DAFTAR PUSTAKA

Dewiani, Syarif, S., Anshar, M., Areni, I. S., Palantei, E., Panggalo, S., . . . Hanan, M. (2021). Sosialisasi Penggunaan Parkir Cerdas Pada Departemen Teknik Elektro. *Jurnal Tepat (Teknologi Terapan Untuk Pengabdian Masyarakat)*, 94-95. doi: 10.25042/jurnal_tepat.v4i1.164.

Bramasta, M. N., Anshar, M., & Nurtanio, I. (2022). Prototipe Sistem Pengenalan Pelat Kendaraan Otomatis Berbasis *YOLO* pada Mekanisme Pintu Masuk Departemen Elektro UNHAS. *Seminar Nasional Elektroteknik dan Teknologi Informasi* (pp. 129 - 135). Makassar: Universitas Hasanuddin.

Indravadanbhai, C., Atul, & Dharmendra. (2012). Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study, Volume 55– No.10. *International Journal of Computer Applications (0975 – 8887)* (pp. 50-51). Charotar University of Science and Technology(CHARUSAT). DOI:10.5120/8794-2784.

Imaduddin, H., Anwar, M. K., Perdana, I., Sulistijono, I. A., & Risnumawan, A. (2018). Indonesian Vehicle License *Plate* Number Detection using Deep Learning Convolutional Neural Network. *International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)* (pp. 1-2). DOI:10.1109/KCIC.2018.8628488.

Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2013). Automatic License *Plate* Recognition (ALPR): A State-of-the-Art Review. *IEEE Transactions on Circuits and Systems for Video Technology 23(2)*, 311-325. DOI:10.1109/TCSVT.2012.2203741.

O. Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

B. Yan, P. Fan, X. Lei, Z. Liu, and F. Yang, “A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved *YOLO*v5,” Remote Sens., vol. 13, no. 9, 2021, doi: 10.3390/rs13091619.

A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "*YOLO*v4: Optimal Speed and Accuracy of Object Detection," ArXiv, vol. abs/2004.10934, 2020.

J. Shashirangana, H. Padmasiri, D. Meedeniya, and C. Perera, "Automated License *Plate* Recognition: A Survey on Methods and Techniques," IEEE Access, vol. 9, pp. 11203–11225, 2021, doi: 10.1109/ACCESS.2020.3047929.

dog-qiuqiu, dog-qiuqiu/*YOLO*-FastestV2: V0.2. Zenodo, 2021. doi: 10.5281/zenodo.5181503.

N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 116–131.

qiuqiuqiu, "*YOLO*-FastestV2: Gèng kuài, gèng qīng, yídòng duān kě dá 300FPS, cānshù liàng jǐn 250k [*YOLO*-FastestV2: Faster, lighter, up to 300FPS on mobile, only 250k parameters]." https://zhuanlan.zhihu.com/p/400474142

S, Akhil. (2016). *An overview of Tesseract OCR Engine.* National Institute of Technology, Calicut Department of Computer Science and Engineering.

zdenop. (2023). *Tesseract OCR*. Retrieved from Github: https://github.com/tesseract-ocr/tesseract

OpenCV.org. (2023). *OpenCV*. Retrieved from https://opencv.org/

BARDI. (2023).*BARDI Smarthome Indonesia*. Retrieved from https://bardi.co.id/

Saputra, I. P. (2023). COMPARISON OF PERFORMANCE AND RELIABILITY OF SURVEILLANCE SYSTEMS: CCTV VS IP CAMERA. *BULLETIN OF NETWORK ENGINEER AND INFORMATICS Vol. 1 No 2* (pp. 71-72). https://doi.org/10.59688/bufnets.

Posey, B. (2023). *Real Time Streaming Protocol (RTSP)*. Retrieved from TechTarget: https://www.techtarget.com/searchvirtualdesktop/definition/Real-Time-Streaming-Protocol-RTSP

M. Maksimović, V. Vujović, V. Milošević, and B. Perišić, “Raspberry Pi as Internet of Things Hardware: Performances and Constraints,” Int. Conf. Electr. Electron. Comput. Eng. IcETRAN, 2014.

“DATASHEET – Raspberry Pi 4 Model B.” raspberrypi.org, 2019. [Online]. Available: https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-datasheet.pdf.

