = '/dev/ttyUSB0'

BAUD_RATE = 9600
```

Gambar 3.23 Pengaturan PORT, Url, dan *baud rate*

```
# Open Serial Port
ser = serial.Serial(PORT, BAUD_RATE)

# Create API Object
XBee = Zigbee(ser,escaped=False)
```

Gambar 3.24 Akses serial XBee

3.7.3. Penerimaan dan pemrosesan data dari *node*

Proses penerimaan data pada *gateway* diawali dengan melakukan konfirmasi dari data yang diterima dari *node*. Konfirmasi dilakukan dengan cara memastikan bahwa *gateway* menerima paket data yang benar dengan melakukan pemeriksaan paket data *source address long* dan RF data yang telah dikirimkan. Pemeriksaan data dilakukan dengan memeriksa *response* yang dikirimkan ke *gateway*.

Setelah mengetahui *source address long* dari *frame* data yang dikirimkan, RF data yang diperoleh dari *node* tersebut diterjemahkan menjadi data yang

berisikan temperatur dan kelembaban udara. Proses inilah yang dinamakan sebagai proses *unpacking* RF *data* dari yang bentuknya *array byte hexadecimal* menjadi sebuah data *float* yang siap dikirimkan ke *server*. Setelah mendapatkan nilai *float* tersebut, data keseluruhan dijadikan sebuah paket yang berisikan seluruh informasi yang dibutuhkan pada *server*. Gambar 3.25 memperlihatkan proses penerimaan dan pemrosesan data pada *gateway.*

```
# Continuously read and print packets
      time = datetime.datetime.now()
      response = XBee.wait_read_frame()
         if 'source_addr_long' in response:
            print 'ada Source Address'
            sa = hex(response['source_addr_long'][4:8])
         else:
            print 'Tidak ada Source Address'
            sa = 'No Source Adress Here'
            pass
         if 'rf_data' in response:
            print 'ada RF'
            rf = hex(response['rf_data'])
         else:
            print 'No RF data Received'
            rf = 'Nothing here to be processed'
            pass
      datalength=len(rf)
      print 'Data Recieved from Node'
          Time = unicode(time)
      # unpack into floats
      if datalength == 16:
         Temp          =
struct.unpack('f',response['rf_data'][0:4])[0]
         Hum           = struct.unpack('f',response['rf_data'][4:8])[0]
         Data={'Time':time,'ID':sa, 'Temp':Temp, 'Hum':Hum}
```

Gambar 3.25 Penerimaan dan pemrosesan data pada *gateway*

3.7.4. Pengiriman data ke *server* dan penerimaan status

Gateway menggunakan fungsi *try* pada bahasa pemrograman Python untuk melakukan pengiriman data ke *server*. Pengiriman data ke *server* dilakukan dengan metode POST yang mengirimkan pesan *request* menggunakan *library URLlib* dan *URLlib2* sesuai dengan Gambar 3.26.

Gateway mengirimkan notifikasi jika berhasil mengirimkan data ke *server*. Data yang berada pada *server* diproses sehingga menghasilkan sebuah informasi yang dikirimkan kembali ke *gateway* dan *node* untuk memberi peringatan dan perlakuan yang dibutuhkan agar temperatur dan kelembaban udara dapat menghambat pertumbuhan hama gudang. *Gateway* harus menerima status yang diberikan oleh *server* dengan menggunakan fungsi *response read* seperti pada Gambar 3.27. *Gateway* menerima pesan dari *server* yang mengirimkan informasi datanya menggunakan pesan *request*. Data yang diterima tersebut berisi *string* yang memiliki informasi status yang dicetak oleh *server* setelah data diuji dengan batas yang diinginkan oleh pengguna.

```
try:
        print "Sending Data to Server"
        print Time
        data = urllib.urlencode(Data)
        req = urllib2.Request(url, data)
        print "Sending to Server Success"
```

Gambar 3.26 Mengirimkan data ke *server*

```
response = urllib2.urlopen(req)
status = response.read()
print status
print response
print "Status Recieved from Server"
```

Gambar 3.27 Menerima status dari *server*

3.7.5. Pengiriman status ke *node*

Pengiriman status kembali ke *node* menggunakan XBee memiliki konsep yang sama ketika menerima data dari *node* pada *gateway*, namun pada kali ini fungsi yang digunakan bukan XBee *read frame* melainkan XBee *Send*. Pada fungsi *send* ini, *gateway* perlu mendefinisikan alamat dari XBee tujuan yang dikirimkan informasinya. Pada kali ini, *gateway* mengirimkan pesan *broadcast* ke seluruh jaringan *node* yang terhubung dengan alamat tujuan yang berisi 0x000000000000FFFF untuk *destination address long* dan 0xFFFC untuk *destination address* sesuai dengan ketentuan yang dibuat oleh protokol jaringan Zigbee. Gambar 3.28 memperlihatkan *gateway* mengirim status ke *node.*

```
XBee.send("tx",
dest_addr_long='\x00\x00\x00\x00\x00\x00\xFF\xFF' , dest_addr
= '\xFF\xFC', data=str(status))
        response = XBee.wait_read_frame()
        print "Status Sent to Node"
```

Gambar 3.28 Mengirim status ke *node*

3.7.6. Sistem peringatan *twitter*

Sistem peringatan *twitter* pada *gateway* bergantung pada informasi status yang dikirimkan dari *server*. Sistem peringatan ini bersifat spesifik terhadap status yang didapatkan, jika hanya AC yang menyala notifikasi yang diberikan hanya

berisi AC yang menyala, begitupun seterusnya. Setelah mengikutsertakan *library tweepy, gateway* menerjemahkan informasi yang di masukan ke notifikasi *twitter.*

Informasi pada Gambar 3.29 menerjemahkan informasi *source address long* menjadi sebuah informasi yang mudah dimengerti oleh pengguna, sehingga pengguna dapat mengetahui keadaan *realtime* pada gudang. Setelah melakukan penerjemahan tersebut, informasi status yang diterima segera dieksekusi dengan perangkat lunak yang ada pada Gambar 3.30

```
if sa == '40eb0b64':
        NODE =  'Gudang Bulog 1'
    else :
        NODE =  'X'
```

Gambar 3.29 Penerjemahan alamat *node*

Status yang diterima memiliki format alamat *node*, status temperatur, status kelembaban udara, ketika bernilai 1 artinya temperatur atau kelembaban udara yang diperoleh melebihi dari batas yang dinginkan oleh pengguna, sedangkan jika bernilai 0 artinya keadaan masih dibawah batas yang ditentukan. Dengan fungsi *api.update_status* yang telah ditentukan oleh *library* tweepy, karakter yang ingin dikirimkan serta variabel dikirimkan ke *twitter* dan segera di *tweet*. Pada fungsi peringatan ini juga terdapat variabel *alert* dan *count* yang telah didefinisikan sebelumnya pada *gateway* dan diatur dengan nilai awal 0. Kedua variabel tersebut berfungsi sebagai proses peringatan kepada pengguna apabila setelah 1 jam proses pemonitoran, temperatur dan kelembaban udara pada gudang tersebut masih melebihi batas yang diinginkan oleh pengguna selama 40 menit. Nilai 1 jam dan 40 menit tersebut merupakan nilai *user defined* sehingga dapat diubah-ubah melalui

program Python pada *gateway*. Gambar 3.30 menunjukkan sistem peringatan *twitter* pada *gateway*.

```
if status =='40eb0b64,1,1':
                    Alert += 1
                    Count += 1
                    api.update_status('Time : %s \n ID: %s \n Hum: %%%s%% \n
                    Temp : %%%s%% \n Melewati Batas \n AC dan Exhaust
                    Menyala'%(Time, NODE, Hum, Temp))
                    print "Tweet Success"
 elif status == '40eb0b64,1,0':
                    Alert += 1
                    Count += 1
                    api.update_status('Time : %s \n ID: %s \n Hum: %%%s%% \n
                    Temp : %%%s%% \n Melewati Batas \n AC Menyala'%(Time,
                    NODE, Hum, Temp))
                    print "Tweet Success"
elif status == '40eb0b64,0,1':
                    Alert += 1
                    Count += 1
                    api.update_status('Time : %s \n ID: %s \n Hum: %%%s%% \n
                    Temp : %%%s%% \n Melewati Batas \n Exhaust
                    Menyala'%(Time, NODE, Hum, Temp))
                    print "Tweet Success"
 else :
              Count +=1
```

Gambar 3.30 Peringatan *twitter gateway*

Count menjadi variabel penghitung waktu yang ditempuh selama proses pemonitoran, sedangkan *alert* hanya bertambah nilainya ketika terdapat sebuah faktor temperatur atau kelembaban udara yang melebihi dari batas yang ditentukan. Pada saat nilai *count* bernilai 12 artinya sudah melakukan pengiriman data dari *node* selama satu jam, *gateway* melakukan proses informasi kedua variabel tersebut dan

mengirimkan *tweet* peringatan kepada pengguna untuk melakukan sebuah tindakan khusus ketika nilai *alert* bernilai 8 atau lebih, yang artinya selama 40 menit ke atas temperatur dan kelembaban udara tersebut melebihi batas. Setelah melakukan proses tersebut maka nilai variabel *alert* dan *count* diatur kembali menjadi 0 begitu juga jika tidak diperlukan peringatan khusus selama 1 jam. Gambar 3.31 memperlihatkan proses peringatan setelah 1 jam pada *gateway.*

```
print "5 Menit ke"
print Count
print "Jumlah Alert"
print Alert
if Count == 12:
        if Alert >= 8:
                        api.update_status("Time : %s \n @BULOG Segera
                        Lakukan Fumigasi Pada %s Karena ada Kemungkinan
                        Ledakan Hama"%(Time, NODE))
                        print "Tweet Alert Success"
                        Alert = 0
                        Count = 0
        else:
                        Alert = 0
                        Count = 0
```

Gambar 3.31 Peringatan setelah 1 jam

3.8. Perancangan Perangkat Lunak *Server*

Perancangan perangkat lunak pada *server* berfungsi sebagai proses interaksi antara jaringan sensor dengan pengguna, sehingga dibutuhkan beberapa fungsi yang dapat menyajikan data yang dapat dimengerti oleh pengguna dan dapat berinteraksi agar jaringan sensor dapat bekerja sesuai dengan keinginan dari pengguna. *Server* berfungsi sebagai pemberi informasi pada halaman *web* dan

mengirimkan status setelah proses pengujian data temperatur dan kelembaban udara.

