TUGAS AKHIR – EE 184801

ROBOT PENGIKUT MANUSIA MENGGUNAKAN KAMERA TERMAL DAN *SINGLE BOARD COMPUTER* LATTEPANDA

Satrio Maulana Tsubasa
NRP. 07111745000031

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Muhammad Attamimi, B.Eng., M.Eng., PhD.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – EE 184801

ROBOT PENGIKUT MANUSIA MENGGUNAKAN KAMERA TERMAL DAN *SINGLE BOARD COMPUTER* LATTEPANDA

Satrio Maulana Tsubasa
NRP. 07111745000031

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Muhammad Attamimi, B.Eng., M.Eng., PhD.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT – EE 184801

HUMAN FOLLOWING ROBOT USING THERMAL CAMERA AND *SINGLE BOARD COMPUTER* LATTEPANDA

Satrio Maulana Tsubasa
NRP. 07111745000031

Advisor
Dr. Muhammad Rivai, ST., MT.
Muhammad Attamimi, B.Eng., M.Eng., PhD.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Robot Pengikut Manusia Menggunakan Kamera Termal dan *Single Board Computer* Lattepanda" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juni 2019

Satrio Maulana Tsubasa
NRP. 07111745000031

-----Halaman ini sengaja dikosongkan-----

ROBOT PENGIKUT PENGIKUT MANUSIA MENGGUNAKAN KAMERA TERMAL DAN *SINGLE BOARD COMPUTER* LATTEPANDA

TUGAS AKHIR

Diajukan untuk 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, S.T., M.T.	M. Attamimi, B.Eng., M.Eng., PhD.
NIP. 196904261994031003	NPP. 1985201711039

SURABAYA
JUNI, 2019

-----Halaman ini sengaja dikosongkan-----

ROBOT PENGIKUT PENGIKUT MANUSIA MENGGUNAKAN KAMERA TERMAL DAN *SINGLE BOARD COMPUTER* LATTEPANDA

Nama : Satrio Maulana Tsubasa
Pembimbing 1 : Dr. Muhammad Rivai, S.T., M.T.
Pembimbing 2 : Muhammad Attamimi, B.Eng., M.Eng., PhD.

ABSTRAK

Salah satu jenis robot untuk mempermudah pekerjaan manusia adalah robot pengikut manusia yang dapat dimanfaatkan sebagai robot pembawa barang. Pendeteksian target merupakan salah satu bagian yang sangat penting dari sistem robot pengikut manusia. Jika pendeteksian tidak berhasil dengan baik, maka robot pengikut manusia tidak akan dapat mengikuti targetnya dengan baik. Salah satu kendala dari pendeteksian target adalah kondisi pencahayaan. Salah satu solusi untuk menangani kendala tersebut adalah dengan menggunakan kamera termal yang dapat mendeteksi radiasi inframerah. Pada tugas akhir ini, proses pendeteksian target dilakukan pada *platform* robot *omni-directional wheels* menggunakan kamera termal AdaFruit AMG8833 dan *single board computer* Lattepanda. Citra yang diambil dari kamera termal berupa nilai suhu yang di konversi menjadi termografi dalam bentuk ruang warna RGB sehingga menggambarkan pemetaan suhu atau *thermography*. Thermography tersebut akan di proses menggunakan *image processing* pada *single board computer* lattepanda. Sistem ini mampu mendeteksi target dalam kondisi pencahayaan ruangan yang gelap maupun terang. Pada pengujian keseluruhan sistem yang telah dilakukan pada ruangan ber-AC dengan suhu 24^{o}C, target menggunakan celana pendek dan target berjalan dengan kecepatan 0.81 m/s, tingkat keberhasilan robot pengikut manusia menggunakan kamera termal dan *single board computer* lattepanda untuk mendeteksi dan mengikuti target seorang manusia yaitu 90%.

Kata Kunci: Kamera Termal, Lattepanda, *Omni-Directional Wheels Robot*, Pengolahan Citra, Robot Pengikut Manusia.

-----Halaman ini sengaja dikosongkan-----

HUMAN FOLLOWING ROBOT USING THERMAL CAMERA AND SINGLE BOARD COMPUTER LATTEPANDA

Name : **Satrio Maulana Tsubasa**
1st Advisor : **Dr. Muhammad Rivai, S.T., M.T.**
2nd Advisor : **Muhammad Attamimi, B.Eng., M.Eng., PhD.**

ABSTRACT

One type of robot to simplify human activity is a human following robot that can be used as a cargo robot. Target detection is one of the most important parts of a human following robot system. If target detection does not work well, human following robot will not able to follow the target perfectly. One obstacle of target detection is lighting condition. The solution for dealing with these obstacles is to use thermal camera that able to detect infrared radiation. In this thesis, the target detection process is carried out on an omni-directional robot platform using AdaFruit AMG8833 thermal camera and single board computer Lattepanda. The image taken from a thermal camera is in the form of a temperature values, then the values are converted into a thermography in RGB colorspace thus describes the temperature mapping. The thermography will be processed using image processing on the single board computer lattepanda. This system is able to detect the target under various lightining condition. In the overall system testing that has been done in air-conditioned room with room temperature is 24^{o}C, target using short pants and target walks at a speed of 0.81 m/s, the success rate of human following robot using thermal camera and single board computer lattepanda to detecting and following a human target is 90%.

Keywords: *Human Following Robot, Image Processing, Lattepanda, Omni-Directional Wheels Robots, Thermal Camera.*

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang selalu memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik. Penulis menyadari bahwa dalam menyelesaikan tugas akhir ini banyak mengalami kendala, namun berkat bantuan, bimbingan dan kerjasama dari berbagai pihak kendala tersebut dapat diatasi. Oleh karena itu, penulis ingin menyampaikan banyak terima kasih dan rasa hormat setinggi-tingginya kepada:

1. Bapak Dr. Muhammad Rivai, S.T., M.T. dan Bapak Muhammad Attamimi, B.Eng., M.Eng., PhD. selaku dosen pembimbing, yang telah memberikan bimbingan dan motivasi kepada penulis.
2. Bapak Totok Mujiono, Ir., M. Kom., Dr.Eng., Bapak Astria Nur Irfansyah, S.T., M.Eng., PhD. dan Bapak Tasripan, Ir., M.T. selaku dosen penguji yang memberikan banyak saran dan pengarahan kepada penulis.
3. Bapak Dr. Eng Ardyono Priyadi, S.T., M.Eng. selaku Ketua Departemen Teknik Elektro Fakultas Teknologi Elektro-ITS.
4. Ir. Riady Madyadinata dan Ir. Dianawati selaku orang tua dari penulis yang selalu memberikan do'a, nasihat serta memfasilitasi penulis dalam berbagai hal selama menjalani perkuliahan.
5. Putik Ramiacy Ulfami Lase, A.Md.A.K selaku kekasih dari penulis yang selalu memberikan motivasi dan dukungan.
6. Semua sahabat dan teman teman dari penulis yang senantiasa membantu dan mendukung penulis selama menjalani perkuliahan dan pengerjaan tugas akhir.

Pembuatan dan penyusunan tugas akhir ini masih jauh dari kesempurnaan, untuk itu penulis sangat mengharapkan kritik dan saran yang bersifat membangun sebagai bentuk perbaikan untuk tugas akhir ini kedepannya. Penulis berharap semoga tugas akhir ini bisa bermanfaat bagi penulis khususnya dan bagi pembaca pada umumnya.

Surabaya, Juni 2019

Satrio Maulana Tsubasa
NRP. 07111745000031

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

-----Halaman ini sengaja dikosongkan-----

DAFTAR GAMBAR

DAFTAR TABEL

-----Halaman ini sengaja dikosongkan-----

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tujuan robot yang paling umum adalah mempermudah pekerjaan manusia. Salah satu jenis robot untuk mempermudah pekerjaan manusia adalah robot pengikut manusia yang dapat dimanfaatkan sebagai robot pembawa barang. Pendeteksian target merupakan salah satu bagian yang sangat penting dan merupakan titik tolak kerja sistem robot pengikut manusia. Jika pendeteksian tidak berhasil dengan baik, maka robot pengikut manusia tidak akan dapat mengikuti targetnya dengan baik.

Pendeteksian target untuk robot pengikut manusia dapat menggunakan sebuah kamera yang akan menangkap warna pakaian yang digunakan oleh target. Warna pakaian tersebut akan dilakukan normalisasi untuk memperkuat jenis warna dari pengaruh pencahayaan yang berbeda [1]. Pendeteksian target tersebut menggunakan algoritma yang berbasiskan histogram warna dari pakaian yang digunakan oleh target. Dengan menggunakan histogram warna pada komponen R dan G dalam sistem RGB, sistem pendeteksian akan melakukan *learning* terhadap distribusi warna bagian pada target ketika input awal citra diberikan [2]. Metode pendetekesian target tersebut memiliki kelemahan jika lingkungan berada dalam kondisi gelap atau tidak ada cahaya, maka akan sulit untuk melakukan pendeteksian.

Radiasi inframerah merupakan salah satu alternatif untuk mengatasi kelemahan pendeteksian target yang masih bergantung pada cahaya. Radiasi inframerah mengandung informasi yang dapat mewakili kondisi atau karakteristik suatu benda. Informasi tersebut menunjukkan perbedaan pada suhu permukaan target. Implementasi radiasi inframerah pada robot pengikut manusia dapat menggunakan kamera termal sebagai pengganti kamera normal. Kamera termal dapat mendeteksi gelombang elektromagnetik yang berupa radiasi inframerah untuk mendapatkan gambar termal (*thermography*) [3].

Robot pengikut manusia membutuhkan proses pengolahan citra untuk mengolah hasil tangkapan kamera menjadi data yang dapat digunakan untuk menentukan arah pergerakan robot sehingga robot dapat mengikuti target yang telah di tentukan. Pada tugas akhir ini, proses pengolahan citra akan dilakukan menggunakan *single board computer* Lattepanda. Lattpanda memiliki beberapa kelebihan sebagai mini PC,

salah satunya adalah built-in Arduino dan support operating system windows 10. Sehingga Lattepanda dapat dimanfaatkan untuk *mobile* robot yang membutuhkan banyak I/O serta pengolahan citra.

Pada tugas akhir ini dilakukan kontrol pergerakan *omni-directional wheels robot* pada *platform* segitiga dengan tiga roda omni berdasarkan proses pengolahan citra yang diambil dari kamera termal infra merah 8x8 pixel. Robot ini akan mendeteksi suhu tubuh manusia yang menjadi target dengan kamera termal. Hasil tangkapan kamera termal akan diproses menggunakan pengolahan citra pada *single board computer lattepanda* menggunakan OpenCV dan Visual Studio 2017. Kemudian hasil dari pengolahan citra akan menentukan arah pergerakan robot.

1.2 Perumusan Masalah

Permasalahan yang akan dibahas dalam tugas akhir ini adalah:

1. Pendeteksian manusia pada saat kondisi gelap dan terang.
2. Metode robot untuk mengikuti target seorang manusia.
3. Jenis processor yang digunakan untuk pengolahan citra.

1.3 Tujuan

Tujuan pada Tugas Akhir ini dibagi menjadi dua:

1. Penggunaan kamera termal AMG8833 dengan resolusi 8x8 *pixel* pada robot pengikut manusia.
2. Implemetasi *image processing* menggunakan OpenCV.
3. Penggunaan *single board computer* lattepanda.

1.4 Batasan Masalah

Batasan masalah pada tugas akhir ini adalah:

1. Kamera termal yang digunakan adalah Adafruit AMG8833 8x8 *pixel*.
2. Robot mengikuti manusia dalam kondisi hanya ada satu orang manusia yang tertangkap oleh kamera.
3. Pengujian prototipe robot dilakukan di dalam ruangan.
4. Robot berupa prototipe sederhana dengan jarak jangkauan ±80 centimeter.

1.5 Metodologi Penelitian

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

1. **Studi Literatur**

   Studi literatur berisi serangkaian kegiatan pencarian dan pengkajian sumber-sumber yang relevan dan terpercaya yang menjadi acuan dalam penulisan tugas akhir ini. Literatur yang digunakan dititik-beratkan pada kinematika robot omni, *thermography*, *image processing*, dan *firmware* lattepanda.

2. **Pembuatan Desain Badan**

   Perencanaan yang di maksud bertujuan untuk membuat desain dasar robot serta catu daya untuk robot omni. Dasar pembuatan desain badan robot didasarkan pada beda sudut 120 derajat untuk jarak sudut setiap roda.

3. **Implementasi Desain dan Elektronik Robot**

   Implementasi desain badan robot direalisasikan pada bahan *acrylic* berwarna hitam dan pengaturan penempatan komponen komponen yang digunakan pada robot pengikut manusia yang meliputi 3 buah motor DC, *singleboard computer* lattepanda, 2 buah DC *buck converter*, 2 buah modul motor *driver* L298N, Arduino Nano, AMG8833 8x8 *pixel thermal camera*, PCB *connector* serta catu daya yang berupa 8 buah *battery* Sony tipe US18650 VTC6 3100mAh 30A Li-Ion.

4. **Pengujian Deteksi Citra Obyek oleh Robot**

   Pengujian deteksi citra obyek dilakukan untuk mengetahui parameter (kondisi) terbaik untuk sistem. Parameter terbaik didapatkan dari pengujian berdasarkan jarak yang berbeda antara robot dengan target, suhu ruangan yang berbeda dan kondisi pencahayaan yang berbeda.

5. **Pengujian Kinematika Robot Omni**

   Pergerakan robot omni ditentukan oleh kinematika robot omni baik secara *forward* maupun *invers.* Pegujian awal yang dilakukan meliputi pengujian respon kecepatan tiap-tiap motor dc untuk beberapa *duty cycle.* Pengujian kombinasi kecepatan tiap-tiap motor dc untuk beberapa sudut yang diujikan.

6. **Tahap Pengujian Sistem Keseluruhan**

   Pengujian ini dimaksudkan untuk memastikan bahwa kinerja dari hasil perealisasian sistem dapat berfungsi sesuai dengan yang diharapkan.

7. **Evaluasi**

   Tahap evaluasi dan penyempurnaan kinerja sistem dilakukan setelah pengujian dilakukan sebelumnya. Pada tahap ini akan dinilai sistem kerja dari alat, baik dari metode, kondisi jarak dan pencahayaan yang paling sesuai.

8. **Penyusunan Laporan Akhir**

   Penyusunan laporan dilakukan setelah semua tahap terselesaikan sehingga hasil yang diperoleh dari pembuatan alat dapat dijelaskan secara rinci sesuai dengan data yang diperoleh.

9. **Penulisan Makalah Jurnal POMITS**

   Penulisan jurnal dilakukan sebagai sarana publikasi penelitian. Hal ini juga dilakukan agar penelitian mendapatkan legalitas dan pengakuan resmi dari dunia pendidikan.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

- Bab I : Pendahuluan
  Bab ini meliputi latar belakang penelitian, perumusan masalah, tujuan penelitian, batasan masalah, metodologi, sistematika penulisan penelitian, dan relevansi.
- Bab II: Dasar Teori
  Bab ini menjelaskan tentang dasar teori yang dibutuhkan dalam pengerjaan penelitian ini meliputi dasar teori dari *black* body, infra merah, pengolahan citra, kamera termal, lattepanda, dan kinematika
- Bab III: Perancangan Alat
  Bab ini menjelaskan perencanaan dari sistem yang meliputi hardware, mekanik, dan software untuk mencapai tujuan yang diharapkan dari penelitian ini.
- Bab IV: Pengujian Alat
  Bab ini menjelaskan tentang hasil yang didapat dari sistem serta hasil evaluasi dari sistem tersebut.
- Bab V: Penutup
  Bab ini menjelaskan tentang kesimpulan yang meliputi kekurangan-kekurangan pada kerja alat dari hasil yang telah didapatkan serta saran pengembangan untuk ke depan.

1.7 Relevansi

Omni-directional wheels robot merupakan salah satu jenis *mobile robot* yang banyak dikembangkan. Penggunaan termal kamera dapat dimanfaatkan kedalam berbagai macam aplikasi. Di masa depan, *mobile robot* dengan tipe *omni-directional wheels* dan kamera termal dapat diterapkan pada fungsi yang lebih luas.

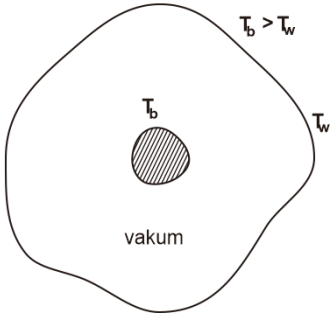
-----Halaman ini sengaja dikosongkan-----

BAB II
TEORI PENUNJANG

2.1 *Blackbody Radiation*

Setiap objek memancarkan radiasi sebagai hasil dari temperatur objek tersebut. Istilah ini pertama kali di kemukakan oleh fisikawan berkebangsaan Jerman bernama Gustav Robert Kirchoff (1824-1887). *Blackbody* didefinisikan sebagai *body* atau objek yang mampu menyerap radiasi termal yang datang pada objek tersebut [4]. Dalam literatur lain, dikatakan bahwa sebuah *blackbody* mampu menyerap (*absorbing*) sekaligus memancarkan (*emitting*) radiasi dalam jumlah yang sama besar [5].

Untuk ilustrasi yang ditunjukkan pada gambar 2.1, sebuah *blackbody* berukuran kecil dengan temperatur Tb diletakkan di dalam sebuah wadah tertutup dengan temperatur di sekeliling wadah adalah Tw dan Tb > Tw. Transfer panas berupa konduksi dan konveksi dianggap tidak terjadi karena *blackbody* dan wadah disusun sedemikian sehingga blackbody tidak menyentuh dinding wadah dan ruang wadah merupakan ruang hampa (*vacuum*).



Gambar 2. 1. Ilustrasi *blackbody* sebagai pemancar radiasi.

Setelah beberapa saat, *blackbody* akan mencapai kesetimbangan termal dengan temperatur lingkungan. Objek *blackbody* di atas menyerap dan memancarkan radiasi dalam jumlah yang sama. Jika jumlah penyerapan dan emisi berbeda, akan terjadi laju net dari perubahan entalpi; sebagai konsekuensinya *blackbody* mengalami perubahan

temperature (naik-turun) pada saat tercapai kesetimbangan termal yang bertentangan dengan hukum kedua termodinamika. Dengan demikian, *blackbody* harus menyerap dan memancarkan radiasi dengan jumlah yang sama [5].

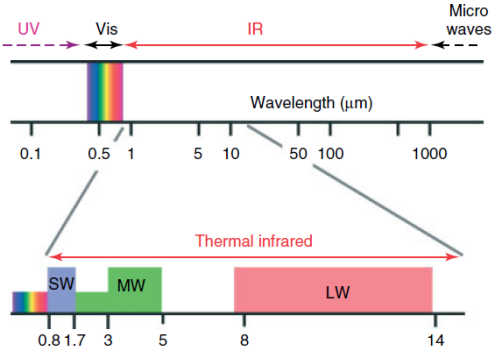
Radiasi tidak tergantung pada ukuran dan bentuk. Jika dua buah *blackbody*, ukuran dan bentuk yang berbeda, masing-masing memiliki temperatur Tb dan temperatur Tw lingkungan dan dengan menganggap bahwa tidak terjadi konduksi dan konveksi, maka kedua *blackbody* suatu saat akan memiliki temperatur yang sama dengan Tw. Selama tertutup rapat, *blackbody* akan terus memancarkan radiasi yang sama (*isotropic*), dan hanya tergantung pada temperatur dari *blackbody* tersebut [6].

Blackbody radiation mula-mula diajukan oleh Josef Stefan dan Ludwig Boltzmann pada akhir abad ke-19 yang menyatakan bahwa radiasi yang diemisikan oleh *blackbody* sebanding dengan pangkat empat dari temperatur badan.

2.2 Gelombang Inframerah

Inframerah adalah gelombang elektromagnetik dengan rentang panjang gelombang 0.7 – 1000 μm. Inframerah disebut juga dengan cahaya inframerah. Panjang gelombang cahaya inframerah lebih besar dari panjang gelombang cahaya tampak, sehingga meskipun disebut sebagai "cahaya", inframerah tidak bisa terlihat oleh mata telanjang. Cahaya inframerah memiliki sifat yang sama dengan cahaya tampak seperti dapat difokuskan, dipantulkan dan juga dapat dipolarisasikan [7].

Dalam ilmu termografi, cahaya inframerah yang digunakan adalah cahaya dengan rentang panjang gelombang 0.8 – 14 μm. Rentang ini dibedakan menjadi tiga jenis. Jenis pertama disebut shortwave (SW) dengan rentang 0.8 – 1.7 μm, jenis kedua disebut midwave (MW) dengan rentang 3 – 5 μm dan jenis ketiga adalah longwave (LW) dengan rentang 7 – 14 μm. Kamera digital biasa, peka terhadap ketiga jenis inframerah tersebut [8]. Rentang rentang panjang gelombang tersebut dibuat berdasarkan pertimbangan jumlah radiasi panas yang dibutuhkan, bentuk dari detektor, dan sifat hantar udara. Gambar 2.2 menunjukkan rentang jenis panjang gelombang inframerah.

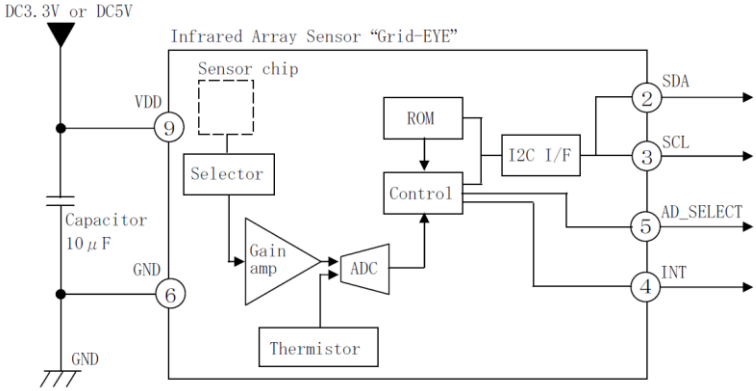


Gambar 2. 2. Area dan batas spektral dari gelombang inframerah [7].

Sensor pada kamera digital sangat peka pada rentang gelombang 0.7 – 1 μm. Rentang ini sering disebut sebagai near infrared (NIR), artinya infrared pada rentang ini dekat dengan rentang gelombang cahaya tampak [7].

2.3 Adafruit AMG 8833 8 x 8 *Pixel Thermal Camera*

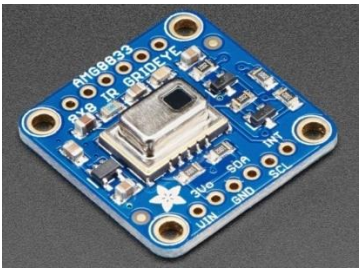
Kamera termal 8x8 piksel adafruit AMG 8833 adalah sebuah sensor *array* 8x8 infra merah berupa kamera yang mendeteksi suhu. Kamera ini akan memberikan nilai *array* dari 64 piksel pembacaan suhu melalui inframerah melalui I2C. Kamera termal ini akan mengukur suhu mulai dari 0 ° C hingga 80 ° C (32 ° F hingga 176 ° F) dengan akurasi + - 2,5 ° C (4,5 ° F) dan dapat mendeteksi manusia dengan jarak hingga 7 meter. Kamera thermal ini juga memiliki *frame rate* maksimum 10Hz. Pada gambar 2.3 ditunjukkan skema rangkaian dari AMG 8833, dan pada gambar 2.4 adalah bentuk fisik dari perangkat Adafruit AMG 8833 thermal camera. Spesifikasi fungsi utama dari kamera termal ini ditunjukkan pada tabel 2.1.



Gambar 2. 3. Blok Diagram AMG8833.

Tabel 2. 1. Spesifikasi Fungsi Utama Adafruit AMG8833.

Jumlah Piksel	64 (8x8 Matriks)
External Interface	I2C (*Fast Mode*)
Frame Rate	10 frames/sec atau 1 frame/sec
Mode Operasi	Normal / *Sleep* / *Stand-by*
Mode Output	Temperatur Output
Calculate Mode	*No moving average or Twice moving average*
Resolusi Temperatur Output	0.25°C
Jumlah Alamat Sensor	2（I2C Slave Address）
Skala Temperatur Output Termistor	-20°C～80°C
Resolusi Output Termistor	0.0625℃



Gambar 2. 4. Adafruit AMG8833 8x8 piksel *Thermal Camera.*

2.4 *Single Board Computer* Lattepanda

Single board computer adalah sebuah komputer lengkap yang dibuat pada papan *circuit* tunggal. Didalamnya berisi mikroprosesor, memori, input/output (I/O) dan fitur-fitur lain yang dibutuhkan untuk fungsionalitas computer. *Single board computer* dibuat sebagai *platform* pengembangan sistem, untuk pendidikan, atau digunakan untuk kontroler komputer yang tertanam. *Single board computer* didesain sederhana dan umumnya menggunakan RAM statis dengan 8 bit atau 16 bit prosesor.

Lattepanda adalah mini pc yang telah diintegrasikan dengan mikrokontroller Arduino didalamnya. Mini pc ini mampu menjalankan OS linux dan Windows 10 yang telah terdapat Visual Studio, NodeJS, Java, Processing didalamnya. Mini pc ini diperkuat dengan Intel Cherry Trail Z8300 Quad Core 1.8 GHz, 4 GB DDR3L RAM, 64 GB memori internal, dan prosesor ATmega32u4 yang berfungsi sebagai Arduino Leonardo. Lampiran A menunjukkan *basic* diagram Lattepanda.

Arduino Leonardo merupakan sebuah board mikrokontroller berbasis atmega32u4. Dengan tambahan WiFi dan Bluetooth 4.0 pada lattepanda sebagai sarana komunikasi. Arduino Leonardo memiliki 20 digital pin input/output dimana 7 pin tersebut dapat digunakan sebagai output PWM dan 12 pin sebagai input analog. Perbedaan Arduino Leonardo dengan Arduino lainnya yaitu ATmega32u4 secara terintegrasi telah memiliki komunikasi USB, sehingga tidak lagi membutuhkan prosesor sekunder sebagai converter USB-to-serial. Single board computer Lattepanda dapat dilihat pada gambar 2.5.

Gambar 2. 5. Single Board Computer Lattepanda.

Tabel 2.2 dibawah ini menunjukkan spesifikasi lengkap dari *single board computer* Lattepanda.

Tabel 2. 2. Spesifikasi Lattepanda 4GB / 64GB.

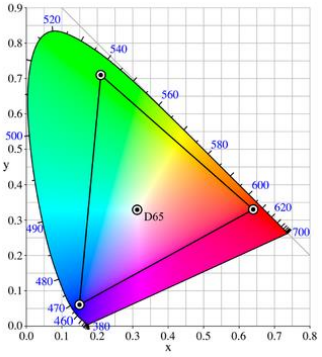
Processor	Intel Cherry Trail Z8350 Quad Core 1.8GHz
Operation System	*Pre-installed full edition of* Windows 10
RAM	4GB DDR3L
Storage Capability	64GB
GPU	Intel HD Graphics, 12 EUs @200-500 Mhz, *single-channel memory*
USB *Port*	1 x USB3.0 *port* dan 2 x USB 2.0 *ports*
Connectivity	WiFi and Bluetooth 4.0
Built-in Arduino Co-processor	ATmega32u4
Power	5V/2A
Dimension of board	88 x 70 mm
Packing Size	110 x 94 x 30 mm
GPIO	6 GPIO Cherry Trail *processor*
	20 GPIO Arduino Leonardo
	6 *Plug and play Gravity* sensor *connectors*

2.5 OpenCV

OpenCV (Open source Computer Vision) merupakan sebuah pustaka (library) fungsi programming untuk pengolahan citra [9]. Pustaka OpenCV dapat digunakan dengan beberapa bahasa pemgrograman seperti C++ dan Python untuk melakukan pengambilan, pengolahan, serta menampilkan citra baik berupa citra diam (*still image*) maupun citra bergerak (*moving image*). Pustaka OpenCV bersifat open srouce sehingga memungkinkan penggunaan baik untuk keperluan akademik maupun komersil. Pustaka OpenCV dapat berjalan pada beberapa *Operating System* seperti Windows, Linux, Android, dan MacOs.

2.6 *Color Space*

Warna adalah hasil visual dari spektrum cahaya baik itu ditransmisikan lewat media transparan, diserap, dan dipantulkan oleh sebuah permukaan. Warna adalah panjang gelombang cahaya yang dapat diterima dan diproses oleh mata manusia dengan nilai antara 400nm sampai dengan 700nm. *Color space* atau ruang warna adalah sebuah metode yang bisa dispesifikasikan dan divisualkan warnanya ke dalam representasi data. *Color space* adalah penjabaran warna menjadi komponen-komponen terpisah yang membangun warna itu sendiri. Ruang warna RGB menjabarkan suatu warna tertentu menjadi kombinasi antara *red*, *green*, dan *blue*, sedangkan ruang warna HSV menyatakan warna tertentu ke dalam *hue*, *saturation*, dan *value*. Semua representasi warna tersebut digunakan pada kondisi tertentu tergantung kebutuhan. Pada umumnya, ruang warna direpresentasikan sebagai data representasi 3-D seperti gambar 2.6.



Gambar 2. 6. Ruang Warna RGB [10].

Seperti digambarkan diatas, pada ruang warna RGB, *warna red*, *green*, *blue* dipetakan pada sistem koordinat kartesian 3-D yang membentuk kubus 3-D. Titik-titik kubus adalah warna primer (*red*, *green*, dan *blue*) dan warna sekunder (*cyan*, *yellow*, dan *magenta*). Origin dari koordinat adalah *black* dimana komponen warna red, green, dan blue semuanya bernilai 0. Pojok kebalikan kubus secara diagonal adalah *white*, dimana nilai komponen RGB bernilai maksimum.

Cahaya yang sampai pada mata manusia adalah sebuah fungsi dari pantulan permukaan, warna cerah, dan *lighting geometry*. Warna warna yang kita terima bergantung pada hampir sebagian besar pada pantulan permukaan. Ketergantungan karena lighting geometry dan warna cerah dihilangkan melalui sebuah langkah normalisasi citra [11].

2.7 *Color Maps*

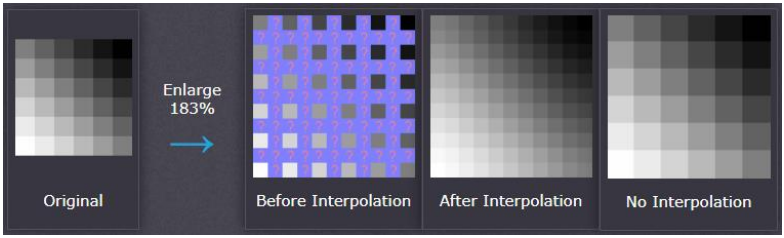
Color map atau pemetaan warna adalah fungsi yang mengubah warna dari suatu gambar ke warna lain. Pemetaan warna dapat disebut sebagai algoritma yang menghasilkan fungsi pemetaan atau algoritma yang mengubah nilai warna gambar. Pengamatan oleh mata manusia tidak sensitif apabila mengamati perubahan halus pada gambar dengan warna abu-abu *(grayscale)*. Mata manusia lebih sensitif untuk mengamati perubahan antar warna yang lebih bervariasi, sehingga dibutuhkan pemetaan warna abu-abu menjadi bentuk warna yang lain.

Pada gambar 2.7 ditunjukkan contoh jenis pemetaan warna yang digunakan pada OpenCV. Pada tugas akhir ini, digunakan *color map* Jet untuk memetakan warna suhu yang didapat dari kamera termal.

Gambar 2. 7. Contoh *Color Mapping*.

2.8 Interpolasi Gambar

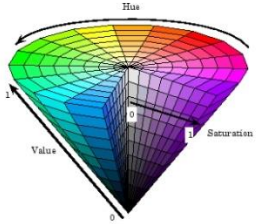
Interpolasi gambar merupakan sebuah metode untuk mengubah ukuran sebuah gambar. Mengubah ukuran gambar diperlukan ketika perlu menambah atau mengurangi jumlah piksel. Interpolasi gambar berfungsi dalam dua arah yaitu sumbu X dan sumbu Y. Interpolasi gambar akan mendapatkan hasil prediksi terbaik dari warna dan intensitas sebuah piksel berdasarkan nilai piksel di sekitarnya. Gambar 2.8 menggambarkan contoh cara kerja dari interpolasi gambar:



Gambar 2. 8. Contoh Interpolasi.

2.9 HSV *Color Space*

Ruang warna HSV terdiri dari tiga bagian yaitu *Hue*, *Saturation* (*Chroma*), dan *Value*. *Hue* adalah istilah yang digunakan untuk menyatakan warna dasar sebenarnya seperti biru, hijau, dan merah. Hue dinyatakan dalam derajat (degree) bervariasi antara 0 sampai 360. *Saturation* adalah nilai warna putih ke warna murni. *Value* menunjukkan kecerahan dari warna. *Saturation* dan *Value* bervariasi antara 0 sampai 1. Dibawah ini adalah gambar ilustrasi dan formula dari HSV.



