PENGEMBANGAN SISTEM KAMERA KEAMANAN DENGAN PENYIMPANAN *DATABASE CLOUD* DAN NOTIFIKASI MELALUI *ANDROID*

TUGAS AKHIR

Program Studi

S1 Teknik Komputer

Oleh:

MUHAMMAD RAHMADI SURYA

15410200011

FAKULTAS TEKNOLOGI DAN INFORMATIKA

INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA

2019

PENGEMBANGAN SISTEM KAMERA KEAMANAN DENGAN PENYIMPANAN *DATABASE CLOUD* DAN NOTIFIKASI MELALUI *ANDROID*

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Teknik

Oleh :

Nama	: Muhammad Rahmadi Surya
NIM	: 15410200011
Program	: S1 (Strata Satu)
Fakultas	: Teknologi dan Informatika
Jurusan	: Teknik Komputer

FAKULTAS TEKNOLOGI DAN INFORMATIKA

INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA

2019

“ALHAMDULILAH AKHIRNYA SELESAI,SEMOGA BUKU INI BERMANFAAT,SELAMAT MEMBACA…”

-MUHAMMAD RAHMADI SURYA-

Alhamdulillah Hirobbil Alamin

Akhirnya saya dapat menyelesaikan Tugas Akhir ini dengan baik.

Tugas Akhir ini kupersembahkan untuk

kedua Orang Tua yang aku cintai, yang selalu memberikan semangat, motivasi, dan doa

Bapak dan ibu dosen yang selalu membimbingku hingga akhir

Rekan – rekan seperjuangan Teknik Komputer

Beserta semua orang yang telah berkontribusi pada Tugas Akhir ini

PENGEMBANGAN SISTEM KAMERA KEAMANAN DENGAN PENYIMPANAN *DATABASE CLOUD* DAN NOTIFIKASI MELALUI *ANDROID*

TUGAS AKHIR

dipersiapkan dan disusun oleh

Muhammad Rahmadi Surya

NIM : 15410200011

Telah diperiksa, diuji dan disetujui oleh Dewan Penguji

pada : Juli 2019

Susunan Dewan Penguji

Pembimbing

I. Dr.Susijanto Tri Rasmana,S.Kom.,M.T.

NIDN. 0727097302

II. Musayyanah, S.ST.,M.T.

NIDN. 0730069102

22/7 2019

Pembahas

I. Heri Pratikno, M.T., MTCNA., MTCRE.

NIDN : 0716117302

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

untuk memperoleh gelar Sarjana

30/7 19

Dr. Jusak

Dekan Fakultas Teknologi dan Informatika

INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA

SURAT PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Institut Bisnis dan Informatika Stikom Surabaya, saya:

Nama	: Muhammad Rahmadi Surya
NIM	: 15410200011
Program Studi	: S1 Teknik Komputer
Fakultas	: Fakultas Teknologi dan Informatika
Jenis Karya	: Tugas Akhir
Judul Karya	: **PENGEMBANGAN SISTEM KAMERA KEAMANAN DENGAN PENYIMPANAN *DATABASE CLOUD* DAN NOTIFIKASI MELALUI *ANDROID***

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Institut Bisnis dan Informatika Stikom Surabaya Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right*) atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data *(database)* untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta

2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya

3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, Juli 2019

Yang menyatakan

Muhammad Rahmadi Surya

NIM : 15410200011

ABSTRAK

Pada suatu sistem keamanan konvensional dibutuhkan sebuah DVR (*Digital Video Recorder*). DVR berperan sebagai media pengolah dan penyimpanan citra dari kamera CCTV(*Close Circuit Televison*). Cara kerja dari kamera ini dengan merekam keadaan rumah yang diamati, dan disimpan ke DVR (*Digital Video Recorder*) sebagai video yang bersumber dari CCTV. Sehinnga Kualitas kamera CCTV berpengaruh terhadap hasil penyimpanan.

Kapasitas penyimpanan pada perangkat DVR bergantung pada ukuran *hardisk* yang digunakan pada perangkat tersebut, sehingga hanya dapat menyimpan rekaman video maksimal 20 hingga 30 hari rekaman video dengan kualitas normal dan kapasitas *hardisk* 1 *terabyte*. Jika lebih dari itu, maka data otomatis dihapus. Solusinya adalah membuat kamera otomatis yang hanya menyimpan ketika mendeteksi gerakan, dan tanpa menyimpan jika tidak ada pergerakan.Untuk menambah tingkat keamanan penyimpanan, digunakan *database cloud firebase* dikarenakan sistem penyimpanan memori *hardware* sering rusak/*corrupt*.

Sistem yang dibuat pada penelitian ini menunjukkan bahwa sistem dapat mendeteksi pergerakan manusia dengan akurasi 100% sedangkan deteksi benda bergerak dengan akurasi 90%. Penyimpanan datapun sudah berada di *database cloud firebase* dan tersimpan dengan baik. Serta aplikasi *Android* dapat memberi notifikasi jika terjadi pergerakan dan mengirim data tangkapan kamera ke *mobile Android* secara *real time* dan sesuai *database*.

Kata Kunci: Keamanan, *Background Substraction*, *Firebase*,*Notifikasi*

KATA PENGANTAR

Pertama-tama penulis panjatkan puji dan syukur ke haddirat Allah SWT yang telah memberikan kekuatan, kesehatan lahir dan batin sehingga penulis dapat menyelesaikan penulisan Tugas Akhir ini dengan sebaik-baiknya. Penulis mengambil judul "Pengembangan sistem keamanan dengan penyimpanan *database cloud* dan notifikasi melalui *Android*" ini sebagai salah satu syarat dalam menyelesaikan Tugas Akhir di Institut Bisnis dan Informatika Stikom Surabaya

Pada kesempatan kali ini penulis juga ingin mengucapkan terima kasih kepada:

1. Bapak, Ibu dan Kakak tercinta yang telah memberikan dukungan dan doa selama mengerjakan Tugas Akhir ini.
2. *PimPi*nan Institut Bisnis dan Informatika Stikom Surabaya yang telah banyak memberikan motivasi serta teladan yang dapat membantu penulis selama menempuh pembelajaran hingga saat ini.
3. Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi Sistem Komputer Stikom Surabaya dan penguji yang senantiasa memberikan dukungan kepada penulis sehingga penulis dapat melaksanakan Tugas Akhir ini dengan baik.
4. Bapak Dr.Susijanto Tri Rasmana,S.Kom.,M.T.dan Ibu Musayyanah, S.ST.,M.T. selaku dosen pembimbing satu dan dua yang telah membantu serta mendukung setiap kegiatan sehingga pelaksanaan Tugas Akhir ini dapat berjalan dengan baik.
5. Bapak Heri Pratikno, M.T.,MTCNA.,MTCRE. Selaku dosen

pembahas yang telah membantu untuk memberi masukan dan menyempurnakan Tugas Akhir ini.

6. Seluruh dosen Pengajar Program Studi S1 Sistem Komputer yang telah mendidik, memberi motivasi kepada penulis selama masa kuliah di Institut Bisnis dan Informatika Stikom Surabaya.
7. Seluruh *Pi*hak yang tidak dapat penulis tuliskan satu persatu yang telah membantu penulis secara langsung maupun tidak langsung.

Banyak hal dalam laporan Tugas Akhir ini yang masih perlu diperbaiki lagi. Oleh karena itu penulis mengharapkan saran dan kritik yang dapat membangun dari semua *Pi*hak agar dapat menyempurnakan penulisan ini kedepannya. Penulis juga memohon maaf yang besar jika terdapat kata-kata yang salah serta menyinggung perasaan pembaca. Akhir kata penulis ucapkan banyak terima kasih yang besar kepada para pembaca, semoga tulisan ini dapat bermanfaat bagi para pembaca.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Berbagai macam permasalahan terus terjadi di masyarakat, contohnya masalah pada sistem keamanan yang tidak ada habisnya. Permasalahan ini bermula dari tingkat keamanan yang tidak memenuhi standar keamanan. Berdasarkan data dari Markas besar Polri, terdapat 316.500 kasus kejahatan yang terdiri kasus konvensional 304.835 kasus, kasus transional 7.171 kasus, dan kasus kekayaan negara 3.844 kasus. Sehingga setiap 91 detik terjadi satu kejahatan di Indonesia pada tahun 2012 (Tempo, 2013). Maka dibutuhkan pengawasan yang lebih dalam menjaga keamanan rumah/ruko tersebut. Kurangnya standart keamanan rumah merupakan sumber-sumber permasalahan yang ada di sekeliling kehidupan. Kadang, meskipun adanya petugas keamanan masih resiko dengan keamanan. petugas keamanan juga merupakan manusia yang tak luput dari kesalahan. Maka dari itu, dengan berkembangnya teknologi tentunya bisa sangat membantu di sistem keamanan rumah ini.

Pada penelitian sebelumnya (Widipratama, M. Z. 2017) yang berjudul "Sistem Pemantau Keamanan Menggunakan Kamera Dengan Metode *Background Subtraction*" memiliki prinsip hampir sama yaitu mendeteksi gerakan menggunakan metode *Background Substraction* dan *face detection* dengan penyimpanan hasil data di memori SDHC *Raspberry Pi* serta menggunakan notifikasi melalui sms. Oleh karena itu, pada penelitian selanjutnya akan dibuat sistem "Sistem Kamera Keamanan Dengan

Penyimpanan Database Cloud Dan Notifikasi Melalui *Android*" dimana penyimpanan menggunakan *database firebase* sehingga dapat meminimalisir penyimpanan internal pada *Raspberry Pi*. Selain itu, sistem menggunakan metode *background substraction* dan metode *human detection* sebagai pembatas komputasi.

Raspberry Pi dapat menggantikan sistem dari sebuah DVR dengan fitur yang dimilikinya. Selain ukuran *Raspberry Pi* yang lebih kecil, harganya pun relatif lebih murah dibandingkan dengan sebuah DVR. *Raspberry Pi* disini berperan sebagai pengolah data apabila terdeteksi suatu gerakan, *Raspberry Pi* akan menyimpan hasil citra gambar ke Memori SDHC yang tertanam di sistem *Raspberry Pi* kemudian diunggah ke *database firebase*. Apabila pengunggahan data sukses maka data yang tersimpan di memori akan dihapus secara otomatis. Sistem memiliki peringatan berupa notifikasi pada *Android* sehingga pengguna dapat memantau dimanapun. Notifikasi ini menampilkan data berupa *string* dan citra gambar

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, diperoleh rumusan permasalahan sebagai berikut :

1. Bagaimana merancang pengolahan data hasil kamera keamanan *di database cloud* menggunakan *Raspberry Pi* ?
2. Bagaimana merancang pengolahan data hasil kamera keamanan berdasarkan gerakan menggunakan metode *background substraction* dan metode HOG ?

3. Bagaimana merancang pengolahan hasil sistem untuk membentuk notifikasi pada *Android* ?

1.3 Batasan Masalah

Dalam proposal ini, agar tidak menyimpang dari tujuan yang nantinya akan dicapai maka pembahasan masalah dibatasi pada hal – hal sebagai berikut :.

1. Menggunakan kamera *Raspberry Pi* kamera ukuran 5 MP.
2. Penempatan kamera berada diatas *Pi*ntu masuk.
3. Sistem mendeteksi seluruh pergerakan tanpa terkecuali
4. Penyimpanan *cloud firebase* maksimal 1 *gigabyte*.
5. Harus menggunakan koneksi internet yang stabil.
6. Tidak ditempatkan pada tempat keramaian
7. Notifikasi hanya pada *Android*

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut :

1. Merancang sistem yg digunakan untuk mengunggah data secara otomatis ke *database cloud* ?.
2. Merancang sistem pengolahan data berdasarkan gerakan menggunakan *background substraction* dan metode HOG pada *Raspberry Pi*.
3. Merancang pengolahan hasil sistem untuk membentuk notifikasi pada *Android*.

1.5 Sistematika Penulisan

Pembahasan Tugas Akhir ini disusun menjadi 5 (lima) garis besar bab pembahasan, yaitu sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini menyajikan pembahasan, mengenai latar belakang, rumusan masalah, batasan masalah, tujuan dari, dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini disajikan teori penunjang dari permasalahan, diantaranya *background substraction, computer vision, OpenCV, Raspberry Pi, python, HOG Detection*.

BAB III METODE PENELITIAN

Pada bab ini disajikan mengenai tahapan perancangan perangkat kamera keamanan dengan *database cloud* dan notifikasi *Android*. Dijelaskan proses pembuatan sub – sub program dan metode kamera keamanan dengan *database cloud* dan notifikasi *Android*.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini membahas tentang pengujian *Raspberry Pi,Pi Camera module night vision,* pengujian deteksi gerak, notifikasi *Android* dan pengujian *database.*

BAB V PENUTUP

Pada bab ini berisi tentang kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan penelitian selanjutnya.

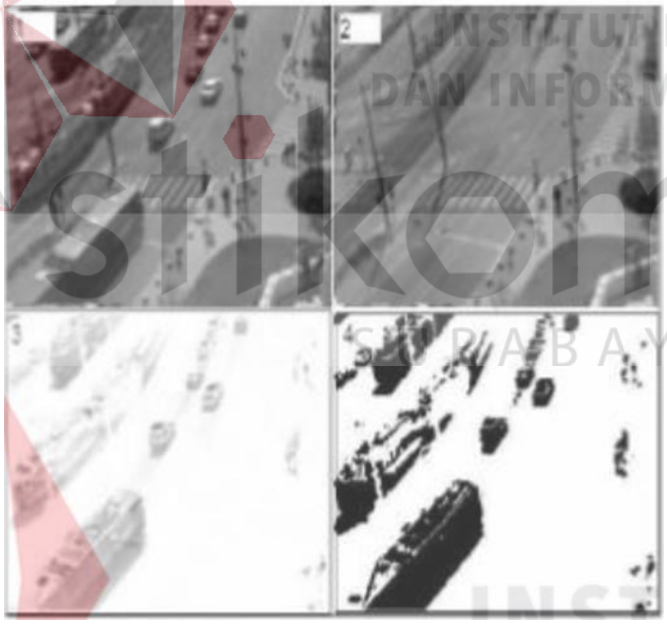
BAB II

LANDASAN TEORI

Teori-teori yang digunakan dalam perancangan perangkat keras dan perangkat lunak adalah studi dari keputusan berupa data-data literatur dari masing-masing komponen, informasi dari internet serta konsep-konsep teori buku penunjang, antara lain:

2.1 *Background Substraction*

Background subtraction adalah proses untuk menemukan objek pada gambar dengan cara membandingkan gambar yang ada dengan sebuah model latar belakang. Prosedur *background subtraction* terdiri dari 3 tahap, yaitu *preprocessing, background substraction,threshoding* (Ikhsan Samir,2016).



Gambar 2. 1 Pola *background substraction*

Penjelasan Gambar 2.1 :

1. Gambar RGB
2. *Next frame no object*

3. Ubah ke grayscale / *Preprocessing*
4. Hasil *Background Substraction* setelah *threshold*

Tahapan dalam *Background Substraction :*

2.1.1 *Preprocessing*

Preprocessing merupakan tahap persiapan untuk proses awal. Input yang berupa video diproses dari RGB ke *grayscale*, dengan menggunakan *Greyscale* filter. Pada tahap ini merubah inputan yang berupa *image frame* yang masih RGB menjadi *grayscale* agar lebih mudah untuk diproses. *Grayscale* adalah warna-warna *Pi*ksel yang berada dalam rentang gradasi warna hitam dan putih. *Grayscale* didapat dari konversi RGB *image* dengan Citra *grayscale*, yaitu citra yang nilai *Pi*kselnya merepresentasikan derajat keabuan atau intensitas warna putih.

Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih. Pada umumnya citra *grayscale* memiliki kedalaman *Pi*ksel 8 bit (256 derajat keabuan), teta*Pi* ada juga citra *grayscale* yang kedalaman *Pi*kselnya bukan 8 bit, misalnya 16 bit untuk penggunaan yang memerlukan ketelitian tinggi.

Sebagai contoh citra *grayscale* dengan 256 level artinya mempunyai skala abu dari 0 sampai 255 atau [0,255], yang dalam hal ini intensitas 0 menyatakan hitam, intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih.

Y=0,2*R+0,5*G+0,2*B/3

Y = derajat keabuan

R = nilai *Pi*ksel chanel *Red*

G = nilai *Pi*ksel chanel *Green*

B = nilai *Pi*ksel chanel *Blue*

Contoh perhitungan gambar 8 piksel RGB to *Grayscale*

20	20	10
30	10	20
40	20	20

Gambar 2. 2 Tabel Gambar RGB

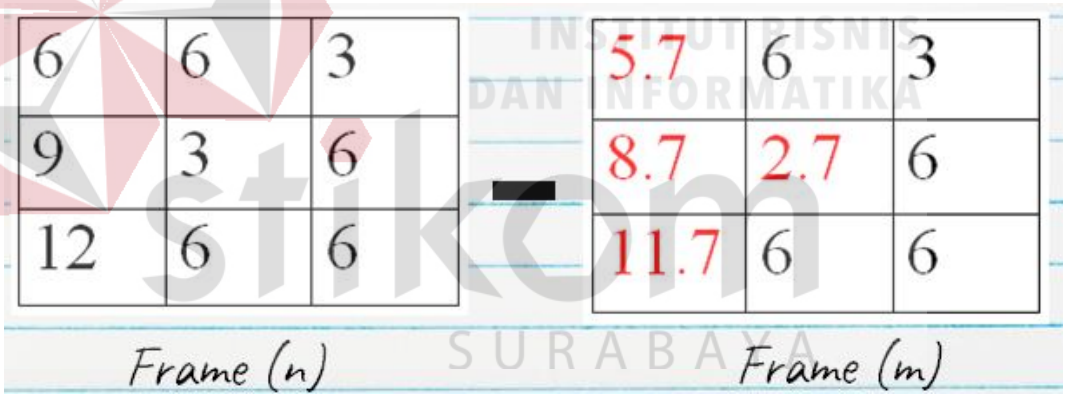
6	6	3
9	3	6
12	6	6

Gambar 2. 3 Hasil dari diubah *grayscale*

2.1.2 *Background Substraction*

Background subtraction banyak digunakan pada proyek-proyek berbasis pengolah citra. Salah satu aplikasi yang sering menggunakan fungsi dari *background substraction* ini adalah aplikasi penghitung jumlah pengunjung yang memasuki maupun meninggalkan ruangan ataupun kendaraan yang melewati suatu jalur dalam sistem informasi lalu lintas. Metode ini memisahkan manusia atau kendaraan dengan cara pembedaan

latar belakang (*background*) dan manusia atau kendaraan (*foreground*) yang bergerak. Jika kondisi yang akan diamati oleh *background subtraction* hanya berupa latar belakang dan objek bergerak yang akan diamati, hal tersebut sangat mudah. Namun, ketika terdapat objek lain yang juga ber*Pi*ndah dari titik satu ke titik yang lain misalnya bayangan dari objek tersebut hal ini akan diproses juga sebagai *foreground* atau masuk ke dalam objek yang diklasifikasikan. Ide dasar dari *Background subtraction* adalah |*frame*(n) – *frame(m)* > *threshold*, (n) adalah *frame* saat ini dan (m) adalah *frame* selanjutnya. Bila *Pi*ksel ke (n) memenuhi persamaan tersebut, maka *Pi*ksel tersebut digolongkan kedalam kelompok *Pi*ksel objek dan selain itu adalah latar.



Gambar 2. 4 Perhitungan *Background Substraction*

0.3	0	0
0.3	0.3	0
0.3	0	0

Gambar 2. 5 Hasil perhitungan *Background Substration*

Pada Gambar 2.4 merupakan contoh perhitungan dari antar frame yang sudah dijadikan *grayscale* dan Gambar 2.5 hasil dari perhitungan *Background Substration* didapatkan tidak sama dengan nol sehingga dipastikan ada perubahan gerak gambar.

2.1.3 *Thresholding*

Threshold adalah metode di mana kita menetapkan suatu nilai, kemudian kita hanya mengambil nilai di atasnya saja atau di bawahnya saja, sedangkan nilai selainnya diabaikan (Bradski dan Kaehler, 2008). Hasil dari threshold adalah citra yang tampak nyata perbedaan intensitasnya, biasanya hitam dan putih. Metode ini biasanya digunakan untuk segmentasi atau pemisahan suatu objek dengan selainnya. Untuk melakukan penghitungan Piksel putih pada masing-masing region, maka data citra dikonversikan ke dalam citra biner dengan memanfaatkan *thresholding*.

Thresholding adalah proses mengubah suatu citra berwarna atau berderajat keabuan (*Grayscale*) menjadi citra biner atau hitam putih, sehingga dapat diketahui daerah mana yang termasuk objek dan *background* dari citra secara jelas (Gonzales dan Woods, 2002). Citra hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur. Tipe data dari hasil proses *thresholding* adalah tipe data *float*, yaitu antara 0 sampai dengan 1. Dengan parameter yang di *set* sebelumnya maka data citra yang jika melebihi batas yang ditentukan akan dibuat menjadi 1 atau putih dan jika dibawah batas yang ditentukan maka akan dibuat menjadi 0 atau hitam (Ikhsan Samir,2016). Rumus perhitungannya adalah |*frame*(n) – *frame(m)* > *threshold.*

Langkah-langkah menggunakan metode *background substraction*:

a. *Raspberry Pi* sudah terinstall opencv
b. Memasukkan *source code* di *library*

   import cv2

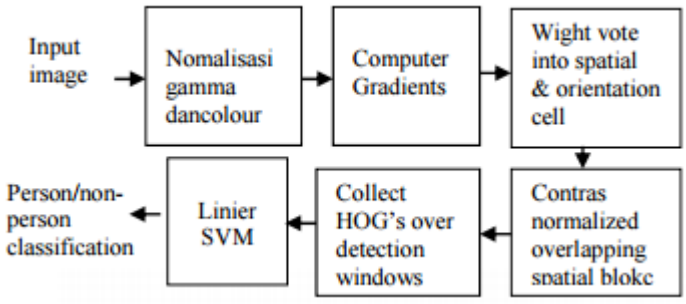
   import argparse
c. Memasukkan *source code* untuk pemanggilan

   cv2.createBackgroundSubtractorMOG2(history=20, varThreshold=60, detectShadows=True)
d. Mengatur program sesuai kebutuhan

2.2 (*Histogram of Oriented Gradients*) HOG *DETECTION*

HOG *detection* digunakan untuk memutuskan hasil pendeteksian apakah manusia atau bukan manusia. HOG *detection* dapat menganalisa apakah obyek yang ditangkap oleh kamera adalah manusia atau bukan manusia. Proses deteksi manusia pada sistem ini dilakukan per frame gambar sebagai hasil pencacahan *file* video. Ide dasar dari metode HOG detection adalah Tampilan obyek lokal dan bentuk dapat dicirikan cukup baik oleh distribusi intensitas gradien lokal atau arah te*Pi*, hal ini diimplementasikan dengan membagi jendela gambar ke dalam daerah ruang-ruang kecil ("sel"), untuk setiap sel mengumpulkan histogram arah gradien lokal 1D nya atau berorientasi sudut terhadap *Pi*ksel dari sel. Gabungan masukan histogram membentuk representasinya. Untuk invarian yang lebih baik terhadap pencahayaan, bayangan, juga berguna untuk menormalkan kontras tanggapan lokal sebelum menggunakannya. Hal ini dapat dilakukan dengan mengumpulkan

suatu ukuran histogram lokal "energi" terhadap wilayah spasial yang lebih besar ("blok") dan menggunakan hasilnya untuk menormalkan semua sel didalam bloknya. Gambar 2.6 merupakan rantai jalannya metode HOG.



Gambar 2. 6 Rantai pendeteksian objek menggunakan metode HOG

(Dalal, 2005:3)

2.2.1 *Normalize gamma and color*

Proses *Normalize gamma and color* atau normalisasi gamma dan warna pada *frame* gambar dari video terjadi dalam *framework* HOG pada *OpenCV*. Normalisasi gamma dan warna pada *frame* gambar pada sistem ini menggunakan instruksi : *void normalize_hists*. Proses normalisasi gamma dan warna dari *frame* gambar pada *framework* HOG dalam *library OpenCV* dilakukan pada *frame-frame* gambar dari video yang ditangkap oleh *webcam*. Instruksi *OpenCV* ini akan menormalisasikan *gamma* dan warna pada *frame-frame* gambar tersebut.

2.2.2 *Computer Gradients*

Proses *computer gradients* atau perhitungan gradien pada *frame* gambar dari video yang ditangkap oleh *webcam* terjadi dalam *framework* HOG pada *OpenCV* seperti yang terlihat pada rantai pendeteksian HOG

detection. Komputasi gradien pada *frame* gambar menggunakan instruksi : *void cv::gpu::HOGDescriptor::computeGradient*. Proses komputasi gradien gambar pada *framework* HOG dalam *library* OpenCV dilakukan pada *frame-frame* gambar dari video yang ditangkap oleh *webcam*. Instruksi *OpenCV* ini akan mengkomputasikan gradien pada gambar-gambar tersebut.

2.2.3 *Weighted vote into spatial and orientation cells*

Pada rantai HOG *detection* terdapat proses *weighted vote into spatial and orientation cells* atau pemilihan bobot kedalam sel berorientasi dan spasial pada *frame* gambar dari video yang ditangkap oleh *webcam* yang terjadi dalam *framework* HOG pada *OpenCV*.

Pemilihan bobot kedalam sel berorientasi dan spasial pada *frame* gambar dari video yang ditangkap oleh webcam menggunakan instruksi *OpenCV : cv::gpu: HOGDescriptor::HOGDescriptor*. Proses pemilihan bobot kedalam sel berorientasi dan spasial pada *framework* HOG dalam *library OpenCV* dilakukan pada *frame-frame* gambar dari video yang ditangkap oleh *webcam*. Instruksi *OpenCV* ini akan melakukan pemilihan bobot kedalam sel berorientasi dan spasial pada *frame-frame* gambar tersebut.

2.2.4 *Contrast normalized over overlapping spatial blocks*

Proses *contrast normalized over overlapping spatial blocks* atau normalisasi kontras terhadap blok spasial yang bertumpukan pada *frame* gambar dari video yang ditangkap oleh webcam terjadi dalam framework

HOG pada *OpenCV* seperti yang terlihat pada rantai pendeteksian HOG detection. Normalisasi kontrasterhadap blok spasial yang bertumpukan pada openCV menggunakan instruksi: *void cv::gpu::HOGDescriptor:: computeBlockHistograms*. Proses normalisasi kontras terhadap blok spasial yang bertumpukan pada *framework* HOG dilakukan pada *frame-frame* gambar dari video yang ditangkap oleh *webcam*.

2.2.5 *Collect HOG's over detection windows*

Pada rantai HOG detection terdapat proses *Collect HOG's over detection windows* atau penghimpunan HOG terhadap jendela deteksi pada *frame* gambar dari video yang ditangkap oleh webcam yang terjadi dalam *framework* HOG pada *OpenCV*. Penghimpunan HOG terhadap jendela deteksi pada *frame* gambar dari video yang ditangkap oleh *webcam* menggunakan instruksi *OpenCV* : *voidcv::gpu::HOGDescriptor:: detectMultiScale*. Proses penghimpunan HOG terhadap jendela deteksi gambar pada *framework* HOG dalam *library OpenCV* dilakukan pada keseluruhan *frame* gambar dari video yang ditangkap oleh *webcam*. Instruksi *OpenCV* ini akan melakukan penghimpunan HOG terhadap jendela deteksi pada gambar-gambar tersebut.

2.2.6 Linier SVM

Dalam penelitian ini proses Linear SVM atau SVM linear pada gambar dari video yang ditangkap oleh webcam terjadi dalam framework HOG pada OpenCV seperti yang terlihat pada rantai HOG detection. SVM linear pada gambar menggunakan instruksi : *void cv::gpu:: HOG*

Descriptor:: setSVMDetector. Proses SVM linear dari gambar pada framework HOG dalam *library OpenCV* dilakukan pada *frame-frame* gambar dari video yang ditangkap oleh *webcam*. Instruksi OpenCV ini akan melinier SVM-kan gambar-gambar tersebut. Kemampuan melinier SVM-kan HOG detection diperoleh melalui *training* yang disimpan dalam database *OpenCV*. Pada proses ini tidak perlu melakukan *training* lagi karena sistem dapat menggunakan *database training* pada *OpenCV*.

Langkah-langkah penggunaan HOG*(Histogram of Oriented Gradients):*

a. Sudo pip install dlib
b. Library yang dibutuhkan

   import cv2

   import dlib

   import imutils

   from imutils import face_utils
c. *Sourcode* untuk pemanggilan

   cv2.HOGDescriptor()

   dlib.get_frontal_face_detector()

   hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
d. Mengatur program sesuai yang diinginkan
e. Gambar 2.7 merupakan hasil deteksi HOG.

Gambar 2. 7 Hasil Deteksi HOG

2.3 Bahasa *Python*

Bahasa pemrograman *Python* ini pertama kali dibuat oleh Guido van Rossum pada awal tahun 1990 di negeri Belanda sebagai pengganti bahasa pemrograman yang disebut ABC. Walaupun Guido adalah orang yang pertama kali menciptakan bahasa pemrograman ini, teta*Pi* bahasa pemrograman *Python* yang digunakan sekarang merupakan konstribusi dari berbagai sumber.

Gambar 2. 8 Logo Phyton

(Sumber : https://encrypted-tbn0.gstatic.com)

Gambar 2.8 merupakan logo dari bahasa *python*. Bahasa pemrograman *Python* merupakan bahasa pemrograman yang dapat dikembangkan oleh siapa

saja karena besifat *Open Source* atau dengan kata lain bahasa pemrograman ini gratis, dapat digunakan tanpa lisensi, dan dapat dikembangkan semampu yang dapat dilakukan.

Sebenarnya bahasa pemrograman *Python* ini mudah dipelajari karena penulisan sintaks yang lebih fleksibel. Selain itu, bahasa pemrograman *Python* ini memiliki efisiensi tinggi untuk struktur data level tinggi, pemrograman berorientasi objek lebih sederhana tetapi efektif, dapat bekerja pada multi platform, dan dapat digabungkan dengan bahasa pemrograman lain untuk menghasilkan aplikasi yang diinginkan. *Python* dikenal sebagai bahasa pemograman interpreter, karena *Python* dieksekusi dengan sebuah interpreter. Terdapat dua cara untuk menggunakan interpreter, yaitu dengan mode baris perintah dan modus script. Pada mode baris, perintah diketikkan pada shell atau *command line* dan *Python* langsung menampilkan hasilnya. Bila menggunakan *shell*, semua definisi yang telah dibuat baik fungsi atau variabel akan dihapus. Cara lain adalah dengan menyimpan perintah – perintah *python* dalam satu file, yang disebut selanjutnya sebagai *script* (Kadir,2005).

