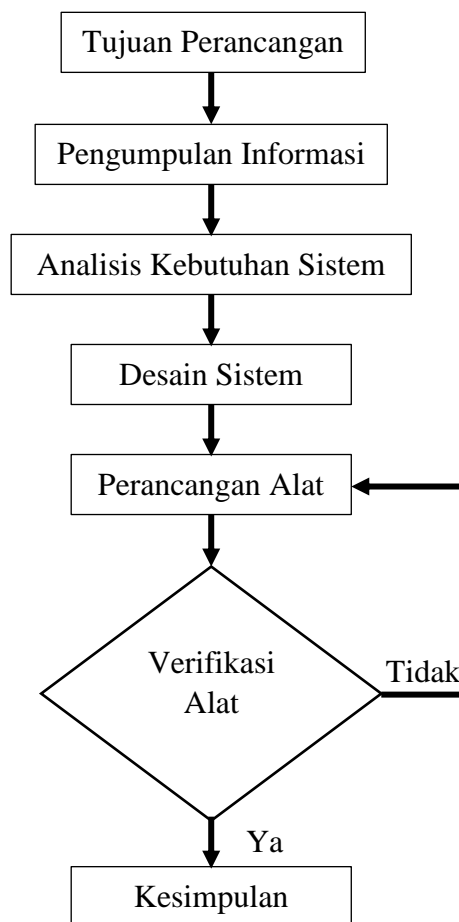


BAB III
METODOLOGI PERANCANGAN

3.1 Prosedur Perancangan

Prosedur perancangan merupakan langkah – langkah dalam pembuatan tugas akhir ini. Dan prosedur perancangan ini digambarkan pada diagram alir berikut:



Gambar 3.1 Diagram Blok Prosedur Perancangan

3.1.1 Penjelasan Blok Diagram

1) Tujuan Perancangan

Penelitian ini diawali dengan pembuatan tujuan perancangan dasar mengenai fungsi kerja dari rancang bangun pendeteksian citra warna pada sistem kendali *autonomous* pada kapal cepat tanpa awak (USV).

2) Pengumpulan Informasi

Pada tahap ini akan dikumpulkan data-data dan informasi dari buku jurnal maupun informasi dari internet.

3) Analisis Kebutuhan Sistem

Sistem ini memiliki beberapa kebutuhan yang harus dicapai agar dapat sempurna dan sesuai dengan tujuan yang akan dicapai. Kebutuhan-kebutuhan pokok yang harus terpenuhi untuk merancang sistem adalah sebagai berikut :

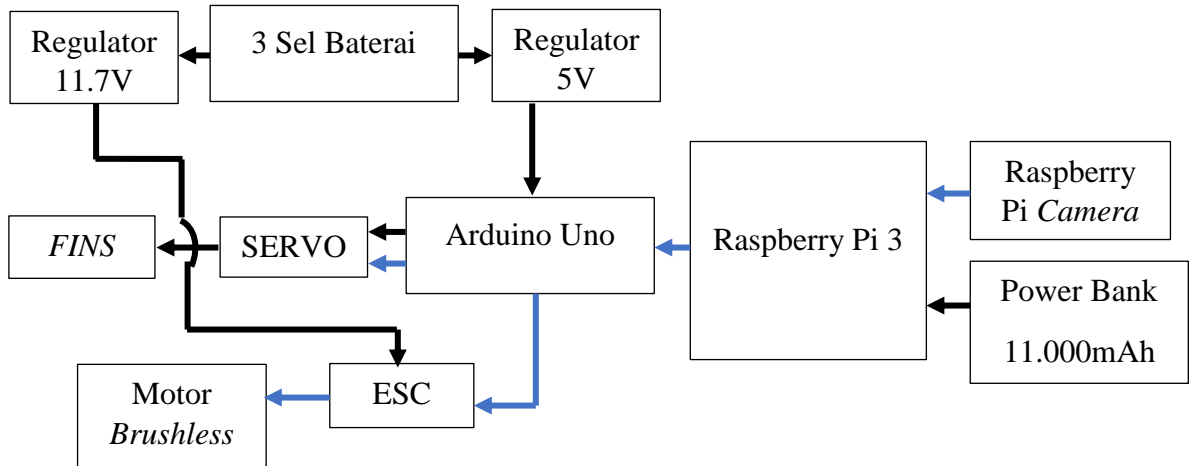
- Komputer yang digunakan
- Mikrokontroler yang digunakan
- Modul kamera yang berfungsi sebagai sensor sebagai pendeteksi citra warna
- Komponen-komponen pendukung seperti, regulator, kabel, pin *header*, pin deret, *socket*, dll.
- Servo yang berfungsi untuk mengendalikan arah *fins*
- Motor DC *Brushless* yang berfungsi untuk pendorong laju kapal dengan propulsi udara