3.8.1. Pengambilan data *trigger*

Pengambilan data *trigger* atau batas yang diinginkan oleh pengguna menggunakan bahasa pemrograman PHP. Dengan fungsi *update* yang dimiliki oleh PHP, nilai batas temperatur dan kelembaban udara diperbaharui pada *database* ketika pengguna melakukan *submit* data batas temperatur dan kelembaban udara melalui halaman *web*. Gambar 3.32 menunjukkan program proses pembaruan data pada *database*.

Variabel suhu dan lembab pada program menyimpan nilai yang diinginkan oleh pengguna pada halaman *web trigger* dengan metode POST yang berasal dari nama box yang ada pada *web*. Pembaruan nilai batas tersebut dilakukan dengan fungsi *mysqli_query.* Dengan *query* ini, *server* memperbaharui data yang terdapat pada *database* dengan data yang baru di *submit* oleh pengguna.

```
<?php
 include 'configweb.php';
   $suhu = $_POST['batas-suhu'];
   $lembab = $_POST['batas-kelembaban'];
   $update_triger = mysqli_query($conn, "UPDATE treshold SET
BatasSuhu = '$suhu', BatasKelembaban = '$lembab' WHERE No = 1");
   header('Location:index.php');
?>
```

Gambar 3.32 Program pembaruan batas

Gambar 3.33 adalah program yang digunakan untuk melakukan *input* nilai yang diinginkan oleh pengguna pada halaman *web*. Dengan fungsi *form-control* pada PHP, maka data yang telah di-*submit* oleh pengguna segera diproses oleh program *trigger_update.php* yang melakukan pembaharuan pada *database* tadi dan siap untuk melakukan pengujian pada data selanjutnya dengan batas yang baru.

```
<div class="box-body">
          <?php
           $get_triger = mysqli_query($conn, "SELECT * FROM treshold
WHERE No = 1");
           $triger = mysqli_fetch_assoc($get_triger);
           $batassuhu = $triger['BatasSuhu'];
           $bataskelembaban = $triger['BatasKelembaban'];
          ?>
           <form action="triger_update.php" method="POST">
            <div class="col-md-offset-3 col-md-2">
             <input type="number" name="batas-suhu" class="form-
control" value="<?php echo $batassuhu;?>">
             <div class="text-center">Batas Suhu (˚C) </div>
            </div>
            <div class="col-md-offset-2 col-md-2">
             <input type="number" name="batas-kelembaban"
class="form-control" value="<?php echo $bataskelembaban;?>">
             <div class="text-center">Batas Kelembaban (%)</div>
            </div>
            <div class="col-md-offset-3 col-md-6">
             <hr>
            </div>
            <div class="col-md-offset-3 col-md-6 text-center">
             <input type="submit" class="form-control btn btn-primary
btn-flat" value="Submit">
            </div>
           </form>
          </div><!-- /.box-body -->
```

Gambar 3.33 Program halaman web *input trigger*

3.8.2. Proses pengujian data

Dengan metode POST, data yang diterima dimasukkan ke variabel masing-masing dan segera dimasukan ke *database* tabel *smart* tempat seluruh data dari temperatur, kelembaban udara, waktu, dan status berada sehingga data segera disimpan. Setelah data tersebut tersimpan, maka program pada *server* mengambil kembali seluruh data yang ada pada tabel *smart* dan *treshold* untuk dilakukan pengujian. Data yang diambil berbentuk *array* dengan mengambil setiap komponen data tersebut kembali dimasukkan ke variabel yang dibutuhkan untuk melakukan pengujian.

Gambar 3.34 menunjukan proses pengujian data yang dilakukan pada *server*. Pengujian yang dilakukan terdapat 4 jenis, yaitu ketika temperatur dan kelembaban udara keduanya melebihi batas dari keinginan pengguna, ketika hanya temperatur melebihi batas, dan ketika hanya kelembaban udara yang melebihi batas dari pengguna dan jika tidak termasuk dari ketiga pengujian tersebut, maka nilai temperatur dan kelembaban udara yang ada pada gudang masih dalam batas aman.

Setelah pengujian tersebut, dapat ditentukan status dari keadaan gudang yang dikirimkan ke *node* dan *gateway*. *Server* melakukan pembaruan status pada *database* tabel *smart* dengan fungsi *update* dan *server* mencetak status tersebut dengan fungsi *echo* agar ketika *gateway* membuka *URL*, *gateway* mendapatkan *response* status yang berupa *string*.

```
if($cek_kelembaban > $bataskelembaban && $cek_temperatur >
$batassuhu)
{
$ubah = $conn -> query ("UPDATE smart set Stat = '$ID,1,1' where Time =
'$cek_time'");
                    }
else if($cek_kelembaban > $bataskelembaban)
{
$ubah = $conn -> query ("UPDATE smart set Stat = '$ID,0,1' where Time =
'$cek_time'");
                    }
elseif ($cek_temperatur > $batassuhu)
{
$ubah = $conn -> query ("UPDATE smart set Stat = '$ID,1,0' where Time =
'$cek_time'");
   }
Else
{
$ubah = $conn -> query ("UPDATE smart set Stat = '$ID,0,0' where Time =
'$cek_time'");
   }
}
}
$status = mysqli_query($conn, "SELECT Stat FROM smart WHERE ID = '$ID'
AND Time ORDER BY NO DESC LIMIT 1");
$statusok = mysqli_fetch_assoc($status);
$statusok = $statusok['Stat'];
echo "$statusok";
```

Gambar 3.34 Pengujian pada *server*

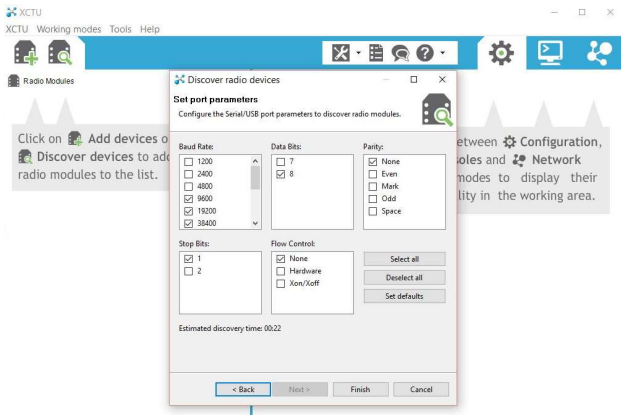
3.9. Konfigurasi XBee

Konfigurasi XBee pada penelitian ini dilakukan dengan perangkat lunak XCTU. Konfigurasi ini dilakukan dengan pertama kali memilih *baud rate* serta *data bits* yang dimiliki oleh XBee seperti pada Gambar 3.35. Setelah melakukan

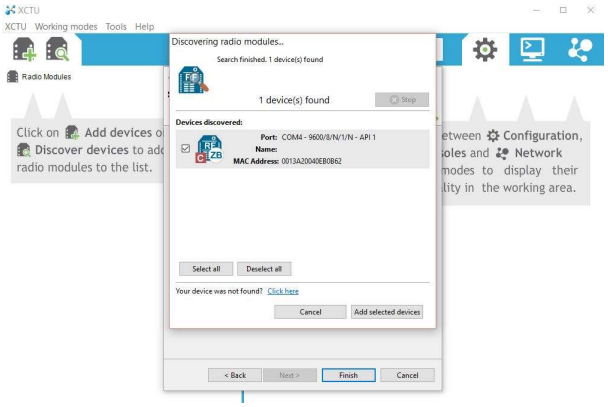
pemilihan tersebut, maka muncul XBee yang sudah terhubung dengan komputer pengguna seperti pada Gambar 3.36.

Pengaturan jenis XBee antara *coordinator* dan *end device* juga dilakukan pada XCTU. XCTU dengan versi yang semakin baru memiliki versi *coordinator* atau *end device* yang semakin baru juga. Pengaturan untuk XBee tipe *end device* dapat diamati pada Gambar 3.37 dan untuk tipe *coordinator* dapat diamati pada Gambar 3.38.