2.4 *Database Cloud*

Facebook *notification*, *Whatsapp*, Bbm, Disqus adalah contoh kecil aplikasi *real-time* yang sering dipakai. Saat *user* lain memberikan komentar atau mengirim pesan, pada saat itu juga mendapat notifikasi dan pesan. Secara general, jika ada input data, maka output pada saat itu langsung tampil.

Gambar 2. 9 Logo *Firebase*

Gambar 2.9 logo *firebase*. Orang yang mengoperasikan semua komputer *client* yang terhubung dengan aplikasi tersebut. Bahkan, bisa secara serentak. Walaupun *delay*, tidak akan lama waktunya (*Low Latency*). Untuk saat ini, yang masih hangat-hangatnya adalah web socket. Teknologi ini sudah di standarisasi tahun 2011 lalu, beberapa *browser* sudah mendukung *web socket*. Sudah ada beberapa contoh implementasinya (*library* dan cara bikin *socket* server) seperti Socket.io (Node), Ratchet (PHP), Tornado (*Python*), Socky (Ruby).

Solusi lain, adalah menggunakan layanan *Pi*hak ketiga. Cara ini lebih simple, dapat mengurangi waktu learning dan *research* secara drastis. *Firebase* bisa disebut sebagai layanan DbaaS (Database as a Service) dengan konsep realtime. Tidak hanya untuk simpan data, *Firebase* juga sediakan A*PI* untuk implementasi web *socket*.*Firebase* menyediakan library untuk berbagai *client platform*. *Browser* dengan *Javascript* A*PI* dan mobile pakai OBJ-C atau *Android* A*PI*.Bagi *web developer* memungkinkan untuk membuat aplikasi "Only HTML Css Js" karena sisi *server* dan *database* sudah *dihandle* oleh *Firebase*. *Firebase* menyediakan hosting untuk *static file* tersebut lengkap dengan fasilitas CDN dan SSL.Firebase juga dapat digunakan sebagai pelengkap aplikasi.

Langkah-langkah menggunakan *database firebase*:

a. Pastikan sudah *install firebase* di *device*

   Sudo *Pi*p install pyrebase

b. Daftar akun di firebase

c. Buat *new project*

d. Setelah buat new project akan didapatkan:

   -A*PI* key

   -Authdomain

   -databaseurl

   -storagebucket

   Kegunaan adalah ketika eksekusi pemanggilan alamat database

e. Pengolahan data

   -Push

   data = {"name": "Mortimer 'Morty' Smith"}

   db.child("users").push(data)

   -Set

   data = {"name": "Mortimer 'Morty' Smith"}

   db.child("users").child("Morty").set(data)

   -Update

   db.child("users").child("Morty").update({"name":"Mortiest Morty"})

   -Remove

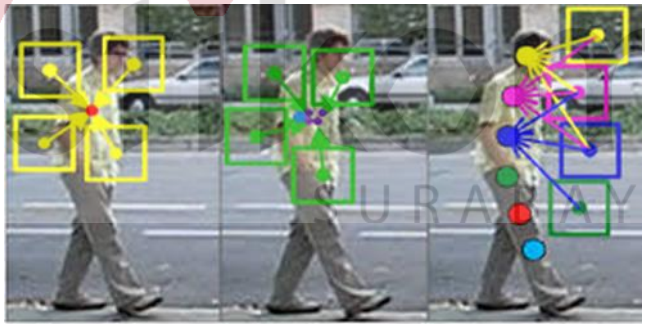
   db.child("users").child("Morty").remove()

2.5 *Computer Vision*

Computer Vision merupakan salah satu cabang ilmu pengetahuan yang bertujuan untuk membuat suatu keputusan yang berguna mengenai objek fisik nyata dan keadaan berdasarkan sebuah gambar atau citra (Sha*Pi*ro & Stockman, 2001). *Computer vision* merupakan kombinasi antara :

a. Pengolahan Citra Pengolahan Citra (*Image Processing*) merupakan bidang yang berhubungan dengan proses perubahan pada citra agar mendapatkan kualitas citra yang lebih baik.

b. Pengenalan Pola Pengenalan Pola (*Pattern Recognition*) merupakan bidang yang berhubungan dengan proses identifikasi objek pada citra atau interpretasi citra, yang bertujuan untuk mengekstrak informasi yang disampaikan oleh citra. Gambar 2.10 merupakan contoh identifikasi objek gambar pada pengolahan citra.



Gambar 2. 10 Contoh identifikasi objek

(Sumber : www.nottingham.ac.uk/research/groups/cvl/images/home-page-rotate/home-banner-3.png)

2.5 Open CV

. OpenCV adalah sebuah library yang bersifat *open source* (bebas digunakan untuk hal yang bersifat akademis maupun komersial) yang berisi banyak fungsi pemrograman untuk tehknologi computer vision secara *real-time*. *Library* ini ditulis dalam C dan C ++ dan berjalan di bawah Linux, Windows dan Mac OS X. OpenCV ditulis dalam C yang dioptimalkan dan dapat memanfaatkan prosesor *multicore*.

Gambar 2. 11 Logo OpenCV

(Sumber : https://www.meccanismocomplesso.org)

Gambar 2.11 merupakan logo dari OpenCV. Contoh aplikasi dari OpenCV antara lain interaksi manusia dengan komputer, pengenalan wajah, pengenalan gerak, struktur dari gerakan, robotika (Helmirawan, 2012). Berikut fitur – fitur yang terdapat pada OpenCV antara lain:

1. Manipulasi data *image* (alokasi, rilis, duplikasi, pengaturan, konversi).
2. Image dan I/O video (masukan berbasis file dan kamera, keluaran *image*/video file).
3. Manipulasi matriks dan *vector* serta aljabar linier (Produk, solusi, *eigenvalues*, SVD).
4. Beragam struktur data dinamis (daftar, baris, grafik).

5. Dasar pengolahan citra (filter, deteksi te*Pi*, deteksi sudut, pengambilan sampel dan interpolasi, konversi warna, operasi morfologi, *histogram*).
6. Analisis struktur (komponen yang berhubungan, pengolahan kontur, transformasi jarak, variasi momen, transformasi Hough, perkiraan *polygonal*, menyesuaikan garis, *Delaunay triangulation*).
7. Kalibrasi kamera (menemukan dan menelusuri pola kalibrasi, dasar estimasi matriks, estimasi homografi, korespondensi stereo).
8. Analisis gerakan (*optical flow*, segmentasi gerakan, penelusuran).
9. Pengenalan objek (metode *eigen*, HMM).
10. *Graphical User Interface* Atau GUI (menam*Pi*lkan image/ video, penanganan *mouse* dan *keyboard*, *scroll-bars*).
11. Pelabelan image (garis, *polygon*, gambar teks).

Untuk menggunakan *Raspberry Pi* memerlukan operating system(contoh OS : windows, linux,mac ,Unix dst) yg dijalankan dari SD *card pad board Raspberrry* tidak seperti pada board microcontroller AVR yg selama ini kita pakai 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 tidak dari lokasi lain. OS yang bisa di jalankan di *Raspberry board* antara lain Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Inferno, Raspbian OS, RISC OS dan *Slackware* Linux. Jadi dalam menggunakan *microcomputer Raspberry Pi* ini seperti menggunakan PC yg berbasis linux plus yg mempunyai *input output* digital seperti yg ada di board *microcontroller*.

2.6 *Android* Studio

Android studio adalah IDE(*Integrated Development Environment*) resmi untuk pengembangan aplikasi *Android* dan bersifat *opensource* atau gratis. Peluncuran *Android* Studio ini diumumkan oleh *Google* pada 16 mei 2013 pada *event* Google I/O *Conference* untuk tahun 2013.Sejak saat itu, *Android Studio* mengantikan *Eclipse* sebagai IDE resmi untuk mengembangkan *aplikasi Android* . *Android studio* sendiri dikembangkan berdasarkan IntelliJ IDEA yang mirip dengan *Eclipse* disertai dengan ADT *plugin* (*Android Development Tools*).

Gambar 2. 12 *Android Studio*

(Sumber : https://techcrunch.com)

Gambar 2.12 merupakan logo *Android studio*. *Android studio* memiliki fitur

a. Projek berbasis pada *Gradle Build*

b. *Refactory* dan pembenahan *bug* yang cepat

c. *Tools* baru yang bernama "*Lint*" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.

d. Mendukung *Proguard And App-signing* untuk keamanan.

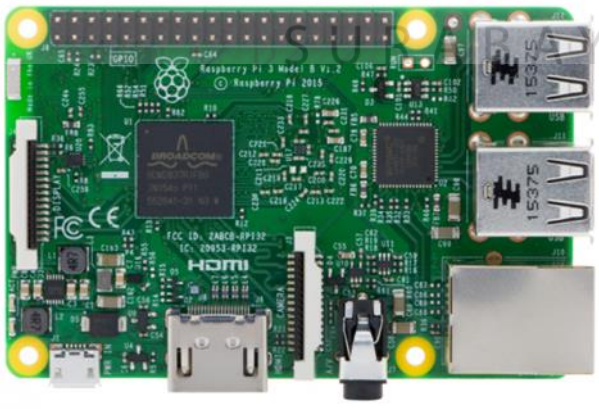
e. Memiliki GUI aplikasi *Android* lebih mudah

f. Didukung oleh *Google Cloud Platfrom* untuk setiap aplikasi yang dikembangkan.

2.8 *Raspberry Pi* model B

Raspberry Pi 3 adalah komputer *Single Board Circuit* (SBC) yang memiliki ukuran kecil setara dengan ukuran ATM, KTP atau Kartu Kredit. Perangkat ini dapat digunakan untuk menjalankan program perkantoran, permainan, pemutar video beresolusi tinggi dan lain sebagainya. *Raspberry Pi* 3 ini sangat cocok untuk pada developer karena ukurannya yang kecil teta*Pi* memiliki kinerja yang baik. Selain itu kelebihan *Raspberry Pi* 3 dibanding dengan *mikrokontroler* lain yaitu mempunyai Port I/O, *display* HDMI, koneksi USB dan banyak kelebihan lainnya. *Raspberry Pi* 3 dikembangkan oleh yayasan nirlaba, *Raspberry Pi Foundation*, yang anggotanya merukan ahli komputer dari Universitas Cambridge, Inggris.

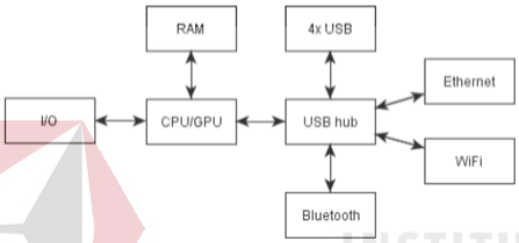
Terdapat 2 versi dari *Raspberry* yaitu model A dan model B. Perbedaan dari model A dan model B adalah kapasitas penyimpanannya, model B menggunakan penyimpanan yang lebih besar dari pada model A. Selain itu model B juga sudah menggunakan *Ethernet*, *Bluetooth* dan *Wl-FI* sebagai sarana komunikasi. Gambar 2.13 merupakan *Rasberry Pi* 3 model B.



Gambar 2. 13 *Rasberry Pi* 3 model B

(Sumber : https://www.element14.com)

Penyimpanan data tidak didesain untuk menggunakan cakram keras atau *solid-state drive*, melainkan menggunakan mikro SD untuk menjalankan sistem dan media penyimpanan dalam jang panjang. Pengguna dianjurkan menggunakan *Flashdisk* atau penyimpanan lain untuk menyimpan *file* yang berukuran besar. Perangkat *Raspberry Pi* 3 ini adalah hasil perkembangan dari perangkat yang dahulu yaitu *Raspberry Pi* 2. Banyak sekali perubahan yang terjadi mulai dari RAM, *processor* dan lain sebagainya.



Gambar 2. 14 Blok diagram *Raspberry Pi 3 model B*

Gambar 2.14 merupakan Blok *Diagram Raspberry Pi* 3 Model B, mempunyai input dan output antara lain:

a. HDMI, sebagai penghubung ke *LCD Monitor* yang mempunyai *port* HDMI atau dengan *converter* HDMI ke VGA untuk bisa digunakan pada *LCD Monitor* yang memiliki *port* VGA.

b. *Audio output* menggunakan jack 3,5mm, sebagai output untuk *speaker* atau sejenisnya yang menggunkan *jack* 3,5mm.

c. 4 buah *port USB*

d. 40 *Pi*n I/O

e. CSI (*Camera Serial Interface*), digunakan khusus untuk kamera yang menggunakan CSI sebagai antarmuka

f. DSI (*Display Serial Interface*)

g. *LAN port* (*Network*)

h. *WiFi*

i. *Bluetooth*

j. SD *Card slot* untuk SD *Card memory* penyimpanan sistem operasi juga sebagai penyimpanan file.

2.8.1 Spesifikasi *Raspberry Pi* 3 model B

Tabel 2. 1 Spesifikasi *Raspberry Pi* 3

Processor		Broadcom BCM2387 *chipset.*
		1.2GHz *Quad-Core* ARM Cortex-A53
		802.11 *b/g/n Wireless LAN and Bluetooth 4.1*
		(Bluetooth Classic and LE)
GPU		*Dual Core VideoCore IV® Multimedia Co-*
		Processor. Provides Open GL ES 2.0, *hardware-*
		accelerated OpenVG, and 1080p30 H.264 *high-*
		profile decode. Capable of 1G*Pi*xel/s, 1.5Gtexel/s
		or 24GFLOPs *with texture filtering and* DMA
		Infrastructure
Memory		1GB LPDDR2

Operating System	*Boots from Micro SD card,* □ Raspbian

	Ubuntu MATE Snappy Ubuntu Core Windows 10 IoT Core RISC OS Debian Arch Linux ARM *Android*
Dimensions	85 x 56 x 17mm
Power	*Micro USB socket* 5 Volt 2.5A

2.8.2 Konektor *Raspberry Pi* 3 model B

Dalam *Raspberry Pi* 3 Model B, terdapat beberapa konektor yang dapat digunakan sebagai I/O, yang dapat dilihat pada Tabel 2, dan gambar I/O ditunjukkan pada Gambar 2.15.

Tabel 2. 2 Konektor *Raspberry Pi* 3

Ethernet	10/100 *BaseT Ethernet socket*
Video Output	HDMI *(rev* 1.3 & 1.4
	Composite RCA (PAL and NTSC)
Audio Output	*Audio Output* 3.5mm *jack,*
	HDMI USB 4 x USB 2.0 *Connector*
GPIO Connector	40-*Pi*n 2.54 mm (100 mil) *expansion header:* 2x20 *strip Providing* 27 G*PIO Pins as well as* +3.3 V, +5 V *and* GND *supply lines*
Camera Connector	15-*Pin* MIP*I Camera Serial Interface* (CSI-2*)*
Display Connector	*Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock Lane*
Micro sd slot	*Push/pull Micro SDIO*

Gambar 2. 15 I/O *Raspberry Pi* 3

2.9 *Raspberry Pi Camera Module Supports Night Vision*

Modul *camera* dengan *Adjustable-Focus* yang cocok untuk semua tipe *Raspberry Pi*. Modul camera ini sudah dilengka*Pi* dengan *Infrared LED Board* yang mendukung fitur *Night Vision* sehingga memungkinkan untuk melihat dalam kondisi cahaya rendah. Contoh kamera seperti diperlihatkan pada gambar 2.16.

