TUGAS AKHIR – TE 141599

MONITORING *ONLINE* KETERSEDIAAN *SLOT* PARKIR BERBASIS KAMERA VIA *WEBSITE* MENGGUNAKAN RASPBERRY PI 3

Della Sabila Azkarika
NRP 2213100178

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – **TE 141599**

ONLINE MONITORING OF AVAILABLE PARKING SLOT BASED ON CAMERA VIA WEBSITE USING RASPBERRY PI 3

Della Sabila Azkarika
NRP 2213100178

Advisor
Dr. Muhammad Rivai, ST., MT.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "MONITORING ONLINE KETERSEDIAAN SLOT PARKIR BERBASIS KAMERA VIA WEBSITE MENGGUNAKAN RASPBERRY PI 3" adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-ba han yang tidak diijinkan dan bukan merupakan karya orang lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun durujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 5 Juni 2017

Della Sabila Azkarika
NRP.2213100178

MONITORING *ONLINE* KETERSEDIAAN *SLOT* PARKIR BERBASIS KAMERA VIA *WEBSITE* MENGGUNAKAN RASPBERRY PI 3

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui,

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Muhammad Rivai, ST., MT.	Dr. Ir. Hendra Kusuma, M.Eng.Sc.
NIP. 196904261994031003	NIP. 196409021989031003

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO

SURABAYA
JULI, 2017

MONITORING *ONLINE* KETERSEDIAAN SLOT PARKIR BERBASIS KAMERA VIA *WEBSITE* MENGGUNAKAN RASPBERRY PI 3

Nama : Della Sabila Azkarika
Pembimbing 1 : Dr. Muhammad Rivai, ST., MT.
Pembimbing 2 : Dr. Ir. Hendra Kusuma, M.Eng.Sc.

ABSTRAK

Salah satu hal yang penting dalam melakukan monitoring pada area parkir yaitu, memberikan informasi kepada pengunjung mengenai jumlah kendaraan pada area parkir. Terdapat beberapa sistem yang telah digunakan, salah satunya menggunakan sensor ultrasonik. Tetapi seringkali pengunjung menghabiskan waktu untuk mencari parkir. Mereka tidak mendapatkan informasi mengenai slot parkir secara detil.

Pada Tugas Akhir ini, telah dibuat sistem yang dapat memantau slot parkir via *website*. Pendeteksian ini menggunakan kamera dengan metode *template matching*. Cara kerja dari metode ini yaitu dengan mendeteksi *pattern* yang berupa *template.* Apabila dari hasil perhitungan memperoleh nilai mendekati 1 maka letak *template* sesuai dengan *pattern*, dengan begitu akan terdeteksi tidak terdapat kendara, begitu pun sebaliknya. Sistem *monitoring* parkir ini menggunakan Raspberry Pi 3 untuk melakukan pengolahan citra . Data berupa informasi ketersediaan slot ini, kemudian dikirim dan disimpan di *database* pada MySQLITE. Informasi yang ditampilkan berupa warna yang akan menunjukkan kosong atau isi pada *website*. Apabila berwarna merah maka terdapat kendaraan, dan warna hijau tidak terdapat kendaraan.

Pengujian dengan metode *template matching* membutuhkan waktu 14-16 detik untuk setiap kali *looping*. Pendeteksian dilakukan dengan intensitas cahaya dari 7,7-500 lux. Sistem ini diharapkan dapat membantu pengendara mobil untuk lebih mudah mengetahui letak slot parkir yang kosong.

Kata kunci: raspberry pi 3, *slot* parkir, *template matching*

Halaman ini sengaja dikosongkan

ONLINE MONITORING OF AVAILABLE PARKING SLOT BASED ON CAMERA VIA WEBSITE USING RASPBERRY PI 3

Name	**: Della Sabila Azkarika**
Supervisor	**: Dr. Muhammad Rivai, ST., MT.**
Co-Supervisor	**: Dr. Ir. Hendra Kusuma, M.Eng.Sc.**

ABSTRACT

One of the most important aspect of monitoring a parking area is to collect information about the number of vehicles parked and available parking spots, thus providing a useful information for the visitors. There are few monitoring systems that have already implemented such as one using ultrasonic sensor. Oftentimes the visitors are forced to spend a considerable amount of time to search for an available parking spot because the lack of detailed information.

In this final assignment, a system is designed to monitor available parking spot and put the information through a website. A camera and template matching method are utilized in this system. The procedure of this method is to detect a pattern that converted into a template. If the result of calculation close to 1 then the template matches correctly with the pattern, thus the result will show that there is no vehicle on the spot, and vice versa. The system utilizes a Raspberry Pi 3 for image processing to monitor the parking area. The data collected are sent and stored in the database at MySQLITE. The information is displayed on the website as color coded blocks that shows whether it is available or not. If the color of the block is Red that means the parking spot is unavailable, and if it's shown in Green means that the spot on that location is available.

Experiments done using template matching method has shown that it requires 14-16 seconds for each loop. Detection is conducted in a situation with 7.4-500 lux of light intensity. The system is expected to help drivers to find the location of an available parking spot.

Keywords: rapberry pi 3, parking slot, template matching

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillaahi Robbil 'Alamin, segala puji dan syukur dipanjatkan kehadirat Allah SWT atas limpahan rahmat dan karunianya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Monitoring *Online* Ketersediaan *Slot* Parkir Berbasis Kamera Via *Website* dengan Raspberry Pi 3". Adapun tujuan dari penyusunan Tugas Akhir ini adalah sebagai salah satu persyaratan untuk mendapatkan gelar sarjana teknik pada bidang studi Teknik Elektronika, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam kesempatan ini penulis ingin mengucapkan terima kasih kepada pihak-pihak yang banyak berjasa terutama dalam penyusunan tugas Akhir ini, yaitu :

1. Kedua orang tua dan kakak yang senantiasa memberikan doa, nasihat, motivasi, dukungan dan karena keberadaan merekalah penulis tetap semangat untuk menyelesaikan penelitian ini.
2. Dr. Muhammad Rivai, ST., MT. selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjaan penelitian tugas akhir ini.
3. Dr. Ir. Hendra Kusuma, M.Eng.Sc. selaku dosen pembimbing kedua, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjaan penelitian tugas akhir ini.
4. Astria Nur Irfansyah, Rachmad Setiawan, Siti Halimah dan Tri Arief selaku dosen penguji yang telah membatu untuk menyelesaikan tugas akhir ini.
5. Rekan-rekan yang banyak membantu dalam penyelesaian tugas akhir ini.

Besar harapan penulis agar tugas akhir ini dapat memberikan manfaat dan masukan bagi banyak pihak. Oleh karena itu penulis mengharapkan kritik, koreksi, dan saran dari pembaca yang bersifat membangun untuk pengembangan ke arah yang lebih baik.

Surabaya, Juli 2017

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

TABLE OF CONTENT

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Peningkatan penduduk di dunia semakin pesat. Hal ini memberi pengaruh pada peningkatan jumlah produksi kendaraan setiap tahunnya. Dari peningkatan ini, memiliki dampak pada lahan parkir di setiap wilayah. Contohnya pada tempat hiburan atau tempat umum, terkadang jumlah kendaraan yang datang tidak menentu setiap harinya. Terkadang tidak membutuhkan waktu yang lama untuk mencari slot parkir. Tetapi sering pula menghabiskan waktu hanya untuk mecari parkir. Hal ini menyebabkan kerugian waktu.

Terdapat beberapa cara untuk memberikan informasi mengenai slot parkir. Salah satunya menggunakan metode sensor laser *scanner*, sensor ultrasonik dan kamera sebagai penghitungan jumlah kendaraan yang masuk dan keluar, serta mengetahui slot kosong pada parkir [1]. Tetapi sering kali informasi yang diterima tidak sesuai dengan keadaan sesungguhnya. Dan informasi yang diterima tidak terlalu detail, karena tidak mencantumkan slot mana saja yang kosong. Cara ini belum cukup membantu untuk meminimalisir waktu untuk mencari parkir.

Ada pula yang menggunakan metode histogram dengan menggunakan kamera webcam. Metode ini akan diuji sebagai input awal kemudian dinormalisasi untuk setiap lahan parkir yang di set sebagai tempat pengujian. Pada slot parkir yang kosong akan diberi label negatif (-1), sementara slot parkir yang terisi akan diberi label positif (+1) [2].

Pada tugas akhir ini metode yang akan digunakan yaitu metode *template matching* dengan menggunakan *webcam*. Metode tersebut akan mendeteksi bahwa slot tersedia atau tidak. Setelah kamera mengolah data di Rapsberry Pi 3, kemudian dikirimkan ke *webserver* dan disimpa di *database*. Apabila data telah diterima pada *webserver*, data dapat ditampilkan pada *website* yang akan disediakan sebagai *interface* untuk pengguna.

1.2. Perumusan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini adalah:

1. Bagaimana cara membedakan slot yang kosong atau yang sudah terisi .
2. Bagaimana pengiriman data dapat dimonitoring secara jarak jauh.
3. Implementasi metode pendeteksian slot kosong.

1.3. Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah

1. Kamera diletakkan fix tidak bergerak atau berubah tempat.
2. Pendeteksian membutuhkan cahaya yang tetap, perubahan cahaya dapat merubah hasil.
3. Jumlah slot parkir maksimal 3 slot.

1.4. Tujuan

Tujuan yang ingin dicapai dalam perancangan ini adalah:

1. Untuk membedakan slot kosong atau sudah terisi dengan menggunakan kamera dengan metode t*emplate matching*.
2. Pengiriman data melalui media internet.
3. Pengimplementasian menggunakan kamera dan raspberry pi 3.

1.5. Metodologi

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan Tugas Akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, dan artikel-artikel di internet dan forum-forum diskusi internet.

2. Perancangan *Hardware*

Perencangan *hardware*, secara umum meliputi kamera dan raspberry pi 3. Kamera akan diletakkan secara fix tidak bergerak atau berubah tempat. Kamera diletakkan tepat di atas slot parkir. Slot parkir yang akan

dideteksi yaitu 3 slot parkir dengan menggunakan miniatur. Miniatur tersebut merupakan perbandingan skala dari lapangan parkir dosen Departemen Teknik Elektro ITS.

3. Perancangan *Software*

Perancangan *software*, terdapat dua tahap yang dilakukan. Pertama *software* untuk mengolah citra yang dilakukan pada OpenCV dengan menggunakan bahasa python. Pada OpenCV akan diolah informasi yang telah diberikan dari kamera berupa informasi slot kosong atau terisi. Metode yang digunakan untuk mendeteksi yaitu *template matching*. *Software* kedua yaitu digunakan untuk menampilkan data yang dikirim untuk ditampilkan pada *website*. *Software* tersebut menggunakan PHP, Java Script, dan HTTP.

4. Pengujian Alat

Pengujian alat dilakukan untuk menentukan keadaan dari sistem yang telah dirancang. Pengujian untuk melihat apakah *software* dan *hardware* dapat bekerja secara baik. Untuk pengujian dapat dilakukan dalam tiga tahap. Pertama adalah mendeteksi slot parkir yang tersedia atau tidak. Setelah terdeteksi, kemudian mengirimkan data ke rapsberry pi 3. Berikutnya, data yang sudah diterima pada raspberry pi 3, kemudian dikirimkan ke *webserver*. Tahap terakhir, data pada *webserver* di tampilkan pada *device* sebagai *interface* kepada pengguna.

5. Analisa

Analisa dilakukan terhadap hasil dari pengujian sehingga dapat ditentukan tingkat keberhasilan dari *software* dan *hardware* yang telah dibuat. Dari hasil yang diperoleh, dapat dilakukan analisa apabila terdapat *error*.

6. Penyusunan Laporan Tugas Akhir

Tahap penulisan laporan tugas akhir adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Laporan tugas akhir berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, tinjauan pustaka dan teori penunjang, perancangan sistem, pengujian, dan penutup.

7. Penulisan Makalah Jurnal POMITS

Penulisan jurnal ilmiah berdasarkan data hasil penelitian tugas akhir.

1.6. Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

1. BAB I : PENDAHULUAN

Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

2. BAB II : TINJAUAN PUSTAKA DAN TEORI PENUNJANG

Bab ini menjelaskan tentang teori penunjang dan *literature* yang dibutuhkan dalam pengerjakan tugas akhir ini. Dasar teori yang menunjang meliputi OpenCV, citra digital, *template matching*, Mikrokontroler Raspberry Pi 3, dan PHP. Bagian ini memaparkan mengenai beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

3. BAB III : PERANCANGAN SISTEM

Bab ini menjelasakan tentang perencanaan sistem baik perangkat keras (*hardware*) maupun perangkat lunak (*software*) untuk sistem pendeteksian slot parkir.

4. BAB IV : PENGUJIAN

Pada bab ini akan menjelaskan hasil uji coba sistem beserta analisanya.

5. BAB V : PENUTUPAN

Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran untuk pengembangan lebih lanjut.

1.7. Relevansi dan Manfaat

Hasil yang didapat dari tugas akhir ini diharapkan dapat membantu pengunjung dibeberapa tempat hiburan atau tempat umum untuk lebih mudah dalam pencarian parkir. Kemudian tidak memerlukan waktu yang lama untuk mencari dimana slot parkir yang kosong.

BAB II
TEORI PENUNJANG

Pada bab ini diuraikan mengenai literatur yang digunakan sebagai penunjang dalam penyusunan Tugas Akhir. Diantaranya menggunakan Sistem Parkir pada Tunjungan Plaza 4, Raspberry Pi 3, OpenCV, algoritma *Template Matching*, dan PHP (*Personal Home Page*).

2.1 Sistem Perparkiran Mobil

Pada studi lapangan ini dilakukan di Tunjungan Plaza 4. Terdapat beberapa sistem yang digunakan, yaitu papan penunjuk jumlah yang terpasang di pintu masuk, penunjuk jumlah dan arah yang diletakkan disetiap lantainya dan sensor ultrasonik beserta LED.

