

Aircraft Docking Guidance System (ADGS) Simulation Design And Learning Media At Medan Aviation Polytechnic

Fauzi Ikhsan Amin^{1*}, Catra Indra Cahyadi², Donna Nurhaida Masdina Sirait³

¹²³ Politeknik Penerbangan Medan, Indonesia

* ikhsanamin515@gmail.com, catraindracahyadi@gmail.com, dehijo@gmail.com

ABSTRACT

The Aircraft Docking Guidance System (ADGS) is an automated guidance system designed to assist pilots in parking aircraft precisely and efficiently at the airport apron. This technology provides visual information regarding the aircraft's position relative to a predefined stopping point, thereby supporting safety and smooth ground operations. Considering the importance of understanding this system in aviation, there is a need for a learning medium that can realistically simulate the operation of ADGS. This study aims to develop an ADGS simulation as a microcontroller-based educational tool at Politeknik Penerbangan Medan. The system is built using an ESP32 microcontroller as the central controller, a LiDAR sensor for accurate aircraft distance detection, a P10 dot matrix display for visual information output, and a servo motor to simulate mechanical elements commonly found in ADGS systems. Through the design and testing stages, the simulation has proven capable of interactively representing the operational principles of ADGS. The results show that the system provides accurate visual and mechanical responses to distance changes, making it suitable as a learning aid for students to better understand modern ground handling systems in the aviation industry.

Keywords: Aircraft Docking Guidance System, ESP32 Microcontroller, LiDAR Sensor, Dot Matrix P10, Learning Tool



This Is Open Access Article Under The CC Attribution- ShareAlike 4.0 License.



INTRODUCTION

An airport is an airport area according to Law No. 1 of 2009 and PM 39 of 2019, which is a land/water area with safety, security, and air transportation support facilities. This definition is in line (Utami et al., 2020) the one that states that an airport is an area for aircraft movement including buildings and supporting installations. To support flight safety, one important facility is the Aircraft Docking Guidance System (ADGS) (Sugiyono, 2018) According to ICAO Doc 9157 Part 4, ADGS provides visual guidance for pilots when parking aircraft on the apron so that they are positioned correctly, efficiently, and safely, especially at airports with high traffic (Grzesik et al., 2016). ADGS must be able to operate in various weather conditions and be integrated with the apron management system .

In Indonesia, the application of ADGS is still limited so that the understanding of technicians is also minimal (Utami & Faizah, 2020). ADGS is crucial for reducing human error during parking, such as aircraft misalignment with the centerline, which can interfere with parking stands and jet bridges. In vocational education, the Airfield Lighting System (AFL) course at the Medan Aviation Polytechnic covers ADGS as part of the airport visual aid material (Undang Undang No. 1 Tahun 2009 Tentang Penerbangan Dan Peraturan Menteri Perhubungan Nomor PM 39 Tahun 2019 Tentang Tatanan Kebandarudaraan Nasional, 2020). However, ADGS learning remains limited to theory due to the lack of simulation tools or practical media.

This hinders students from understanding both the automated system (ADGS) and manual guidance (marshalling). As a solution, a buggy-based ADGS simulation was designed to represent the aircraft. The purpose of designing this tool is to facilitate students' understanding of the ADGS system and to broaden their knowledge of microcontroller utilization. d in the creation of learning media tools (Setiyo et al., 2023). This simulation is expected to be a comprehensive learning medium for AFL practice as well as marshalling training, so that students gain real-world experience in integrating automated and manual parking systems in the aviation world

RESEARCH METHODS

This research was conducted using the Research and Development (R&D) method because the primary goal was not only to generate data but also to design and develop a product in the form of an Aircraft Docking Guidance System (ADGS) simulation. The R&D approach was chosen so that the research process could encompass the conceptual, design, implementation, and evaluation stages of the product's effectiveness. According to (Rumayani et al., 2020), the Research and Development (R&D) method is method designed research For produce something product new (tools, media, learning models, etc.) and measure effectiveness product the through A series of trials and evaluations. The research was conducted by designing and observing existing systems. and record the function of each component (Priambodo, 2021). Method research used is design or design system. Where the design process discussed as something analysis. Design results will tested in a way systematic For get data (Organization, 2022). The ADGS system was developed using a Lidar sensor as an object position detector and a DMD P10 display module as a visual information medium. The design process was carried out through a series of stages starting from identifying needs, installing systems and software, creating designs, to testing prototype performance (Organization, 2024).

