INFORMATION SYSTEM DEVELOPMENT
VOLUME X NO.X (20XX) | p70-78
e-ISSN 2528-5114 | p-ISSN 2477-863X
http://dx.doi.org/10.19166/xxxx

SMART BELL BERBASIS ARDUINO DAN WEB SEBAGAI APLIKASI IoT PADA SISTEM KEAMANAN RUMAH

Dale Watson[1], Hendra Tjahyadi[2]

[1]Sistem Komputer, Universitas Pelita Harapan, Tangerang, Indonesia [Times New Roman 11 normal]
[2]Magister Informatika, Universitas Pelita Harapan, Tangerang, Indonesia [Times New Roman 11 normal]
E-mail: [1] dale.watson@gmail.com, [2] hendra.tjahyadi@uph.edu

Abstract – For a porpose of home security improvement, in this paper, a smart bell system is designed and realized. The system uses a passive-infrared sensor to detect motion and activate the system to logged detection time to database, sent notification, and command a webcam to take pictures and record video. An Arduino microcontroller is used to process data that will be presented on web pages which are then converted into Progressive Web Applications. Node.js is used as the platform to creare a server, and MongoDB is employed as database for time detection. For communication system protocol between the database and web application pages the REST-API method is used. During the detection process, the system will send a notification using webpush with the Firebase Cloud Messanging framework. The system was tested to detect, record videos, create GIFs, write to databases, and send notifications into web applications. The experiments shows a success of 81.25% for displaying videos and of 93.75% for displaying GIFs.

***Keywords**: Passive Infra Red sensor, Progressive Web Application, Arduino*

Abstrak – Dengan tujuan untuk meningkatkan sistem keamanan rumah, dalam tulisan ini didesain dan direalisasikan sebuah sistem smart bell. Sistem menggunakan sensor passive-infrared untuk mendeteksi gerakan yang akan mengaktifkan sistem untuk mencatatkan waktu ke dqlam basis data, mengirimkan notifikasi kepada pemakai dan memerintahkan webcam untuk mengambil gambar dan merekam video. Sebuah mikrokontroler Arduino dipakai untuk mengolah data yang disajikan di dalam halaman web yang selanjutnya diubah menjadi Progressive Web Application. Node.js dipakai sebagai platform untuk membuat server, dan MongoDB dipakai sebagai basis data untuk mencatat waktu pendeteksian. Sementara untuk protokol sistem komunikasi antara basis data dan halaman aplikasi web digunakan metode REST-API. Saat proses pendeteksian berlangsung, sistem akan mengirim notifikasi menggunakan webpush dengan framework Firebase Cloud Messanging. Sistem diuji untuk mendeteksi, merekam video, membuat GIF, menulis ke basis data, dan mengirim notifikasi ke dalam aplikasi web. Hasil pengujian untuk menampilkan video menunjukkan persentase keberhasilan sebesar 81.25% sementara pengujian untuk menampilkan GIF menunjukkan persentase keberhasilan sebesar 93.75%

Kata Kunci: *sensor passive infra red, Progresive Web Application*

PENDAHULUAN

Internet of Things (IoT) merupakan salah satu teknologi yang berkembang di era Industri 4.0 yang dicirikan dengan digitalisasi. Digitalisasi telah memudahkan interkoneksi diantara "*things*" seperti mobil, bangunan, sensor dan juga manusia sehingga melahirkan konsep "*smar thingst*" seperti *smart phone*, *smart car*, *smart home* sampai pada *smart city*.

Salah satu aspek penting dalam *smart home* dan *smart city* adalah pemantauan keamanan. Diharapkan pemantauan keamanan bisa terjadi secara otomatis tanpa memerlukan intervensi manusia yang intensif. Beberapa penelitian untuk mewujudkan keamanan secara otomatis ditemukan dalam literatur. Saranu *et al.* [1] membuat alat pendeteksi pencuri secara otomatis dengan menggunakan *Passive Infra Red* (PIR) sensor dan Raspberry Pi. Kumar dan Ramesh [2] membuat sistem keamanan pintar berbasis Raspberry Pi dan memanfaatkan *cloud* untuk penyimpanan data. Tina *et al.* [3] merealisasikan pengaturan keamanan dan pencahayaan secara otomatis berbasiskan Arduino. Sahoo dan Pati [4] membuat sistem keamanan rumah dengan menggunakan PIR sensor, ZigBee dan ESP8266 untuk mengirimkan data kepada ThingSpeak *server*. Said *et al.* [5] memanfaatkan Arduino Mega dan PIR sensor serta sensor *microwave* untuk sistem keamanan yang mampu memberikan peringatan kepada pemilik rumah melalui SMS.

Dari penelitian terdahulu terlihat sebagian besar belum memanfaatkan internet dan *smart phone* secara optimal untuk sistem keamanan rumahnya. Padahal saat ini hampir semua *smart phone* telah terkoneksi dengan internet dan sebagian besar orang mengakses internet melalui *smart phone* secara intensif sehingga notifikasi melalui *smart phone* akan sangat efektif. Untuk memanfaatkan internet dan *smart phone* secara lebih optimal dalam membangun sistem keamanan secara

otomatis maka penelitian yang dibahas pada tulisan ini diusulkan.

Sistem yang akan dibangun disebut sebagai *smart bell*. Sistem ini akan mendeteksi kehadiran seseorang dan mengirimkan notifikasi kepada perangkat pengguna yang memiliki *web browser*. Sistem juga akan mencatat waktu pendeteksian yang dicatat dalam database serta memberikan visual berupa video atau gambar pada *smart phone* pengguna.