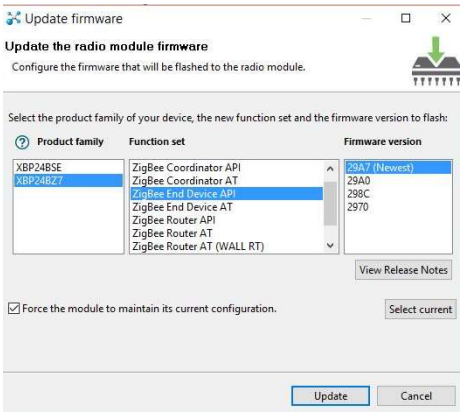
Setiap XBee diberikan PAN ID yang sama dan bersifat spesifik, dengan PAN ID ini seluruh XBee yang memiliki PAN ID yang sama dapat saling berkomunikasi sesuai dengan tipenya masing-masing. PAN ID ini dapat memiliki panjang 64-bit maksimal. Pada penelitian kali ini, PAN ID yang digunakan pada jaringan Zigbee adalah 99 seperti pada Gambar 3.39.



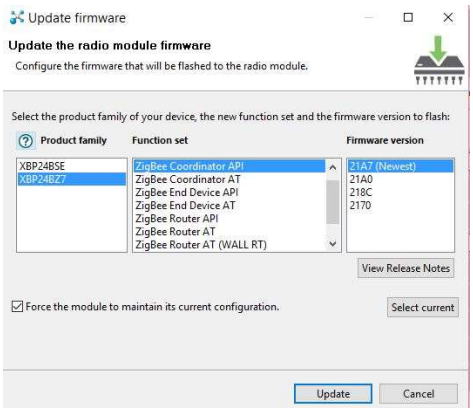
Gambar 3.35 Pengaturan *baud rate* dan *data bits* XCTU



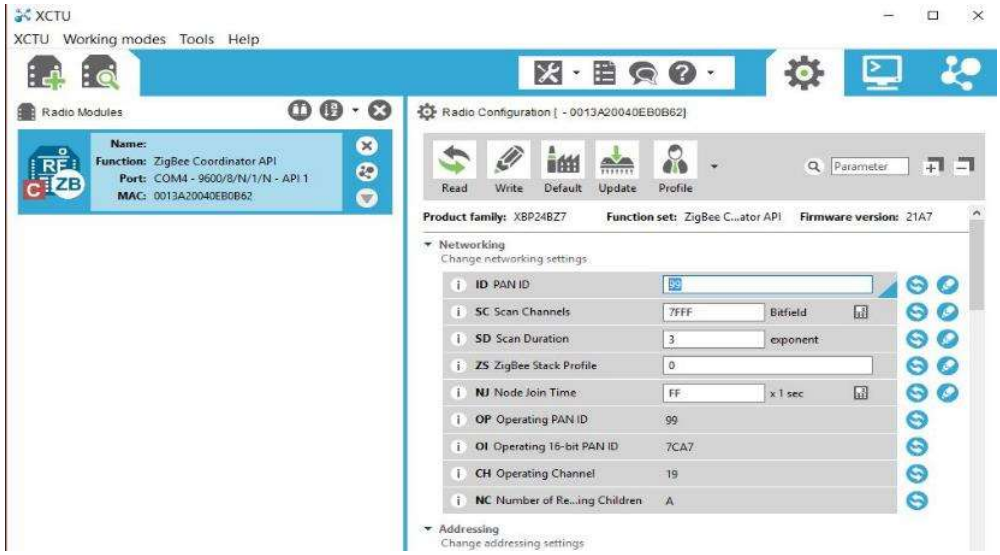
Gambar 3.36 Pemilihan XBee pada XCTU



Gambar 3.37 Pengaturan XBee tipe *end device*



Gambar 3.38 Pengaturan XBee tipe *coordinator*



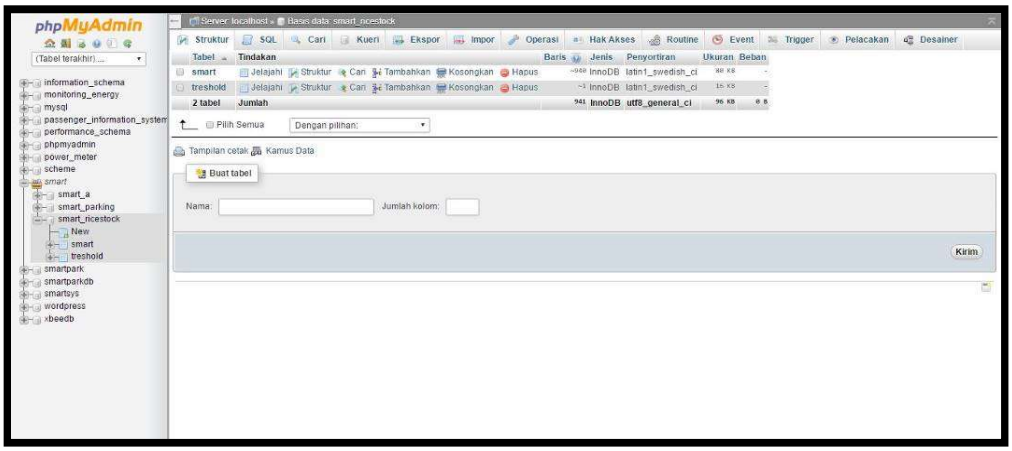
Gambar 3.39 Pengaturan PAN ID

3.10. Persiapan *Database Server*

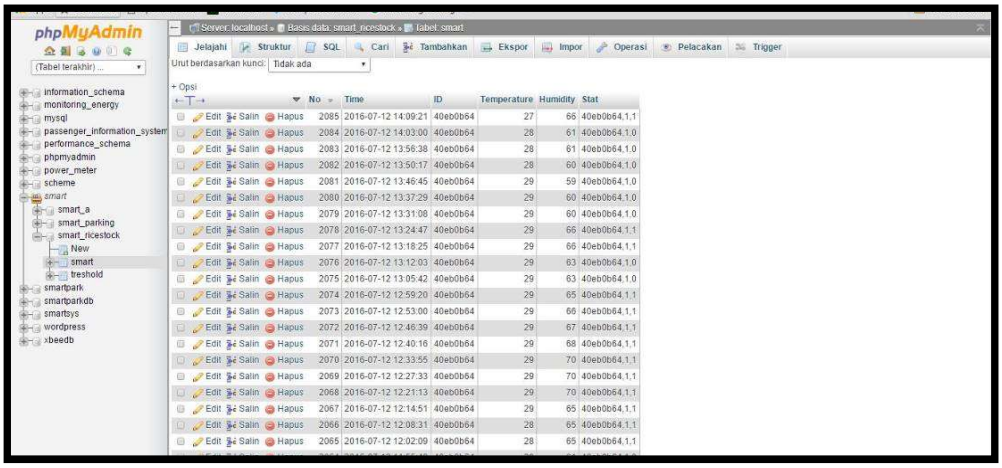
Perisapan *database* pada *server* dilakukan dengan mengakses http://10.13.247.248/phpmyadmin dan melakukan pembuatan *database* bernama *smart*_ricestock sesuai dengan Gambar 3.40. Pembuatan *database* ini berguna untuk *gateway* dalam melakukan akses ke *server* yang melakukan proses pengolahan data.

Setelah melakukan pembuatan *database*, maka dibutuhkan tabel dalam *database* tersebut untuk menyimpan data yang spesifik untuk diproses, dalam hal ini dibutuhkan 2 tabel yang berhubungan dengan sistem ini, yaitu tabel *smart* yang berisi data waktu, identitas, temperatur, kelembaban udara, dan status. Tabel kedua yang dibutuhkan adalah tabel *treshold*, dalam tabel tersebut berisi nilai batas temperatur dan kelembaban udara yang diinginkan oleh pengguna dan selalu

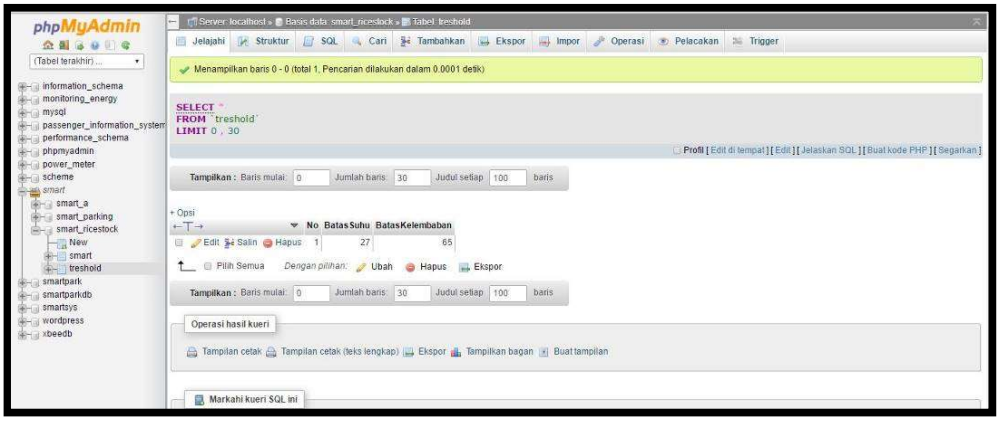
diperbaharui ketika pengguna melakukan *submit* data yang baru. Gambar 3.41 dan 3.42 memperlihatkan tabel smart dan tabel treshold



Gambar 3.40 *Database* smart_ricestock



Gambar 3.42 Tabel smart

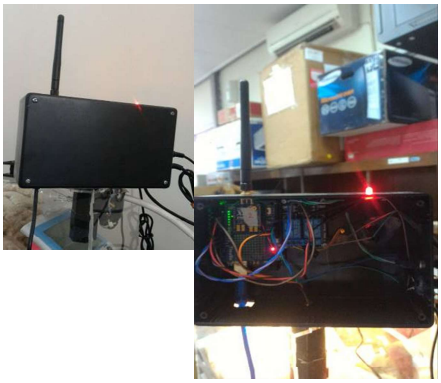


Gambar 3.41 Tabel treshold

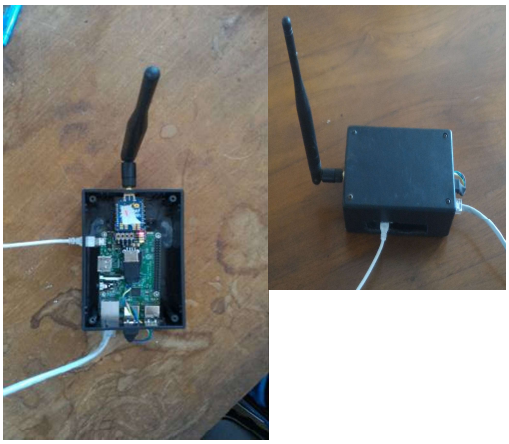
BAB IV
HASIL DAN PEMBAHASAN

4.1. Persiapan Pengujian Sistem

Persiapan pengujian sistem dilakukan dengan terlebih dahulu merangkai sensor *node* dan juga *gateway* dari perancangan perangkat keras menjadi sebuah rangkaian yang nyata sehingga dapat dilakukan pengujian secara nyata. Hasil rangkaian *node* dan *gateway* dapat diamati pada Gambar 4.1 dan 4.2.