4) Desain Sistem

Desain sistem kendali *autonomous* kapal cepat tanpa awak (USV) ini terdiri dari dua bagian, yaitu desain sistem perangkat keras (*hardware*) dan desain perangkat lunak (*software*). Perangkat keras meliputi perakitan elektronik sedangkan untuk pembuatan perangkat lunak meliputi pembuatan program pendeteksian citra warna (*Image Color Tracking*) menggunakan Raspberry Pi *Camera V2* serta pembuatan program untuk penggerak servo dan motor DC *brushless* pada mikrokontroler.



A. Desain Sistem Perangkat Keras

Rancang keseluruhan sistem ditunjukkan dalam diagram blok seperti berikut:



Gambar 3.2 Diagram Blok Keseluruhan Sistem

Keterangan :

-  = Data
-  = Tegangan

Prinsip Kerja :

1. *Power Bank* 11.000mAh sebagai catu daya *Raspberry Pi 3*
2. Baterai LiPo 3 sel sebagai catu daya *Arduino Uno*, *ESC* dan *Servo*
3. *Raspberry Pi Camera V2* memberi masukan citra yang diterima
4. *Raspberry Pi 3* mengolah citra yang diterima oleh kamera
5. *Arduino Uno* menerima data yang telah diolah dalam bentuk komunikasi serial dari *Raspberry Pi 3*
6. *Arduino Uno* mengirimkan sinyal (PWM) ke *ESC* dan *servo* untuk menggerakkan motor DC *brushless* dan *fins*

Komponen-komponen yang dibutuhkan untuk membuat sistem adalah :

1. Alat

- a. Laptop
- b. Solder dan Tenol
- c. Tang potong
- d. Gunting
- e. *Multimeter* analog atau digital
- f. Monitor
- g. *Keyboard* dan *Mouse*
- h. Lem *Silicon*

2. Bahan

- a. Papan PCB lubang
- b. *Container box*
- c. Kabel *Jumper*
- d. Mur dan Baut
- e. Pin *Header Male* dan *Female*
- f. *Acrylic*
- g. DC *Jack*
- h. Selongsong bakar
- i. Jack XT60
- j. Dan komponen elektronika lainnya

B. Desain Sistem Perangkat Lunak

Perangkat lunak dibuat untuk memproses dan mengontrol proses kerja dari keseluruhan sistem. Desain perangkat lunak kali ini menggunakan OpenCV dengan bahasa pemrograman Python, dan Arduino IDE dengan Bahasa pemrograman C.

5) Perancangan

Tahap berikutnya adalah perancangan yaitu rancangan pembuatan sistem rangkaian elektronik, perancangan penyambungan kabel *jumper* antar komponen elektronik dengan mikrokontroler, dan penyambungan kabel *jumper* mikrokontroler ke Raspberry Pi 3.

6) Verifikasi Alat

Setelah alat dibuat, maka dilakukan verifikasi untuk mengetahui apakah alat sudah bekerja dengan baik. Jika masih terdapat kesalahan maka dilakukan pengecekan dan perbaikan kembali sehingga dapat bekerja dengan kondisi normal sesuai dengan tujuan pembuatan alat. Jika alat telah bekerja dengan baik maka dilanjutkan ke tahap berikutnya.

