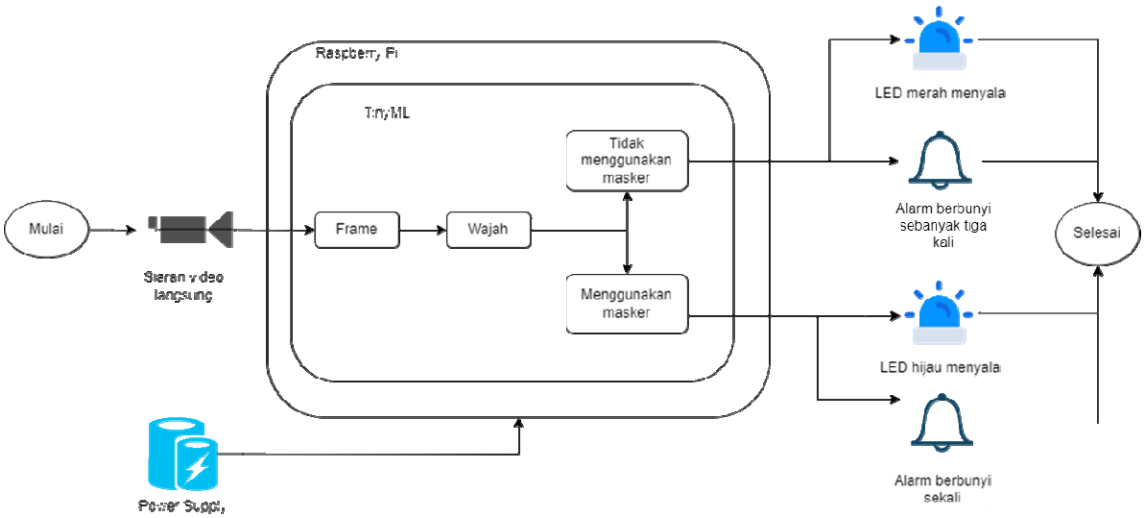
BAB 3 PERANCANGAN DAN IMPLEMENTASI

3.1 Arsitektur Sistem

Alur kerja sistem dapat dilihat pada gambar 3.1, pengunjung yang akan memasuki kawasan wajib masker berdiri menghadap ke kamera Raspberry Pi. Frame dari siaran langsung kamera akan diteruskan ke Raspberry Pi untuk di proses oleh model *Machine Learning* yang ada. Frame akan dianalisa oleh detektor wajah *Deep Neural Network* (DNN) yang berupa bawaan dari versi OpenCV 3.3. Detektor wajah ini berupa model *Caffe* yang sudah dilatih (*pre-trained*) yang menggunakan *Single Shot-Multibox Detector* (SSD) dan ResNet-10. Tensorflow Lite akan melakukan prediksi berdasarkan frame yang didapat untuk menentukan apakah pengunjung menggunakan masker atau tidak. Hasil prediksi akan ditampilkan ke monitor dan pengunjung dapat melihat hasil prediksi. Bila pengunjung tidak menggunakan masker, maka lampu LED berwarna merah akan menyala dan buzzer akan berbunyi sebanyak tiga kali. Bila pengunjung menggunakan masker, maka lampu LED berwarna hijau akan menyala dan buzzer akan berbunyi satu kali selama 2 detik lalu sistem akan menunggu sebesar 1 detik.



Gambar 3.1 Diagram Blok Arsitektur Sistem

3.2 Cara Kerja Sistem

Dalam menjalankan sebuah sistem, tentunya terdapat input, proses, dan output dari sistem tersebut. Berikut ini adalah input, proses, dan output dari sistem yang telah dibuat:

3.2.1 Input Sistem

Kamera Raspberry Pi berperan penting dalam akan menangkap frame dari siaran video secara langsung (*real-time*). Sistem akan membaca frame per frame dari siaran kamera Raspberry. Setelah itu, frame yang dibaca akan diubah (*resize*) lalu diteruskan ke proses sistem