Gamba 2.1 Papan Penunjuk Jumlah pada Pintu Masuk

Gambar 2.2 Papan Penunjuka di Dalam Gedung

Pada gambar 2.1 merupakan papan penunjuk jumlah pada pintu masuk. Dengan berfokus pada bagian TP4 saja. Setelah dilakukan pengamatan, antara informasi yang diberikan pada papan informasi dengan keadaan di parkiran berbeda. Keadaan di parkiran lebih banyak jumlahnya, sebab papan penunjuk yang disediakan di dalam gedung menunjukkan jumlah yang lebih banyak. Pada gambar 2.2 dapat dilihat bahwa tanda panah lurus menunjukkan kondisi yang ada pada bagian depan dari alat penunjuk tersebut. Sedangkan pada tanda panah ke kanan menunjukkan kondisi yang kosong di beberapa lantai berikutnya. Setelah dilakukan pengamatan antara informasi yang ada pada papan penunjuk di dalam gedung ini sudah sesuai.

Kemudian untuk sensor yang digunakan untuk mendeteksi parkiran yaitu menggunakan sensor ultrasonik, yang terdapat pada gambar 2.3. sensor tersebut masih bisa mendeteksi orang yang berdiri di bawah sensor tersebut, sehingga LED yang digunakan sebagai indikator berubah warna menjadi merah. Dapat dilihat pada gambar 2.4. Kemudian sensor ultrasonik akan mendeteksi kurang lebih membutuhkan waktu 2-4 detik untuk merubah warna LED.

Gambar 2.3 Sensor Ultrasonik

Gambar 2.4 Kondisi Terdapat Orang pada Sensor Ultrasonik

2.2 Kamera *Webcam*

Kamera *webcam* berfungsi untuk pengambilan citra. Terdapat banyak tipe dan merek kamera *webcam* yang diperjual belikan. Pada tugas akhir ini digunakan kamera *webcam* logitech c525. Terdapat beberapa spesifikasi dari kamera ini, yaitu :

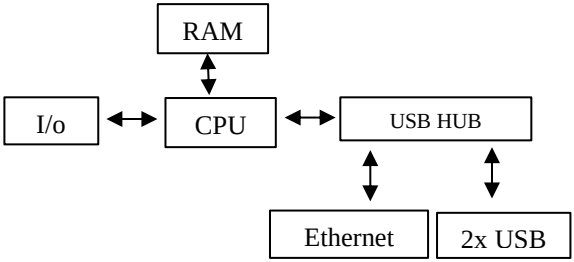
1. Perekaman video HD hingga 1280x720 piksel.
2. *Autofocus*.
3. Perputaran kamera hingga 360 derajat.
4. Resolusi foto mencapai 8 megapiksel.

Dari beberapa spesifikasi yang tertera, fungsi dari kamera *webcam* ini dapat membantu untuk proses pengambilan citra yang dibutuhkan.

Gambar 2.5 Kamera *Webcam* Logitech C525

2.3 Raspberry Pi 3

Raspberry Pi adalah modul mikrokomputer yang memiliki *input* dan *output* digital *port* seperti pada *board* mikrokontroler. Raspberry Pi merupakan sebuah komputer mini ukuran saku, yang terdiri dari beberapa generasi. Namun dari generasi tersebut, dibagi kembali menjadi beberapa tipe Raspberry Pi *board*. Terdapat tipe A, A+, B dan B+. Perbedaan antara kedua tipe tersebut pada RAM dan *Port* LAN. Untuk tipe A memiliki RAM 256 Mb dan tidak memiliki *port* LAN (ethernet), kemudian untuk tipe B memiliki RAM 512 Mb dan memiliki *port* LAN.

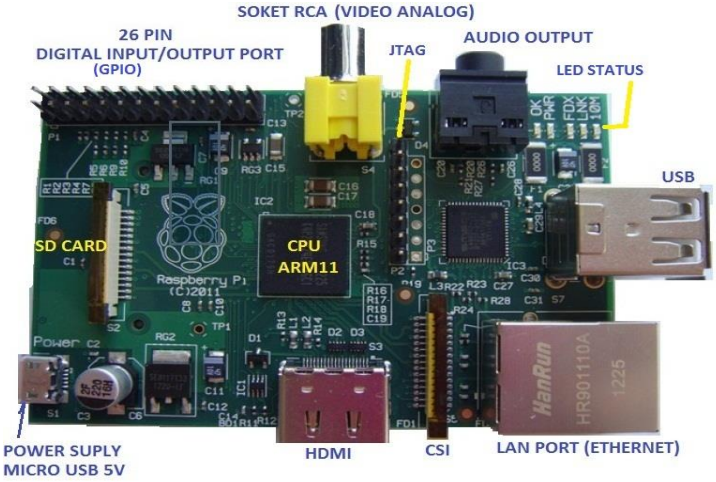


Gambar 2.6 Diagram Blok Raspberry Pi

Pada blok diagram di atas menjelaskan bahwa CPU berguna sebagai *processor* yang dapat mengakses pi GPIO sebagai *wiring*, kemudian RAM digunakan sebagai menyimpan data sementara. Setelah itu terdapat USB HUB yang bercabang menjadi ethernet dan USB. Ethernet disini berguna sebagai *remote* desktop. Untuk USB terdapat 2 *port* yang bisa digunakan untuk flashdisk, webcam, kamera, mouse, *keyboard*, dan sebagainya.

Raspberry Pi *board* memiliki *input* dan *output* beserta kegunaannya, yaitu:

1. HDMI, berguna untuk menyambungkan ke layar atau TV yang terdapat *port* HDMI juga. Cara menghubungkan port HDMI Raspberry Pi ke layar dengan menggunakan kabel konverter HDMI ke VGA.
2. Video analog (RCA *port*), dapat dihubungkan ke TV apabila tidak mempunya monitor PC.
3. Audio *output*, berguna untuk mengeluarkan suara dengan menggunakan 3.5 mm jack.
4. 2 buat *port* USB, digunakan untuk menghubungkan *keyboard* dan *mouse*.
5. 26 pin I/O digital.
6. CSI *port* (*Camera Serial Interface*), merupakan *port* khusus dapat menyambungkan kamera Pi.
7. DSI (*Display Serial Interface*).
8. LAN *port*, dapat digunakan untuk menyambungkan Raspberry Pi menuju modem dengan menggunakan kabel LAN.
9. SD *card slot*, berguna untuk menyimpan SD *card* memori untuk menyimpan sistem operasi yang berfungsi seperti *hardisk*.



Gambar 2.7 Raspberry Pi *Board*

Generasi yang akan digunakan yaitu generasi ke 3. Raspberry Pi 3 ini memiliki beberapa fungsi yaitu:

1. Sebagai *file server*

Generasi pertama maupun kedua dapat berfungsi sebagai *file server* dengan pengaturan yang tepat. Berbasis Samba, dapat memanfaatkan *hard disk* eksternal (dihubungkan ke Raspberry Pi via *port* USB) untuk menjadi media penyimpan data.

2. Sebagai *downloder server*

Downloader server ini merupakan mesin yang akan terhubung ke jaringan Torrent menggunakan aplikasi transmission atau deluge tanpa layar monitor dan *keyboard*. Pengontrolan dan pengelolaan data yang akan diunduh pada Raspberry Pi 3 ini bisa dilakukan via web. Hanya mengakses alamat web transmission itu menggunakan *browser*, baik via komputer *desktop* maupun via hp.

3. Sebagai *access point*

Untuk dijadikan *access point* diperlukan penambahan aplikasi seperti hostapd dan dhcp server. Hostapd merupakan aplikasi server untuk mengelola adapter WiFi yang akan berfungsi sebagai *access point* dan

sekaligus melakukan proses validasi terhadap permintaan koneksi dari klien yang akan terhubung. Sementara dhcp server berfungsi menyediakan alamat IP yang akan dipakai oleh klien setelah proses alidasinya berhasil dilakukan oleh hostapd.

4. Sebagai *server* DNS

Server DNS berfungsi sebagai *caching* yang akan menampung semua informasi DNS dari *server* DNS yang memiliki otoritas untuk menjawab semua pertanyaan domain. Namun demikian, keberadaan *server* DNS ini dalam jaringan lokal bisa mengurangi permintaan ke luar jaringan.

Pada saat menggunakan Raspberry Pi diperlukan *operating system* yang dijalankan dari SD *card* pada *board* Raspberry. OS yang sering digunakan yaitu Arch Linux ARM, Debuan GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Inferno, Raspbian OS, RISC OS dan Slackware Linux. OS disimpan pada SD *card*, kemudian pada saat *booting* OS hanya dapat digunakan dari SD *card.*

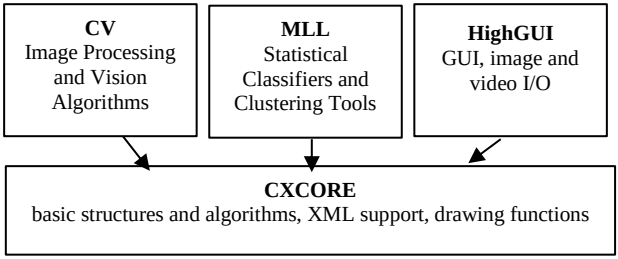
2.4 OpenCV

OpenCV merupakan suatu *library* dari *computer vision* yang *open source* (gratis digunakan baik untuk urusan akademik ataupun komersil) [3]. OpenCV (*Open Computer Vision*) merupakan salah satu produk *open source* yang merupakan sebuah API (*Application Programming Interface*) dengan *library* yang sudang sangat familiar pada pengolahan citra *computer vision*. OpenCV dapat diterapkan pada pemrograman C++, C, Python, Java, dan MATLAB. OpenCV dapat pula digunakan untuk sistem operasi Windows, Linux, Android dan Mac OS.

Computer vision adalah salah satu cabang dari ilmu pengolahan citra yang dapat mengolah komputer agar bisa terlihat nyata. OpenCV mempermudah bisnis-bisnis untuk memanfaatkan dan memodifikasi kode. *Library* OpenCV mempunyai lebih dari 2500 algoritma yang telah dioptimalkan. Dimana meliputi sebuah himpunan menyeluruh dari keduanya yaitu klasi dan seni, beberapa algoritma *computer vision* dan *mechine learning*. Algoritma-algoritma tersebut dapat digunakan untuk mendeteksi dan mengenali wajah, mengindentifikasi objek, mengklasifikasi tindakan manusia dalam video, mengikuti jejak perpindahan objek, mengekstrak model-model 3D objek, menghasilkan titik awan 3D dari kamera stereo, dan lain sebagainya.

Di dalam OpenCV terdapat 4 *library* utama yang bisa digunakan sesuai kebutuhan, yaitu:

1. CV : digunakan sebagai algoritma *image processing* dan *vision*.
2. Highgui : digunakan untuk GUI, gambar dan video I/O.
3. CXCORE : digunakan untuk struktur data, mendukung XML dan fungsi-fungsi grafis.
4. ML : digunakan untuk *machine learning library*.



Gambar 2.8 Struktur dan Konten OpenCV

2.5 Pengolahan Citra Digital

Citra atau yang sering disebut dengan gambar merupakan informasi yang berbentuk visual. Cara yang diperlukan untuk memperoleh citra yaitu dengan penangkapan kekuatan sinar yang dipantulkan oleh objek. Pada saat sumber cahaya menerangi objek, pemantulan dari objek akan kembali sebagai cahaya tersebut. Pantulan tersebut dapat ditangkap oleh alat-alat pengindraan optik. Salah satunya yaitu menggunakan kamera.

Citra digital adalah tampilan yang berasal dari citra dengan bentuk pendekatan berupa sampling dan kuantisasi. Hasil dari proses digitalisasi citra adalah sebuah matriks yang berisi nilai-nilai riil. Elemen-elemen dari matriks tersebut disebut *picture element* atau sering disebut piksel. Sebuah piksel mempunyai dua properti yaitu koordinat posisi dari piksel tersebut dan nilai dari piksel tersebut [4]. Pada citra digital memiliki sebuah fungsi 2D yang berupa f(x,y), yang berfungsi sebagai intensitas cahaya. Nilai dari x dan y merupakan koordinat spasial, kemudian nilai fungsi di setiap titik (x,y) merupakan tingkat dari *grayscale* citra pada titik tersebut. Citra digital dinyatakan dengan matriks dimana baris dan kolom menunjukkan titik pada citra tersebut dan elemen matriksnya (*pixel*)

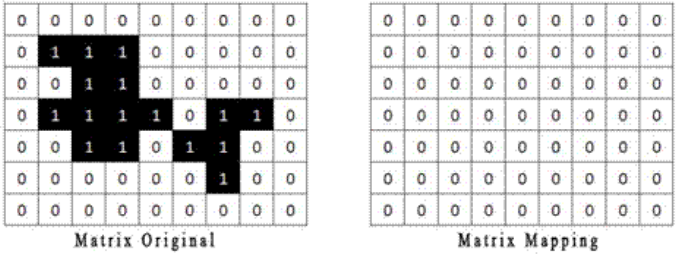
menyatakan tingkat *grayscale* pada titik tersebut. Matriks pada citra digital berukuran NxM (tinggi x lebar), dimana:

N = jumlah baris	$0 < y \leq N - 1$
M = jumlah kolom	$0 < x \leq M - 1$
L = derajat keabuan	$0 \leq f(x,y) \leq L - 1$

Pada sebuah citra terdapat beberapa cara penyimpanan untuk menentukan jenis citra yang akan dibentuk. Beberapa jenis citra yang seringkali digunakan antara lain, yaitu:

2.4.1 Citra Biner (Monokrom)

Pada citra biner memiliki dua warna yaitu hitam dan putih. Nilai yang dimiliki oleh hitam yaitu 1, sedangkan sebaliknya untuk warna putih memiliki nilai 0. Untuk menyimpan menyimpan kedua warna tersebut dibutuhkan memori sebesar 1 bit.



Gambar 2.9 Citra Biner

2.4.2 Citra *Grayscale* (Skala Keabuan)

Citra *grayscale* disebut juga citra intensitas atau citra *gray level* [5]. Berbeda dengan citra biner, memori yang digunakan 8 bit dan ssetiap *pixel* pada citra *grayscale* memiliki nilai 0 sampai 255. Nilai 0 merupakan warna putih dan 255 merupakan warna putih. Di antara 0 dan 255 merupakan warna abu-abu, warna tersebut bergantung pada kecerahan dari *pixel* tersebut. Semakin tinggi nilainya, maka semakin cerah *pixel* tersebut. Gambar 2.10 merupakan rentang *gray level* dari hitam hingga putih yang identik dengan terang.