7) Kesimpulan

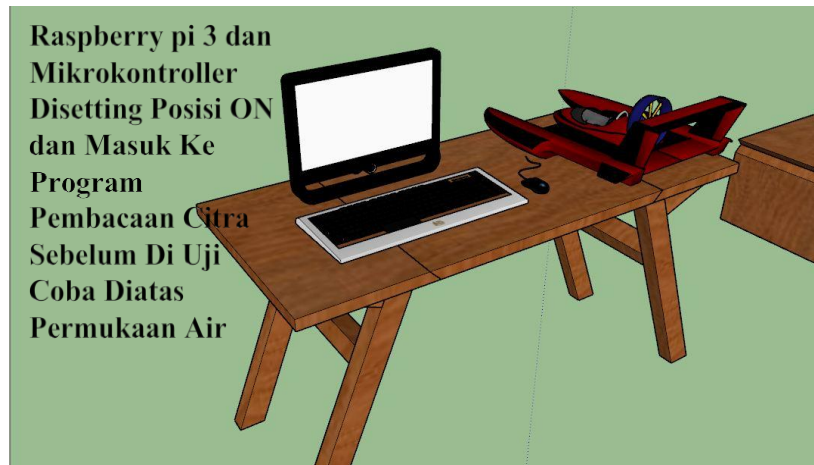
Berisikan hasil akhir dari uji coba dan penelitian yang telah dilakukan.

3.2 Skenario Alat

Untuk mengetahui secara jelas bagaimana cara kerja alat yang dibuat maka dibuatlah skenario alat seperti penjelasan dan gambaran dibawah.

1. Alat yang terdiri dari kapal *boat* dan *container box* berisikan Raspberry Pi 3 dan mikrokontroller di *ON* kan atau dinyalakan dalam posisi *idle* terlebih dahulu. Percobaan ini dilakukan di atas permukaan air yang tenang.
2. Alat kemudian di uji coba berjalan lurus melewati objek bola berwarna merah dan hijau, dimana kamera akan mendeteksi keberadaan bola merah dan hijau.
3. Ketika kamera berhasil mendeteksi keberadaan bola yang ada pada permukaan air, maka data yang diterima oleh kamera akan diolah oleh Raspberry Pi 3 untuk memindai warna merah dan hijau yang selanjutnya akan dieksekusi.
4. Setelah pemindaian oleh Raspberry Pi 3, data yang telah didapat kemudian dikomunikasikan ke mikrokontroler, ketika mikrokontroler menerima data dari Raspberry Pi 3 secara otomatis mikrokontroler merubah arah gerak servo dan

mengatur kecepatan rpm motor untuk dapat melaju di antara bola merah dan hijau.



Gambar 3.3 Skenario Pertama Pengaturan Awal Sebelum Kapal Melakukan Uji Coba Diatas Permukaan Air

Sebelum melakukan uji coba kapal diatas permukaan air, awal mula menghidupkan komputer dan mikrokontroler lalu melakukan pengaturan terhadap komputer Raspberry Pi dan juga Arduino.



Gambar 3.4 Skenario Kedua Pada Posisi Awal *Track* Lurus



Gambar 3.5 Skenario Ketiga Kamera Berhasil Membaca Warna dan Melaju Lurus Diantara Kedua Bola Tersebut

Kapal diletakan diatas air sebelum diantara bola merah dan hijau, yang mana kapal tersebut akan melaju lurus melawati objek tersebut. Ketika kapal berhasil mendeteksi adanya objek bola merah dan hijau, maka secara otomatis *propeller* akan berputar untuk menjalankan kapal, dan *fins* akan merespon untuk keadaan berbelok atau lurus.



Gambar 3.6 Skenario Keempat Kondisi Awal Kapal Melakukan Uji Kendali Pada Track S



Gambar 3.7 Skenario Kelima Kamera Berhasil Membaca Objek Bola Merah dan Hijau dengan Laju Kendali yang Baik



Gambar 3.8 Skenario Keenam Kapal Telah Berhasil Melakukan Uji Kendali pada Track S

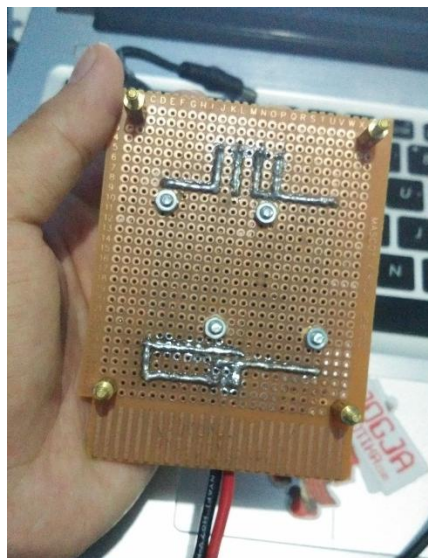
Pada pengujian track S hampir sama seperti pengujian di track lurus, akan tetapi pada track ini diatur agar berkelok, supaya dapat mengetahui respon antara pendeteksian objek dengan servo apakah dapat bekerja dengan baik atau tidak.