E. Upton, “8GB Raspberry Pi 4 on sale now at $75,” Raspberry Pi Blog, 2020.” https://www.raspberrypi.org/blog/8gb- raspberry-pi-4-on-sale-now-at75/

Čakić, S., Popović, T., Šandi, S., Krčo, S., & Gazivoda, A. (2020). The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing. *24th International Conference on Information Technology (IT)* (pp. 18-22). DOI: 10.1109/IT48810.2020.9070558.

A. Al-Mahturi and R. Rahim, “Ultrasonic Sensor for Distance Measurement,” Prog. Process Tomogr. Instrum. Syst., vol. 24, pp. 9–14, 2016.

S. Dswilan, Harmadi, and Marzuki, “Flood monitoring system using ultrasonic sensor SN-SR04T and SIM 900A,” J. Phys. Conf. Ser., vol. 1876, no. 1, 2021, doi: 10.1088/1742- 6596/1876/1/012003.

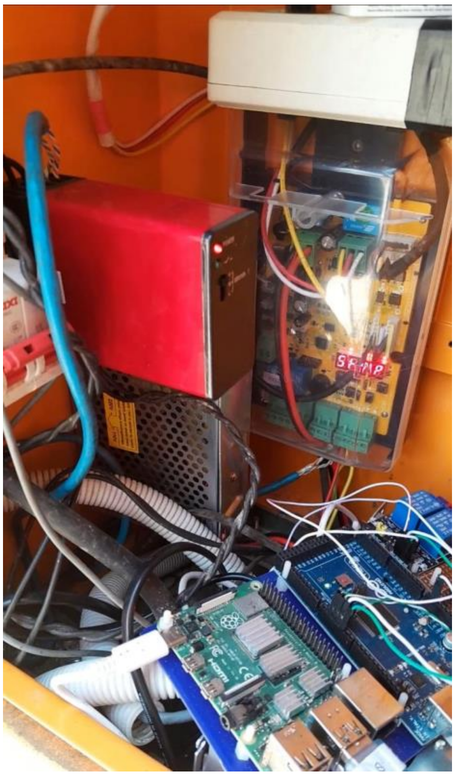
“Barrier Gate Manual.” OneTech. [Online]. Available: https://qdigital.mx/content/Barreras/Barreras tradicional/Manual de barrera tradicional- tarjeta 2.0.pdf

Kho, D. (2023). *Pengertian Relay dan Fungsinya*. Retrieved from teknikelektronika.com: https://teknikelektronika.com/pengertian-relay-fungsi-relay/

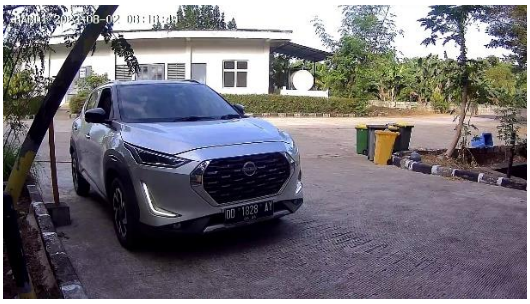
LAMPIRAN

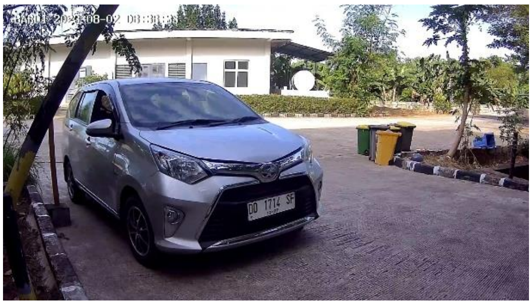
Lampiran 1 Dokumentasi Pelaksanaan Penelitian