Selanjutnya diskusi pada tulisan ini akan membahas mengenai metode penelitian yang dipakai, cara kerja serta komponen-komponen yang dipakai mencakup perangkat keras dan perangkat lunak, perancangan dan realisasi dari sistem yang lebih fokus pada perancangan perangkat lunak, pengujian dan pembahasan hasil, serta ditutup dengan suatu kesimpulan dan saran untuk penelitian lebih lanjut.

METODOLOGI PENELITIAN

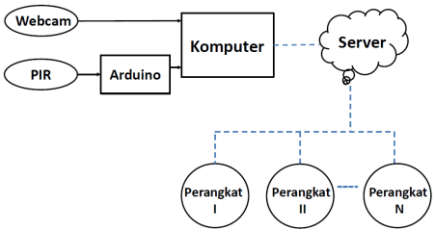
Metodologi yang digunakan dalam menyelesaikan persoalan yang menjadi masalah penelitian adalah studi literatur dan studi eksperimen. Studi literatur mencakup perangkat keras dan perangkat lunak diantaranya adalah konsep dasar komunikasi antara Arduino dan *server* Node.js, HTML, CSS, JavaScript, *Web starter-kit* dan *NPM-package.* Untuk studi eksperimen dilakukan eksperimen untuk menguji kinerja dari sensor dan pengiriman data ke dalam Server Node.js, komunikasi antar webcam dengan *server* Node.js, komunikasi antara *server* Node.js dengan aplikasi web, kinerja notifikasi di dalam *Progressive Web Application* serta uji keseluruhan sistem untuk mengukur keberhasilan implementasi sistem.

KOMPONEN DAN CARA KERJA

Penjelasan mengenai cara kerja sistem dibagi menjadi dua bagian yaitu perangkat keras dan perangkat lunak dari sistem.

Perangkat Keras Sistem

Diagram blok dari sistem *smart bell* ditunjukkan pada Gambar 1. Perangkat keras sistem terdiri dari sensor PIR dan webcam, sebuah mikrokontroller Arduino, dan sebuah komputer. Sistem akan dapat diakses oleh perangkat-perangkat luar yang bisa mengakses *web browser* melalui koneksi dengan *server*. Sistem secara terus menerus mendeteksi kehadiran suatu obyek melalui deteksi gerakan oleh sensor PIR. Saat sensor PIR mendeteksi adanya obyek maka sistem akan mencatat waktu pendeteksian, mengaktifkan webcam untuk melakukan perekaman, dan mengirimkan notifikasi.



Gambar 1. Diagram Blok Sistem

Passive Infra Red Sensor

Passive Infra Red sensor merupakan suatu sensor elektronik yang dipakai untuk mengukur jumlah dari radiasi cahaya dari suatu obyek yang berada pada daerah kerjanya atau dikenal sebagai *Field of View* (FoV) [6], [7]. Perbedaan atau perubahan dari jumlah radiasi cahaya yang terukur bisa digunakan untuk mengubah gerakan menjadi sinyal listrik.

Pada penelitian ini digunakan modul PIR sensor HC-SR501. Alat ini dapat mendeteksi gerakan dalam area kerucut 110 derajat dengan jangkauan dari 3 hingga 7 meter. Alat ini memiliki dua slot didalamnya yang terbuat dari bahan khusus yang peka terhadap inframerah. Ketika sensor dalam posisi *idle*, kedua slot mendeteksi jumlah inframerah yang sama yang dipancarkan dari ruangan atau dinding. Saat suatu obyek yang memiliki panas tubuh memasuki FoV sensor maka akan terjadi perubahan differensial positif antara kedua slot, dan akan terjadi perubahan differensial negatif saat obyek keluar dari area kerja sensor. Perubahan sinyal tersebut menjadi indikasi adanya gerakan yang terdeteksi. Alat ini membutuhkan setidaknya satu menit sebagai response transient. Selama jangka ini, alat dapat mengeluarkan sinyal deteksi palsu [8].

Arduino Uno

Arduino Uno adalah sebuah *microcontroller* yang berbasiskan ATmega328P dan *open-source*. Dilengkapi dengan, 14 pin digital *input*/*output* dimana enam dari pin tersebut dapat digunakan sebagai keluaran *Pulse-Width-Modulation (PWM)*, enam *input analog*, sebuah koneksi *Universal Serial Bus (USB)*, kabel *power*, dan tombol *reset*. Arduino menggunakan bahasa pemrograman tersendiri dan memiliki kemiripan dengan bahasa pemrograman C. Pemrograman Arduino dilakukan dengan Arduino IDE. IDE ditulis menggunakan bahasa pemrograman Java, yang dapat berjalan di sistem operasi Linux, Windows, maupun Mac OS [9].

Perangkat Lunak Sistem

Di dalam komputer terdapat tiga *server* yang akan diaktifkan untuk menjalankan sistem yang dirancang. Ketiga *server* tersebut adalah *server input*, *server* basis data dan *server* API yang akan dijalankan melalui koneksi lokal. Melalui *server-server* tersebut notifikasi akan diterima oleh perangkat-perangkat yang diinginkan untuk membuka aplikasi web. Selain untuk membuat ketiga *server* tersebut, perangkat lunak juga dipakai untuk

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

beberapa fungsi lainnya yaitu membangun aplikasi web, dan mengubah video menjadi gambar. Diskusi berikut menjelaskan secara singkat dari perangkat lunak yang dipakai untuk merealisasikan sistem.