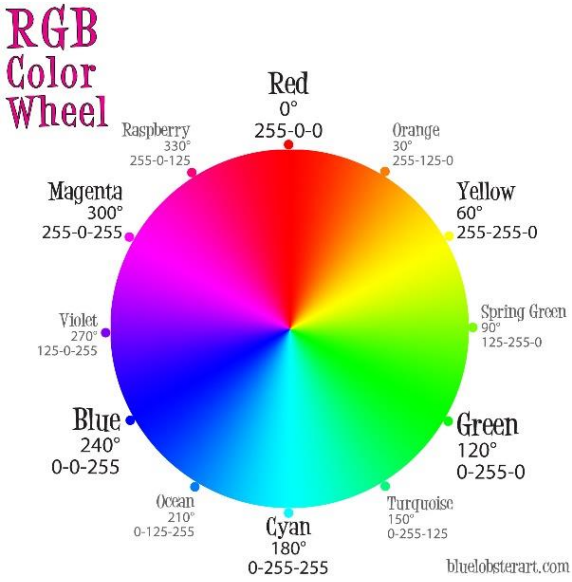
Gambar 2.10 Rentang *Gray Level*

Citra *grayscale* seringkali digunakan dalam pengolahan citra, sebab jumlah data yang tidak terlalu besar. Hal ini dapat mempercepat proses pengolahan data. Selain itu, *grayscale* tidak menghilangkan informasi penting dari citra. Oleh karena itu, pada saat citra asli yang berupa citra warna akan diproses untuk pengolahan citra, dilakukan konversi menjadi *grayscale* dengan cara mencari nilai rata-rata dari nilai kanal merah, hijau dan biru. Gambar 2.6 menunjukkan perbedaan antara citra warna dengan citra *grayscale*.

Gambar 2.11 Citra RGB dan Citra *Grayscale*

2.4.3 Citra Warna (*True Color*)

Citra warna adalah citra yang setiap pikselnya ditentukan oleh tiga nilai masing-masing untuk komponen merah, biru, dan hijau [6]. Penyimpanan disetiap warna dasarnya sebesar 8 bit atau 1 *byte*, artinya setiap warna mempunyai gradasi sebanyak 255 warna. Jika tipe datanya 16 bit, rentang nilai pada setiap kanal adalah antara 0 sampai 65535. Jumlah bit yang lebih banyak akan memberi variasi warna yang lebih banyak pula. Akan tetapi, jika jumlah bit yang lebih sedikit akan mempercepat pengolahan citra.



Gambar 2.12 Citra Warna

Dalam sebuah citra digital memiliki elemen-elemen yang berfungsi untuk proses pengolahan citra. Terdapat beberapa elemen pada citra digital, yaitu:

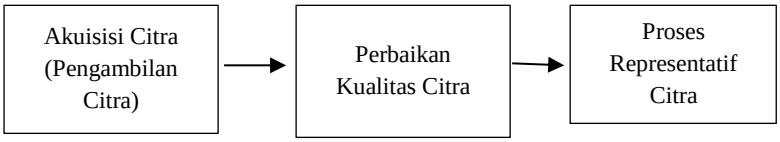
1. Kecerahan (*Brightness*) merupakan intensitas cahaya yang dipancarkan oleh piksel dari citra yang dapat ditangkap oleh sistem pengelihatan. Kecerahan pada sebuah titik (piksel) di dalam citra merupakan intensitas rata-rata dari suatu area yang melingkupinya.
2. Kontra (*Contrast*), menyatakan sebaran terang dan gelap dalam sebuah citra. Pada citra yang baik, komposisi gelap dan terang tersebar secara merata.
3. Kontur (*Contour*), adalah keadaan yang ditimbulkan oleh perubahan intensitas pada piksel-piksel yang berdekatan. Karena adanya perubahan intensitas, mata mampu mendeteksi tepi-tepi objek di dalam citra.
4. Warna, sebagai persepsi yang ditangkap disitem visual terhadap panjang gelombang cahaya yang dipantulkan oleh objek.

5. Bentuk (*Shape*), adalah properti intrinsik dari objek 3 dimensi yang berperan utama untuk sistem visual manusia.
6. Tekstur (*Texture*), dapat dicirikan dengan distribusi spasial dari derajat *grayscale* di dalam sekumpulan piksel-piksel yang berdekatan. Tekstur adalah karakteristik yang dimiliki oleh suatu daerah yang cukup besar, sehingga secara alami sifat-sifat tersebut dapat berulang dalam daerahnya. Tekstur merupakan ketentuan pola-pola tertentu yang terbentuk dari susunan piksel dalam citra digital. Informasi tekstur dapat digunakan untuk membedakan sifat-sifat permukaan suatu benda dalam citra yang berhubungan dengan kasar atau halus. Kemudian sifat spesifik dari kekasaaran dan kehalusan permukaan yang sama sekali terlepas dari warna permukaan tersebut.

Pengolahan citra merupakan sebuah proses pengolahan dengan *input* yaitu citra. Hasil yang didapatkan dari proses tersebut berupa citra atau sekumpulan karakteristik yang berhubungan dengan citra. Terdapat beberapa fungsi yang dimiliki oleh pengolahan citra, yaitu:

1. Digunakan untuk proses memperbaiki kualitas pada citra agar lebih mudah pada saat pengolahan.
2. Digunakan untuk mentransformasikan citra menjadi citra lain, agar citra tersebut dapat dilakukan pemampatan citra (*image compressio*) sebagai tahap awal pada *image preprocessing*.

Proses pengolahan citra secara diagram dimulai dengan pengambilan citra, kemudian perbaikan kualitas citra, dan proses representatif citra.



Gambar 2.13 Diagram Blok Proses Pengolahan Citra

Pengolahan citra dibagi menjadi tiga karakter, diantaranya adalah:

1. Kategori pengolahan tingkat rendah (*Low Level Processing*), dalam hal ini melibatkan operasi-operasi sederhana seperti pra-

pengolahan citra untuk mengurangi *noise*, pengaturan kontras, dan pengaturan ketajaman citra. Pada pengolahan citra kategori rendah, memiliki *input* dan *output* citra.

2. Kategori pengolahan tingkat menengah (*Mid-Level Processing*), melibatkan operasi-operasi seperti segmentasi, dan klasifikasi citra, deskripsi objek dan klasifikasi objek terpisah. Pada pengolahan citra kategori menengah ini memiliki *input* berupa citra, kemudian *output* berupa fitur citra yang telah dipisahkan dari citra *input*.
3. Kategori pengolahan tingkat tinggi (*High-Level Processing*), merupakan citra yang telah dikenali sebagai objek kemudian diimplementasikan sebagai aplikasi, serta melakukan fungsi-fungsi yang berhubungan dengan penggabungan dengan vision.

Dari ketiga tahap pengolahan citra digital tersebut, terdapat pula teknik-teknik yang diperlukan untuk mengolah citra digital. Berikut beberapa teknik pengolahan citra digital, yaitu *image enhacement, image restortion, morphological processing, image segmantation*, dsb.

2.5 *Template Matching*

Template matching adalah salah satu teknik dalam pengolahan citra digital yang berfungsi untuk mencocokan tiap-tiap bagian dari suatu citra dengan citra yang menjadi template (acuan) [7]. *Template* tersebut akan ditempatkan pada pusat bagian citra kemudian dihitung titik yang paling sesuai. Nilai yang paling besar menandakan bahwa antara gambar dan template merupakan citra yang paling sesuai dengan citra *input*. Terdapat beberapa metode pada *template matching* yaitu *square difference matching method, correlation matching method,* dan *correlation coefficient matching method*.

1. *Square difference matching method* (SQDIFF) merupakan salah satu dari metode dari *template matching.* Cara pencocokannya dengan perhitungan bedaan kuadran. Jika hasil yang diperoleh 0 atau mendekati 0 maka akan terjadi kecocokan [8].

$$R(x,y) = \sum_{x',y'}[T(x',y') - I(x+x',y+y')]^2 \qquad (2.1)$$

2. *Correlation matching method* (CCORR) merupakan metode yang melakukan pencocokan dengan hasil dari perkalian. Jika memiliki nilai lebih kecil atau 0 maka tidak terjadi kecocokan. Sebaliknya, jika nilai yang diperoleh lebih besar maka akan terjadi kecocokan.

$$R(x,y) = \sum_{x',y'}[T(x',y').I(x+x',y+y')]^2 \qquad (2.2)$$

3. *Correlation coefficient matching method* (CCOEFF). Pada metode ini akan melakukan pencocokan antara *template* dan gambar dengan perhitungan rata-ratanya. Batas untuk nilai pada metode ini adalah 1 sampai -1. Hasil yang paling cocok maka memiliki nilai 1, dan sebalikya.Berikut perhitungan untuk menentukan letak *template* pada sebuah gambar.

$$R(x,y) = \sum_{x',y'}(T'(x',y').I'(x+x',y+y')) \qquad (2.3)$$

Dimana :
- T' = Template
- I' = Gambar yang Dicocokkan
- (x,y) = Koordinat Gambar yang Dicocokkan
- (x',y') = Koordinat Template

Ketiga metode pada *template matching* ini terdapat pula tipe untuk *normalized.* Tipe ini berguna untuk mengurangi efek dari perbedaan cahaya antara gambar dengen *template*. Sebagai contoh pada *normalized correlation coefficient matching method.* Rumus yang ada dirata-ratakan kemudian dibagi.

$$Z(x,y) = \sqrt{\sum_{x',y'} T'(x',y')^2 . \sum_{x',y'} I'(x+x',y+y')^2} \qquad (2.4)$$

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y').I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}T'(x',y')^2 . \sum_{x',y'}I'(x+x',y+y')^2}} \qquad (2.5)$$

2.6 PHP

PHP pada awalnya diciptakan untuk membuat suatu halaman *website* yang dinamis dan sederhana, untuk saat ini merupakan bahasa pemrograman yang sangan populer [9]. PHP adalah bahasa *scripting* web HTML-embedded. Hal tersebut berarti kode PHP dapat disisipkan ke dalam HTML halaman web. Pada saat halaman PHP diakses, maka kode PHP dibaca oleh server. *Output* dari PHP pada halaman biasanya dikembalikan sebagai kode HTML yang dapat dibaca oleh *browser*. Karena kode PHP diubah menjadi HTML sebelum halaman dibuka, jadi kode PHP tidak dapat dilihat pada halaman tersebut. Dengan demikian halaman PHP aman untuk mengakses *database* dan informasi lainnya.

PHP memiliki beberapa kelebihan dari bahasa programming web lainnya, diantara lain:

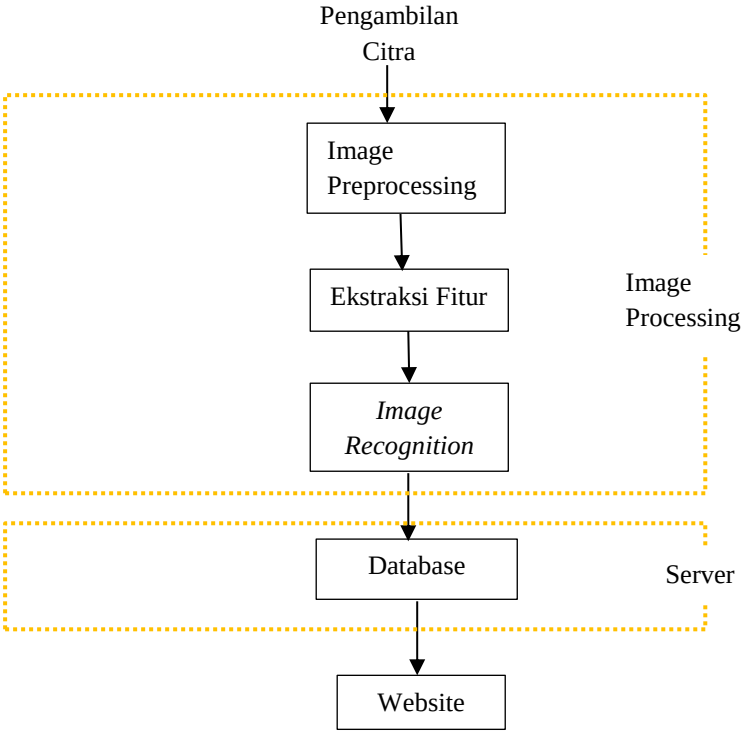
1. Bahasa pemrograman PHP adalah sebuha bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. Webserver yang mendukung PHP dapat ditemukan pada apache, IIS, Lighttpd, dan Xitami.
3. Dalam pengembangan lebih mudah.
4. Dalam sisi pemahaman, PHP merupakan bahasa yang mudah dimengerti.
5. PHP adalah bahasa *open source* yang dapat digunakan pada Linux, Unix, Macintosh, dan Windows.

Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai perancangan sistem. Dimulai dengan pengambilan sampel slot parkir yang akan diolah sebagai pengenalan citra. Metode yang digunakan untuk pengolahan citra yaitu *template matching*. Setelah citra diolah, hasilnya akan ditampilkan pada *website* sebagai *interface*. Dalam bab ini akan dijelaskan cara kerja dari pengolahan citra dengan metode *template* matching, pengiriman data dari Raspberry Pi 3 menuju *webserver*, dan pembuatan *website* dengam pemrograman PHP.