Gambar 2. 16 *Pi camera module night vision*

(Sumber : http://ecadio.com)

Spesifikasi:

a. Dapat digunakan untuk semua tipe *Raspberry Pi*.

b. Menggunakan *sensor camera OV5647* dengan resolusi 5MP.

c. Tersedia 4 lubang sekrup yang digunakan untuk *Infrared LED board* sekaligus memberikan catu daya 3,3 V.

d. Antarmuka: 15-*Pi*n CSI (Camera Serial *Interface*).

e. Dimensi: 25mm x 24mm.

f. Paket penjualan: R*Pi Camera* (F) x1, *Infrared LED Board* (B) x2, 15-*Pi*n FFC (opposite sides contact) x1.

Spesifikasi kamera:

a. Ukuran CCD: 1/4inch

b. *Aperture* (F): 1.8

c. *Focal Length*: 3.6mm (adjustable)

d. Diagonal: 75.7°

e. Resolusi sensor: 1080p

2.10 Jaringan

Dibutuhkannya konesi Internet untuk pengupload hasil video hasil deteksi yang telah tersimpan di memori *Raspberry* ke drive peyimpanan Firebase. Pada intinya menggunakan internet atau *server Lan* atau langsung sumber wifi,atau lainnya tidak masalah, yang terpenting adalah bisa mengunggah data gambar ke *Firebase*.

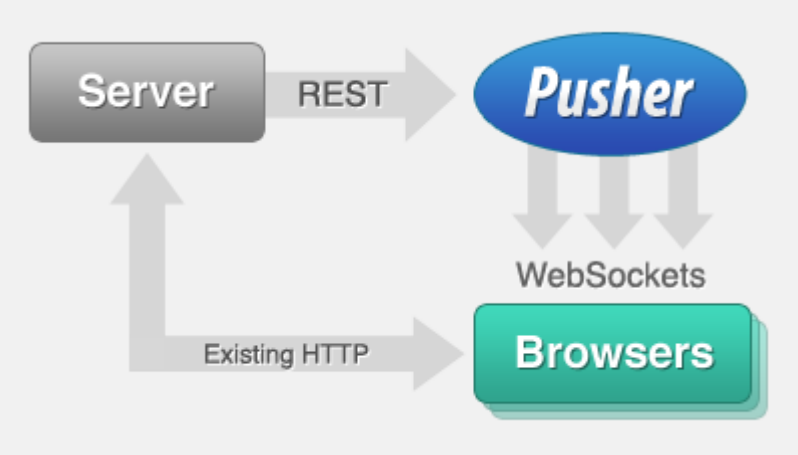
Gambar 2. 17 Jaringan *Wifi* dan LAN

2.11 *Pusher*

Beams adalah A*PI* untuk mengirim pemberitahuan push ke aplikasi iOS dan *Android*. Ini termasuk layanan yang diinangi dan SDK khusus untuk mengelola *token push* perangkat aplikasi yang dinginkan. Untuk mengirim pemberitahuan, dapat mengotentikasi perangkat sebagai Pengguna yang Diotentikasi untuk pemberitahuan pribadi kepada pengguna tertentu di seluruh perangkat mereka atau dapat berlangganan perangkat ke Minat Perangkat publik untuk pengiriman pemberitahuan *batch*.

Gambar 2. 18 Logo *Pusher*

Gambar 2.18 Konsol *Debug* seketika dan Wawasan untuk melacak kesehatan layanan notifikasi. Pusher mengandalkan *Firebase Cloud Messaging* (FCM) untuk mengirimkan pemberitahuan push kepada pengguna aplikasi *Android* atas nama program/*project* yang berkaitan. Saat mengirimkan pemberitahuan push menggunakan kredensial FCM yang berkaitan. Setelah melalui proses mendapatkan Kunci *Server* FCM dan bagaimana memberikannya kepada Pusher.



Gambar 2. 19 Jalannya *Pusher*

(https://medium.com/@ranggaantok/laravel-pusher-real-time-notification-e8a0012a25c3)

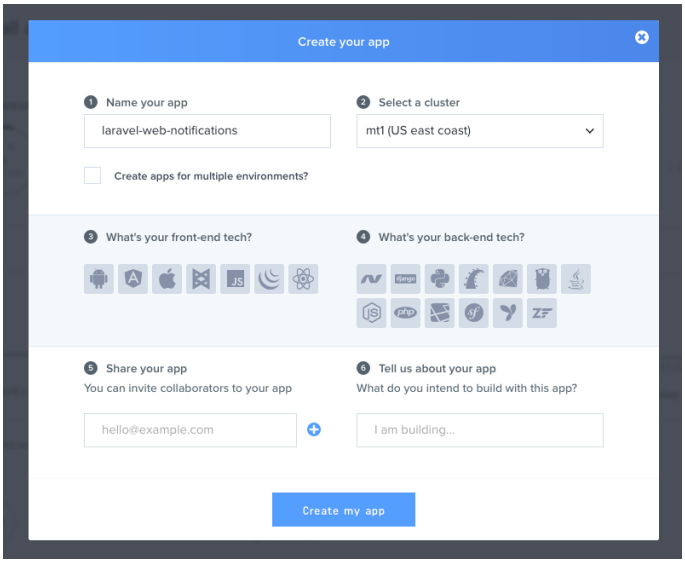
Pusher merupakan salah satu penyedia layanan *web socket* yang populer. dengan menggunakan *third party* maka kita tidak perlu mendevelop *web socket* sendiri pada *server* kita. mengapa kita perlu *websocket*. *websocket* digunakan untuk komunikasi 2 arah. protokol http yang selama ini digunakan hanya berkomunikasi secara satu arah client mengirim *request* lalu server mengembalikan respon. lalu bagaimana jika ingin mebuat sebuah *website* yang *realtime*. Gambar 2.19 merupakan contoh cara kerja pusher yang menyediakan wadah terminal *server* ke *browser* untuk memberi text *message*/notifikasi. *Browser* disini yang dimaksud adalah *Firebase Cloud*.

Langkah-langkah menggunakan beams pusher:

a. Menginstall php SDK untuk pusher

```
import com.pusher.pushnotifications.PushNotifications;
```

b. Membuat akun pusher



Gambar 2. 20 Tampilan *login firebase*

Disini pusher sudah bisa digunakan banyak bahasa sehingga memudahkan *user* untuk menggerakan project antar bahasa. Setelah mendaftar dan mengatur project akan didapatkan id dan *key id* untuk pengalamatan ketika menggunakan pusher di program atau *project* yang dikaitkan.

c. Connect to pusher

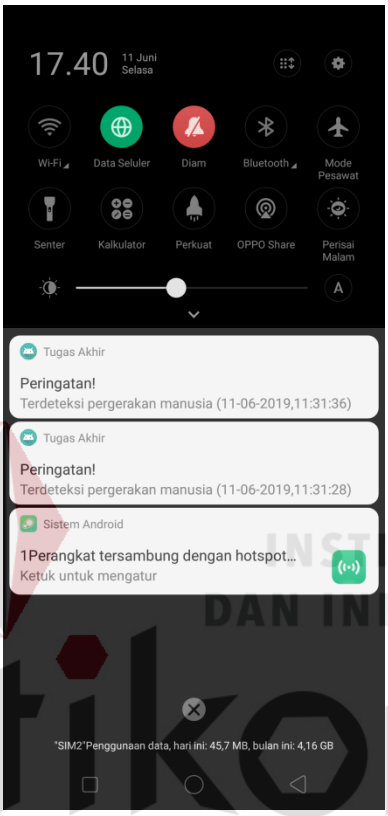
```
pn_client = PushNotifications(
  instance_id='3d84516d-7ba8-44d3-bf6f-873cdb9ec857',

secret_key='5FA28A093A6B428BECD6E952670F491F0993913F7C5166889DE
3D24C6E5F60D4',
)
```

Disini untuk menghubungkan pusher sesuai id dan key yang telah diberikan ketika membuat akun pusher.

d. Menulis notifikasi yang akan ditam*Pi*lkan

Pn_client = {"data": dataint, "thumb_image": dataurl, "date_project": current_datetime.strftime("%d-%m-%Y"), "time_project": current_datetime.strftime("%H:%M:%S"), "type_motion": motion}

e. Hasil notifikasi



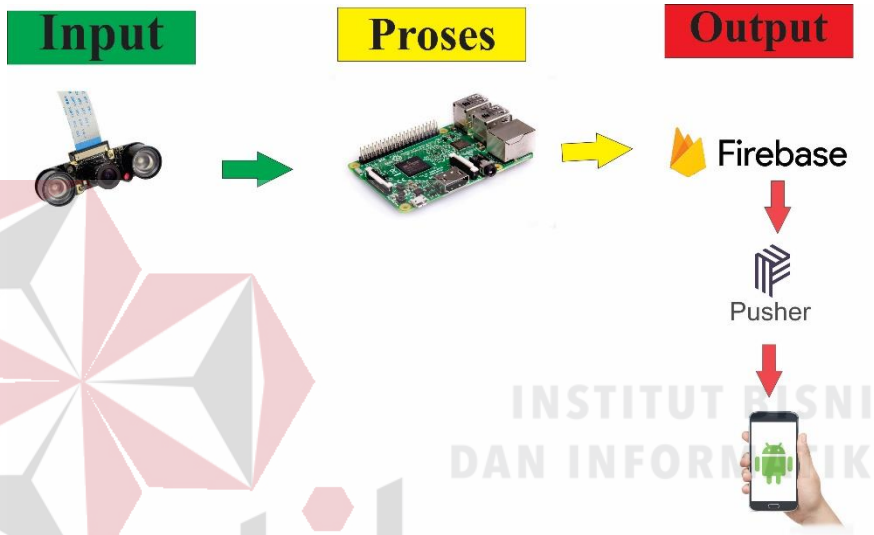
Gambar 2. 21 Tampilan jendela notifikasi

BAB III

METODE PENELITIAN

3.1 Metodologi Penelitian

Secara umum perancangan perangkat sistem meliputi unit masukan dan keluaran pada mikrokontroler. Perancangan Blok Diagram pada perancangan ini dapat dilihat pada gambar 3.1 dibawah ini.



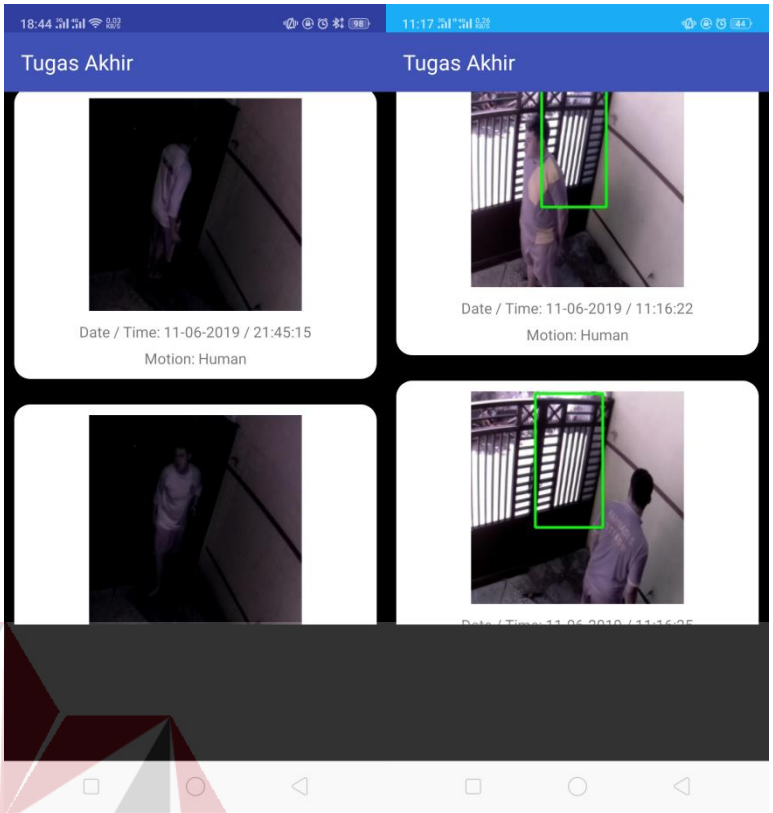
Gambar 3. 1 Blok diagram perangkat sistem

Tiap-tiap bagian dari diagram blok sistem pada gambar diatas dapat dijelaskan sebagai berikut :

1. Input pada *Mikrokontroler* :

- *Pi* Kamera disertai lampu *infrared*

Pi Kamera sebagai input gambar yang akan diambil dan diolah. Tentunya nantinya kamera tersebut dilengka*Pi* dengan mode sinar infrared agar kamera masih berfungsi degan baik ketika malam hari. Posisi kamera nantinya akan ditempatkan dipan objek yang akan dipantau. Gambar 3.2 membuktikan jika menggunakan kamera tersebut tidak didapatkangangguan pencahayaan.



Gambar 3. 2 Gambar yang didapatkan malam (kiri) dan siang (kanan)

-Program pemanngilan kamera

```
from imutils.video import VideoStream;
cap = VideoStream(usePiCamera=args["Picamera"] > 0).start();
frame = cap.read();
frame = cv2.resize(frame, (240,240));
 cv2.imshow("frame", frame);
```

2. Proses pada *Mikrokontroler* :

Raspberry disini digunakan untuk pemprosesan data gambar yang diambil oleh kamera.pemprosesan antar *frame* meliputi

a. Metode *substraction*

Metode *substraction* merupakan proses untuk menemukan objek pada gambar dengan cara membandingkan gambar yang ada dengan sebuah model latar belakang (Ardhianto,Hadikurniawati,&Budiarso,2013). Jadi, disini metode ini dimanfaatkan untuk mendeteksi objek yang bergerak. Gambar 16 merupakan contoh hasil pergerakan yang di tangkap oleh background substraction. Gambar 3.3 adalah hasil *background substraction* dan inilah yang menjadi *trigger* atau pemicu jika didapatkan suatu gerakan dan dilanjutkan ke pengolahan dengan metode HOG untuk dideteksi bahwa gerakan tersebut manusia atau bukan. Nilai tetapan/ nilai *Treshold* adalah 60, jika dimasukkan rumus hasil perhitungan *Background Substraction > threshold.*

Gambar 3. 3 Hasil *background substraction*

Program eksekusi metode substraction:

```
subtractor = cv2.createBackgroundSubtractorMOG2(history=20,
varThreshold=60, detectShadows=True);
mask = subtractor.apply(frame) ;
```

```
gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY);
whitepxl = np.sum(mask > 0): #jika terjadi gerakan
  if whitepxl > 100 and delay < 1:
  #masuk_HOG
```

b. Metode HOG

Metode HOG merupakan proses untuk menemukan objek manusia pada data gambar yang telah didapatkan dari hasil metode *background substraction*. Sehingga bisa diketahui apakah didata gambar tersebut terdapat manusia atau tidak.