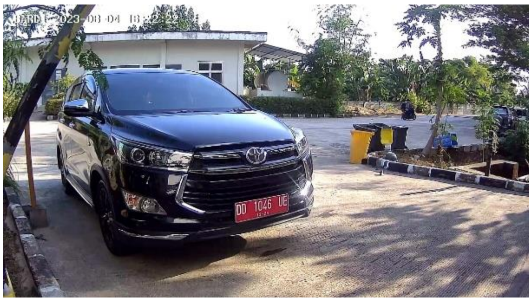
1.1 Tahap perancangan

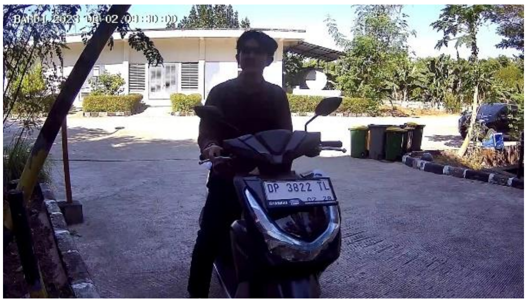


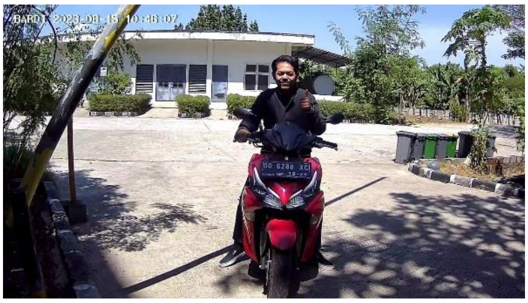
1.2 Tahap pengujian

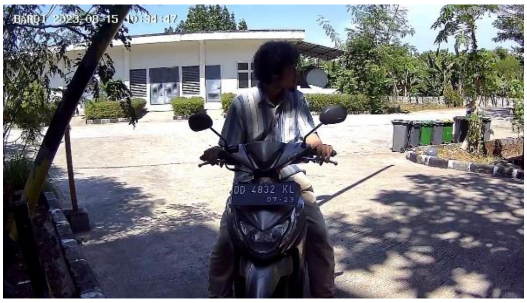


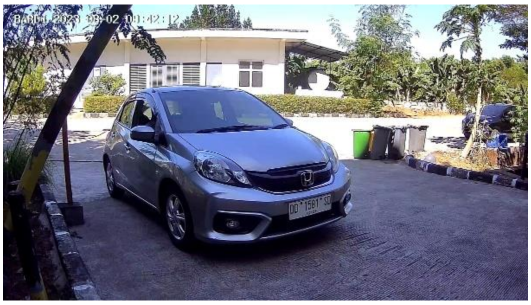


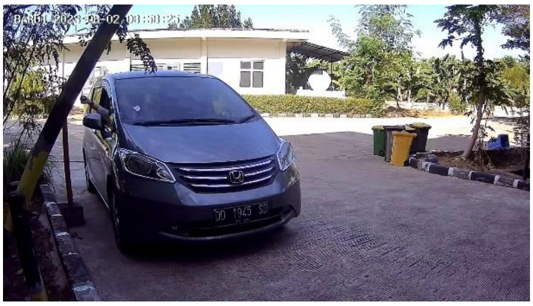


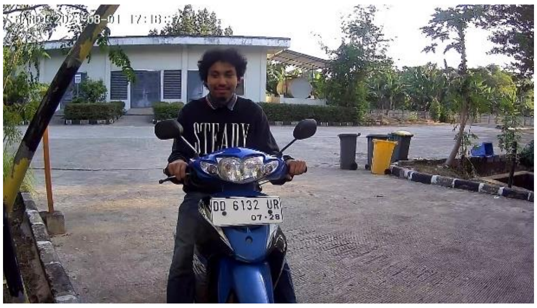


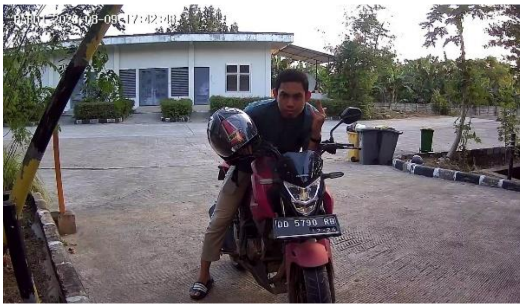












DD 2813 YB

DD 1975 LY

DD 1604 KR

DD 2840 EU

B 1259 UJM

DD 1714 SF

DD 4639 OE

DD 4096 OT

DD 3198 LQ

Lampiran 2 Kode Pemrograman

2.1 Program utama untuk menjalankan prototipe

```
//g++ -o smartgate smartgate.cpp -I/usr/include/tesseract -L/usr/local/lib -llept -ltesseract -lpigpio -ljsoncpp
`pkg-config --libs opencv4` -std=c++11 -I/home/arya/json/include -I/usr/local/include/opencv4

#include <iostream>

#include <fstream>

#include <string>

#include <pigpio.h>

#include <unistd.h>

#include <nlohmann/json.hpp>

#include <chrono>

#include <opencv2/opencv.hpp>

#include <tesseract/baseapi.h>

#include <leptonica/allheaders.h>

using json = nlohmann::json;

const int RELAY_UP = 23;

const int RELAY_DOWN = 24;

const int BUZZER = 18;

void open_gate() {

    std::cout << "Gate is now OPEN." << std::endl;

    gpioWrite(RELAY_UP, 0);

    usleep(2000000);

    gpioWrite(RELAY_UP, 1);

}

void close_gate() {

    std::cout << "Gate is now CLOSED." << std::endl;

    gpioWrite(RELAY_DOWN, 0);

    usleep(2000000);

    gpioWrite(RELAY_DOWN, 1);

}

double distance(int GPIO_TRIGGER, int GPIO_ECHO) {

    gpioSetMode(GPIO_TRIGGER, PI_OUTPUT);

    gpioSetMode(GPIO_ECHO, PI_INPUT);

    gpioWrite(GPIO_TRIGGER, 1);
    usleep(10);
    gpioWrite(GPIO_TRIGGER, 0);
    double start_time = 0.0;
    double stop_time = 0.0;
    while (gpioRead(GPIO_ECHO) == 0) {
        start_time = time_time();
    }
    while (gpioRead(GPIO_ECHO) == 1) {
        stop_time = time_time();
    }
    double time_elapsed = stop_time - start_time;
    double distance = (time_elapsed * 34300) / 2;
    return distance;
}
int main() {
    std::cout << "SmartGate starting..." << std::endl;
    const std::string path = "/home/arya/smartgate/detected_chars.txt";
    std::string plate;
    if (gpioInitialise() < 0) {
        std::cerr << "Failed to initialize pigpio library." << std::endl;
        return 1;
    }
    try {
        while (true) {
            float dist_entrance = distance(5, 6); // distance for entrance vehicle sensor
            std::cout << "Measured Distance = " << dist_entrance << " cm" << std::endl;
            time_sleep(1);
            if (dist_entrance < 50) {
                cv::VideoCapture
camera("rtsp://admin:arya2023@192.168.213.148:8554/Streaming/Channels/101");
                cv::Mat frame;
                camera >> frame;
                cv::imwrite("./tmp/cam.jpg", frame);
                auto start = std::chrono::high_resolution_clock::now();
```

```
//system("bash analyse_photo.sh > /dev/null 2>&1");
// Initialize Tesseract OCR
tesseract::TessBaseAPI tesseract;
tesseract.Init(NULL, "eng"); // Specify language data file here
// Load image