3.2.2 Proses Sistem

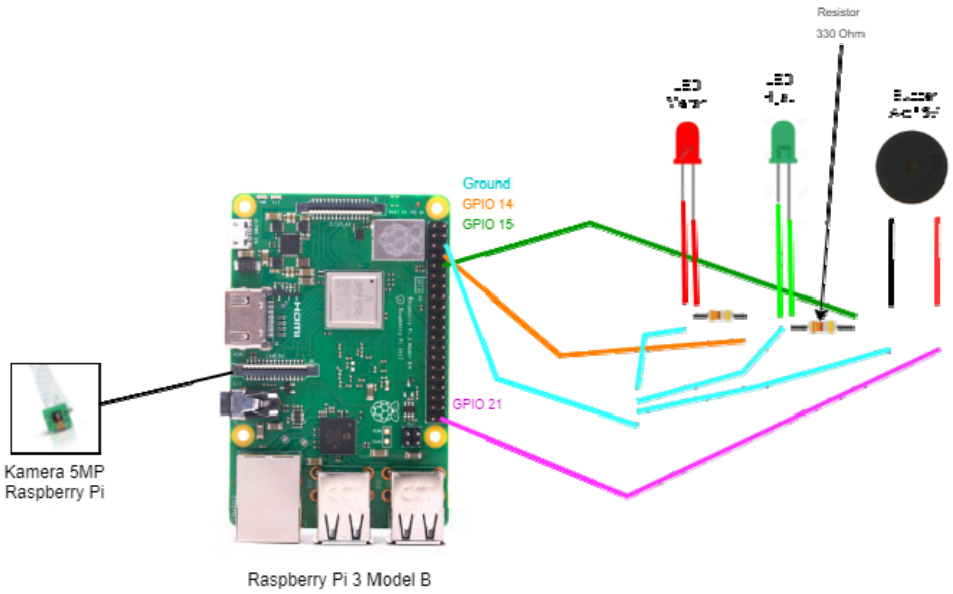
Frame dari input akan diambil dimensinya lalu sebuah *blob* akan dikonstruksi dari frame yang didapat. Blob (*Binary large object*) adalah sebuah group dari pixel (dalam sebuah image) yang saling terhubung dengan properti yang sama. Blob akan diteruskan ke dalam jaringan dari model wajah untuk mendapatkan deteksi wajah. Menyaring deteksi yang lemah dengan memastikan *confidence* yang didapat lebih besar dari *confidence* minimum (0.5). Menghitung koordinat (x,y) untuk menaruh kotak di sekitar wajah sebagai penanda. Melakukan ekstraksi ROI (*region(s) of interest*) dari wajah seperti mata, mulut, dan hidung. ROI wajah diubah dari BGR menjadi RGB, lalu ROI wajah dilakukan pengubahan ukuran (resize), dan ROI wajah kemudian di praproses (*preprocess*). Bila ada wajah yang terdeteksi, maka hasil ROI wajah akan diteruskan ke dalam Tensorflow Lite (TFLite). TFLite akan mendeteksi apakah wajah menggunakan masker atau tidak, hasil prediksi akan diteruskan ke pada output

3.2.3 Output Sistem

Hasil prediksi dari proses akan diambil dan dilakukan penentuan apakah wajah menggunakan masker atau tidak Bila masker terdeteksi, maka sistem akan melakukan print “No Mask Detected!”, print hasil prediksi yang didapat, menyalakan LED hijau, membunyikan buzzer selama 2 detik, sistem menunggu 1 detik, lalu mematikan semua LED dan buzzer. Bila masker tidak terdeteksi, maka sistem akan melakukan print “Mask Detected!”, print hasil prediksi yang didapat, menyalakan LED merah, membunyikan buzzer sebanyak tiga kali, lalu mematikan semua LED dan buzzer. Bila tidak ada wajah yang terdeteksi, maka sistem akan melakukan print print “No Face Detected!”