Pada tugas akhir ini, pengolahan citra digunakan untuk mendeteksi slot parkir yang kosong. Menggunakan kamera webcam, kemudian diperoleh citra yang diingikan. Setelah diperoleh hasil citra yang diinginkan, dilanjut dengan metode *tempate matching* yang akan membedakan antara slot yang kosong dengan yang sudah terisi. Perbedaan yang akan terlihat pada pengolahan citra yaitu, terdapat dua kondisi yaitu terdapat satu *pattern* berupa huruf P dan terdapat dua *pattern* berupa huruf P dan angka. Apabila huruf dan atau angka tersebut terdeteksi oleh kamera, maka status slot parkir kosong. Kemudian sebaliknya, apabila huruf tidak terdeteksi oleh kamera, maka statusnya telah terisi mobil. Pengolahan citra ini dapat membedakan antara objek yang berada di slot parkir berupa mobil atau bukan. Selain itu, pengolahan citra dapat pula membedakan letak dari slot parkir, sehingga didapatkan info berupa slot parkir mana yang kosong. Info tersebut dikirimkan menuju *database* untuk disimpan, diolah dan ditampilkan pada *website*. Pada *webserver* bahasa pemrograman yang digunakan yaitu PHP dan javascript. Di dalam PHP dapat digunakan sebaga *webserver* dan javascript digunakan untuk membuat tampilan pada *website*. *Database* dan *webserver* disimpan pada cloud, sebab untuk mengurangi beban pada Raspberry Pi 3 agar tidak terlalu berat saat mendeteksi slot parkir. Proses dari sistem ini dapat dilihat dari blok diagram pada gambar 3.1.



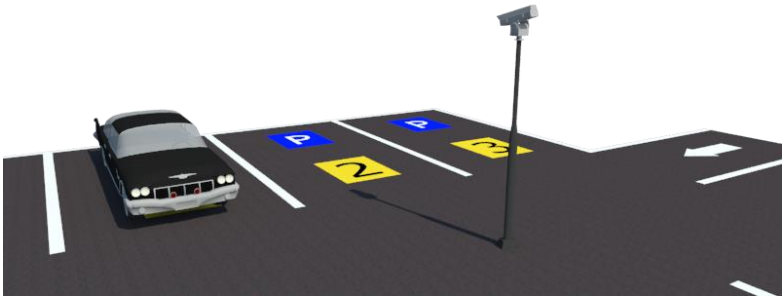
Gambar 3.1 Diagram Blok Sistem

3.1 Perancangan *Hardware*

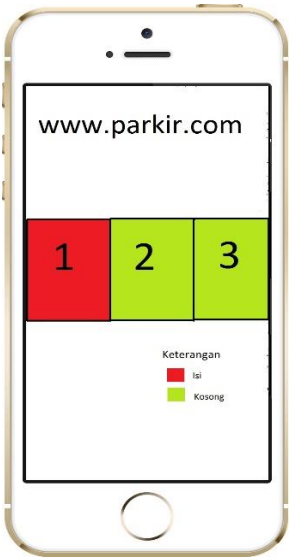
Dalam gambar 3.2 mengilustrasikan sistem yang akan dibuat. Kamera diletakkan di depan slot parkir. Hal ini dikarenakan, apabila kamera diletakkan pada bagian samping atau ujung-ujung slot parkir, maka kamera tidak dapat memberikan gambar dan informasi berupa *pattern* secara jelas. Posisi kamera yang berada di tengah-tengah, dapat memberikan informasi berupa tiga slot parkir.

Pada gambar 3.3 mengilustrasikan tampilan pada *website*. Data yang akan ditampilkan di *website*, sesuai dengan keadaan pada yang sedang berlangsung. Terdapat pula keterangan slot mana saja yang kosong atau

terisi. Apabila slot kosong akan ditandakan dengan warna hijau, dan warna merah untuk slot yang sudah terisi.



Gambar 3.2 Rancangan Wilayah



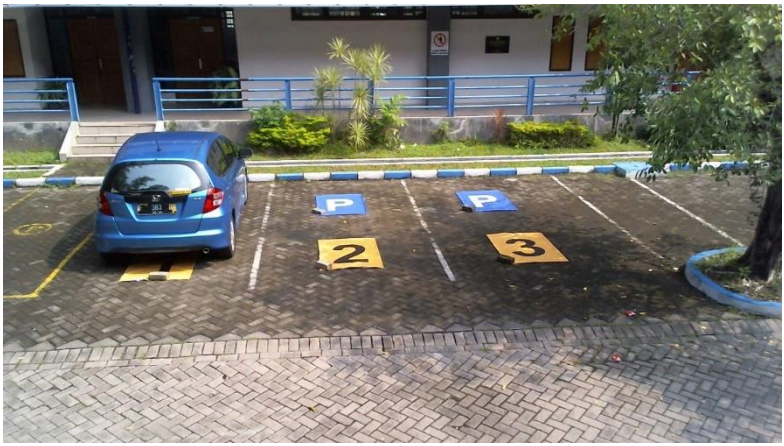
Gambar 3.3 Tampilan pada *Website*

3.2 Perancangan dan Pengambilan Sampel

Dalam tugas akhir ini digunakan tiga buah slot parkir sebagai sampel pengolahan citra. Pengambilan sampel dilakukan di lapangan parkir dosen Departemen Teknik Elektro ITS. Waktu yang dilakukan untuk pengambilan sampel yaitu siang hingga sore hari, sebab pada jangka waktu itu gambar dapat diperoleh secara jelas dan pencahayaannya masih baik. Pada gambar 3.3 merupakan wilayah slot parkir yang dijadikan sampel.

Gambar 3.4 Wilayah Slot Parkir Teknik Elektro ITS

Kemudian terdapat pula beberapa sampel yang menggambarkan beberapa kasus, contohnya terdapat mobil di salah satu slot atau terdapat dua mobil dari tiga slot atau setiap slot terisi oleh mobil. Pada gambar 3.5 hingga gambar 3.7 merupakan sampel dari slot parkir yang telah terisi oleh mobil.



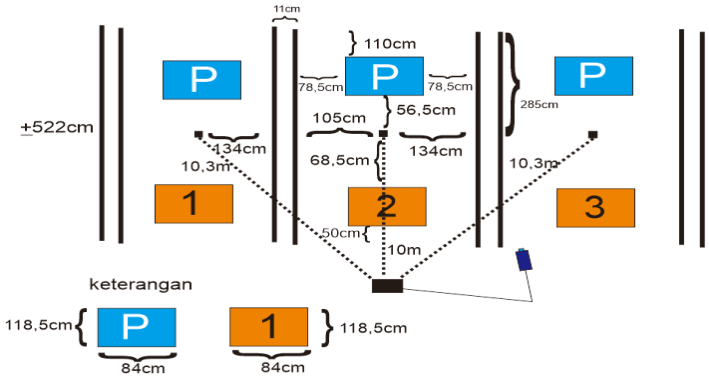
Gambar 3.5 Terdapat Satu Mobil di Salah Satu Slot Parkir

Gambar 3.6 Terdapat Dua Mobil dari Tiga Slot Parkir

Gambar 3.7 Slot Parkir Terisi Penuh

3.2.1 Pengukuran dan Pembuatan Miniatur

Setelah dilakukan pengambilan sampel, tahap selanjutnya membuat miniatur dari slot yang sudah ada. Pembuatan miniatur ini diawali dengan menentukan skala antara slot parkir yang asli dengan miniatur. Perbandingan skalanya yaitu 1:10. Berikut hasil pengukurannya.



Gambar 3.8 Skala Asli Slot Parkir

Langkah selanajutnya setelah didapatkan hasil dari pengukuran, membuat slot parkir yang berukuran sepuluh kali lebih kecil dibandingkan dengan yang asli. Berikut hasil dari pembuatan miniatur slot parkir dengan perbandingan 1:10.

Gambar 3.9 Slot Parkir Miniatur

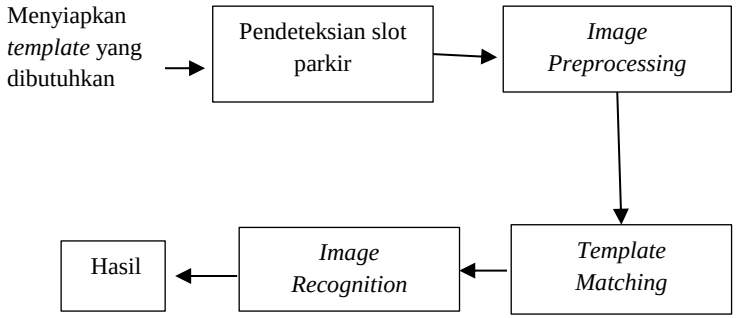
Gambar 3.9 disederhanakan dengan menghilangkan *pattern* angka seperti pada Gambar 3.10. Hal ini disebabkan untuk mengurangi *pattern* yang dideteksi, dan juga saran dari dosen pembimbing.

Gambar 3.10 Slot Parkir Miniatur Tanpa *Pattern* Angka

3.3 *Image Processing*

Image processing merupakan bagian terpenting dalam sistem ini. Pada proses *image processing* terdapat beberapa tahap. Tahap pertama yaitu ekstraksi fitur. Proses ini dilakukan untuk mendapatkan karakteristik yang terdapat dalam setiap citra yang diperoleh. Hal ini untuk mempermudah pada saat dilakukan tahap selanjutnya yaitu *image recognition*. Pada ekstraksi fitur terdapat metode yang dilakukan yaitu *template matching*, dengan dua kondisi yaitu hanya terdapat satu *pattern* yaitu huruf, dan kondisi kedua terdapat *pattern* huruf dan angka.

Template matching berguna untuk pendeteksian *pattern* slot parkir dan jumlah kendaraan. OpenCV merupakan perangkat lunak yang digunakan untuk pengolahan citra. Pada metode ini bahasa pemrograma yang digunakan yaitu python.

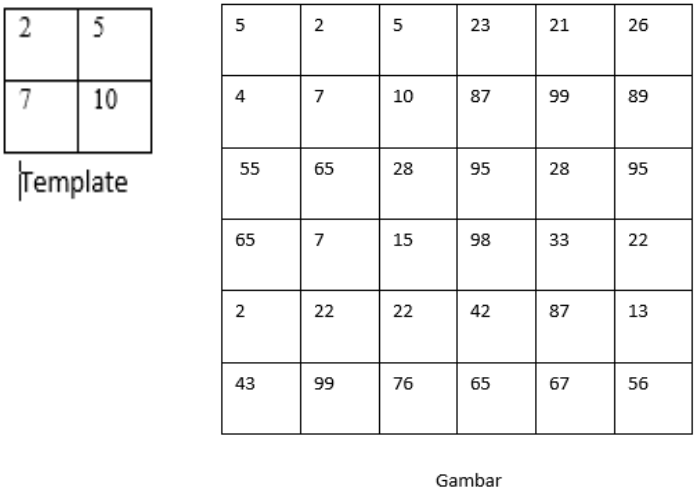


Gambar 3.11 Diagram Blok *Template Matching*

Proses metode *template matching* dimulai dengan menyiapkan *template* yang dibutuhkan untuk melakukan pendeteksian. Untuk kondisi satu *pattern*, *template* yang digunakan yaitu hanya huruf P. Dan untuk kondisi dua *pattern*, *template* yang digunakan yaitu huruf P, angka 1,2 dan 3. Kemudian pengambilan citra untuk mendeteksi slot parkir menggunakan kamera. Kamera yang digunakan yaitu kamera *webcam* dengan peletakan posisi yang tetap (tidak bergerak). Citra yang diambil dari setiap kamera harus diolah terlebih dahulu dengan pengkonversian citra menjadi *grayscale.* Pengkonversian ini dapat mempercepat proses pengolahan citra karena hanya memiliki lebar data sebesar 8 bit. Dalam

proses ini, digunakan *library* OpenCV yaitu cvtColor dengan koefisien COLOR_BGR2GRAY. Proses ini merupakan proses *image preprocessing*.

Tahap berikutnya untuk mendeteksi slot yang kosong atau tidak. *Templat* yang sudah disediakan kemudian dilakukan pencocokan dengan pengambilan citra secara *real-time*. Perhitungan dilakukan agar diketahui kecocokan dari *template* yang sudah disediakan, dengan *pattern* yang ada pada slot parkir. Perhitungan tersebut dilakukan menggunakan rumus pada *template matching* dengan metode *correlation coefficient matching method normalized*. Metode ini menggunakan hasil dari perhitungan rata-rata antara *template* dan gambar yang akan dicari kecocokannya. Kemudian *normalized* disini berguna agar lebih tahan terhadap pencahayaan. Berikut merupakan contoh dari perhitungan untuk mencari kecocokan *template*.



Gambar 3.12 Contoh *Template Matching*

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y').I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}T'(x',y')^2.\sum_{x',y'}I'(x+x',y+y')^2.}}$$

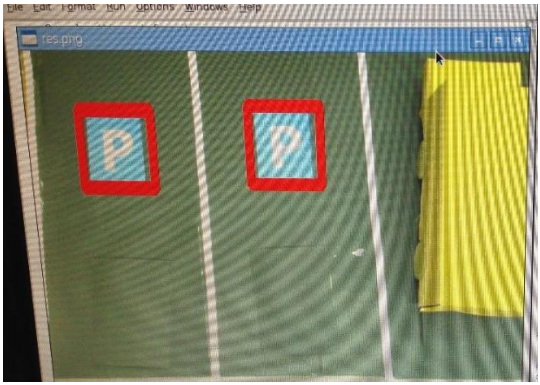
$$R(0,0) = \frac{(2.\ 5) + (5.2) + (7.4) + (10.7)}{\sqrt{(4 + 25\ + 49\ + 100).\ (25 + 4 + 16 + 49)}} = 0.91$$

$$R(1,0) = \frac{(2.\ 2) + (5.5) + (7.7) + (10.10)}{\sqrt{(4 + 25\ + 49\ + 100).\ (4 + 25 + 49\ + 100)}} = 1$$

Gambar 3.13 Perhitungan dari *Template Matching*

Pada gambar 3.13 merupakan perhitungan untuk mengetahui kecocokan antara *template* dan gambar, berikut penjelasannya. Dimisalkan terdapat *template* dengan ukuran 2x2 piksel, dan gambar ukuran 6x6 piksel. *Template* akan melakukan penggeseran ke kanan per piksel, dengan piksel pada sumbu y tetap. Perhitungan dimulai dari koordinat titik 0,0. Jika hasil yang diperoleh 1 maka terdapat kecocokan, sedangkan kalau hasilnya 0 sampai -1 maka tidak terdapat kecocokan. Hasil yang diperoleh pada koordinat 0,0 yaitu 0.911 sudah mendekati 1 tetapi tetap dilakukan kembali perhitungan. Kemudian bergeser ke koordinat berikutnya yaitu 1,0. Hasil yang diperoleh yaitu 1, berarti telah ditemukan letak *template.* Jika *template* telah ditemuka, maka *pattern* pun terdeteksi.