RESULTS AND DISCUSSION

Hardware Installation

At this stage, the ADGS tool is designed using the main components: such as the ESP32 microcontroller, TF-Luna LiDAR sensor, LM2596 IC, ULN2003 driver, P10 Dot Matrix LED, servo motor, pilot lamp, and power supply. The hardware installation is carried out on the PCB board by arranging and placing the components according to the design, as shown in Figure 1 below (Khoiruroziq & Nilakresna, 2021a).

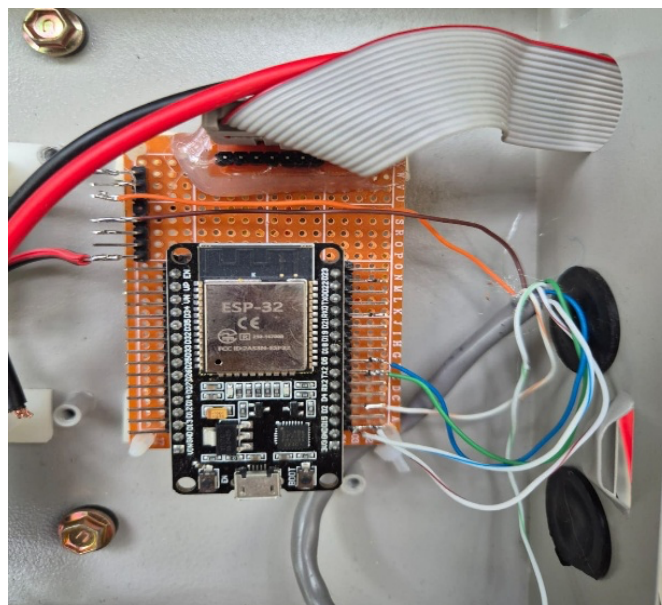


Figure 1Component Arrangement on the PCB Board

At this stage, the main circuit is assembled using a perforated PCB as a connection medium between components. The soldering process is carried out manually using soldering iron, paying attention to the connection path to prevent short circuits. Figure 2 shows the results of the soldering that has been done (Grzesik et al., 2016).

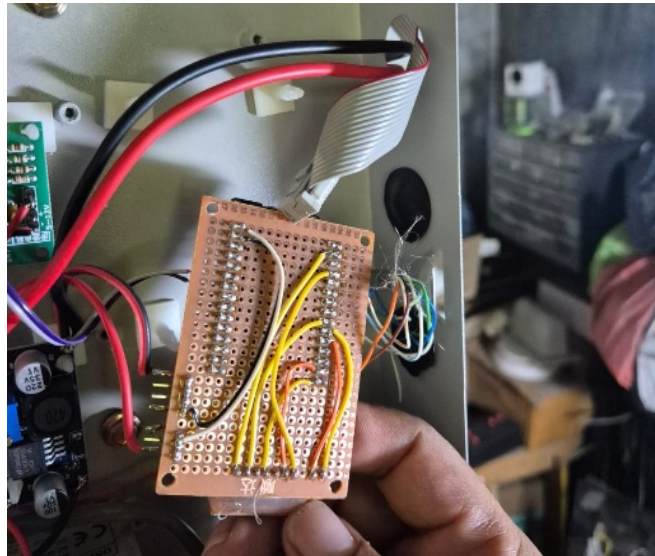


Figure 2Soldering on the PCB Board

The hardware is assembled in a metal box that serves as a protective system. The main components installed include an ESP32 microcontroller, a 12V power supply, a step-down converter module, and a driver module. Each module is mounted using bolts and adhesive, then connected with jumper cables according to the circuit diagram shown in Figure 3 below (Dermawan et al., 2021).

