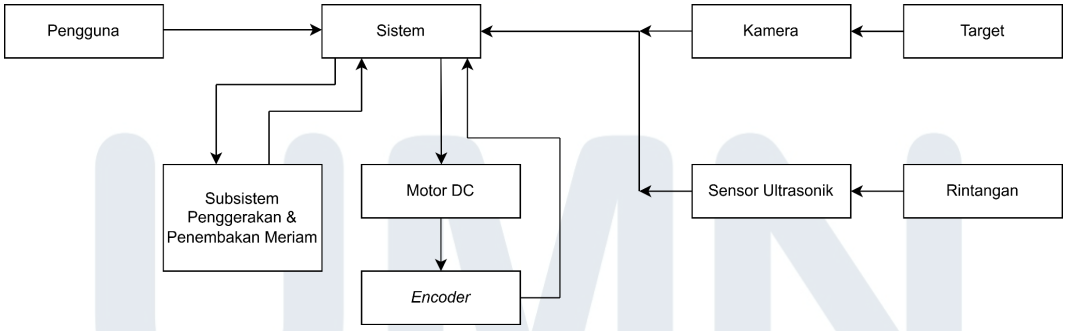
BAB III
PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Tinjauan Desain Sistem

Keseluruhan sistem *mobile robot* terdiri dari dua subsistem untuk bekerja secara otonom. Pada subsistem pengenalan dan deteksi objek, digunakan kamera DSLR dan Raspberry Pi. Raspberry Pi dalam subsistem ini berfungsi untuk melakukan pengolahan citra menggunakan algoritma YOLO dengan data foto yang ditangkap oleh kamera.

3.1.1 Desain Sistem Keseluruhan

Secara umum, sistem menerima input berupa gambar digital dari kamera, jarak rintangan dari sensor ultrasonik, jarak tempuh *mobile robot* dari *encoder*, dan konfigurasi yang dilakukan oleh pengguna. Input diproses oleh mikrokontroler dengan algoritma masing-masing subsistem. Hasil pengolahan input tersebut menghasilkan output berupa pergerakan meriam dan pergerakan motor DC, sehingga *mobile robot* dapat beroperasi sesuai fungsinya. *Data Flow Diagram* (DFD) level 0 dari subsistem ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 DFD level 0 *mobile robot* dengan sistem penembakan

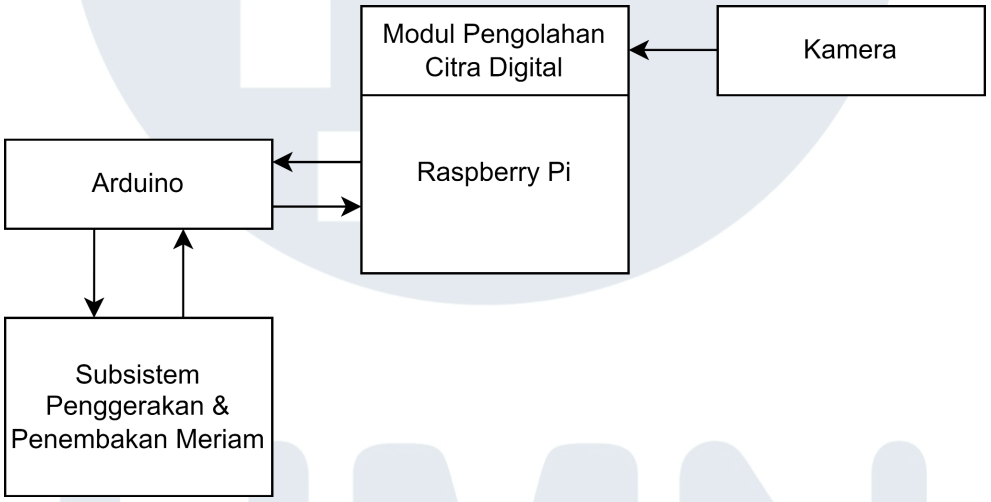
Tabel 3.1 Penjelasan DFD level 0 *mobile robot* dengan sistem penembakan

Parameter	Keterangan
Input	● Konfigurasi sistem dari pengguna. ● Gambar digital dari tangkapan kamera. ● Jarak dari sensor ultrasonik. ● *Feedback* berupa rotasi *encoder*.
Output	● Aktuator subsistem penggerakan dan penembakan meriam. ● Pergerakan motor DC.

Parameter	Keterangan
Fungsi	● Melakukan pengenalan dan deteksi target. ● Aktuator penggerakan dan penembakan meriam menembak jenis target yang sudah ditentukan. ● Mendapatkan jarak antara rintangan dan *mobile robot*. ● *Mobile robot* bergerak menyesuaikan lintasan dan rintangan di hadapannya.

3.1.2 Desain Subsistem

Pengolahan data subsistem pengenalan dan deteksi objek dilakukan oleh Raspberry Pi. Hasil pengolahan citra dikirim ke subsistem penembakan meriam yang dikendalikan oleh Arduino. Kedua mikrokontroler berkomunikasi secara serial melalui koneksi USB. DFD level 1 dari subsistem ini dapat dilihat pada Gambar 3.2.



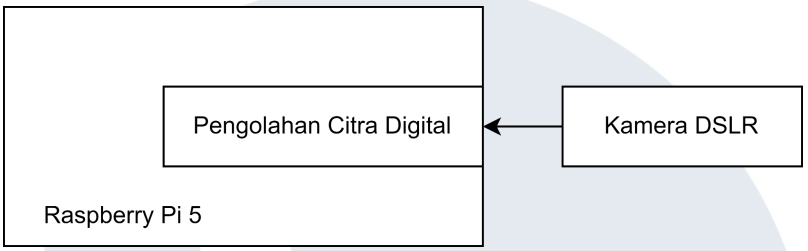
Gambar 3.2 DFD level 1 subsistem pengenalan dan deteksi objek

Tabel 3.2 Penjelasan DFD level 1 subsistem pengenalan dan deteksi objek

Parameter	Keterangan
Input	● Gambar digital dari tangkapan kamera.
Output	● Aktuator subsistem penggerakan dan penembakan meriam.
Fungsi	● Melakukan pengenalan dan deteksi jenis target. ● Aktuator penggerakan dan penembakan meriam menembak jenis target yang sudah ditentukan.

Subsistem ini berperan untuk memperoleh posisi dan jarak dari meriam ke target serta jenis target. Input foto diperoleh melalui kamera DSLR. Raspberry Pi digunakan untuk pengolahan citra. Proses pembedaan target dilakukan menggunakan algoritma YOLO, dan proses akuisisi jarak

menggunakan *bounding box* dan algoritma *monocular vision*. DFD level 2 dari subsistem ini dapat dilihat pada Gambar 3.3.



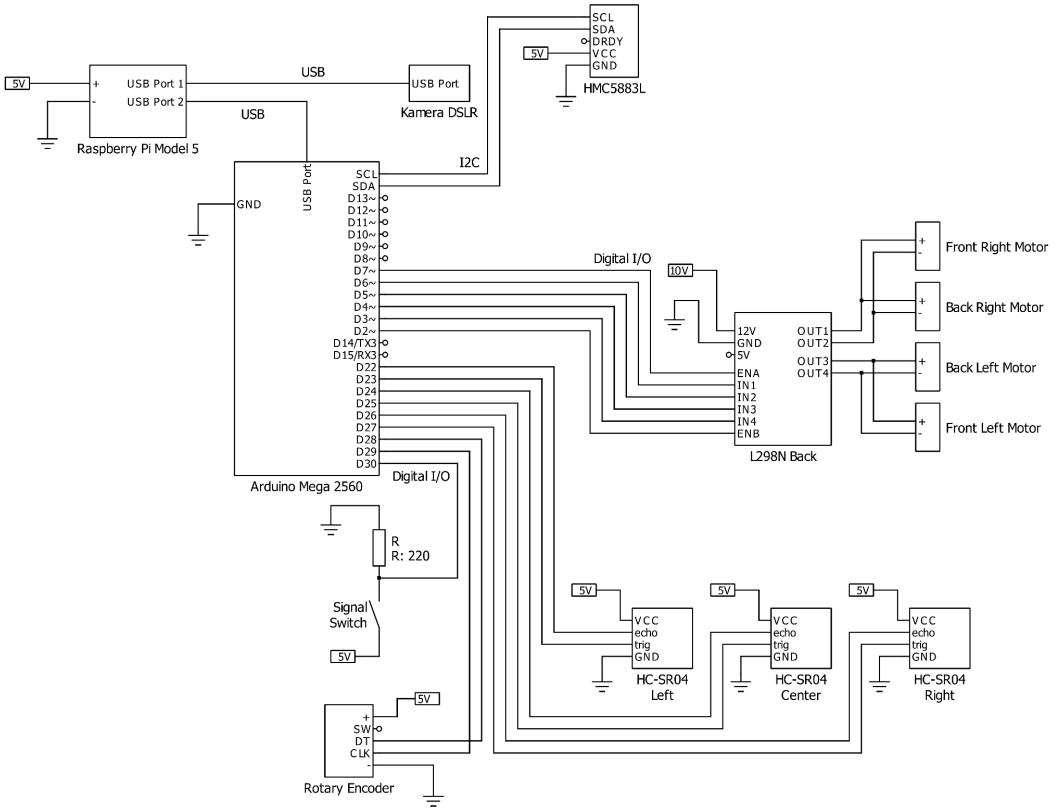
Gambar 3.3 DFD level 2 subsistem pengenalan dan deteksi objek

Tabel 3.3 Penjelasan DFD level 2 subsistem pengenalan dan deteksi objek

Parameter	Keterangan
Input	● Gambar digital dari tangkapan kamera.
Output	● Jenis target hasil pengklasifikasian. ● Posisi target. ● Jarak target.
Fungsi	● Membedakan jenis target dengan algoritma YOLO. ● Mendapatkan posisi dan jarak target dengan dengan *bounding box* dan algoritma *monocular vision*.