Hasil yang diperoleh akan ditunjukkan pada gambar 3.14 yang menunjukkan *template* melakukan pencocokan dengan satu *pattern*. Garis merah mreupakan hasil dari perhitungan yang mendekati nilai 1. Jika garis merah menunjukkan sangat tebal, maka bagian tersebut memiliki nilai yang paling besar. Dan sebaliknya, jika garis merah lebih tipis dari yang lainnya maka memiki nilai yang kecil. Perhitungan dari nilai ini merupakan proses dari *image* recognition. Apabila terdapat halangan atau kendaraan maka template tidak akan terdeteksi. Pada gambar 3.14 terdapat slot yang dimisalkan terdapat kendaraan, maka template tidak akan terdeteksi.



Gambar 3.14 Tampilan *Template Matching*

Pada gambar 3.15 merupakan tampilan *template matching* dengan menggunakan dua *pattern*, yaitu angka dan huruf. Perhitungan yang dilakukan untuk mencari *template* yang cocok pun sama. Tampilan pada gambar 3.15 ini tidak tedapat kendaraan.

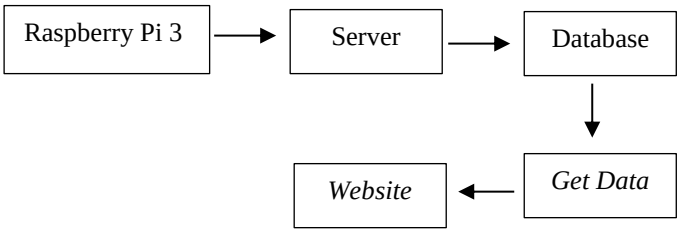
Gambar 3.15 Tampilan *Template Matching* dengan Dua *Pattern*

Untuk mengetahui slot yang terdeteksi oleh template, maka diberi nilai 0 dan 1. Jika *template* terdeteksi pada gambar, maka akan bernilai 0 dan sebaliknya. Jika template tidak terdeteksi maka akan bernilai 1.

Kemudian untuk menentukan letak setiap slot yaitu dengan menentukan batas koordinat disetiap slotnya.

3.4 *Website*

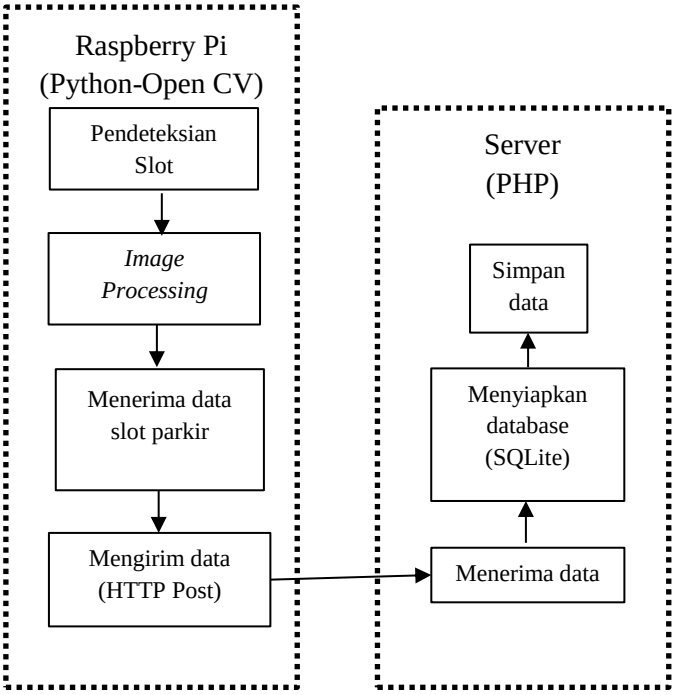
Proses ini digunakan sebagai *reporting* data untuk setiap slot parkir. Secara garis besar proses *reporting* dibagi menjadi dua bagian yaitu simpan data dan proses menampilkan data pada website. Pada gambar blok diagram 3.16 menunjukkan secara garis besar alur hingga ditampilkan pada *website*.



Gambar 3.16 Diagram Blok pada *Website*

3.5.1 Proses Simpan Data

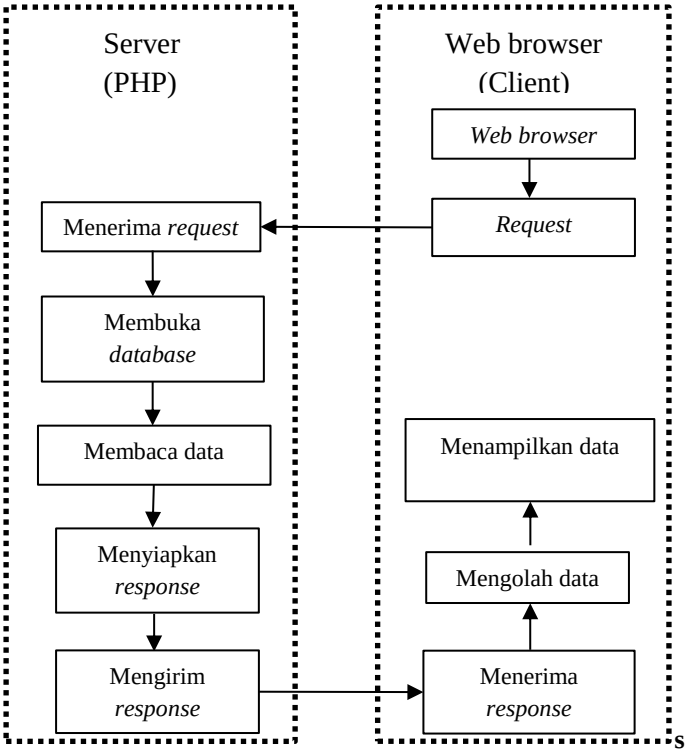
Proses awal pengiriman data dimulai dari raspberry pi sebagai client. Proses yang dilakukan pada raspberry pi yaitu *image* processing. Kemudian setelah diperoleh hasil yang berupa data kosong atau isi, data tersebut dikirim menuju database dengan menggunakan internet. Data yang telah dikirim kemudian diterima di server. Pada server data disimpan di database. Database yang digunakan adalah Sqlite. Pengiriman data dilakukan pada setiap saat iterasi perhitungan selesai. Proses ini dapat dilihat prosesnya pada gambar 3.17



Gambar 3.17 Diagram Blok Simpan Data

3.5.2 Proses Menampilkan Data

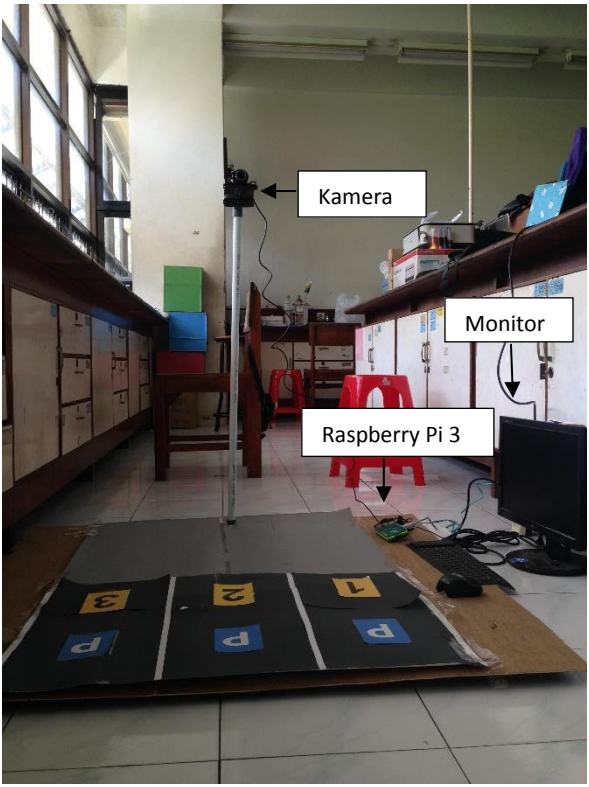
Pada proses pengambilan data web browser sebagai client. Setiap 500ms web browser akan melakukan request data baru ke server untuk kemudian melakukan update tampilan web. Bermula dari web browser meminta data dari server, kemudian permintaan diterima pada server. Setelah diperoleh data yang dibutuhkan, kemudian dikirimkan respon sesuai datanya. Respon akan mengolah berupa warna yang menunjukkan kondisi slot parkir. Setelah diolah lalu ditampilkan di web browser. Proses ini dilakukan berulang kali setiap 500ms pada saat *image processing* dimulai. Proses ini dapat dilihat pada gambar 3.18.



Gambar 3.18 Diagram Blok Pengambilan Data

BAB IV
PENGUJIAN DAN PEMBAHASAN SISTEM

Pada bab ini membahas tentang perancangan sistem yang sudah dirancang, dimana hasil rancangan sistem tersebut akan diuji dan dibahas. Cara pengujian dan pembahasan pada bab ini adalah dengan membahas tiap blok dari perancangan sistem secara keseluruhan dengan disertai tabel dan gambar yang mendukung pengujian dan pembahasan sistem. Gambar 4.1 merupakan sistem parkir yang telah dirancang.



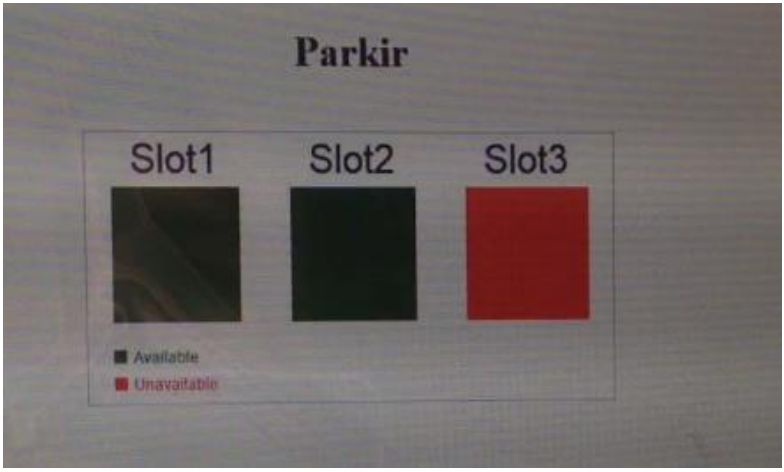
Gambar 4.1 Gambar Keseluruhan Sistem Parkir

4.1 Pengujian Pendeteksian pada Parkiran Dosen

Pengujian berikut dilakukan pada parkiran dosen Departemen Teknik Elektro ITS. Pengujian ini dilakukan menggunakan *laptop*. Pengolahan citra diproses sama seperti pada Raspberry Pi 3 yaitu di OpenCV dengan bahasa python. Kondisi yang digunakan pada pengujian ini yaitu dengan menggunakan satu *pattern*. Pada gambar 4.2 merupakan slot parkir yang akan dideteksi. Berikut hasil dari pengujiannya.

Gambar 4.2 Slot Parkir pada Parkiran Dosen yang akan Diuji

Gambar 4.3 Ketiga Slot Parkiran Dosen Kosong

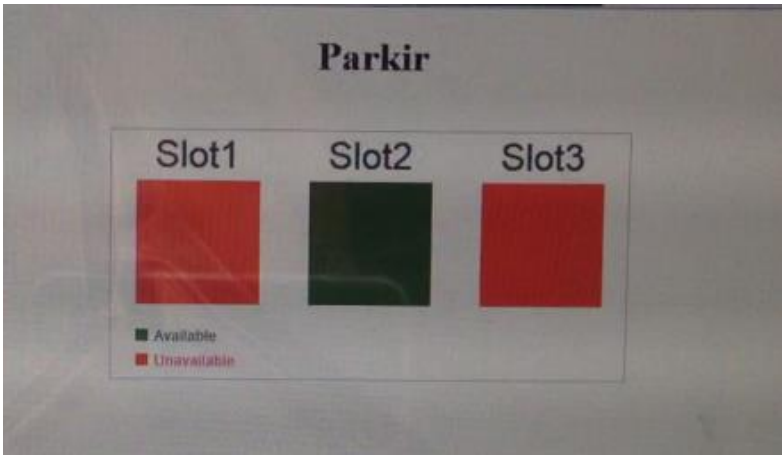


Gambar 4.4 Tampilan pada *Website* Ketika Ketiga Slot Parkiran Dosen Kosong

Pada gambar 4.3 menunjukkan bahwa ketiga slot parkir pada parkiran dosen kosong. Tetapi dapat dilihat bahwa slot ketiga tidak terdeteksi. Hal ini diperkirakan karena *template* yang digunakan terdapat perbedaan warna yang cukup jauh sehingga slot 3 tidak terdeteksi. Perbedaan warna dikarenakan pantulan cahaya, pada slot pertama tidak terlalu tehalang oleh pohon sehingga sangat sesuai dengan *template*. Sedangkan pada slot ketiga lebih banyak pohon sehingga cahaya yang diperoleh lebih sedikit, sehingga tidak terdeteksi.

Dari pendeteksian tersebut, dihasilkan tampilan pada *website* slot 3 berwarna merah. Sebab slot 3 tidak terdeteksi. Kondisi pada *website* dan keadaan sesungguhnya sudah sesuai.