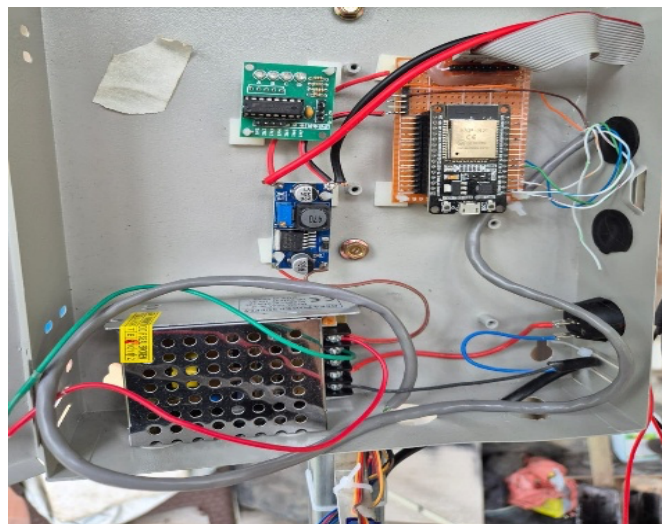


Figure 3. Component Installation in the Panel Box

After that, enter the stage of connecting all the components, connecting the LM2596 IC pin to the ESP32 with the VCC pin configuration on the Arduino to OUT+ on the LM2596 IC as a voltage source for the other components used. The GND pin on the ESP32 is connected to the OUT- pin on the LM2596 IC. The configuration can be seen in Figure 4 as follows (Dermawan & Purnomo, 2021).

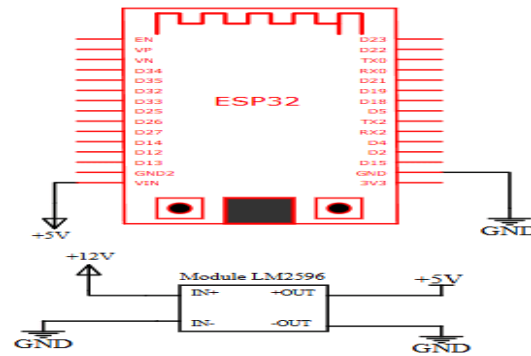


Figure 4 ESP32 Pin Configuration with LM2596 IC

This circuit connects the ESP32 to two P10 Dot Matrix LED modules using the CLK, LAT, OE, and A, B, and C pins to control the display. The first panel receives data from the ESP32 and passes it to the second panel via a daisy chain, with a shared +5V DC supply and ground. These P10 DMD modules are typically used to display running content. (Priambodo, 2021). The P10 displays *real-time* aircraft parking guidance (left, right, straight, STOP), making this configuration practical, efficient, and easy to develop.

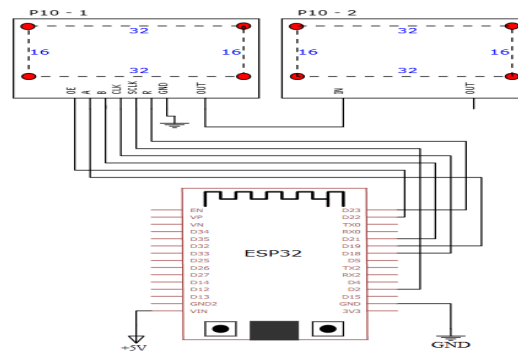


Figure 5 ESP32 Pin Configuration with P10 DMD LED

The motor functions as the arm of the sensor to direct the sensor to the media to be measured, namely 3 observation points, left, right and center. (Dermawan et al., 2021). connecting the servo motor pins to the ESP32 and the LM2596 IC, the VCC pin is connected to the OUT+ pin of the LM2596 for power supply. The GND pin is connected to the OUT- pin of the LM2596 for voltage reference (Cahyadi & Siregar, 2018). The IN pin is connected to the digital pin D15 on the ESP32 to control the direction of motor rotation and control the motor steps.