3.3 Perancangan Perangkat Keras

Dalam penelitian ini, sistem menggunakan perangkat keras sebagai input dan output dari sistem dapat dilihat oleh pengunjung. Skematik dari keseluruhan perancangan perangkat keras dapat dilihat pada Gambar 3.2, kamera 5MP Raspberry Pi terhubung dengan port kamera yang terdapat pada Raspberry Pi 3B. GPIO 14 (Tx) dihubungkan dengan pin positif LED merah dan diberi resistor sebesar 330 Ohm untuk menyalakan LED merah dan GPIO 15 (Rx) dihubungkan dengan pin positif LED hijau dan diberi resistor sebesar 330 Ohm untuk menyalakan LED hijau. Sedangkan, GPIO 21 (SCLK) dihubungkan dengan pin positif buzzer. Terakhir, seluruh sirkuit dihubungkan dengan ground Raspberry Pi.



Gambar 3.2 Skematik Keseluruhan Perangkat

3.3.1 Raspberry Pi 3B

Raspberry Pi merupakan sebuah komputer berukuran kecil yang terpasang pada papan sirkuit tunggal. Raspberry Pi dapat melakukan banyak hal layaknya sebuah komputer biasa seperti menulis dokumen, memainkan permainan, dan memutar video. Selain itu, Raspberry Pi memiliki pin-pin GPIO (general purpose input/output) untuk menjalankan komponen elektrik, hal ini memungkinkan pengguna untuk mengontrol komponen elektrik seperti LED dan menjelajahi Internet of Things (IoT). Pada sistem ini Raspberry Pi 3 model B berfungsi sebagai otak dalam menerima masukan gambar dari kamera, menjalankan aplikasi detektor masker dan mengeluarkan keluaran dalam bentuk LED dan buzzer. Gambar 3.3 menampilkan

bentuk fisik dari Raspberry Pi 3 model B yang digunakan dalam sistem ini. Spesifikasi Raspberry Pi 3 model B dapat dilihat lengkap pada Lampiran A.1

Gambar 3.3 Bentuk Fisik dari Raspberry Pi 3 Model B [21]

3.3.2 *Piezoelectric Buzzer*

Piezoelectric buzzer merupakan sebuah komponen yang menghasilkan getaran suara dari sinyal listrik yang diberikan. Piezoelectric buzzer adalah salah satu jenis *buzzer* yang menghasilkan suara dengan menggunakan efek *piezoelectric*. Hal ini dapat dilakukan dengan memberikan tegangan listrik ke bahan *piezoelectric* sehingga terjadi gerakan mekanis, gerakan tersebut dapat didengar oleh manusia dengan menggunakan diafragma. Gambar 3.4 menampilkan bentuk fisik *piezoelectric buzzer*. Spesifikasi *piezoelectric buzzer* dapat dilihat lengkap pada Lampiran A.2

Gambar 3.4 Bentuk Fisik *Piezoelectric Buzzer* [22]

3.3.3 *Dual In-Line Package* (DIP) LED

DIP (Dual In-Line Package) LED adalah sebuah lampu LED tradisional. DIP LED terlihat paling seperti lampu yang umum digunakan dengan chip terbungkus plastik keras dengan 2 pin penghubung paralel lurus. Ketika chip LED terhubung ke sumber listrik, arus listrik akan mengalir melalui chip. Ketika hal ini terjadi, cahaya (dengan warna tertentu) dipancarkan. Gambar 3.5 merupakan bentuk fisik dari DIP LED.