Gambar 4.5 Slot Pertama pada Parkiran Dosen Terdapat Kendaraan

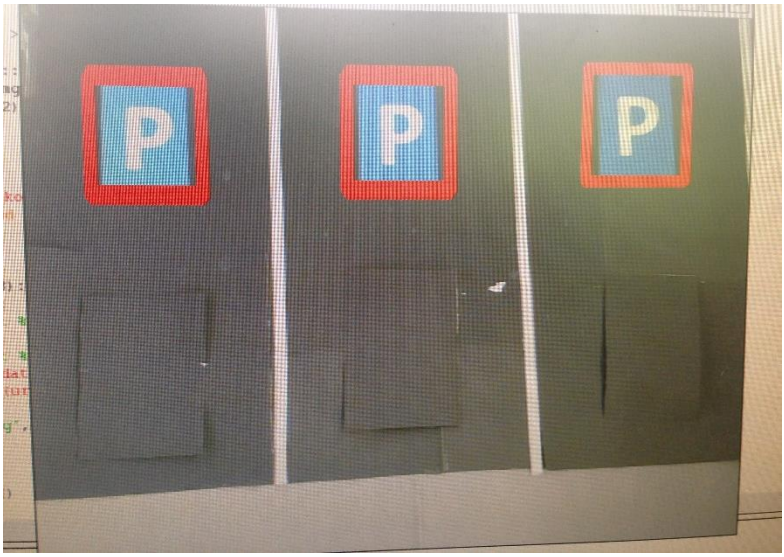


Gambar 4.6 Tampilan pada *Website* Ketika Slot Pertama Terdapat Kendaraan pada Parkiran Dosen

Pada gambar 4.5 merupakan kondisi saat slot pertama terisi oleh mobil. Terdapat dua slot yang kosong yaitu slot kedua dan ketiga, tetapi hanya terdeteksi slot kedua saja. Sehingga pada gambar 4.6 merupakan tampilan pada *website*, slot pertama dan ketiga berwarna merah.

4.2 Pengujian Pendeteksian 3 Slot Parkir dengan Satu *Pattern*

Pengujian berikutnya pendeteksian 3 slot parkir dengan menggunakan metode *template matching* dengan kondisi hanya terdapat *template* huruf. Berikut hasil dari pengujiannya.



Gambar 4.7 Ketiga Slot Kosong

Pada Gambar 4.7, gambar 4.8, gambar 4.9 dan gambar 4.10 menunjukkan beberapa kondisi dari ketiga slot parkir tersebut. Dapat dilihat pada gambar 4.7 tidak terdapat halangan pada slot parkir tersebut, terdapat tanda merah pada ketiga slot tersebut. Hal ini menandakan bahwa *template* yang telah dibuat yaitu berupa *pattern* dengan huruf P terdeteksi kosong. Kemudian pada gambar 4.8 *pattern* 3 tertutup, hasil yang diperoleh slot 1 dan 2 bertanda merah yang artinya kosong. Gambar 4.9 slot 1 dan 3 terdapat kendaraan, hasil yang diperoleh slot 2 terdeteksi kosong. Gambar 4.10 ketiga slot kosong dan tidak terdeteksi *template*. Kesimpulannya jika tidak terdapat tanda merah, maka *template* tidak ditemukan atau terdapat halangan pada slot parkir tersebut. Begitupun sebaliknya. Terdapat tampilan yang menjelaskan bahwa slot tersebut kosong atau isi pada tampilan python. Kemudian untuk satu kali *looping* dibutuhkan waktu 14-16 detik.

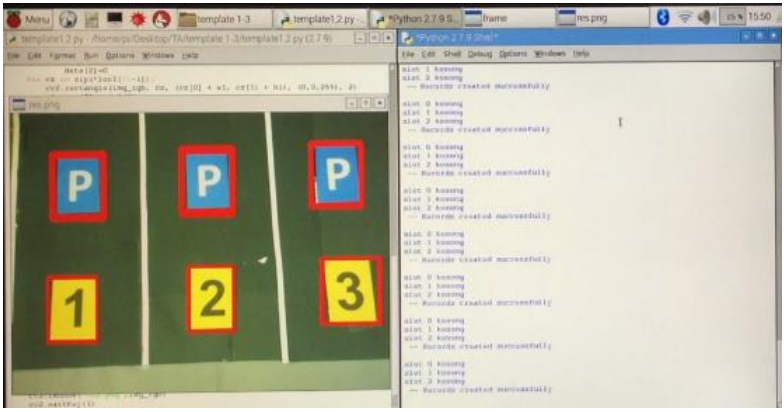
Gambar 4.8 Slot 3 Terdapat Halangan

Gambar 4.9 Slot 2 dan 3 Terdapat Halangan

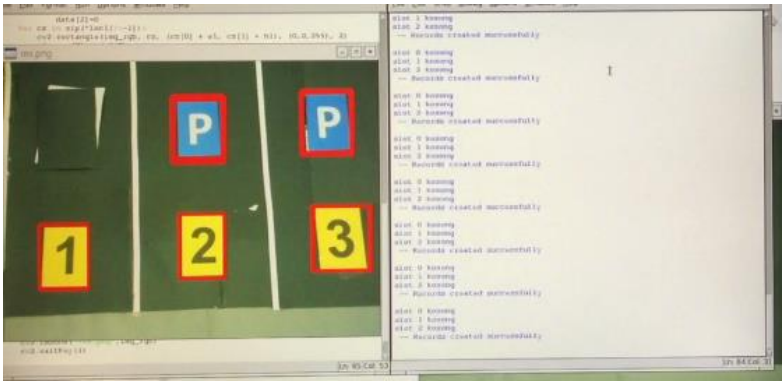
Gambar 4.10 Semua Slot Terdapat Halangan

4.3 Pengujian Pendeteksian 3 Slot Parkir dengan Dua *Pattern*

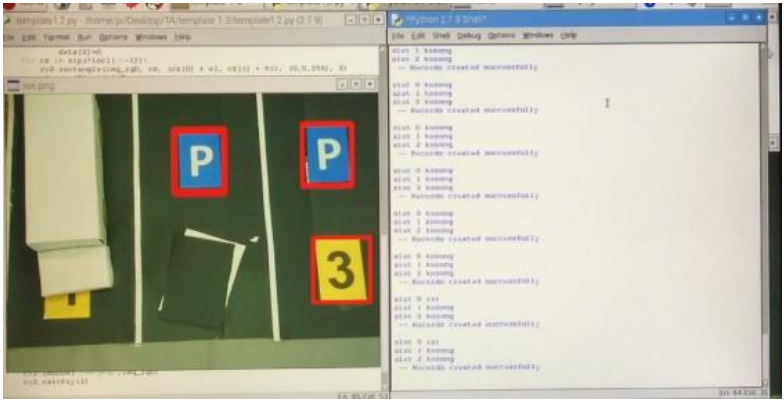
Pengujian berikutnya pendeteksian 3 slot parkir dengan menggunakan metode *template matching* dengan kondisi terdapat dua *pattern*, yaitu angka dan huruf. Berikut hasil dari pengujiannya.



Gambar 4.11 Semua *Pattern* Terdeteksi



Gambar 4.12 *Pattern* P pada Slot 1 Tertutup



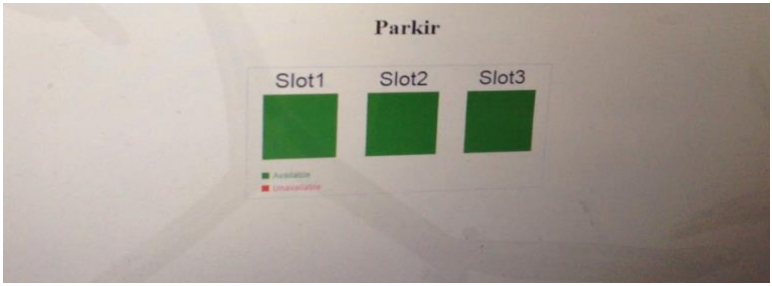
Gambar 4.13 Slot Pertama Tertutup dan *Pattern* 2 Tertutup

Pada Gambar 4.11 merupakan slot kosong yang mendeteksi huruf P dan setiap angkanya, sehingga terdeteksi kosong. Kemudian gambar 4.12 huruf P pada slot 1 tertutup, memperoleh hasil tetap kosong. Gambar 4.13 memiliki kondisi pada slot 1 *pattern* P dan angka tertutup dan juga angka 2 tertutup. Hal ini diperoleh hasil untuk slot 1 terdeteksi isi dan untuk slot 2 terdeteksi kosong. Maka dapat disimpulkan, apabila kondisinya hanya salah satu *pattern* yang tertutup, maka akan terdeteksi kosong. Jika kedua

pattern tertutup maka akan terdeteksi isi. Program ini sengaja dibuat pada saat kedua *pattern* tertutup maka baru akan terdeteksi isi, sebab pendeteksian ini dikhususkan untuk mobil dan telah di design jarak antara huruf dan angka akan tertutupi oleh mobil. Jadi apabila ada kendaraan lain selain mobil maka dia akan dideteksi kosong. Kemudian apabila orang yang berdiri dan menghalangi salah satu *pattern*, kondisinya tidak membutuhkan waktu lebih lama dibandingkan pada saat mobil parkir.

4.4 Pengujian Pada Website dengan Satu *Pattern*

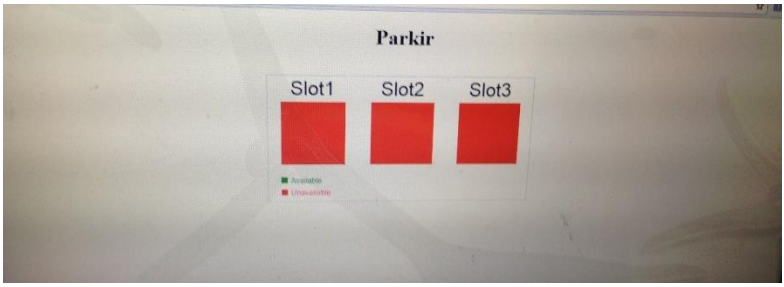
Pangujian selanjutnya yaitu keakuratan tampilan pada website dengan satu *pattern*. Data yang terdapat pada database diperoleh dari hasil pendeteksian. Berikut hasil pengujian pada *website*.



Gambar 4.14 Tampilan pada Website Ketiga Slot Kosong

Gambar 4.15 Tampilan pada Website Slot Ketiga Terisi

Gambar 4.16 Tampilan pada Website Slot Pertama dan Ketiga Terisi



Gambar 4.17 Tampilan pada Website Ketiga Slot Terisi

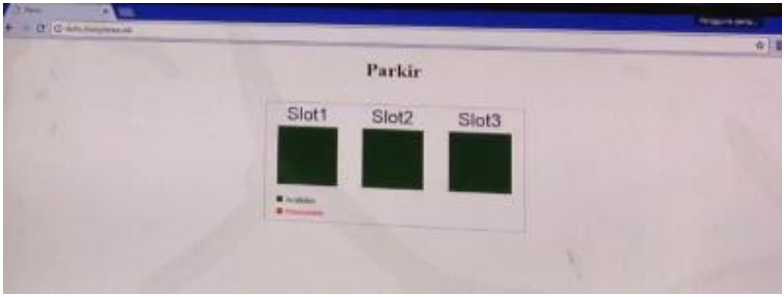
Gambar 4.14 menyesuaikan dengan kondisi pada gambar 4.7 yaitu kondisi pada saat tidak terdapat kendaraan. Kemudian gambar 4.15 mengikuti kondisi pada gambar 4.8. Dimana kondisi pada gambar 4.8 slot ketiga terdapat kendaraan, sehingga pada *website* berwarna merah pada bagian slot 3. Pada gambar 4.16 merupaakan kondisi gambar 4.9. Kondisinya terdapat kendaraan pada slot 1 dan slot 3, sehingga pada *website* berwarna merah pada slot tersebut. Pada gambar 4.17 sesuai dengan kondisi pada gambar 4.10 yaitu semua slot tertutup kendaraan. Sehingga pada *website* semua slot berwarna merah.

Dari data yang diperoleh, kondisi pada *website* sudah sesuai dengan kondisi miniatur. Apabila tidak terdapat kendaraan maka tampilan pada

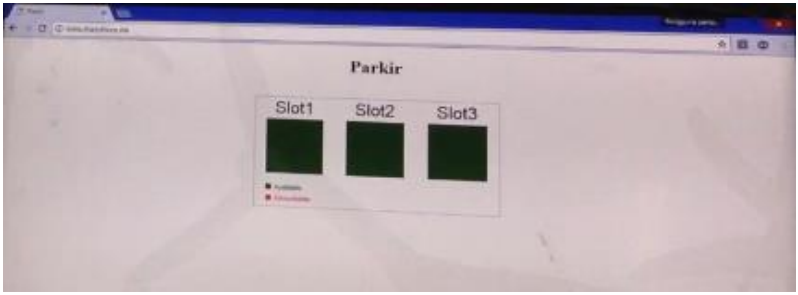
web akan berwarna hijau. Begitu pula sebaliknya, apabila berwarna merah maka terdapat kendaraan.

4.5 Pengujian Pada Website dengan Dua *Pattern*

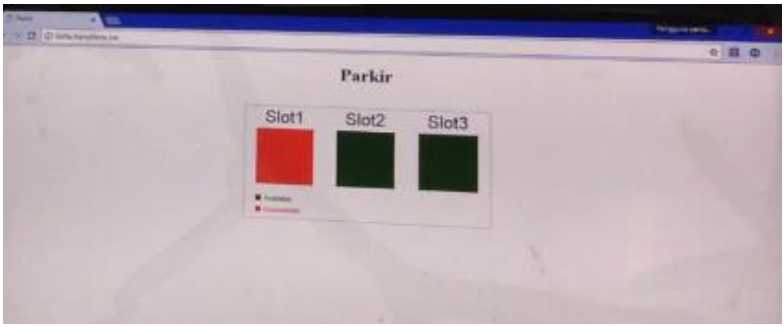
Pangujian berikutnya yaitu keakuratan tampilan pada website dengan menggunakan dua *pattern*. Data yang terdapat pada database diperoleh dari hasil pendeteksian. Berikut hasil pengujian pada *website*.



Gambar 4.18 Tampilan Website Semua Slot Kosong



Gambar 4.19 Tampilan Website *Pattern* P pada Slot 1Tertutup



Gambar 4.20 Tampilan Website Slot Pertama Terdapat Kendaraan dan *Pattern* 2 Tertutup

Pada gambar 4.18 kondisinya sesuai dengan gambar 4.11 tidak terdapat kendaraan. Kemudian gambar 4.19 kondisinya sesuai dengan gambar 4.12. Huruf P pada slot pertama tertutup sehingga tampilan pada *website* tetap berwarna hijau. Pada gambar 4.20 memiliki kondisi sesui dengan gambar 4.13. Slot pertama tertutup kendaraan dan slot kedua *pattern* angka 2 tertutup, sehingga pada *website* hanya slot pertama yang berwarna merah.