Service Worker

Service worker adalah suatu *proxy* yang dapat diprogram untuk berjalan di belakang halaman *browser*, dan tidak membutuhkan interaksi dari pengguna untuk menjalankan tugasnya. Dengan begitu *service worker* tetap berjalan walaupun halaman webnya tidak terbuka. *Service worker* memiliki kemampuan untuk mencegat, menangani HTTP-*requests* dan juga meresponnya dalam berbagai cara, juga dapat menangani *network requests*, *push notifications*, perubahan konektivitas dan masih banyak lagi. Program ini tidak dapat mengakses *Document Object Module* (DOM) untuk mengubah halaman *browser* tetapi melalui API untuk *fetch* dan *cache*, sehingga dapat menyimpan semua sumber daya yang statis dan secara otomatis mengurangi permintaan jaringan dan meningkatkan kinerja halaman [10].

Node,js

Node.js merupakan *platform* yang dibuat dari Chrome JavaScript *runtime* yang beguna untuk membangun aplikasi jaringan dengan cepat dan *scalable*. Node.js menggunakan model *event-driven*, *non-blocking* I/O sehingga Node.js menjadi ringan dan efisien, tepat untuk pembuatan aplikasi *real-time*. Node.js diprogram dengan bahasa pemrograman JavaScript [11].

JavaScript

JavaScript adalah skrip yang beroperasi di belakang *web browser*, dengan memanfaatkan *engine* yang tertanam di dalamnya. Fitur - fitur yang ada pada JavaScript diantaranya adalah, mengendalikan interaksi antara halaman web dengan pengguna, seperti mengendalikan menu, tombol, dan *toolbars*, memvalidasi *web form* sebelum *browser* mengirimkan ke *server*. Secara umum JavaScript mengurangi proses kinerja *server*, karena sebagian proses telah dikerjakan oleh *web browser*.

Express.js

Express.js adalah *framework* untuk aplikasi web yang berjalan menggunakan Node.js, diluncurkan sebagai perangkat lunak yang *open-source* dengan MIT license. Didesain untuk membangun aplikasi web dan API, telah menjadi standar *web framework* untuk Node.js.

API

Application Program Interface (API*)* adalah seperangkat rutinitas, protokol, dan alat untuk membangun aplikasi perangkat lunak. Pada dasarnya, API menentukan bagaimana komponen perangkat lunak harus berinteraksi. API digunakan saat memprogram komponen antarmuka pengguna grafis (GUI).

Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) yang sebelumnya lebih dikenal sebagai *Google Cloud Messaging* (GCM) adalah *mobile notification service* tanpa biaya yang dirancang oleh Google, memungkinkan developer untuk mengirim *messages* antar *server* dan aplikasi klien tanpa biaya. Ini termasuk *downstream messages* dari server ke aplikasi klien, dan *upstream messages* dari aplikasi klien ke *server*.

Firmata

Firmata adalah sebuah protokol komunikasi antar perangkat lunak komputer dengan pengendali mikro. Protokol dapat diimplementasi di dalam *firmware* dari sebuah pengendali mikro atau pada paket perangkat lunak (*software package)* [12].

FFmpeg

FFmpeg adalah proyek perangkat lunak untuk mengkonversi video dan audio. Mempunyai *library* dan program untuk mengatasi data multimedia. FFmpeg membaca jumlah dari input ”*file* ” yang berubah-ubah, yang ditentukan dari pilihan“-i”, dan menulis ke sejumlah output “*files*” yang berubah-ubah [13].

Johhny-Five

Johhny-Five adalah protokol untuk Firmata, dan *platform* dari JavaScript Robotics dan IoT yang *open-source*. Johnny-Five berfokus pada pengiriman ke dalam API yang secara konsisten mendukung semua platform perangkat keras [14].

NoSQL

NoSQL adalah tipe basis data yang dapat disimpan tanpa menggunakan skema atau dalam bentuk yang bebas. Salah satu jenis model dari database NoSQL adalah menyimpan data berupa dokumentasi basis data. Data yang disisipkan akan disimpan dalam bentuk struktur JSON atau dokumen, dimana datanya dapat berupa apapun dari bentuk integer, bentuk string, hingga dalam bentuk teks bebas [15].

MongoDB

MongoDB adalah basis data yang *open-source* menggunakan model data yang berorientasi dokumen. Seperti basis data NoSQL yang lainnya, MongoDB mendukung desain skema yang dinamis, memungkinkan dokumen dalam *collections* memiliki berbagai bidang dan struktur. Basis data menggunakan format penyimpanan dokumen dan pertukaran data yang disebut BSON, yang menyediakan representasi biner dokumen seperti JSON [16].