Gambar 2. 9. HSV *Diagram* [12].

Warna didefinisikan oleh (R, G, B) pada ruang warna RGB. Untuk menentukan nilai HSV dari RGB, digunakan rumus berikut:

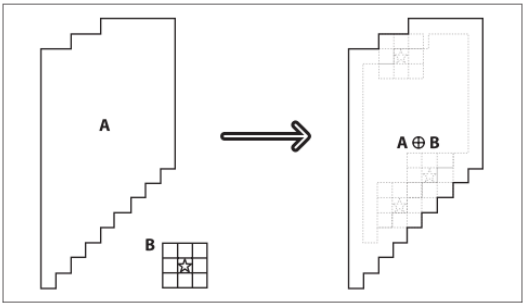
$$H = \begin{cases} \frac{G-B}{Max-Min} x60, if\ R = Max \\ 120 + \frac{B-R}{Max-Min} x60, if\ G = Max \\ 240 + \frac{R-G}{Max-Min} x60, if\ B = Max \end{cases} \quad (2.1)$$

$$S = \frac{Max-Min}{Max}; V = Max \quad (2.2)$$

Max merupakan nilai maksimum dari RGB. Min merupakan nilai minimum dari RGB.

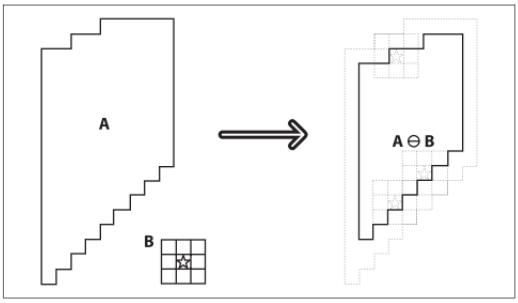
2.10 *Image Morphology*

Image morphology terdiri dari dilasi dan erosi. Dilasi adalah sebuah konvolusi citra A dengan suatu kernel B. Kernel dapat berbentuk apapun dan memiliki *anchor point* tunggal yang ditentukan. Pada umumnya, kernel berupa kotak solid kecil atau lonjong dengan *anchor point* di tengahnya. Kernel bisa dianggap sebagai *template* atau *mask* dan efek kernel untuk dilasi adalah *local maximum operator*. Ketika kernel B di-scan terhadap suatu citra, akan dihitung nilai piksel maksimal yang dilewati oleh kernel B dan mengganti piksel citra di bawah *anchor point* dengan nilai maksimal tersebut. Hal ini menyebabkan bagian putih pada citra akan bertambah seperti pada gambar 2.10. Pertambahan ini adalah asal muasal istilah operator dilasi [13].



Gambar 2. 10. Dilasi: nilai maksimum di bawah kernel B.

Erosi adalah operasi kebalikan dilasi. Aksi operator erosi adalah menghitung *local minimum* dari kernel. Erosi menghasilkan citra baru dari citra asli dengan algoritma: saat kernel B di-*scan* terhadap citra, akan dihitung nilai piksel minimal yang dilewati kernel B dan mengganti piksel citra di bawah *anchor point* dengan nilai minimal tersebut [13]. Proses erosi dijelaskan pada gambar 2.11.



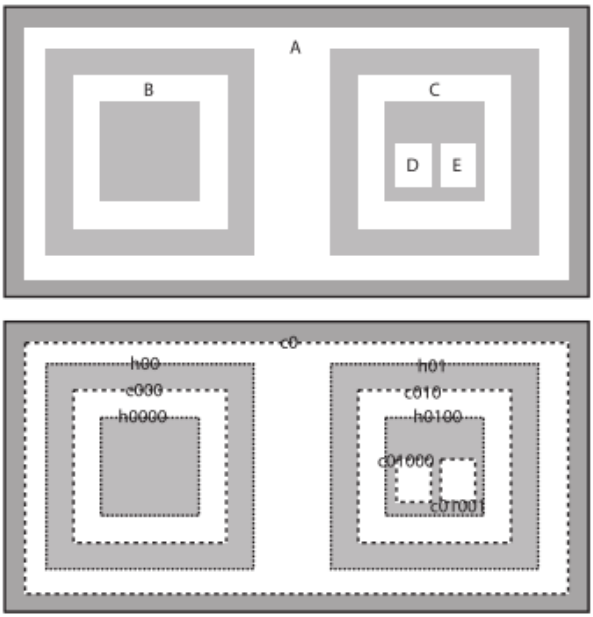
Gambar 2. 11. Erosi: nilai minimum di bawah kernel B.

Secara umum, dilasi menambah region dan erosi mengurangi region. Dilasi akan cenderung menghaluskan cekungan, sementara erosi menghaluskan tonjolan. Tentunya hasil akhir bergantung pada tipe kernel yang digunakan [13].

2.11 *Find Contours*

Sebuah kontur adalah sebuah daftar titik-titik yang merepresentasikan sebuah kurva dalam suatu citra. Kontur direpresentasikan oleh OpenCv berupa sequences yang tiap *sequence* berisi informasi lokasi titik selanjutnya pada kurva. Fungsi cvFindContours() menghitung kontur-kontur dari citra biner yang bisa berasal dari cvCanny() yang memiliki piksel-piksel edge atau citra yang dihasilkan dari fungsi cvThreshold() atau cvAdaptiveThreshold() dimana edge adalah batas antara daerah positif dan negative [13].

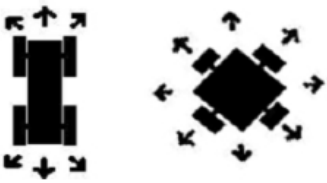
Gambar 2.12 bagian atas menunjukkan bagian citra dengan daerah putih dengan label A hingga E pada *background* gelap. Gambar 2.12 bagian bawah menunjukkan gambar yang sama dengan kontur-kontur yang akan dihitung oleh cvFindContours(). Kontur-kontur diberi label cX dan hX, dimana c untuk kontur, h untuk *hole*, dan X adalah angka. Beberapa kontur berupa garis putus-putus yang merepresentasikan batas luar daerah putih.



Gambar 2. 12. (Atas) Citra tes untuk dilakukan cvFindContour: (bawah) kontur (garis putus) dan *hole* (garis titik).

2.12 *Omni-Directional Wheels Robot*

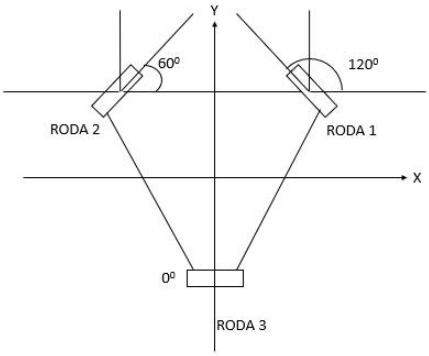
Omni-Directional Wheels Robot adalah sebuah rancang bangun *holonomic mobile robot* yang menggunakan roda omni (*omni wheels*) sehingga bisa bergerak ke segala arah di sudut manapun tanpa memutar badan terlebih dahulu. Beberapa *omni-directional wheels robot* menggunakan *platform* segitiga dan persegi empat. Pada *platform* segitiga, robot omni menggunakan tiga buah roda omni yang terpisah sejauh 120° dan pada *platform* persegi empat, robot omni menggunakan empat buah roda omni [14]. Perbedaan *holonomic* dan *non-holonomic mobile robot* gambar 2.13.



Gambar 2. 13. *Non-Holonomic* & *Holonomic Mobile Robot* [15].

Dua hal mendasar dalam perancangan robot omni yaitu roda omni yang salah satu contohnya ditunjukkan pada gambar 2.14 dan kinematika gerak dari roda omni.

Gambar 2. 14. *Omni Wheels.*



Gambar 2. 15. Robot omni dengan platform segitiga.

Kinematika gerak roda omni didapatkan dengan menentukan komponen kecepatan dengan acuan sumbu x dan y robot. Untuk menentukan sumbu x dan y robot, maka ditentukan satu posisi roda sebagai acuan sumbu x dan y robot. Gambar 2.15 menjelaskan sumbu x dan y robot. Dari gambar tersebut, dapat diturunkan kecepatan tiap-tiap roda terhadap sumbu x dan y robot sehingga bisa didapatkan Vx dan Vy robot. Penurunan kecepatan tiap roda dapat dijelaskan sebagai berikut:

Untuk roda 1, didapatkan komponen kecepatan sebagai:

$$V_{x1}=V_{motor}cos(angle/wheel\ radius) \quad (2.3)$$

$$V_{y1}=V_{motor}sin(angle/wheel\ radius) \quad (2.4)$$

$$V_1=V_{x1}cos(\frac{120+\alpha}{wheel\ radius}) + V_{y1}sin(\frac{120+\alpha}{wheel\ radius}) + R\dot{\theta} \quad (2.5)$$

Untuk roda 2, didapatkan komponen kecepatan sebagai:

$$V_{x2}=V_{motor}cos(angle/wheel\ radius) \quad (2.6)$$

$$V_{y2}=V_{motor}sin(angle/wheel\ radius) \quad (2.7)$$

$$V_2=V_{x2}cos(\frac{60+\alpha}{wheel\ radius}) + V_{y2}sin(\frac{60+\alpha}{wheel\ radius}) + R\dot{\theta} \quad (2.8)$$

Untuk roda 3, didapatkan komponen kecepatan sebagai:

$$V_{x3}=V_{motor}\ cos(angle/wheel\ radius) \quad (2.9)$$

$$V_{y3}=V_{motor}sin(angle/wheel\ radius) \quad (2.10)$$

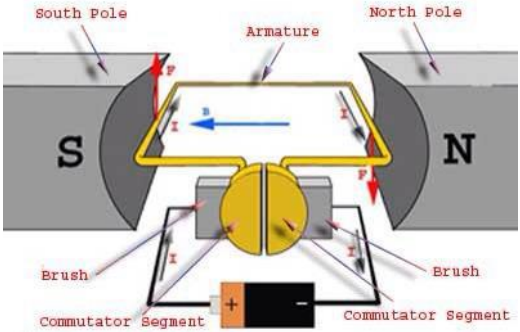
$$V_3=V_{x3}cos(\frac{0+\alpha}{wheel\ radius}) + V_{y3}sin(\frac{0+\alpha}{wheel\ radius}) + R\dot{\theta} \quad (2.11)$$

2.13 Motor DC

Motor DC merupakan motor elektrik yang digerakkan dengan menggunakan tegangan dengan arus searah atau DC dimana mengubah energy listrik menjadi energy kinetik. Motor DC memiliki dua komponen utama yaitu stator, bagian yang tidak berputar yang terdiri dari kumparan magnet dan rangka. Bagian lain dari motor DC yaitu rotor, bagian yang berputar terdiri dari jangkar [16].

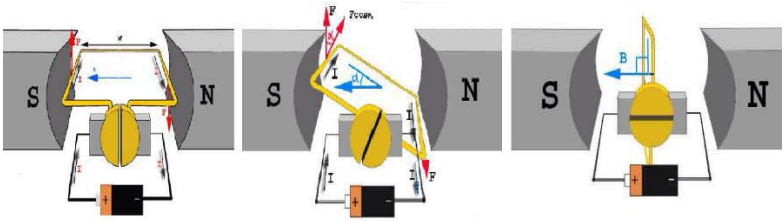
Pada gambar 2.19, rotor pada sebuah motor memiliki kawat armatur penghantar listrik yang berbentuk persegi panjang. Pada kedua ujung kawat tersebut terpasang sebuah komutator berbentuk lingkaran yang tengahnya terbelah atau disebut cincin belah, cincin belah ini ikut berputar bersama rotor. Stator motor tersusun atas dua magnet dengan kutub utara dan selatan yang saling berhadapan. Pada gambar diatas,

masing-masing kutub baterai terhubung dengan sikat karbon, sehingga tercipta arus DC yang bergerak dari kutub positif ke negative melalui sikat karbon.



Gambar 2. 16. Struktur dari motor DC.

Pada gambar 2.20 sisi kiri, garis medan magnet mengarah ke kiri yang ditandai dengan garis biru dan huruf B. Arus listrik ditandai dengan garis berwarna hitam dan huruf I. Dengan menggunakan kaidah tangan kiri, maka gaya dorong F akan mengarah keatas. Gaya dorong yang tegak lurus lagsung terhadap kawat armatur kanan dan kiri ini menghasilkan torsi yang paling besar pada rotor motor, torsi ini yang menyebabkan motor berputar.



Gambar 2. 17. Arah putar motor DC.

Pada gambar 2.20 tengah, cincin belah terhubung dengan sikat karbon sehingga arah dari arus listrik tetap. Dengan kaidah tangan kiri, arah gaya dorong mengarah ke atas, torsi yang terjadi yaitu lebih kecil sebesar cos a daripada gaya F. Torsi motor akan bernilai nol pada saat posisi kawat tegak seperti pada gambar 2.20 sisi kanan. Dengan kaidah tangan kiri, jika kawat armatur terdapat arus listrik, maka arah dari gaya

dorong kawat akan mengarah ke atas atau ke arah bawah. Sudut a sebesar 90 derajat menandakan tidak ada gaya torsi, karena nilai dari cos 90 bernilai nol. Nilai torsi nol tidak akan membuat motor berhenti berputar dikarenakan sifat daripada kelembaman rotor maka motor akan terus berputar selama terdapat arus yang mengalir melalui kawat armatur. Kelebihan dari penggunaan motor DC yaitu mudah dalam hal pengaturan kecepatan dan juga sederhana dalam pengaturannya. Selain itu dalam hal konsumsi daya motor DC merupakan salah satu motor yang memiliki penggunaan daya yang cukup kecil.

2.14 Kontrol PID

PID Kontrol atau proportional-integral-derivative kontroller adalah sebuah kontroler yang berguna untuk menentukan kepresisian atau keakuratan suatu sistem instrumentasi dengan karakteristik umpan balik atau feedback. PID digunakan cukup sering pada sistem kontrol industri. PID kontroller terdiri dari 3 jenis, yaitu proportional, integrative, dan derivative. Ketiganya dapat digunakan secara bersamaan.

Pada PID kontroller sendiri terdapat 3 macam yaitu kontrol PI, PD, dan PID. Kontroller PI merupakan kontrol yang menggunakan komponen P dan I, sedangkan PD menggunakan komponen P dan D, dan terakhir PID menggunakan ketiga komponen yaitu P, I, dan D. Kontrol Kp akan memberikan efek mengurangi waktu naik tetapi tidak memberi perubahan pada setting time. Kontrol Ki akan memberikan efek hilangnya steayd state error. Kontrol Kd akan memberikan efek meningkatnya stabilitas sistem dan mengurangi overshoot dan setting time pada sistem [17]. Efek dari setiap pengendali dalam sistem lingkar tertutup dapat dilihat pada tabel 2.3

Tabel 2. 3. Tabel Pengaruh P, I dan D terhadap sistem.

Closed Loop	Rise Time	Overshoot	Settling Time	Steady State Error
Kp	Berkurang	Meningkat	Perubahan Kecil	Berkurang
Ki	Berkurang	Meningkat	Meningkat	Hilang
Kd	Perubahan Kecil	Berkurang	Berkurang	Perubahan Kecil

Dari Tabel 2.3 dapat diketahui bahwa pengendali proporsional akan mengurangi waktu naik, meningkatkan persentase lewatan

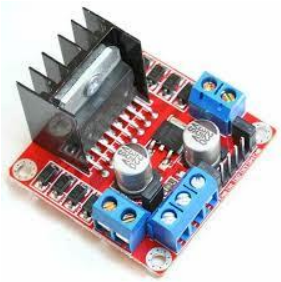
maksimum dan mengurangi keadaan tunak. Sedangkan pengendali proporsional derivatif mereduksi lewatan maksimum dan waktu turun. Selain itu, pengendali proporsional integral menurun pada waktu naik, meningkatkan lewatan maksimum dan waktu turun dan akan menghilangkan kesalahan keadaan. Karena masing-masing mempunyai kelebihan, untuk menentukan besar nilai Kp, Ki, dan Kd dapat menggunakan tuning secara manual untuk mendapatkan nilai yang sesuai [18].

PID Kontrol menghasilkan sinyal keluaran untuk mengatur aksi dari suatu plant berdasarkan spesifikasi performa [19]. Kontroller akan menerima input, dimana nilainya berbeda dengan error (e) antara nilai yang dinginkan (set point / r) dan nilai hasil ukur yang actual (y) didukung oleh sinyal umpanbalik dari suatu divais yang ingin dikontrol. Kontrol PID menghasikan output (u), yang mana merupakan proportional, integral, maupun derivative dari error, dapat dilihat pada gambar dibawah ini. Apabila menggunakan sebuah kontrol proporsional, kontrol tersebut akan melakukan koreksi pada variable terkontrol yang sebanding dengan perbedaan nilai yang diinginkan dan nilai yang terukur [20]. Output dari control proporsional adalah produk perkalian dari sinyal error dan nilai gain proporsional.

2.15 L298 *Motor Driver*

L298 aadalah driver motor berbasis H-Bridge, mampu menangani beban hingga 4A pada tegangan 6V – 46V. Dalam chip L298 terdapat dua rangkaian H-Bridge. Selain itu driver ini mampu mengendalikan 2 motor secara bersamaan dengan arus beban 2A. Kelebihan lainnya dari modul L298 *motor driver* adalah lebih presisi dalam mengontrol motor dan dilengkapi dengan heatsink sehingga lebih tahan panas. Gambar 2.17 menunjukkan modul driver motor L298.

Untuk pemasangan driver L298N ini dibutuhkan 6 buah pin mikrokontroler. 2 buah untuk pin Enable (masing-masing satu untuk tiap motor DC) dan 4 buah pin untuk mengatur kecepatan motor DC tersebut. Pada prinsipnya rangkaian driver motor L298 ini dapat mengatur tegangan dan arus sehingga kecepatan dan arah motor dapat diatur. Tabel 2.3 menunjukkan spesifikasi lengkap dari driver motor L298.



Gambar 2. 18. Modul L298 *Motor Driver*.

Tabel 2. 4. Spesifikasi Modul Driver L298.

Tipe Driver	*Dual H-Bridge*
Chip Control	ST L298N
Logic voltage	5V DC
Drive voltage	5V DC – 35V DC
Logic current	0 mA – 36 mA
Drive current	2A (maximum on single bridge)
Batas Temperatur	-20 º C sampai 135 º C
Power Maksimum	25W
Berat	30g
Dimensi	43 mm x 43 mm x 27 mm

2.16 Arduino Nano

Arduino merupakan board sistem minimum mikrokontroler yang mempunyai sifat open source. Board Arduino ini menggunakan IC mikrokontroler AVR yang merupakan produk dari Atmel. Arduino Nano menggunakan IC mikrokontroler ATmega 328 (Arduino Nano 3.x) atau ATmega 168 (Arduino Nano 2.x). Selain bersifat open source Arduino juga memiliki bahasa pemprograman sendiri berupa bahasa C. Arduino Nano memiliki DC power jack, port USB Mini-B yang digunakan untuk upload source code program ke dalam mikrokontroler. Gambar 2.18 menunjukkan modul arduino nano. Spesifikasi lengkap dari arduino nano ditunjukkan pada table 2.4.

Gambar 2. 19. Arduino Nano.

Tabel 2. 5. Spesifikasi Arduino Nano (ATmega328).

Chip	ATmega328
Operating Voltage (logic level)	5V DC
Input Voltage (Recommended)	7V DC - 12V DC
Input Voltage (limits)	6V DC - 20V DC
Digital I/O Pins	14 (6 *provide PWM Output*)
Analog Input Pins	8
DC *Current* per I/O Pin	40 mA
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

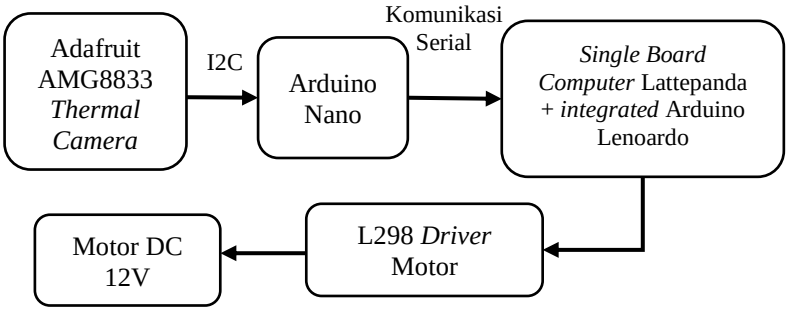
-----Halaman ini sengaja dikosongkan-----

BAB III
PERANCANGAN SISTEM

Perancangan sistem robot pengikut manusia menggunakan kamera termal dan *single board* computer lattepanda terdiri dari beberapa bagian yang meliputi diagram blok sistem, perancangan mekanik, perancangan elektronik, dan perancangan perangkat lunak. Perancangan mekanik meliputi pembuatan badan dasar robot dari bahan acrylic. Perancangan elektronik meliputi pembuatan rangkaian *battery*, rangkaian *connector*, MOSFET drivers serta GPIO lattepanda. Perancangan perangkat lunak meliputi pembuatan program untuk deteksi target, pembacaan data kamera termal, serta program untuk kontrol pergerakan robot menuju target.

3.1 Diagram Blok

Sistem robot pengikut manusia menggunakan kamera termal dan *single board* computer lattepanda dapat digambarkan pada sebuah diagram blok. Diagram blok sistem membantu menjelaskan hubungan satu sama lain antar bagian-bagian penyusun robot. Hubungan elektrikal antar bagian-bagian penyusun robot dapat dijelaskan dengan diagram blok pada gambar 3.1.



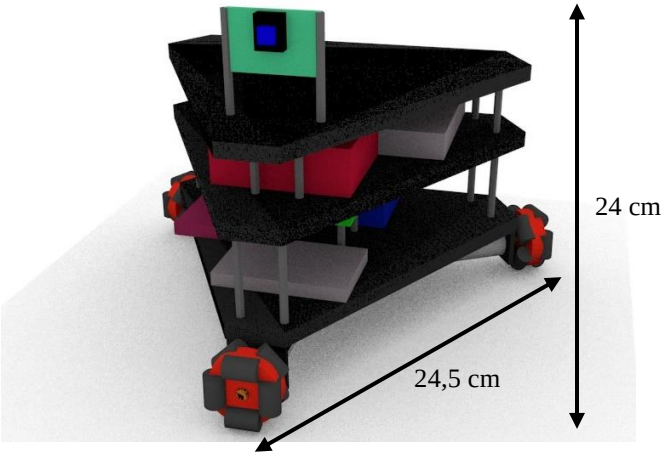
Gambar 3. 1. Diagram Blok Keseluruhan Sistem.

1. Kamera termal AdaFruit AMG8833 berfungsi untuk menangkap citra suhu di sekitar robot yang selanjutnya dikirimkan ke Arduino Nano melalui I2C. Mekanisme pembacaan kamera oleh Ardunio Nano akan dijelaskan pada *flowchart* di sub bab 3.4.2.

2. Arduino Nano berfungsi untuk mengolah data dari kamera termal. Data dari kamera termal akan ditampung terlebih dahulu sebanuyk 64 data yang berupa nilai suhu. Setelah data terkumpul sebanyak 64 lalu data tersebut dikirimakn ke *single board computer* Lattepanda melalui komunkasi serial.
3. Lattepanda berfungsi untuk melakukan *image processing* menggunakan C++ dan OpenCV agar didapatkan citra *thermal* dari target. Hasil dari *image processing* pada Lattepanda adalah berupa nilai posisi koordinat dan luas blob dari target. Data tersebut akan diolah kembali oleh Lattepanda menggunakan C# yang terintegrasi dengan *firmware* Arduino Leonardo sehingga Lattepanda dapat mengirim perintah ke *integrated* Arduino Leonardo secara langsung. Arduino Leonardo tersebut akan mengirim sinyal kepada *motor driver* L298.
4. Driver L298 berfungsi sebagai penggerak motor DC. Driver akan menerima sinyal dari Arduino Leonardo yang terintegrasi dengan Lattepanda.
5. Motor DC berfungsi sebagai aktuator agar robot bergerak menuju obyek yang terdeteksi dengan memanfaatkan kombinasi kecepatan dari tiga buah motor DC 12V.

3.2 Perancangan Mekanik

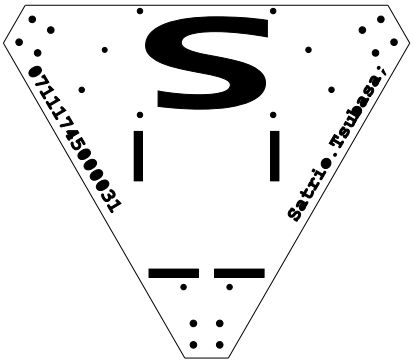
Perancangan mekanik utama yang dibuat pada tugas akhir ini merupakan 3 lapis badan robot dengan ukuran keseluruhan yaitu 22 cm x 24,5 cm x 24 cm. 3 lapis badan dasar robot diwujudkan dari bahan *acrylic*. Perancangan desain mekanik didasarkan pada perbedaan sudut antar tiga roda omni sebesar 120 derajat. Perbedaan sudut antar masing-masing roda berguna untuk kinematika robot sehingga menjadi dasar pertimbangan dalam desain robot omni dengan tiga roda. Selain dasar perbedaan sudut tiap roda, badan robot juga harus mampu menampung bagian-bagian penyusun robot lainnya. Gambar 3.2 menunjukkan desain 3 dimensi secara keseluruhan dari robot pengikut manusia pada tugas akhir ini.



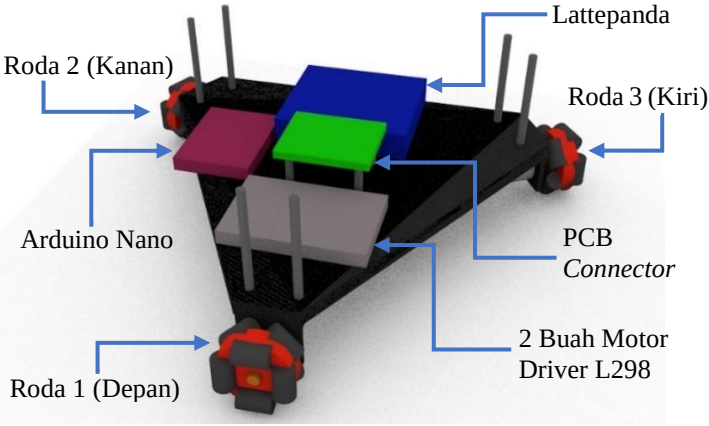
Gambar 3. 2. Desain 3D Keseluruhan.

3.2.1 Perancangan *Layer* Pertama Robot

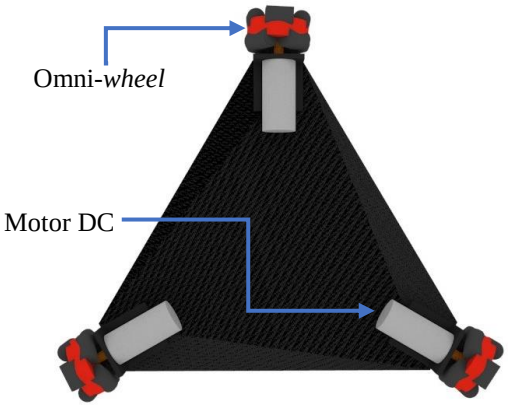
Lapisan pertama atau lapisan dasar dari robot akan dipasang 2 buah *motor driver* L298, *single board computer* lattepanda, PCB *connector*, *spacer*, 3 buah *omni-wheel* dan 3 buah motor DC. Gambar 3.3 menunjukkan desain 2 dimensi lapisan pertama dari robot yang akan dibuat pada tugas akhir ini. Sedangkan gambar 3.4 dan 3.5 menunjukan desain 3 dimensi lapisan pertama beserta tata letak dari komponen-komponen yang akan dipasang.



Gambar 3. 3. Desain 2D *Layer* Pertama.



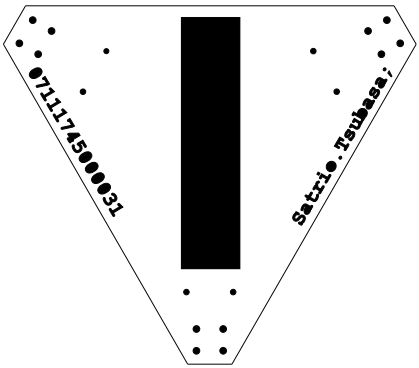
Gambar 3. 4. Desain 3D *Layer* Pertama.



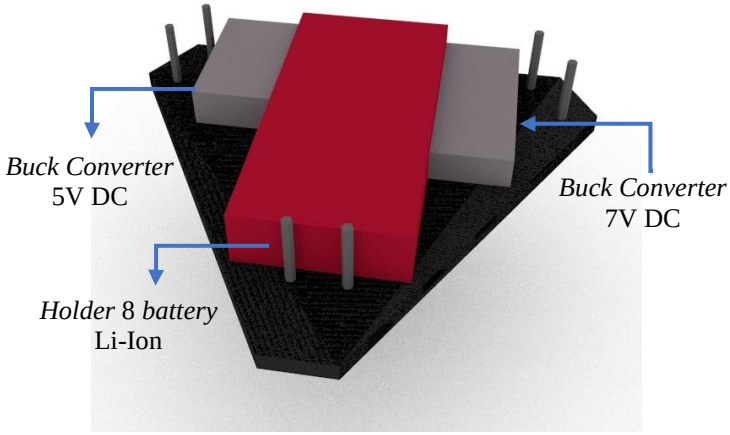
Gambar 3. 5. Desain 3D *Layer* Pertama Bagian Bawah.

3.2.2 Perancangan *Layer* Kedua Robot

Lapisan kedua dari robot akan dipasang 8 buah *battery* Li-Ion dengan holder battery, 2 buah *buck converter* dan *spacer*. Gambar 3.6 menunjukkan desain 2 dimensi lapisan kedua dari robot yang akan dibuat pada tugas akhir ini. Sedangkan gambar 3.7 menunjukan desain 3 dimensi lapisan kedua beserta tata letak dari komponen-komponen yang akan dipasang.



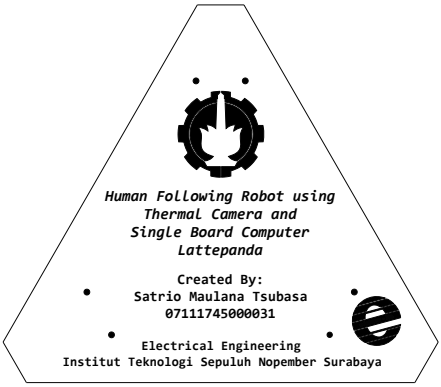
Gambar 3. 6. Desain 2D *Layer* Kedua.



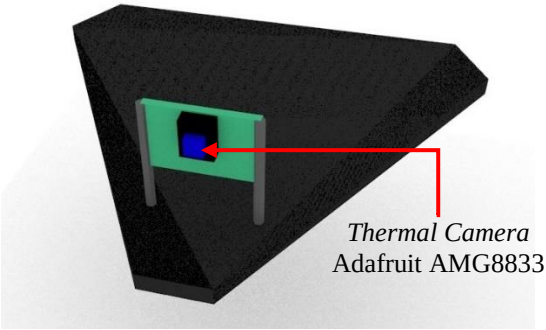
Gambar 3. 7. Desain 3D *Layer* Kedua.

3.2.3 Perancangan *Layer* Ketiga Robot

Lapisan ketiga dari robot akan dipasang kamera termal Adafruit AMG8833 8x8 piksel dan *spacer*. Gambar 3.8 menunjukkan desain 2 dimensi lapisan ketiga dari robot yang akan dibuat pada tugas akhir ini. Sedangkan gambar 3.9 menunjukan desain 3 dimensi lapisan ketiga beserta tata letak dari komponen-komponen yang akan dipasang.



Gambar 3. 8. Desain 2D *Layer* Ketiga.



Gambar 3. 9. Desain 3D *Layer* Ketiga.

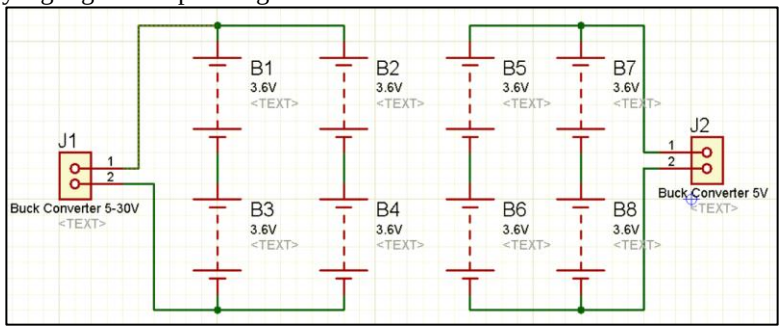
3.3 Perancangan Elektronik

Perancangan elektronik robot meliputi perancangan catu daya yang tersusun dari 8 buah battery Li-Ion, penentuan pin Arduino nano dan penentuan pin GPIO *single board computer* Lattepanda.