Program eksekusi metode HOG:

```
rects, weights = hog.detectMultiScale(gray_frame, winStride=winStride,
  padding=padding, scale=args["scale"], useMeanshiftGrou*Pi*ng=meanShift)
  if delay_nonhuman < 1:
      datastr = str(dataint) + ".jpg"
      cv2.imwrite(datastr, frame)
      current_datetime = datetime.datetime.now()
      up_img(datastr, current_datetime, "Non-Human")
      dataint = dataint + 1
      delay_nonhuman = delay_nonhuman + 1
      print("Motion : Non Human")
```

```
            elif delay_nonhuman > 0:
                delay_nonhuman = delay_nonhuman + 1
                if delay_nonhuman > 300:
                    delay_nonhuman = 0
            if len(rects) > 0:
                print("===ALERT===")
                datastr = str(dataint) + ".jpg"
                cv2.imwrite(datastr, frame)
                current_datetime = datetime.datetime.now()
                up_img(datastr, current_datetime, "Human")
                my_stream =
db.child("dataimage").child(str(dataint)).child("data").stream(stream_hand
ler,None)
                dataint = dataint + 1
                delay = delay + 1
                end_time = time.time()
                print("Motion : Human")
                print("Waktu proses: ", end_time-start_time)
        elif delay > 0:
            delay = delay + 1
            if delay > 60:
                delay = 0
                print("Scanning...")
```

c. Proses pada memori

Memori yang digunakan *Raspberry* adalah menjadi penyimpanan sementara sebelum dikirim di database. Ketika terdeteksi suatu gerakan manusia akan terekam dan tersimpan secara otomatis di memori dan ketika data berhasil dikirim ke memori penyimpanan terahir/database maka data akan dihapus secara otomatis untuk mengurangi beban kapasitas *Raspberry*.

Program eksekusi pada memori:

```
db.child("dataimage").child(dataint).set(datajson) #send data ke firebase
    print("Data ke " + str(dataint) + " telah diupload")
    #dikomen
    os.remove(datastr)
```

3. Proses pada internet

Internet betugas sebagai jembatan antara *Raspberry* dengan database firebase agar hasil data tersimpan dengan baik. Waktu pegiriman data dipengaruhi banyaknya data atau bisa juga dari kecepatan koneksi internet.

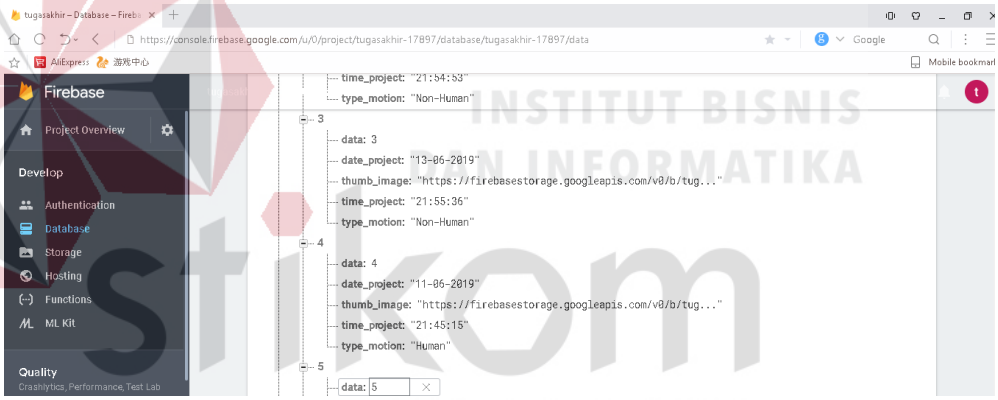
Program pengiriman data:

```
storage.child("images/"+datastr).put(datastr) #send data image
    dataurl = storage.child("images/"+datastr).get_url(1)
    datajson = {"data": dataint, "thumb_image": dataurl, "date_project": current_datetime.strftime("%d-%m-%Y"), "time_project": current_datetime.strftime("%H:%M:%S"), "type_motion": motion}
    db.child("dataimage").child(dataint).set(datajson) #send data ke firebase
    print("Data ke " + str(dataint) + " telah diupload")
    #dikomen
```

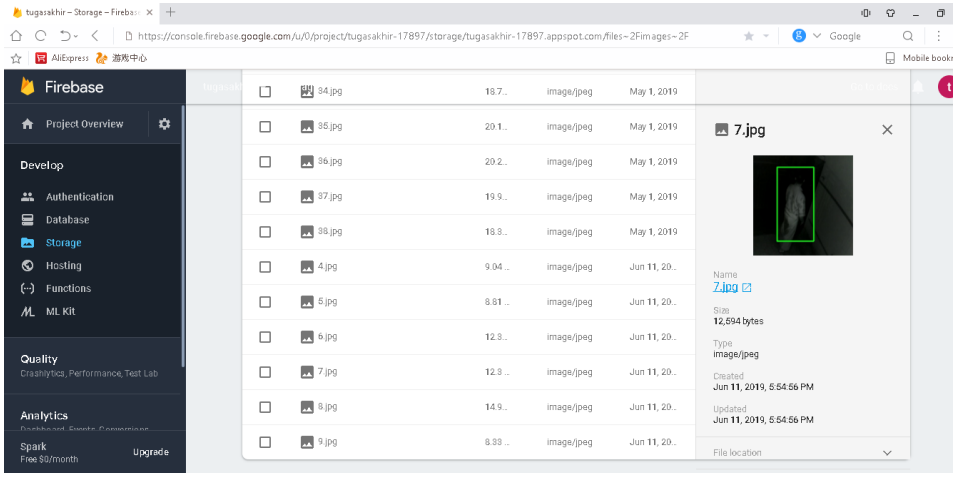
```
os.remove(datastr) #delete
print("Data ke " + str(dataint) + " telah dihapus")
global isibody
isibody = "Terdeteksi pergerakan manusia (" + current_datetime.strftime("%d-%m-%Y,%H:%M:%S") + ")"
```

4. Output pada *firebase*

Firebase diharapkan menjadi tempat yang lebih baik dan aman daripada memori di *Raspberry Pi.firebase* juga diharapkan lebih mudahkan dan praktis buat para pengguna untuk melihat hasil data gambar.



Gambar 3. 4 Tampilan data *string* di *firebase*

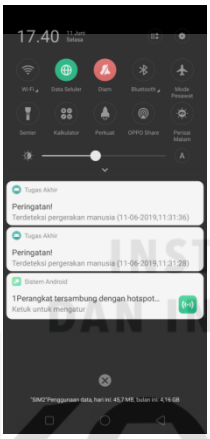


Gambar 3. 5 Tampilan data gambar di *firebase*

Digambar 3.4 dan 3.5 merupakan penam*Pi*lan hasil data yang berhasil diupload dari *Raspberry Pi*. Data dibagi menjadi dua yaitu data string dan gambar. Data *string* terlebih dulu sampai di *database* karena kecilnya ukuran data daripada data gambar.

5. *Output* pada notifikasi *Android mobile*

Tahapan ahir dari proses adalah memberi notifikasi kepada pengguna user *Android* mobile. Jadi, mereka bisa mengatahui jika terjadi suatu gerakan yang perlu dilihat.adapun sebagai contoh hasil notifikasi aplikasi seperti gambar 3.6.



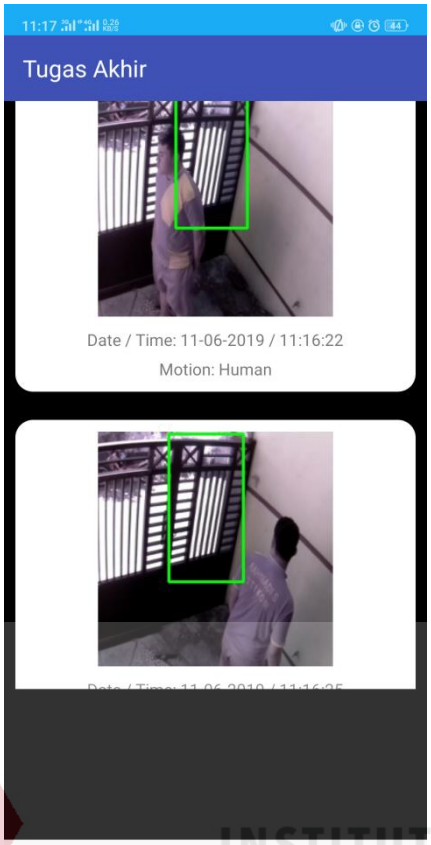
Gambar 3. 6 Gambar hasil notifikasi program

Dinotifikasi tersebut akan diberi jeda/delay agar data tidak rentet terus menerus dan data tidak *double*. Terdapat dua jenis notifikasi yang ditam*Pi*lkan yaitu *motion*/ gerakan manusia(*human*) atau bukan manusia(*non human*) yang dibedakan nanti jarak waktu *delay* perdata masuk. Adapun Tampilan program terdapat gambar 3.7 terdapat data gambar yang disertai detail waktu beserta keterangan pergerakan(*motion*).

Program untuk mengirim notifikasi dan peampilan updta data ke *Android*:

PushNotifications.start(getApplicationContext(), "3d84516d-7ba8-44d3-bf6f-873cdb9ec857");

```
PushNotifications.subscribe("hello");
if(ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)                    !=
PackageManager.PERMISSION_GRANTED)
requestPermissions(new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},PERMISSION
_REQUEST_CODE);

 databaseReference                                              =
FirebaseDatabase.getInstance().getReference().child("dataimage");
      databaseReference.keepSynced(true);
      mStorageRef                                               =
FirebaseStorage.getInstance().getReference().child("images/");

      user_list = findViewById(R.id.user_list);
      user_list.setHasFixedSize(true);
      user_list.setLayoutManager(new LinearLayoutManager(this));
```



Gambar 3. 7 Tampilan program *Android*

3.2 Perancangan Sistem

Perancangan sistem terdiri dari 2 proses yaitu proses pembuatan hardware dan hanya membutuhkan kamera *Raspberry Pi* v2.1 ke *Raspberry Pi* 3 dan ada beberapa yang harus diperhatikan untuk merancang sistem ini :

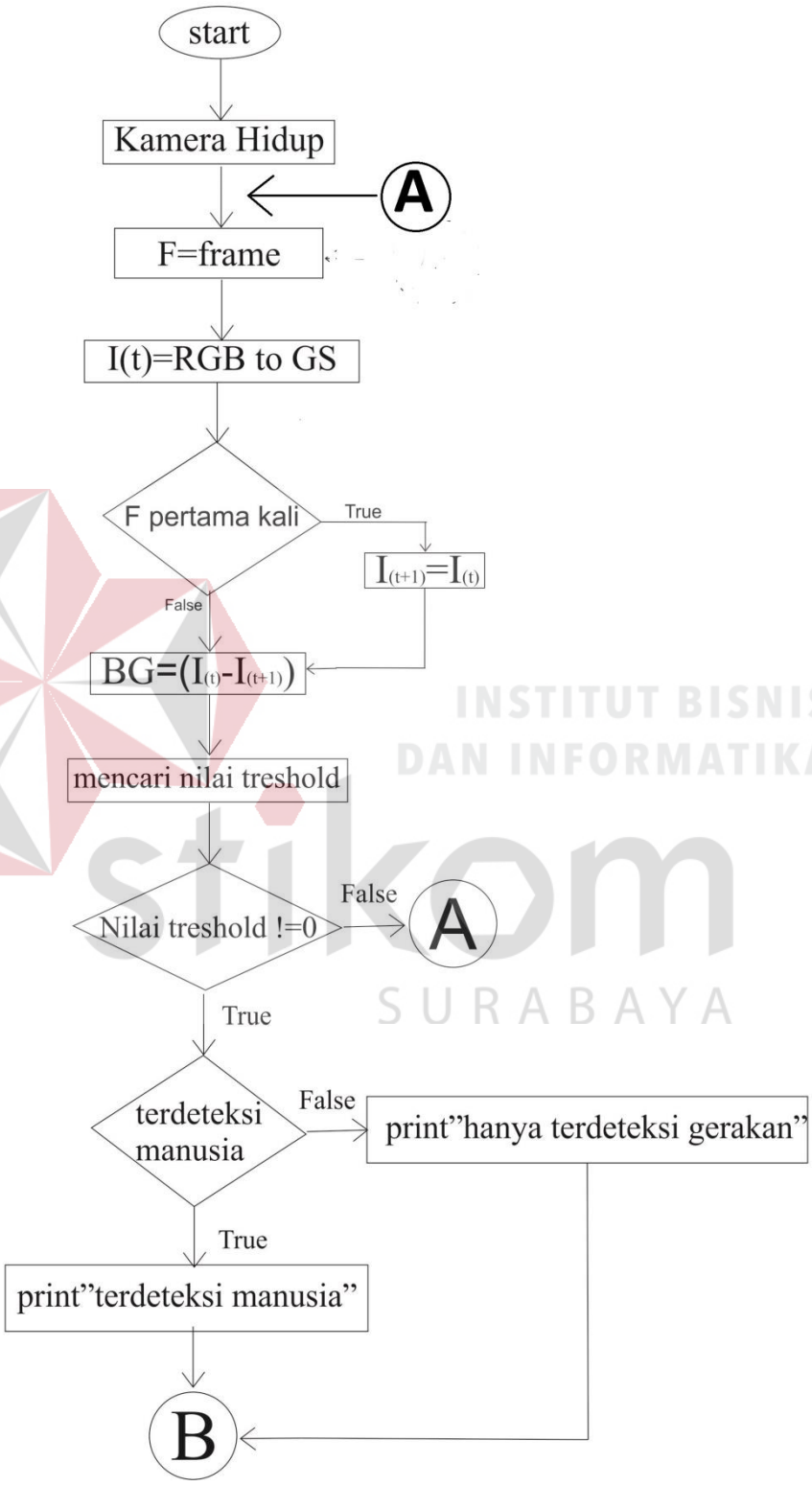
1. Perancangan *Mini pc* dan Kamera Keamanan

Perancangan penempatan kamera dan *Raspberry Pi* membutuhkan suatu wadah yang aman yang akan aman saat digunakan di segala medan. Tentunya juga menseting arah kamera agar area target dapat terdeteksi dengan baik. Pemberian lampu infrared agar kamera bisa berfungsi baik di malam hari / kurangnya pencahayaan

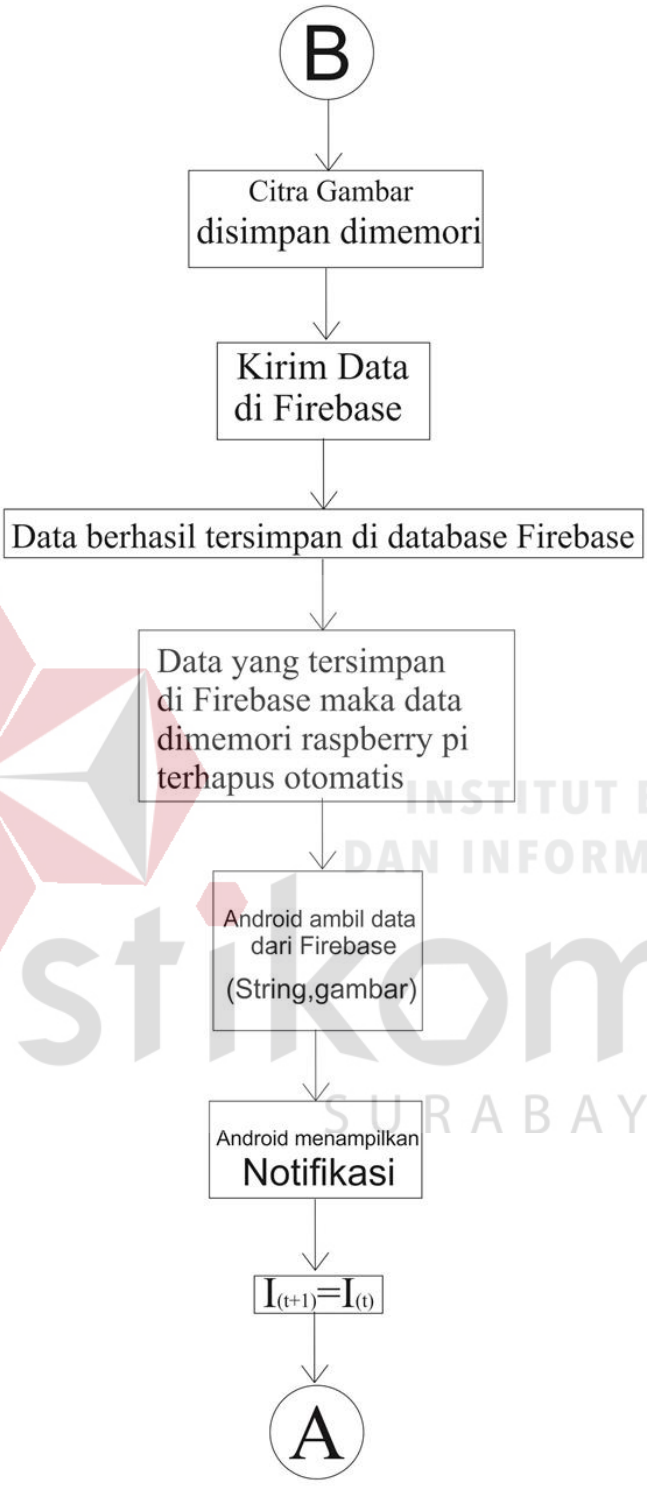
2. Merancang *database* tempat penyimpanan akhir data, agar dapat menyimpan hasil data gambar tersebut.
3. Pemrograman metode *background Substraction*

   Pemograman untuk kamera agar bisa pendeteksian pergerakan beserta deteksi HOG.
4. Pengujian *system* agar lebih baik lagi.