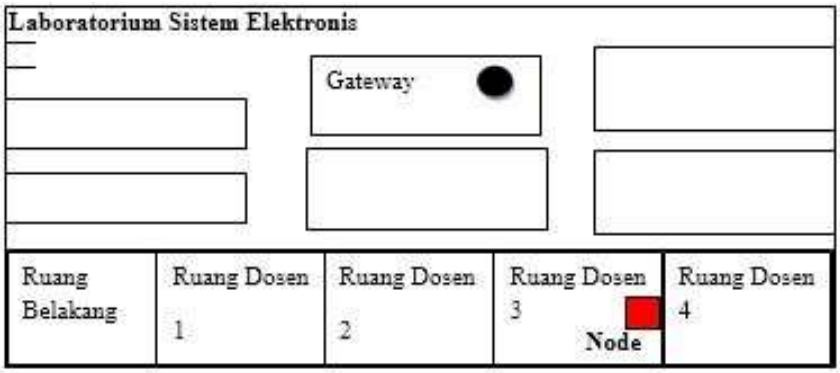
Gambar 4.1 Tampak luar dan dalam *node*



Gambar 4.2 Penampakan luar dan dalam *gateway*

Setelah selesai merangkai perangkat keras, pengujian sistem keseluruhan dapat dilakukan ketika program yang telah dibuat oleh pengguna diunduh ke dalam Arduino pada sisi *node*, Raspberry Pi pada sisi *gateway*, dan pada sisi *server*. Pengujian pada *node* dilakukan dengan menggunakan *serial monitor* yang disediakan oleh *software* Arduino IDE. Pengujian pada *gateway* dilakukan dengan *software* PuTTY, sebuah *software* yang memungkinkan komunikasi dengan protokol SSH sehingga dapat mengakses Raspberry Pi. Tampilan dari PuTTY berupa terminal yang menjalankan program pada Raspberry Pi pada komputer yang sedang melakukan akses.

Pengujian sistem ini dilakukan dengan meletakan *gateway* pada meja yang berada pada Laboratorium Sistem Elektronis dan *node* pada ruang dosen 3 yang berada pada Laboratotium yang sama. Denah dapat diamati pada Gambar 4.3. Pengujian pada sensor *node* dibantu dengan penggunaan lampu pijar untuk membantu proses pengujian sistem terhadap fluktuasi temperatur dan kelembaban udara.



Gambar 4.3 Denah tempat pengujian

4.2. Pengujian Sistem

Proses pengujian sistem pada penelitian ini dibagi menjadi 3, yaitu pengujian sistem pada *node, gateway,* dan *server.* Tujuan dari pengujian sistem ini adalah untuk memastikan bahwa sistem sudah berjalan sesuai fungsi yang dibutuhkan dan sebagai penilaian indikator dari kebutuhan sistem yang ada pada proses pemonitoran dan pengendalian gudang penyimpanan pascapanen. Pengujian sistem yang dilakukan pada *node* berupa:

a. Pengujian pengiriman data ke *gateway,* pengujian ini bertujuan untuk memastikan bahwa *node* dapat menjalankan fungsinya setelah membaca data, yaitu untuk mengirimkan data ke *gateway* dengan waktu yang sudah ditentukan.

b. Pengujian penerimaan data status, selain pengiriman data temperatur dan kelembaban udara pada *node* maka *node* juga harus mampu menerima data *broadcast* status yang dikirimkan oleh *gateway.*

c. Pengujian penerjemahan data status dan aktuasi sistem, setelah menerima data status maka *node* diuji untuk memastikan bahwa status dapat diterjemahkan oleh *node* dan melakukan segera aktuasi sistem.

Sedangkan pada *gateway* dibutuhkan beberapa pengujian, yaitu:

a. Pengujian pengiriman data ke *server,* pada *gateway* pengujian ini bertujuan untuk memastikan bahwa *gateway* dapat meneruskan data dari *node* ke *server.*

b. Pengujian penerimaan status dari *server* dan sistem peringatan *twitter,* pengujian ini bertujuan untuk memperlihatkan kemampuan *gateway* dalam menerima status dari *server* serta menerjemahkan status yang diterima dari *server* dan kemampuan untuk memberikan peringatan kepada pengguna.

Terakhir pada *server* dilakukan pengujian sebagai berikut:

a. Pengujian pengambilan data batas yang diinginkan oleh pengguna, pengujian ini bertujuan untuk membuktikan bahwa pengguna dapat selalu melakukan pembaharuan terhadap batas temperatur dan kelembaban udara yang diinginkan.

b. Pengujian data, pengujian data yang dimaksud pada pengujian ini adalah melakukan pembandingan data pada *server* antara data yang diterima dari sistem dengan data yang diinginkan oleh pengguna.

c. Pengujian penampilan data dalam bentuk tabel dan grafik, pengujian ini membuktikan bahwa *server* dapat menampilkan data yang diterima dan diproses secara sederhana.

Pengujian seluruh sistem tersebut merupakan pengujian keseluruhan sistem yang bekerja dalam perancangan sistem pengendalian dan pemonitoran ini. Pengujian sistem tersebut memperlihatkan cara kerja dari sistem yang telah dirancang mulai dari awal pengambilan data, pengiriman, dan proses aktuasi baik dari sisi *node, gateway,* dan *server* karena ketiga hal tersebut adalah satu kesatuan

sistem ini. Untuk penjelasan lebih lanjut tentang masing-masing poin dari pengujian sistem dibahas pada sub-bab berikutnya.

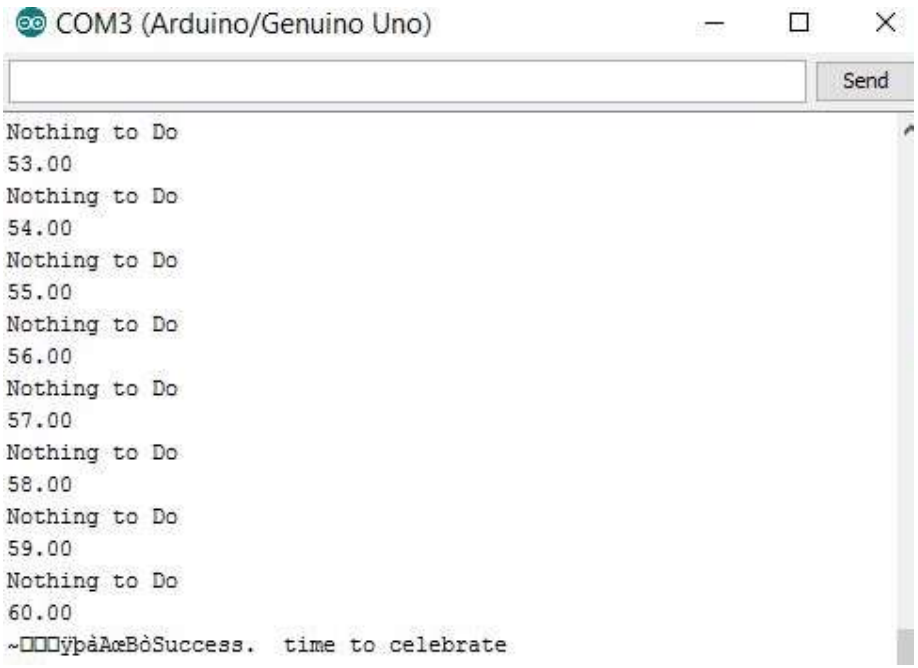
4.3. Pengujian *Node*

Pada sub-bab ini dibahas keseluruhan pengujian sistem kerja dari *node*. Sensor *node* pada sistem ini memiliki fungsi sebagai pengambil data, dan penerima status aktuasi sehingga dalam pengujiannya, harus memperlihatkan bahwa *node* dapat melakukan tugas tersebut sesuai dengan keinginan dari pengguna dan dapat terhubung dengan keseluruhan jaringan sistem dengan *gateway* dan *server.*

4.3.1. Pengujian pengiriman data

Pengujian pengiriman data ini dilakukan dengan menggunakan *serial monitor* pada Arduino IDE. Sesuai dengan perancangan sistem, serial *monitor* pada Arduino IDE akan mencetak "*Success time to celebrate"* ketika berhasil mengirimkan data ke *gateway* seperti pada Gambar 4.4.