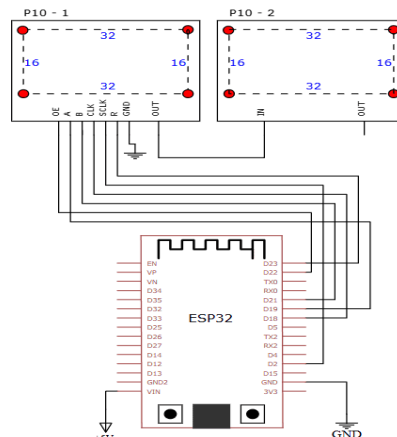
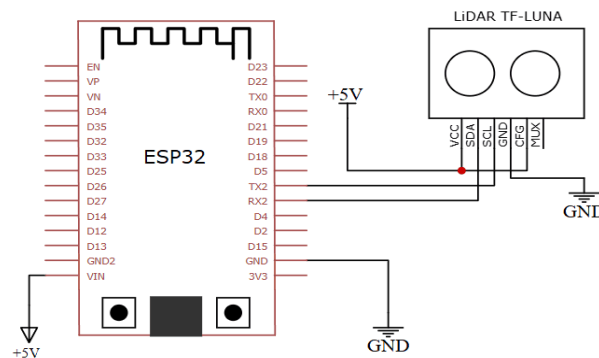
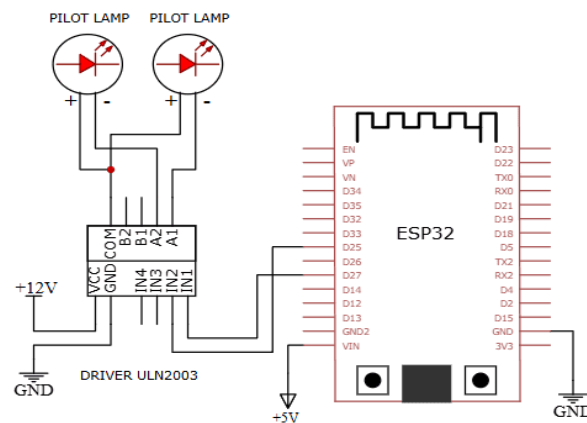


Figure 6 ESP32 Pin Configuration with Servo Motor

Then, connect the Tf-Luna LiDAR sensor pin to the ESP32 and IC LM2596 (Widiasaputra et al., 2022). VCC is connected to the 5V pin of the ESP32 and functions to provide a 5V power supply to the sensor. GND is connected to the GND pin of the ESP32, the RX pin of the Tf-Luna LiDAR sensor is connected to the RX2 pin of the ESP32 and functions to send distance data from the sensor to the ESP32 (Adetayo & Feyisola, 2021). The TX pin of the Tf-Luna LiDAR sensor is connected to the TX2 pin of the ESP32 and functions to receive commands from the ESP32 to the sensor.

**Figure 7 ESP32 configuration with TF-Luna LiDAR Sensor**

driver circuit that uses the ULN2003 IC as a controller for the guide light uses 2 types of colors, namely red and blue. green (Rumayani et al., 2020). The ULN2003 IC functions as an electronic switch that connects the current from the +5V source to the GND via the lamp when it receives an input signal from the microcontroller (Sugiyono, 2020). The IN1 and IN2 pins receive the control signal, while OUT1 and OUT2 are connected to the negative side of the lamp. The positive side of the lamp goes directly to +5V. When the input is active, the transistor in the ULN2003 conducts current to GND, so the lamp lights up. The COM pin is connected to +5V as protection against reverse voltage.

**Figure 8 ULN2003 Driver Configuration with Pilot Lamp**

After all the hardware is assembled and tested, the next step is to design and assemble the entire system so that the device can work in an integrated manner. This circuit combines all the main components, such as the ESP32 as the system's brain, the DMD display as the visual display, the sensors as detectors, and the servo motors and pilot lamps as supporting indicators (Maradin, 2021).



Figure 9 and Construction of ADGS Simulation Tool

Software Installation

The software used to program and run the circuit is Arduino. software (*Integrated Development Environment*) IDEA (Khoiruroziq & Nilakresna, 2021) . The software installation stage for programming the ESP32 uses the Arduino IDE software to compile it. and upload the program or sketch into the ESP32 microcontroller (IEA, 2023). The first step is to install the required *libraries according to the components used, including the TFmini Library, Servo Library, DMD32 Library, and SystemFont 5x7 Library*. After all *libraries* are installed, the program or instructions are compiled in the Arduino IDE according to the desired logic to control the modules in the system (Sitanggang et al., 2022). Next, the upload process is carried out by connecting the laptop to the ESP32 microcontroller using a USB cable, so that the instructions that have been created can be embedded and executed by the ESP32 sequentially according to the program flow.

```

VDGS_V05.ino
1  // Includes the Servo library
2  #include <ESP32Servo.h>.
3  #include <DMD32.h>
4  #include "fonts/SystemFont5x7.h"
5  #include "fonts/Arial_black_16.h"
6  #define DISPLAYS_ACROSS 2
7  #define DISPLAYS_DOWN 1
8
9  DMD dmd(DISPLAYS_ACROSS, DISPLAYS_DOWN);
10
11  hw_timer_t * timer = NULL;
12
13  void IRAM_ATTR triggerScan()
14  {
15      dmd.scanDisplayBySPI();
16  }