3.3.1 Perancangan Catu Daya

Daya yang digunakan oleh robot berasal dari 8 buah *battery* Li-Ion 3.6V DC 3100mAH. Catu daya berfungsi untuk menyediakan tegangan yang dibutuhkan bagian-bagian robot untuk bekerja. Secara garis besar, bagian-bagian robot membutuhkan tegangan 7V DC dan 5V DC. Bagian robot yang membutuhkan tegangan 7V DC yaitu dua buah

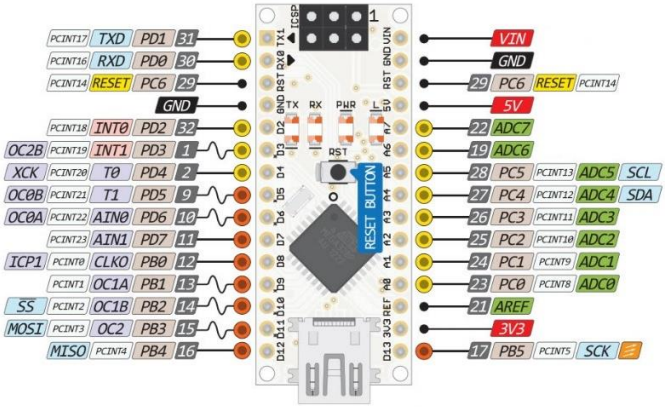
motor driver L298 dan tiga buah motor DC. Sedangkan bagian robot yang membutuhkan tegangan 5V DC yaitu *singleboard* computer Lattepanda, kamera termal Adafruit AMG8833 Arduino Nano, dan dua buah *motor driver* L298. Gambar 3.10 merupakan gambar rangkaian dari catu daya yang digunakan pada tugas akhir ini.



Gambar 3. 10. Rangkaian Catu Daya.

3.3.2 Konfigurasi Pin Arduno Nano

Arduino Nano pada tugas akhir ini berfungsi sebagai pemroses data kamera termal melalui komunikasi I2C. Pada Arduino Nano ini data dari I2C akan diubah kedalam bentuk satuan derajat celcius berukuran matriks 8x8. Data tersebut selanjutnya akan dikirimkan ke Lattepanda melalui komunikasi serial. Gambar 3.11 menunjukkan pin Arduino nano.



Gambar 3. 11. Pin Arduino Nano.

Tabel dibawah ini menunjukkan pin arduino nano yang digunakan pada tugas akhir ini.

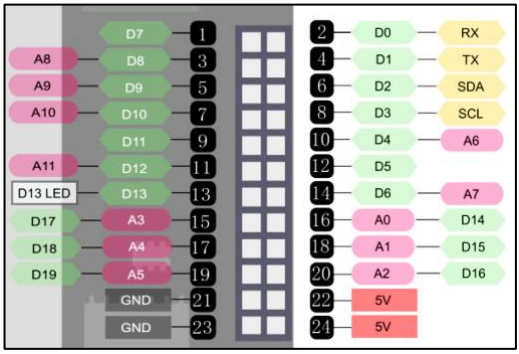
Tabel 3. 1. Penggunaan Pin Arduino Nano.

Pin Arduino Nano	Keterangan
SDA	Terhubung dengan pin SDA kamera termal
SCL	Terhubung dengan pin SCL kamera termal

3.3.3 Konfigurasi GPIO Lattepanda

Single board computer Lattepanda pada tugas akhir ini berfungsi sebagai pengolah citra dari data yang dikirimkan Arduino Nano. Lattepanda akan melakukan image processing menggunakan Visual Studio C++ *console app* dan OpenCV untuk mendapatkan citra thermal dari target yang akan diikuti oleh robot.

Hasil dari *image processing* pada Lattepanda adalah berupa nilai posisi koordinat dan luas blob dari target. Data tersebut akan diolah kembali oleh Lattepanda menggunakan Visual Studio C# *console app* yang terintegrasi dengan firmware Arduino Leonardo menggunakan Lattepanda Firmata, sehingga Lattepanda dapat mengirim perintah ke *integrated* Arduino Leonardo secara langsung. Arduino Leonardo tersebut akan mengirim sinyal kepada motor driver L298 untuk mengontrol motor. Gambar 3.12 menunjukkan GPIO Lattepanda melalui *integrated* Arduino Leonardo.



Gambar 3. 12. Arduino Nano Pin.

Tabel 3.2 menunjukkan GPIO dari *single board computer* Lattepanda *integrated* arduino Leonardo yang digunakan pada tugas akhir ini.

Tabel 3. 2. Penggunaan Pin GPIO Lattepanda.

Pin Arduino Leonardo / Pin GPIO Lattepanda	Keterangan
6 / D2	PWM Motor Depan
7 / D10	Logic Motor Kanan +
8 / D3	Logic Motor Kanan -
9 / D11	PWM Motor Kanan
10 / D4	PWM Motor Kiri
11 / D12	Logic Motor Kiri +
12 / D5	Logic Motor Kiri -
18 / D15	Logic Motor Depan +
19 / D19	Logic Motor Depan -

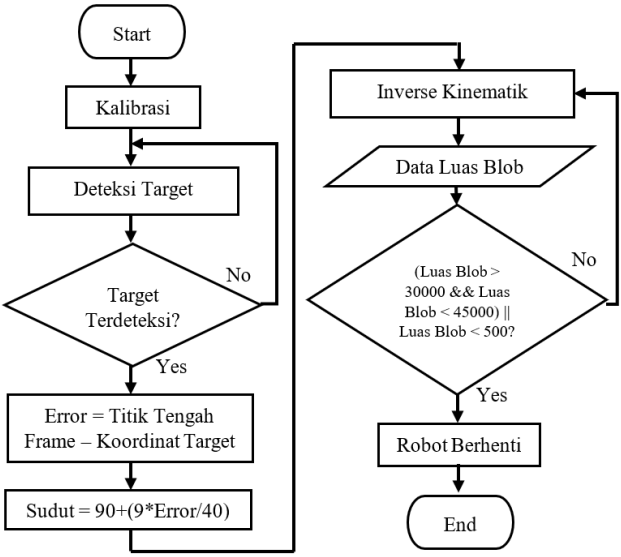
3.4 Perancangan Perangkat Lunak

Perancangan perangkat lunak bertujuan untuk pembuatan program untuk deteksi target, dan kontrol pergerakan robot menuju target.

Perancangan perangkat lunak pada tugas akhir ini secara detail terbagi menjadi tiga yaitu perancangan sistem navigasi, perancangan perangkat lunak *buffer* data dari kamera termal pada arduino nano, perancangan perangkat lunak pengolahan citra (*image processing*) pada Lattepanda menggunakan C++ dan OpenCV.

3.4.1 Perancangan Perangkat Lunak Sistem Navigasi

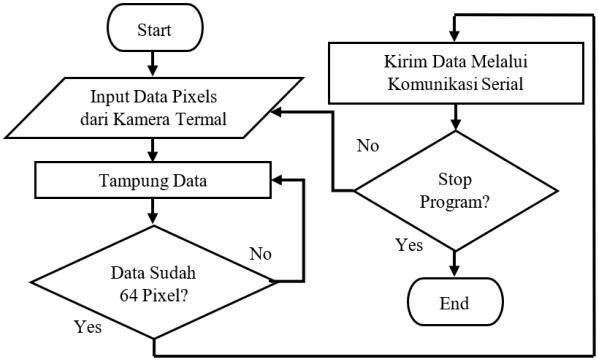
Pembuatan program untuk deteksi target memanfaatkan library OpenCV. Pustaka OpenCV digunakan dalam proses-proses pengolahan citra yang ditangkap kamera termal hingga didapatkan informasi dari citra target yang dimaksud. Informasi ini berupa jumlah blob (gumpalan pixel) untuk penentuan obyek terdeteksi atau tidak, sehingga bisa didapatkan data sudut, dan luas blob untuk penentuan posisi berhenti robot dari target. Gambar 3.13 menunjukkan flowchart sistem navigasi dari robot pengikut manusia menggunakan kamera termal dan *single board computer* Lattepanda.



Gambar 3. 13. Flowchart Sistem Navigasi.

3.4.2 Perancangan Perangkat Lunak *Buffer Data*

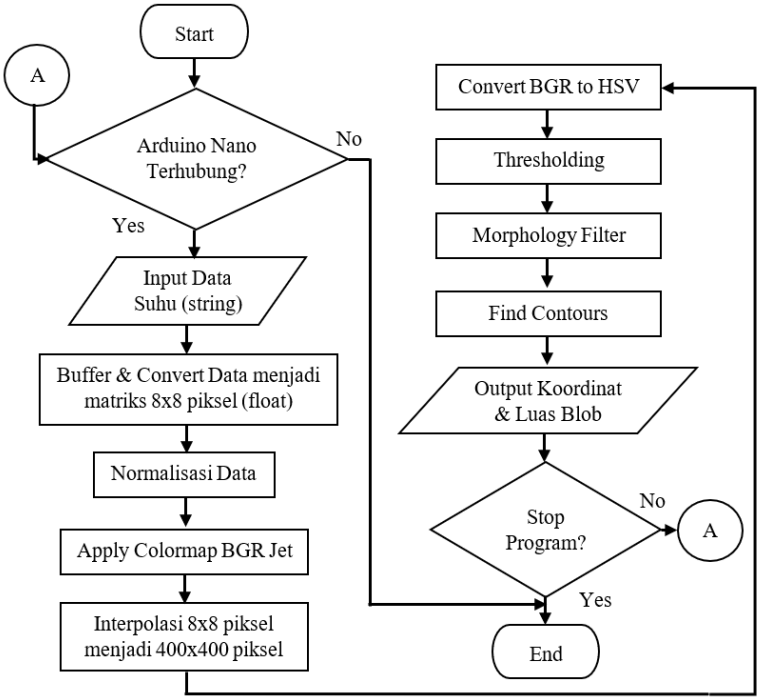
Arduino nano berfungsi sebagai penerima data (*buffer* data) dari kamera termal melalui I2C dan mengubah data tersebut kedalam nilai matriks 8 x 8 dengan bentuk derajat celcius. Data matriks tersebut kemudian dikirimkan ke Lattepanda melalui komunikasi serial. Gambar 3.14 menunjukkan *flowchart buffer data* dari kamera termal.



Gambar 3. 14. *Flowchart Buffer Data*.

3.4.3 Perancangan Perangkat Lunak *Image Prcessing*

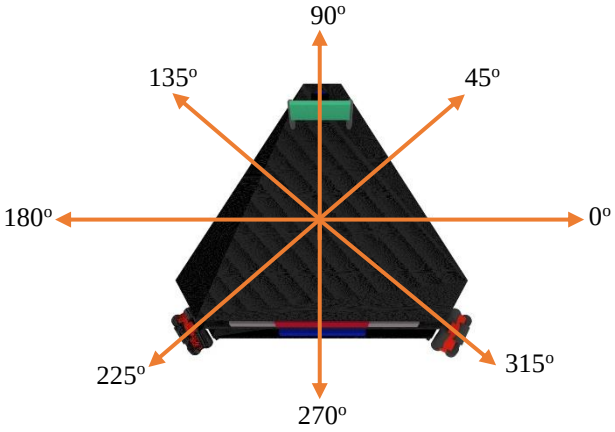
Image processing dilakukan menggunakan bahasa pemrograman C++ dan library open computer vision (OpenCV). Proses ini dilakukan pada *singleboard computer* Lattepanda. Dalam OpenCV terdapat berbagai macam *library* yang dapat digunakan untuk melakukan pengolahan citra termal. Dalam tugas akhir ini digunakan beberapa library OpenCV seperti matriks, applycolormap, interpolate, morphoplpgy filter dan find contours. Gambar 3. 15 menunjukkan *flowchart* dari *image processing* untuk mengolah *thermography* dari kamera termal.



Gambar 3. 15. *Flowchart Image Processing.*

3.5 Penentuan Orientasi Sudut Robot

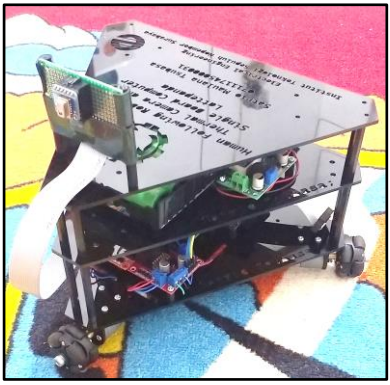
Penentuan orientasi sudut ini dilakukan untuk mempermudah identifikasi arah pergerakan dari robot pengikut manusia. Pada tugas akhir ini ditentukan apabila robot maju kedepan maka sudut yang akan dituju oleh robot tersebut adalah 90^o dengan posisi kamera termal menghadap depan. Gambar 3.17 dibawah ini menjelaskan secara detail mengenai orientasi sudut dan nomor motor dari sistem robot pengikut manusia yang dibuat pada tugas akhir ini.



Gambar 3. 16. Penentuan Sudut Robot.

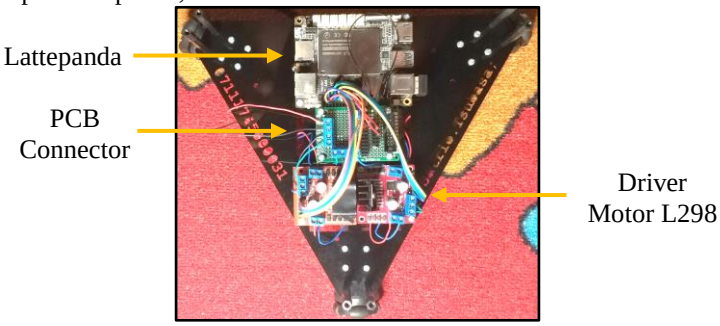
BAB IV
PENGUJIAN DAN ANALISA SISTEM

Bab ini membahas tentang pengujian dari sistem yang telah dirancang beserta analisis data yang diperoleh saat pengujian. Pengujian-pengujian yang dilakukan meliputi pengujian deteksi target oleh robot dan pengujian arah gerakan robot. Berdasarkan desain pada bab tiga, dilakukan implementasi desain robot omni-directional wheels dengan ukuran 22 cm x 24,5 cm x 25 cm seperti yang ditunjukkan pada gambar 4.1



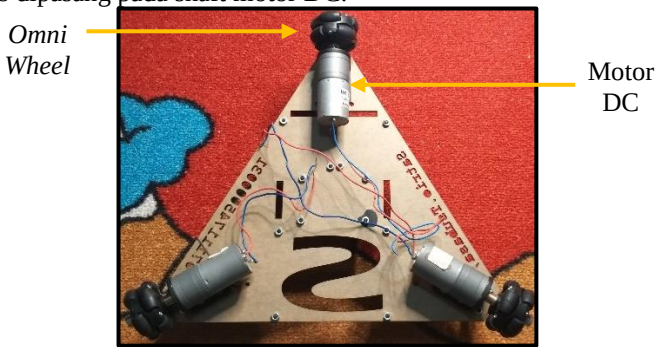
Gambar 4. 1. Realisasi Desain Robot Pengikut Manusia.

Gambar 4.2 menunjukkan realisasi desain *layer* pertama bagian atas dari robot pengikut manusia pada tugas akhir ini. Pada layer ini terdapat Lattepanda, PCB connector dan 2 buah driver motor L298.



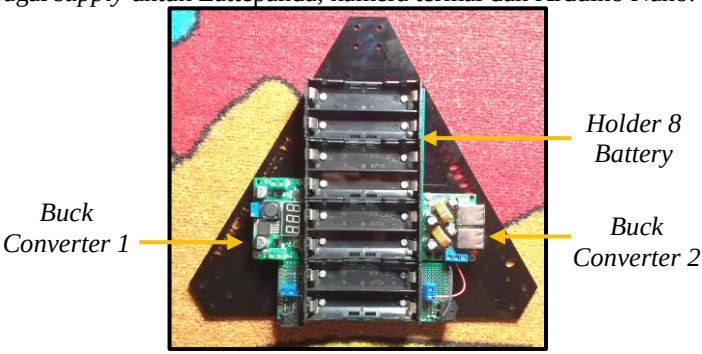
Gambar 4. 2. Realisasi *Layer* Pertama Bagian Atas.

Gambar 4.3 menunjukkan realisasi desain *layer* pertama bagian bawah dari robot pengikut manusia. Pada layer ini dipasang holder dari motor DC disambung menggunakan mur dan baut ke akrilik sehingga motor DC dapat menempel pada bagian bawah *layer* pertama. *Omni wheels* dipasang pada shaft motor DC.



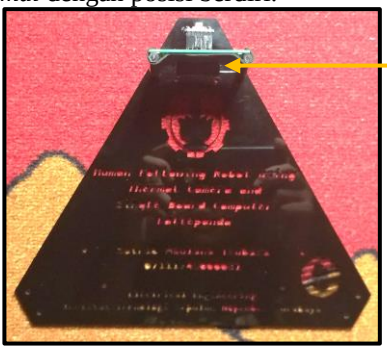
Gambar 4. 3. Realisasi *Layer* Pertama Bagian Bawah.

Realisasi desain *layer* kedua robot pengikut manusia pada tugas akhir ini ditunjukkan pada gambar 4.4. Pada layer ini dipasang holder dari 8 buah *battery* Li-Ion dan dua buah buck converter. Buck converter yang pertama yaitu *adjustable buck converter* dengan output mulai dari 5V hingga 35V DC. *Buck converter* yang pertama ini berfungsi sebagai *supply* untuk driver motor L298 yang diatur tetap pada 7V DC. Sedangkan *buck converter* yang kedua memiliki output tetap yaitu 5V DC, berfungsi sebagai *supply* untuk Lattepanda, kamera termal dan Arduino Nano.



Gambar 4. 4. Realisasi Desain *Layer* Kedua.

Realisasi *layer* ketiga dari robot pengikut manusia pada tugas akhir ini ditunjukkan pada gambar 4.5. Pada layer ini hanya dipasang PCB untuk *camera thermal* dengan posisi berdiri.

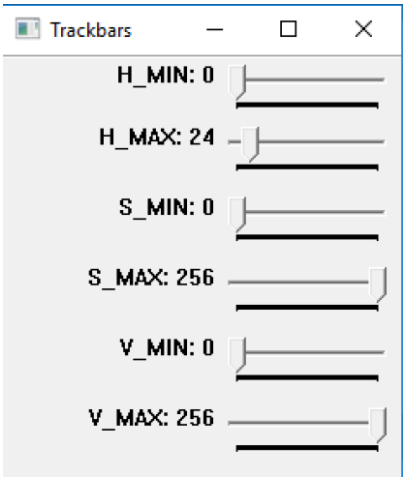


Gambar 4. 5. Realisasi Desain *Layer* Kedua.

4.1 Kalibrasi Manual

Kalibrasi manual perlu dilakukan sebelum proses utama pengolahan citra obyek. Kalibrasi manual bertujuan untuk mendapatkan nilai atas (*upper*) dan bawah (*lower*) yang akan digunakan pada proses utama pengolahan citra obyek sebagai batasan *threshold* sehingga didapatkan citra obyek sesuai yang dikehendaki. Proses kalibrasi manual dapat dilakukan dengan memanfaatkan *trackbar* dari *library* opencv sehingga nilai *Hue, Saturation*, dan *Value* dapat digeser sesuai keinginan. *Trackbar* tersebut berfungsi untuk penentuan nilai atas dan nilai bawah dari citra target.

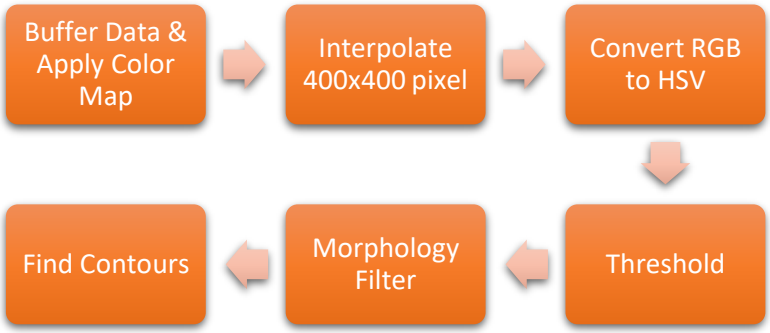
Gambar 4.6 merupakan *window* yang muncul untuk menentukan nilai parameter *Hue, Saturation*, dan *Value*. Jendela ini memiliki enam *trackbar* yang digunakan untuk menentukan tiga nilai bawah HSV dan tiga nilai atas HSV.



Gambar 4. 6. Kalibrasi dengan *Trackbar*.

4.2 Pengolahan Citra Target

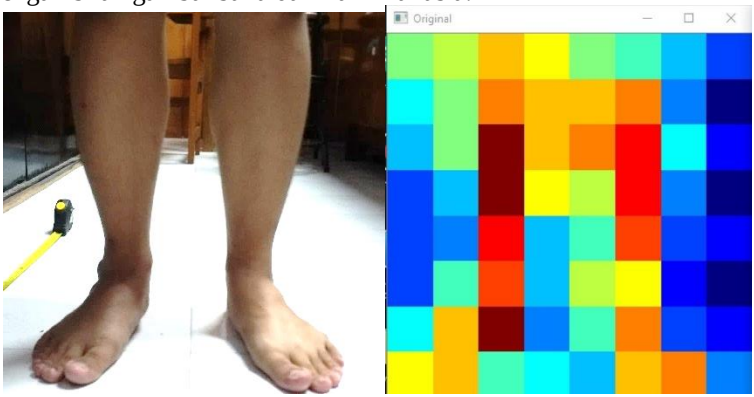
Citra yang ditangkap kamera termal Adafruit AMG8833 akan melalui beberapa proses pengolahan citra sehingga didapatkan citra yang dikehendaki. Citra hasil beberapa proses juga berguna untuk penentuan jarak dimana robot berhenti dari posisi obyek. Grafik pada gambar 4.7 berikut ini menjelaskan proses pengolahan citra yang dilakukan pada tugas akhir ini.



Gambar 4. 7. Urutan pengolahan citra.

4.2.1 Buffer Data dan Apply Color Map

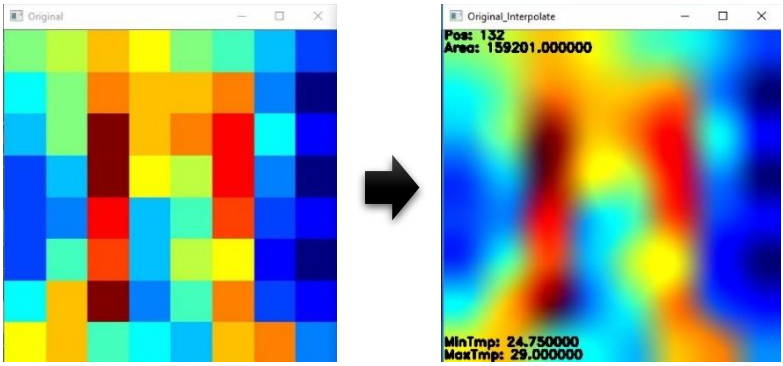
Langkah pertama yang dilakukan yaitu kamera termal mengambil citra termal di area sekitar robot. Citra yang diambil oleh kamera termal berukuran 8x8 piksel. Data tersebut adalah suhu yang terbaca oleh kamera termal. Data suhu tersebut ditampung dalam sebuah matriks berukuran 8x8. Nilai suhu dalam matrisk tersebut kemudian di-*mapping* kedalam sebuah *color map* pada *library* OpenCV yaitu COLORMAP_JET yang merupakan salah satu *color map* RGB sehingga menghasilkan thermography berukuran 64 piksel seperti yang ditunjukkan pada gambar 4.8. Pada contoh dibawah ini merupakan hasil pengambilan gambar suhu dari kaki manusia.



Gambar 4. 8. *Thermography* 8x8 piksel (kanan).

4.2.2 Interpolasi

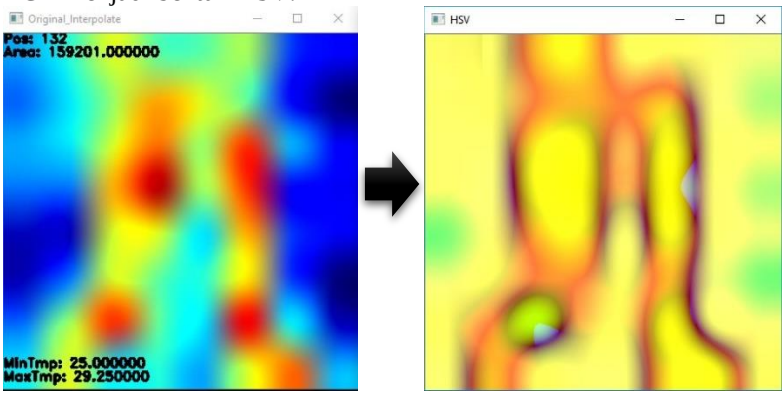
Data yang telah didapat setelah *apply color map* kemudian akan di interpolasi atau *resize* menjadi lebih besar, pada tugas akhir ini matriks 8x8 piksel dari proses sebelumnya di interpolasikan sebanyak 50 kali sehingga menghasilkan matriks 400 x 400 piksel. Tujuan dari interpolasi ini adalah untuk mempermudah proses find contours. Gambar 4.9 adalah hasil interpolasi gambar 8x8 piksel menjadi 400 x 400 piksel.



Gambar 4. 9. Hasil Interpolasi *Thermography* 400x400 piksel.

4.2.3 HSV *Conversion*

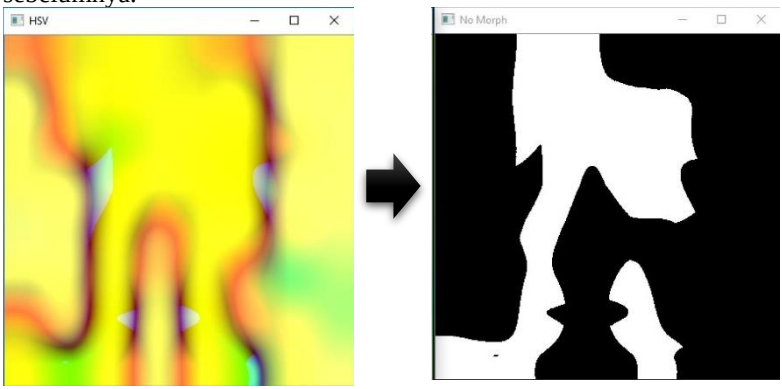
Proses konversi RGB ke HSV mengubah citra dari ruang warna RGB ke ruang warna HSV. Proses konversi diperlukan sebelum dilakukan thresholding dengan nilai kalibrasi bawah dan atas untuk citra obyek. Gambar 4.10 menunjukkan hasil konversi dari *thermography* yang merupakan *mapping color* dari library COLORMAP_JET dalam bentuk RGB menjadi bentuk HSV.



Gambar 4. 10. Hasil konversi *Thermography* RGB ke HSV.

4.2.4 *Thresholding*

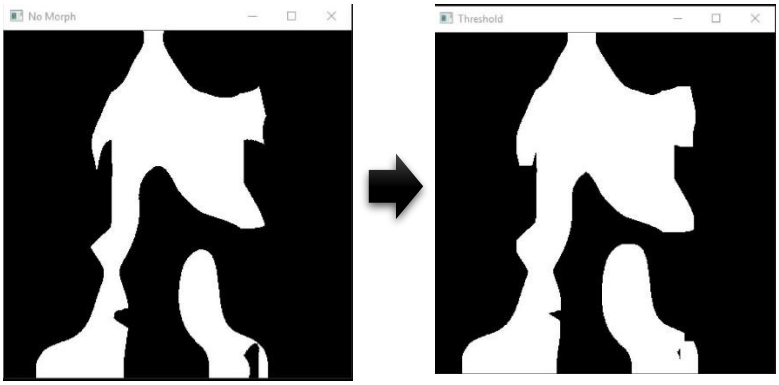
Proses Threshold (pembatasan) yaitu menerapkan nilai bawah (lower) dan nilai atas (upper) yang didapatkan dari kalibrasi sebagai nilai batasan sehingga didapatkan gumpalan (blob) citra obyek. Gambar 4.11 menunjukkan hasil penerapan *threshold* yang telah ditentukan oleh kalibrasi manual menggunakan *trackbar*. Proses *threshold* diterapkan pada citra *thermography* dalam bentuk HSV yang telah didapat sebelumnya.



Gambar 4. 11. Hasil *threshold* dari *thermography* HSV.

4.2.5 *Morphology Filter*

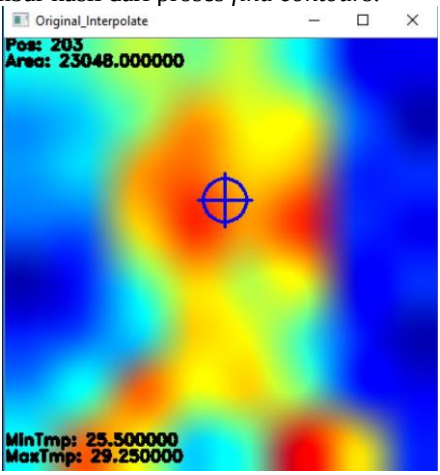
Citra hasil proses threshold masih memilik *noise* pada background dan *hole* pada obyek. Untuk mengatasi kedua hal tersebut, dapat dilakukan transformasi morphologi yaitu erosi dan dilasi. Erosi membantu menghilangkan titik-titik putih (noise) yang berada pada background. Sementara dilasi membantu mengisi lubang / bagian hitam yang ada pada obyek (foreground). Proses erosi dan dilasi dilakukan beberapa kali hingga didapatkan hasil yang diinginkan. Berikut adalah hasil setelah beberapa kali erosi dan dilasi.



Gambar 4. 12. Hasil *morphology filter* dari *threshold* HSV.

4.2.6 *Find Contours*

Proses *Find Contours* dilakukan setelah citra melalui proses transformasi *morphology* beberapa kali. Proses *Find Contours* akan menghitung kontur dari citra *binary*. Kontur-kontur ini dapat digambarkan dengan beberapa bentuk seperti lingkaran dan kotak. Pada tugas akhir ini, bentuk yang digunakan yaitu lingkaran. Dibawah ini ditunjukkan gambar hasil dari proses *find contours*.



Gambar 4. 13. Hasil *Find Contours* dari *Thermography.*

4.3 Pengujian Deteksi Citra Target

Pengujian deteksi citra target merupakan pengujian yang penting dalam penelitian ini. Pengujian deteksi citra obyek meliputi pengujian pada intensitas cahaya yang berbeda, pengujian pada jarak yang berbeda, pengujian pada suhu ruangan yang berbeda dan pengujian dengan kondisi target yang berbeda.

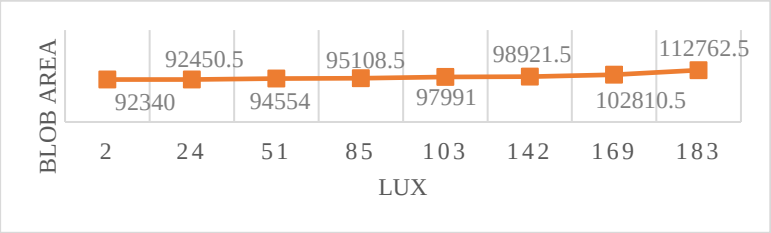
4.3.1 Pengujian pada Intensitas Cahaya Berbeda

Pengujian deteksi target ini dilakukan pada kondisi lux (pencahayaan) lingkungan yang berbeda. Target diuji pada parameter yang sama berupa jarak (cm) 20cm, suhu ruangan ber AC terbaca pada kamera termal bernilai 24^{o} C, kondisi target menggunakan celana pendek dan nilai threshold yang sama seperti ditunjukkan pada gambar 4.6.

Pengujian dilakukan pada Laboratorium Elektronika Industri B402 ITS Surabaya dengan rentang intensitas cahaya 2 – 183 lux. Pengujian dilakukan untuk mengetahui kinerja sistem saat di tempat dengan lux yang berbeda. Tabel 4.1 menunjukkan hasil pengujian sistem deteksi target pada lux berbeda.

Tabel 4. 1. Pengujian pada intensitas cahaya berbeda.