Pengiriman data dilakukan ketika ada angka 60 pada *serial monitor*, artinya program yang dijalankan Arduino untuk mengirimkan data setiap 5 menit sekali berjalan dengan baik, karena 60 berarti adalah 5 detik yang ke 60 kali sehingga jika dikalikan menghasilkan 300 detik atau 5 menit.



Gambar 4.4 Pengiriman data dari *node*

4.3.2. Penerimaan data status

Pengujian ini bertujuan untuk menguji *node* yang dapat menerima data dari *gateway* bukan hanya mengirim data ke *gateway* untuk menjalankan sistem aktuasi. Pengujian penerimaan data status pada *node* dilakukan dengan *serial monitor* Arduino IDE. Pengujian ini dilakukan secara terus menerus selama *node* menyala. Sesuai dengan ketentuan sistem yang telah dirancang bahwa setiap 5 detik sekali *node* mendeteksi jika terdapat status yang dikirimkan dari *gateway* yang membawa informasi keadaan yang harus dihubungkan oleh *node*. Dengan pengujian sebanyak 100 kali menerima informasi yang dikirimkan dari *gateway* setiap 5 menit, didapatkan persentase keberhasilan menerima data dari *gateway* sebesar 100 persen sesuai dengan pengambilan data pada tanggal 12 Juli 2016 yang dimulai pada Pukul 9.11 WIB sampai dengan 20.03 WIB sebanyak 90 data dan 20 data pada 13 Juli

2016 mulai pukul 08.03 WIB sampai 09.02 WIB. Pengambilan sampel sebanyak 100 kali sesuai dengan jumlah sampel minimal menurut Roscoe (Roscoe, 1975). Dengan demikian dapat disimpulkan bahwa *node* berhasil dalam menjalankan sistem pengendalian dengan dapat menerjemahkan status yang diterjemahkan menjadi informasi untuk memberikan aktuasi.

Penerimaan status memiliki kode *response* unik tersendiri pada *node* ketika menerima status tersebut. Kode unik ini terlihat pada *serial monitor* sebagai salah satu indikator yang disediakan oleh XBee bahwa XBee menerima data, kode tersebut adalah 144, jika diubah menjadi *hexadecimal* adalah 0x90. Kode tersebut merupakan pengaturan yang ada pada *library* XBee untuk menandakan *RX_Response* atau berhasil menerima data. Pengamatan pada Tabel 4.1 dinyatakan berhasil ketika kode unik tersebut muncul. Gambar 4.5 dapat memperlihatkan proses penerimaan status pada *node.*

Tabel 4.1 Hasil pengamatan penerimaan status

Waktu	Status
7/12/2016 9:11	Berhasil
7/12/2016 9:17	Berhasil
7/12/2016 9:23	Berhasil
7/12/2016 9:29	Berhasil
7/12/2016 9:36	Berhasil
7/12/2016 9:42	Berhasil
7/12/2016 9:48	Berhasil
7/12/2016 9:55	Berhasil
7/12/2016 10:01	Berhasil
7/12/2016 10:08	Berhasil
7/12/2016 10:14	Berhasil
7/12/2016 10:20	Berhasil
7/12/2016 10:27	Berhasil

```
144
40eb0b64,0,1
Nice JOB!!
Nothing to Do
0.00
Nothing to Do
1.00
Nothing to Do
2.00
Nothing to Do
3.00
```

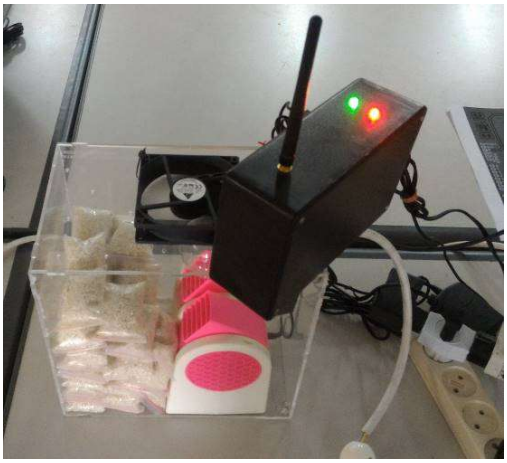
Gambar 4.5 Penerimaan status pada *node*

4.3.3. Penerjemahan data status dan aktuasi sistem

Penerjemahan data status dilakukan oleh *node* agar dapat mengetahui aktuasi yang harus dilakukan yang diakibatkan temperatur dan kelembaban udara melebihi batas yang ditentukan pengguna. Penerjemahan data dilakukan dengan pertama kali membandingkan nilai identitas pada data status dengan identitas XBee pada *node.* Setelah melakukan pengujian pada identitas, *node* segera mengeksekusi status 1 atau 0 yang dikirimkan dari *gateway* untuk mengendalikan AC atau *exhaust fan* juga LED indikator. Pada pengujian penerjemahan status, *node* berhasil menerjemahkan data tersebut menjadi aktuasi bagi AC dan *exhaust fan* serta LED indikator yang dapat ditunjukan pada Gambar 4.6, 4.7, 4.8, dan 4.9.

Gambar 4.6 Status AC menyala

Gambar 4.7 Status keduanya non-aktif



Gambar 4.8 Status *exhaust fan* menyala

Gambar 4.9 Status AC dan e*xhaust fan* menyala

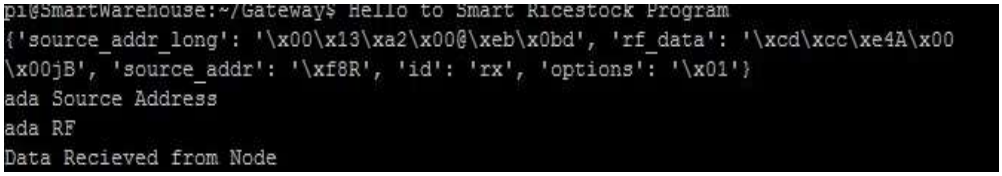
4.4. Pengujian *Gateway*

Sub-bab ini membahas pengujian fungsionalitas dari *gateway* dimulai dari penerimaan data, pengiriman data ke *server,* penerimaan data status, sistem aktuasi, serta pengiriman status ke *node*. Pengujian pada *gateway* dilakukan pada PuTTY agar bisa selalu terlihat proses berjalannya dari *gateway* selama pengujian setiap proses yang berjalan pada *gateway* terus terlihat pada terminal komputer ketika terkoneksi dengan *gateway* melalui *software* PuTTY.

4.4.1. Pengujian pengiriman data ke *server*

Pengujian pengiriman data pada *server* dilakukan dengan PuTTY yang memperlihatkan terminal program pada *gateway*. Sebelum melakukan pengiriman ke *server*, *gateway* harus terlebih dahulu dapat menerjemahkan data yang diterima dari *node* serta memastikan bahwa data yang dikirimkan oleh *node* benar sehingga *server* tidak menerima data yang salah. Pemeriksaan data dilakukan dengan

memastikan bahwa RF *data* dan *Source address long* dimiliki oleh *frame* data yang diterima pada *gateway*. Terminal PuTTY tersebut mencetak tulisan ada RF *data* dan ada *Source address* jika *frame* data tersebut lengkap. Setelah melakukan pemeriksaan, maka RF *data* diterjemahkan sehingga dapat dikirimkan bersama dengan paket data yang dikirimkan ke *server*. Gambar 4.10 Memperlihatkan terminal PuTTY ketika memberikan keterangan terdapat RF data dan *Source address*.



Gambar 4.10 Pemeriksaan data

Pengiriman data ke *server* dilakukan segera setelah proses pemeriksaan tersebut. Pengiriman data tersebut juga memberikan keterangan pada terminal PuTTY di komputer, keterangan yang diberikan adalah *sending data to server.* Keterangan ini mengindikasikan bahwa *gateway* sedang melakukan pengiriman data ke *server*, ketika *gateway* sudah berhasil mengirim data ke *server* maka keterangan yang muncul pada terminal tersebut adalah *sending to success* yang dapat diamati pada Gambar 4.11.

Gambar 4.11 Proses pengiriman data ke *server*

Pada pengujian ini juga diuji data yang diterima pada *server* sama dengan data yang dikirimkan dari *gateway*. Cara untuk melakukan pengujian tersebut adalah dengan membandingkan data yang diterima pada *database* yang terdapat pada *server* dengan data yang dikirimkan dari *gateway*. Perbedaan yang terdapat ada pada nilai angka di belakang koma yang tidak sebanyak yang dikirimkan dari *gateway* namun merupakan angka yang tidak berdampak secara signifikan untuk faktor perkembangbiakan hama gudang. Dari tabel perbandingan yang ditunjukan pada Tabel 4.2, maka dapat dikatakan bahwa data yang diterima pada *server* bernilai sama dengan data yang dikirimkan dari *gateway*.