3.3 Perancangan Perangkat Keras

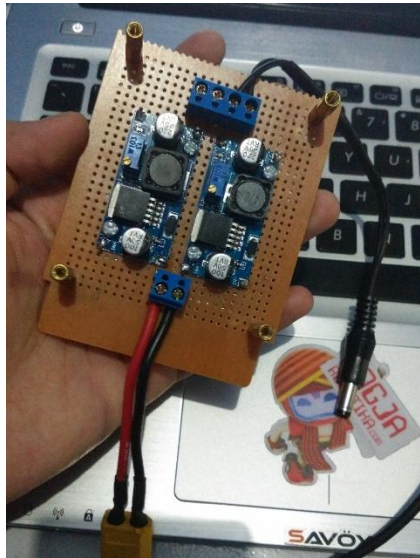
Perancangan perangkat keras pada alat ini menggunakan beberapa komponen pokok untuk mengatur kinerja sistem *autonomous* pada kapal. Sensor yang digunakan adalah jenis kamera. Kamera ini berfungsi untuk mendeteksi citra

ketika citra telah di tangkap oleh kamera kemudian Raspberry Pi mengolah citra tersebut dan memfilter warna RGB menjadi HSV (*Hue Saturation Value*), dan diteruskan oleh proses *thresholding*, *erode*, *dilation*. setelah objek yang terdeteksi itu didapat maka data tersebut diolah apakah kordinat pixel bola yang terdeteksi berada pada kordinat pixel yang telah ditentukan, yang kemudian data pixel pada ruang pembacaan dijadikan algoritma umpan balik berupa *char*, yang mana data *char* ini dijadikan *input* ke mikrokontroler. Ketika mikrokontroler menerima *input* komunikasi berupa data *char* dari Raspberry Pi 3 maka mikrokontroler akan mengirimkan sinyal PWM ke motor DC *brushless* dan servo untuk mengatur kecepatan dan arah gerak servo. Sebelum dilakukan pembuatan sistem rangkaian alat maka dibuat rangkaian catu daya terlebih dahulu

proses ini untuk menghubungkan *socket T-block* pada 2 regulator DC-DC *step-down* dengan cara disolder dan dihubungkan secara parallel pada pin masukan (*input*). Dan pin keluaran (*output*) pada kedua regulator dihubungkan dengan cara di solder pada masing-masing pin *T-block*



Gambar 3.9 Tampilan jalur catu daya pada PCB

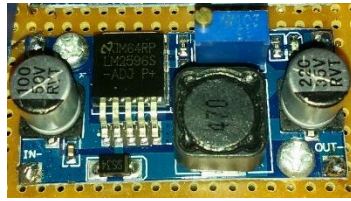


Gambar 3.10 Tampilan rangkaian regulator DC-DC *Step-Down*

Mikrokontroler yang digunakan adalah Arduino Uno dengan ATmega328. Arduino Uno dalam sistem ini bertugas sebagai penerima data dari Raspberry Pi 3 melalui pin *serial Tx* (Pin 08 UART) dan pada Arduino Uno pada pin *serial Rx* (Pin D0) dan pin *ground* pada Raspberry Pi 3 dihubungkan ke pin *ground* pada Arduino Uno, untuk jalur data ke servo menggunakan sinyal PWM melalui jalur pin D8 dan jalur data pada motor DC *brushless* yaitu sama menggunakan sinyal PWM pada pin D9.