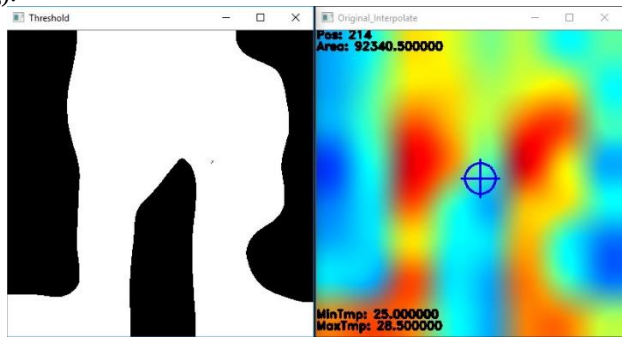
No	Lux	Jarak (cm)	Suhu Ruang (^{o} C)	Blob Area	Ket. Target
1	2	20	24	92340	Terdeteksi
2	24	20	24	92450.5	Terdeteksi
3	51	20	24	94554	Terdeteksi
4	85	20	24	95108.5	Terdeteksi
5	103	20	24	97991	Terdeteksi
6	142	20	24	98921.5	Terdeteksi
7	169	20	24	102810.5	Terdeteksi
8	183	20	24	112762.5	Terdeteksi



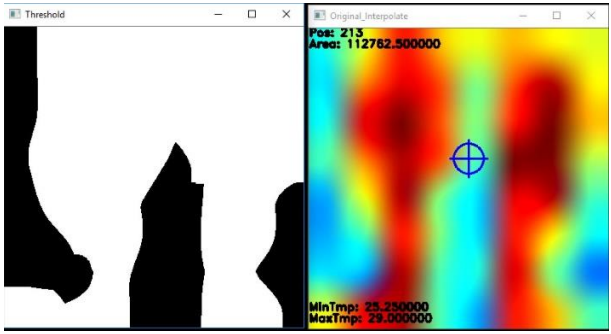
Gambar 4. 14. Grafik Pengujian Intensitas Cahaya Berbeda.

Gambar 4.14 menjelaskan grafik hubungan antara lux dan jumlah area blob. Semakin besar nilai lux, ada peningkatan jumlah area blob dengan jumlah yang kecil sehingga tidak terlalu berpengaruh pada pendeteksian target.

Gambar 4.15 menunjukkan hasil *thermography* dari pengujian dengan nilai lux paling kecil (gelap), sedangkan gambar 4.16 menunjukkan hasil *thermography* dari dengan nilai lux paling besar (terang).



Gambar 4. 15. *Thermography* pada Nilai Lux 2.



Gambar 4. 16. *Thermography* pada Nilai Lux 183.

Gambar 4.17 menunjukkan nilai lux paling kecil (gelap) yang nilai lux paling besar (terang) yang terukur ketika pengujian dilakukan.

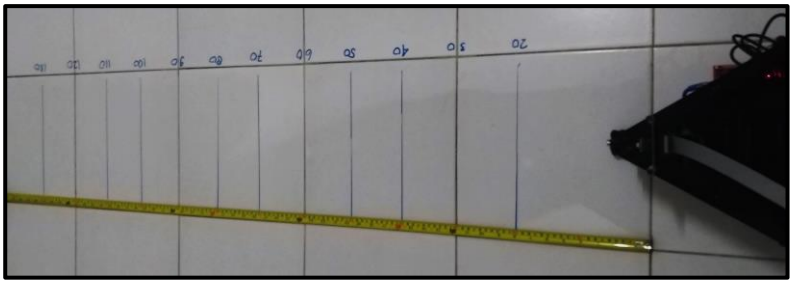
Gambar 4. 17. Nilai lux terbesar dan terkecil ketika pengujian.

Berdasarkan hasil pengujian pada tabel 4.1 dan grafik pada gambar 4.14, target berhasil dideteksi dengan kondisi pencahayaan ruangan dari paling gelap hingga paling terang. Terdapat sedikit perbedaan pada nilai luas area blob yang terdeteksi namun tidak mempengaruhi pembacaan target dikarenakan sistem menggunakan kamera termal yang membaca suhu dari target sehingga tidak bergantung pada kondisi pencahayaan. Sehingga dari pengujian ini dapat disimpulkan, pengaruh cahaya terhadap pendeteksian target adalah sangat kecil.

4.3.2 Pengujian pada Jarak Berbeda

Pengujian selanjutnya dilakukan pada kondisi jarak yang berbeda dari kamera robot dengan target. Target diuji pada parameter yang sama berupa nilai lux bernilai 169, suhu ruangan non-AC terbaca pada kamera termal bernilai 27.5^{o} C, kondisi target menggunakan celana pendek dan nilai threshold seperti yang ditunjukkan pada gambar 4.6.

Pengujian dilakukan pada ruang kamar yang berlokasi di daerah Keputih, Sukolilo, Surabaya. Rentang jarak antara robot dengan target berkisar dari 20 cm hingga 120 cm dengan perubahan setiap 10 cm seperti yang ditunjukkan pada gambar 4.17. Pengujian dilakukan untuk mengetahui kinerja sistem saat mendeteksi target dengan jarak yang berbeda. Tabel 4.2 menunjukkan hasil pengujian sistem deteksi target pada jarak berbeda.



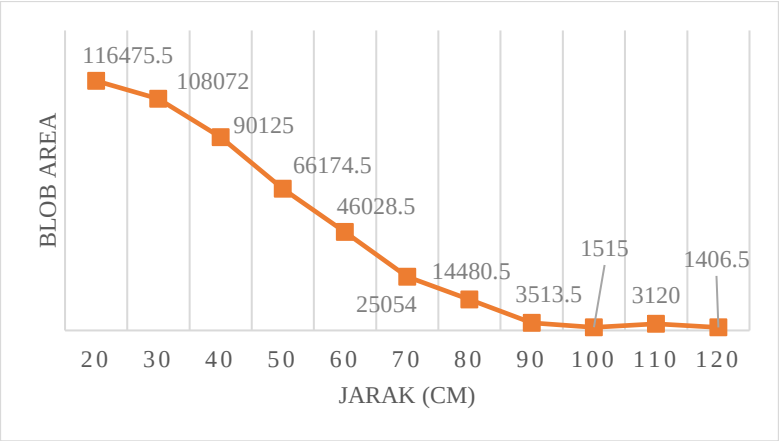
Gambar 4. 18. Metode pengujian jarak berbeda.

Tabel 4. 2. Pengujian pada jarak berbeda.

Jarak (cm)	Suhu Ruang (º C)	Lux	Blob Area	Ket. Target
20	27.5	183	116475.5	Terdeteksi
30	27.5	183	108072	Terdeteksi
40	27.5	183	90125	Terdeteksi
50	27.5	183	66174.5	Terdeteksi
60	27.5	183	46028.5	Terdeteksi
70	27.5	183	25054	Terdeteksi
80	27.5	183	14480.5	Terdeteksi
90	27.5	183	3513.5	Tidak Terdeteksi
100	27.5	183	1515	Tidak Terdeteksi
110	27.5	183	3120	Tidak Terdeteksi
120	27.5	183	1406.5	Tidak Terdeteksi

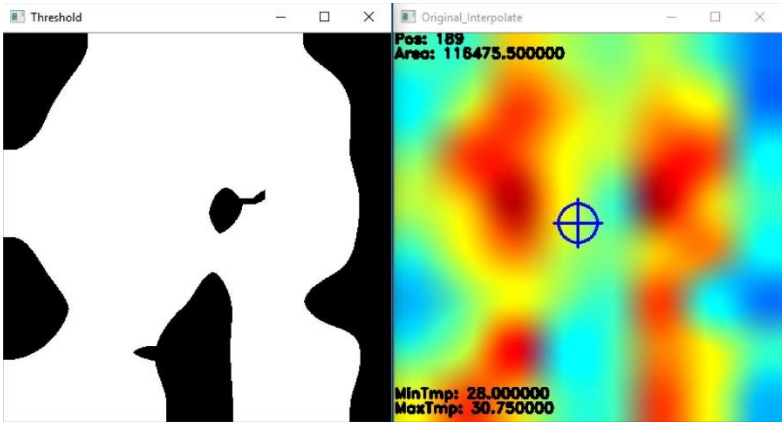
Gambar 4.19 menjelaskan grafik hubungan antara jarak dan jumlah area blob. Semakin besar jarak pada target, maka semakin kecil jumlah area blob yang terdeteksi. Pada pengujian ini menggunakan parameter yang ada didapatkan batas maksimum jarak paling jauh untuk pendeteksian target adalah 80 cm. Hal ini disebabkan ketika sumber panas (manusia) berada pada jarak yang terlalu jauh, kamera termal hanya akan membaca suhu ruangan saja.

Sehingga dari pengujian ini dapat disimpulkan, semakin jauh jarak target maka kemampuan pendeteksian target semakin berkurang dan semakin dekat jarak target maka pendeteksian akan semakin baik.

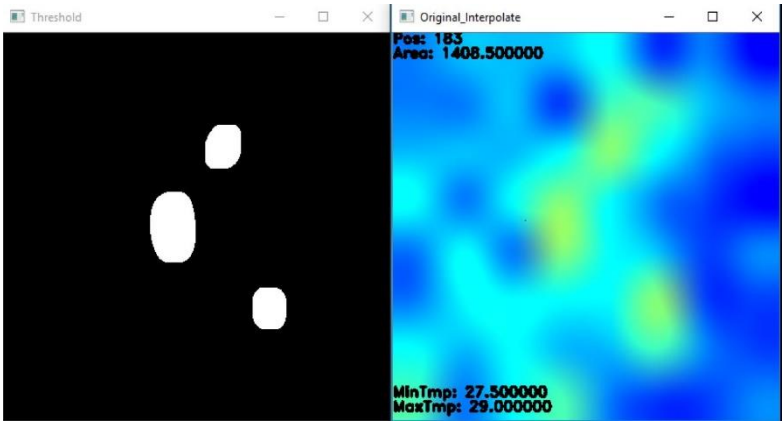


Gambar 4. 19. Grafik pengujian jarak berbeda.

Gambar 4.20 menunjukkan hasil *thermography* dari pengujian dengan jarak paling dekat (20cm), sedangkan gambar 4.21 menunjukkan hasil *thermography* dari dengan jarak paling jauh (120cm).



Gambar 4. 20. *Thermography* pada Jarak 20 cm.



Gambar 4. 21. *Thermography* pada Jarak 120 cm.

4.3.3 Pengujian pada Suhu Ruangan Berbeda

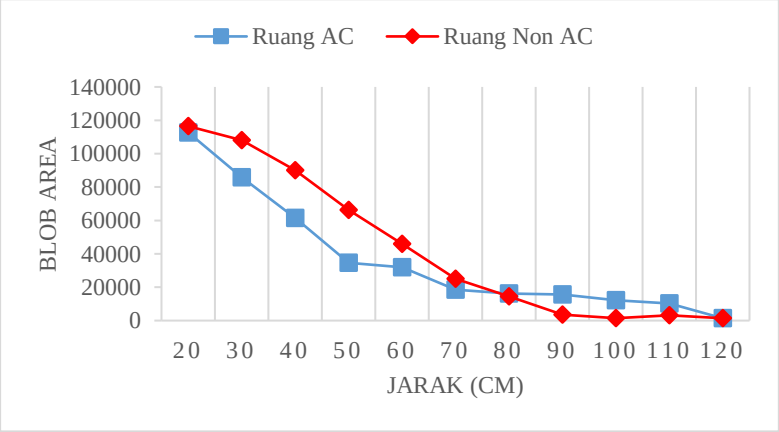
Pengujian selanjutnya dilakukan pada kondisi suhu ruangan yang berbeda yaitu ruangan ber AC nan ruangan non-AC. Target diuji pada parameter jarak yang berubah ubah dan parameter yang sama berupa nilai lux sebesar 183, kondisi target menggunakan celana pendek dan nilai threshold seperti yang ditunjukkan pada gambar 4.6.

Pengujian pada ruang non-AC dilakukan pada ruang kamar yang berlokasi di daerah Keputih, Sukolilo, Surabaya dengan suhu ruangan terbaca pada kamera termal bernilai 27.5^{o}C. Pengujian pada ruang ber-AC dilakukan pada Laboratorium Elektronika Industri B402 ITS Surabaya dengan suhu ruangan terbaca pada kamera termal bernilai 24^{o}C. Gambar 4.22 kiri menunjukkan pembacaan nilai suhu ruangan ber-AC dan Gambar 4.22 kanan menunjukkan pembacaan nilai suhu ruangan non-AC pada kamera termal. Rentang jarak antara robot dengan target berkisar dari 20 cm hingga 120 cm dengan perubahan setiap 10 cm. Pengujian dilakukan untuk mengetahui kinerja sistem saat mendeteksi target dengan suhu ruangan yang berbeda. Tabel 4.3 menunjukkan hasil pengujian.

Gambar 4. 22. Suhu Ruangan AC (kiri) dan Ruangan non-AC (kanan).

Tabel 4. 3. Pengujian pada suhu ruangan berbeda.

Jarak (cm)	Blob Area Ruang AC (24°C)	Blob Area Ruang Non-AC (27.5°C)	Ket. Target pada Ruang AC	Ket. Target pada Ruang Non-AC
20	112762.5	116475.5	Terdeteksi	Terdeteksi
30	85746.5	108072	Terdeteksi	Terdeteksi
40	61544.5	90125	Terdeteksi	Terdeteksi
50	34609.5	66174.5	Terdeteksi	Terdeteksi
60	32011	46028.5	Terdeteksi	Terdeteksi
70	18459.5	25054	Terdeteksi	Terdeteksi
80	16292	14480.5	Terdeteksi	Terdeteksi
90	15703.5	3513.5	Terdeteksi	Tidak Terdeteksi
100	12106	1515	Terdeteksi	Tidak Terdeteksi
110	10337	3120	Terdeteksi	Tidak Terdeteksi
120	1374	1406.5	Tidak Terdeteksi	Tidak Terdeteksi

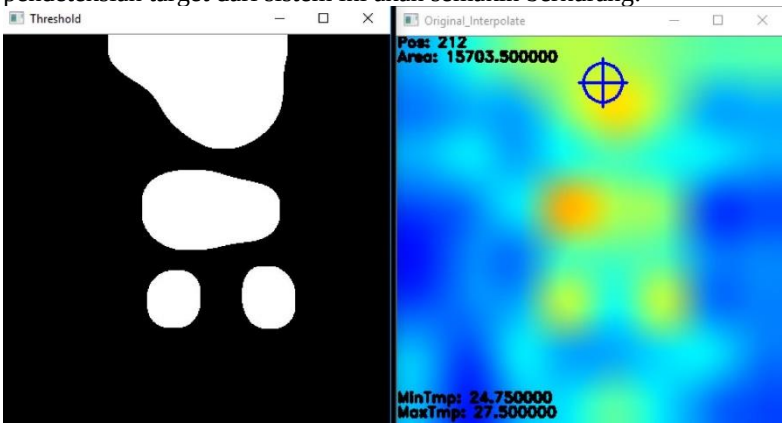


Gambar 4. 23. Grafik pengujian suhu ruangan berbeda.

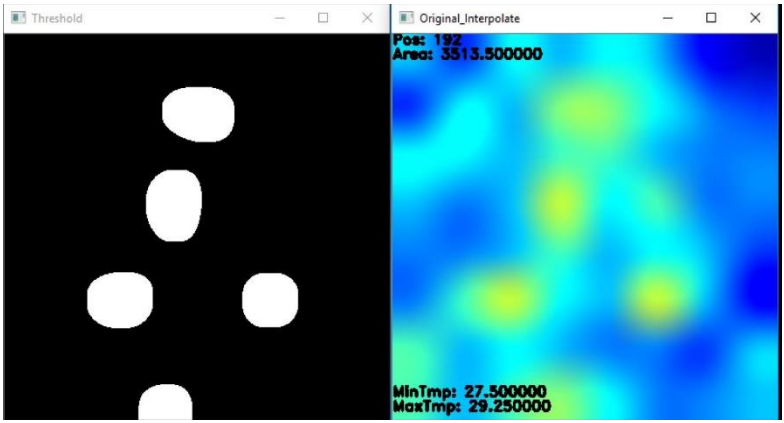
Gambar 4.23 menjelaskan grafik hubungan suhu ruangan yang berbeda, jarak dan jumlah area blob. Pada grafik 4.23 dan tabel 4.3 dapat dijelaskan bahwa pendeteksian target pada suhu ruangan yang lebih dingin jauh lebih baik dari pendeteksian target pada suhu ruangan yang lebih panas. Berdasarkan parameter yang ada, hasil pengujian ini menunjukkan pendeteksian target pada ruangan yang lebih dingin (24ºC) dapat dilakukan pada jarak yang lebih jauh mencapai 110 cm, sedangkan pada ruangan yang lebih panas (27.5ºC) target hanya dapat dideteksi pada jarak maksimum 80 cm.

Fenomena ini dikarenakan pada suhu ruangan 27.5ºC memiliki selisih nilai suhu yang lebih kecil dengan suhu tubuh manusia dewasa normal yang berkisar 36ºC. Sehingga pada jarak yang lebih jauh kemampuan pembacaan suhu tubuh manusia akan berkurang. Gambar 4.24 (ruangan AC) dan 4.25 (ruangan non-AC) menunjukkan *thermography* dari kedua kondisi ruangan pada jarak 90 cm. Dari gambar tersebut dapat diamati pada ruangan AC, target masih terdeteksi dan masih berbentuk seperti kaki manusia. Sedangkan pada ruangan non-AC, target sudah tidak terdeteksi dikarenakan yang terdeteksi hanyalah noise.

Sehingga dari pengujian ini dapat disimpulkan, semakin rendah suhu ruangan maka semakin baik pendeteksian target dan semakin suhu ruangan mendekati nilai suhu tubuh manusia, maka kemampuan pendeteksian target dari sistem ini akan semakin berkurang.



Gambar 4. 24. *Thermography* ruangan AC, jarak 90cm.



Gambar 4. 25. *Thermography* ruangan non-AC, jarak 90cm.

4.3.4 Pengujian pada Kondisi Target Berbeda

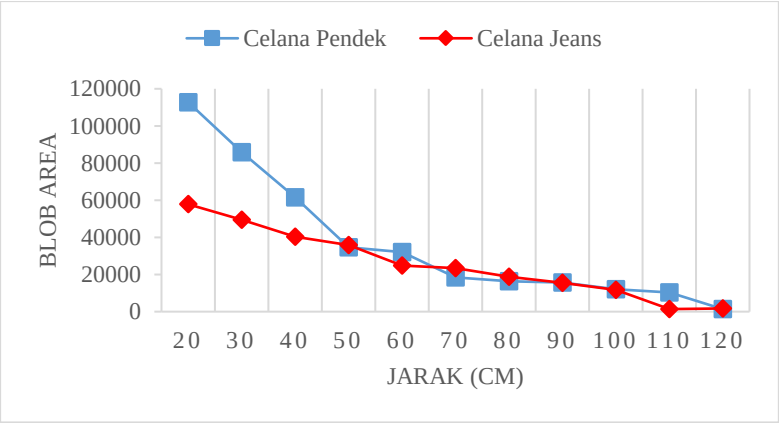
Pengujian selanjutnya dilakukan pada kondisi target yang berbeda yaitu dengan menggunakan celana pendek dan menggunakan celana berbahan jeans tebal. Target diuji pada parameter jarak yang berubah ubah dan parameter lain yang sama berupa nilai lux sebesar 183, suhu ruangan AC dan nilai threshold seperti yang ditunjukkan pada gambar 4.6.

Pengujian dilakukan pada Laboratorium Elektronika Industri B402 ITS Surabaya dengan suhu ruangan terbaca pada kamera termal bernilai 24^{o}C. Gambar 4.26 kiri menunjukkan kondisi target menggunakan celana pendek dan Gambar 4.26 kanan menunjukkan kondisi target menggunakan celana berbahan jeans. Rentang jarak antara robot dengan target berkisar dari 20 cm hingga 120 cm dengan perubahan setiap 10 cm. Pengujian dilakukan untuk mengetahui kinerja sistem saat mendeteksi target dengan kondisi tarhet yang berbeda. Tabel 4.4 menunjukkan hasil pengujian.

Gambar 4. 26. Pengujian Kondisi Target yang Berbeda.

Tabel 4. 4. Pengujian pada Kondisi Target Berbeda.

Jarak (cm)	Blob Area Celana Pendek	Blob Area Celana Jeans	Ket. Target Celana Pendek	Ket. Target Celana Jeans
20	112762.5	57818	Terdeteksi	Terdeteksi
30	85746.5	49473.5	Terdeteksi	Terdeteksi
40	61544.5	40300	Terdeteksi	Terdeteksi
50	34609.5	35792	Terdeteksi	Terdeteksi
60	32011	24891	Terdeteksi	Terdeteksi
70	18459.5	23353.5	Terdeteksi	Terdeteksi
80	16292	18701.5	Terdeteksi	Terdeteksi
90	15703.5	15465.5	Terdeteksi	Terdeteksi
100	12106	11662.5	Terdeteksi	Terdeteksi
110	10337	1380.5	Terdeteksi	Tidak Terdeteksi
120	1374	1636.5	Tidak Terdeteksi	Tidak Terdeteksi

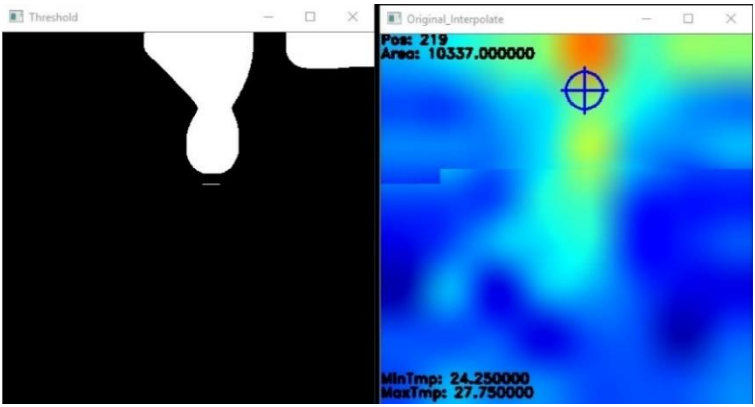


Gambar 4. 27. Grafik pengujian kondisi target berbeda.

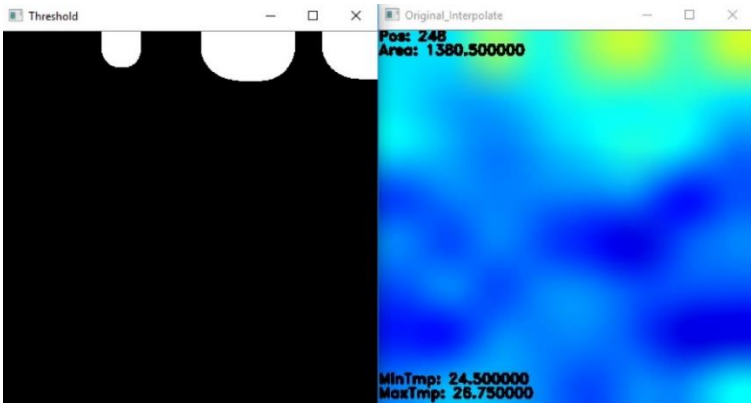
Gambar 4.27 menjelaskan grafik hubungan kondisi target yang berbeda, jarak dan jumlah area blob. Pada grafik 4.27 dan tabel 4.35 dapat dijelaskan bahwa pendeteksian target dengan menggunakan celana pendek lebih baik dari pendeteksian target yang menggunakan celana jeans. Berdasarkan parameter yang ada, hasil pengujian ini menunjukkan pendeteksian target yang menggunakan celana pendek dapat dilakukan pada jarak maksimum mencapai 110 cm, sedangkan pada target yang menggunakan celana jeans hanya dapat dideteksi pada jarak maksimum 100 cm.

Fenomena ini dikarenakan pada kamera termal akan membaca suhu dari lapisan luar dari target, sehingga ketika target menggunakan celana jeans, suhu yang terukur oleh kamera termal sudah tidak murni berupa suhu tubuh manusia sepenuhnya melainkan sudah tercampur oleh suhu dari celana jeans tersebut. Gambar 4.28 (target menggunakan celana pendek) dan 4.29 (target menggunakan celana jeans) menunjukkan *thermography* dari kedua kondisi target pada jarak 110 cm. Dari gambar tersebut dapat diamati pada target yang menggunakan celana pendek, target masih terdeteksi. Sedangkan pada target yang menggunakan celana jeans, target sudah tidak terdeteksi dikarenakan yang terdeteksi hanyalah noise.

Sehingga dari pengujian ini dapat disimpulkan, semakin sedikit kondisi halangan pada tubuh target maka semakin baik pendeteksian target.



Gambar 4. 28. *Thermography* Target Menggunakan Celana Pendek.



Gambar 4. 29. *Thermography* Target Menggunakan Jeans.

4.4 Pengujian Kecepatan Motor

Pengukuran kecepatan motor DC dilakukan dengan memberi input tegangan dengan *duty cycle* yang berbeda. Tegangan maksimum yang didapatkan oleh motor DC adalah 7V DC yang berasal dari *battery* li-ion. Tujuan pengukuran ini yaitu untuk mengetahui kecepatan (rpm) yang dihasilkan tiap motor DC karena arah pergerakan robot ditentukan oleh kombinasi kecepatan motor DC. Prosedur pengukuran dilakukan dengan Digital Photo Tachometer. Hasil pengukuran kecepatan motor ditunjukkan pada tabel 4.5.

Tabel 4. 5. Pengujian Kecepatan Motor.

***Duty Cycle* (%)**	**Kecepatan Motor**		
	Motor 1 (*Front*) (RPM)	Motor 2 (*Right*) (RPM)	Motor 3 (*Left*) (RPM)
0	0	0	0
25	7.3	7.2	7.2
50	15.1	14.8	14.7
75	23.6	23.3	23.1
100	30.7	30.2	30

Dari pengukuran kecepatan tiga motor DC yang telah dilakukan, didapatkan hasil yang menunjukkan bahwa saat diberi *duty cycle* yang sama, tiga motor DC memiliki respon kecepatan yang berbeda.

4.5 Pengujian Arah Gerakan Robot

Pengujian arah gerakan robot dilakukan untuk mengetahui error dari gerakan robot omni. Pengujian dilakukan dengan mengirim perintah robot omni untuk bergerak pada sudut-sudut yang diujikan. Prosedur pengukuran error dilakukan dengan busur sebagai alat ukur simpangan yang terjadi. Penanda berupa spidol ditempelkan searah sudut uji pada badan robot untuk mengetahui simpangan arah gerakan robot. Gambar 4.30 menjelaskan pengujian arah gerakan robot. Hasil pengujian arah pergerakan robot ditunjukkan pada tabel 4.6.

Gambar 4. 30. Pengujian arah gerakan robot.

Tabel 4. 6. Pengujian arah pergerakan robot.

Sudut Uji (º)	Pergerakan (º)	Error (%)
0	6	1.67
45	59	3.89
90	93	0.83
135	151	4.44
180	187	1.94

Error muncul pada semua sudut pengujian. Error muncul diakibatkan karena kecepatan putar dari ketiga motor berbeda ketika diberi *duty cycle* yang sama seperti yang diketahui dari hasil pengujian sub bab 4.2.

4.6 Pengujian Kecepatan Robot

Pengujian ini dilakukan dengan mengubah kecepatan motor DC melalui *duty cycle*. Sumber tegangan maksimum yang didapat oleh motor diatur sebesar 7V DC melalui *buck converter*. Hasil pengujian arah pergerakan robot ditunjukkan pada tabel 4.7.

Tabel 4. 7. Pengujian Kecepatan Robot.

Duty Cycle (%)	Waktu (s)	Jarak (cm)	Kecepatan (cm/s)
0	0	100	0
10	0	100	0
20	0	100	0
30	0	100	0
40	0	100	0
50	14.58	100	6.17
60	5.93	100	15.17
70	4.42	100	20.36
80	3.87	100	23.26
90	3.43	100	26.24
100	3.43	100	26.24

Dari hasil pengujian diketahui bahwa ketika diberikan *duty cycle* dibawah 50%, robot tidak mampu bergerak, dikarenakan *duty cycle* yang dihasilkan terlalu kecil sehingga motor DC tidak mampu menggerakkan beban robot. Ketika *duty cycle* yang diberikan lebih dari atau sama dengan

50%, robot mampu mengikuti target. Semakin tinggi *duty cycle* yang diberikan, maka akan semakin tinggi kecepatan robot.

4.7 Pengujian Respon Robot

Pengujian kecepatan respon robot dilakukan untuk mengetahui kemampuan respon robot untuk mengikuti variasi gerakan dari target. Pengujian ini dilakukan dengan mengubah jenis pergerakan yang akan dilakukan oleh target. Hasil pengujian ini ditunjukkan pada tabel 4.8.

Tabel 4. 8. Pengujian Respon Robot.

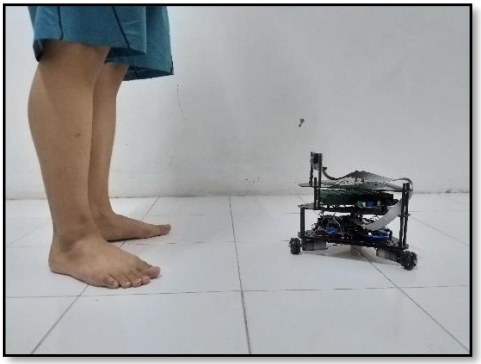
Kecepatan Pergerakan Manusia (m/s)	**Target Terikuti**	
	Ya	Tidak
0.43	✓	
0.81	✓	
1.57		✓
2.95		✓

Dari hasil pengujian diketahui bahwa ketika target bergerak dengan kecepatan 0.81 m/s, robot masih mampu mengikuti target. Namun ketika target bergerak lebih dari 0.81 m/s, robot sudah tidak mampu mengikuti target. Hal ini dikarenakan spesifikasi *frame rate* dari kamera termal yang digunakan pada tugas akhir ini adalah 10 fps. Sedangkan untuk standard *frame rate* kamera normal pada umumnya adalah 30 fps.

4.8 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem dilakukan dengan tujuan untuk mengetahui tingkat keberhasilan sistem. Sistem robot dinyatakan berhasil apabila berhasil mendeteksi target dan bergerak menuju target.

Pengujian ini dilakukan 20 kali. Prosedur pengujian keseluruhan sistem yaitu dilakukan pada ruangan ber-AC dengan suhu 24^{o}C, kondisi pencahayaan bernilai 183 lux, target menggunakan celana pendek dan target berjalan dengan kecepatan 0.81 m/s. Di dalam ruangan hanya terdapat 1 orang manusia. Gambar 4.31 menunjukkan ilustrasi pengujian keseluruhan sistem ini, dimana target seorang manusia akan bergereak dan robot dinyatakan berhasil apabila mampu mengikuti target target yang bergerak. Hasil pengujian dicatat pada tabel pengujian 4.9.



Gambar 4. 31. Pengujian keseluruhan sistem.

Tabel 4. 9. Pengujian keseluruhan sistem.

Pengujian ke-	Target Terdeteksi		Target Terikuti	
	Ya	Tidak	Ya	Tidak
1	✓		✓	
2	✓		✓	
3	✓		✓	
4	✓		✓	
5	✓		✓	
6	✓		✓	
7	✓		✓	
8	✓		✓	
9	✓		✓	
10	✓		✓	
11	✓		✓	
12	✓		✓	
13	✓		✓	
14	✓		✓	
15	✓		✓	
16	✓		✓	
17	✓		✓	
18	✓		✓	
19		✓		✓
20		✓		✓

Tabel 4.9 menjelaskan hasil yang didapat saat pengujian keseluruhan sistem pada ruangan ber-AC dengan suhu 24ºC, target menggunakan celana pendek dan target berjalan dengan kecepatan 0.81 m/s. Dari 20 kali percobaan yang dilakukan, tingkat keberhasilan robot mendeteksi target dan mengikuti target seorang manusia adalah 90%.

Dari 20 kali percobaan, robot pengikut manusia yang telah dibuat pada tugas akhir ini mengalami 2 kali kegagalan untuk mengikuti target pada percobaan ke 19 dan 20. Hal ini dikarenakan sistem robot sudah terlalu lama bekerja sehingga terjadi peningkatan suhu pada *single board computer* lattepanda. Peningkatan suhu ini menyebabkan terjadinya perlambatan pada kemampuan *processor* dari lattepanda sehingga *thermography* yang didapatkan sudah tidak *real time*.

-----Halaman ini sengaja dikosongkan-----

BAB V
PENUTUP

5.1 Kesimpulan

Pada tugas akhir ini telah dirancang dan dibuat sebuah sistem robot pengikut manusia menggunakan kamera termal dan *single board computer* Lattepanda. Beberapa pengujian yang telah dilakukan pada tugas akhir ini sehingga dapat diambil beberapa kesimpulan. Sistem pendeteksian robot pengikut manusia yang telah dibuat mampu mendeteksi target pada ruangan gelap maupun terang sehingga kondisi cahaya tidak banyak berpengaruh. Berdasarkan pengujian yang telah dilakukan, jarak maksimal untuk robot ini mampu mendeteksi target adalah 80 cm pada suhu ruangan 27.5^{o}C dan 110 cm pada suhu ruangan 24^{o}C. Kondisi pakaian yang target gunakan juga berpengaruh terhadap jarak jangkauan pendeteksian robot ini. Berdasarkan pengujian yang dilakukan pada suhu ruang sebesar 24^{o}C, jarak maksimal untuk robot ini mampu mendeteksi target adalah 110 cm dengan kondisi target menggunakan celana pendek dan 100 cm dengan kondisi target menggunakan celana jeans. Pengujian kecepatan robot menunjukkan bahwa ketika diberi *duty cycle* kurang dari 50% maka robot tidak akan bergerak. Pengujian respon robot menunjukkan bahwa robot tidak mampu mengikuti target yang bergerak lebih dari 0.81 m/s. Pengujian keseluruhan sistem menunjukkan tingkat keberhasilan robot mendeteksi dan mengikut target seorang manusia yaitu 90%.