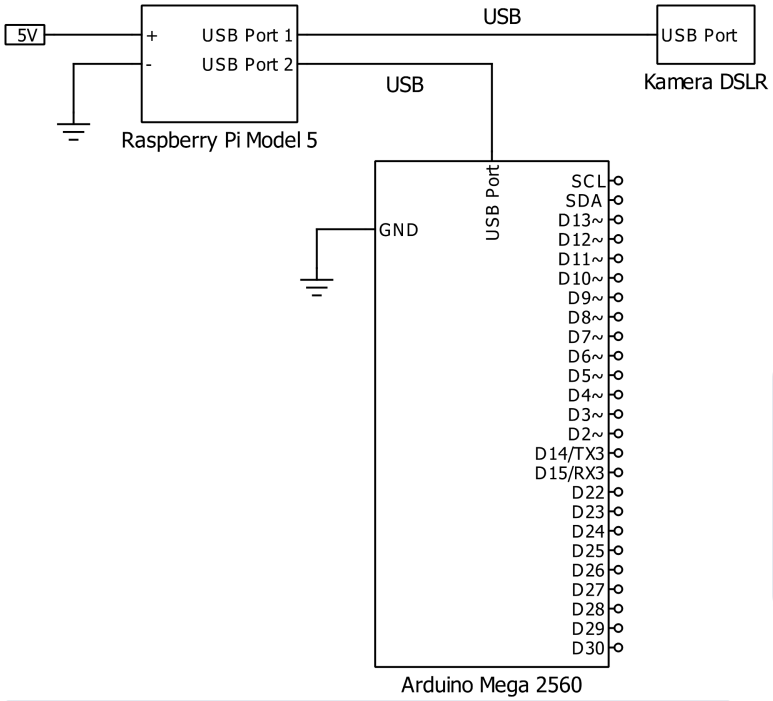
3.1.3 *Wiring Diagram* Sistem

Wiring diagram terdiri dari *wiring diagram* subsistem pengenalan dan deteksi objek serta subsistem navigasi. Subsistem pengenalan dan deteksi objek terdiri Raspberry Pi dan kamera DSLR. Subsistem navigasi terdiri dari Arduino, *motor driver*, motor DC, sensor ultrasonik, *rotary encoder*, dan *signal switch*. Jenis koneksi yang digunakan untuk komunikasi antara sensor, aktuator, dan mikrokontroler pada keseluruhan sistem antara lain USB, I2C, dan digital I/O. Gambar 3.4 merupakan *wiring diagram* keseluruhan dari produk yang dibuat.



Gambar 3.4 *Wiring diagram* keseluruhan sistem

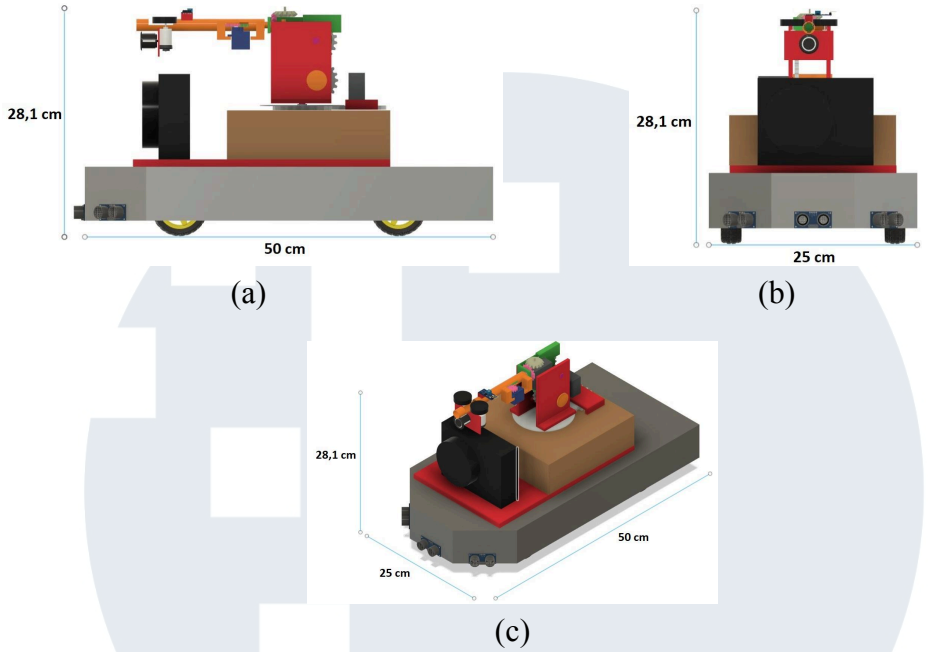
Raspberry Pi memberikan perintah kepada kamera DSLR untuk mengambil foto. Komunikasi antara Raspberry Pi dan kamera DSLR menggunakan USB. Pengolahan citra hasil tangkapan kamera DSLR dilakukan oleh Raspberry Pi. Data hasil pengolahan citra dikirim dari Raspberry Pi ke Arduino dengan komunikasi USB. Gambar 3.5 merupakan *wiring diagram* dari subsistem pengenalan dan deteksi objek.



Gambar 3.5 *Wiring diagram* subsistem pengenalan dan deteksi objek

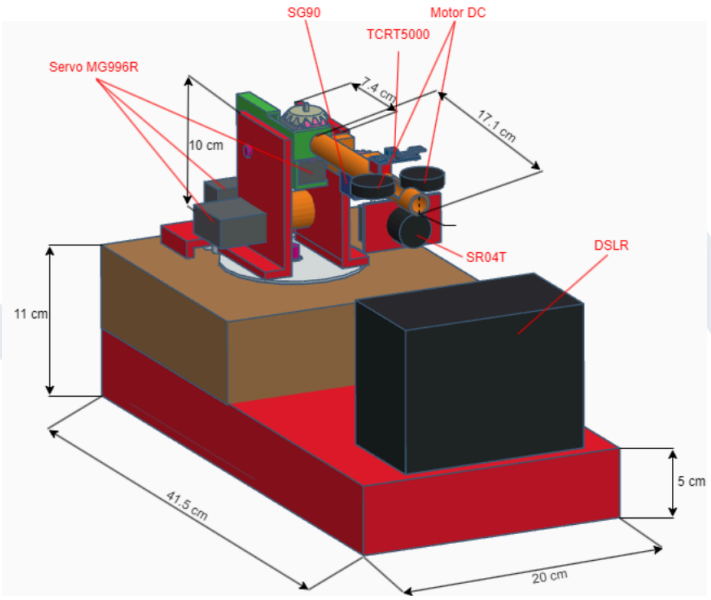
3.2 Implementasi Sistem

Mobile robot memiliki panjang keseluruhan 50 cm, lebar keseluruhan 25 cm, dan tinggi keseluruhan 28,1 cm. Badan *mobile robot* menggunakan akrilik dengan ketebalan 3 mm pada setiap dinding. Bagian-bagian yang rumit, seperti penyambung dinding dan *housing* komponen dibuat menggunakan *printer* 3D dengan bahan filamen ABS. Bagian-bagian yang dicetak memiliki ketebalan sebesar 3 mm. Berikut merupakan bentuk dari desain *mobile robot* yang dibuat.



Gambar 3.6 Dimensi produk (a) tampak samping, (b) tampak depan, (c) tampak isometrik

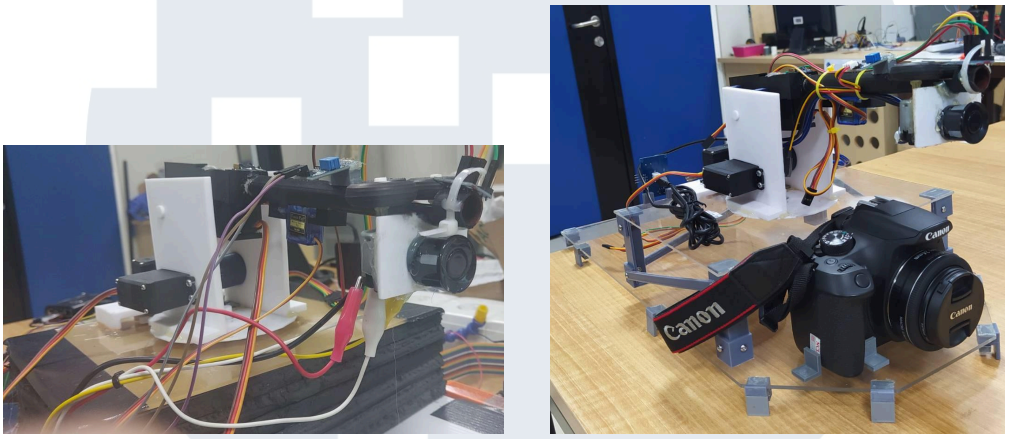
Desain dan dimensi subsistem pengenalan dan deteksi objek menggunakan produk yang sudah dibuat oleh peneliti sebelumnya, yakni subsistem penggerakan dan penembakan meriam. Gambar 3.7 merupakan desain meriam dari peneliti sebelumnya. Desain dan dimensi detail telah ditampilkan pada dokumen peneliti sebelumnya.



Gambar 3.7 Dimensi meriam dari peneliti sebelumnya

3.2.1 Hasil Implementasi

Platform subsistem penembakan dari penelitian sebelumnya disesuaikan agar bisa digunakan di atas platform *mobile robot*. Fondasi yang semulanya terbuat dari busa diganti menjadi tiang yang dibaut dengan penutup platform *mobile robot*. Fondasi serta kondisi meriam pada kondisi sebelum dan sesudah perbaikan dapat dilihat lebih jelas pada Gambar 3.8 berikut.



(a) (b)

Gambar 3.8 (a) Prototipe meriam buatan peneliti sebelumnya, (b) Prototipe meriam yang sudah diperbaiki

Secara umum, subsistem penembakan dapat bekerja sesuai dengan fungsinya, namun memiliki beberapa kekurangan. Untuk mengetahui kekurangan tersebut, maka subsistem penembakan perlu diuji ulang. Pengujian ulang dimulai dengan kalibrasi ulang servo, motor DC, sensor ultrasonik, sensor *infrared*, dan sensor *gyrometer*. Kalibrasi servo dilakukan untuk mengetahui jangkauan maksimal sudut servo dan sudut servo saat meriam mengarah lurus ke depan (posisi normal). Kalibrasi servo dilakukan dengan cara mengatur sudut servo secara manual melalui program. Kalibrasi motor DC dilakukan untuk mengetahui arah putaran dan kecepatan motor. Kalibrasi motor DC dilakukan secara langsung menggunakan *power supply*. Kalibrasi sensor ultrasonik dilakukan untuk memastikan jarak pengukuran sensor ultrasonik mendekati dengan jarak pengukuran asli. Kalibrasi sensor ultrasonik dilakukan dengan memprogram sensor. Kalibrasi sensor *infrared* dilakukan untuk mengetahui apakah peluru terdeteksi ketika dimasukkan ke dalam laras meriam. Kalibrasi sensor *infrared* dilakukan dengan memprogram sensor. Kalibrasi sensor *gyrometer* dilakukan

untuk mengetahui apakah sensor *gyrometer* bisa membaca perubahan posisi. Kalibrasi sensor *gyrometer* dilakukan dengan memprogram sensor.