3.3.1 Perancangan Catu Daya

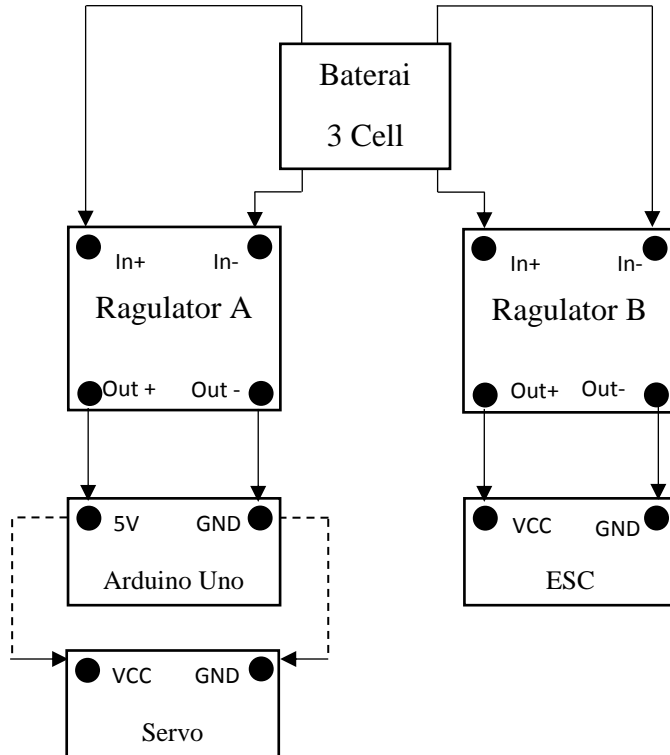
Pada alat ini digunakan 2 buah regulator DC-DC *step-down* yang berfungsi untuk menstabilkan tegangan. Digunakannya 2 regulator ini adalah karena pada sistem ini memiliki tegangan yang berbeda. Regulator pertama memiliki tegangan *output* sebesar 5V yang memberi daya ke Arduino Uno dan servo, sedangkan regulator kedua memiliki tegangan 11V yang terhubung ke tegangan inputan pada motor DC *brushless* pada ESC. Dengan catu daya yang digunakan berupa baterai Li-Po 3 sel yang memiliki tegangan 11.1V, Regulator yang digunakan pada alat ini berupa regulator DC-DC *step-down*, dan untuk catu daya pada Raspberry Pi 3 menggunakan *power bank* berkapasitas 11.000mAh dengan *output* tegangan 2.1V seperti pada gambar dibawah.



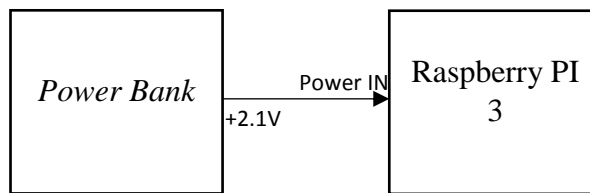
Gambar 3.11 Regulator L2596 *Step-Down* DC-DC



Gambar 3.12 Power Bank 11.000mAh *dual output*



Gambar 3.13 Skema Perancangan Regulator



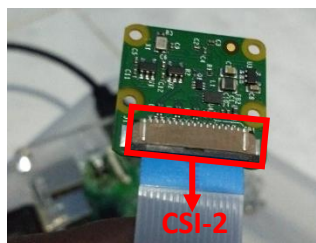
Gambar 3.14 Skema Catu Daya Raspberry Pi 3

3.3.2 Perancangan Sensor

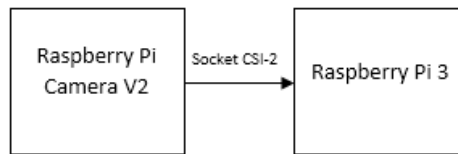
Sistem ini menggunakan kamera sebagai sensor utamanya, kamera yang digunakan yaitu Raspberry Pi *Camera V2* yang mana fungsi dari kamera ini tidak lain untuk pengambilan citra yang terhubung oleh Raspberry Pi 3. Pada Raspberry Pi 3 sudah menyediakan *socket* untuk kamera ini, jadi kamera hanya dihubungkan dengan kabel *ribbon* yang sudah didapatkan pada paket pembelian Raspberry Pi *Camera V2* ke *socket CSI-2 (Camera Serial Interface)* yang ada pada Raspberry Pi 3 maupun pada Raspberry Pi *Camera V2*.