```

Figure 10 Program View in Arduino IDE

This Arduino program functions to manage the obstacle detection and avoidance system using a servo motor and a distance sensor. The main system parameters include stepMod 15° per step, speedRun 25, and servo movement from 60° to 120°. The sensor is able to detect objects up to a distance of 800 cm, with the system automatically stopping if the object is within 200 cm. The pins used are pin 25 for the red LED, pin 27 for the green LED, and pin 15 for the servo motor (Sulistiyo, 2020).

```

// SETTING =====
int stepMod = 15; // step sudut derajat
int speedRun = 20; //30 kecepatan putar servo
int angle_min = 60; // sudut min
int angle_max = 120; // sudut max
int max_distance = 800; // jarak max
int stop_distance = 200; // jarak stop
// =====

// init PIN -----
int pinPilodRed = 25;
int pinPilodGreen = 27;
int pinServo = 15;
//-----

// P10
bool fl;
int urut = 0;
int cnt, jarak;
long tmrRun, tmrJarak, tmrShow;
int posY = 9;
int animCnt;
int angle;
int mid_distance, left_distance, right_distance, show_distance;
int idx;
int nilKiri, nilKanan, nilTengah;
String pesan, arah;
long tmrSpd;
bool stopFirst;

```

Figure 11. Main Parameters of the System

At the start of the program, serial communication is enabled via `Serial.begin(9600)` and `Serial2.begin(115200)` for monitoring and communication. The LED pins are configured as outputs and are disabled to prevent them from lighting up during startup. The servo motors are controlled using the *ESP32PWM Library* with a frequency of 50 Hz, connected to specific pins with minimum and maximum pulses. A timer interrupt is configured to automatically run the `triggerScan` function every 300 ms. Before the main process begins, the DMD screen is cleared and the system is given a short delay to ensure all components are ready.

```

void setup() {
  Serial.begin(9600);
  Serial2.begin(115200);
  pinMode(pinPilodRed, OUTPUT);
  pinMode(pinPilodGreen, OUTPUT);
  digitalWrite(pinPilodRed, LOW);
  digitalWrite(pinPilodGreen, LOW);

  ESP32PWM::allocateTimer(0);
  myServo.setPeriodHertz(50);
  myServo.attach(pinServo, 500, 2400); // Defines on which pin is the servo motor attached
  //P10
  uint8_t cpuClock = ESP.getCpuFreqMHz();

  timer = timerBegin(0, cpuClock, true);
  timerAttachInterrupt(timer, &triggerScan, true);
  timerAlarmWrite(timer, 300, true);

  // Start an alarm
  timerAlarmEnable(timer);

  //clear/init the DMD pixels held in RAM
  dmd.clearScreen( true ); //true is normal (all pixels off), false is negative (all pixels on)
  delay(100);
}

```

Figure 12. System Initialization and Void Setup Configuration

Display () function displays the proximity sensor readings on the DMD screen with a check every 50 ms using `millis()`. If the distance = -1, the display displays “STOP” indicating an obstacle, while if the distance = -2, the symbol “---” is displayed indicating the sensor is not read. For normal distances, data is converted to text and displayed on the screen (Rashid, 2021). The indicator LEDs and display are updated according to the distance conditions to provide *real-time visual information*.

```
void Display(){ // =====
    bool ret = false;
    if(millis() > tmrShow + 50){
        tmrShow = millis();
        if(show_distance == -1){
            if(stopFirst == false) urut = 0;
            if(!stp){
                stp = true;
                dmd.clearScreen( true );
            }
            dmd.drawString(20, 0, "    ", 5, GRAPHICS_NORMAL );
            dmd.drawString(20, 0, "STOP", 4, GRAPHICS_NORMAL );
            RedOnOff(true);
            GreenOnOff(false);
            stopFirst = true;
            ret = true;
        }else if(show_distance == -2){
            dmd.drawString(25, 0, "---", 3, GRAPHICS_NORMAL );
            RedOnOff(false);
            GreenOnOff(false);
            stopFirst = false;
            stp = false;
        }
    }
}
```