Gambar 3.5 Bentuk Fisik DIP LED [23]

3.4 Perancangan Perangkat Lunak

Perangkat lunak digunakan oleh sistem untuk memproses input sistem yang didapatkan dari kamera dan mengeluarkan output sistem berupa LED dan *buzzer*. Dalam penelitian ini, perangkat lunak akan dijalankan di dalam Raspberry Pi 3B. Adapun tahap-tahap dalam merancang perangkat lunak adalah sebagai berikut:

3.4.1 Pelatihan Model Tensorflow

Dalam melatih model Tensorflor, dibutuhkan dataset sebagai acuan bagi model dalam membedakan pengunjung yang menggunakan masker atau tidak. Dataset terdiri dari dua kategori yaitu bermasker dan tidak bermasker. Jumlah dataset total adalah 3.911 files (155 MB) dengan jumlah dataset bermasker adalah 1.968 files (114 MB) dan jumlah dataset yang tidak bermasker adalah 1.943 (41 MB) [24][25][26]. Contoh gambar dataset bermasker dan tidak bermasker dapat dilihat pada Gambar 3.6. Proses pelatihan model Tensorflow dapat dlihat secara lengkap dalam Gambar 3.8. Penjelasan tahap-tahap algoritma dalam melatih model Tensorflow dapat dilihat secara lengkap pada Algoritma 3.1.

Gambar 3.6 Contoh Gambar dalam Dataset yang Digabung [24][25][26]

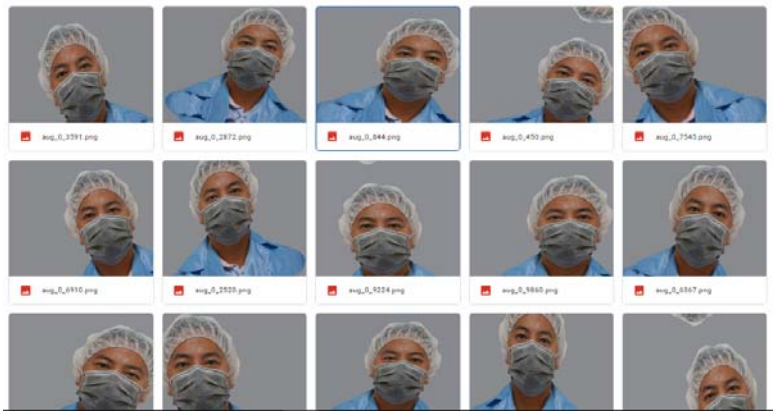
Algoritme 3.1: Algoritma untuk melatih model Tensorflow

Masukan: Dataset bermasker dan tidak bermasker

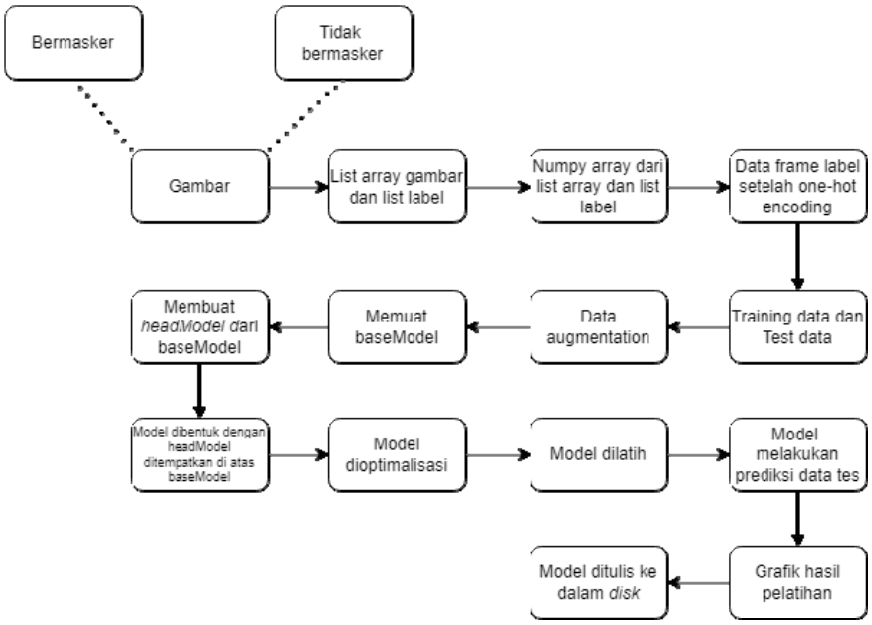
Keluaran: Hasil prediksi data tes, model Tensorflow, dan grafik dari hasil pelatihan model