Tabel 4.2 Perbandingan data pada *gateway* dan *server*

Gateway			*Server*		
Time	*Temp*	*Hum*	*Time*	*Temp*	*Hum*
2016,7, 17, 10, 16, 36, 889854	27.30000114	85.30000305	7/17/2016 10:16	27.3	85.3
2016,7,17,10,22,57,921774	27.39999962	83.5	7/17/2016 10:22	27.4	83.5
2016,7,17,10,29,19,752586	28.30000114	77.40000153	7/17/2016 10:29	28.3	77.4
2016,7,17,10,35,40,749589	28.60000038	74	7/17/2016 10:35	28.6	74
2016,7,17,10,42,2,9186	28.70000076	72.5	7/17/2016 10:42	28.7	72.5
2016,7,17,10,48,23,406975	28.80000114	71.40000153	7/17/2016 10:48	28.8	71.4
2016,7,17,10,54,44,697734	28.80000114	70.59999847	7/17/2016 10:54	28.8	70.6

Pengujian keberhasilan pengiriman data *gateway* ke *server* juga diuji dengan melakukan pengiriman secara terus menerus selama kurang lebih 24 jam pada tanggal 12 Juli 2016 mulai dari jam 9.11 WIB. Dengan menggunakan data

logging pada fungsi Python, didapatkan hasil data bahwa dilakukan pengiriman sebanyak 215 kali, dengan 2 kali terjadi kegagalan. Data *log* pengiriman data oleh *gateway* ke *server* menunjukkan tingkat keberhasilan pengiriman sebesar 99.06% data dan tingkat kegagalan pengiriman sebesar 0.93% data. Kegagalan tersebut terjadi karena tidak terdapat RF data yang dikirimkan dari *node* ke *gateway*. Ketika hal tersebut terjadi maka *gateway* tidak melanjutkan proses penerjemahan serta pengiriman data. Gambar 4.12 dan Gambar 4.13 menunjukan data *logging* keberhasilan pengiriman serta kegagalan pengiriman data ke *server*.

No Source Address		
No RF Data		
Data Recieved from Node	2016-07-12 10:42:06.097562	Data Not Send
Data Recieved from Node	2016-07-12 10:39:48.342635	Sending to Server Success
Data Recieved from Node	2016-07-12 10:53:59.380078	Sending to Server Success
Data Recieved from Node	2016-07-12 10:58:42.106393	Sending to Server Success
Data Recieved from Node	2016-07-12 11:04:58.550604	Sending to Server Success
Data Recieved from Node	2016-07-12 11:11:20.211446	Sending to Server Success
Data Recieved from Node	2016-07-12 11:17:40.976656	Sending to Server Success
Data Recieved from Node	2016-07-12 11:24:02.017080	Sending to Server Success
Data Recieved from Node	2016-07-12 11:30:23.619752	Sending to Server Success

Gambar 4.12 Data *log* pengiriman gagal

Data Recieved from Node	2016-07-12 11:11:20.211446	Sending to Server Success
Data Recieved from Node	2016-07-12 11:17:40.976656	Sending to Server Success
Data Recieved from Node	2016-07-12 11:24:02.017080	Sending to Server Success
Data Recieved from Node	2016-07-12 11:30:23.619752	Sending to Server Success
Data Recieved from Node	2016-07-12 11:36:43.924041	Sending to Server Success
Data Recieved from Node	2016-07-12 11:43:04.771519	Sending to Server Success
Data Recieved from Node	2016-07-12 11:49:26.518884	Sending to Server Success
Data Recieved from Node	2016-07-12 11:55:48.644297	Sending to Server Success
Data Recieved from Node	2016-07-12 12:02:09.786237	Sending to Server Success
Data Recieved from Node	2016-07-12 12:08:31.260562	Sending to Server Success
Data Recieved from Node	2016-07-12 12:14:51.524549	Sending to Server Success
Data Recieved from Node	2016-07-12 12:21:13.055127	Sending to Server Success

Gambar 4.13 Data *log* pengiriman berhasil

4.4.2. Pengujian penerimaan status dari *server* dan sistem peringatan *twitter*

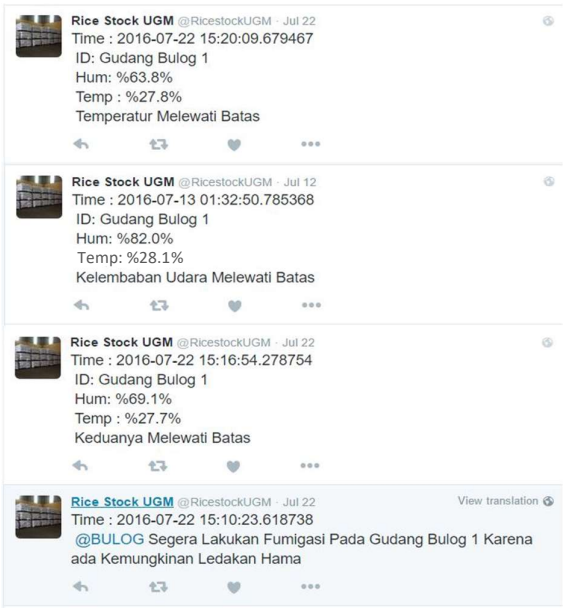
Pengujian penerimaan status pada *gateway* dilakukan dengan PuTTY, pada terminal diberikan keterangan bahwa status telah diterima dari *server*. Gambar 4.14 memperlihatkan bahwa status yang diterima oleh *gateway* diperlihatkan isi status tersebut dan juga memberikan keterangan bahwa *gateway* telah berhasil menerima status.

Gambar 4.14 Penerimaan status dari *server*

Dari status ini, selanjutnya dikirimkan ke *node* melalui pesan *broadcast* dari *coordinator* pada jaringan Zigbee. Status ini juga selanjutnya diterjemahkan oleh *gateway* untuk melakukan peringatan melalui *twitter* kepada pengguna. Pemrosesan data terlebih dahulu mengidentifikasi identitas yang termasuk dalam sebuah paket data status tersebut. Pada 8 karakter awal terlihat bahwa karakter tersebut mengindikasikan identitas dari *node* pengirim, pada angka 0 atau 1 pertama merupakan nilai yang memberikan informasi untuk temperatur sedangkan pada angka 0 atau 1 kedua memberikan informasi untuk kelembaban udara. Jika bernilai 1 artinya temperatur atau kelembaban udara yang dikirimkan oleh *node* bernilai melebihi batas yang diinginkan pengguna.

Gateway menerjemahkan identitas pada status 40eb0b64 sebagai Gudang Bulog 1 dan sisanya diterjemahkan sebagai X. Setelah melakukan penerjemahan tersebut, *gateway* segera melakukan pembacaan data status 0 atau 1 dan segera melakukan eksekusi *tweet* pada akun *twitter* sesuai dengan keadaan status. *Gateway*

juga terus menghitung proses setiap eksekusi penerjemahan status serta menghitung berapa kali terjadinya peringatan melalui *twitter*. Fungsi ini berguna untuk melakukan peringatan jika terjadi peringatan selama 40 menit atau lebih dalam jangka waktu 1 jam. Gambar 4.15 memperlihatkan hasil *tweet* dari *gateway* dengan 4 kondisi yaitu, hanya temperatur melebihi batas atau hanya kelembaban udara melebihi batas atau keduanya melebihi batas dan ketika terjadi peringatan selama 40 menit atau lebih dalam 1 jam. Format *tweet* berawal dari waktu, identitas, nilai kelembaban udara, nilai temperatur, dan keterangan dari status.



Gambar 4.15 *Tweet* peringatan

4.5. Pengujian *Server*

Pengujian *server* dilakukan dengan melakukan pengamatan pada *web* yang telah terhubung dengan sistem jaringan pengendalian dan pemonitoran gudang penyimpanan ini. *Server* selalu melakukan pembaharuan pada setiap waktu

pengguna melakukan *submit* batas yang diinginkan serta ketika *node* mengirimkan data ke *gateway.* Pengujian ini mengamati data yang dikirimkan oleh pengguna maupun sistem serta melakukan pengujian data batas dengan data sistem. Pengujian *server* juga memperlihatkan hasil sederhana dari sistem informasi yang telah dibuat.

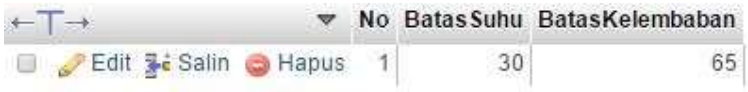
4.5.1. Pengujian pengambilan data batas pengguna

Pengujian pengambilan data batas dilakukan bertujuan untuk memastikan bahwa data yang dimasukan oleh pengguna direkam oleh *database* sehingga ketika data yang masuk dari sistem dapat segera dilakukan pengujian status di *server*. Pada saat memasukan data di *web*, pengguna harus menekan tombol *submit* agar nilai yang diisikan pada kolom batas temperatur dan batas kelembaban masuk ke *database*. Gambar 4.16 memperlihatkan *web* tempat pengguna dapat mengakses dan memasukan nilai batas yang diinginkan.