Figure 13 Information Display Process on a Dot Matrix Display

This code displays an animation of an arrow moving to the right on a dot matrix. In *case 10*, the `cnt` variable is initialized and the screen is cleared before moving to *case 12*. In *case 12*, `cnt` is incremented, displaying the arrow “>>>>>” moving from left to right using `dmd.drawString()`, with a 30 ms delay to smooth the animation (Abidi et al., 2022). Once `cnt` reaches 35, the arrow display is replaced with a brighter one (`GRAPHICS_NOR`) and the system returns to *case 10*, so the arrow animation runs continuously.

```
case 10: // lampu ke kanan -----
    cnt = 0;
    urut = 12;
    dmd.clearScreen( true );
    break;

case 12:
    cnt++;
    dmd.drawString(-32 + cnt, posY, ">>>>>", 5, GRAPHICS_NORMAL );
    delay(30);
    if(cnt >= 35) {
        urut = 10;
        dmd.drawString(-32 + cnt, posY, ">>>>>", 5, GRAPHICS_NOR );
    }
    break;
```

Figure 14. Displaying Distance and Status on P10

This code displays an animation of the arrow “<<<<<” moving left on a dot matrix. In *case 20*, the `cnt` variable is *reset* to 0 and the screen is cleared before moving to *case 22*. In *case 22*, `cnt` is incremented each time, displaying the arrow moving from right to left using `dmd.drawString()` with a decreasing position,

smoothed with a 30 ms *delay*. When cnt reaches 33, the arrow is displayed with a different style (GRAPHICS_NOR) and the system returns to case 20, so the animation runs continuously (Nugroho & Alrasyid, 2020).

```

case 20: // lampu ke kiri -----
    cnt = 0;
    urut = 22;
    dmd.clearScreen( true );
    break;

case 22:
    cnt++;
    dmd.drawString(64 - cnt, posY, "<<<<<", 5, GRAPHICS_NORMAL );
    delay(30);
    if(cnt >= 33) {
        dmd.drawString(64 - cnt, posY, "<<<<<", 5, GRAPHICS_NOR );
        urut = 20;
    }
    break;

```

Figure 15. Displaying Distance and Status on P10

Tool Test Results

The device was tested by measuring the distance read by the sensor and comparing it to the actual distance using a meter. The object used was a *buggy*, which was positioned parallel to the sensor and could be adjusted for height. The car was moved slowly from 1 meter to 8 meters from the sensor in the morning to observe the sensor's response to moving objects under natural lighting conditions (Kallawa et al., 2022). The test focused on comparing the sensor readings with the actual distance, then calculating the difference as a *percentage error* to determine the sensor's error rate (Intang et al., 2020).

$$\% \text{ Error} = \left(\frac{X_s - X_p}{X_s} \right) \times 100\%$$

Information:

X_p = Distance result on the tool (m).

X_s = Actual distance result (m),

% Error = Percentage of error that exists between the meter and sensor measurements (%).

The TF-Luna LiDAR sensor was tested by collecting data at distances of 80 cm to 825 cm. The results showed that the sensor was able to read distances with high accuracy without error at most points (Hasmia et al., 2022). At distances of 652 cm and 790 cm, small deviations of 3 cm and 4 cm were recorded, respectively, resulting in errors of 0.460% and 0.506%. At a distance of 825 cm, the sensor did not detect an object, indicating the maximum range limit. The following is a comparison of the distance results from the device with the actual distance results:

Table 1. Proximity Sensor Test Results

Test	Actual distance in meters (cm)	Distance on Lidar sensor TF Luna(cm)	Error %
1	80	80	0
2	168	168	0
3	225	225	0
4	362	362	0
5	405	405	0
6	530	530	0
7	576	576	0

8	652	655	0.460
9	790	794	0.506
10	825	825	No detected

Next, testing using a *buggy* as a substitute for an airplane was carried out at observation angles of L1 60° (left), 90° (*centerline*), and R1 120° (right), with a distance of 0–800 cm from the sensor. The test results are shown in the following table:

Table 2. ADGS Trial Results with *Buggy* as Object

Test	Point Observation			Display On Tool	Guidance Information
	L1 (60°)	Center (90°)	R1 (120°)		
1	0	1	0		Centerline
2	1	0	0		To Right
3	0	0	1		To the left
4	1	1	1		Parking Stop
5	0	0	0		Pass limit detection

Information :