5.2 Saran

Saran untuk pengembangan penelitian ini yaitu kamera termal yang digunakan dapat diganti dengan kamera termal dengan *frame rate* nya yang lebih baik sehingga respon robot dapat mengikuti pergerakan target yang cepat. Metode untuk deteksi obyek 360 derajat seperti penggunaan kamera tipe omnidirectional yang memiliki *field of view* 360 derajat agar dapat mendeteksi obyek di segala sudut sehingga mampu memaksimalkan kelebihan robot omni-directional.

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

[1] Latif, M. dkk. 2013. "Perancangan Pendeteksian Target Berdasarkan Warna Pakaian Pada Sistem Robot Pengikut Manusia". Prosiding Seminar Nasional Teknologi Informasi dan Multimedia. Yogyakarta., Vol. 22, No. 6

[2] Kwon, H. dkk. 2015. "Person Tracking with a Mobile Robot using Two Uncalibrated Independently Moving Camera*". Proceedings of the 2015 IEEE International Conference on Robotics and Automation*.

[3] Hans, S. 2016. "Implementasi Metoda Thresholding untuk Mendeteksi Objek Manusia Menggunakan Infrared Camera" [skripsi]. Bandung (ID): Universitas Kristen Maranatha Bandung.

[4] Stewart Sean M and Johnson Barry R. 2017. "*Blackbody* Radiation". CRC Press: New York. Hal. 3.

[5] Balaji, C. 2014. "Essential of Radiation Heat Transfer". Ane Books Pvt. Ltd.: New Delhi, India. Hal. 14.

[6] Balaji, C. 2014. "Essential of Radiation Heat Transfer". Ane Books Pvt. Ltd.: New Delhi, India. Hal. 18.

[7] Vollmer, Michael and Mollmann, K.P. 2010. "Infrared Thermal Imaging". Wiley- VCH: Weinheim, Germany. Hal 10.

[8] Syarifudin, M. 2015. *Rancang Bangun Prototipe Troli Pengikut Manusia Dengan Kamera* [skripsi]. Madura (ID): Universitas Truojoyo Madura.

[9] Sudrajat dan M. Rivai, 2019. "Kontrol Pergerakan Omni-Directional Wheels Robot Berdasar Citra Obyek" [skripsi]. Surabaya (ID): Institut Teknologi Sepuluh Nopember. Hal 15.

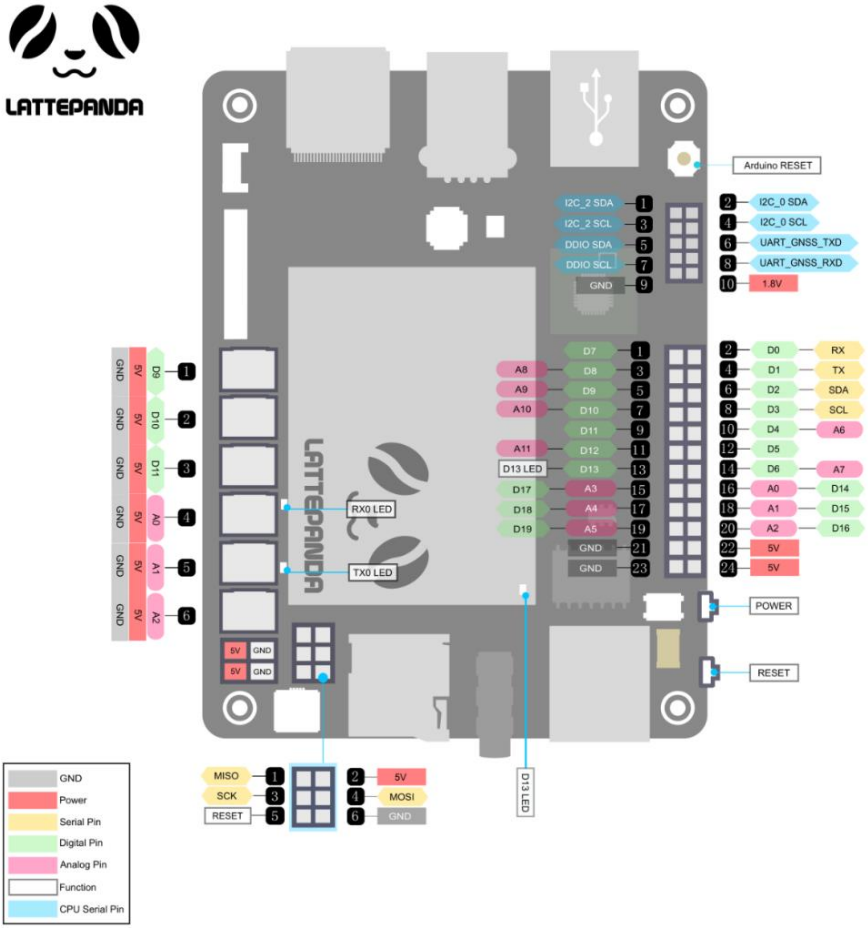
[10] Hadha Afrisal, "Portable Smart Sorting and Grading Machine for Fruits Using Computer Vision," *International Conference on Computer, Control, Informatics and Its Applications*, pp. 71-75, 2013.

[11] S. C. A. G. Martin Loesdau, "Chromatic Indices in the Normalized RGB Color Space," dalam *Research Gate*, Punauia Tahiti, 2017.

[12] J. S. Wibowo, “Deteksi dan Klasifikasi Citra Berdasarkan Warna Kulit Menggunakan HSV,” *Jurnal Teknologi Informasi* DINAMIK, vol. 16, pp. 118-123, Juli 2011.
[13] G. B. &. A. Kaehler, Learning OpenCv, Sebastopol: O'reilly Media, 2008.
[14] D. A. Kalimuddin, “Rancang Bangun Omni-Directional Wheels Robot Sebagai Gas Tracker Berbasis Mikrokontroller STM32F4-Discovery,” dalam *Tugas Akhir*, Surabaya, 2016.
[15] Harianto, M. Rivai dan D. Purwanto, “Implementation of Electronic Nose in Omni-directional Robot,” *International Journal of Electrical and Computer Engineering* (IJECE), vol. 3, pp. 399-406, June 2013.
[16] F. D. Krisandika, Tasripan dan M. Rivai, 2018. “Pelacak Cahaya Matahari Berbasis Citra Pada Panel Surya Menggunakan *Single Board Computer* Lattepanda” [skripsi]. Surabaya (ID): Institut Teknologi Sepuluh Nopember. Hal 29.
[17] Kiam Heong Ang, G. Chong, dan Yun Li, “PID kontrol sistem analysis, design, and technology,” *IEEE Transactions on Kontrol Sistems Technology*, vol. 13, no. 4, hlm. 559–576, Jul 2005.
[18] F. R. Saputra dan M. Rivai, “Autonomous Surface Vehicle sebagai Alat Pemantau Lingkungan Menggunakan Metode Navigasi Waypoint” Jurnal Teknik ITS, vol. 7, no. 1, Mar 2018.
[19] Gapinski, Andrzej J; “A Servo Sistem: PID Kontrol”, *Proceedings of the 2007 IEMS Conference*, 2007
[20] M. S. Ramadhan dan M. Rivai, “Sistem Kontrol Tingkat Kekeruhan pada Aquarium Menggunakan Arduino Uno,” Jurnal Teknik ITS, vol. 7, no. 1, Mar 2018.

LAMPIRAN A

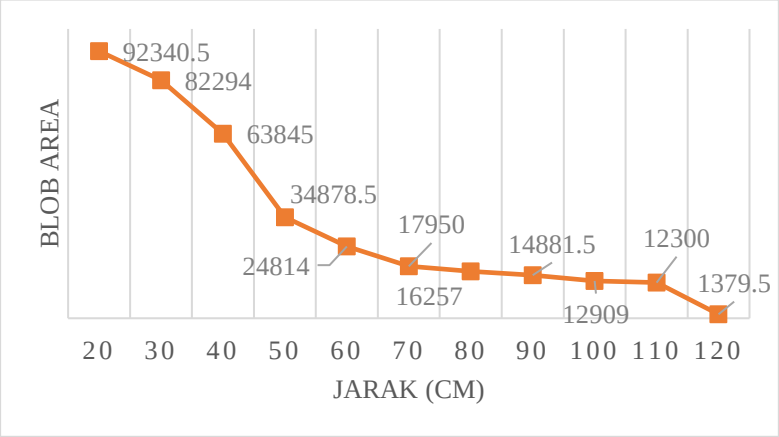
Basic Diagram Single Board Computer Lattepanda



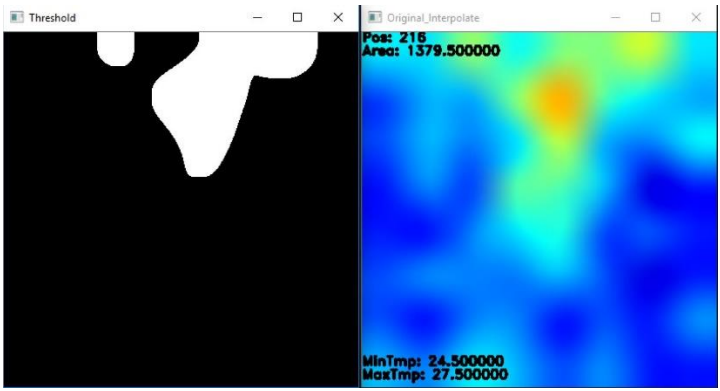
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN B

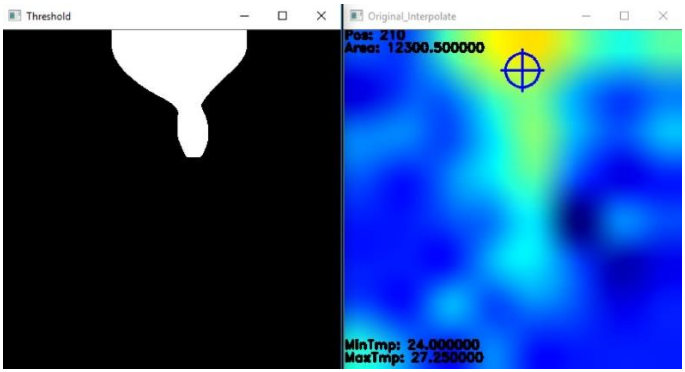
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 24ºC, Lux: 2, Target Menggunakan Celana Pendek



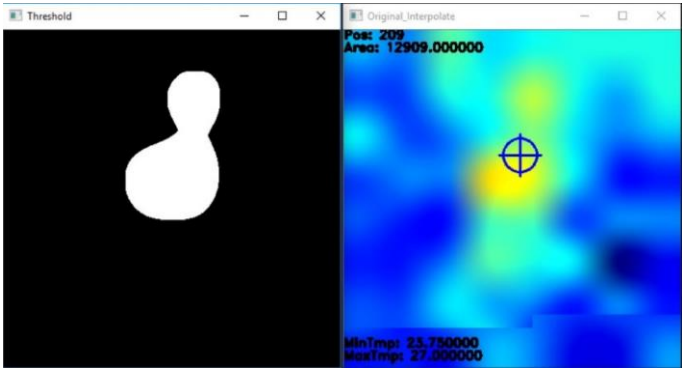
- 120 CM



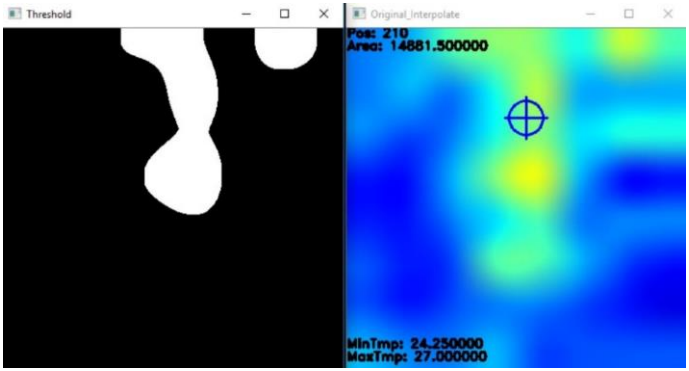
- 110 CM



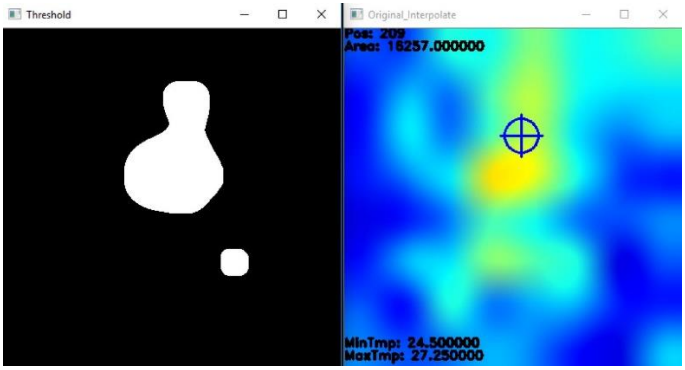
- 100 CM



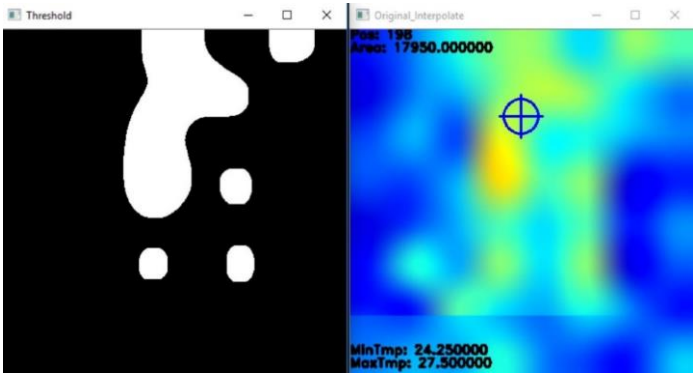
- 90 CM



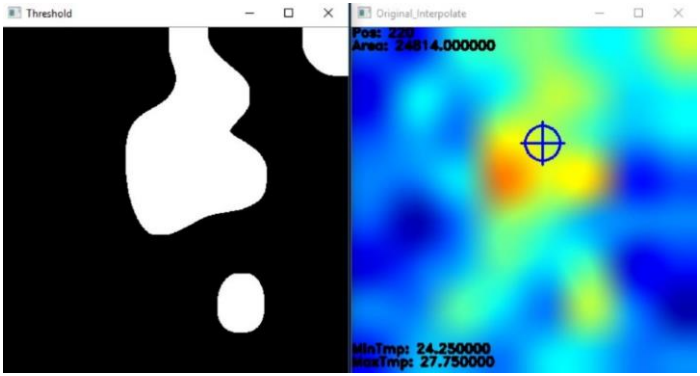
- 80 CM



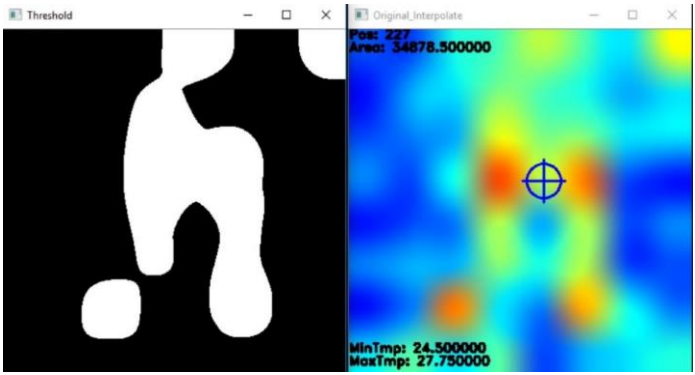
- 70 CM



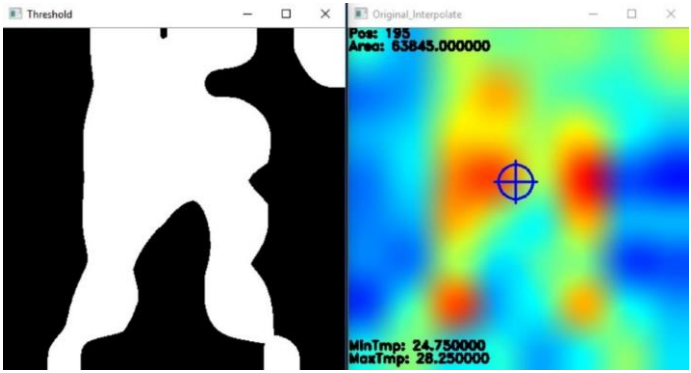
- 60 CM



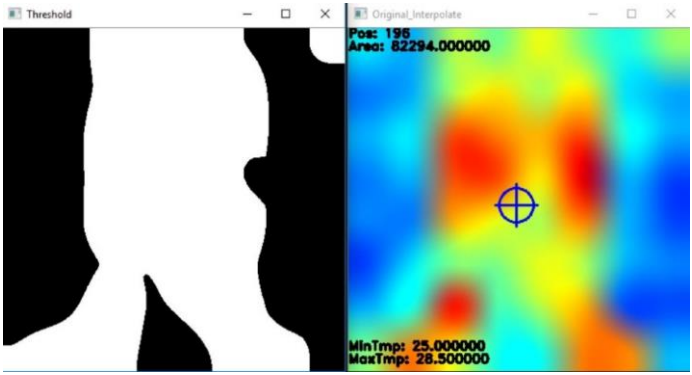
- 50 CM



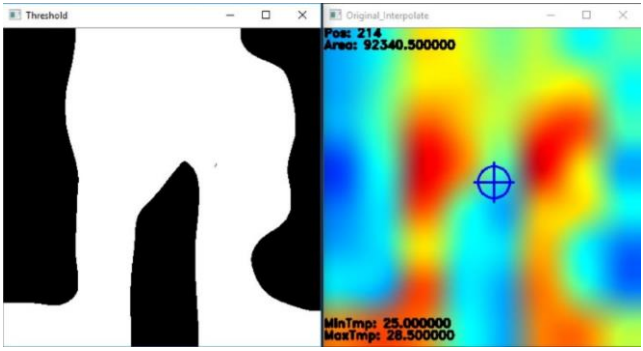
- 40 CM



- 30 CM



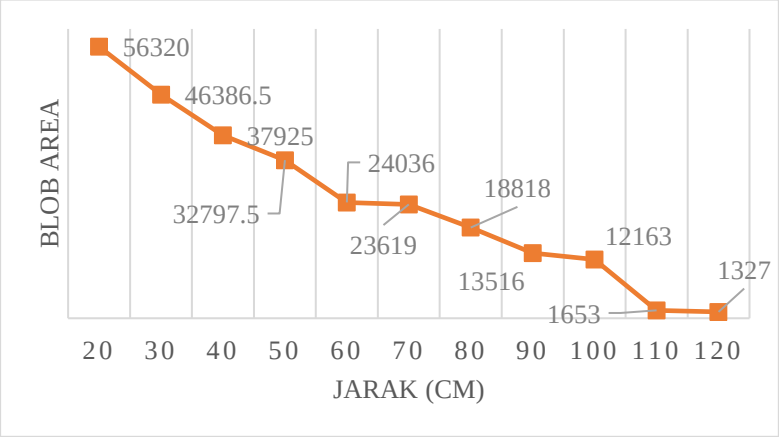
- 20 CM


Threshold
Original_Interpolate
Pos: 214
Area: 92340.500000
MinTmp: 25.000000
MaxTmp: 28.500000

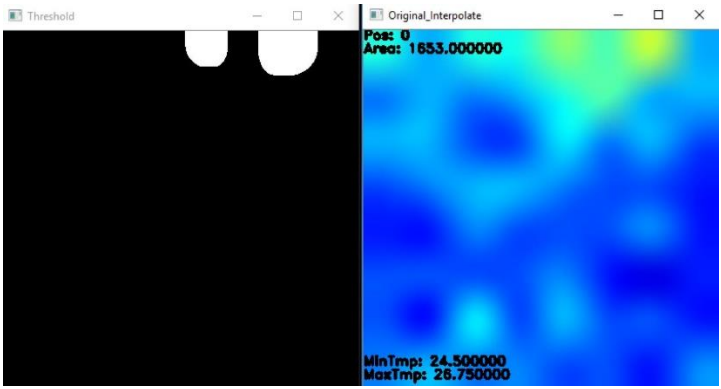
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN C

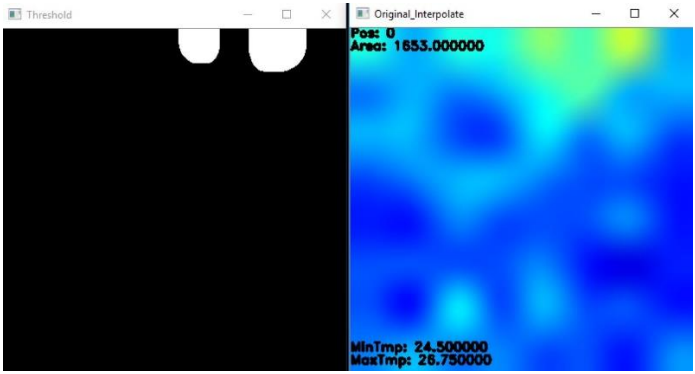
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 24ºC, Lux: 2, Target Menggunakan Jeans



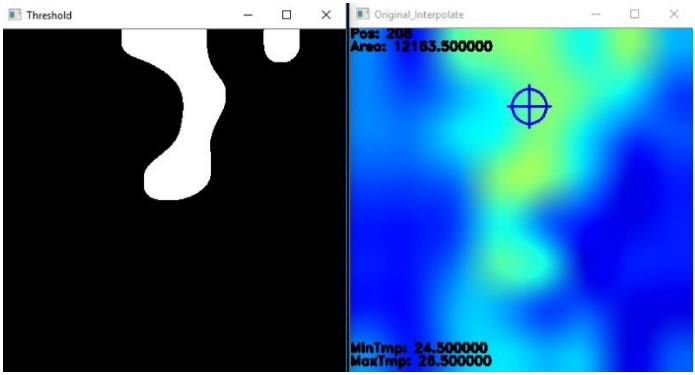
- 120 CM



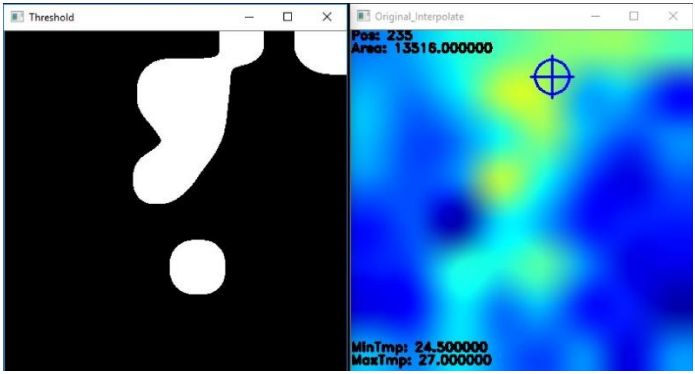
- 110 CM



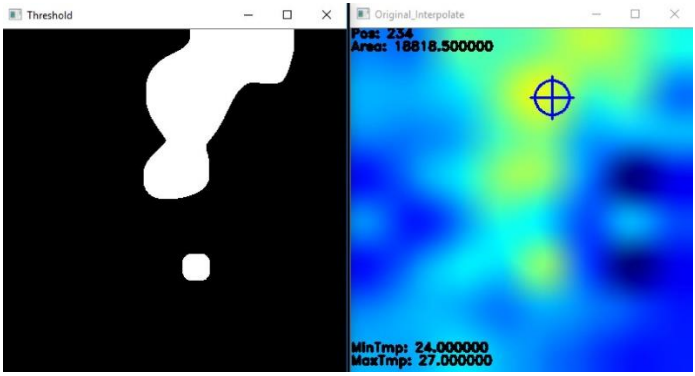
- 100 CM



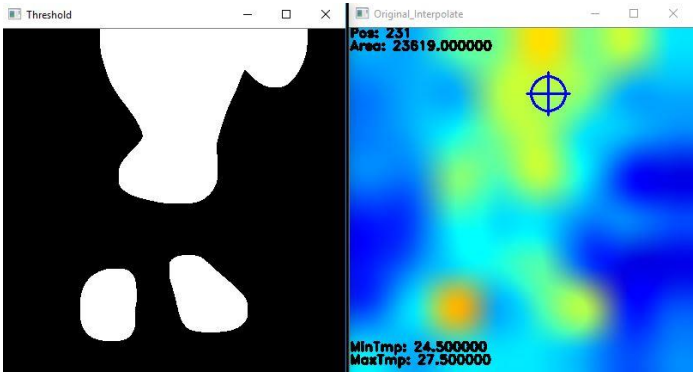
- 90 CM



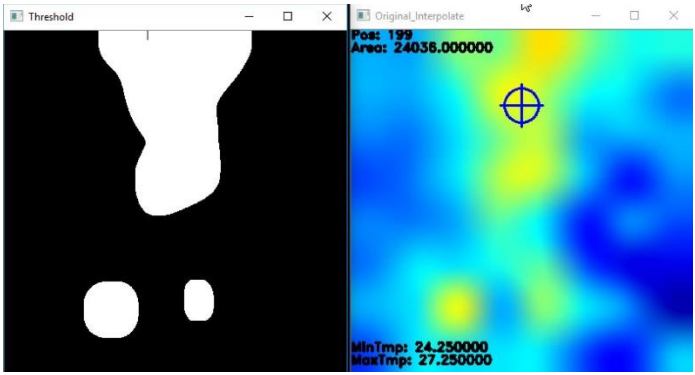
- 80 CM



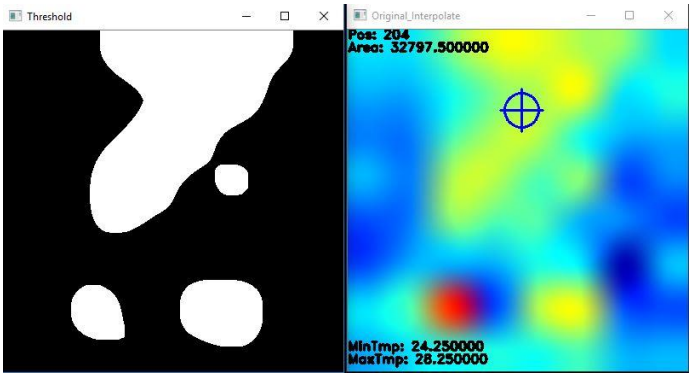
- 70 CM



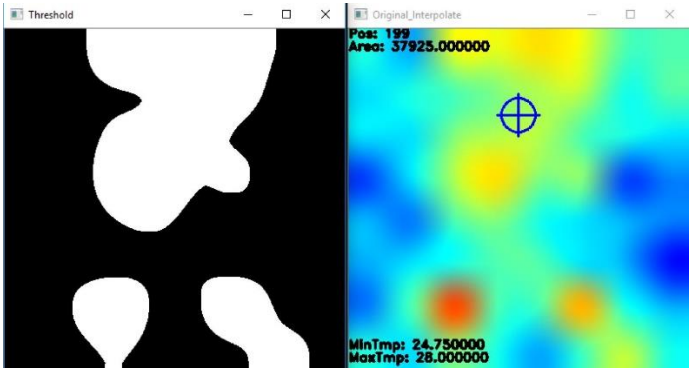
- 60 CM



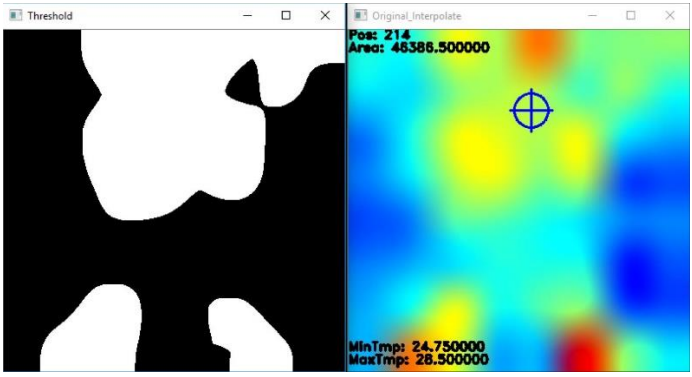
- 50 CM



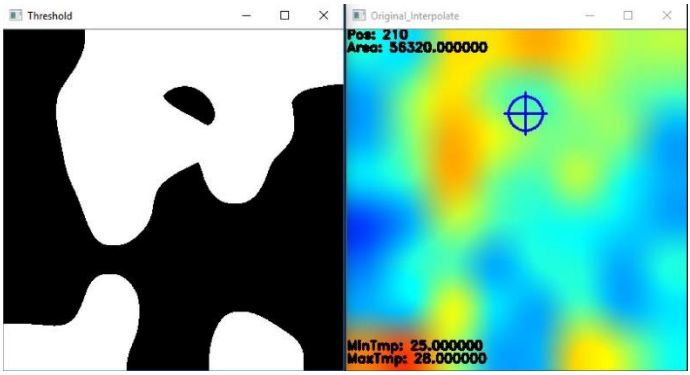
- 40 CM



- 30 CM



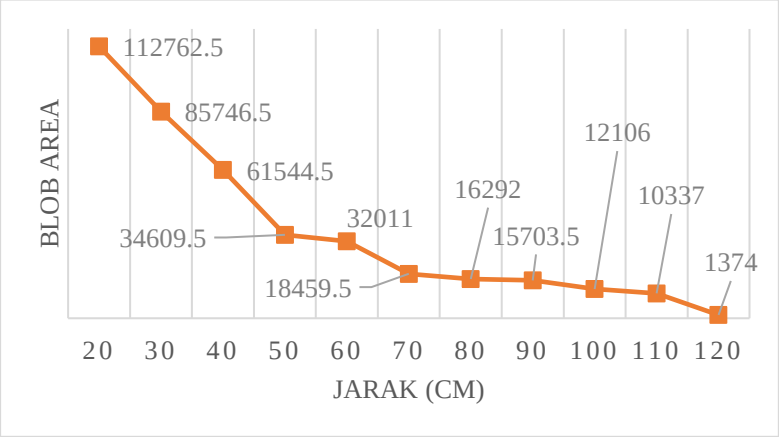
- 20 CM


Threshold
Original_Interpolate
Pos: 210
Area: 56320.000000
MinTmp: 25.000000
MaxTmp: 28.000000

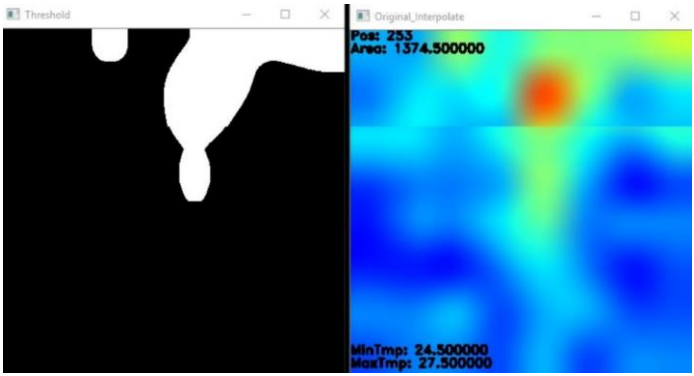
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN D

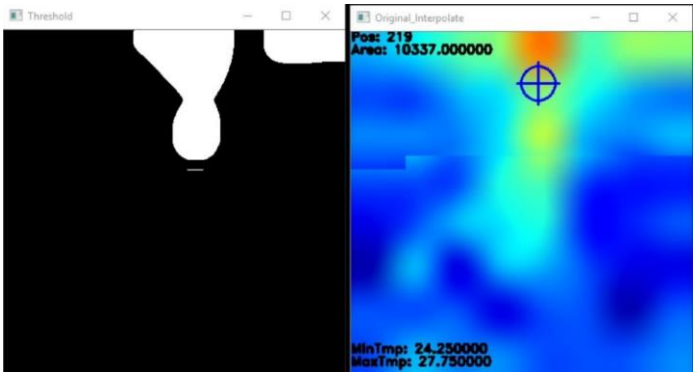
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 24ºC, Lux: 183, Target Menggunakan Celana Pendek