3.3 Flowchart



Gambar 3. 8 *Flowchart* 1 program *Raspberry Pi*



Gambar 3. 9 *Flowchart* 2 program *Raspberry Pi*

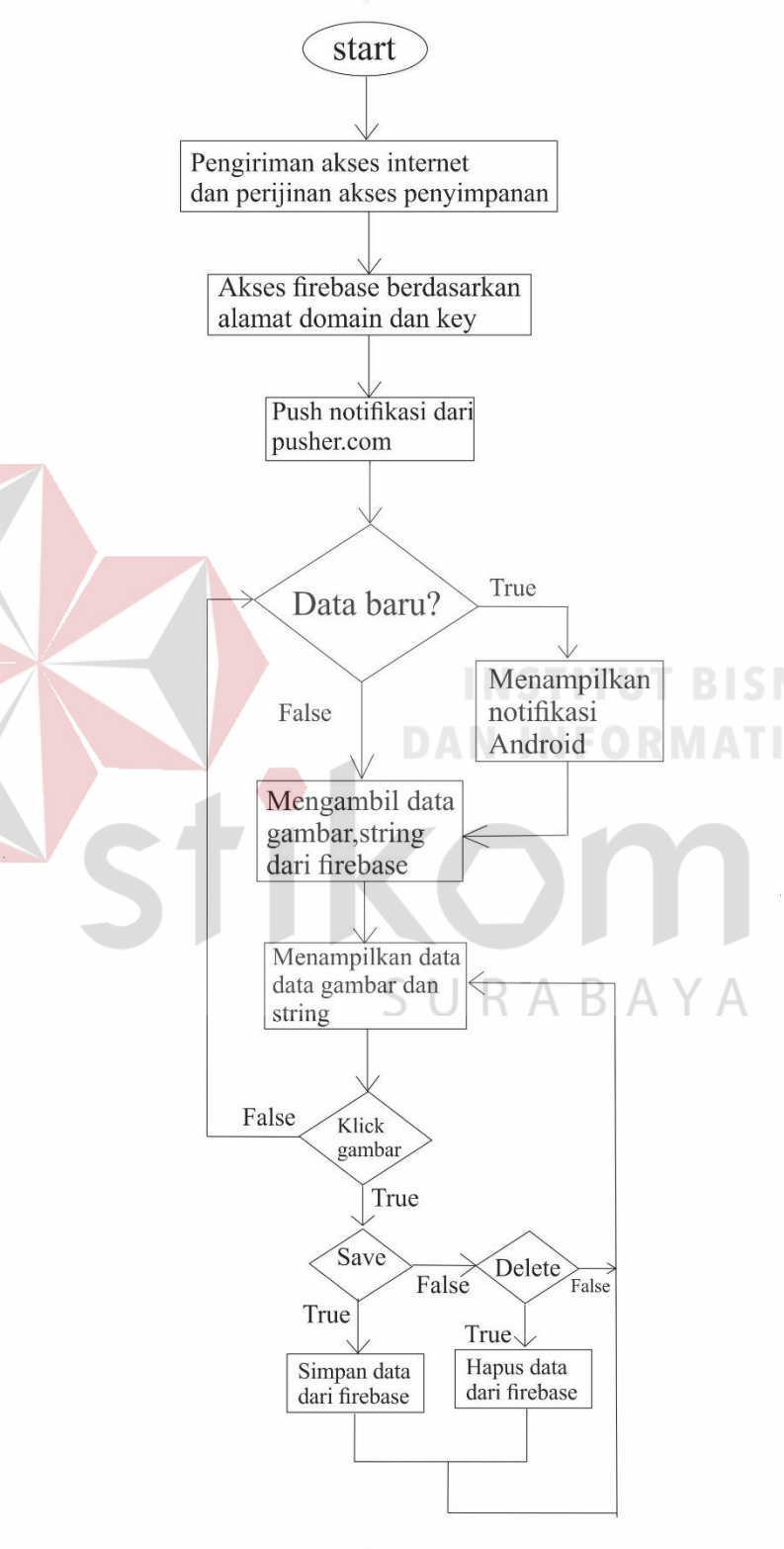
Gambar 3.8 dan 3.9 merupakan *flowchart* dari sistem Pengolahan data hasil kamera keamanan di database *cloud* berdasarkan deteksi gerakan

dengan metode *background substraction*. Dimulai dari *start* maka kamera akan hidup otomatis masis membuka *frame* baru dan selanjutnya *frame* akan diubah dari RGB ke *grayscale*. Setelah itu dilakukan percabangan ketika kondisi sudah ada atau tidak frame akan dibandingkan. Jika kondisi masih belum ada maka dilakukan inisialisasi untuk pertama kali agar ketika perhitungan *background substraction* ada perhitungan antara 2 frame. BG(*background substraction*) = I(t)-I(t+1) adalah perhitungan antara hitungan *frame* lama dikurangi dengan *frame* baru.

Kemudian menuju menghitung nilai per*Pi*ksel dalam satu *frame* yang dihasilkan dari perhitungan frame lama dengan *frame* baru/menghitung nilai threshold. Jika hasil perhitungan nilai *threshold* sama dengan 0 maka akan kembali ke inisialisasi *frame*/(A). Teta*Pi* jika nilai threshold tidak sama dengan 0 maka akan masuk percabangan deteksi manusia. Setelah masuk dipercabangan deteksi manusia didapatkan objek manusia maka akan membuat data string bahwa terjadi deteksi manusia dan dikirim ke(B).

Jika tidak ada terdeteksi objek manusia maka akan dibuat data string bahwa hanya terjadi sebuah pergerakan dan dikirim ke (B). Masuk keflowchart 2 *Raspberry* menyimpan data kememori. Kemudian data dikirim ke *database firebase*. Jika sudah tersimpan di database firebase dilakukan penghapusan otomatis data yang telah terkirim di *database*. Setelah itu sistem *Android* mengambil data keterangan notifikasi beserta gambar kemudian *Android* menam*Pi*lkan notifikasi. Kemudian *frame*

terahir kembali lagi ke awal untuk dibandingkan lagi dengan frame selanjutnya/kembali ke(A).



Gambar 3. 10 *Flowchart* program *Android*

Gambar 3.10 merupakan *flowchart* dari program *Android* yang digunakan untuk memberi notifikasi/peringatan dan menam*Pi*lkan data yang diperoleh dari alat kepada *user*. Dimulai dari start program apk akan secara otomais meminta persetujuan akses penyimpanan agar *user* dapat menyimpan data ke penyimpanan *Android* tersebut. Selanjutnya mengakses alamat domain dan *key database firebase* agar mendapatkan data yang tersimpan bisa ditam*Pi*lkan di jendela Tampilan program. Berikutnya adalah mengakses website server pusher.com agar membantu *database firebase* meberikan notifikasi berupa *message text* agar muncul di jendela notifikasi *Android*.

Dan dipercabangan apakah di database firebase terbaca ada data baru jika *true*, maka akan menam*Pi*lkan notifikasi di jendela notifikasi *Android* jika ada baru. Setalah itu mengambil data yang tersimpan di *database firebase* untuk ditam*Pi*lkan di jendela aplikasi *Android*. Ketika data gambar ditekan yang tam*Pi*l di jendela aplikasi maka akan tam*Pi*l 2 *Pi*lihan menu yaitu *save*/*delete*. Menu *save*/*delete* adalah untuk menghapus atau menyimpan data yang di*Pi*lih dari *database firebase* juga bukan hanya di aplikasi *Android*.

3.4 Metode pengujian

Pada bab ini menjelaskan tentang pengujian dari sistem yang telah dirancang, yang bertujuan untuk mengetahui apakah maksud dari seluruh tujuan sistem ini berjalan dengan baik atau tidak. Dimulai dari pengujian perangkat keras hingga penerapan pada sistem keamanan.

3.4.1 Pengujian *Raspberry Pi* 3 Model B

Pengujian *Raspberry Pi* 3 bertujuan untuk mengetahui apakah *Raspberry Pi* 3 dapat berjalan dengan baik serta dapat mengeksekusi program yang menggunakan bahasa pemrograman *Python*.

3.4.2 Pengujian *Pi Camera*

Pengujian *Pi Camera* bertujuan untuk mengetahui apakah *Pi Camera* dapat berjalan dengan baik pada *Raspberry Pi* 3 serta dapat merekam maupun memfoto suatu objek.

3.4.3 Pengujian Pendeteksian pergerakan

Pengujian ini digunakan untuk mengetahui seberapa besar kamera dapat mendeteksi dengan di beberapa percobaan objek yang bergerak atau tidak dan bisa membedakan objek manusia atau bukan.

3.4.4 Pengujian aplikasi notifikasi *Android*

Pada pengujian ini untuk mengetahui dapatkah aplikasi berbasis *Android* memberi perbedaan jeda waktu notifikasi jika terjadi pergerakan *human* atau *non human* dan mengirim data tangkapan kamera kemobile *Android* secara real time atau terdapat masalah ataupun terjadi delay.

3.4.5 Pengujian aplikasi *database firebase*

Pada pengujian ini digunakan untuk mengatahui dapatkah aplikasi in dapat memabantu pengguna sistem keamanan untuk menyimpan data

gambar dengan baik dan dapat dengan mudah oleh pengguna ketika ingin melihat hasil gambar.

BAB IV

HASIL DAN PEMBAHASAN

Pengujian sistem yang telah dilakukan penulis ini merupakan pengujian terhadap perangkat keras serta perangkat lunak dari sistem secara keseluruhan yang telah selesai dibuat untuk mengetahui kerja dari sistem berjalan dengan baik atau tidak.

4.1 Pengujian *Raspberry Pi* 3 Model B

4.1.1 Tujuan Pengujian *Raspberry Pi* 3 Model B

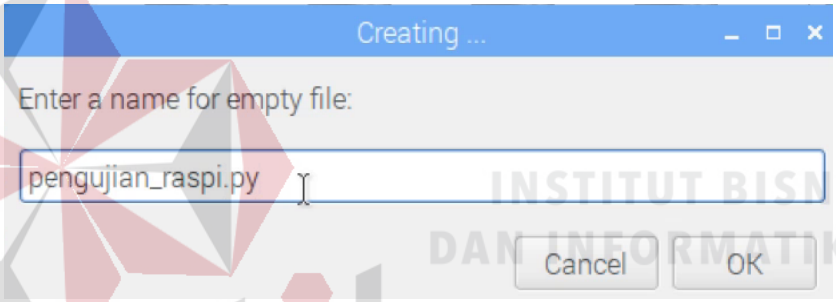
Pengujian *Raspberry Pi* 3 bertujuan untuk mengetahui apakah *Raspberry Pi* 3 dapat berjalan dengan baik serta dapat mengeksekusi program yang menggunakan bahasa pemrograman *Python*.

4.1.2 Peralatan Pengujian *Raspberry Pi* 3 Model B

1. *Raspberry Pi* 3 Model B
2. Monitor
3. Mouse
4. Keyboard
5. *Charger Handphone* (*Output* minimal 2A)

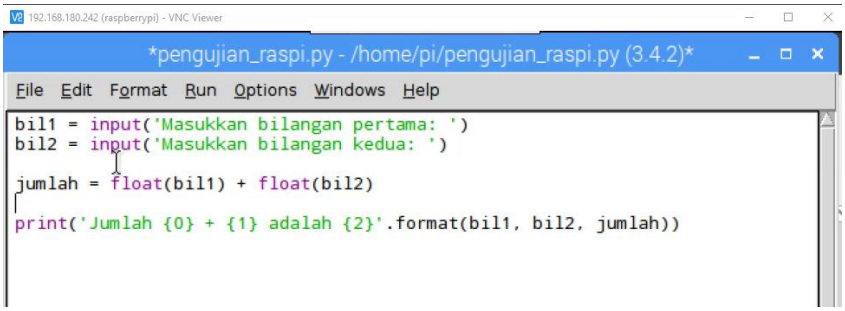
4.1.3 Prosedur Pengujian *Raspberry Pi* 3 Model B

1. Menyambungkan kabel HDMI dari monitor ke *Raspberry Pi* 3, menyambungkan juga keyboard dan mouse ke *Raspberry Pi* 3.
2. Menyalakan *Raspberry Pi* 3 dengan menyambungkan kabel *charger handphone*.
3. Membuka file manager.
4. Memilih *directory* yang diinginkan untuk menyimpan file *Python*. Kemudian mengeklik kanan, memilih *create new*, memilih *new file*.
5. Memberikan nama sesuai keinginan dengan ekstensi *.py pada file tersebut.



Gambar 4. 1 Pemberian Nama *File* Berekstensi *.py

6. Membuat program sederhana untuk pertambahan dua bilangan. Contoh program sederhana dapat ditunjukan pada Gambar 4.2.

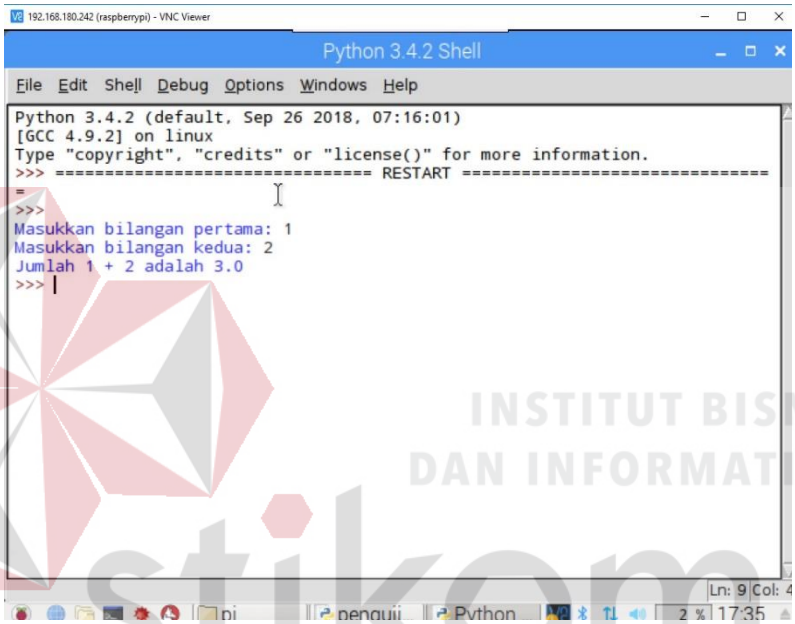


Gambar 4. 2 Contoh *Program* Sederhana Pertambahan Dua Bilangan Menggunakan *Python*

7. Kemudian *run program* tersebut.

4.1.4 Hasil Pengujian *Raspberry Pi* 3 Model B

Apabila *program* tidak ditemukan *error*, maka akan muncul *window* baru serta menam*Pi*lkan *program* yang diinginkan. Contoh hasil *running program* dapat ditunjukan pada Gambar 4.3.