Setelah semua sensor dan aktuator dikalibrasi, pengujian ulang program dilakukan. Program yang diuji ulang adalah program penentuan sudut parabola dan program PID kestabilan meriam. Pengujian ulang program penentuan sudut parabola dilakukan dengan langsung menembakan peluru ke target di depan meriam dengan variasi jarak, lalu peluru diamati apakah mengenai target atau tidak. Pengujian ulang program PID kestabilan meriam dilakukan mengamati langsung gerakan meriam dengan parameter PID yang sudah dibuat.

Pengujian ulang program yang dilakukan menghasilkan beberapa perbaikan dalam program. Perhitungan sudut parabola sebelumnya menggunakan metode *brute force* dan tidak sesuai dengan pendekatan perhitungan fisika. Perbaikan dilakukan agar sudut parabola dapat diperoleh dari data jarak target, kecepatan awal peluru, ketinggian meriam, dan ketinggian target. Persamaan (3.1) digunakan untuk menentukan sudut parabola berdasarkan data yang telah disebutkan. Persamaan (3.1) didapatkan melalui penurunan persamaan dasar gerak parabola.

1. Nilai t (waktu) dihitung dari komponen x.

$$x = v_{0x}t$$

$$x = v_0 cos(\theta)t$$

$$t = \frac{x}{v_0 cos(\theta)}$$

2. Nilai t pada komponen x dan y adalah sama. Nilai t yang diperoleh dari komponen x disubstitusi ke persamaan GLBB (komponen y).

$$h_t = h_c + v_0 sin(\theta)t - \frac{1}{2}gt^2$$

$$h_t = h_c + v_0 sin(\theta)\frac{x}{v_0 cos(\theta)} - \frac{1}{2}g\left(\frac{x}{v_0 cos(\theta)}\right)^2$$

$$h_t = h_c + \frac{xsin(\theta)}{cos(\theta)} - \frac{1}{2}g\frac{x^2}{v_0^2 cos^2(\theta)}$$

$$h_t cos^2(\theta) = h_c cos^2(\theta) + xsin(\theta)cos(\theta) - \frac{1}{2}g\frac{x^2}{v_0^2}$$

3. Konstanta dikelompokkan, dan dimisalkan sebagai a.

$$a = \frac{gx^2}{2v_0^2}$$

$$h_t cos^2(\theta) = h_c cos^2(\theta) + xsin(\theta)cos(\theta) - a$$

4. Bentuk persamaan diubah dengan identitas trigonometri $sin^2(\theta) + cos^2(\theta) = 1$.

$$h_t(1 - sin^2(\theta)) = h_c(1 - sin^2(\theta)) + xsin(\theta)cos(\theta) - a$$

$$h_t - h_t sin^2(\theta) = h_c - h_c sin^2(\theta) + xsin(\theta)cos(\theta) - a$$

$$-h_t sin^2(\theta) = -h_c sin^2(\theta) + xsin(\theta)cos(\theta) - a + h_c - h_t$$

5. Bentuk persamaan diubah dengan identitas trigonometri $sin^2(\theta) = \frac{1}{2}(1 - cos(2\theta))$ dan $sin(\theta)cos(\theta) = \frac{1}{2}sin(2\theta)$.

$$-\frac{h_t}{2}(1 - cos(2\theta)) = -\frac{h_c}{2}(1 - cos(2\theta)) + \frac{x}{2}sin(2\theta) - a + h_c - h_t$$

$$-\frac{h_t}{2} + \frac{h_t}{2}cos(2\theta) = -\frac{h_c}{2} + \frac{h_c}{2}cos(2\theta) + \frac{x}{2}sin(2\theta) - a + h_c - h_t$$

$$\frac{h_t}{2}cos(2\theta) = \frac{h_c}{2}cos(2\theta) + \frac{x}{2}sin(2\theta) - a + h_c - h_t - \frac{h_c}{2} + \frac{h_t}{2}$$

$$\frac{h_t}{2}cos(2\theta) = \frac{h_c}{2}cos(2\theta) + \frac{x}{2}sin(2\theta) - a + \frac{h_c}{2} - \frac{h_t}{2}$$

$$h_t cos(2\theta) = h_c cos(2\theta) + xsin(2\theta) - 2a + h_c - h_t$$

$$0 = -h_t cos(2\theta) + h_c cos(2\theta) + xsin(2\theta) - 2a + h_c - h_t$$

$$0 = (h_c - h_t)cos(2\theta) + xsin(2\theta) - 2a + h_c - h_t$$

6. Bentuk persamaan diubah dengan identitas trigonometri $Asin(\alpha) + Bcos(\alpha) = Ccos(\alpha - \beta)$, dimana $C = \sqrt{A^2 + B^2}$ dan $\beta = tan^{-1}\left(\frac{A}{B}\right)$.

$$0 = \sqrt{x^2 + (h_c - h_t)^2}cos(2\theta - \beta) - 2a + h_c - h_t \text{ dengan } \beta = tan^{-1}\left(\frac{x}{h_c - h_t}\right)$$

$$2a - h_c + h_t = \sqrt{x^2 + \left(h_c - h_t\right)^2} cos(2\theta - \beta)$$

$$cos(2\theta - \beta) = \frac{2a - h_c + h_t}{\sqrt{x^2 + \left(h_c - h_t\right)^2}}$$

$$2\theta - \beta = cos^{-1}\left(\frac{2a - h_c + h_t}{\sqrt{x^2 + \left(h_c - h_t\right)^2}}\right)$$

$$2\theta = cos^{-1}\left(\frac{2a - h_c + h_t}{\sqrt{x^2 + \left(h_c - h_t\right)^2}}\right) + \beta$$

$$\theta = \frac{cos^{-1}\left(\frac{2a - h_c + h_t}{\sqrt{x^2 + \left(h_c - h_t\right)^2}}\right) + \beta}{2}$$

7. Nilai a dan β dikembalikan ke dalam persamaan.

$$\theta = \frac{cos^{-1}\left(\frac{\frac{gx^2}{v_0^2} - h_c + h_t}{\sqrt{x^2 + \left(h_c - h_t\right)^2}}\right) + tan^{-1}\left(\frac{x}{h_c - h_t}\right)}{2} \quad (3.1)$$

Keterangan:

θ : Sudut parabola (°)

g : Percepatan gravitasi bumi (9,8 m/s)

v_0 : Kecepatan awal peluru (m/s)

x : Jarak antara target dan meriam (m)

h_c : Ketinggian meriam (m)

h_t : Ketinggian target (m)

Implementasi dilanjut ke subsistem pengenalan dan deteksi objek. Subsistem pengenalan dan deteksi objek berfungsi untuk membedakan target, apakah target termasuk ke dalam target tembak atau tidak. Target diklasifikasikan menjadi tiga, yakni “teroris”, “tentara”, dan “warga sipil”. “Teroris” merupakan target yang akan ditembak, sedangkan “tentara” dan “warga sipil” merupakan target yang tidak ditembak. “Teroris” memiliki ciri-ciri membawa senjata api, menggunakan atribut-atribut tempur yang tidak terlalu lengkap (hanya helm, rompi, atau sepatu bot), dan menutupi area kepala menyisakan mata dan mulut.

“Tentara” memiliki ciri-ciri memakai seragam loreng tentara dan menggunakan atribut-atribut tempur yang lengkap, seperti senjata api, helm, rompi, tas, dan sepatu bot. “Warga sipil” memiliki ciri-ciri memakai pakaian warga sipil pada umumnya, seperti kaos, kemeja, celana pendek, dan celana panjang.

Algoritma yang digunakan untuk pengenalan dan deteksi objek adalah YOLO11m. Algoritma ini dipilih dikarenakan menghasilkan performa terbaik dari tes terhadap 12 sampel data target pada jarak 2 meter. Performa perbandingan algoritma-algoritma dapat dilihat lebih detail pada Tabel 3.4 hingga 3.6.

Tabel 3.4 Perbandingan performa algoritma-algoritma pengenalan dan deteksi objek

	Algoritma		
	MobileNet V2 SSD FPN-Lite 640x640	YOLO11s	YOLO8s
Parameter umum	● Jarak target 2 m ● Ukuran foto 5184×3456 px (di-*resize* menjadi 640×640 px saat *inference*) ● 3 “warga sipil”, 3 “tentara”, 3 “teroris”, 3 “*background*” ● *Confidence level* 50%		
Parameter *training*	“img_size” = 640 “steps” = 20000 “batch” = 16 Durasi = 8 jam	“img_size” = 640 “epoch” = 100 “batch” = 16 Durasi = 0,5 jam	“img_size” = 640 “steps” = 100 “batch” = 16 Durasi = 0,5 jam
Akurasi	0,737	0,87	0,857
Presisi	0,583	0,769	0,75
Recall	0,583	0,769	0,75
F1 *Score*	0,583	0,769	0,75
Kesimpulan	Terbaik ketiga	Terbaik pertama	Terbaik kedua

Tabel 3.5 Perbandingan performa algoritma YOLO11

	Algoritma		
	YOLO11s	YOLO11m	YOLO11l
Parameter umum	● Jarak target 2 m ● Ukuran foto 5184×3456 px (di-*resize* menjadi 640×640 px saat *inference*) ● 3 “warga sipil”, 3 “tentara”, 3 “teroris”, 3 “*background*” ● *Confidence level* 50%		
Parameter *training*	“img_size” = 640 “epoch” = 100 “batch” = 16	“img_size” = 640 “epoch” = 100 “batch” = 8	“img_size” = 640 “steps” = 100 “batch” = 8

	Algoritma		
	YOLO11s	YOLO11m	YOLO11l
	Durasi = 0,5 jam	Durasi = 1,5 jam	Durasi = 2 jam
Akurasi	0,87	0,96	0,909
Presisi	0,769	0,923	0,833
Recall	0,769	0,923	0,833
F1 *Score*	0,769	0,923	0,833
Kesimpulan	Terbaik ketiga	Terbaik pertama	Terbaik kedua

Tabel 3.6 Perbandingan performa algoritma YOLO11m pada berbagai *epoch*

	Epoch		
	100	140	244
Parameter umum	• Jarak target 2 m • Ukuran foto 5184×3456 px (di-*resize* menjadi 640×640 px saat *inference*) • 3 "warga sipil", 3 "tentara", 3 "teroris", 3 "*background*" • *Confidence level* 50%		
Parameter *training*	"img_size" = 640 "batch" = 8 Durasi = 1,5 jam	"img_size" = 640 "batch" = 8 Durasi = 2 jam	"img_size" = 640 "batch" = 8 Durasi = 3 jam
Akurasi	0,96	0,857	0,909
Presisi	0,923	0,75	0,833
Recall	0,923	0,75	0,833
F1 *Score*	0,923	0,75	0,833
Kesimpulan	Terbaik pertama	Terbaik ketiga	Terbaik kedua

Pemrograman menggunakan aplikasi Thonny dengan bahasa pemrograman Python. Mikrokontroler yang digunakan adalah Raspberry Pi 5. *Dataset* untuk proses training diambil dari platform Roboflow. Proses *training dataset* menggunakan GPU lokal dengan CUDA dan cuDNN yang sudah di-*install*. *Framework* yang digunakan dalam proses training ini adalah PyTorch.