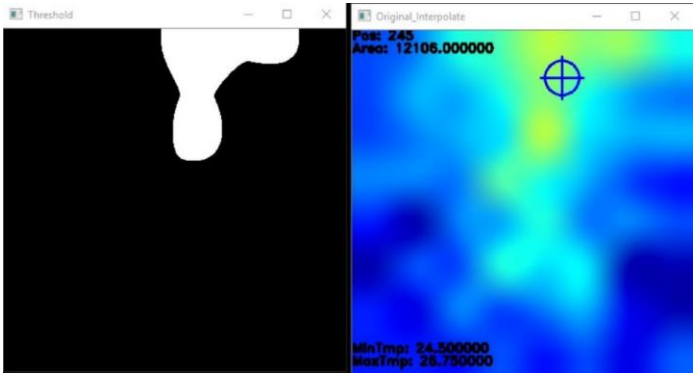
- 120 CM



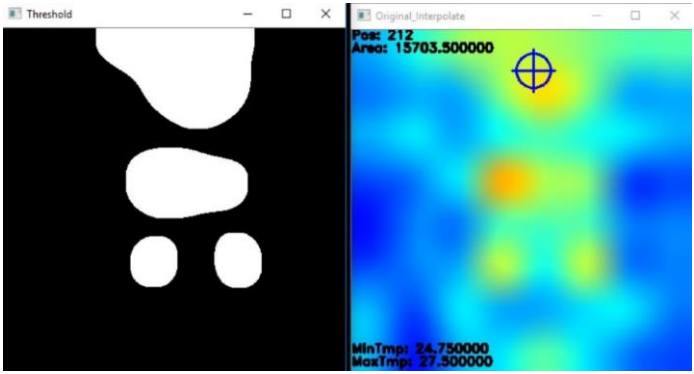
- 110 CM



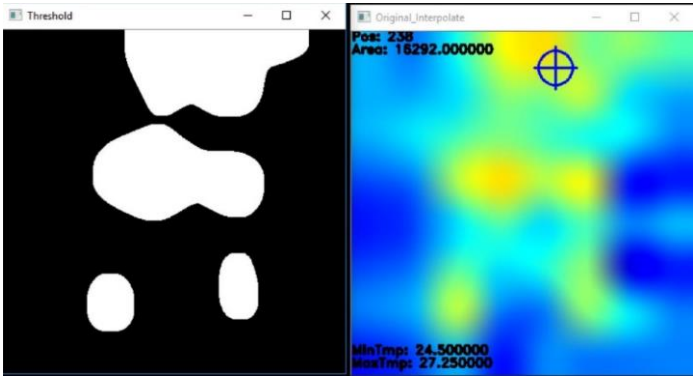
- 100 CM



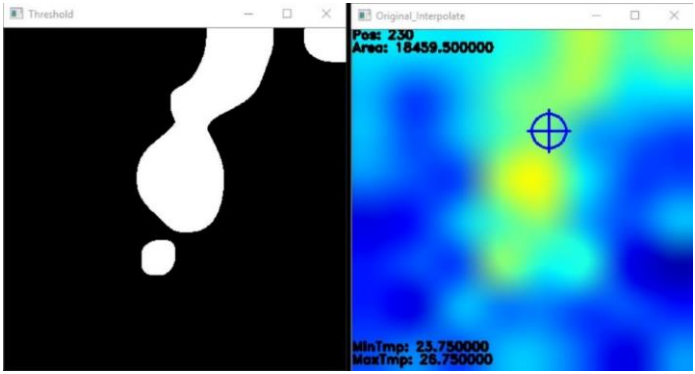
- 90 CM



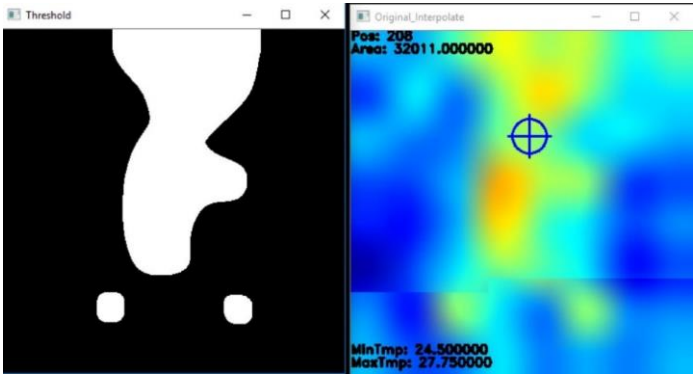
- 80 CM



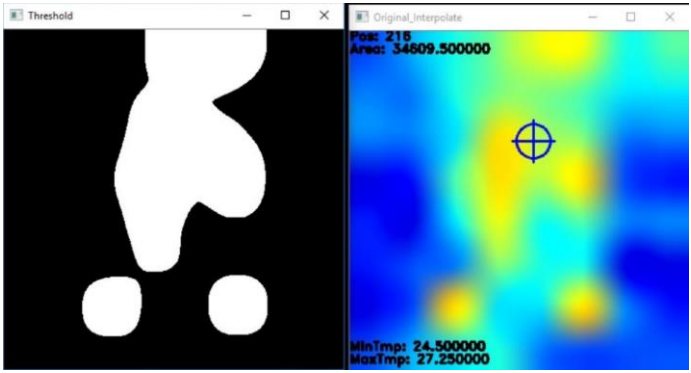
- 70 CM



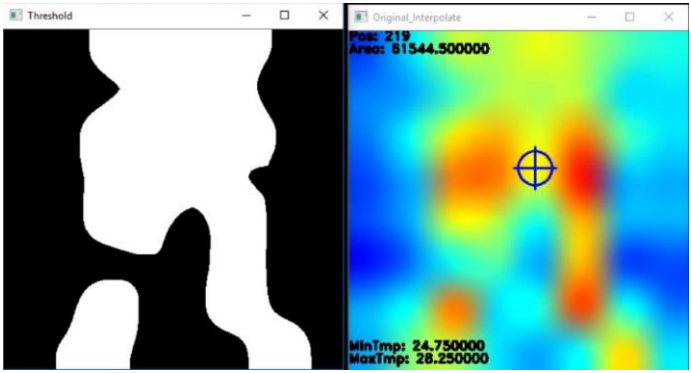
- 60 CM



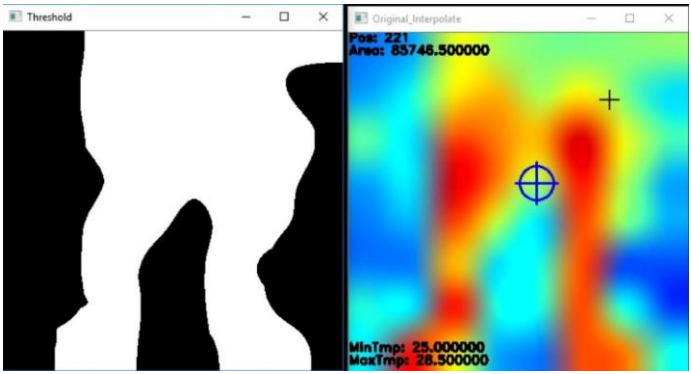
- 50 CM



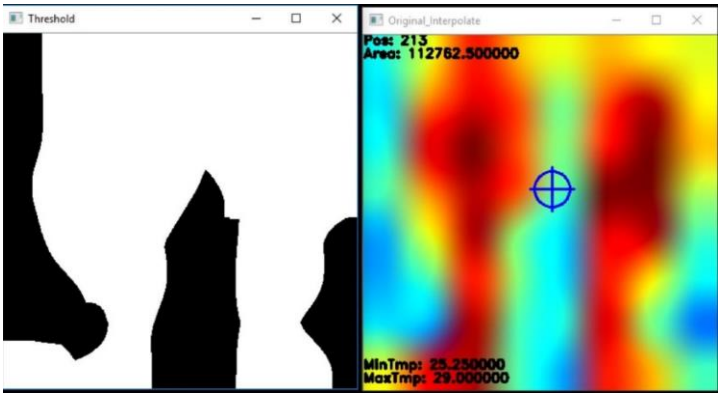
- 40 CM



- 30 CM



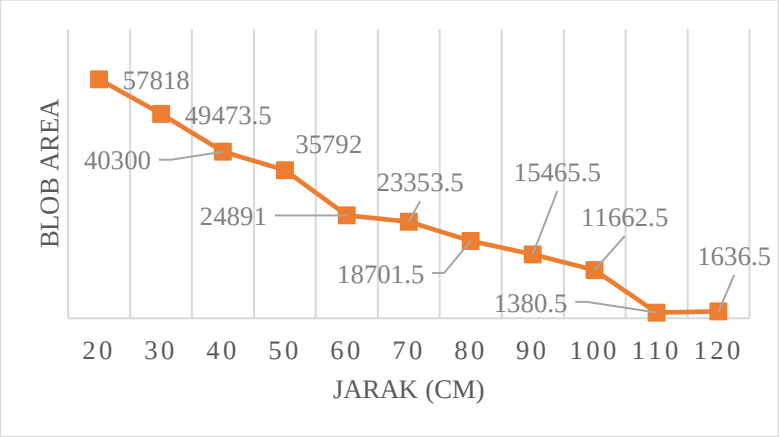
- 20 CM


Threshold
Original_Interpolate
Pos: 213
Area: 112762.500000
MinTmp: 25.250000
MaxTmp: 29.000000

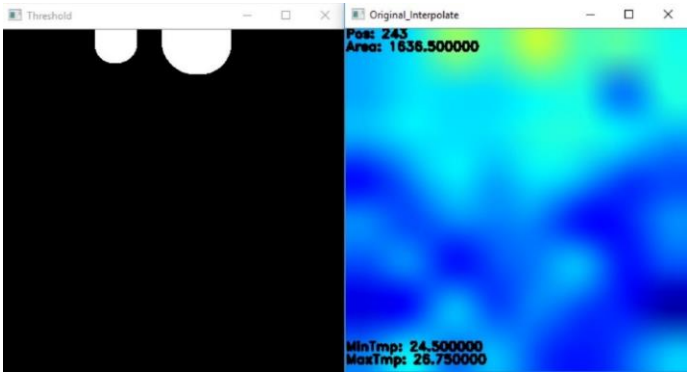
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN E

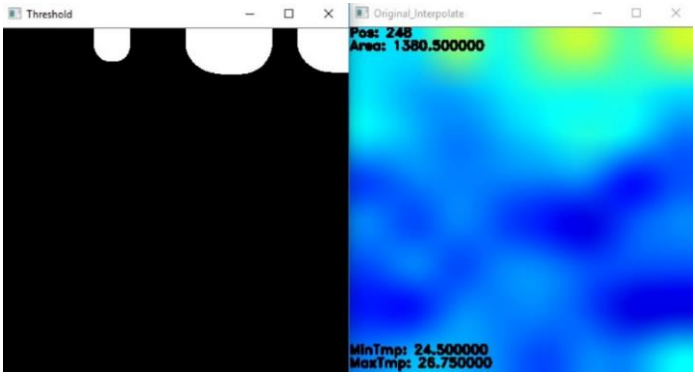
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 24ºC, Lux: 183, Target Menggunakan Jeans



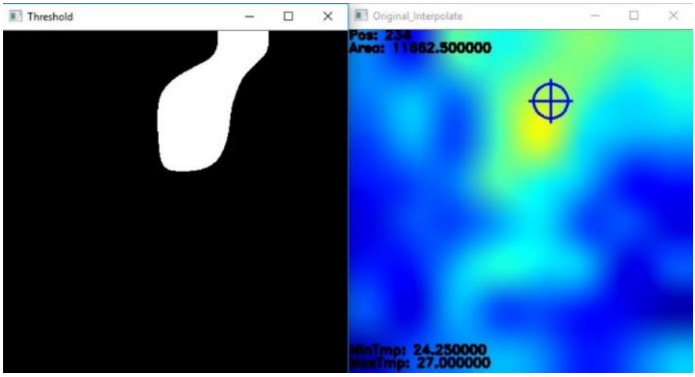
- 120 CM



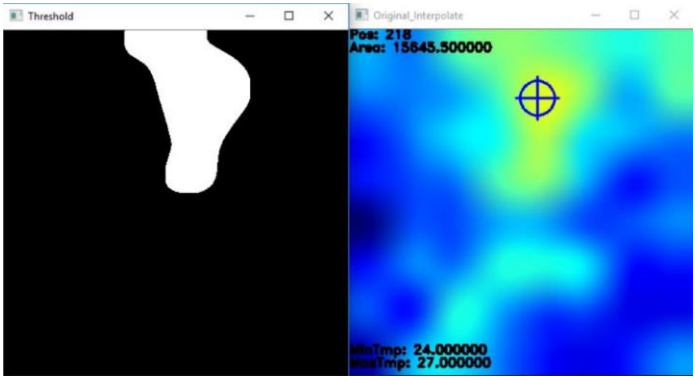
- 110 CM



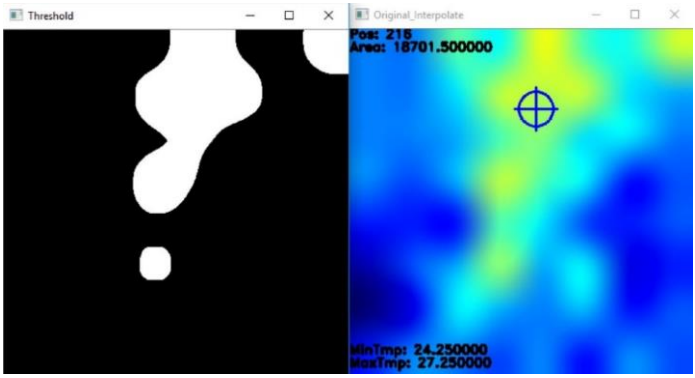
- 100 CM



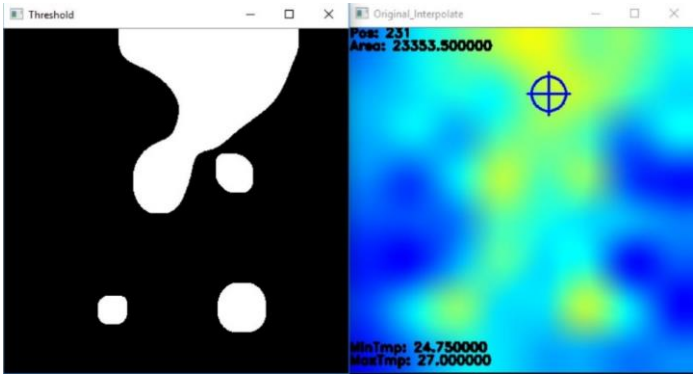
- 90 CM



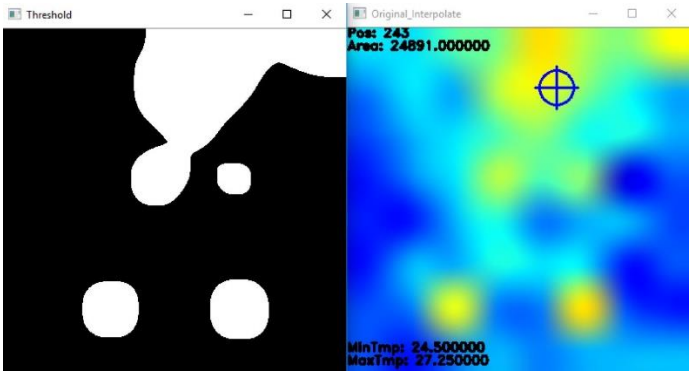
- 80 CM



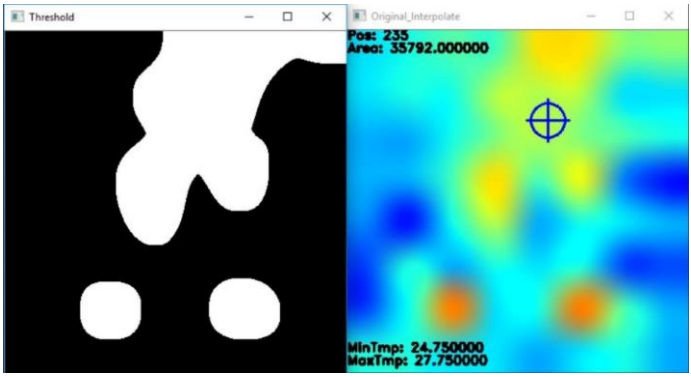
- 70 CM



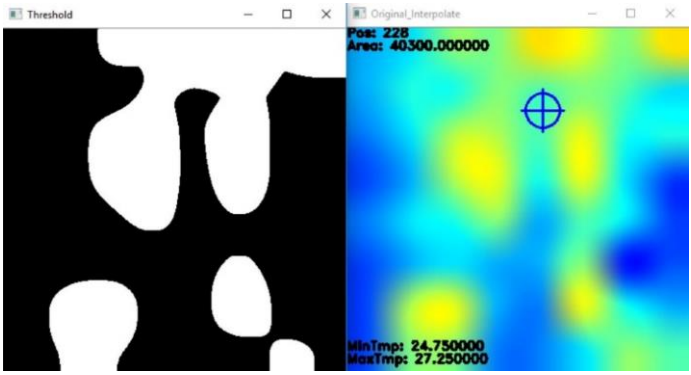
- 60 CM



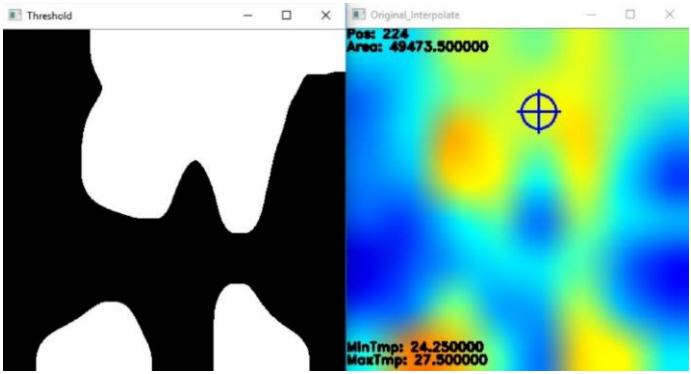
- 50 CM



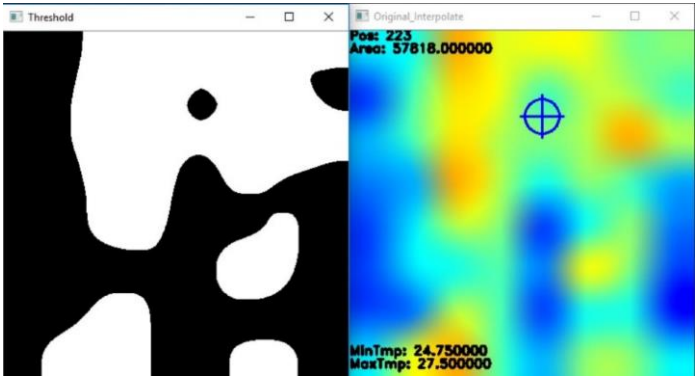
- 40 CM



- 30 CM



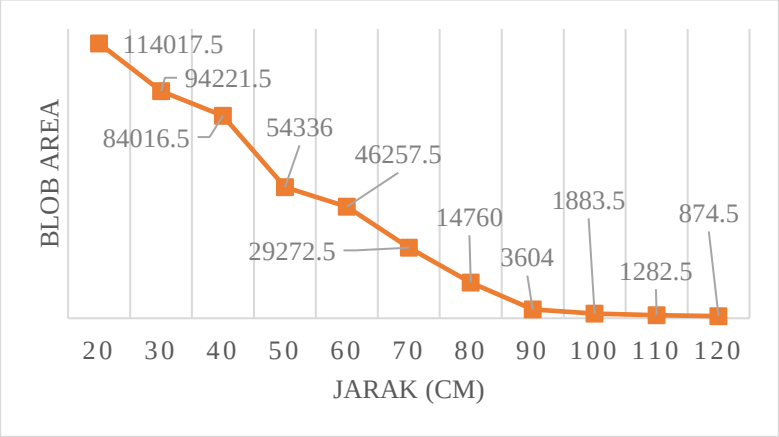
- 20 CM


Threshold
Original_Interpolate
Pos: 223
Area: 57818.000000
MinTmp: 24.750000
MaxTmp: 27.500000

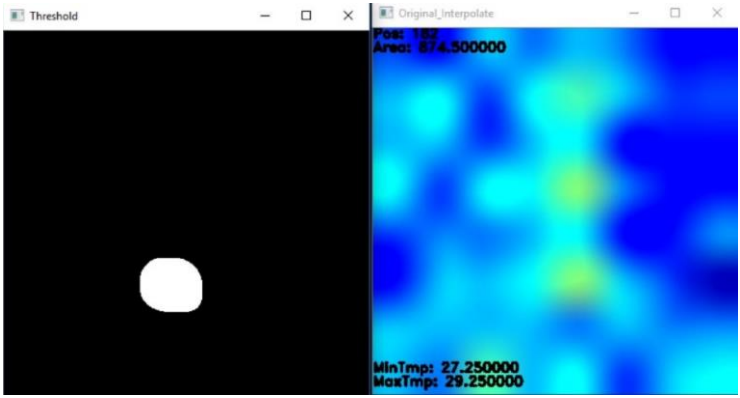
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN F

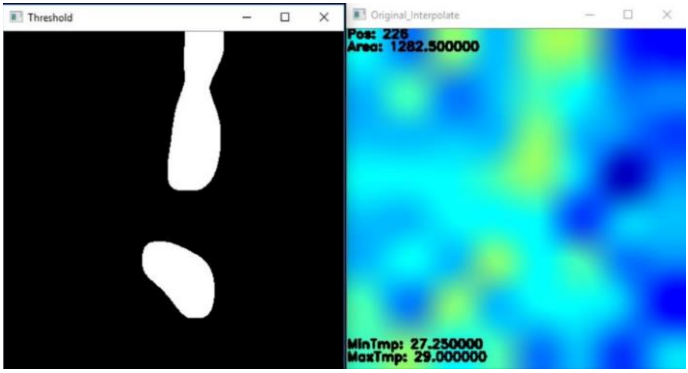
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 27ºC, Lux: 24, Target Menggunakan Celana Pendek



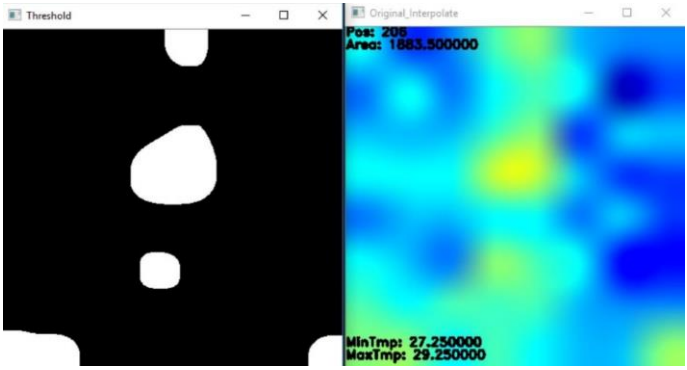
- 120 CM



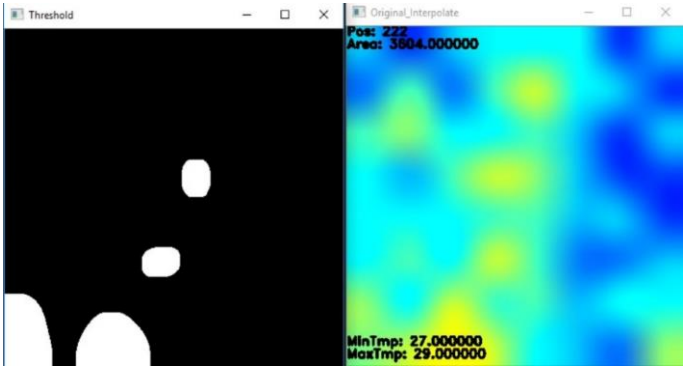
- 110 CM



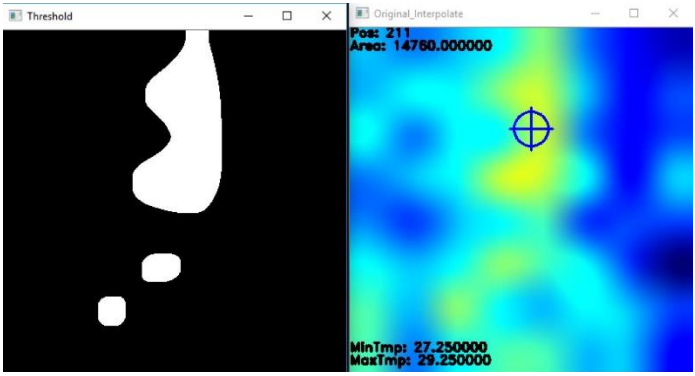
- 100 CM



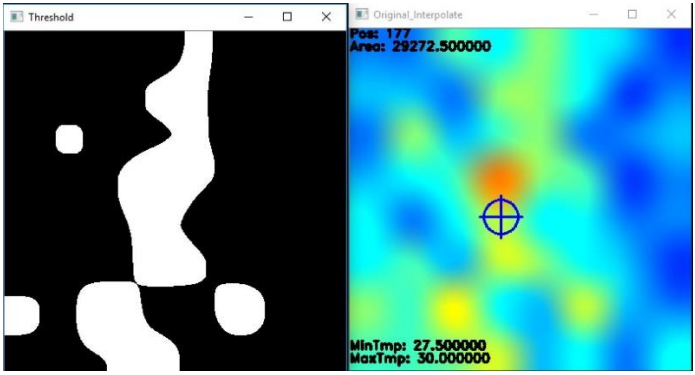
- 90 CM



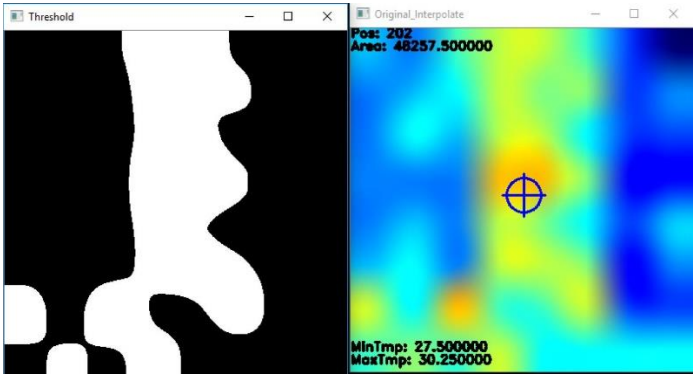
- 80 CM



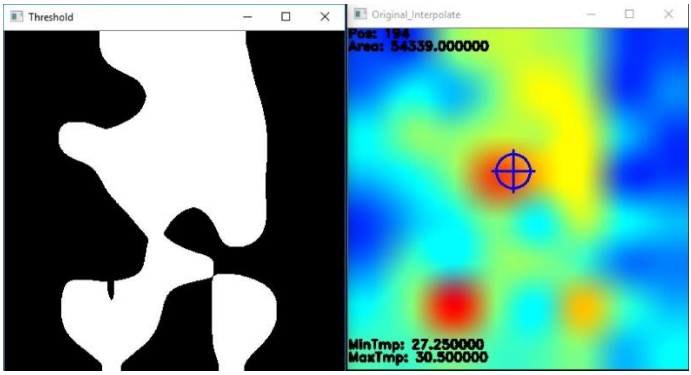
- 70 CM



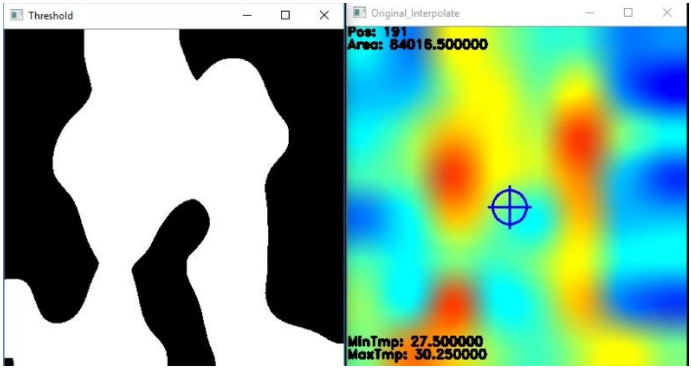
- 60 CM



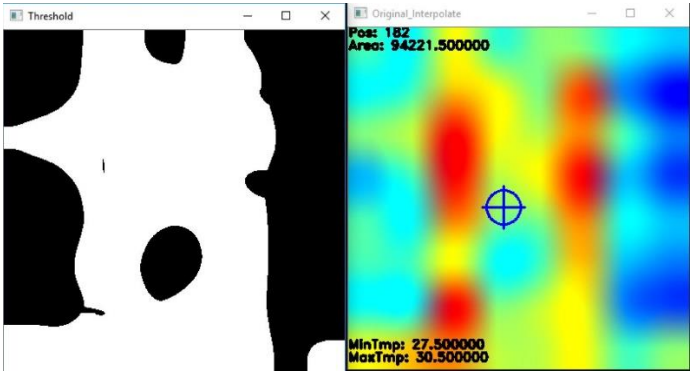
- 50 CM



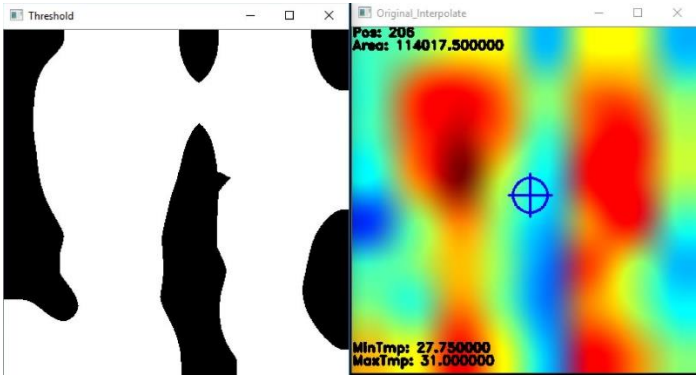
- 40 CM



- 30 CM



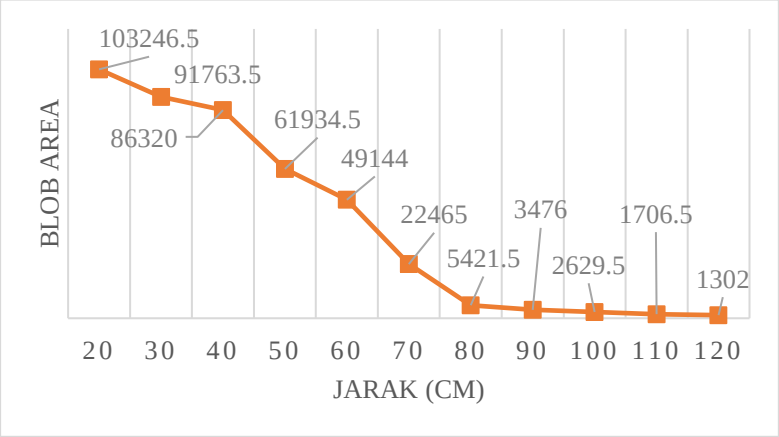
- 20 CM


Threshold
Original_Interpolate
Pos: 205
Area: 114017.500000
MinTmp: 27.750000
MaxTmp: 31.000000

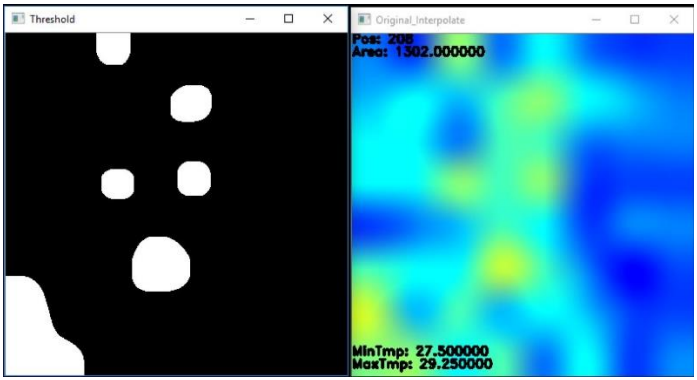
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN G

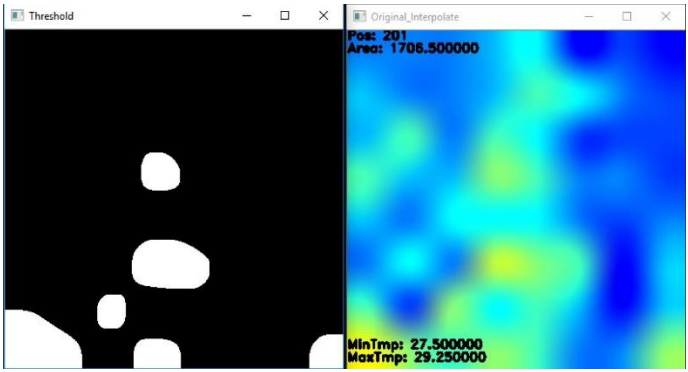
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 27ºC, Lux: 24, Target Menggunakan Jeans