Perancangan Perangkat Lunak

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

Bagian ini menjelaskan rancangan perangkat lunak dengan menjelaskan penggunaan setiap *server*, penggunaan *library*, dan *framework* yang dibutuhkan. Pembangunan setiap *server* dibuat dengan menggunakan Node.js. Perancangan perangkat lunak mencakup:
(i) perancangan *server input*. Server ini merupakan sistem pemicu untuk pengiriman notifikasi, tempat berjalannya kendali sensor dan webcam. Sebelum sensor mengirim data ke dalam *server input*, Arduino akan menerima input dengan menggunakan program Firmata dan dijalankan dalam komputer dengan menggunakan program Johnny-Five. Kemudian *webcam* mengirim data melalui program FFmpeg.
(ii) Perancangan basis data. Basis data dipakai untuk menyimpan data pengunjung dan identitas perangkat. Data pengunjung adalah informasi-informasi yang dikirim dari *server input* berupa waktu pendeteksian terjadi. Sedangkan identitas perangkat dikirim melalui s*erver* API. Adapun yang dimaksud dengan identitas perangkat adalah nomor serial yang dibuat oleh *Firebase Cloud Messaging* sebagai bentuk dari identitas perangkat yang dipakai oleh para pengguna. Hal tersebut diperlukan dalam sistem pengiriman notifikasi, karena dalam sistem notifikasi aplikasi ini memerlukan identitas perangkat sebagai alamat untuk pengiriman notifikasi. (iii) Perancangan halaman aplikasi web. Pembuatan halaman aplikasi web dibangun dengan HTML, CSS, dan JavaScript untuk penggunaan manipulasi elemen HTML DOM. Pada HTML DOM terdapat *event-event listening* untuk dapat mengubah elemen-elemen dalam HTML. (iv) Perancangan *service worker*. *Service worker* diperlukan untuk melakukan sinkronisasi halaman aplikasi web secara otomatis disaat pengguna masuk dan untuk keperluan pengiriman notifikasi. (v) Perancangan notifikasi. Sistem notifikasi menggunakan layanan *Firebase Cloud Messanging* sebagai proses pengiriman notifikasi ke dalam aplikasi web. Untuk dapat menggunakan layanan tersebut dibutuhkan *server key* di dalam *server* aplikasi web. Aplikasi klien akan mendaftarkan perangkat kepada *server* API untuk dapat menerima notifikasi. Dalam *Server* API, sistem akan menunjukkan *server-key* sebagai izin penggunaan layanan notifikasi kepada *Firebase Cloud Messanging*. Untuk mendapatkan *server-key* yang ditaruh dalam *Server* API dan *sender-id* yang ditaruh dalam aplikasi web perlu dilakukan pembuatan akun *Firebase Cloud Messanging* terlebih dulu.
(vi) Perancangan server API. Pada perancangan ini akan dibangun s*erver* API yaitu *web framework* untuk *server* aplikasi web dan protokol API yang mengatur jalur sistem keluar masuk informasi data yang diperlukan pada aplikasi web. Pembuatan *web framework* untuk *server* menggunakan *package manager* dari Node.js yaitu Express.js. *Server* API ini tidak hanya bertugas untuk menyimpan dan menghapus identitas perangkat tetapi juga mengontrol cara kerja pengiriman notifikasi.

Diagram Alir Sistem

Diagram alir dari sistem yang menjelaskan cara kerja sistem untuk dapat mendeteksi gerakan melalui sensor hingga aplikasi web dapat menerima notifikasi ditunjukkan pada Gambar 2. Pada tahap awal saat sistem berjalan, sistem akan mengaktifkan s*erver* Input yang mengontrol PIR sensor untuk mempersiapkan pendeteksian kemudian saat sistem mendeteksi suatu gerakan, maka sistem akan melakukan pencatatan waktu dari pendeteksian dan catatan waktu akan dikirim ke dalam s*erver basis data* untuk disimpan, pengiriman ke dalam *server basis data* melalui URL yang telah dijalankan oleh s*erver basis data* tersebut.

Server basis data berisi semua catatan waktu pendeteksian dan data dari identitas perangkat. Setelah proses pencatatan waktu pendeteksian maka sistem akan melakukan pengiriman notifikasi dari *server* input melalui URL ke klien, dimana *server* API adalah sebagai klien yang dituju. Saat *server* API menerima pesan tersebut, *server* API akan mengambil data identitas perangkat yang telah tersimpan di dalam *server basis data*. *Server* API akan memproses pengiriman pesan notifikasi menggunakan *web push* ke dalam *web browser* atau aplikasi web klien sesuai dengan identitas perangkat yang dituju, sehingga setiap perangkat yang membuka aplikasi web dan sesuai dengan identitas perangkat yang telah terdaftar akan mendapatkan notifikasi.

Bagi yang membuka aplikasi web tanpa terdaftar maka tidak akan menerima notifikasi. Setelah melakukan pengiriman notifikasi maka sistem mengaktifkan *webcam* untuk merekam selama waktu yang ditentukan dalam program yaitu 5 detik, video yang direkam tersimpan dalam format mp4, kemudian sistem akan mengkonversi menjadi format webm. Video akan tersimpan dalam direktori aplikasi web sehingga secara otomatis nantinya akan mengganti video di dalam *server* aplikasi web yang sedang berjalan.

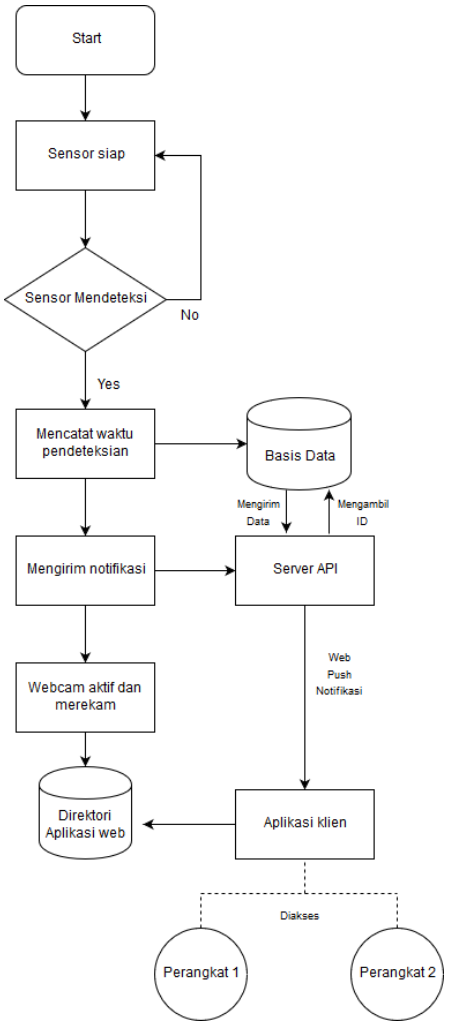
IMPLEMENTASI

Implementasi perangkat keras ditunjukkan pada Gambar 3. Terlihat rangkaian sangat sederhana hanya terdiri dari sensor PIR, webcam, Arduino Uno dan sebuah laptop. Sementara untuk implementasi perangkat lunak, sesuai dengan perancangan akan meliputi implementasi *server input*, implementasi basis data, implementasi halaman aplikasi web, implementasi *service worker*, implementasi notifikasi dan implementasi *server* API.