Dataset yang digunakan untuk *training* diambil dari Roboflow. *Dataset* terdiri dari gambar "teroris", "tentara", dan "warga sipil" yang telah dianotasi serta beberapa gambar tanpa anotasi (tanpa "teroris", "tentara", dan "warga

sipil"). Jumlah *dataset* yang digunakan adalah 1703 buah gambar dengan rasio 70% untuk *train*, 15% untuk *validation*, dan 15% untuk *test.* Rasio tersebut umum digunakan dalam partisi *dataset* keperluan *machine learning*, seperti dalam jurnal [22] dan [23]. *Dataset* disimpan dalam format YOLO yang terdiri dari gambar *dataset*, anotasi dalam format txt, dan konfigurasi dalam format yaml.

Dataset yang digunakan dalam proses *training* adalah *dataset* bagian *train* dan *validation*. Proses training menggunakan parameter "epoch" sebesar 100 dan "batch" sebesar 8. Proses *training* memakan waktu kurang lebih 1,5 jam. Setelah proses *training* selesai, hasil *training* disimpan ke dalam *file* berformat pt. *File* pt tersebut digunakan dalam program untuk pendeteksian target.

Program pengolahan citra menggunakan beberapa *library* Python, yakni time, serial, gphoto2, signal, os, subprocess, cv2, numpy, dan ultralytics. Program dimulai dengan deklarasi variabel berupa ukuran gambar input (piksel) dan jangkauan sudut servo (derajat).

Program akan menjalankan *setup* kamera DSLR. Proses *setup* DSLR dimulai dengan mematikan proses *library* gphoto2 yang sedang berjalan. Setelah proses yang sedang berjalan dimatikan, *folder* penyimpanan foto hasil tangkapan kamera dibersihkan (semua *file* dihapus). Setelah *folder* bersih, kamera mengambil foto lalu disimpan di *folder* yang ditentukan dengan format penamaan tertentu.

Setelah foto disimpan dalam *folder*, foto tersebut diproses dengan OpenCV dan algoritma YOLO11m (hasil *training*) untuk mengetahui apakah ada target yang terdeteksi. Pengolahan citra menghasilkan data nama kelas target, ID kelas target (0/1/2), *confidence level* (0 hingga 1), titik tengah target (piksel), dan ukuran target (piksel).

Data ukuran target (piksel) digunakan dalam pengukuran jarak dengan *monocular vision*. Pengukuran *monocular vision* memerlukan data lebar sesungguhnya (cm) dan *focal length* (piksel). *Focal length* pada umumnya memiliki satuan mm, sehingga perlu dikonversi ke piksel dengan Persamaan (3.2). Persamaan (3.2) didapatkan melalui perbandingan antara ukuran *focal length* dengan ukuran sensor kamera.

$$\frac{d_{fl\,mm}}{d_{fl\,px}} = \frac{width_{sensor\,mm}}{width_{res\,px}}$$

$$d_{fl\,px} = \frac{width_{res\,px} \cdot d_{fl\,mm}}{width_{sensor\,mm}} \qquad (3.2)$$

Keterangan:

$d_{fl\,px}$: Jarak *focal length* (piksel)

$d_{fl\,mm}$: Jarak *focal length* (mm)

$width_{res\,px}$: Lebar resolusi foto maksimum (piksel)

$width_{sensor\,mm}$: Lebar sensor gambar pada kamera (mm)

Kamera DSLR yang digunakan memiliki spesifikasi lensa tetap dengan *focal length* sebesar 24 mm, lebar sensor gambar sebesar 22,3 mm, dan lebar resolusi foto maksimum sebesar 5184 piksel. Dengan menggunakan Persamaan (3.2), didapat jarak *focal length* dalam piksel sebesar 5579,193 px. Setelah mendapatkan nilai tersebut, perhitungan *monocular vision* dapat dilakukan dengan Persamaan (3.3) [24]. Persamaan (3.3) didapatkan melalui perbandingan antara ukuran *focal length* dengan ukuran target.

$$\frac{d_{monocular}}{d_{fl\,px}} = \frac{width_{real}}{width_{img\,process}}$$

$$d_{monocular} = \frac{width_{real} \cdot d_{fl\,px}}{width_{img\,process}} \qquad (3.3)$$

Keterangan:

$d_{monocular}$: Jarak *monocular vision* (cm)

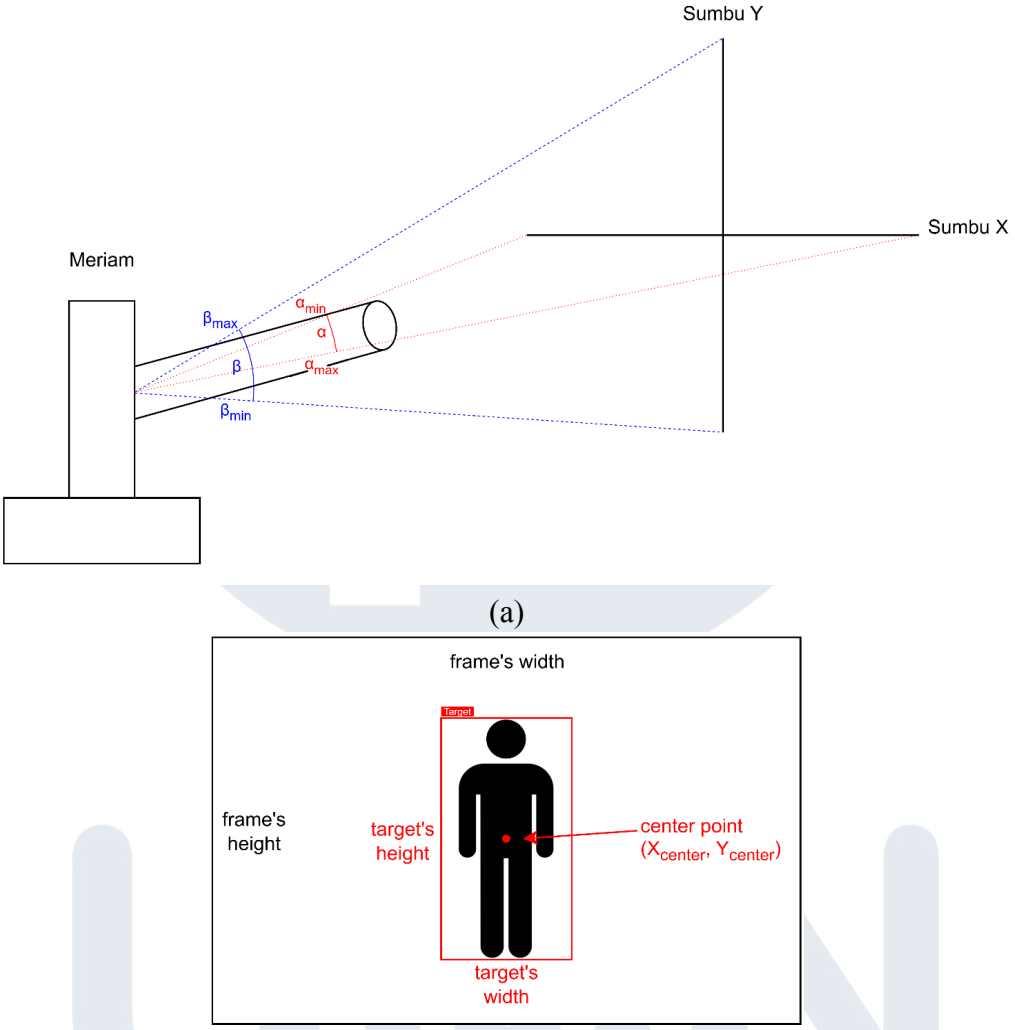
$d_{fl\,px}$: Jarak *focal length* (px)

$width_{real}$: Lebar target sesungguhnya (cm)

$width_{img\,process}$: Lebar target hasil *image processing* (px)

Setelah mendapatkan data jarak *monocular vision* dan titik tengah target, data tersebut dimasukkan ke dalam *array*. Data yang dimasukkan ke dalam *array* hanya data milik "teroris" (syarat ID kelas adalah 2). Bila ada beberapa "teroris" yang terdeteksi, maka fungsi mengembalikan data "teroris" dengan jarak terdekat. Bila tidak ada target yang terdeteksi ("teroris", "tentara", dan "warga sipil"), maka fungsi mengembalikan "None".

Titik tengah target "teroris" yang sebelumnya masih berupa piksel diubah menjadi sudut servo. Pergerakan servo meriam dinyatakan dalam sumbu X (horizontal) dan Y (vertikal). Visualisasi bidang pergerakan servo dan data-data *image processing* target dapat dilihat pada Gambar 3.9.



Gambar 3.9 (a) Visualisasi sudut pergerakan servo meriam, (b) Visualisasi data-data *image processing* target

Perhitungan sudut servo menggunakan Persamaan (3.4) dan (3.5) [25]. Persamaan (3.4) dan (3.5) didapatkan melalui perbandingan antara ukuran resolusi foto, *Field of View* (FOV) kamera, dan jangkauan sudut servo. Adapun jangkauan sudut servo diatur agar sama dengan FOV kamera yang digunakan.