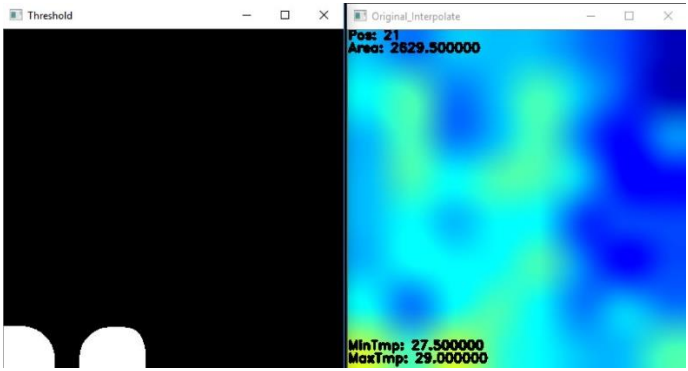
- 120 CM



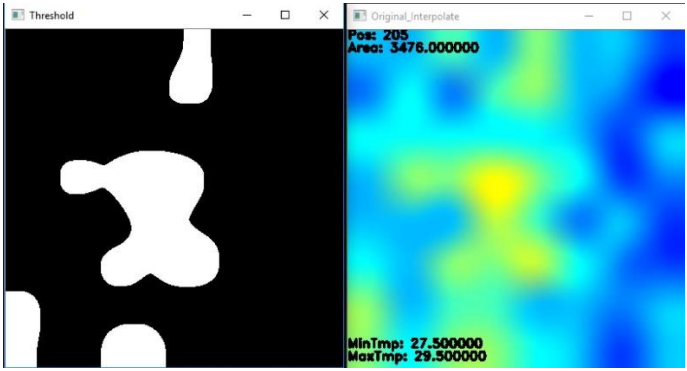
- 110 CM



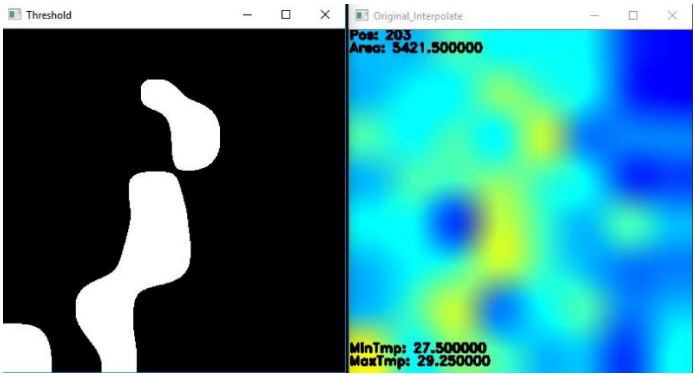
- 100 CM



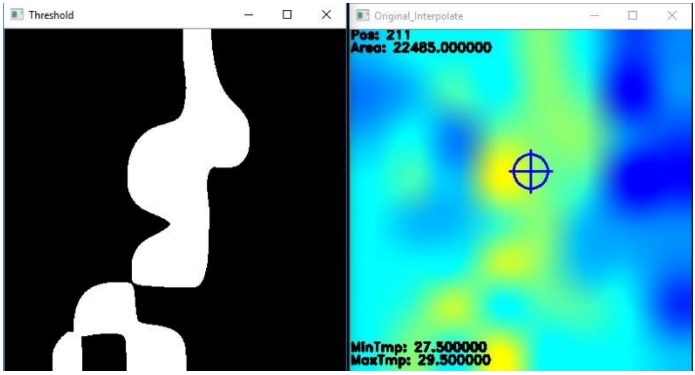
- 90 CM



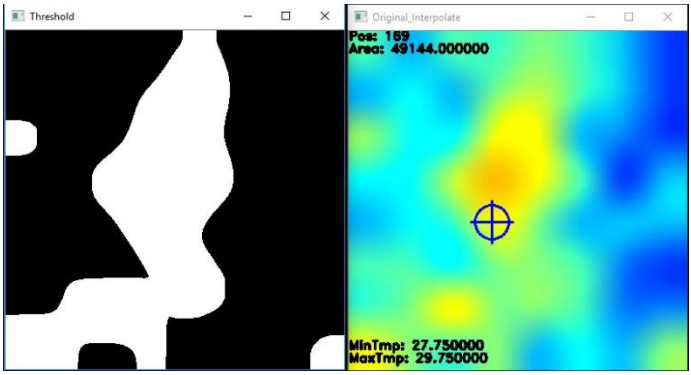
- 80 CM



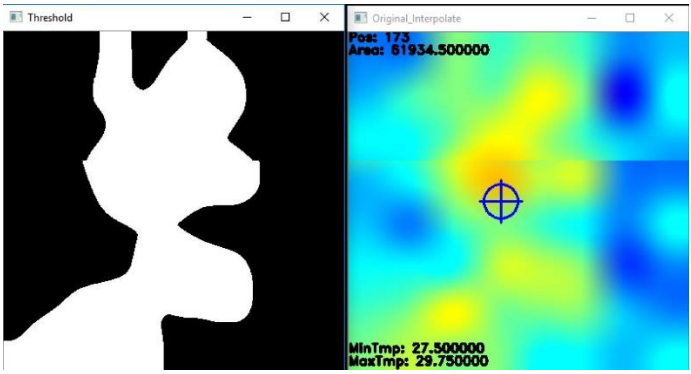
- 70 CM



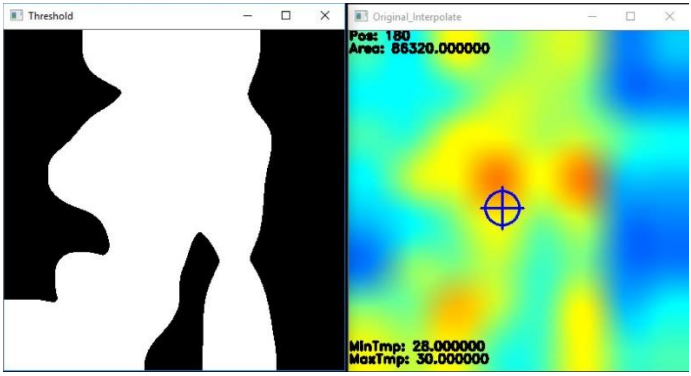
- 60 CM



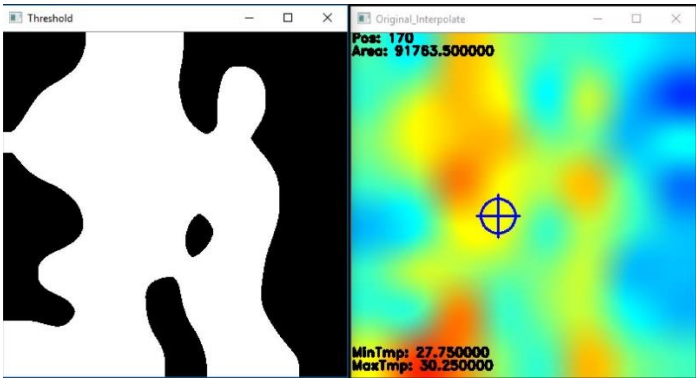
- 50 CM



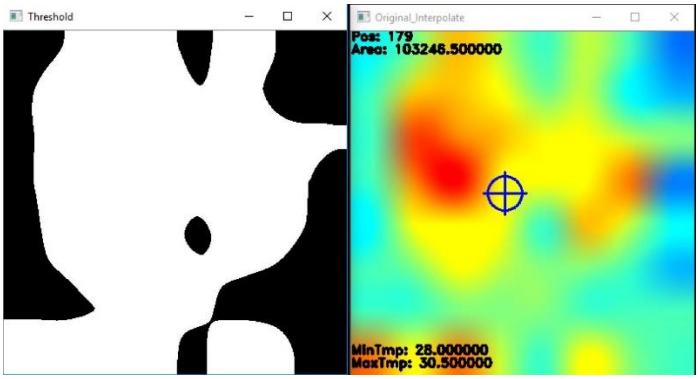
- 40 CM



- 30 CM



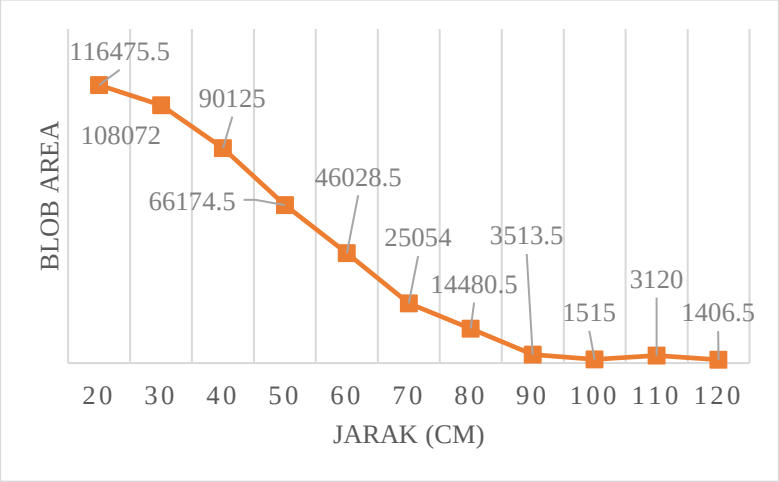
- 20 CM


Threshold
Original_Interpolate
Pos: 179
Area: 103248.500000
MinTmp: 28.000000
MaxTmp: 30.500000

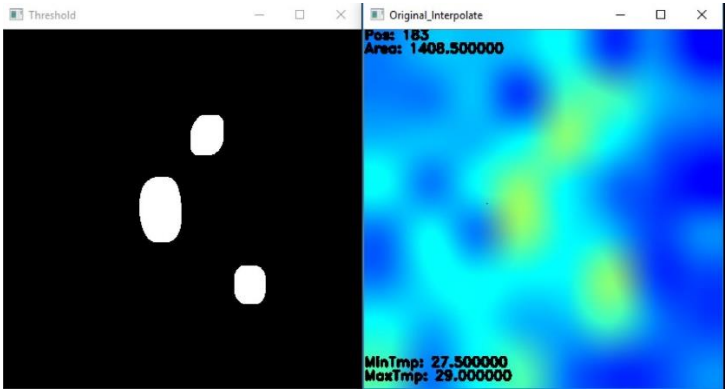
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN H

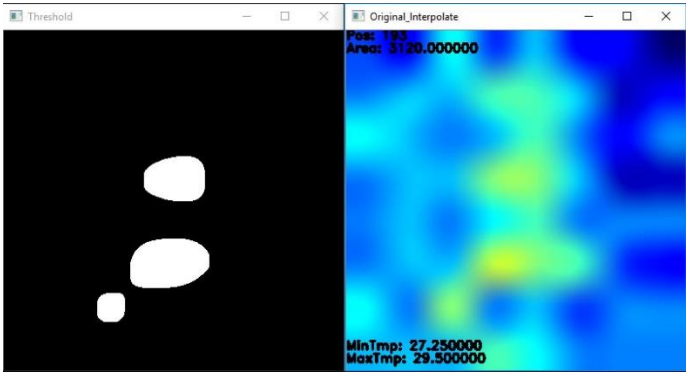
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 27ºC, Lux: 169, Target Menggunakan Celana Pendek



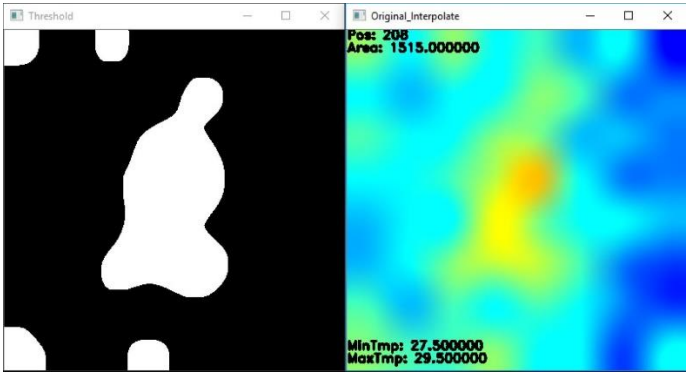
- 120 CM



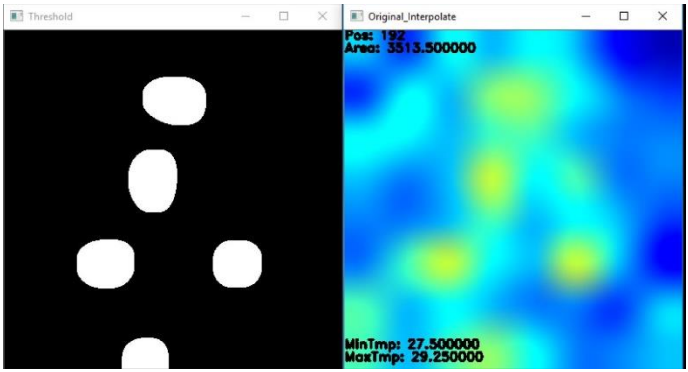
- 110 CM



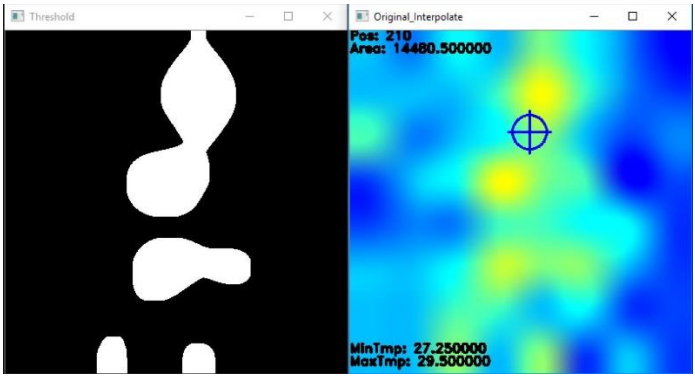
- 100 CM



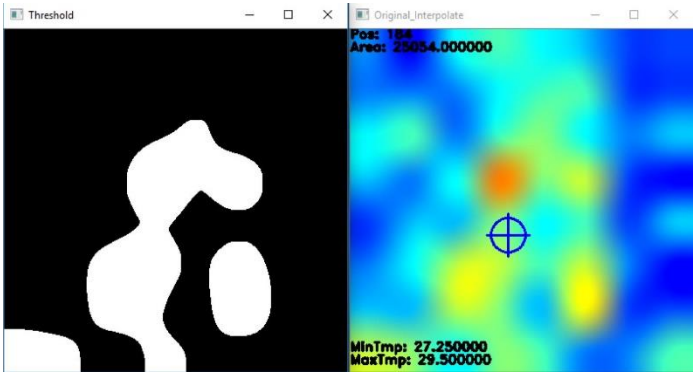
- 90 CM



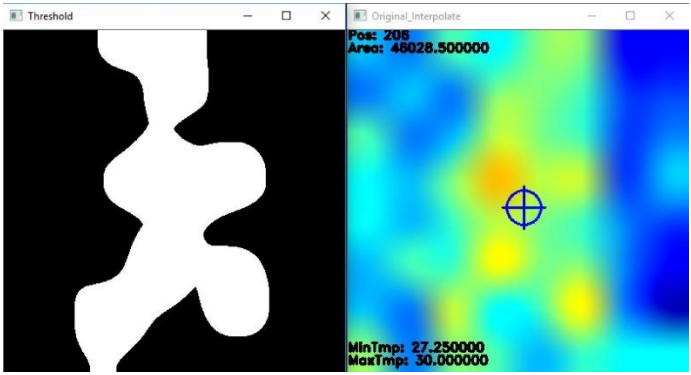
- 80 CM



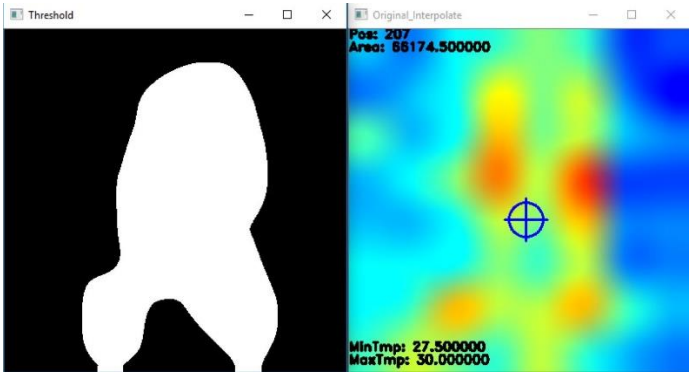
- 70 CM



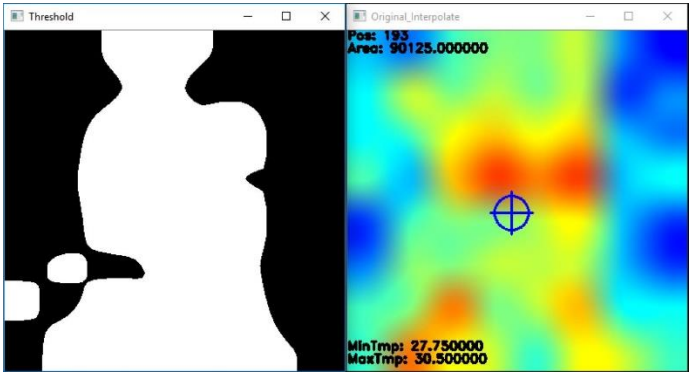
- 60 CM



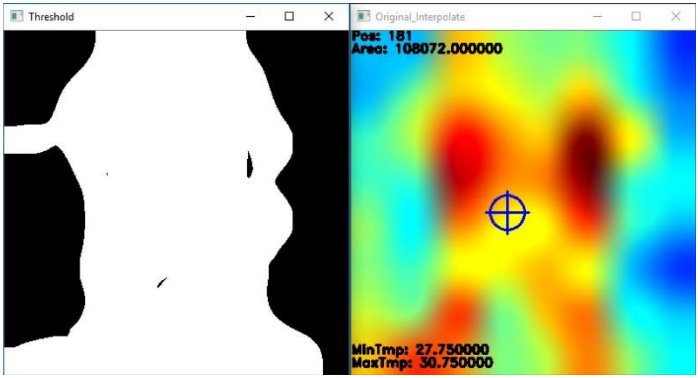
- 50 CM



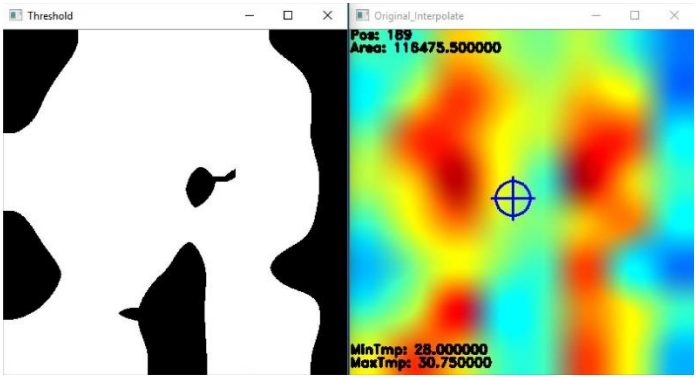
- 40 CM



- 30 CM



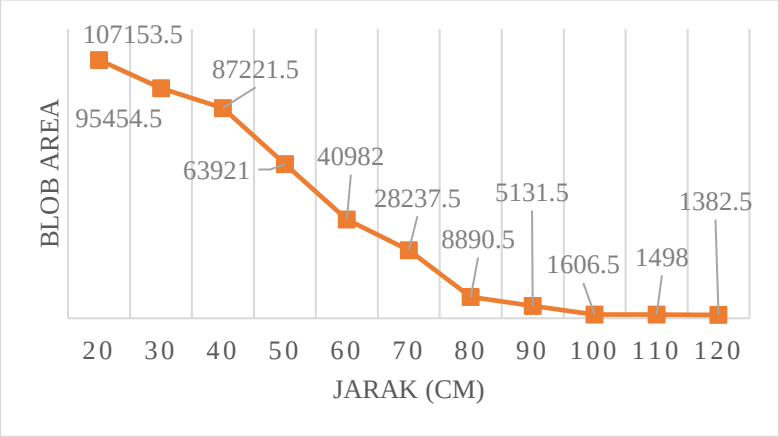
- 20 CM


Threshold
Original_Interpolate
Pos: 189
Area: 116475.500000
MinTmp: 28.000000
MaxTmp: 30.750000

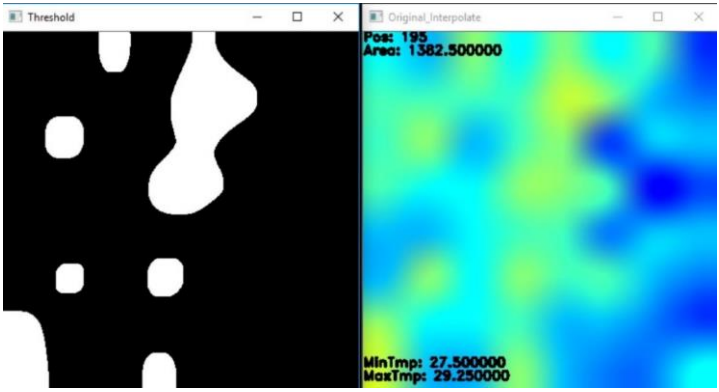
-----Halaman ini sengaja dikosongkan-----

LAMPIRAN I

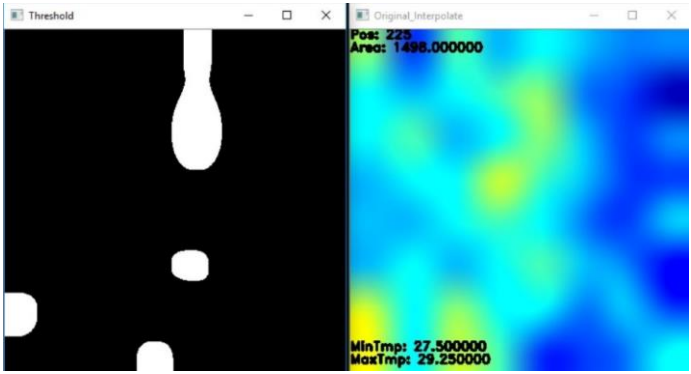
Dokumentasi Pengujian Deteksi Citra Target
Suhu Ruangan: 27ºC, Lux: 169, Target Menggunakan Jeans



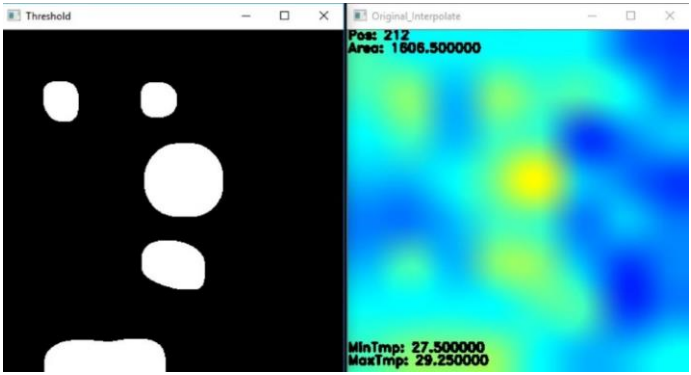
- 120 CM



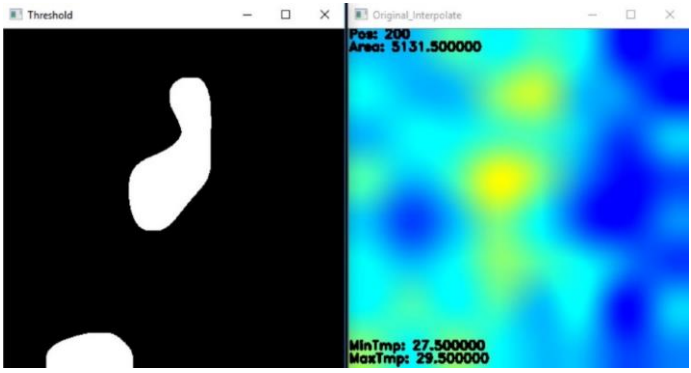
- 110 CM



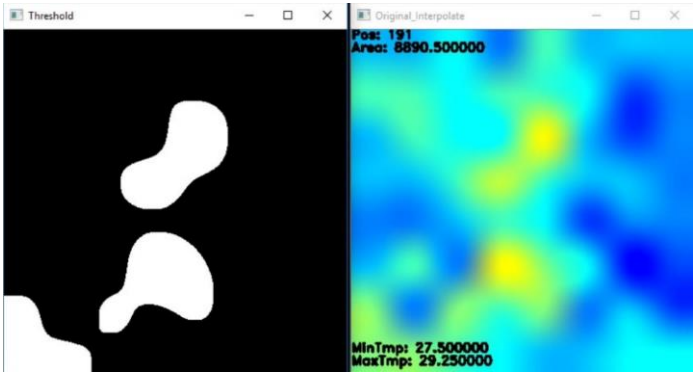
- 100 CM



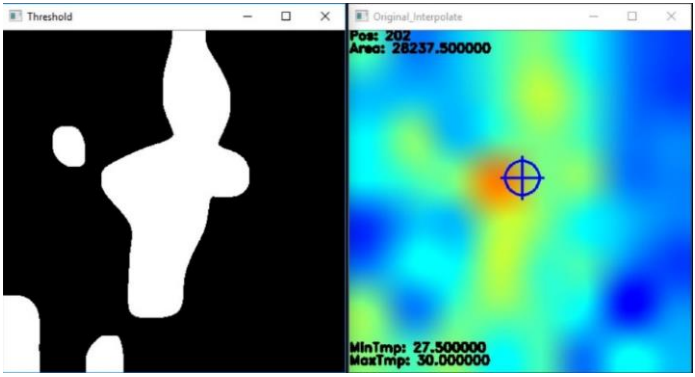
- 90 CM



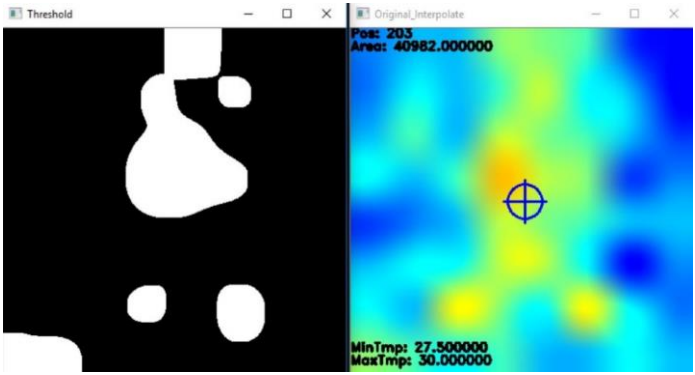
- 80 CM



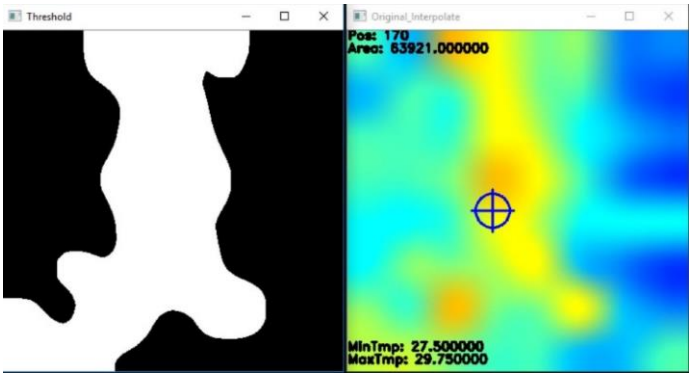
- 70 CM



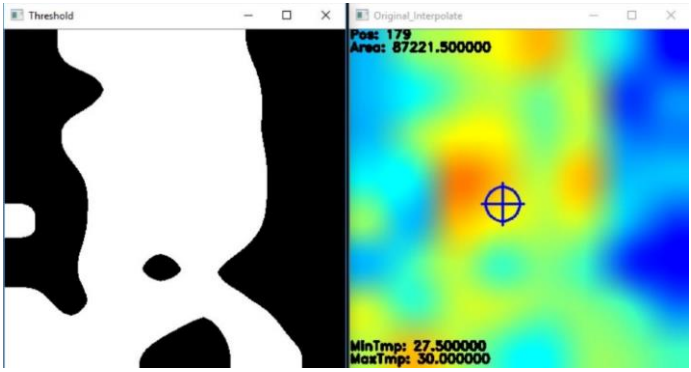
- 60 CM



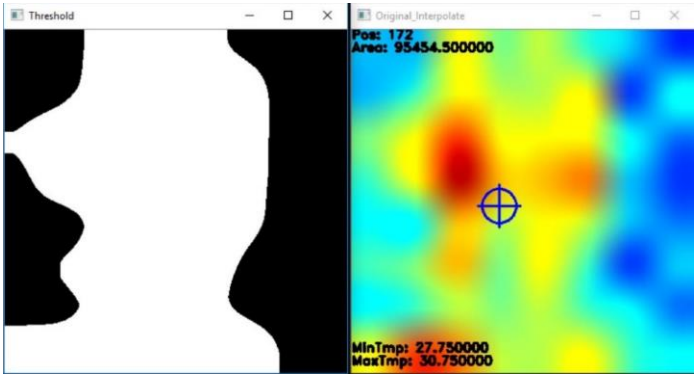
- 50 CM



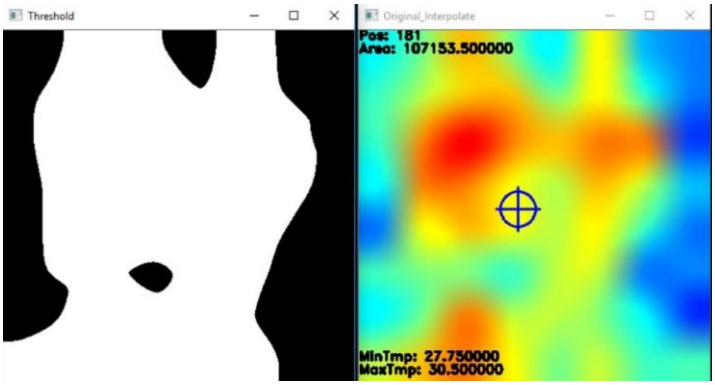
- 40 CM



- 30 CM



- 20 CM


Threshold
Original_Interpolate
Pos: 181
Area: 107153.500000
MinTmp: 27.750000
MaxTmp: 30.500000