Implementasi *Server Input*

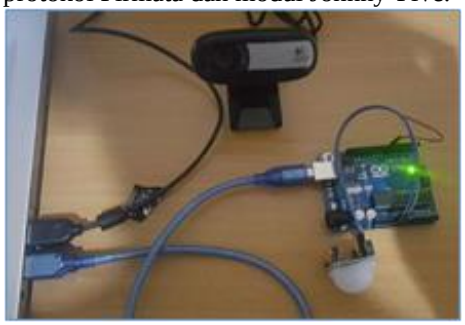
Langkah-langkah implementasi *server input* adalah: (i) instalasi Node.js. Pada penelitian ini digunakan sistem operasi Linux Manjaro dan Ubuntu, dan dalam sistem operasi ini instalasi dilakukan melalui terminal-line. Pengunduhan Node.js dilakukan dengan menuliskan sintaks "*Sudo pacman-Syu Nodejs*".

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>



Gambar 2. Diagram alir sistem

(ii) instalasi *package manager* ke dalam sistem dengan menuliskan "*npm install*" pada *terminal line*. (iii) instalasi Arduino ke dalam Node.js. Langkah ini dilakukan agar sensor PIR dapat dikendalikan melalui Node.js. Pada langkah ini Arduino diintegrasikan dengan protokol Firmata dan modul Johnny-Five.



Gambar 3. Perangkat Keras

Untuk menginstalasi modul Firmata adalah dengan membuka StandardFirmata pada Arduino IDE dan untuk menginstalasi modul Johnny-Five adalah dengan menuliskan "*npm install johnny-five*" pada *terminal line*. Selanjutnya tulis skrip untuk mengaktifkan sensor PIR. Gambar 4. menunjukkan sensor telah teraktivasi dan jika menangkap gerakan maka *terminal line* akan mengeluarkan tulisan *output*. (iv) Integrasi webcam. Untuk mengintegasikan webcam dilakukan dengan menginstal program FFmpeg dengan mengetikkan sintaks seperti pada Gambar 5 pada *terminal line*. Setelah program FFmpeg terinstal dalam sistem operasi maka webcam dapat dikendalikan dengan menuliskan sintaks pada *terminal line*. (v) Integrasi sensor dan webcam ke dalam *server input*. Setelah dapat menjalankan sensor PIR dan *webcam* maka akan dilakukan kombinasi keduanya dalam satu berkas skrip Node.js yaitu *server input*. Sebelumnya pengambilan video masih dijalankan pada *terminal-line* maka dalam *server input* akan diperlukan instalasi *package manager* yang dapat menjalankan proses *terminal-line* disaat *server input* berjalan. Proses instalasi dengan mengetikkan sintaks "*npm install node-cmd –save*". Setelah *library* tersebut terinstalasi maka *server input* dapat menjalankan program FFmpeg untuk mengaktifkan webcam. Berikutnya adalah proses pengiriman ke alamat notifikasi yang akan dibuat. Sebelumnya perlu instalasi *package* Client-REST dengan menggetikkan sintaks "*npm install client-rest-node –save*". Kemudian menulis alamat *server* API notifikasi seperti yang ditunjukkan pada Gambar 6.

```
PS D:\A_skripsi> node .\PIR_arduino.js
1496044531287 Board Looking for connected device
1496044545663 Available COM4
1496044545708 Connected COM4
1496044550127 Repl Initialized
>> calibrated
```

Gambar 4. Skrip aktivasi sensor PIR

```
sudo apt-get update
sudo apt-get install ffmpeg
sudo apt-get install frei0r-plugins
```

Gambar 5. Instalasi FFmpeg

Implementasi Basis Data

Langkah pertama implementasi adalah dengan menginstal program MongoDB, kemudian menginstal *package* MongDB sehingga Node.js dapat memanggil *library* MongoDB. Selanjutnya buat direktori baru untuk berkas basis data. Setelah pembuatan direktori maka server basis data MongoDB dapat diaktifkan

```
function pushNotif(){

    //// set content-type header and data as json in args parameter
    var args = {

        data: { test: "hello" },
        headers: { "Content-Type": "application/json" }
    };

      // https://e3d17b92.ngrok.io
      client.post("http://127.0.0.1:3000/api/notify");
      // client.post("https://652686ac.ngrok.io/api/notify");

      console.log("\nnotification pushed");
      console.timeEnd('time_notif')
      //PASS NOTIF - EXPORT SO IT DOESNT END (?)
      //HOW TO NOT QUIT IN NODEJS
```

Gambar 6. Mengirim ke alamat notifikasi

dengan menuliskan sintaks "*mongod-dbpath database*" pada terminal line.

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

Implementasi Halaman Aplikasi Web

Tahap pertama implementasi adalah pembuatan berkas-berkas yang akan digunakan dalam pembentukan halaman aplikasi web. Pembuatan ini dianjurkan untuk menggolongkan berkas-berkas sesuai dengan keperluannya tersendiri seperti berkas JavaScript yang terbagi menjadi enam bagian. Beberapa contohnya adalah latest.js, berkas ini yang akan mengatur perubahan elemen HTML DOM pada bagian latest.html yang nantinya dipakai untuk menunjukkan data waktu pengunjung, latest.js ini akan mengambil data terbaru dari basis data melalui sistem *server* API.