1. Perhitungan perbandingan antara resolusi foto dan jangkauan sudut servo.

$$\frac{\alpha}{\alpha_{max} - \alpha_{min}} = \frac{X_{center}}{width_{frame}}$$

$$\alpha = \frac{X_{center} \cdot (\alpha_{max} - \alpha_{min})}{width_{frame}}$$

$$\frac{\beta}{\beta_{max} - \beta_{min}} = \frac{Y_{center}}{height_{frame}}$$

$$\beta = \frac{Y_{center} \cdot (\beta_{max} - \beta_{min})}{height_{frame}}$$

2. Sudut hasil perhitungan disesuaikan dengan konfigurasi meriam. Sudut hasil perhitungan dapat ditambah dengan sudut minimal atau dikurang dengan sudut maksimal servo. Prototipe ini menggunakan konfigurasi sudut maksimal servo dikurang sudut hasil perhitungan.

$$\alpha_{target} = \alpha_{max} - \alpha$$

$$\alpha_{target} = \alpha_{max} - \left(\frac{X_{center}}{width_{frame}} \cdot (\alpha_{max} - \alpha_{min}) \right) \quad (3.4)$$

$$\beta_{target} = \beta_{max} - \beta$$

$$\beta_{target} = \beta_{max} - \left(\frac{Y_{center}}{height_{frame}} \cdot (\beta_{max} - \beta_{min}) \right) \quad (3.5)$$

Keterangan:

α_{target} : Sudut bidang horizontal mengarah ke titik tengah target (°)

β_{target} : Sudut bidang vertikal mengarah ke titik tengah target (°)

α_{max} : Sudut gerak maksimal servo di bidang horizontal (°)

β_{max} : Sudut gerak maksimal servo di bidang vertikal (°)

α_{min} : Sudut gerak minimal servo di bidang horizontal (°)

β_{min} : Sudut gerak minimal servo di bidang vertikal (°)

X_{center} : Koordinat X dari titik tengah target (px)

Y_{center} : Koordinat Y dari titik tengah target (px)

$width_{frame}$: Lebar *frame* foto target (px)

$height_{frame}$: Tinggi *frame* foto target (px)

Servo yang digunakan memiliki tingkat keakuratan sebesar 1°. Data untuk perhitungan resolusi servo adalah sebagai berikut:

α_{max} = 111°

β_{max} = 106°

α_{min} = 59°

β_{min} = 78°

$width_{frame}$ = 5184 px

$height_{frame}$ = 3456 px

Resolusi servo dapat dihitung dengan Persamaan (3.6) dan (3.7).

$$Res_{horizontal} = \frac{width_{frame}}{\alpha_{max} - \alpha_{min}} = \frac{5184px}{111° - 59°} = 99,692\ px/° \quad (3.6)$$

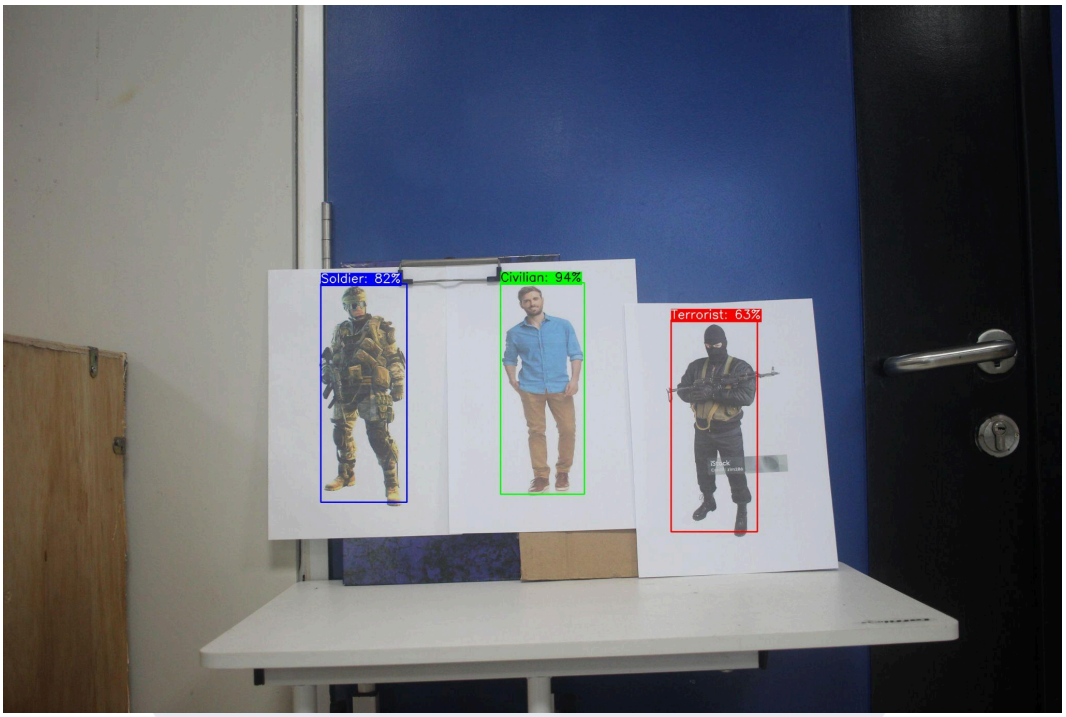
$$Res_{veritkal} = \frac{height_{frame}}{\alpha_{max} - \alpha_{min}} = \frac{3456px}{106° - 78°} = 96,768\ px/° \quad (3.7)$$

Keterangan:

$Res_{horizontal}$: Resolusi servo bidang horizontal (px/°)

$Res_{vertikal}$: Resolusi servo bidang vertikal (px/°)

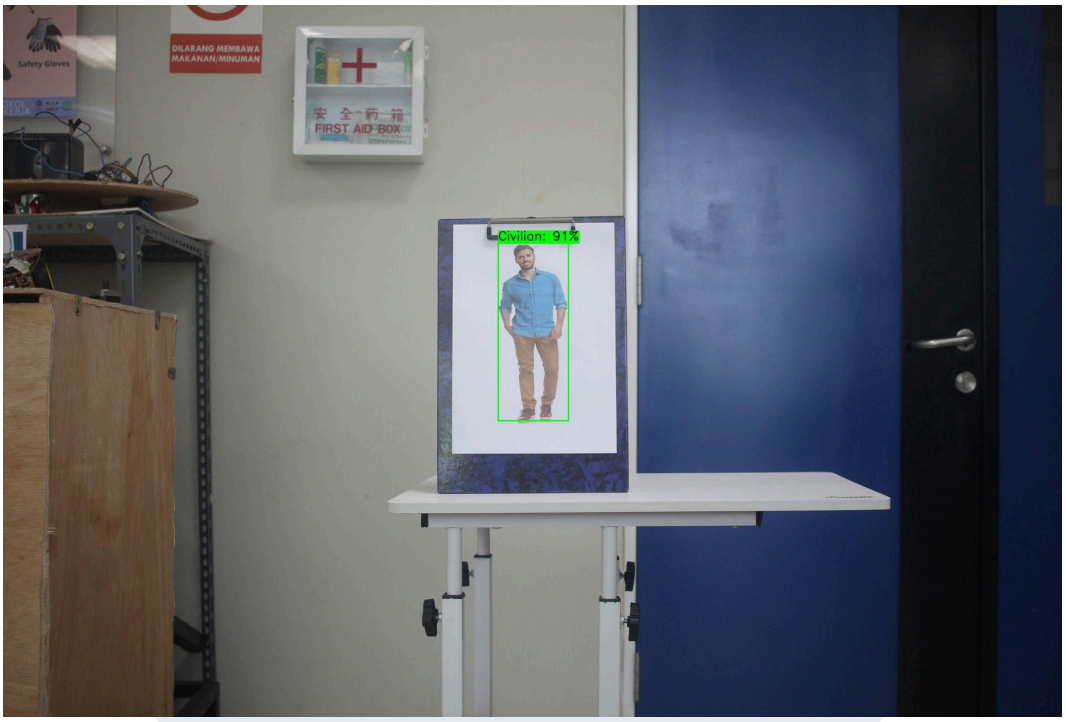
Contoh hasil pengolahan citra dapat dilihat pada Gambar 3.10 hingga 3.13. Gambar 3.10 menampilkan target yang terdeteksi, yakni "teroris" dengan *confidence level* sebesar 63%, "tentara" dengan *confidence level* sebesar 82%, dan "warga sipil" dengan *confidence level* sebesar 94%. Gambar 3.11 menampilkan data-data penting dari masing-masing target, seperti nama kelas target, ID kelas target, *confidence level*, ukuran target (piksel), titik tengah target (piksel), dan jarak *monocular vision* (cm). Sesuai dengan syarat sebelumnya, data yang akan dikirim ke Arduino hanya data milik "teroris" dengan jarak terdekat. Gambar 3.12 menampilkan target "warga sipil" dengan *confidence level* sebesar 91%. Gambar 3.13 menampilkan data-data penting dari "warga sipil" yang terdeteksi, serta menampilkan keterangan "*No Terrorist Detected!*".



Gambar 3.10 Foto hasil pengolahan citra dengan keberadaan “teroris”

```
Class ID: 0
Class Name: Civilian
Confidence Level: 0.94
Bounding Box Width: 410, Height: 1032
Center Point: (2638, 1871)
Monocular Distance: 129.51 cm
--------------------------------------------------
Class ID: 1
Class Name: Soldier
Confidence Level: 0.82
Bounding Box Width: 421, Height: 1065
Center Point: (1764, 1893)
Monocular Distance: 126.13 cm
--------------------------------------------------
Class ID: 2
Class Name: Terrorist
Confidence Level: 0.63
Bounding Box Width: 422, Height: 1033
Center Point: (3478, 2053)
Monocular Distance: 125.83 cm
--------------------------------------------------
Detection result saved as detection_result.jpg
X degree:  76
Y degree:  92
```

Gambar 3.11 Parameter-parameter hasil pengolahan citra dengan keberadaan “teroris”