Dari hasil yang diperoleh, tampilan pada *website* sudah sesuai dengan kondisi yang ada. Apabila kedua *pattern* tertutup maka terdapat kendaraan, dan tampilan pada *website* akan berwarna merah. Sedangkan jika hanya salah satu *pattern* yang terdeteksi maka *website* akan menampilkan warna hijau, atau tidak terdeteksi kendaraan.

4.6 Pengujian Berdasarkan Pengukuran Intensitas Cahaya

Pangujian selanjutnya yaitu pengukuran berdasarkan intensitas cahaya. Kondisi yang digunakan yaitu dengan dua *pattern*. Terdapat empat kali pengujian dengan nilai intensitas cahaya yang berbeda-beda. Berikut hasil pengujiannya.

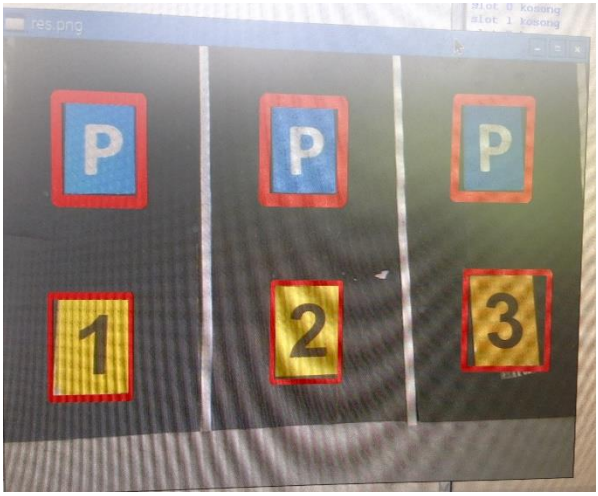
Tabel 4.1 Hasil Pengujian Pengukuran Intensitas Cahaya

Intensitas Cahaya	Tingkat Keberhasilan
500 Lux	Berhasil
279 Lux	Berhasil
158 Lux	Berhasil
7,7 Lux	Berhasil
< 6 lux	Tidak Berhasil

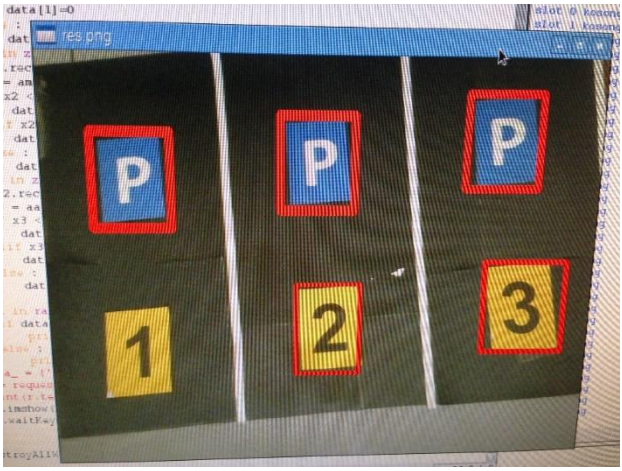
Dari keempat intensitas cahaya tersebut tingkat keberhasilnya ada yang sudah berhasil dan ada pula yang berhasil dengan syarat. Pada intensitas cahaya 500 dan 279 lux memiliki *template* masing-masing, dalam arti setiap kali pengujian memiliki *template* dengan kondisi intensitas yang sama. Pada gamabr 4.22 hasil dari pengujian dengan intensitas 500 lux. Kemudian untuk intensitas 158 lux, pertama-tama menggunakan *template* yang sama pada saat pengujian intensitas 279 lux. Diperoleh hasilnya yaitu slot 3 tidak terdeteksi, hal ini disebabkan oleh perbedaan intensitas cahaya yang dimiliki. Slot 3 tidak mendeteksi karena huruf p lebih redup dibandingkan dengan slot 1 dan 2. Tetapi setelah diganti *template*-nya dengan kondisi yang sama, maka semua slot parkir terdeteksi.

Kemudian sama halnya untuk pengukuran menggunakan intensitas 7,7 lux. Pengujian awal menggunakan *template* dengan intensitas cahaya 158 lux. Hasil yang diperoleh dapat dilihat pada gambar 4.23. Slot 1 pada *pattern* tidak terdeteksi. Hal ini disebabkan pada slot 1 kecocokan antara *template* dengan gambar tidak ditemukan nilai yang sesuai.

Gambar 4.21 Pengujian dengan 500 lux



Gambar 4.22 Hasil Pengujian dengan 500 lux



Gambar 4.23 Hasil Pengujian dengan 7,7 lux

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Terdapat beberapa kesimpulan yang bisa diambil oleh penulis dari tugas akhir ini. Menggunakan metode *template matching* yang diproses pada Raspberry Pi 3, pendeteksian membutuhkan waktu 14-16 detik untuk setiap kali *looping*.

Kemudian dilakukan pengujian dengan mengukuran intensitas cahaya. Terdapat empat kali pengujian, untuk intensitas cahaya 500 lux dan 279 dapat dideteksi setiap *pattern*. Tetapi untuk intensitas cahaya 158 lux dan 7,7 lux telah berhasil tetapi dengan syarat. Jika melakukan pengujian pada 158 lux tetapi menggunakan *template* 279 maka slot 3 tidak terdeteksi. Jika pengujian 279 lux menggunakan *template* yang sesuai yaitu pada intensitas cahaya 279 lux, maka semua slot akan terdeteksi. Begitupun pada pengujian 7,7 lux, *pattern* angka 1 tidak terdeteksi deteksi apabila menggunakan *template* dengan intensitas 158 lux.

Pada miniatur slot parkir ini menggunakan dua kondisi, yang pertama hanya menggunakan satu *pattern* yaitu hanya huruf P. Kondisi kedua menggunakan dua *pattern*, yaitu huruf P dan angka. Pada satu *pattern*, apabila *pattern* huruf P tertutup maka akan terdeteksi isi. Tetapi untuk dua *pattern* jika salah satu dari *pattern* tertutup maka akan terdeteksi kosong. Kemudian jika kedua *pattern* tertutup maka akan terdeteksi isi.

5.2 Saran

Beberapa saran yang penulis bisa berikan untuk pengembangan tugas akhir adalah:

1. Gunakan beberapa kamera untuk mendeteksi banyak *slot* parkir.
2. Jika ingin mencoba metode lain selain *template matching*untuk mendeteksi, dapat digunakan *backgroun substraction*, *pattern recognition*, dsb.
3. Gunakan metode yang sangat tahan terhadap cahaya agar tingkat akurasi sesuai.

4. Jika membutuhkan waktu yang cepat untuk melakukan *image processing*, gunakan mikrokontroler minipc atau menggunakan laptop.

DAFTAR PUSTAKA

[1] Lee, S. dan Seung-Woo, S., "Available Parking Slot Recognition Based on Slot Context Analysis", IET Journals, Vol 10, No.9, pp. 594, 2016

[2] Baroffio, L. Luca, B. Matteo, C. Alessandro, E, R. dan Marco, T., "A Visual Sensor Network for Parking Lot Occupancy Detection in Smart Cities", IEEE, 2015

[3] Utama, A, M., "Rancang Bangun Sistem Pengukur Posisi Terget dengan Kamera Stereo untuk Pengarah Senjata Otomatis", Tugas Akhir, 2016.

[4] Hermawati, F, A., "Pengolahan Citra Digital Konsep & Teori", Penerbit Andi, Yogyakarta, Bab 2, 2013.

[5] Kumar, T. dan Verma, K., "A Theory Based on Conversion of RGB Image to Gray Image", International Journal of Computer Applications, vol. 7 - no. 2, pp. 7-10, September, 2010.

[6] Kumar, T. dan Verma, K., "A Theory Based on Conversion of RGB Image to Gray Image", International Journal of Computer Applications, vol. 7 - no. 2, pp. 7-10, September, 2010.

[7] Wardhana, A, W. dan Prayudi, Y., "Penggunaan Metode Template Matching untuk Identifikasi Kecacatan pada PCB", SNATI, 2008

[8] Bradski, G. dan Kaehler, A., "Learning OpenCV Computer Vision with the OpenCV Library", O'Reilly, Sebastopol, Ch.1, 2008.

[9] Anderson, D. dan Mark, H., "Query Construction Pattern in PHP", IEEE, 2017

Halaman ini sengaja dikosongkan

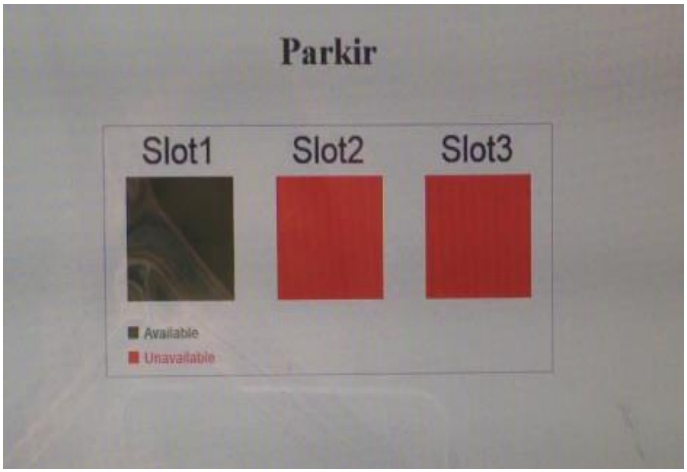
LAMPIRAN

Gambar keseluruhan sistem parkir

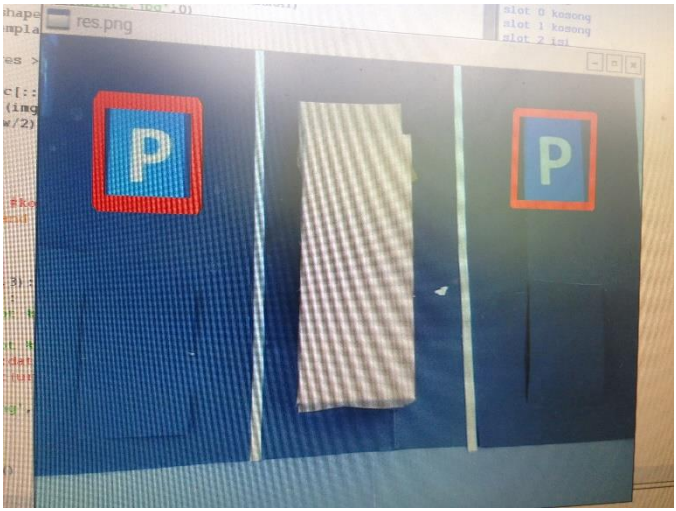

3
2
1
P
P
P

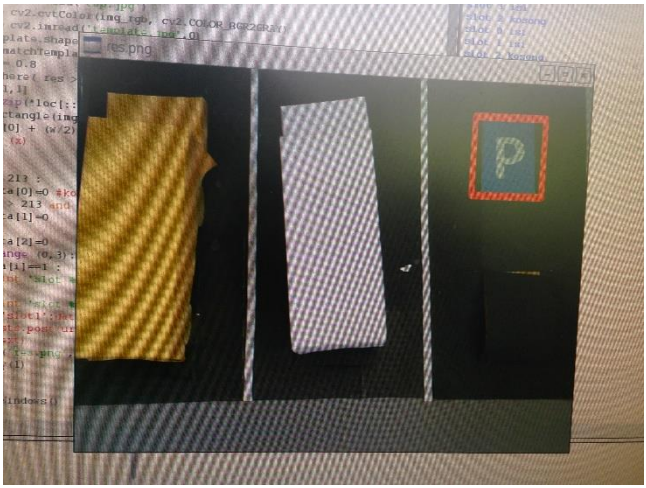

1
P
2
P
3
P

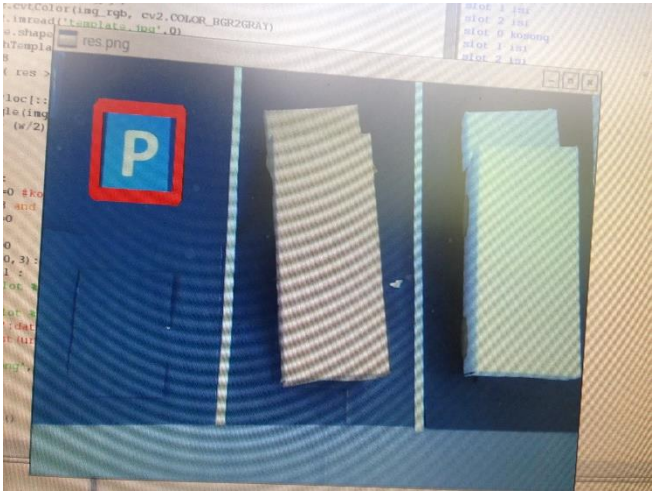
Kondisi pada parkiran dosen



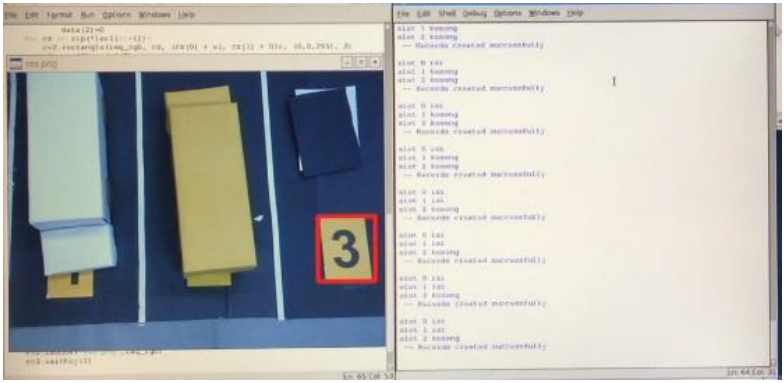
Kondisi dengan Satu *Pattern*



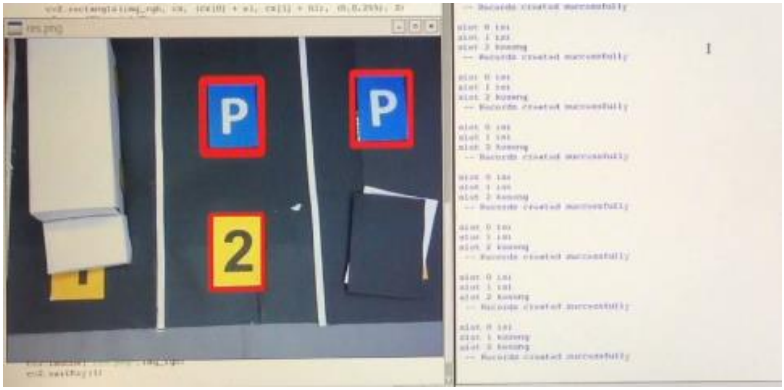

res.png


res.png

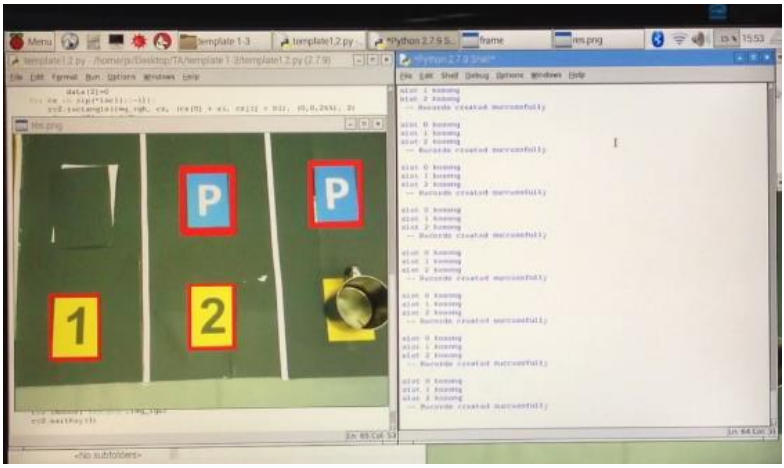
Kondisi dengan dua *pattern s*lot pertama dan kedua tertutup dan *pattern* 3 tertutup