Proses:

1: Memuat *dataset* yang terdiri atas bermasker dan tidak bermasker
2: Melakukan *looping* dari setiap gambar untuk mengekstraksi label dari nama *file*, mengubah gambar menjadi *array*, melakukan praproses gambar, dan hasil dimasukan ke dalam *list array* gambar dan *list* label
3: *List array* gambar dan *list* label diubah menjadi *Numpy array*
4: Melakukan *one-hot encoding* pada list label
5: Data dipisah menjadi data tes dan data pelatihan dengan rasio masing-masing sebesar 25% dan 75%
6: Melakukan augmentasi data dengan parameter sebagai berikut: rotasi sebesar 20 derajat, zoom sebesar 15%, pemindahan ke atas dan samping sebesar 20%, pengubahan sudut gambar sebesar 15%, gambar dibalik secara horizontal, dan fill mode adalah nearest
7: Memuat *baseModel* dari MobileNetV2
8: Membangun *headModel* dari baseModel
9: Menempatkan *headModel* di atas *baseModel*
10: Membekukan seluruh layer di dalam baseModel
11: Model disusun dan dioptimalkan dengan *Adam* dengan *learning rate* sebesar 0.0001
12: Model dilatih dengan *batch size* sebesar 32
13: Model melakukan hasil prediksi berdasarkan data tes
14: Membuat grafik berdasarkan hasil pelatihan model
15: Menuliskan model ke dalam *disk*



Gambar 3.7 Contoh Gambar Hasil Augmentasi Data



Gambar 3.8 Diagram Blok Pelatihan Model *Tensorflow*

3.4.2 Pengubahan Model dari *Tensorflow* menjadi *Tensorflow Lite*

Setelah model *Tensorflow* dilatih dan disimpan, maka tahap selanjutnya adalah mengubah model Tensorflow tersebut menjadi model Tensorflow Lite. Proses pengubahan tersebut dapat dilihat secara keseluruhan pada Gambar 3.8, adapun tahap-tahap dalam proses pengubahan dapat dilihat dalam Algoritma 3.2. Dalam Algoritma 3.2, Model Tensorflow diubah menjadi model Tensorflow Lite tanpa melakukan kuantisasi. Pengubahan tanpa kuantisasi ini dilakukan untuk menjaga keakuratan dari model Tensorflow yang telah dilatih. Setelah model Tensorflow diubah, model Tensorflow Lite disimpan ke dalam disk

Algoritme 3.2: Algoritma untuk mengubah model Tensorflow menjadi Tensorflow Lite

Masukan: Model Tensorflow

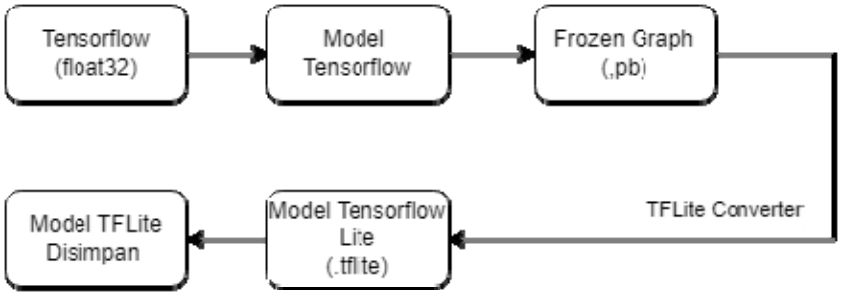
Keluaran: Model Tensorflow Lite

Proses:

1: Memuat model Tensorflow dari *disk* ke dalam TFLiteConverter

2: Mengubah model Tensorflow menjadi model Tensorflow Lite

3: Menyimpan model Tensorflow Lite ke dalam disk



Gambar 3.8 Diagram Blok Konversi Mode Tensorflow

3.5 Integrasi Sistem

Setelah model dilatih dan diubah menjadi Tensorflow Lite, model akan diintegrasikan ke dalam sistem agar tujuan penelitian tercapai. Dalam Gambar 3.9, proses intergrasi sistem dapat dilihat secara keseluhan. Adapun penjelasan tahap-tahap integrasi sistem dapat dilihat pada Algoritma 3.3.