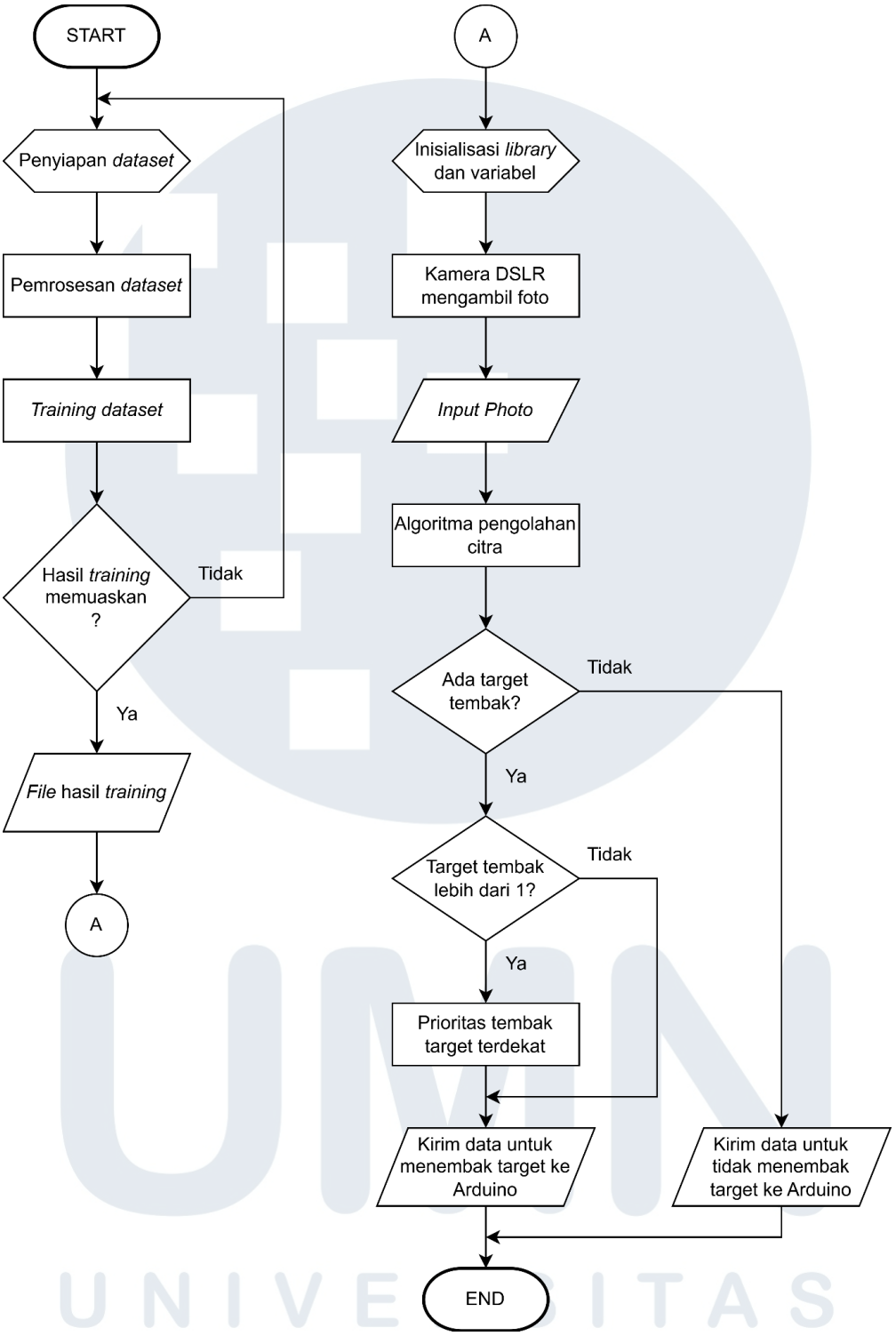
Gambar 3.12 Foto hasil pengolahan citra tanpa keberadaan “teroris”

```
Class ID: 0
Class Name: Civilian
Confidence Level: 0.92
Bounding Box Width: 344, Height: 869
Center Point: (2596, 1582)
Monocular Distance: 154.36 cm
--------------------------------------------------
Detection result saved as detection_result.jpg
No Terrorist Detected!
```

Gambar 3.13 Parameter-parameter hasil pengolahan citra tanpa keberadaan “teroris”

Data jarak *monocular vision* dan titik tengah dalam sudut dari target “teroris” dikirim ke Arduino Mega dengan komunikasi serial. *Baud rate* diatur sebesar 115200 bps dan *timeout* sebesar 0,05 detik. Bila tidak ada “tentara” yang terdeteksi, maka data yang dikirimkan adalah nilai titik tengah sudut (X = 85° dan Y = 92°) serta jarak *monocular vision* sebesar 999 cm. Dengan data jarak *monocular vision* sebesar 999 cm (9,99 m), meriam tidak akan menembak peluru, dikarenakan jarak tembak maksimal berada di kisaran 2,7 m (perhitungan peneliti sebelumnya).

Keseluruhan proses implementasi subsistem pengenalan dan deteksi objek dapat dinyatakan dalam *flowchart* pada Gambar 3.14.



Gambar 3.14 *Flowchart* implementasi subsistem pengenalan dan deteksi objek

Implementasi dilanjut ke proses integrasi *mobile robot*. Integrasi *hardware* menghasilkan sebuah *mobile robot* yang dapat bergerak dari titik *start* ke titik *finish* serta mampu menghindari rintangan di dalam arena. *Mobile robot* dilengkapi dengan meriam yang dapat menembak target "teroris". *Mobile robot*

dilengkapi dengan sakelar *power* dan sakelar *signal*. Sakelar *power* digunakan untuk menyalakan atau mematikan *mobile robot*. Sakelar *signal* digunakan untuk menjalankan *mobile robot* atau me-*reset* algoritma navigasi *mobile robot*.

Integrasi *software* menghasilkan program yang dapat mengoperasikan *mobile robot*. Urutan kerja program *mobile robot* per subsistem adalah sebagai berikut:

1. Subsistem pengenalan dan deteksi objek
    a. Program melakukan inisialisasi.
    b. Program membuka komunikasi serial.
    c. Program mengeksekusi *loop* dengan interval 5 detik.
    d. Program mengecek karakter ‘b’ sebagai indikator *mobile robot* mencapai titik *finish*. Bila indikator ‘b’ diterima, maka program akan keluar dari *loop* utama dan lanjut ke tahap 1k. Bila tidak menerima indikator ‘b’, maka program akan melanjutkan proses.
    e. Program mengirim karakter ‘a’ untuk meminta *mobile robot* berhenti.
    f. Program mengambil foto dengan kamera.
    g. Program memproses foto hasil tangkapan kamera. Data ‘None’ akan dihasilkan bila tidak ada target tembak. Bila ada target tembak, maka data sudut servo X, sudut servo Y, dan jarak monokular dari target tembak terdekat akan dihasilkan.
    h. Bila ada target tembak, data sudut servo X, sudut servo Y, dan jarak monokular dikirim.
    i. Program menunggu karakter ‘c’ sebagai indikator proses penembakan hingga selesai. Setelah proses penembakan selesai, program mengulangi *loop* utama dari langkah 1c.
    j. Bila tidak ada target tembak, data yang dikirim adalah 85, 92, dan 999 sebagai nilai *default* servo X dan Y serta jarak monokular yang diatur bernilai besar. Program mengulangi *loop* utama dari langkah 1c.

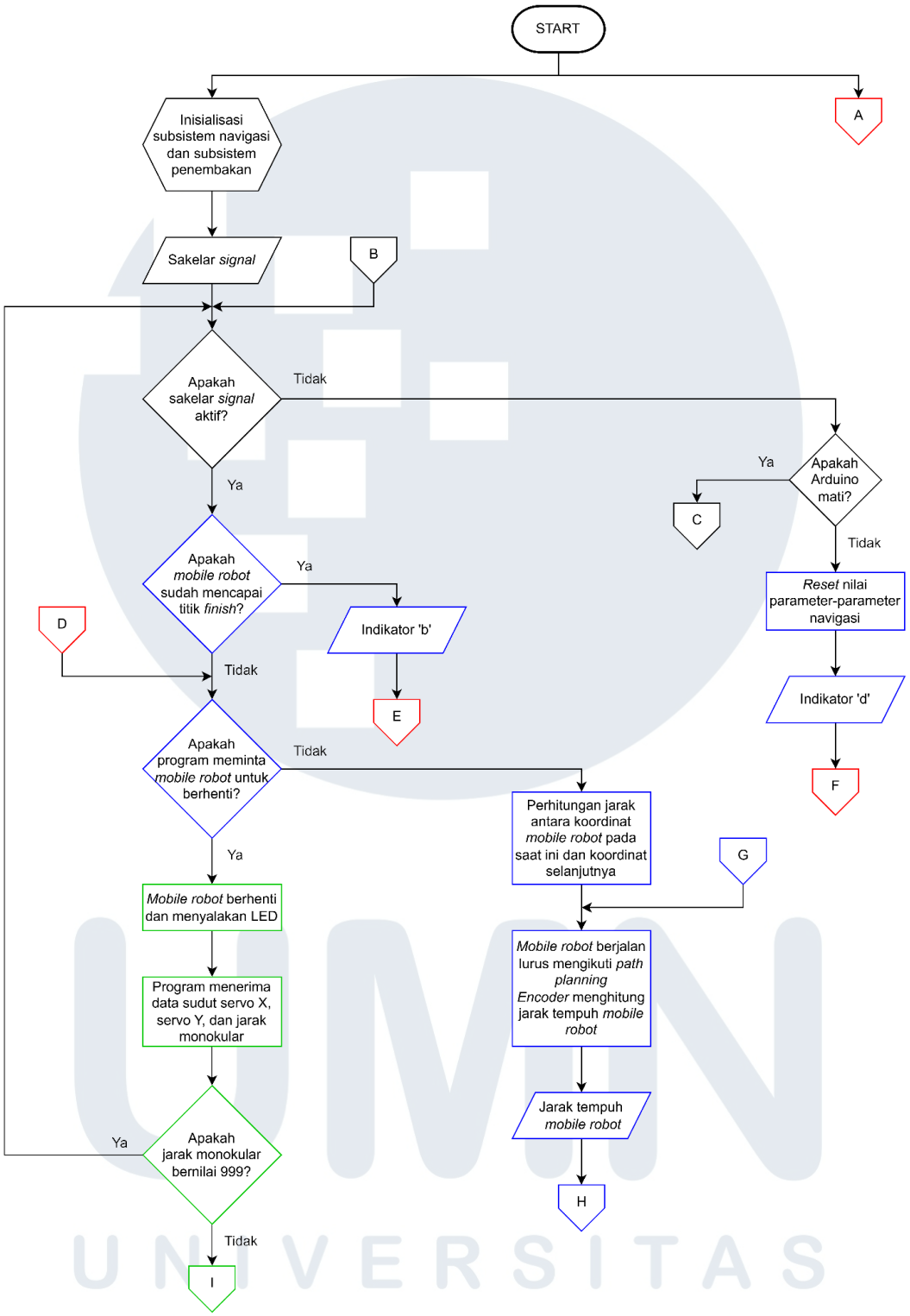
k. Program keluar dari *loop* utama, lalu menunggu instruksi *reset* dari karakter ‘d’ (sakelar *signal*). Bila ada karakter ‘d’ yang masuk, maka program akan kembali ke tahap 1c.