- 1 : logic 1 represents the sensor detecting the presence of an object at a corner point within range.
 0 : logic 0 represents the sensor does not detect the presence of an object at a corner point within range
 (Habibullah et al., 2020).
 >>|<<< : The result indicates that the object is in the center/ *centerline*.
 <<<<< : The result indicates that the *buggy object* is further to the right.
 >>>>> : The result indicates that the *buggy object* is further to the left
 STOP : The result indicates that when the *buggy object* is < 200 cm from the sensor

CONCLUSION

The Aircraft Docking Guidance System (ADGS) Simulation Design test show that the system is designed according to the ADGS working principles at the airport, starting from initial detection, position correction to the centerline, to the instruction to stop at the parking stop. The system uses a TF-Luna Lidar sensor, ESP32 microcontroller, Dot Matrix P10, servo motor, pilot lamp, ULN2003 driver, and an integrated LM2596 module. A buggy car is used as a representation of an aircraft, where the TF-Luna sensor detects the distance and position of the object, then the data is processed by the ESP32 to display direction and distance instructions on the Dot Matrix P10. Tests show that the sensor is able to read distances with high accuracy up to ± 8 meters, although at 8.25 meters the object is not detected. The entire system is controlled via the Arduino IDE to process data, display visual instructions, and automatically regulate actuators from initial detection to parking stop

BIBLIOGRAPHY

- Abidi, Z., Ilmi, U., & Ashari, F. B. (2022). Desain Speed Bump Penyimpan Energi Berbasis Sensor Piezoelektrik. *Jurnal Jeetech*, 3(2), 84–89. <https://ejournal.ftundar.ac.id/index.php/jeetech/article/view/200>
- Adetayo, O. A., & Feyisola, O. A. (2021). Acceptance Of Visual Docking Guidance System By Ground Marshallers In Nigerias' Airport. *International Journal Of Advanced Networking And Applications*, 13(01), 4845–4854. <https://doi.org/10.35444/ijana.2021.13107>
- Cahyadi, C. I., & Siregar, M. F. (2018). Implementation Of Automatic Car Cleaning System With Microcontroller System Atmega 8. *International Journal Of Engineering And Technology (Uae)*, 7(2.13 Special Issue 13), 418–420. <https://doi.org/10.33258/Birci.V4i3.2548>
- Dermawan, D., & Purnomo, M. J. (2021). Perancangan Tampilan Visual Docking Guidance System (Vdgs) Pada Sistem Parkir Pesawat Terbang. *Angkasa: Jurnal Ilmiah Bidang Teknologi*, 7(2), 181. <https://doi.org/10.28989/Angkasa.V7i2.160>
- Dermawan, D., Setiawan, P., Basukesti, A., & Muhammad, R. N. (2021). Rancang Bangun Visual Docking Guidance System (Vdgs) Sebagai Pendeteksi Arah Gerak Longitudinal Pesawat Pada Sistem Parkir Pesawat Terbang. *Avitec*, 3(2), 167–179. <https://doi.org/10.28989/Avitec.V3i2.910>
- Grzesik, N., Kuźma, K., & Żurawski, P. (2016). Aircraft Docking Guidance System To The Gate Using Fuzzy Logic. *Journal Of Kones Powertrain And Transport*, 25(3). <https://doi.org/10.5604/01.3001.0012.4381>
- Habibullah, M., Mulyanto, Y., & Sofya, N. D. (2020). Rancang Bangun Aplikasi Pemandu Wisata Museum