Pix *image = pixRead("./tmp/cam.jpg");
if (!image) {
   std::cerr << "Failed to load image." << std::endl;
   return 1;
}
// Convert image to grayscale
Pix *grayscaleImage = pixConvertTo8(image, 0);
if (!grayscaleImage) {
   std::cerr << "Failed to convert image to grayscale." << std::endl;
   pixDestroy(&image);
   return 1;
}
// Set grayscale image for OCR
tesseract.SetImage(grayscaleImage);
// Perform OCR
char *text = tesseract.GetUTF8Text();
if (!text) {
   std::cerr << "Failed to recognize text from image." << std::endl;
   pixDestroy(&grayscaleImage);
   pixDestroy(&image);
   return 1;
}
// Write the recognized text to a file
std::ofstream outputFile("./detected_chars.txt");
if (!outputFile.is_open()) {
   std::cerr << "Failed to open output file." << std::endl;
   delete[] text;
   pixDestroy(&grayscaleImage);
```

```
    pixDestroy(&image);
    return 1;
}
outputFile << text << std::endl;
outputFile.close();
// Clean up
delete[] text;
pixDestroy(&grayscaleImage);
pixDestroy(&image);
tesseract.End();
std::ifstream inputFile("./detected_chars.txt");
std::ifstream file("./tools/database.json");
json database;
file >> database;
const auto& plates = database["plates"];
std::string desiredText;
    std::string desiredTexts;
std::string line;
bool foundDesiredText = false;
for (const auto& plate : plates) {
    desiredText = plate["licence"];
    desiredTexts = plate["owner"];
    inputFile.clear();
    inputFile.seekg(0, std::ios::beg);
    while (std::getline(inputFile, line)) {
        if (line.find(desiredText) != std::string::npos) {
        foundDesiredText = true;
            break;
        }
    }
    if (foundDesiredText) {
        break;
    }
```

```
            }
            inputFile.close();
            file.close();
            if (foundDesiredText) {
                std::cout << "Plate " << desiredTexts << " accepted" << std::endl;
                open_gate();
                auto end = std::chrono::high_resolution_clock::now();
                std::chrono::duration<double> duration = end - start;
                std::cout << "Execution time: " << duration.count() << " seconds" << std::endl;
            } else {
                std::cout << "Plate rejected" << std::endl;
                close_gate();
                auto end = std::chrono::high_resolution_clock::now();
                std::chrono::duration<double> duration = end - start;
                std::cout << "Execution time: " << duration.count() << " seconds" << std::endl;
                continue;
            }
        }
        else {
            continue;
        }
        while (true) {
            float dist_mid_sensor = distance(5, 6);
            time_sleep(1);
            std::cout << "After gate opened: " << dist_mid_sensor << std::endl;
            if (dist_mid_sensor > 200) {
                close_gate();
                break;
            }
        }
    }
}catch (const std::exception& e) {
    std::cerr << "Exception: " << e.what() << std::endl;
    }
    gpioTerminate();
    std::cout << "SmartGate stopped manually." << std::endl;
    return 0;
}
```