2. Subsistem navigasi
   a. Program melakukan inisialisasi.
   b. Program mendapatkan informasi *path planning* dalam bentuk *list* koordinat-koordinat.
   c. Program mengecek status sakelar *signal*. Bila dalam keadaan mati, maka program akan menghentikan *mobile robot*, me-*reset* parameter-parameter navigasi, dan mengirimkan karakter ‘d’. Bila dalam keadaan hidup, maka program akan melanjutkan proses.
   d. Program mengecek apakah *mobile robot* sudah mencapai titik *finish* atau belum. Bila sudah, maka *mobile robot* akan berhenti dan mengirimkan indikator karakter ‘b’. Bila belum, maka program akan melanjutkan proses.
   e. Program menghitung jarak antara koordinat *mobile robot* pada saat ini dan koordinat selanjutnya.
   f. Pada setiap koordinat, *mobile robot* berjalan maju sebagai *default*. Program akan membandingkan nilai jarak tempuh hasil pengukuran dengan hasil perhitungan. Bila nilai jarak tempuh hasil pengukuran dan hasil perhitungan telah bernilai sama, maka program akan memvalidasi keadaan rintangan. Bila belum, maka *mobile robot* akan terus berjalan hingga kedua nilai tersebut sama.
   g. Bila keadaan rintangan belum tervalidasi aman, maka *mobile robot* akan bergerak maju sampai rintangan tervalidasi aman. Bila keadaan rintangan telah tervalidasi aman, maka *mobile robot* akan berbelok ke kiri atau kanan sesuai dengan *path planning*. Program memperbaharui koordinat *mobile robot* saat ini dengan menambah *increment* dalam iterasi.
   h. Program mengecek apakah ada karakter ‘a’. Bila ada karakter ‘a’ yang masuk, maka *mobile robot* akan berhenti dan bersiap

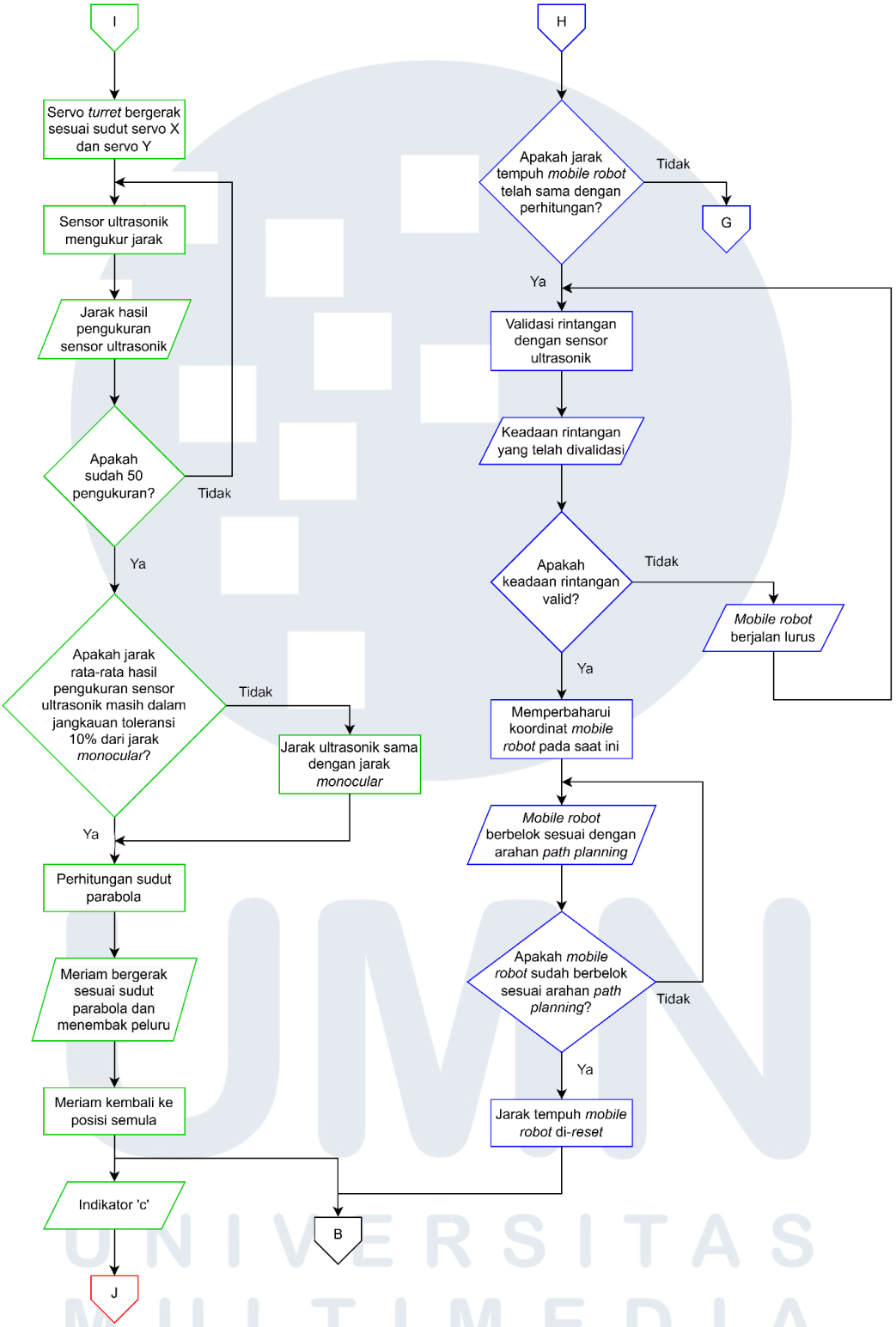
menjalankan proses penembakan. Bila tidak ada, maka program akan kembali ke tahap 2c.

3. Subsistem penembakan
    a. Program melakukan inisialisasi.
    b. Program menerima data sudut servo X, sudut servo Y, dan jarak monokular.
    c. Bila program menerima jarak monokular sebesar 999, maka program akan menggerakkan kembali *mobile robot*.
    d. Bila program tidak menerima jarak monokular sebesar 999, maka program akan memutar *turret* meriam sesuai besar sudut servo X dan servo Y. Program mengukur jarak dengan sensor ultrasonik. Pengukuran dilakukan sebanyak 50 kali. Jarak rata-rata dari hasil pengukuran yang masih dalam toleransi 10% dari jarak *monocular* digunakan sebagai nilai representatif sensor ultrasonik.
    e. Bila semua data sudut servo X, sudut servo Y, jarak monokular, dan jarak ultrasonik valid (masih dalam jangkauan tembak), maka program akan menghitung sudut parabola meriam.
    f. Program melakukan proses penembakan dengan menggerakkan laras meriam sesuai sudut parabola, memutar roda pelontar, menggerakkan pelatuk, dan mengembalikan posisi meriam seperti semula.
    g. Program mengirim karakter ‘c’ sebagai indikator proses penembakan telah dilakukan. Program lanjut menggerakkan *mobile robot*.
    h. Bila semua data sudut servo X, sudut servo Y, jarak monokular, dan jarak ultrasonik tidak valid (di luar jangkauan tembak), maka program tidak akan menembakkan meriam. Program langsung menuju ke tahap 3g.

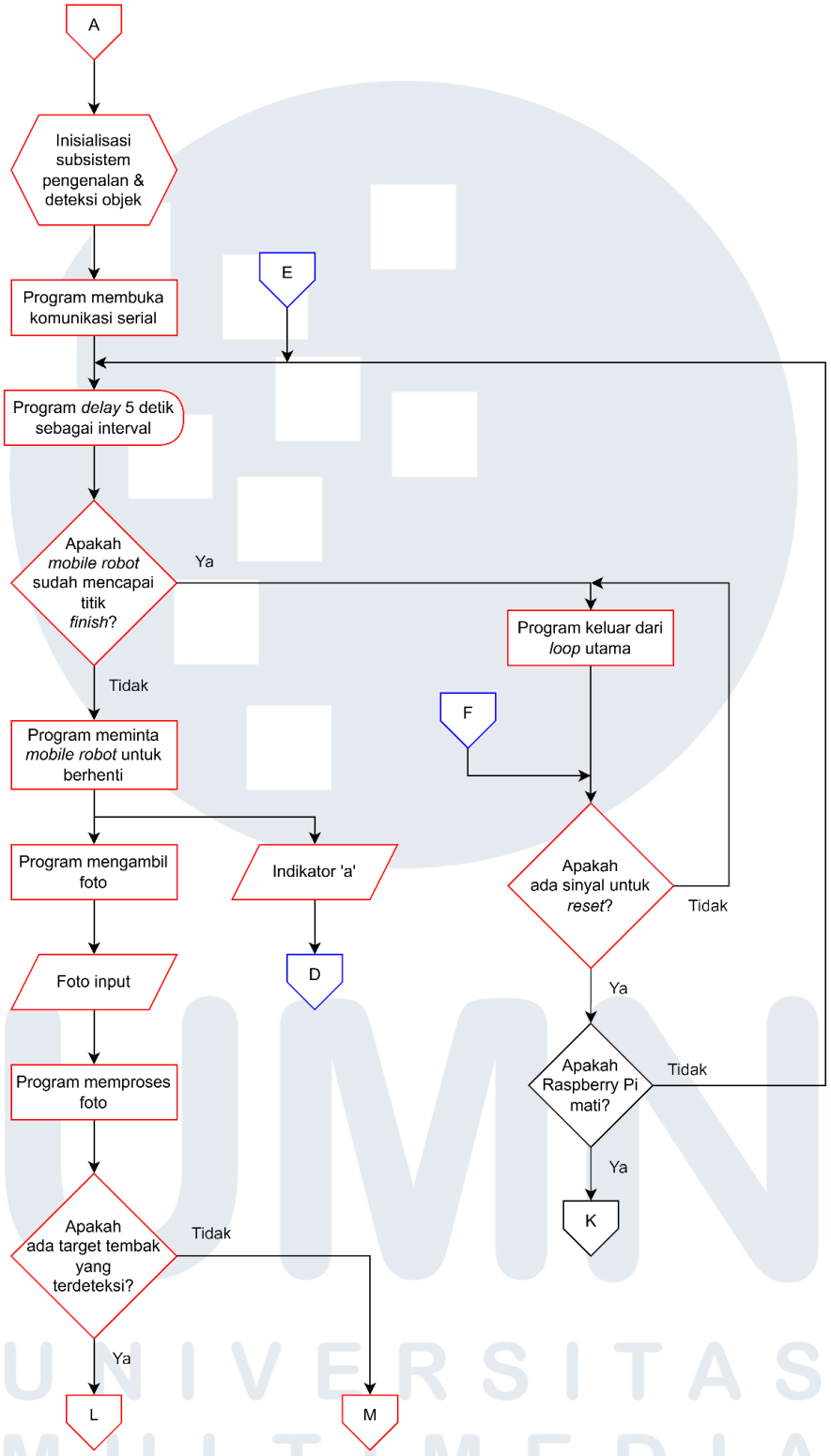
Urutan kerja program tersebut dapat dinyatakan sebagai *flowchart* pada Gambar 3.15. *Source code* program dilampirkan pada bagian Lampiran.