Gambar 4. 3 Hasil *Running Program* Sederhana Pertambahan Dua Bilangan Menggunakan *Python*

4.2 Pengujian *Pi Camera*

4.2.1 Tujuan Pengujian *Pi Camera*

Pengujian *Pi Camera* bertujuan untuk mengetahui apakah *Pi Camera* dapat berjalan dengan baik pada *Raspberry Pi* 3 serta dapat merekam maupun memfoto suatu objek.

4.2.2 Peralatan Pengujian *Pi Camera*

1. *Raspberry Pi* 3 Model B

2. *Pi Camera module support night vision*
3. Monitor
4. Mouse
5. Keyboard
6. *Charger Handphone* (*Output* minimal 2A)

4.2.3 Prosedur Pengujian *Pi Camera*

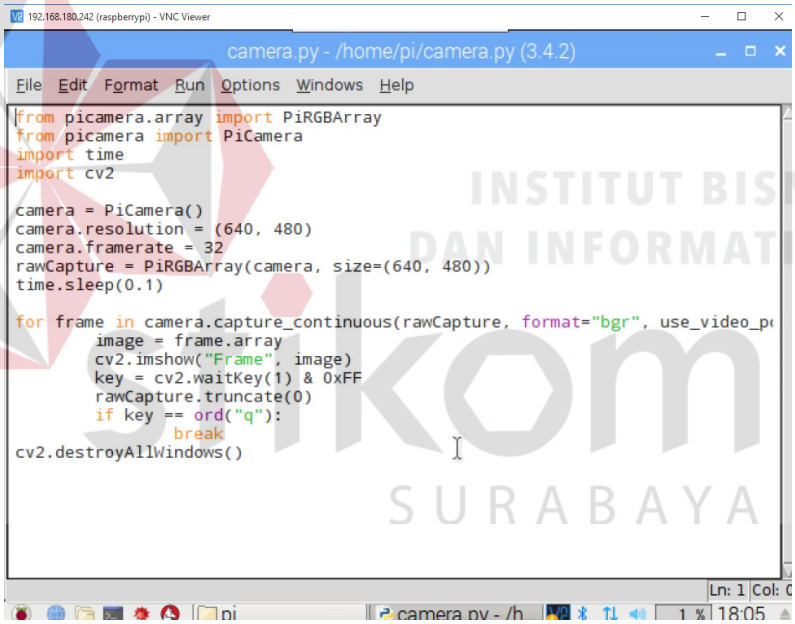
1. Menyambungkan kabel HDMI dari monitor ke *Raspberry Pi* 3, menyambungkan juga keyboard dan mouse ke *Raspberry Pi* 3.

   Menyambungkan *Pi Camera module support night vision* ke *Raspberry Pi* 3.

Gambar 4. 4 Pemasangan *Pi Camera* ke *Raspberry Pi*

2. Menyalakan *Raspberry Pi* 3 dengan menyambungkan kabel *charger handphone*.
3. Mengeklik *Raspberry Pi*, memilih *Preferences*, memilih *Raspberry Pi Configuration*.

4. Memilih menu *Interfaces*, kemudian mengubah ke *option Enable* pada *Pi*lihan *Camera*. Kemudian *Reboot Raspberry Pi*.
5. Membuka file manager.
6. Memilih *directory* yang diinginkan untuk menyimpan file *Python*. Kemudian mengeklik kanan, memilih *create new*, memilih *new file*.
7. Memberikan nama sesuai keinginan dengan ekstensi *.py pada file tersebut.
8. Membuat program sederhana untuk merekam video. Contoh program sederhana dapat ditunjukan pada Gambar 4.5.

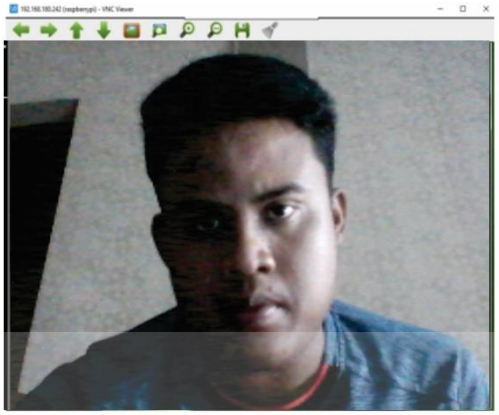


Gambar 4. 5 Contoh *Program* Sederhana Rekam Video Menggunakan *Pi Camera*

9. Kemudian *run program* tersebut.

4.2.4 Hasil Pengujian *Pi* Camera

Apabila *program* tidak ditemukan *error*, maka akan muncul *window* baru serta menam*Pi*lkan *program* yang diinginkan. Contoh hasil *running program* dapat ditunjukan pada Gambar 4.6.



Gambar 4. 6 Hasil *Running Program* Sederhana Rekam Video Menggunakan *Pi Camera*

4.3 Pengujian pendeteksian gerakan

4.3.1 Tujuan

Pengujian ini digunakan untuk mengetahui seberapa besar kamera dapat mendeteksi gerakan dengan di beberapa percobaan objek yang bergerak atau tidak dan bisa membedakan objek manusia atau bukan.

4.3.2 Alat yang Digunakan

Peralatan yang dibutuhkan untuk pengujuan ini adalah sebagai berikut :

1. Raspbery *Pi* 3
2. *Raspberry Pi* Kamera *module support nightvision*
3. Dual inframerah
4. Laptop atau Komputer

5. Kabel VGA 5 meter
6. *Mouse* dan *keyboard wireless*
7. Program *Background Subtraction,HOG,Database Cloud Firebase*
8. *Power supply* 2.5 A - 5V

4.3.2 Prosedur pengujian

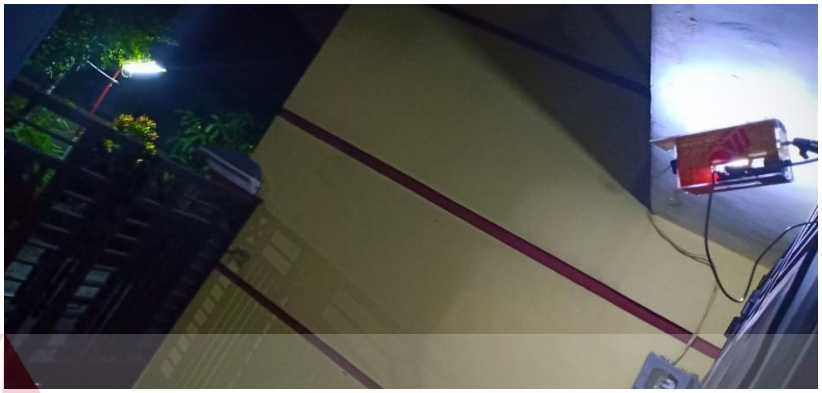
Langkah – langkah yang dilakukan untuk melakukan pengujian sistem adalah seperti berikut :

1. Merancang casing agar *Raspberry Pi* beserta camera dapat mudah diletakkan dimanapun

Gambar 4. 7 *Casing* alat (1)

Gambar 4. 8 *Casing* alat (2)

2. Mengatur lokasi yang akan di awasi beserta pemasangan kabel VGA beserta *power supply* dan *wireless mouse beserta keyboard.* Dipengambilan data ini yang diawasi adalah pagar masuk rumah.



Gambar 4. 9 Foto letak alat dari samping

Gambar 4. 10 Foto letak dari depan

3. Menghidupkan *Raspberry Pi* .
4. Pastikan adanya koneksi internet
5. Update waktu *Raspberry Pi* agar data yang dikirimkan ke database sesuai dengan waktu sekarang.
   a. Buka cmd
   b. Ketik “sudo ras*Pi*-config”

c. *Pi*lih *internationalization options*

d. *Pi*lih *Change Timezone*

e. *Pi*lih *geographic area Asia*

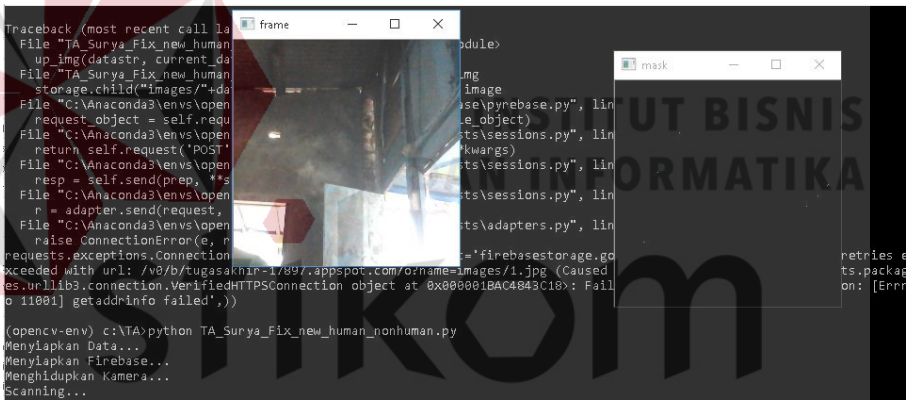
f. *Pi*lih *Timezone* Jakarta

g. Setelah itu jam di *Raspberry Pi* terupdate ketepatan sesuai waktu sebenarnya.

6. Buka cmd dan jalankan program yang sudah dibuat

   Python3 TA_Surya_Fix_new_human_nonhuman.py –*Pi*camera 1

7. Program berjalan



Gambar 4. 11 Gambar program berjalan

8. Mengawasi hasil program sampai 10 data sampel

4.3.3 Hasil pengujian

Dari prosedur pengujian diatas, apabila nilai hasil nilai *background subtraction* pada pada metode HOG melebihi nilai batas dan dipatkan terdeteksi manusia maka dinyatakan sebagai gerakan manusia, teta*Pi* jika nilai berada di atas batas ta*Pi* tidak diteksi manusia dinyatakan bukan manusia. Urutan data akan diurutkan berdasarkan urutan data di *database firebase*.

Hasil dari pengujian dapat dilihat pada Tabel 4.1 human dan Tabel 4.2 *Nonhuman*.

Tabel 4. 1 Data uji deteksi manusia

Data Ke -	*Human*		*Detect*		
	Berhasil	Gagal	Jam	Menit	Detik
23	v		09	29	18
24	v		09	29	35
25	v		09	29	40
26	v		09	29	43
27	v		09	29	49
28	v		00	3	53
29	v		00	3	57
30	v		00	4	0
31	v		00	4	6
32	v		00	4	9
Total	10	0			

Dari hasil diagram diatas, dapat dijelaskan bahwa dari 10 data *sample* uji coba terhadap siang dan malam hari dapat mendeteksi gerak dengan baik dan akurat. Sehingga persentase data dapat dihitung sebagai berikut:

$$\text{Persentase data} = \frac{(banyak\ pengujian\ x\ jumlah\ koresponden)}{jumlah\ seluruh\ data}\ x\ 100\%$$

$$= \frac{(10x10)}{10}\ x\ 100\%$$

$= \frac{100}{100} \; x \; 100\%$

= 100%

Tabel 4. 2 Data uji deteksi bukan manusia

Data Ke -	*Non-Human*		*Detect*		
	Berhasil	**Gagal**	**Jam**	**Menit**	**Detik**
2	v		9	31	35
3	v		9	31	53
5	v		9	32	9
6	v		9	34	20
7	v		9	37	55
8	v		2	4	54
9	v		2	6	54
10		v	2	6	57
11	v		2	12	38
12	v		2	14	19
Total	9	1			

Dari hasil diagram diatas, dapat dijelaskan bahwa dari 10 data *sample* uji coba terhadap siang dan malam hari dan dapat mendeteksi gerak dengan baik dan akurat. Terdapat kegagalan 1 data ke-10 dengan gerakan orang berpayung sehingga sistem gagal medeteksi bahwa gambar yang didapat adalah termasuk ada gerakan manusia.

Sehingga persentase data dapat dihitung sebagai berikut:

$$\text{Persentase data} = \frac{(banyak\ pengujian\ x\ jumlah\ koresponden)}{jumlah\ seluruh\ data}\ x\ 100\%$$

$$= \frac{(10x9)}{10}\ x\ 100\%$$

$$= \frac{90}{100}\ x\ 100\%$$

$$= 90\%$$

Dari hasil dua tabel (Tabel 4.1.1 dan 4.1.2) diatas yaitu 10 data pergerakan manusia dan 10 data pergerakan benda. Didapatkan data human tingkat keberhasilan sebesar 100% dan non human tingkat keberhasilan 90%. Dapat disimpulkan bahwa sistem ini lebih akurat digunakan jika mendeteksi pergerakan manusia. Teta*Pi* hasil ini tetap bisa dikatakan berhasil karena di sistem keamanan justru yang lebih penting adalah mengawasi pergerakan manusia. Dan untuk gerakan bukan manusia hanya dibuat sebagai pembanding deteksi.

4.4 Pengujian aplikasi notifikasi *Android*

4.4.1 Tujuan

Pada pengujian ini untuk mengetahui dapatkah aplikasi berbasis *Android* dapat memberi notifikasi jika terjadi pergerakan dan mengirim data tangkapan kamera kemobile *Android* secara real time atau terdapat masalah ataupun terjadi delay.

4.4.2 Alat yang digunakan

Peralatan yang dibutuhkan untuk pengujuan ini adalah sebagai berikut :

1. Raspbery *Pi* 3
2. *Raspberry Pi* Kamera *module support nightvision*
3. Dual inframerah
4. Laptop atau Komputer
5. Smartphone *Android*
6. Kabel VGA 5 meter
7. *Mouse* dan *keyboard wireless*
8. Program *Background Subtraction,HOG,Database Cloud Firebase*
9. *Power supply* 2.5 A - 5V

4.4.3 Prosedur pengujian

Langkah – langkah yang dilakukan untuk melakukan pengujian sistem adalah seperti berikut :

1. Mengatur lokasi yang akan di awasi kamera beserta pemasangan kabel VGA beserta *power supply* dan *wireless mouse beserta keyboard.* Dipengambilan data ini yang diawasi adalah pagar masuk rumah.
2. Menghidupkan *Raspberry Pi*
3. Pastikan adanya koneksi internet
4. Update waktu *Raspberry Pi* agar data yang dikirimkan ke database sesuai dengan waktu sekarang.
    a. Buka cmd
    b. Ketik “sudo ras*Pi*-config”
    c. *Pi*lih *internationalization options*
    d. *Pi*lih *Change Timezone*

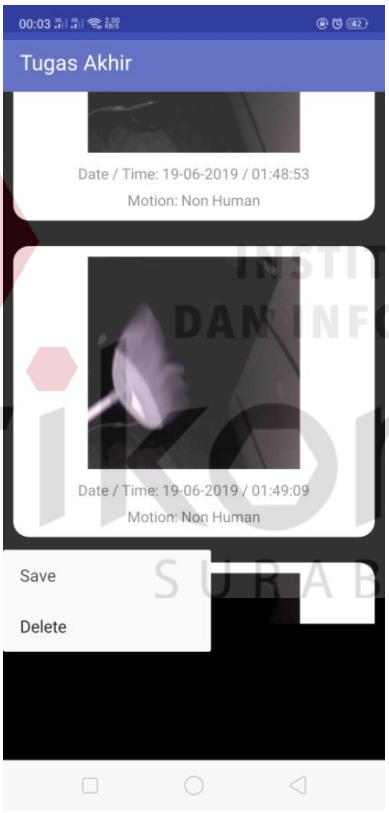
e. *Pi*lih *geographic area Asia*

f. *Pi*lih *Timezone* Jakarta

g. Setelah itu jam di *Raspberry Pi* terupdate ketepatan sesuai waktu sebenarnya.