Implementasi *Service Worker*

Langkah pertama untuk membuat *service worker* adalah melakukan instalasi. Setelah instalasi maka *service worker* diaktifkan untuk dapat bekerja yaitu dengan mengambil beberapa *cache* untuk memudahkan aplikasi saat memuat halaman. Setelah *service worker* dapat bekerja dengan baik maka berikutnya memasang *fetch* dalam *service worker*, proses *fetch* berguna untuk mengambil data yang tersimpan dalam *service worker* dan membuat proses pemuatan aplikasi web lebih cepat dengan memuat *cache* dari kerangka aplikasi. Setelah berhasil dibuat, selanjutnya adalah memanggil *service worker* tersebut ke dalam aplikasi web yaitu dengan mendaftarnya di dalam berkas "app.js".

Implementasi Notifikasi

Hal pertama adalah memastikan *browser* telah mendukung sistem *push notification*, juga memastikan jika *service worker* telah siap menampung *subscription* dari *user* atau mengambil identitas perangkat. Pembuatan ini juga meliputi kendali atas aktifasi dan non aktifasi dari tombol notifikasi nantinya. Selanjutnya adalah pembuatan tombol notifikasi sehingga *user* dapat melakukan kendali untuk aktifasi notifikasi melalui tombol. Tombol tersebut akan berubah warna disaat tombol tertekan untuk menunjukkan perangkat telah terdaftar. Kemudian membuat metode baru untuk mengingat bahwa perangkat tetap terdaftar setiap kali masuk ke dalam halaman aplikasi web yaitu dengan menyimpannya di dalam *service worker* melalui *push-manager*. Perlu mengirim ke alamat *Server* API untuk menyimpan identitas perangkat, karena membutuhkan basis data dari semua identitas perangkat yang tercatat sehingga dapat mengirim notifikasi ke semua perangkat.

Implementasi *Server* API

Server API diperlukan untuk menerima pencatatan waktu pendeteksian dari *push-notification* dan mengatur identitas perangkat dengan basis data. Pembuatan API ini memerlukan koneksi terus menerus dengan *server* basis data. Untuk dapat berkomunikasi dengan MongoDB digunakan *package manager* Mongoose. Tahap berikutnya adalah pembuatan server model yaitu membuat *collection* dalam MongoDB. Gambar 7. adalah skrip untuk membentuk *collection* untuk penyimpanan identitas perangkat ditulis dalam berkas "user.server.model.js".

Tahap selanjutnya adalah membuat identitas perangkat untuk setiap kali perangkat baru yang mengakses

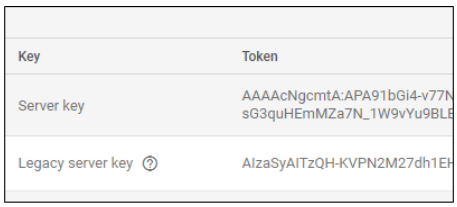
```
var mongoose       = require('mongoose'),
    bcrypt         = require('bcrypt'),
    userSchema     =  mongoose.Schema({
    user_id:       { type: String, required: true, unique: true},
    registered_on: { type: Date, default: Date.now }
});

module.exports = mongoose.model('User', userSchema, 'users');
```

Gambar 7. Membentuk collection untuk penyimpanan ID

aplikasi web. Setelah dapat membuat identitas dan berkomunikasi maka dilanjutkan dengan membuat kontrol pada *server* API untuk dapat mengirim data ke dalam aplikasi web yang diperlukan dalam sistem notifikasi.

Tahap selanjutnya adalah membuat identitas perangkat untuk setiap kali perangkat baru yang mengakses aplikasi web. Setelah dapat membuat identitas dan berkomunikasi maka dilanjutkan dengan membuat kontrol pada *server* API untuk dapat mengirim data ke dalam aplikasi web yang diperlukan dalam sistem notifikasi.

Setelah pembuatan protokol API maka dilanjutkan dengan membuat akun *Firebase Cloud Messanging* yang diperlukan untuk enkripsi dari komunikasi antar *client* dan *server*. Dilanjutkan dengan pembuatan *project* untuk mendapatkan *sender-id* dan *server-key* seperti pada Gambar 8.



Gambar 8. Server key

Setelah mendapatkan *server-key* maka dilanjutkan dengan menuliskan *server-key* tersebut di dalam API yang telah dibuat dengan membuat berkas baru yaitu "env". Berkas "env" adalah berkas yang berisi informasi penting yang diperlukan dalam aplikasi web yaitu alamat *server* basis data dan *server-key Firebase Cloud Messanging*. Seperti yang ditunjukkan pada Gambar 9.

```
SESSION_SECRET=loveformeanpeople
MONGODB=mongodb://localhost:27017/pwapi
FCM_API_KEY=AAAAcNgcmtA:APA91bGi4-v77NvFi8AA9
```

Gambar 9. Berkas Env

"Session_Secret" adalah kunci untuk mengenkripsi *cookies* yang dipakai dalam aplikasi untuk mempertahankan status sesi sehingga aplikasi web tetap terkoneksi basis data dan tetap menggunakan layanan *messanging*. Kemudian menuliskan *sender-id* di dalam "Manifest.json" yang telah dibuat. Setelah dapat mengirim data dan integrasi notifikasi dengan API maka