Gambar 4.16 Tampilan Memasukkan Batas

Setelah memasukan data tersebut, maka dilakukan pengamatan pada *database* yang dimiliki oleh *server*. Dengan mengakses

http://10.13.247.248/phpmyadmin/ dan masuk ke *database smart*_ricestock dan tabel *treshold*. Pengamatan menghasilkan bahwa data yang dimasukan oleh pengguna sama dengan nilai batas yang dimasukan oleh pengguna. Nilai yang dimasukan oleh pengguna harus bertipe *integer*, ketika nilai data yang dimasukan oleh pengguna memiliki koma dibelakang bilangan *integer*, maka sistem pada *web* menolak angka yang dimasukan dan memberikan perintah kepada pengguna untuk memasukan data *integer*. Gambar 4.17 menunjukan data yang diterima di *database* ketika pengguna berhasil memasukan data ke *server*.



Gambar 4.17 Batas pada *database*

4.5.2. Pengujian data

Pengujian data bertujuan untuk melakukan pengujian penentuan status. Pengujian ini dilakukan dengan melihat kolom status pada *server* dan menganalisis keseuaian status yang diberikan oleh *server* bernilai benar dengan yang diinginkan oleh pengguna. Dengan pengujian batas temperatur 30 ˚C dan batas kelembaban udara 65 % dilakukan pengamatan pada status yang dihasilkan oleh *server.* Status yang diharapkan adalah bernilai 1 ketika data yang diberikan oleh sistem melewati batas yang diinginkan dan bernilai 0 jika sebaliknya.

Pada pengujian status ini didapatkan bahwa algoritma yang digunakan untuk melakukan pengujian batas pada temperatur dan kelembaban udara yang dikirimkan selalu menghasilkan status yang sesuai dengan ketentuannya. Gambar 4.18 memperlihatkan pencetakan nilai status dari hasil pengujian data pada *server*

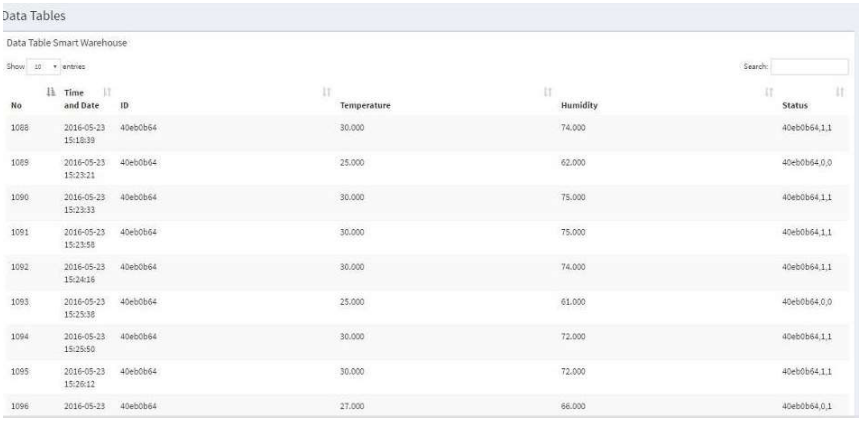
di 4 keadaan, kelembaban udara melebihi batas, temperatur melebihi batas, keduanya melebihi batas dan tidak ada yang melebihi batas yang diinginkan oleh pengguna.

Temperature	Humidity	Status
36.000	65.000	40eb0b64,1,1
35.000	62.000	40eb0b64,1,0
26.700	69.400	40eb0b64,0,1
25.000	62.000	40eb0b64,0,0

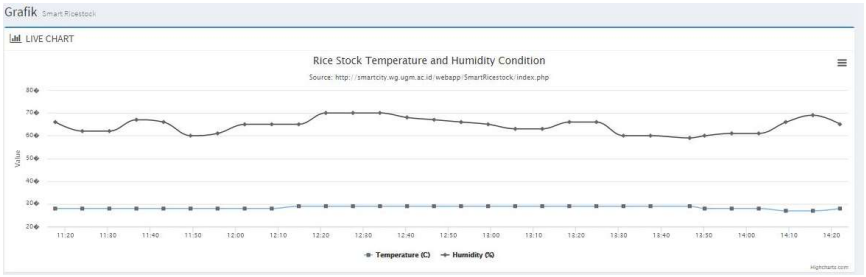
Gambar 4.18 Hasil pengujian data pada *server*

4.5.3. Penampilan sederhana tabel dan grafik pada *web*

Penampilan data tabel dan grafik sederhana pada *web* bertujuan untuk melihat hasil olahan data *server* yang disediakan bagi pengguna. Informasi ini berguna untuk pengguna dapat melihat secara langsung melalui *web* sehingga pengguna dapat dengan mudah melihat informasi secara *realtime. Web* ini dapat diakses melalui alamat *smart*city.wg.ugm.ac.id/*web*app/*Smart*Ricestock/index.php. Gambar 4.19, Gambar 4.20, dan Gambar 4.21 menunjukan tampilan *web* yang ada pada sistem informasi sederhana.



Gambar 4.19 Tampilan tabel pada web



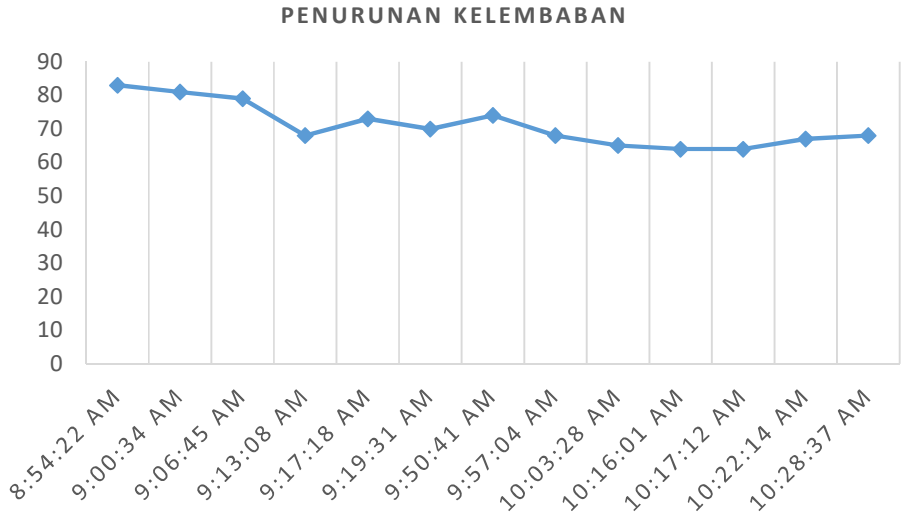
Gambar 4.20 Tampilan grafik web

Gambar 4.21 Tampilan web memasukkan batas

4.6. Pengujian Simulasi

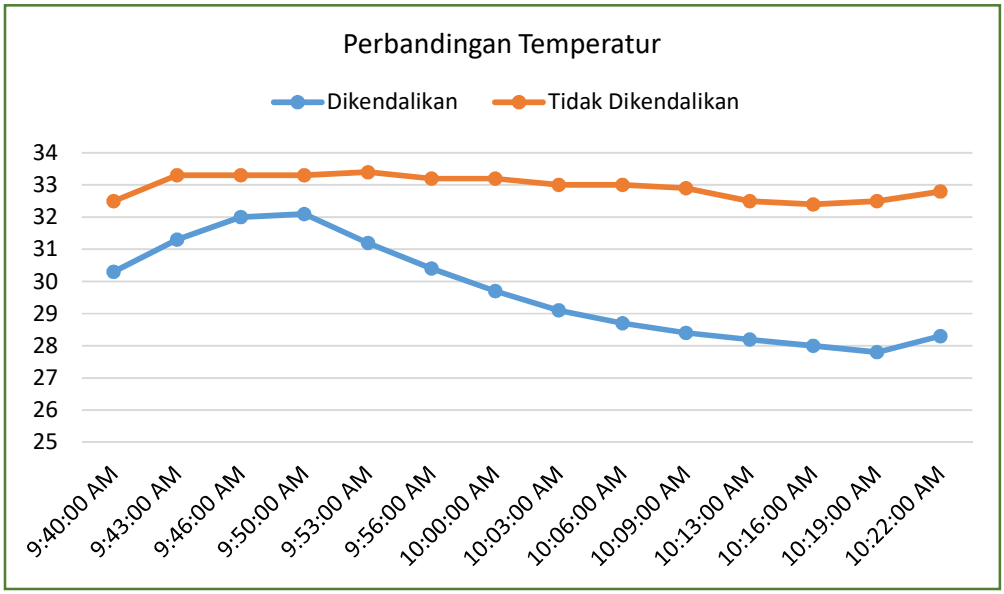
Pengujian simulasi ini bertujuan untuk menguji fungsionalitas sistem pengendalian dan pemonitoran ini. Sistem diharapkan dapat mengendalikan dan memonitor keadaan temperatur dan kelembaban udara pada gudang penyimpanan pascapanen secara *realtime.* Untuk pengujian penurunan temperatur dibantu dengan penggunaan lampu pijar sebagai stimulasi untuk menaikan suhu dalam prototipe gudang, sedangkan untuk pengujian penurunan kelembaban bergantung dengan keadaan sekitar.