Algoritme 3.3: Algoritma untuk mendeteksi penggunaan masker

Masukan: Frame dari siaran langsung kamera Raspberry Pi

Keluaran: Hasil prediksi penggunaan masker wajah, warna lampu LED, dan buzzer

Proses:

1: Memuat model *Caffe* untuk deteksi wajah
2: Memuat model *Tensorflow Lite* untuk deteksi masker wajah
3: Menjalankan siaran langsung kamera Raspberry Pi
4: Memulai *looping* untuk deteksi wajah
5: Membaca *frame* dari siaran langsung
6: Melakukan pengubahan ukuran *frame* tersebut
7: Memasukan *frame,* model Caffe, dan model Tensorflow Lite yang telah diubah ke dalam fungsi
8: Mengambil dimensi dari *frame*
9: Menyusun blob dari dimensi *frame*
10: Blob dioperkan ke detektor wajah untuk mendapatkan deteksi wajah
11: Melakukan filter untuk menyaring deteksi wajah yang lemah
12: Menempatkan kotak di sekitar wajah yang terdeteksi
13: Melakukan ekstraksi *Region of Interest* (ROI) dari wajah
14: Mengubah channel BGR menjadi channel RGB dari hasil ekstraksi wajah
15: Mengubah ukuran hasil ekstraksi wajah
16: Melakukan pra proses hasil ekstraksi wajah
17: Tensorflow Lite melakukan prediksi berdasarkan hasil ekstraksi wajah
18: Hasil prediksi dari Tensorflow Lite dimasukan ke dalam array dan dikeluarkan dari fungsi

Algoritme 3.3: Algoritma untuk mendeteksi penggunaan masker

19: Membandingkan hasil prediksi yang didapat

20: Bila hasil prediksi menyatakan bahwa wajah mengenakan masker

21: Lampu LED berwarna hijau menyala

22: Buzzer mengeluarkan bunyi

23: Sistem menunggu selama 2 detik

24: Buzzer dimatikan

25: Lampu LED berwarna hijau dimatikan

26: Hasil prediksi ditampilkan di layar

27: Bila hasil prediksi menyatakan bahwa wajah tidak mengenakan masker

28: Lampu LED berwarna merah menyala

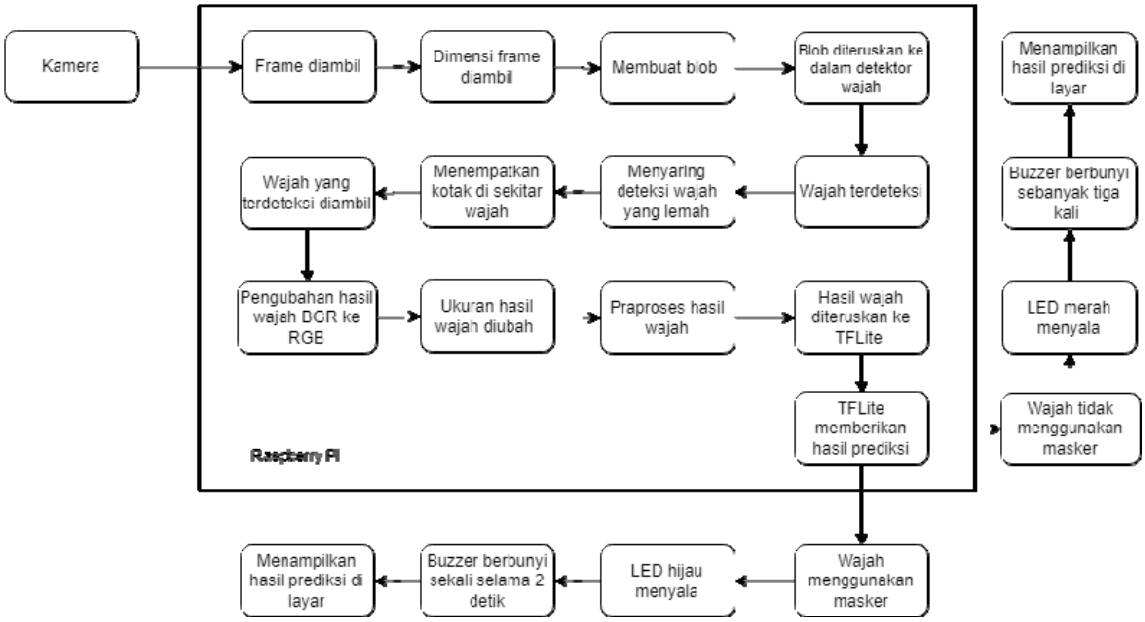
29: Melakukan looping sebanyak tiga kali, buzzer mengeluarkan bunyi