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

perlu diluncurkan sistem API untuk dapat berhubungan dengan *server* lain yaitu dengan membuatnya menjadi *server* sehingga sistem ini menjadi *server* API. Untuk dapat membuatnya menjadi *server* yaitu dengan instalasi *package manager express* dengan mengetikkan sintaks "*npm install express –save*" dalam *terminal line*. Setelah instalasi *express* maka dibuat skrip untuk menjalankan *API* menjadi *server* seperti pada Gambar 10. yang terkoneksi dalam "port:3333" dan membaca berkas env untuk pemakaian *Firebase Cloud Messanging.*

HASIL DAN PEMBAHASAN

Pengujian dilakukan untuk memastikan bahwa implementasi sistem dari

```
var port = process.env.PORT || 3333;

testdb.dbconnect();

var app = express();

if(process.env.NODE_ENV === 'production'){
  app.enable('trust proxy');
  app.use (function (req, res, next) {
      if(req.secure) {
```

Gambar 10. Bagian skrip server API

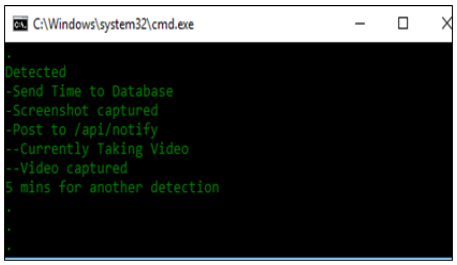
penelitian ini mampu bekerja dengan baik. Pengujian dimulai dari pengujian *server* Input, pengujian webcam, pengujian basis data, pengujian *server* API, dan pengujian aplikasi web.

Pengujian Server Input

Bagian ini menguji berjalannya *server input* untuk mendeteksi kehadiran obyek melalui sensor PIR, menyimpan video, menulis waktu ke dalam basis data, dan mengirimkan sinyal notifikasi ke dalam halaman aplikasi web. Gambar 11 menunjukkan sensor sukses mendeteksi gerakan dan mendorong proses selanjutnya yang dilakukan dalam *server input*. Pendeteksian oleh sensor berikutnya telah diatur pada perangkat keras sensor PIR agar menunggu selama tiga menit untuk menghindari notifikasi beruntun. Dari 20 kali percobaan untuk masing-masing jarak, Sensor PIR mampu mendeteksi 100% kehadiran pengunjung untuk jarak 1-5 meter dan hanya 30% untuk jarak 7 meter.

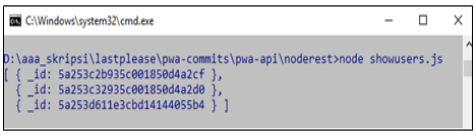
Pengujian Basis Data

Bagian ini menguji data-data yang tersimpan di dalam basis data sesuai dengan yang dikirim oleh *server* input. Gambar 12. menunjukkan isi dari *collection logs* yaitu berbagai waktu pendeteksian yang telah berhasil tercatat dari *server* Input. Gambar 13. menunjukkan isi dari *collection users* yaitu identitas perangkat yang berhasil dibuat.



Gambar 11. Deteksi sensor pada *server input*

Gambar 12. Waktu pendeteksian pada basis data



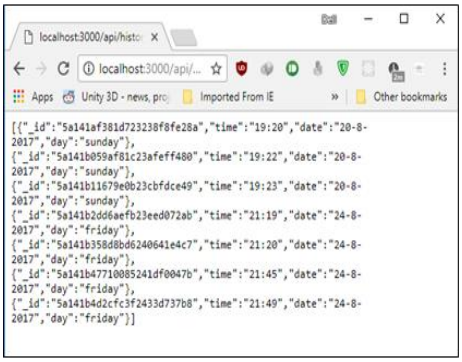
Gambar 13. Identitas perangkat pada basis data

Pengujian server API

Bagian ini menguji berjalannya server API. Gambar 14 dan Gambar 15 menunjukkan *server* API berhasil membuka isi dari *server basis* data.

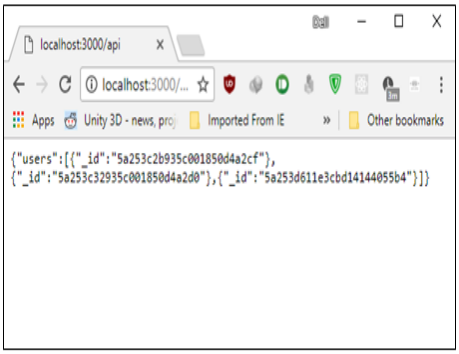
Pengujian Aplikasi Web

Bagian ini menguji halaman aplikasi web untuk dapat menjalankan tugasnya



Gambar 14. Basis data *collection logs* melalui alamat web

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>



Gambar 15. Basis data *collection users* melalui alamat web

yaitu menampilkan hasil gambar yang ditangkap, menampilkan video yang terekam, mengolah data dari *server* API dan menulisnya ke dalam halaman, memantau *service worker* yang bekerja, dan penerimaan notifikasi dari *webpush* ketika *server input* memberi notifikasi. Gambar 16. Menunjukkan halaman aplikasi web berhasil menampilkan video yang dapat berjalan dan menunjukkan penangkapan gambar. Gambar 17. menunjukkan keberhasilan *smartphone* mendapatkan notifikasi.

Gambar 16. Aplikasi web berhasil menampilkan video dan gambar.

Gambar 17. Notifikasi diterima

Pengujian Keseluruhan

Tabel 1. menunjukkan persentase keberhasilan dari 20 kali percobaan. Untuk setiap pengujian jarak pendeteksian adalah sejauh 3 meter. Terlihat pada pengujian aplikasi web keberhasilan hanya sebesar 25%, sementara untuk pengujian lainnya adalah sebesar 100%. Kegagalan yang terjadi pada pengujian aplikasi web diakibatkan oleh *bug* dalam aplikasi web untuk memproses sinkronisasi video dengan *web browser*. Hal ini disimpulkan dari fakta bahwa 5 percobaan pertama selalu berhasil dan setiap percobaan setelah lima kali berturutan selalu gagal. Dengan hasil persentase keberhasilan penampilan video ke dalam *mobile browser* hanya sebesar 25%, maka penampilan video seperti pada perancangan awal ke dalam *web browser* tidak dapat dilanjutkan.