Pengujian simulasi untuk melihat pengaruh alat terhadap temperatur dan kelembaban udara dilakukan dengan mengambil data ketika terjadi perubahan temperatur dan kelembaban udara. Gambar 4.22 dan 4.23 memperlihatkan hasil pengujian simulasi dari sistem.



Gambar 4.22 Grafik penurunan kelembaban udara

Gambar 4.22 memperlihatkan fenomena penurunan kelembaban yang terjadi ketika sistem dijalankan untuk mengendalikan dan memonitor keadaan dalam prototipe gudang. Dengan demikian untuk proses pengendalian kelembaban udara sistem dapat dinyatakan berhasil.



Gambar 4.23 Grafik perbandingan temperatur

Gambar 4.23 adalah grafik hasil pengujian perbandingan suhu antara di dalam prototipe gudang yang dibuat, pengukuran suhu diluar prototipe dilakukan dengan sensor yang sama. Dari kedua grafik tersebut, dapat dikatakan proses pengendalian temperatur dan kelembaban udara pada sistem berjalan dengan baik. Namun sistem ini juga memiliki tindakan preventif jika nilai temperatur dan kelembaban udara yang diinginkan oleh pengguna tidak dapat tercapai dalam jangka waktu tertentu, sistem memberikan peringatan khusus.

4.7. Kelebihan dan Kekurangan Sistem

Sistem yang telah dikembangkan sudah mampu untuk melakukan pengendalian dan pemonitoran temperatur dan kelembaban udara pada gudang penyimpanan pascapanen dengan mengirimkan data temperatur dan kelembaban udara secara *realtime* dan melakukan pengujian data sehingga menghasilkan status yang dibutuhkan oleh *node* dan *gateway*. Kelebihan sistem yang dikembangkan adalah sebagai berikut :

a. Hanya membutuhkan 1 *gateway* untuk beberapa gudang.
b. *Server* dapat memberikan serta mengambil informasi dari pengguna untuk melakukan pengendalian dan pemonitoran gudang pascapanen melalui internet.
c. Pengendalian pada jaringan sensor *node* sudah dapat diaplikasikan untuk beberapa buah *node* dengan sistem pemeriksaan identitas pada status yang diterima.
d. *Gateway* dapat memberikan peringatan kepada pengguna jika terjadi adanya potensi ledakan hama pada gudang pascapanen.

Kekurangan sistem yang selanjutnya dapat diperbaiki atau disempurnakan adalah sebagai berikut:

a. Masih belum bisa mendeteksi hama gudang secara langsung.
b. Sistem bergantung penuh dengan kualitas koneksi internet.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, didapatkan beberapa kesimpulan sebagai berikut :

a. Perancangan sensor *node* berhasil mengirimkan data temperatur dan kelembaban udara ke *gateway* serta menerima dan menerjemahkan perintah aktuasi dari *gateway.*

b. Perancangan *gateway* berhasil mengirimkan perintah aktuasi ke *node*, melakukan pengiriman data ke *server*, dan memberikan peringatan kepada pengguna untuk melakukan pengendalian dan pemonitoran gudang penyimpanan pascapanen.

c. Perancangan *server* berhasil menerima data dari sistem maupun pengguna untuk menguji temperatur dan kelembaban udara sistem terhadap keinginan pengguna.

d. Sistem memiliki keberhasilan 99% untuk melakukan pemonitoran dan pengendalian berdasarkan pengujian dari *gateway*, *node*, dan *server.*

5.2. Saran

Berdasarkan penelitian yang telah dilakukan, saran yang dapat digunakan untuk penelitian selanjutnya adalah sebagai berikut:

a. Dibutuhkan sensor yang dapat langsung mendeteksi banyaknya hama pada gudang penyimpanan pascapanen.
b. Melakukan integrasi untuk menentukan lama waktu yang dibutuhkan untuk mengirim peringatan pada *web.*
c. Melakukan pengembangan karakteristik pengendalian AC dan *exhaust fan* melalui sensor IR.
d. Mengembangkan penampilan *web* untuk *smartphone*.
e. Melakukan pengembangan dengan *gateway* menggunakan *mode*m untuk koneksi internet dan daya melalui *solar panel* pada *node.*

DAFTAR PUSTAKA

5hertz.com. (2016). Desbloqueando un XBee (Unbrick). Diambil kembali dari: http://5hertz.com/tutoriales/?p=46

Anggara, A. W., & Sudarmaji. (2008). *Hama Pascapanen Padi dan Pengendaliannya.*

Arduino. (2014). Arduino Uno. Diambil kembali dari: http://www.arduino.cc/en/Main/ArduinoBoardUno

Artofciruits.com. (2016). Opto-isolated 2 Channel 5V Relay Module. Diambil kembali dari: http://artofcircuits.com/product/opto-isolated-2-channel-5v-relay-module

Baldasarri, N., Martini, A., Cavicchi, S., & Baronio, P. (2005). Effect of Low Temperature on Adult Survival and Reproduction of *Rhyzopertha dominica. Bulletin of Insectology 58 (2): 131-134, 2005 ISSN 1721-8861.*

Bell, Charles. (2013). *Beginning Sensor Networks with Arduino and Raspberry Pi.*

Beritadaerah.co.id. (2016). Stok Beras Bulog pada Musim Kemarau. Diambil kembali dari http://beritadaerah.co.id/2014/09/25/stok-beras-bulog-pada-musim-kemarau/

Digi. (2016). XBee RF Solutions. Diakses Februari, 2016 dari Website Digi: http://www.digi.com/products/xbee-rf-solutions.

Electronicsdesign.com. (2016). Engineering Essentials: IoT Standards and Frameworks. Diambil kembali dari: http://electronicdesign.com/iot/engineering-essentials-iot-standards-and-frameworks

Electroschematics.com. (2016). Arduino DHT22 (AM2302) Tutorial + Library. Diambil kembali dari: http://www.electroschematics.com/11291/arduino-dht22-am2302-tutorial-library/

Haryadi. (2006). *Teknologi Pengolahan Beras*. Gadjah Mada University Press, Yogyakarta.

Jia, J., Kuang, J., He, Z., & Mu, Y. (2009). Design of Monitor System for Grain Depots Based on Wireless Sensor Network. *International Conference on Mechatronics and Automation. IEEE*

Modules, Xbee Xbee-PRO RF. (2009). XBee / XBee-PRO RF Modules.

Pitaloka, A. L. (2012). Gambaran Beberapa Faktor Fisik Penyimpanan Beras, Identifikasi Dan Upaya Pengendalian Serangga Hama Gudang (Studi Di Gudang Bulog 103 Demak Sub Dolog Wilayah I Semarang).

Osman, N. B., Wright, V. F., & Mills, R. B. (1983). The Effects of Rearing Temperatures on Certain Aspects of the Biology of Corcyra cephalonica (Stainton), the rice moth. *Department of Entomology Kansas State University. Manhattan, Kansas.*

Pixabay.com. (2016). Gudang Silo Pertanian Musim Panas. Diambil kembali dari: https://pixabay.com/id/gudang-silo-pertanian-musim-panas-963071/

Raspberrypi.org. (2016). RP3 not reading PN532 (RFID) over UART using node-serialport. Diambil kembali dari: https:// www.raspberrypi.org /forums/ viewtopic.php?t=144858&p=955297

Riaz, T., Shakoori, F. R., & Ali, S. S. (2014). Effect of Temperature on the Development, Survival, Fecundity, and Longevity of Stored Grains Pest, Trogoderma granarium. *Pakistan J. Zool.,* vol. 46(6), pp. 1485-1489.

Ritvaldi, M. R. (2016). Perancangan Sistem Deteksi Kendaraan Menggunakan Kombinasi Sensor Ultrasonik dan Medan Magnet untuk Mendukung Framework Smart Parking.

Roscoe, J.T. (1975) Fundamental Research Statistics for the Behavioural Sciences, 2nd edition. *New York: Holt Rinehart & Winston.*

Setyolaksono, P. (2013). *Ekologi Hama Pascapanen (Hama Gudang)*. Diambil kembali dari: http://ditjenbun.pertanian.go.id/bbpptpambon/berita-177-ekologi-hama-pascapanen-hama-gudang-.html

Singh, S., & Prakash, S. (2015). Effect of Temperature and Humidity on the Culture of Tribolius castaneum, Herbst (Coleoptera: Tenebrionidae) in the laboratory. *International Journal of Scientific and Research Publications, Volume* 5, *Issue* 7.

Sitinjak, K. (1986). *Pasca Panen*. Fakultas Pertanian USU. Medan.

Thehackernews.com. (2016). Raspberry Pi 3 – New $35 MicroComputer with Builth-in Wi-fi and Bluetooth. Diambil kembali dari: http://thehackernews.com/2016/02/raspberry-pi-3-microcomputer.html

Wu, Li-hua, Yao, Jia-yu, Zhang, Hai-yang, & Lin, Xi-rong. (2014). Design of an Intelligent Granary Monitoring System. *International Conference on Future Generation Communication and Networking.*

Zafalon, R. (2013). Smart System Design: Industrial Challenges and Perspectives. *International Conference on Mobile Data Management*. IEEE.

Zhou, H., Zhang, F., Liu, J., & Zhang, F. (2009). *A Real-time Monitoring and Controlling System for Grain Storage with ZigBee Sensor Network.* IEEE

LAMPIRAN