Kondisi dengan dua *pattern* slot pertama tertutup dan *pattern* 3 tertutup



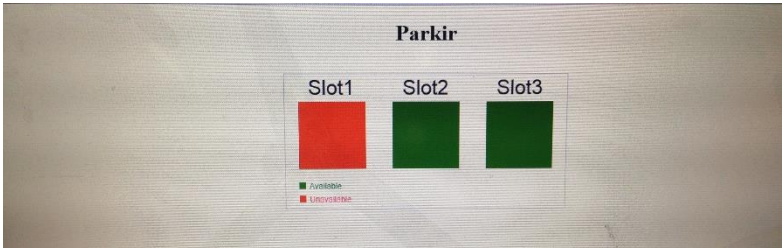
Kondisi dengan dua *pattern*, *pattern* P pada slot 1 dan *pattern* 3 tertutup



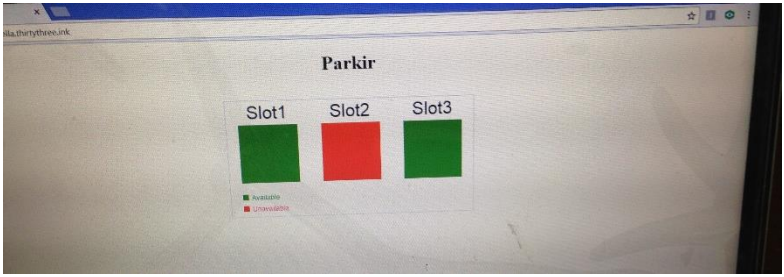
Kondisi dengan dua *pattern*, *pattern* P pada slot 1 dan 2 tertutup dan *pattern* 3 tertutup

Tampilan pada *website* dengan satu *pattern,* Slot Kedua dan Ketiga Terisi

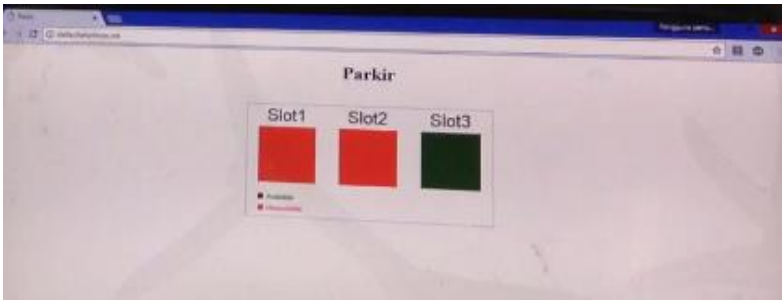
Tampilan pada *website* dengan satu *pattern* Slot Pertama Terisi



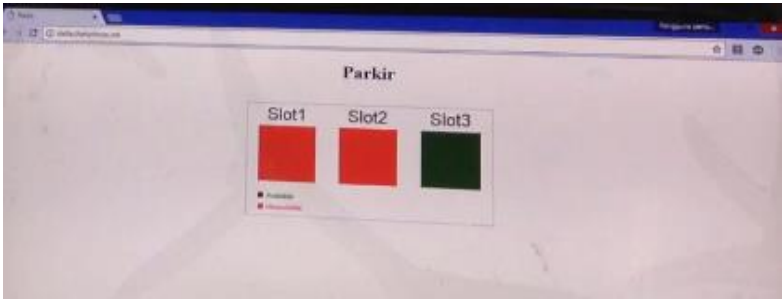
Tampilan pada *website* dengan satu *pattern* slot kedua terisi



Tampilan pada *website* dengan satu *pattern,* Slot pertama dan kedua terisi



Tampilan pada *website* dengan dua *pattern,* slot pertama dan kedua terdapat kendaraan dan *pattern* P pada slot ketiga tertutup



Tampilan pada *website* dengan dua *pattern* , slot pertama terdapat kendaraan dan *pattern* 3 pada slot ketiga tertutup

Tampilan pada *website* dengan dua *pattern, pattern* P pada slot pertama tertutup dan *pattern* 3 pada slot ketiga tertutup

Tampilan pada *website* dengan dua *pattern, pattern* P pada slot pertama dan kedua tertutup dan *pattern* 3 pada slot ketiga tertutup

Program *Template Matching* satu *pattern*

```
while(True):
    k = cv2.waitKey(10)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break

    ret, frame = cap.read()
    cv2.imshow('frame',frame)
    cv2.imwrite('cap.jpg', frame)
    cv2.waitKey(1500)
    #cv2.destroyAllWindows()


    img_rgb = cv2.imread('cap.jpg')
    img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
    template = cv2.imread('template.jpg',0)
    w, h = template.shape[::-1]
    res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)
    threshold = 0.8
    loc = np.where( res >= threshold)
    data = [1,1,1]
    for pt in zip(*loc[::-1]):
        cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
        x = pt[0] + (w/2)
        #print (x)


        if x < 213 :
            data[0]=0 #kosong
        elif x > 213 and x < 427 :
            data[1]=0
        else :
            data[2]=0
    for i in range (0,3):
        if data[i]==1 :
            print 'slot %d isi'%(i)
        else :
            print 'slot %d kosong'%(i)
```

Program *Template Matching* dua *pattern*

```
while(True):
    k = cv2.waitKey(10)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break

    ret, frame = cap.read()
    cv2.imshow('frame',frame)
    cv2.imwrite('cap.jpg', frame)
    cv2.waitKey(1500)
    #cv2.destroyAllWindows()


    img_rgb = cv2.imread('cap.jpg')
    img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
    template = cv2.imread('template.jpg',0)
    template1 = cv2.imread('template1.jpg',0)
    template2 = cv2.imread('template2.jpg',0)
    template3 = cv2.imread('template3.jpg',0)
    w, h = template.shape[::-1]
    w1, h1 = template1.shape[::-1]
    w2, h2 = template2.shape[::-1]
    w3, h3 = template3.shape[::-1]
    res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)
    res1 = cv2.matchTemplate(img_gray,template1,cv2.TM_CCOEFF_NORMED
    res2 = cv2.matchTemplate(img_gray,template2,cv2.TM_CCOEFF_NORMED
    res3 = cv2.matchTemplate(img_gray,template3,cv2.TM_CCOEFF_NORMED
    threshold = 0.8
    loc = np.where( res >= threshold)
    loc1 = np.where( res1 >= threshold)
    loc2 = np.where( res2 >= threshold)
    loc3 = np.where( res3 >= threshold)
    data = [1,1,1]
```

```
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
    x = pt[0] + (w/2)
    if x < 213 :
        data[0]=0 #kosong
    elif x > 213 and x < 426 :
        data[1]=0
    else :
        data[2]=0
for cz in zip(*loc1[::-1]):
    cv2.rectangle(img_rgb, cz, (cz[0] + w1, cz[1] + h1), (0,0,255), 2)
    x1 = cz[0] + (w1/2)
    if x1 < 213 :
        data[0]=0 #kosong
    elif x1 > 213 and x < 426 :
        data[1]=0
    else :
        data[2]=0
for am in zip(*loc2[::-1]):
    cv2.rectangle(img_rgb, am, (am[0] + w2, am[1] + h2), (0,0,255), 2)
    x2 = am[0] + (w2/2)
    if x2 < 213 :
        data[0]=0 #kosong
    elif x2 > 213 and x2 < 426 :
        data[1]=0
    else :
        data[2]=0
for aa in zip(*loc3[::-1]):
    cv2.rectangle(img_rgb, aa, (aa[0] + w3, aa[1] + h3), (0,0,255), 2)
    x3 = aa[0] + (w3/2)
    if x3 < 213 :
        data[0]=0 #kosong
    elif x3 > 213 and x3 < 426 :
        data[1]=0
    else :
        data[2]=0
```

Program pada PHP untuk mengambil data

```php
<?php
  class MyDB extends SQLite3{
    function __construct(){
      $this->open('della.db');
    }
  }

  $db = new MyDB();
  if(!$db){
    echo $db->lastErrorMsg();
  }

  $result = $db->query("SELECT * FROM dataupd;");

  while($row = $result->fetchArray(SQLITE3_ASSOC) ){
    $data = array(
      's1' => $row['slot1'],
      's2' => $row['slot2'],
      's3' => $row['slot3'],
      's4' => $row['slot4'],
      's5' => $row['slot5'],
      's6' => $row['slot6']
    );
    header('Content-Type: application/json');
    echo json_encode($data);
  }

?>
```

Program untuk membuat tampilan pada *website*

```
<!DOCTYPE html>
<html>
<head>
  <title>Parkir</title>
  <script src="js/jquery.js"></script>
</head>
<body>
<center>
<h1>Parkir</h1><br>
<canvas id="myCanvas" width="420" height="200" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>
</center>
<script>



var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// ctx.clearRect(0, 0, canvas.width, canvas.height);
ctx.font = "30px Arial";
ctx.fillText("Slot1",35,30);
ctx.fillText("Slot2",175,30);
ctx.fillText("Slot3",315,30);


ctx.beginPath();
ctx.font = "12px Arial";
ctx.fillStyle = "green";
ctx.fillText("Available",35,170);
ctx.rect(20, 160, 10, 10);
ctx.fill();
```

```
var data1;
var data2;
var data3;
var Update = function(){
  $.getJSON('http://della.thirtythree.ink/getdata.php', function(data) {
    data1 = data.s1;
    data2 = data.s2;
    data3 = data.s3;
  });
  ctx.beginPath();
  ctx.rect(20, 40, 100, 100);
  ctx.fillStyle = "green";
  ctx.fill();

  ctx.beginPath();
  ctx.rect(160, 40, 100, 100);
  ctx.fillStyle = "green";
  ctx.fill();

  ctx.beginPath();
  ctx.rect(300, 40, 100, 100);
  ctx.fillStyle = "green";
  ctx.fill();
```

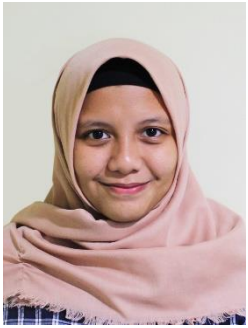
```
  if (data1 == 1) {
    ctx.beginPath();
    ctx.fillStyle = "red";
    ctx.rect(20, 40, 100, 100);
    ctx.fill();
  }
  if (data2 == 1) {
    ctx.beginPath();
    ctx.rect(160, 40, 100, 100);
    ctx.fillStyle = "red";
    ctx.fill();
  }
  if (data3 == 1) {
    ctx.beginPath();
    ctx.rect(300, 40, 100, 100);
    ctx.fillStyle = "red";
    ctx.fill();
  }

  setTimeout (function() { Update(); }, 500);
};


window.onload = function() {
    setTimeout (function() { Update(); }, 500);
};
</script>

</body>
</html>
```

Program untuk menampilkan data pada *website*

```php
<?php
class MyDB extends SQLite3{
  function __construct(){
    $this->open('della.db');
  }
}

if (isset($_POST['slot1']) && isset($_POST['slot2']) && isset($_POST['slot3'])) {
  $db = new MyDB();

  if(!$db){
          echo $db->lastErrorMsg();
    } else {
      // echo "Opened database successfully\n";
    }

  $slot1 = $_POST['slot1'];
  $slot2 = $_POST['slot2'];
  $slot3 = $_POST['slot3'];

  $ret = $db->exec("UPDATE DATAUPD SET slot1 = '$slot1', slot2 = '$slot2', slot3 = '$slot3' ");
  if(!$ret){
    echo $db->lastErrorMsg();
    } else {
      echo " -- Records created successfully\n";
    }

}else{
  echo "hai della";
}
?>
```

Halaman ini sengaja dikosongkan

BIODATA PENULIS



Della Sabila lahir di Bandung pada 5 Desember 1994, yang merupakan anak kedua dari dua bersaudara dari pasangan Lukito Suwarno dan Bina Lohita Sari. Penulis berasal dari Kota Bogor. Penulis menyelesaikan pendidikan dasar di SD Pertiwi Bogor dan dilanjutkan dengan pendidikan menengah di SMP 1 Bogor, SMA 5 Bogor dan pindah pada tahun kedua ke SMA 1 Bogor. Pada tahun 2013, penulis memulai pendidikan di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah, penulis aktif sebagai anggota AIESEC Surabaya serta membantu disetiap kegiatan AIESEC. Penulis juga aktif sebagai asisten laboratorium Elektronika Dasar.

Email :
della.sabila@rocketmail.com

Halaman ini sengaja dikosongkan