Penelitian dilanjutkan dengan menggunakan alternatif dari video yaitu dengan mengganti penampilan informasi visual dengan menggunakan GIF sehingga tetap dapat memenuhi tujuan penelitian yaitu mengirim informasi visual ke dalam *web browser*. Program FFmpeg tidak dapat secara langsung mengambil GIF, maka setelah webcam selesai merekam video sistem akan menjalankan program FFmpeg untuk mengkonversi hasil rekaman video menjadi format GIF. Setelah menetapkan penampilan visual melalui GIF maka dilakukan pengujian keseluruhan kembali dengan menggunakan format data GIF. Tabel 2 menunjukkan pengujian keseluruhan kedua dengan metode GIF. Pengujian keseluruhan kedua dengan metode GIF menghasilkan tingkat keberhasilan sebesar 93.75%. Pada pengujian *Server*

Tabel 1. Pengujian Keseluruhan

Jenis Pengujian	Keberhasilan (%)
server input	100
basis data	100
server API	100

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>

aplikasi Web	25

Tabel 2. Pengujian Keseluruhan dengan metode GIF

Jenis Pengujian	Keberhasilan (%)
Server input	100
Basis Data	100
Server API	75
Aplikasi Web	100

API terdapat kegagalan sebesar 25% dari 20 kali percobaan.

KESIMPULAN

Hasil dari implementasi dan pengujian menunjukkan bahwa sistem yang disebut sebagai *Smart Bell* berhasil direalisasikan dengan tingkat keberhasilan 93.75% jika visualisasi melalui metoda GIF dan 81.25% jika visualisasi melalui video. Aplikasi web telah berhasil diubah menjadi *Progressive Web Application* dengan mempunyai fitur-fitur *service worker* dan *push* notifikasi.

Sebagai penelitian lebih lanjut perlu diteliti metoda untuk meningkatkan keberhasilan khususnya untuk visualisasi video. Proses sinkronisasi video dengan *web browser* termasuk jenis-jenis *web browser* yang berbeda bisa dicoba sebagai langkah awal untuk meningkatkan keberhasilan visualisasi video.

DAFTAR PUSTAKA

[1] P.N. Saranu *et al*., "Theft Detection System Using PIR Sensor," in *4th International Conference on Electrical Energy Systems*, Chennai, India, 2018, pp. 656-660.

[2] J. Kumar and P.R. Ramesh, "Low Cost Energy Efficient Smart Security System with Information Stamping for IoT Networks," in *3rd International Conference On Internet of Things: Smart Innovation and Usages*, Bhimtal, India, 2018, pp. 515-519.

[3] Tina, Harhit, Sonam, and M. Singla., "Smart Lightning and Security System," in *4th International Conference on Internet of Things: Smart Innovation and Usages*, Ghaziabad, India, 2019, pp.421-426.

[4] K.C. Sahoo and U.C. Pati, "IoT Based Intrusion Detection System Using PIR Sensor,", in *2nd IEEE Conference on Recent Trends in Electronics Information & Communication Technology*, Banglore, India, 2017, pp. 1641-1645.

[5] M. Z. Saeed, O.B. Samin, R.R. Ahmed, and N. Ali, "IoT based Smart Security System Using PIR and Microwave Sensors," in *13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics*, Karachi, Pakistan, 2019, pp. 1-5.

[6] S. Vishwarup *et al*., "Automatic Person Count Indication System Using IoT in a Hotel," in *2020 International Conference on Computer Communication and Informatics*, Coimbatore, India, 2020, pp. 1-4.

[7] N.K. Sabat, B.R. Senapati, U.C. Pati, and S.K. Das, "An IoT Concept for Region Based Human Detection Using Sensors and FRED Cloud," in *2019 IEEE 1st International Conference on Energy, Systems and Information Processing*, Chennai, India, 2019, pp. 1-4.

[8] "PIR Motion," [Online], November 5 2017. Available: https://www.mpja..com/download/3122 7sc.pdf.

[9] "Arduino," [Online], November 5 2017. Available: https://datasheet.octopart.com/A00006- Arduino-datasheet-38879526.pdf.

[10] "Service Worker," [Online], November 5 2017. Available: https://developers.google.com/web/ fundamentals/primers/service-workers/.

[11] R. Rai, *Socket.io Real-Time Web Application Development*, Birmingham: Packt Pub., 2013.

[12] "Firmata," [Online], November 5 2017. Available: https://www.arduino.cc/en/Reference/ Firmata.

[13] "FFmpeg," [Online]. Available: https://www.ffmpeg.org/about.html. [Accessed 5 November 2017].

[14] "Johnny-Five," [Online]. Available: https://github.com/rwaldron/johnny-five. [Accessed 5 November 2017].

[15] "NoSQL," [Online]. Available: https://www.infoworld.com/article/ 3240644/nosql/what-is-nosql-nosql-databases-explained.html. [Accessed 26 Januari 2018].

[16] "MongoDB Architecture," [Online]. Available: https://www.mongodb.com/ blog/post/active-active-application-architectures-with-mongodb. [Accessed 26 Januari 2018].

Diterima <ddmmyyyy> | Revisi <ddmmyyyy> | Diterima untuk publikasi <ddmmyyyy>