- Sumbawa Berbasis Android Dengan Memanfaatkan Qr Code. *Jurnal Informatika, Teknologi Dan Sains*, 2(2), 136–145. <https://doi.org/10.51401/jinteks.V2i2.596>
- Hasmia, N., Nirsal, & Jumardi, A. (2022). Rancang Bangun Aplikasi Inventaris Pada Kantor Desa Salulemo Kecamatan Baebunta Kabupaten Luwu Utara. *D'computare: Jurnal Ilmiah Teknologi Informasi Dan Ilmu Komputer*, 12(1). <https://doi.org/10.30605/Dcomputare.V12i1.40>
- Iea. (2023). *Electricity Market Report – January 2023*. International Energy Agency. <https://www.iea.org/reports/electricity-market-report-january-2023>
- Intang, A., Muhaimin, M., & Darmawan, R. (2020). Fenomena Kerugian Tekanan Dalam Saluran Udara Bertekanan Di Atas Satu Atmosfer. *Teknika: Jurnal Teknik*, 7(1), 1–7. <https://doi.org/10.35449/teknika.V7i1.125>
- Kallawa, A. F., Fikri, A., & Mujirudin, M. (2022). Pengaruh Rangkaian Seri Dan Paralel Terhadap Tegangan Pada Piezoelektrik. 52–57.
- Khoiruroziq, F., & Nilakresna, S. (2021a). Perancangan Prototype Adgs Menggunakan Sensor Jarak Berbasis Arduino Di Bandar Udara Djalaluddin Gorontalo. *Airman: Jurnal Teknik Dan Keselamatan Transportasi*, 4, 51–55.
- Khoiruroziq, F., & Nilakresna, S. (2021b). Perancangan Prototype Adgs Menggunakan Sensor Jarak Berbasis Arduino Di Bandar Udara Djalaluddin Gorontalo Design Of Adgs Prototype Using Arduino-Based Distance Sensor At Gorontalo Djalaluddin Airport. 4, 51–55.
- Maradin, D. (2021). Advantages And Disadvantages Of Renewable Energy Sources Utilization. *International Journal Of Energy Economics And Policy*, 11(3), 176–183. <https://doi.org/10.32479/ijee.11027>
- Nugroho, M. A. A., & Alrasyid, M. Z. (2020). Analisa Perbandingan Performansi Akurasi Mesin Cnc Router Berbasis Mach3 Dan Arduino Uno Menggunakan Metode Sqc. *Jurnal Teknik Elektro Dan Komputasi*, 2(2), 75–86. <https://doi.org/10.32528/elkom.V2i2.3464>
- Organization, I. C. A. (2022). *Annex 14 To The Convention On International Civil Aviation, Volume I – Aerodromes: Design And Operations (9th Ed.)*. Icao.
- Organization, I. C. A. (2024). *Aerodrome Design Manual, Part 4: Visual Aids*. Icao.
- Priambodo, A. D. (2021). Rancangan Prototype Visual Docking Guidance System (Vdgs) Berbasis Arduino Uno Dan Rf Lora Sebagai Alat Bant Pembelajaran. <http://repo.poltekbangsby.ac.id/id/eprint/943>
- Rashid, M. H. (2021). *Power Electronics: Circuits, Devices And Applications (4 (Ed.))*. Pearson Education.
- Rumayani, N. A., Warsito, T., & Pambudiyatno, N. (2020). Rancang Bangun Prototype Visual Docking Guidance System (Vdgs) Menggunakan Lampu Pandu Berbasis Raspberry Pi Sebagai Media Pembelajaran Di Politeknik Penerbangan Surabaya.
- Setiyo, C. I. C., Saputra, W., Sudjoko, R. I., & Faizah, F. (2023). Prototype Pengaturan Sistem Kontrol Otomasi Fuel Treatment Tangki Fuel Harian Dari Tangki Bulanan Genset Pada Power Station Bandar Udara. *Jurnal Kumbaran Fisika*, 6(1).
- Sitanggang, R., Dachi, T. U., & Manurung, I. H. G. (2022). Rancang Bangun Sistem Penjualan Tanaman Hias Berbasis Web Menggunakan Php Dan Mysql. *Jurnal Teknologi Kesehatan Dan Ilmu Sosial (Tekesnos)*, 4(1), 84–90. <https://e-journal.sari-mutiara.ac.id/index.php/tekesnos/article/view/2908>
- Sugiyono. (2018). *Metode Penelitian Kuantitatif, Kualitatif Dan R&D (20th Ed.)*. Bandung: Penerbit Alfabeta.
- Sugiyono. (2020). *Metodologi Penelitian Kuantitatif, Kualitatif Dan R & D*.
- Sulistiyono, D. (2020). *Dasar-Dasar Konversi Energi Listrik*. Andi.
- Undang Undang No. 1 Tahun 2009 Tentang Penerbangan Dan Peraturan Menteri Perhubungan Nomor Pm 39 Tahun 2019 Tentang Tata N Kbandarudaraan Nasional. (2020).

Utami, K. P., Faizah, F., Politeknik, S., & Surabaya, P. (2020). Prototipe Aircraft Docking Guidance System (Adgs) Menggunakan Sensor Infrared Dan Ultrasonic Berbasis Internet Of Things. *Prosiding Snitp (Seminar Nasional Inovasi Teknologi Penerbangan)*, 4.