5. Patikan di smartphone *Android* sudah terinstall aplikasi program tugas akhir

6. Tampilan aplikasi *Android* tugas akhir



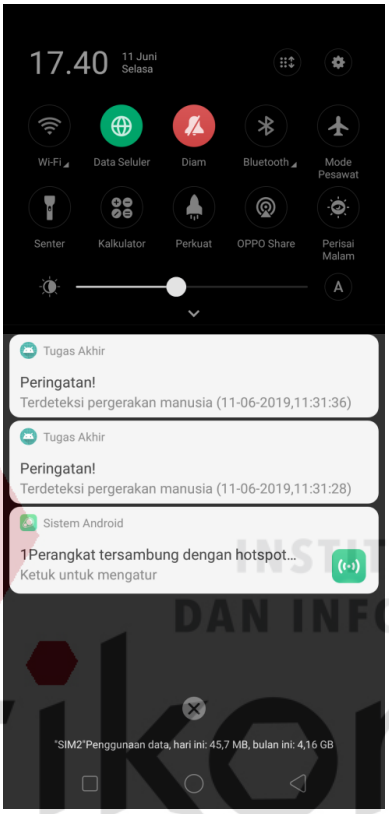
Gambar 4. 12 Tampilan apk

7. Setelah masuk apk keluar/minimize, jangan di tutup agar notifikasi bisa masuk

8. Buka cmd *Raspberry Pi* dan jalankan program yang sudah dibuat

Python3 TA_Surya_Fix_new_human_nonhuman.py –*Pi*camera 1

9. Awasi jendela notifikasi *Android* maka akan muncul seperti gambar dibawah



Gambar 4. 13 Jendela notifikasi *Android*

10. Mengawasi 20 data sample

4.4.4 Hasil pengujian

Dari pengujian diatas diambil 20 data *sample*, akan didapatkan data *realtime* waktu data masuk. Data akan dibandingkan antar data dan diambil rata-rata jarak waktu notifikasi masuk. Data dibuat 2 tabel.yaitu *human* dan *nonhuman*. Di bedakannya data karena dari settingan program ada perbedaan

juga delay notif human dengan *non human*. Jarak waktu notifikasi *nonhuman* lebih lama daripada *human*

Tabel 4. 3 Pengujian notifikasi deteksi manusia

Data Ke -	*Human*		*Detect*			*Report*			*Delay (detik)*
	Berhasil	**Gagal**	**Jam**	**Menit**	**Detik**	**Jam**	**Menit**	**Detik**	
23	v		oo	3	27	oo	3	27	0
24	v		oo	3	34	oo	3	34	7
25	v		oo	3	42	oo	3	42	8
26	v		oo	3	46	oo	3	46	4
27	v		oo	3	49	oo	3	49	3
28	v		oo	3	53	oo	3	53	4
29	v		oo	3	57	oo	3	57	4
30	v		oo	4	0	oo	3	0	3
31	v		oo	4	6	oo	3	6	6
32	v		oo	4	9	oo	3	9	3
Total	10	0							
Rata-rata									4.2

Tabel 4. 4 Pengujian notifikasi deteksi bukan manusia

Data Ke -	*Non-Human*		*Detect*			*Report*			*Delay (detik)*
	Berhasil	Gagal	Jam	Menit	Detik	Jam	Menit	Detik	
2	v		1	48	53	1	48	53	0
3	v		1	49	9	1	49	9	9
5	v		1	52	9	1	52	9	180
6	v		1	53	11	1	53	11	62
7	v		1	55	56	1	55	56	165
8	v		2	4	54	2	4	54	534
9	v		2	6	54	2	6	54	118
10		v	2	6	57	2	6	57	3
11	v		2	12	38	2	12	38	398
12	v		2	14	19	2	14	19	101
Total	9	1							
Rata-rata									157

Dari dua tabel diatas didapatkan bahwa rata-rata jarak notifikasi *human* sebesar 4,2 detik dan *non human* 157 detik atau 2 menit 37 detik. Hasil ini sudah sesuai sistem yang lebih mengutamakan notifikasi *human* daripada *nonhuman*. Dikarenakan didalam sistem keamanan akan lebih baik jika sistem tersebut mengawasi hanya pergerakan manusia. Tetapi, gerakan *non human* hanya sebagai perbandingan. Di

sistem juga memang diatur bahwa jeda delay perbandingan *human* delay sebesar 60 *frame* dan *non human* 300 *frame*.

4.5 Pengujian pendeteksian aplikasi database *firebase*

4.5.1 Tujuan

Pada pengujian ini digunakan untuk mengatahui dapatkah aplikasi database cloud firebase dapat memabantu pengguna sistem keamanan untuk menyimpan data gambar dengan baik dan dapat dengan mudah digunakan oleh pengguna ketika ingin melihat hasil gambar.

4.5.2 Alat yang digunakan

Peralatan yang dibutuhkan untuk pengujuan ini adalah sebagai berikut :

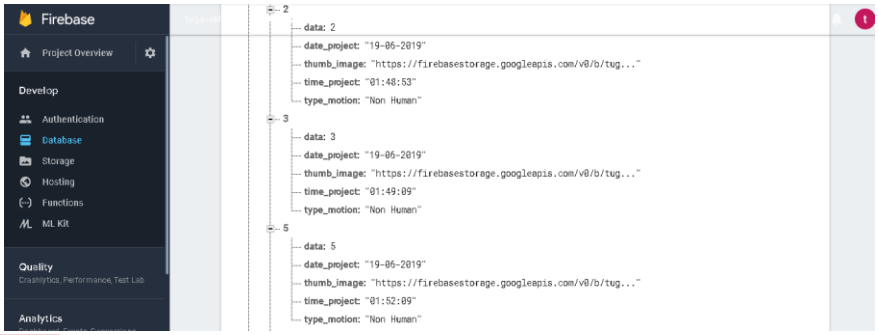
1. *Laptop/computer*
2. *Smartphone Android*
3. Koneksi internet

4.5.3 Prosedur pengujian

Langkah – langkah yang dilakukan untuk melakukan pengujian sistem adalah seperti berikut :

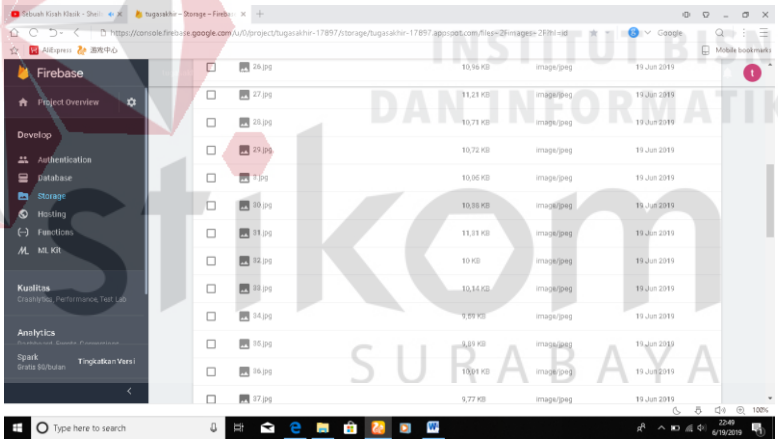
1. Hidupkan *laptop/computer*
2. Pastikan terkoneksi internet
3. Buka *browser*
4. Ketik *url firebase.com*

5. Login
6. *Pi*lih file project tugas akhir
7. Jika ingin cek data *realtime* berupa data *string P*ilih folder *database*



Gambar 4. 14 Tampilan *database* data *string*

8. Jika ingin cek data gambar maka masuk folder *storage*



Gambar 4. 15 Tampilan *database* data gambar

9. Tetapi disini akan diuji data *string*. karena data *string* lah yang *realtime*, karena jika data string berhasil masuk, pasti akan diikuti data gambar tergantung kecepatan data internet.
10. nanti akan dicoba penghapusan dari *Android* mapupun langsung *database firebase*

4.5.4 Hasil pengujian

Dari hasil pengambilan 10 *sample* data pergerakan manusia dan 10 data bukan pergerakan manusia dan didapatkan tabel 4.2.1.

Tabel 4. 5 Data uji database *human*

Data Ke -	*Human*		*Detect*			*database*	
	Berhasil	Gagal	Jam	Menit	Detik	berhasil	gagal
23	v		oo	3	27	v	
24	v		oo	3	34	v	
25	v		oo	3	42	v	
26	v		oo	3	46	v	
27	v		oo	3	49	v	
28	v		oo	3	53	v	
29	v		oo	3	57	v	
30	v		oo	4	0	v	
31	v		oo	4	6	v	
32	v		oo	4	9	v	
Total	10					10	

Dari hasil diagram diatas, dapat dijelaskan bahwa dari 10 data *sample* uji coba terhadap 10 Jenis data *human* dapat mengirim ke *database firebase* dengan berjalan dengan baik.

Persentase data = $\frac{(banyak\ pengujian\ yang\ berhasil\ )}{jumlah\ seluruh\ data} x\ 100\%$

$= \frac{(10)}{10} x\ 100\%$

$= \frac{100}{100} x\ 100\%$

= 100%

Tabel 4. 6 Data uji database *non human*

Data Ke -	*Non-Human*		*Detect*			*Database*	
	Berhasil	**Gagal**	**Jam**	**Menit**	**Detik**	**berhasil**	**gagal**
2	v		1	48	53	v	
3	v		1	49	9	v	
5	v		1	52	9	v	
6	v		1	53	11	v	
7	v		1	55	56	v	
8	v		2	4	54	v	
9	v		2	6	54	v	
10		v	2	6	57	v	
11	v		2	12	38	v	
12	v		2	14	19	v	
Total	9	1				10	0

Dari hasil diagram diatas, dapat dijelaskan bahwa dari 10 data *sample* uji coba terhadap 10 Jenis data *nonhuman* dapat mengirim ke *database firebase* dengan berjalan dengan baik.adapun hasil bukti dari tabel diatas berada di Lampiran.

Persentase data = $\frac{(banyak\ pengujian\ yang\ berhasil\ )}{jumlah\ seluruh\ data} x\ 100\%$

= $\frac{(10)}{10} x\ 100\%$

= $\frac{100}{100}\ x\ 100\%$

= 100%

Dan tabel 4.2.3 merupakan data hasil uji coba penghapusan data diambil 10 *sample* data. Dimana data ke 2-7 diambil dari data pendeteksian *human* dan data ke 23-27 data dari pendeteksian *nonhuman.*

Tabel 4. 7 Data uji *delete* data *firebase* dan *Android*

Delete Data Ke -	Firebase		*Android*	
	Berhasil	Gagal	Berhasil	Gagal
2	v		v	
3	v		v	
5	v		v	
6	v		v	
7	v		v	
23	v		v	
24	v		v	
25	v		v	
26	v		v	
27	v		v	
Total	10	0	10	0

Dari hasil diagram diatas, dapat dijelaskan bahwa dari 10 data *sample* uji coba terhadap 10 Jenis data untuk dihapus dari *Android* maupun langsung *database firebase* dengan berjalan dengan baik dan akurat 100%.

$$\text{Persentase data} = \frac{(banyak\ pengujian\ yang\ berhasil\ )}{jumlah\ seluruh\ data}\ x\ 100\%$$

$$= \frac{(10)}{10}\ x\ 100\%$$

$$= \frac{100}{100}\ x\ 100\%$$

$$= 100\%$$

Dari pengamatan tabel 4.2.1,4.2.2,4.2.3 didapatkan bahwa program *database* dapat berjalan dengan akurat dari pengunggahan data sampai penghapusan data berjalan sangat baik. Sehingga dapat disimpulkan bahwa *database cloud firebase* terhadap program sistem keamanan dapat sangat membantu.

BAB V

KESIMPULAN

5.1 Kesimpulan

Dari perancangan *program* hingga pengujian yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Kamera yang digunakan menggunakan infrared sehingga keadaan lingkungan terutama intensitas pencahayaan tidak mempengaruhi hasil deteksi secara signifikan. Deteksi gerakan manusia (*human*) bekerja dengan akurat dengan akurasi 100% sedangkan deteksi benda (*non-human*) dengan akurasi 90%.
2. *Storage database firebase* hanya berkisar 1 GB tidak terkendala penuhnya penyimpanan data dari *Raspberry Pi*. Dikarenakan data yang dikirimkan hanya data *string* dan gambar. Dan di pengujian menghapus data dari *mobile Android* maupun di *database firebase* tidak didapatkan masalah.
3. Aplikasi berbasis *Android* dapat memberi notifikasi jika terjadi pergerakan dan mengirim data tangkapan kamera kemobile *Android* secara real time dan sesuai *database.* Notifikasi terkirim secara *realtime* ke *Android* dan dapat selalu *update* data dengan menghidupkan aplikasi *Android* terlebih dahulu

5.2 Saran

Dalam pengembangan selanjutnya dapat dilakukan dengan pengiriman data tidak hanya berupa data image tetapi juga data video *streaming* secara *realtime* dengan meningkatkan kapasitas perangkat pengirim. Hal tersebut dikarenakan

kapasitas operating *Raspberry Pi* terbatas dalam pengolahan transfer data *file* yang besar ke *firebase*.

DAFTAR PUSTAKA

Ikhsan Samir,Zuraiyah A. (2016). “Penerapan Algoritma Background Substraction Untuk Tracking dan Klasifikasi Kendaraan”. Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Pakuan

Bradski, G., & Kaehler, A. (2008). “Computer Vision With The OpenCV Library”. Gravenstein Highway North, Sebastopol, CA 95472: O’Reilly Media, Inc.

Pi,R.(2015).Retrievedfrom *Raspberry Pi*: www.rs-components.com/*RaspberryPi*

Abdul Kadir Ir.(2005), Dasar Pemrograman Python, Andi Offset, Yogyakarta.

Juansyah Andi.(2015).Pembangunan Aplikasi Child Tracker Berbasis Assisted.Bandung.

Oriza Ahmad,(2014).Firebase Membantu Kita Membuat Aplikasi Realtime.Retrieved from https://www.codepolitan.com.

Widipratama, M. Z. (2017). *Sistem Pemantau Keamanan Menggunakan Kamera dengan Metode Background Subtraction.* Surabaya: Tugas Akhir Institut Bisnis dan Informatika Stikom Surabaya.

Kusno Suryadi, Supriyanto Sikumbang.(2015). Human Detection Menggunakan Metode Histogram Of Oriented Gradients.Malang: Universitas Gajayana Malang.

Android Developer; "*Android* Studio," *Android* Developer .Realtime.Retrieved from http://developer.*Android*.com/sdk/.

Kompas (2013),” Setiap 91 Detik, Terjadi Satu Kejahatan di Indonesia”.Realtime Retrieved from https://nasional.kompas.com/read/2012/12/26/15260465/Setiap.91.Detik.Terjadi.Satu.Kejahatan.di.Indonesia.