2.2 Program untuk mendeteksi pelat kendaraan

```
./clear
#echo "Detecting licence plate..."
#echo ""
./plate_detector
#echo ""
#======
#echo "Cropping detected plate..."
vips im_extract_area ./tmp/cam.jpg ./tmp/cropped.jpg $(cat coords.txt)
```

2.2.1 Program untuk membersihkan hasil pembacaan karakter sementara

```
#include <iostream>
#include <fstream>
int main() {
    std::ofstream ofs("detected_chars.txt", std::ofstream::out | std::ofstream::trunc);
    return 0;
}
```

2.2.2 Program untuk melokalisasi bagian pelat kendaraan

```
g++ -g\
 src/detector.cpp src/yolo-fastestv2.cpp\
 -o plate_detector\
 -I src/headers -I /home/arya/ncnn/build/install/include/ncnn/home/arya/ncnn/build/install/lib/libncnn.a\
 `pkg-config --libs --cflags opencv` -fopenmp -ldl
g++ -g\
 src/ocr.cpp\
 -o read_plate\
 -I /home/arya/ncnn/build/install/include/ncnn /home/arya/ncnn/build/install/lib/libncnn.a\
 `pkg-config --libs --cflags opencv` -fopenmp -ldl
```

2.2.2.1 detector.cpp

```
#include "yolo-fastestv2.h"
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    static const char* class_names[] = {
        "plate"
    };
    yoloFastestv2 api;
    api.loadModel("./models/detect_sim-opt.param",
                  "./models/detect_sim-opt.bin");
    cv::Mat cvImg = cv::imread("./tmp/cam.jpg");
    std::vector<TargetBox> boxes;
    api.detection(cvImg, boxes);
    for (int i = 0; i < boxes.size(); i++) {
        // std::cout<<boxes[i].x1<<" "<<boxes[i].y1<<" "<<boxes[i].x2<<" "<<boxes[i].y2
        //          <<" "<<boxes[i].score<<" "<<boxes[i].cate<<std::endl;
        std::cout<<boxes[i].x1<<" "<<boxes[i].y1<<" "<<boxes[i].x2<<" "<<boxes[i].y2<<std::endl;
        ofstream labelsFile("coords.txt");
        labelsFile<<boxes[i].x1<<" "<<boxes[i].y1<<" "<<boxes[i].x2 - boxes[i].x1<<" "<<boxes[i].y2 -
boxes[i].y1;
        labelsFile.close();
        char text[256];
        sprintf(text, "%s %.1f%%", class_names[boxes[i].cate], boxes[i].score * 100);
        int baseLine = 0;
        cv::Size label_size = cv::getTextSize(text, cv::FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);
        int x = boxes[i].x1;
        int y = boxes[i].y1 - label_size.height - baseLine;
        if (y < 0)
            y = 0;
        if (x + label_size.width > cvImg.cols)
            x = cvImg.cols - label_size.width;
        cv::rectangle(cvImg, cv::Rect(cv::Point(x, y), cv::Size(label_size.width, label_size.height + 
baseLine)),
                      cv::Scalar(255, 255, 255), -1);
        cv::putText(cvImg, text, cv::Point(x, y + label_size.height),
                    cv::FONT_HERSHEY_SIMPLEX, 0.5, cv::Scalar(0, 0, 0));
        cv::rectangle (cvImg, cv::Point(boxes[i].x1, boxes[i].y1),
                       cv::Point(boxes[i].x2, boxes[i].y2), cv::Scalar(255, 255, 0), 2, 2, 0);
    }
    cv::imwrite("output.png", cvImg);
    return 0;
}
```