-----Halaman ini sengaja dikosongkan-----

LAMPIRAN J

Listing Code Deteksi Target (C++)

```
//#define _CRT_SECURE_NO_WARNINGS
#define _SCL_SECURE_NO_WARNINGS

#include <iostream>
#include <thread>
#include "opencv2/opencv.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/core/core.hpp"
#include <Windows.h>
#include <stdio.h>
#include <sstream>
#include <stdlib.h>
#include <string>
#include <vector>
#include "SerialPort.h"
#include <fstream>

using namespace cv;
using namespace std;
using std::cout;
using std::endl;

char *port_name = "\\\\.\\COM9";
char incomingData[MAX_DATA_LENGTH];

int _countParse, _countParse2, _totalData;
int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
int callibr = 0;
int flag_clrArea = 0;
int safety = 1;
int xx, yy;
```

```
int koordinatX = 0, luasBlob;

const int FRAME_WIDTH = 400;
const int FRAME_HEIGHT = 400;
const int MAX_NUM_OBJECTS = 50;
const int MIN_OBJECT_AREA = 15*15;
const int MAX_OBJECT_AREA = FRAME_HEIGHT * FRAME_WIDTH;
const string trackbarWindowName = "Trackbars";

float _data[63], _dataNorm[70], tempMin = 27, tempMax =
31;
double _area = 0;
double max_temp, min_temp;

vector<string> split(string s, string delimiter)
{
	size_t pos_start = 0, pos_end, delim_len =
delimiter.length();
	string token;
	vector <string> res;

	while ((pos_end = s.find(delimiter, pos_start))
!= string::npos)
	{
		token = s.substr(pos_start, pos_end -
pos_start);
		pos_start = pos_end + delim_len;
		res.push_back(token);
	}

	res.push_back(s.substr(pos_start));
	return res;
}
void on_trackbar(int, void*)
{

}
string intToString(int number)
{
	std::stringstream ss;
```

```
	ss << number;
	return ss.str();
}
void createTrackbars()
{
	char TrackbarName[50];
	namedWindow(trackbarWindowName, 0);
	sprintf_s(TrackbarName, "H_MIN", H_MIN);
	sprintf_s(TrackbarName, "H_MAX", H_MAX);
	sprintf_s(TrackbarName, "S_MIN", S_MIN);
	sprintf_s(TrackbarName, "S_MAX", S_MAX);
	sprintf_s(TrackbarName, "V_MIN", V_MIN);
	sprintf_s(TrackbarName, "V_MAX", V_MAX);
	createTrackbar("H_MIN", trackbarWindowName,
&H_MIN, H_MAX, on_trackbar);
	createTrackbar("H_MAX", trackbarWindowName,
&H_MAX, H_MAX, on_trackbar);
	createTrackbar("S_MIN", trackbarWindowName,
&S_MIN, S_MAX, on_trackbar);
	createTrackbar("S_MAX", trackbarWindowName,
&S_MAX, S_MAX, on_trackbar);
	createTrackbar("V_MIN", trackbarWindowName,
&V_MIN, V_MAX, on_trackbar);
	createTrackbar("V_MAX", trackbarWindowName,
&V_MAX, V_MAX, on_trackbar);
}
void drawObject(int x, int y, Mat &frame)
{
	circle(frame, Point(x, y), 20, Scalar(255, 0, 0),
2);
	if (y - 25 > 0)
		line(frame, Point(x, y), Point(x, y - 25),
Scalar(255, 0, 0), 2);
	else line(frame, Point(x, y), Point(x, 0),
Scalar(255, 0, 0), 2);
	if (y + 25 < FRAME_HEIGHT)
		line(frame, Point(x, y), Point(x, y + 25),
Scalar(255, 0, 0), 2);
	else line(frame, Point(x, y), Point(x,
FRAME_HEIGHT), Scalar(255, 0, 0), 2);
```

```
	if (x - 25 > 0)
		line(frame, Point(x, y), Point(x - 25, y),
Scalar(255, 0, 0), 2);
	else line(frame, Point(x, y), Point(0, y),
Scalar(255, 0, 0), 2);
	if (x + 25 < FRAME_WIDTH)
		line(frame, Point(x, y), Point(x + 25, y),
Scalar(255, 0, 0), 2);
	else line(frame, Point(x, y), Point(FRAME_WIDTH,
y), Scalar(255, 0, 0), 2);
	koordinatX = x;
}
void morphOps(Mat &thresh)
{
	Mat erodeElement =
getStructuringElement(MORPH_RECT, Size(3, 3));
	Mat dilateElement =
getStructuringElement(MORPH_RECT, Size(8, 8));
	erode(thresh, thresh, erodeElement);
	erode(thresh, thresh, erodeElement);
	dilate(thresh, thresh, dilateElement);
	dilate(thresh, thresh, dilateElement);
}
void trackFilteredObject(int &x, int &y, Mat threshold,
Mat &image)
{
	ofstream myfile;
	Mat temp;
	threshold.copyTo(temp);
	vector< vector<Point> > contours;
	vector<Vec4i> hierarchy;
	findContours(temp, contours, hierarchy,
RETR_CCOMP, CHAIN_APPROX_SIMPLE);

	double refArea = 0;
	bool objectFound = false;

	if (hierarchy.size() > 0)
	{
		int numObjects = hierarchy.size();
```

```
            if (numObjects < MAX_NUM_OBJECTS)
            {
                  for (int index = 0; index >= 0;
index = hierarchy[index][0])
                  {
                        Moments moment =
moments((cv::Mat)contours[index]);
                        double area = moment.m00;
                        _area = area;
                        if (area > MIN_OBJECT_AREA
&& area < MAX_OBJECT_AREA && area > refArea)
                        {
                              x = moment.m10 /
area;
                              y = moment.m01 /
area;
                              objectFound = true;
                              int nDummy =
(int)(area * 1000);
                              refArea = area;
                        }
                        else objectFound = false;
                  }
                  if (objectFound == true)
                  {
                        drawObject(x, y, image);
                  }
                  flag_clrArea = 0;
                  myfile.open("data.txt");
                  myfile << koordinatX;
                  myfile << '#';
                  myfile << _area;
                  myfile.close();
                  putText(image, "Pos: " +
intToString(koordinatX), Point(0, 10), 2, 0.5,
Scalar(0, 0, 0), 2);
                  putText(image, "Area: " +
to_string(_area), Point(0, 25), 2, 0.5, Scalar(0, 0,
0), 2);
```

```
                        putText(image, "MaxTmp: " +
to_string(max_temp), Point(0, 390), 2, 0.5, Scalar(0,
0, 0), 2);
                        putText(image, "MinTmp: " +
to_string(min_temp), Point(0, 375), 2, 0.5, Scalar(0,
0, 0), 2);
                }
                else
                {
                        if (flag_clrArea > 11)
                        {
                                _area = 0;
                                flag_clrArea = 12;
                        }
                        flag_clrArea++;
                        myfile.open("data.txt");
                        myfile << koordinatX;
                        myfile << '#';
                        myfile << _area;
                        myfile.close();
                        putText(image, "Pos : " +
intToString(koordinatX), Point(0, 10), 2, 0.5,
Scalar(0, 0, 0), 2);
                        putText(image, "Area : " +
to_string(_area), Point(0, 25), 2, 0.5, Scalar(0, 0,
0), 2);
                        putText(image, "MaxTmp: " +
to_string(max_temp), Point(0, 390), 2, 0.5, Scalar(0,
0, 0), 2);
                        putText(image, "MinTmp: " +
to_string(min_temp), Point(0, 375), 2, 0.5, Scalar(0,
0, 0), 2);
                }

        }
        else
        {
                if (flag_clrArea > 11)
                {
                        _area = 0;
```

```
                        flag_clrArea = 12;
                }
                flag_clrArea++;
                myfile.open("data.txt");
                myfile << koordinatX;
                myfile << '#';
                myfile << _area;
                myfile.close();
                putText(image, "Pos : " +
intToString(koordinatX), Point(0, 10), 2, 0.5,
Scalar(0, 0, 0), 2);
                putText(image, "Area : " +
to_string(_area), Point(0, 25), 2, 0.5, Scalar(0, 0,
0), 2);
                putText(image, "MaxTmp: " +
to_string(max_temp), Point(0, 390), 2, 0.5, Scalar(0,
0, 0), 2);
                putText(image, "MinTmp: " +
to_string(min_temp), Point(0, 375), 2, 0.5, Scalar(0,
0, 0), 2);
        }
}

void ImgProc()
{
        bool trackObjects = true;
        bool useMorphOps = true;
        Mat image(Size(8, 8), CV_32FC1, _dataNorm);
        Mat HSV, thres, image64, imageRAW, beforeMorph;
        vector< vector< cv::Point > > contours; vector<
cv::Vec4i > hierarcy;
        if (safety == 1)
        {
                xx = 0;
                yy = 0;
                createTrackbars();
                safety = 0;
        }
        flip(image, image, +1);
        imageRAW = image.clone();
```

```
		image.convertTo(image, CV_8UC1);
		applyColorMap(image, image, COLORMAP_JET);
		//image64 = image.clone();
		resize(image, image, Size(), 50.0, 50.0,
INTER_CUBIC);
		cvtColor(image, HSV, COLOR_BGR2HSV);
		inRange(HSV, Scalar(H_MIN, S_MIN, V_MIN),
Scalar(H_MAX, S_MAX, V_MAX), thres);
		//inRange(image, Scalar(H_MIN, S_MIN, V_MIN),
Scalar(H_MAX, S_MAX, V_MAX), thres);
		cv::findContours(thres, contours, hierarcy,
cv::RETR_LIST, cv::CHAIN_APPROX_SIMPLE);
		//printf("blob %d \n", contours.size());
		vector<int> sortIdx(contours.size());
		vector<float> areas(contours.size());
		for (int n = 0; n < (int)contours.size(); n++)
		{
			sortIdx[n] = n;
			areas[n] = contourArea(contours[n],
false);
		}
		int idx;
		if (useMorphOps)
			morphOps(thres);

		if (trackObjects)
		{
			trackFilteredObject(xx, yy, thres, image);

		}
		//namedWindow("Original_Interpolate",
WINDOW_AUTOSIZE);
		//namedWindow("Original", WINDOW_FREERATIO);
		//namedWindow("RAW_Image", WINDOW_FREERATIO);
		//imshow("RAW_Image", imageRAW);
		//imshow("Original", image64);
		imshow("Original_Interpolate", image);
		//moveWindow("Original_Interpolate", 120, 220);
		//imshow("HSV", HSV);
		imshow("Threshold", thres);
```

```
	//imshow("No Morph", beforeMorph);
	//inRange(image, (100, 100, 100), (255, 255,
255),mask);
	waitKey(0);
}

int main()
{
	HWND console = GetConsoleWindow();
	RECT ConsoleRect;
	GetWindowRect(console, &ConsoleRect);
	MoveWindow(console, 0, 0, 300, 300, TRUE);
	std::string _terima;
	SerialPort arduino(port_name);

	if (arduino.isConnected()) cout << "Connection
Established" << endl;
	else cout << "ERROR, check port name";

	std::cout << std::fixed;
	koordinatX = 0;
	_area = 0;
	while (arduino.isConnected())
	{
	skip1:
		_countParse = 0;
		_countParse2 = 0;
		memset(incomingData, 0, MAX_DATA_LENGTH);
		int read_result =
arduino.readSerialPort(incomingData, MAX_DATA_LENGTH);
		int pos1 = _terima.find("[");
		if (pos1 == std::string::npos)
		{
			_terima = _terima + incomingData;
			goto skip1;
		}
		int pos2 = _terima.find("]", pos1 + 1);
		if (pos2 == std::string::npos)
		{
			_terima = _terima + incomingData;
```

```
                        goto skip1;
                }
                std::string _terima2 =
_terima.substr((pos1+1), (pos2-pos1-1));
                std::string _delimiter = ",";
                _terima = "";
                vector<string> v = split(_terima2,
_delimiter);
                for (auto i : v)
                {
                        _data[_countParse2] = std::stof(i);
                        _dataNorm[_countParse2] =
((_data[_countParse2] - tempMin) * (255 - 0)) /
((tempMax - tempMin) + 0);
                        _countParse++;
                        _countParse2++;
                        if (_countParse == 8)
                        {
                                _countParse = 0;
                        }
                        if (_countParse2 == 64)
                        {
                                _countParse2 = 0;
                                break;
                        }
                }
                max_temp = *max_element(begin(_data),
end(_data));
                min_temp = *min_element(begin(_data),
end(_data));
                if (callibr < 10)
                {
                        tempMin =
*min_element(begin(_data), end(_data));
                        tempMax =
*max_element(begin(_data), end(_data));
                        if (tempMax < 31)
                        {
                                tempMax = 31;
                        }
```

```
                    callibr++;
                    cout << "Please Wait..
Callibration" << endl;
                    if (callibr == 10) cout <<
"Callibration Finished" << endl;
                    goto skip1;
                }
                thread t1(ImgProc);
                t1.detach();
        }
        waitKey(0);
}
```

-----Halaman ini sengaja dikosongkan-----

LAMPIRAN K

Listing Code Control Motor (C#)

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using LattePanda.Firmata;

namespace ThermalCam
{
    public static class ExtensionMethods
    {
        public static decimal Map(this decimal value, d
ecimal fromSource, decimal toSource, decimal fromTarget
, decimal toTarget)
        {
            return (value - fromSource) / (toSource - f
romSource) * (toTarget - fromTarget) + fromTarget;
        }
    }
    class Motor
    {
        public const int PWM_Front = 6;
        public const int PWM_Left = 10;
        public const int PWM_Right = 9;
        public const int IN_F1 = 18;
        public const int IN_F2 = 19;
        public const int IN_R1 = 7;
        public const int IN_R2 = 8;
        public const int IN_L1 = 11;
        public const int IN_L2 = 12;
    }
    static class Globals
    {
        public static int XTotal = 400;
```

```
        public static int Koordinat_Data = 0;
        public static double Area = 0.00;
        public const double RAD = 57.3; //Jari Jari Rod
a
        public const double ALPHA = 0; //Error Sudut
        public const double RADIUS = 0.14; //Jari Jari
Robot
        public const double max_speed = 200;
        public const int MOTOR_KANAN = 0;
        public const int MOTOR_KIRI = 1;
        public const int MOTOR_DEPAN = 2;
    }
    class Program
    {
        static Arduino arduino = new Arduino();
        static void Main(string[] args)
        {
            string InputText;
            AppDomain.CurrentDomain.ProcessExit += new
EventHandler(OnProcessExit);
            arduino.pinMode(Motor.PWM_Front, Arduino.PW
M);
            arduino.pinMode(Motor.PWM_Left, Arduino.PWM
);
            arduino.pinMode(Motor.PWM_Right, Arduino.PW
M);
            arduino.pinMode(Motor.IN_F1, Arduino.OUTPUT
);
            arduino.pinMode(Motor.IN_F2, Arduino.OUTPUT
);
            arduino.pinMode(Motor.IN_R1, Arduino.OUTPUT
);
            arduino.pinMode(Motor.IN_R2, Arduino.OUTPUT
);
            arduino.pinMode(Motor.IN_L1, Arduino.OUTPUT
);
            arduino.pinMode(Motor.IN_F1, Arduino.OUTPUT
);
            while (true)
            {
```

```
                using (var filex = new FileStream(@"C:\
Users\lattepanda\Documents\SatrioTsubasa\Visual Studio
Program\SerialC\x64\Debug\data.txt", FileMode.Open, Fil
eAccess.Read, FileShare.ReadWrite))
                using (var sourcex = new StreamReader(f
ilex, Encoding.Default))
                {
                    InputText = sourcex.ReadToEnd();
                }
                string[] InputData = InputText.Split('#
');

                if(InputData[0] != "" && InputData[1] !
= "")
                {
                    Globals.Koordinat_Data = Convert.To
Int32(InputData[0]);
                    Globals.Area = Convert.ToDouble(Inp
utData[1]);
                }
                Process();
            }
        }

        static double pwm_motor(double kec_translasi, i
nt nomor_motor, int arah, int kec_putar)
        {
            double[] sudut = {120,60,0};

            double kec_x = kec_translasi * Math.Cos(ara
h / Globals.RAD);
            double kec_y = kec_translasi * Math.Sin(ara
h / Globals.RAD);
            double temp = (kec_x * Math.Cos((sudut[nomo
r_motor] + Globals.ALPHA) / Globals.RAD)) + (kec_y * Ma
th.Sin((sudut[nomor_motor] + Globals.ALPHA) / Globals.R
AD)) + (Globals.RADIUS * kec_putar);
            return temp;
        }
```

```
        static void Process()
        {
            int Koordinat, Abs_Koordinat;
            decimal PWM;
            double Area;
            double _PWM;
            double kec_1, kec_2, kec_3;
            Koordinat = (Globals.XTotal /2) - Globals.K
oordinat_Data;
            Koordinat = (9 * Koordinat /40) + 90;

            Area = Globals.Area;
            Abs_Koordinat = Math.Abs(Koordinat);

            if (Area < 30000 && Area > 500)
            {
                PWM = Convert.ToDecimal(Area).Map(500,
30000, 190, 150);
                _PWM = Convert.ToDouble(PWM);

                kec_1 = pwm_motor(_PWM, Globals.MOTOR_K
ANAN, Koordinat, 0);
                kec_2 = -
1 * pwm_motor(_PWM, Globals.MOTOR_KIRI, Koordinat, 0);
                kec_3 = pwm_motor(_PWM, Globals.MOTOR_D
EPAN, Koordinat, 0);

                if (kec_1 > 0)
                {
                    arduino.analogWrite(Motor.PWM_Right
, Convert.ToInt32(kec_1));
                    arduino.digitalWrite(Motor.IN_R1, A
rduino.LOW);
                    arduino.digitalWrite(Motor.IN_R2, A
rduino.HIGH);
                }
                else
                {
                    arduino.analogWrite(Motor.PWM_Right
, Math.Abs(Convert.ToInt32(kec_1)));
```

```
                    arduino.digitalWrite(Motor.IN_R1, A
rduino.HIGH);
                    arduino.digitalWrite(Motor.IN_R2, A
rduino.LOW);
                }
                if (kec_2 > 0)
                {
                    arduino.analogWrite(Motor.PWM_Left,
 Convert.ToInt32(kec_2));
                    arduino.digitalWrite(Motor.IN_L2, A
rduino.LOW);
                    arduino.digitalWrite(Motor.IN_L1, A
rduino.HIGH);
                }
                else
                {
                    arduino.analogWrite(Motor.PWM_Left,
 Math.Abs(Convert.ToInt32(kec_2)));
                    arduino.digitalWrite(Motor.IN_L2, A
rduino.HIGH);
                    arduino.digitalWrite(Motor.IN_L1, A
rduino.LOW);
                }
                if (kec_3 > 0)
                {
                    arduino.analogWrite(Motor.PWM_Front
, Convert.ToInt32(kec_3));
                    arduino.digitalWrite(Motor.IN_F1, A
rduino.LOW);
                    arduino.digitalWrite(Motor.IN_F2, A
rduino.HIGH);
                }
                else
                {
                    arduino.analogWrite(Motor.PWM_Front
, Math.Abs(Convert.ToInt32(kec_3)));
                    arduino.digitalWrite(Motor.IN_F1, A
rduino.HIGH);
                    arduino.digitalWrite(Motor.IN_F2, A
rduino.LOW);
```

```
                }


            }
            else if (Area > 45000)
            {
                PWM = Convert.ToDecimal(Area).Map(45000
, 155000, 150, 190);
                _PWM = Convert.ToDouble(PWM);
                kec_1 = pwm_motor(_PWM, Globals.MOTOR_K
ANAN, Koordinat+180, 0);
                kec_2 = -
1 * pwm_motor(_PWM, Globals.MOTOR_KIRI, Koordinat+180,
0);
                kec_3 = pwm_motor(_PWM, Globals.MOTOR_D
EPAN, Koordinat+180, 0);

                if (kec_1 > 0)
                {
                    arduino.analogWrite(Motor.PWM_Right
, Convert.ToInt32(kec_1));
                    arduino.digitalWrite(Motor.IN_R1, A
rduino.LOW);
                    arduino.digitalWrite(Motor.IN_R2, A
rduino.HIGH);
                }
                else
                {
                    arduino.analogWrite(Motor.PWM_Right
, Math.Abs(Convert.ToInt32(kec_1)));
                    arduino.digitalWrite(Motor.IN_R1, A
rduino.HIGH);
                    arduino.digitalWrite(Motor.IN_R2, A
rduino.LOW);
                }
                if (kec_2 > 0)
                {
                    arduino.analogWrite(Motor.PWM_Left,
 Convert.ToInt32(kec_2));
```

```
                    arduino.digitalWrite(Motor.IN_L2, A
rduino.LOW);
                    arduino.digitalWrite(Motor.IN_L1, A
rduino.HIGH);
                }
                else
                {
                    arduino.analogWrite(Motor.PWM_Left,
 Math.Abs(Convert.ToInt32(kec_2)));
                    arduino.digitalWrite(Motor.IN_L2, A
rduino.HIGH);
                    arduino.digitalWrite(Motor.IN_L1, A
rduino.LOW);
                }
                if (kec_3 > 0)
                {
                    arduino.analogWrite(Motor.PWM_Front
, Convert.ToInt32(kec_3));
                    arduino.digitalWrite(Motor.IN_F1, A
rduino.LOW);
                    arduino.digitalWrite(Motor.IN_F2, A
rduino.HIGH);
                }
                else
                {
                    arduino.analogWrite(Motor.PWM_Front
, Math.Abs(Convert.ToInt32(kec_3)));
                    arduino.digitalWrite(Motor.IN_F1, A
rduino.HIGH);
                    arduino.digitalWrite(Motor.IN_F2, A
rduino.LOW);
                }
            }
            else
                MotorStop();

        }

        static void MotorStop()
        {
```

```
            arduino.analogWrite(Motor.PWM_Front,0);
            arduino.analogWrite(Motor.PWM_Left, 0);
            arduino.analogWrite(Motor.PWM_Right, 0);
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.L
OW);
        }

        static void MotorMaju()
        {
            arduino.analogWrite(Motor.PWM_Front, 250);
            arduino.analogWrite(Motor.PWM_Left, 250);
            arduino.analogWrite(Motor.PWM_Right, 250);

            //Maju
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.H
IGH);
        }

        static void MotorPutarKiri()
        {
```

```
            arduino.analogWrite(Motor.PWM_Front, 200);
            arduino.analogWrite(Motor.PWM_Left, 200);
            arduino.analogWrite(Motor.PWM_Right, 200);

            //Rotate CCW
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R1, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.H
IGH);
        }

        static void MotorPutarKanan()
        {
            arduino.analogWrite(Motor.PWM_Front, 200);
            arduino.analogWrite(Motor.PWM_Left, 200);
            arduino.analogWrite(Motor.PWM_Right, 200);

            //Rotate CW
            arduino.digitalWrite(Motor.IN_F2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F1, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L1, Arduino.H
IGH);
        }
```

```
        static void Motorx()
        {
            AppDomain.CurrentDomain.ProcessExit += new
EventHandler(OnProcessExit);
            arduino.analogWrite(Motor.PWM_Front, 200);
            arduino.analogWrite(Motor.PWM_Left, 200);
            arduino.analogWrite(Motor.PWM_Right, 200);

            //Maju
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.H
IGH);
            Thread.Sleep(4000);

            arduino.analogWrite(Motor.PWM_Front, 250);
            arduino.analogWrite(Motor.PWM_Left, 140);
            arduino.analogWrite(Motor.PWM_Right, 140);

            //Kanan
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_L1, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_L2, Arduino.L
OW);
```

```
            Thread.Sleep(3000);

            arduino.analogWrite(Motor.PWM_Front, 250);
            arduino.analogWrite(Motor.PWM_Left, 140);
            arduino.analogWrite(Motor.PWM_Right, 140);

            //Kiri
            arduino.digitalWrite(Motor.IN_F1, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_F2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R1, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.H
IGH);
            Thread.Sleep(3000);

            arduino.analogWrite(Motor.PWM_Front, 200);
            arduino.analogWrite(Motor.PWM_Left, 200);
            arduino.analogWrite(Motor.PWM_Right, 200);

            //Rotate CCW
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R1, Arduino.H
IGH);
            arduino.digitalWrite(Motor.IN_R2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.H
IGH);
            Thread.Sleep(2000);
        }
```

```
        static void OnProcessExit(object sender, EventA
rgs e)
        {
            arduino.analogWrite(Motor.PWM_Front, 0);
            arduino.analogWrite(Motor.PWM_Left, 0);
            arduino.analogWrite(Motor.PWM_Right, 0);
            arduino.digitalWrite(Motor.IN_F1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_F2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_R2, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L1, Arduino.L
OW);
            arduino.digitalWrite(Motor.IN_L2, Arduino.L
OW);
            Console.WriteLine("I'm out of here");
        }
    }
}
```

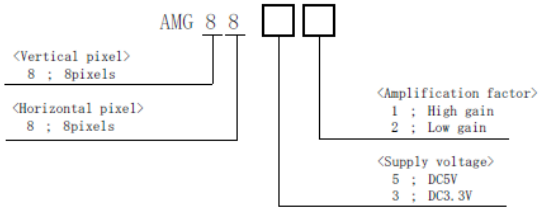
LAMPIRAN L

Datasheet AdaFruit AMG8833

1. Name : Infrared Array Sensor "Grid-EYE"

2. Part No. :

2-1 Part No. System



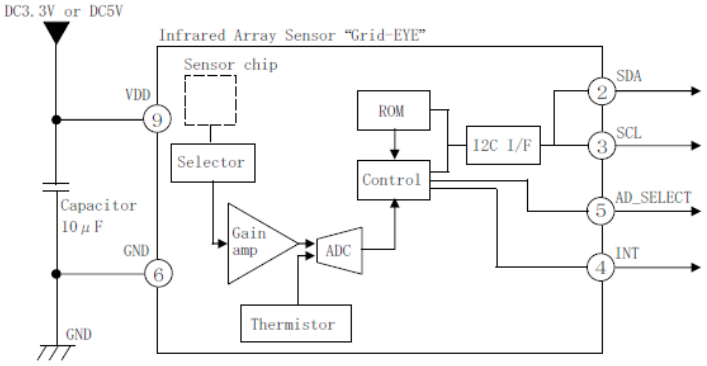
2-2 Part No. List

AMG8831 (VDD=3.3V , High gain type)
AMG8832 (VDD=3.3V , Low gain type)
AMG8851 (VDD=5V , High gain type)
AMG8852 (VDD=5V , Low gain type)

3. Drawing : AMG8851 Product drawing

4. Characteristics

4-1 Internal circuit



*④INT terminal normally has same voltage as VDD. When interrupting, same as GND(0V).
*Regarding of recommended external circuit, please refer to section 4-9.

4-2 Main Functions

Item	Value
Pixel number	64 (8×8 Matrix)
External Interface	I²C (fast mode)
Frame rate	Typ.10 frames/sec or Typ.1 frame/sec
Operating Mode	Normal Sleep Stand-by (10sec or 60sec intermittence)
Output Mode	Temperature Output
Calculate Mode	No moving average or Twice moving average
Temperature Output Resolution	0.25℃
Number of Sensor Addresses	2 (I²C Slave Address)
Thermistor Output Temperature Range	-20℃～80℃
Thermistor Output Resolution	0.0625℃

4-3 Absolute Maximum Ratings

Item	Specification	Terminal
Applied voltage	-0.3～6.5V	VDD, VPP
Input/Output voltage	-0.3～Vdd+0.3V	SCL, SDA, AD_SELECT
Output current	-10～10mA	INT, SDA
ESD (Human Body Model)	1kV	All Terminals
ESD (Machine Model)	200V	All Terminals

4-4 Ratings

Item	Specification	
	High gain	Low gain
Applied voltage	3.3V±0.3V or 5.0V±0.5V	
Temperature Range of Measuring Object	0℃～80℃	-20℃～100℃
Operating temperature	0℃～80℃	-20℃～80℃
Storage temperature	-20℃～80℃	

4-5 Characteristics

Item	Specification	
	High gain	Low gain
Temperature Accuracy	Within Typ. ±2.5℃	Within Typ. ±3.0℃
Rated detection distance ※1	5m (Max.)	
Field of View	Typ.60° (Horizontal, Vertical)	
Optical Axis Gap	Within Typ. ±5.6° (Horizontal, Vertical)	
Current Consumption	Typ. 4.5mA (normal mode) Typ. 0.2mA (sleep mode) Typ. 0.8mA (stand-by mode)	
Setup Time	Typ. 50msec (Time to enable Communication after Setup) Typ. 15sec (Time to stabilize Output after Setup)	

※1 ・To have more than 4℃ of temperature difference from background
・Detection object size : 700×250mm (Assumable human body size)

4-6 Electric characteristics

(1) Characteristics of the SDA and SCL I/O stages

parameter	symbol	Min.	Max.	unit
Low level input voltage	V_{IL}	-0.3	0.3×VDD	V
High level input voltage	V_{IH}	0.7×VDD	VDD+0.3	V
Hysteresis (SDA、SCL)	V_{hys}	0.05×VDD		V
Low level output voltage (at 3mA sink current)	V_{OL}	0	0.4	V
Output fall time from V_{IHmin} to V_{ILmax} with a bus capacitance from 10pF to 400pF	t_{of}	$20+C_b$	250	ns
Pulse width of spikes which must be suppressed by the input filter	t_{SP}	0	50	ns
Input current each I/O pin with an input voltage between 0.1×VDD～0.9×VDD	I_I	-10	10	μA
Capacitance for each I/O pin	C_i	-	10	pF

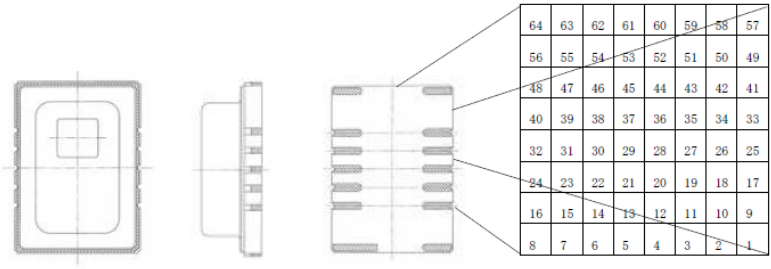
(2) Characteristics of the SDA and SCL bus lines

parameter	symbol	Min.	Max.	unit
SCL clock frequency	f_{SCL}	0	400	kHz
Hold time (repeated) START condition.	$t_{HD:STA}$	600	-	ns
Low period of the SCL clock	t_{LOW}	1.3		μs
High period of the SCL clock	T_{HIGH}	0.6		μs
Set-up time for a repeated START condition	$t_{SU:STA}$	0.6		μs
Data hold time	$t_{HD:DAT}$	0	900	ns
Data set-up time	$t_{SU:DAT}$	100		ns
Rise time of both SDA and SCL signals (f_{SCL}>100kHz)	t_r	20+0.1 ×C_b	300	ns
Rise time of both SDA and SCL signals (f_{SCL}≦100kHz)	t_r		1000	ns
Fall time of both SDA and SCL signals	t_f	20+0.1 ×C_b	300	ns
Set-up time for STOP condition	$t_{SU:STO}$	600		ns
Bus free time between a STOP and START condition	t_{BUF}	1300		ns
Capacitive load for each bus line	C_b		400	pF

4-7 Pixel Array & Viewing Field

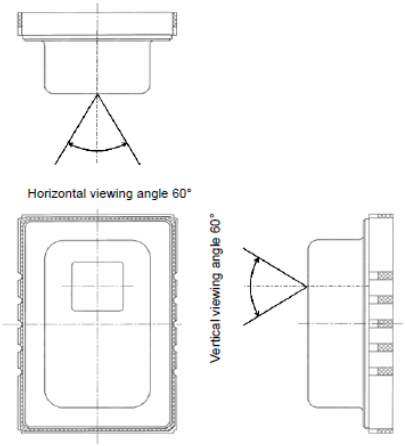
(1)Pixel Array

Pixel Array from 1 to 64 is shown below.



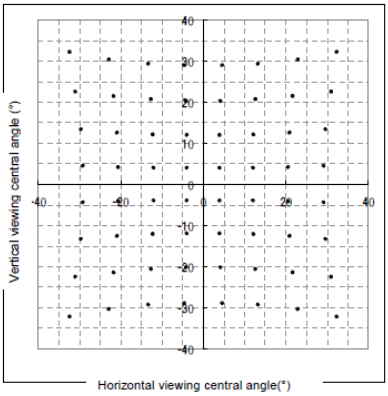
(2)Viewing Field

Sensor Viewing Field (Typical) is shown below.

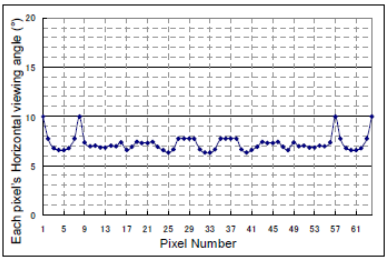


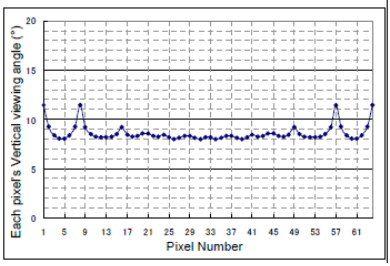
(3)Typical characteristics : Each pixel's viewing central angle
*Regarding of Pixel Array, please refer to 4-7(1).

Sensor's optical center (the origin of graph below) gap
: within Typ. ±5.6° (Both of horizontal and vertical directions)



(4)Typical characteristics : Each pixel's viewing angle (half angle)
Central 4 pixels (Pixel No. 28, 29, 36, 37) viewing angle (half angle) :
horizontal direction Typ. 7.7°
vertical direction Typ. 8°





BIODATA PENULIS

Penulis dilahirkan di Surakarta pada tanggal 31 Juli 1995 sebagai anak pertama dari lima bersaudara. Penulis memulai kegiatan pendidikan formal di SD Islam Hidayatullah Semarang, yang kemudian dilanjutkan di SMP Negeri 21 Semarang. Pada saat penulis melanjutkan pendidikan ke sekolah menengah atas, penulis memilih melanjutkan ke salah satu sekolah vokasi di Indonesia yang membuka program SMK 4 tahun yaitu SMK Negeri 1 Cimahi jurusan Teknik Elektronika Industri dan Komputer. Berawal dari SMK inilah kecintaan penulis terhadap dunia pemrograman komputer dan elektronika mulai tumbuh. Setelah lulus pendidikan SMK pada tahun 2014, penulis melanjutkan pendidikan diploma 3 di Politeknik Manufaktur Negeri Bandung jurusan Teknik Otomasi Manufaktur dan Mekatronika. Pada tahun 2017 penulis melanjutkan pendidikan ke tahap sarjana di Institut Teknologi Sepuluh Nopember Surabaya program studi Teknik Elektronika. Selama menjalani perkuliahan, penulis pernah menjadi mahasiswa berprestasi tingkat nasional, aktif dalam himpunan mahasiswa, serta pernah mengikuti dan menjuarai beberapa perlombaan di bidang robotika dan otomasi tingkat provinsi dan nasional. Penulis juga aktif mengerjakan beberapa permintaan projek dari perusahaan swasta di Indonesia dalam bidang elektronika dan industri 4.0.

Email	: satriomaulana.tsubasa@gmail.com
No. Hp / WA	: 08221-45678-69
Line	: @SatrioTsubasa
IG	: @SatrioTsubasa
Twitter	: @SatrioTsubasa
LinkedIn	: Satrio Maulana Tsubasa