30: Sistem menunggu selama 1 detik

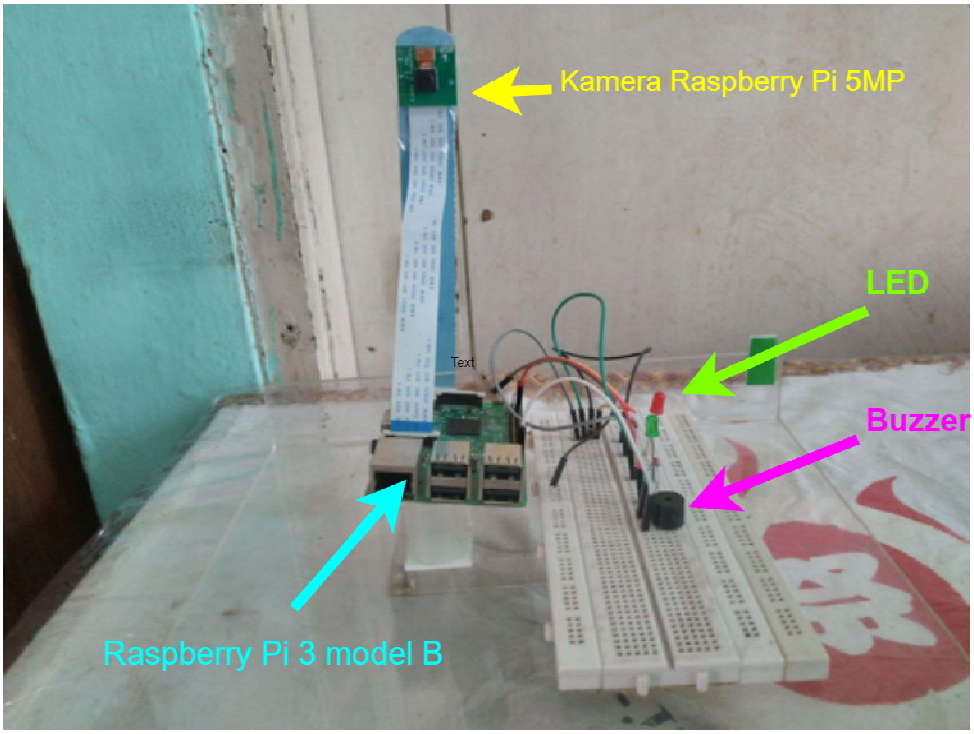
31: Buzzer dimatikan

32: Bila looping telah selesai, lampu LED berwarna merah dimatikan

33: Hasil prediksi ditampilkan di layar



Gambar 3.9 Diagram Blok Implementasi Model TFLite



Gambar 3.6 Prototipe perangkat yang telah digabung