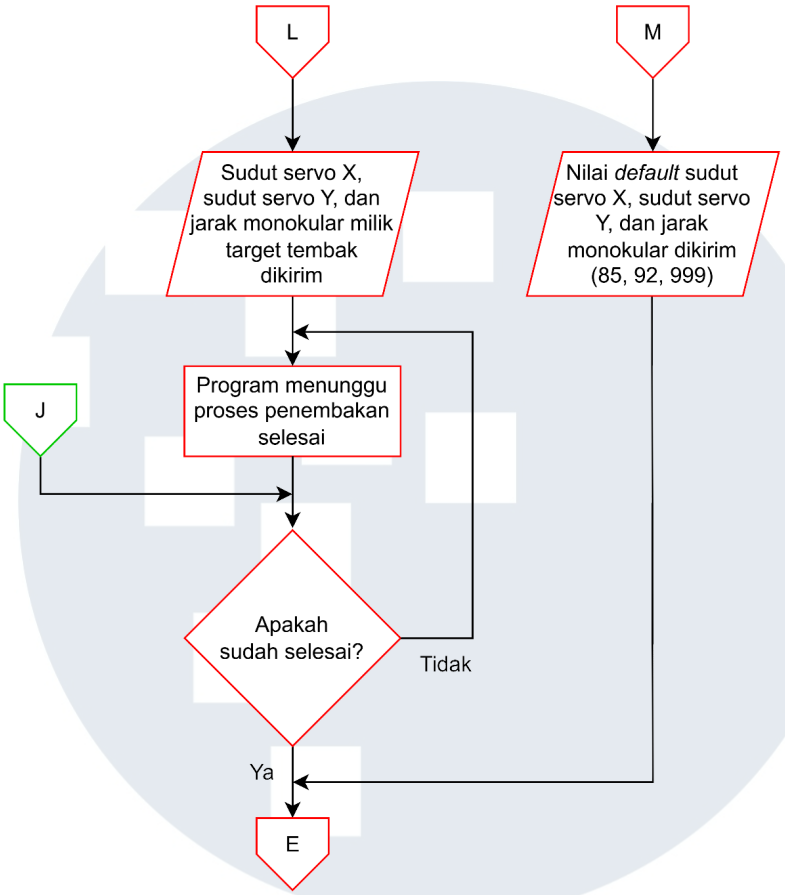
Gambar 3.15 *Flowchart* program integrasi *mobile robot*



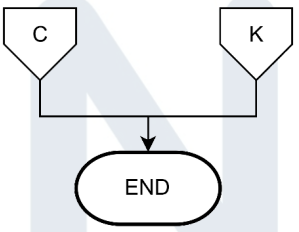
Gambar 3.15 *Flowchart* program integrasi *mobile robot* (lanjutan)



Gambar 3.15 *Flowchart* program integrasi *mobile robot* (lanjutan)



Keterangan warna:
Biru: Subsistem navigasi
Merah: Subsistem pengenalan dan deteksi objek
Hijau: Subsistem penembakan
Hitam: Tidak terkait subsistem (umum)



Gambar 3.15 *Flowchart* program integrasi *mobile robot* (lanjutan)

3.2.2 Hambatan Implementasi

Hambatan yang dijumpai saat perbaikan subsistem penembakan antara lain sebagai berikut:

1. Struktur meriam dirakit dengan lem, sehingga menyulitkan proses perbaikan atau reparasi. Struktur yang dilem tidak bisa dibongkar pasang, karena sudah bersifat permanen.

2. Servo untuk elevasi meriam rusak. Kerusakan yang terjadi adalah gerakan servo tidak bisa dikendalikan dengan mikrokontroler. Kerusakan kemungkinan disebabkan karena *over voltage*, mengingat peneliti sebelumnya mengaku menyuplai tegangan 10 V untuk servo (tegangan normal adalah 5 V).
3. Struktur meriam bergoyang ketika bergerak, bahkan bisa jatuh tersungkur ke depan. Struktur meriam memiliki banyak celah kosong (*gap*), sehingga struktur meriam mudah bergoyang bahkan ketika disentuh.
4. *Wiring* dan penempatan komponen tidak rapi. Kabel dibiarkan menggantung pada setiap komponen. Komponen seperti *motor driver* dan PCB sensor ultrasonik diletakkan pada kotak terpisah dengan struktur meriam. Pada proyek peneliti sebelumnya, *wiring* dan penempatan komponen dirancang sebatas pada meriam yang bersifat stasioner dan tidak bergerak di atas platform.

Hambatan yang dijumpai saat implementasi subsistem pengenalan dan deteksi objek antara lain sebagai berikut:

1. Proses *training dataset* menggunakan GPU lokal. Proses komputasi dalam *training* dibatasi oleh spesifikasi GPU yang dimiliki. GPU yang digunakan dalam proses training memiliki *dedicated memory* sebesar 6 GB.
2. Data sudut servo hasil pengolahan citra tidak menembak target dengan akurat.
3. Versi Python maksimal yang didukung *library* ultralytics adalah Python 3.10, sedangkan versi Python yang digunakan adalah Python 3.11.
4. Beberapa *syntax* dalam *library* gphoto2 tidak berjalan pada kamera DSLR Canon EOS 1300D. Peneliti sebelumnya menggunakan kamera DSLR Canon EOS 80D, sehingga memerlukan penyesuaian.

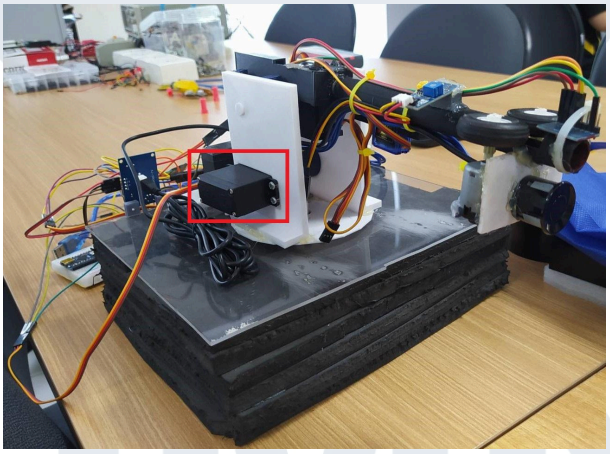
Hambatan yang dijumpai saat implementasi integrasi *mobile robot* antara lain sebagai berikut:

1. Pengoperasian *mobile robot* masih memerlukan proses yang cukup rumit karena memerlukan komputer/laptop yang terhubung dengan Raspberry Pi melalui *remote desktop*. Tujuan melakukan hal ini adalah untuk mengeksekusi atau me-*reset* program pada Raspberry Pi serta melihat indikator-indikator pada *terminal console* di Raspberry Pi.

3.2.3 Solusi yang Diterapkan

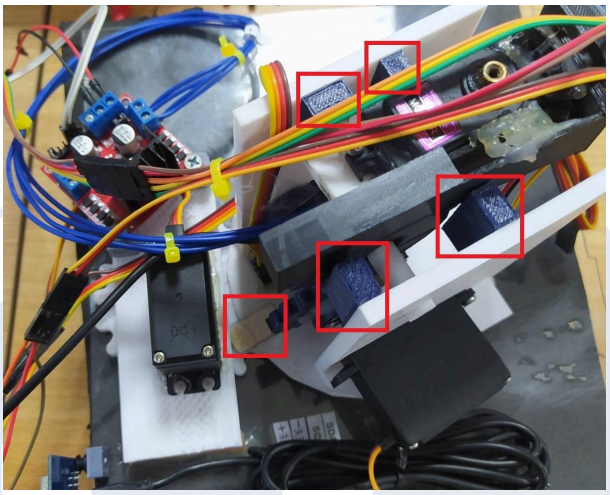
Solusi yang diterapkan untuk mengatasi hambatan perbaikan subsistem penembakan antara lain sebagai berikut:

1. Melakukan perbaikan pada bagian yang tidak dilem, seperti dengan penambahan struktur *support*.
2. Mengganti servo yang rusak dengan servo lain dengan tipe yang sama, yakni MG996R. Servo yang sudah diperbaiki ditampilkan pada Gambar 3.16.



Gambar 3.16 Servo elevasi meriam yang telah diganti

3. Pemasangan *support* untuk mengisi *gap* agar meriam tidak terlalu bergoyang. Sumbu *turret* juga ditambah *support* penahan agar meriam tidak jatuh tersungkur ke depan. Letak *support* dapat dilihat lebih jelas di Gambar 3.17.



Gambar 3.17 *Support* struktur meriam yang dipasang

4. Merapikan *wiring* dengan mengikatnya ke laras meriam, lalu semua konektor kabel diletakkan di belakang struktur meriam. *Wiring* juga disesuaikan agar bisa digunakan di atas *mobile robot*. Komponen *motor driver* dan PCB sensor ultrasonik disatukan dengan struktur meriam.

Solusi yang diterapkan untuk mengatasi hambatan implementasi subsistem pengenalan dan deteksi objek antara lain sebagai berikut:

1. Melakukan komputasi yang optimal adalah mengatur parameter *training* menjadi sedemikian rupa seperti konfigurasi yang telah dijelaskan sebelumnya. Alternatif solusi yang dapat digunakan untuk melakukan komputasi yang lebih berat dapat menggunakan layanan Google Colab dengan *dedicated memory* GPU sebesar 15 GB (perlu diperhatikan juga sisi negatif dari penggunaan layanan Google Colab secara gratis).
2. Kalibrasi parameter kamera DSLR Canon EOS 1300D dengan lensa 24 mm. Peneliti sebelumnya menggunakan kamera DSLR Canon EOS 80D dengan lensa 18-55 mm. Resolusi foto diubah menjadi 5184× 3456 px. Sudut FOV kamera DSLR diubah dengan lebar berada pada rentang sudut 59°-111° dan tinggi pada rentang sudut 78°-106° (asumsi 85° merupakan garis tengah servo *turret* dan 92° merupakan garis tengah servo meriam).

3. Menurunkan versi Python menjadi 3.10 serta meng-*install* ulang beberapa *library* yang telah digunakan pada Python 3.11.
4. Penyesuaian pada *syntax* untuk mengambil gambar. Gambar tangkapan kamera DSLR langsung disimpan di *folder* dalam Raspberry Pi 5, tidak disimpan dalam SD *card* kamera DSLR. *Syntax* di fungsi main juga diubah menggunakan *error handling* dikarenakan *library* gphoto2 sering mengalami *error* yang perlu di-*run* ulang.

Solusi yang diterapkan untuk mengatasi hambatan implementasi integrasi *mobile robot* antara lain sebagai berikut:

1. Menggunakan sakelar serta indikator LED sebagai penanda proses pada program Raspberry Pi. Solusi yang belum dilakukan adalah membuat Raspberry Pi otomatis langsung menjalankan program setelah dinyalakan.