Gambar 3.15 Socket CSI-2 Pada Raspberry Pi 3



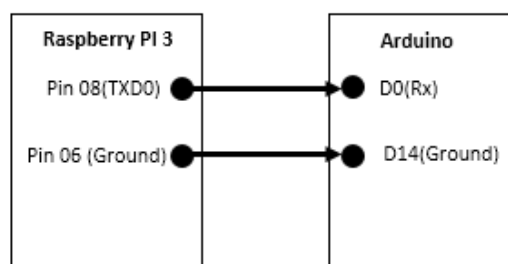
Gambar 3.16 Socket CSI-2 Pada Raspberry Pi *Camera V2*



Gambar 3.17 Skema Rangkaian Raspberry Pi *Camera* ke Raspberry Pi

3.3.3 Perancangan Komunikasi Serial UART

Fungsi komunikasi serial UART (*Universal Asynchronous Receiver-Transmitter*) di sistem ini berfungsi untuk menghubungkan antara komputer Raspberry Pi 3 dengan mikrokontroler, dimana Raspberry Pi 3 mengirimkan data berupa *char* ke mikrokontroler, hasil dari pembacaan dikirimkan menggunakan komunikasi serial UART dengan tujuan agar mikrokontroler dapat mengatur gerak servo dan motor DC *brushless* sesuai apa yang diperintahkan pada program di Raspberry Pi 3, untuk komunikasi serial UART pada raspberry pi 3 yaitu menggunakan pin 08 TXD0 (GPIO14) dan pin 06 (*Ground*), untuk mikrokontroler Arduino menggunakan pin D0 (Rx) dan pin 14 (*Ground*). Pin 08 TXD0 (GPIO14) pada Raspberry Pi 3 dihubungkan ke pin D0(Rx) pada Arduino dan pin 06 (*Ground*) pada Raspberry Pi 3 dihubungkan ke pin 14 (*Ground*) pada Arduino.

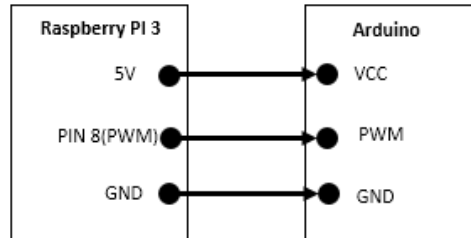


Gambar 3.18 Skema Rangkaian Komunikasi Serial UART

3.3.4 Perancangan Servo

Untuk perancangan pada servo menggunakan Arduino Uno sebagai mikrokontrolernya untuk mengatur arah gerak servo tersebut. Arduino Uno mendapat inputan data dari Raspberry Pi 3 untuk mengirimkan sinyal PWM ke servo sesuai algoritma pembacaan citra pada Raspberry Pi *Camera*. Pin D8 pada Arduino Uno iyalah PWM (*Pulse Widht Modulation*) yang dihubungkan pada kabel

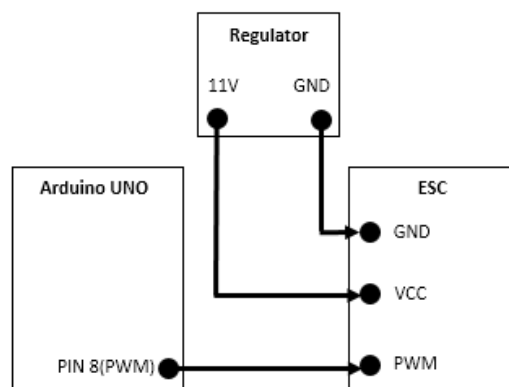
data PWM pada servo, pin 5V pada arduino Uno dihubungkan ke VCC pada servo dan pin GND pada Arduino Uno dihubungkan pada pin GND pada servo. Dibawah adalah skema rangkaian servo dengan Arduino Uno.



Gambar 3.19 Skema Rangkaian Mikrokontroler dan Servo

3.3.5 Perancangan Motor DC *Brushless*

Untuk perancangan motor DC *brushless* sama seperti perancangan servo yaitu menggunakan Arduino Uno sebagai mikrokontrolernya untuk mengatur kecepatan rpm pada motor DC *brushless*. Arduino Uno mendapat inputan data dari Raspberry Pi 3 untuk mengatur kecepatan putaran motor DC *brushless* sesuai algoritma pembacaan citra pada Raspberry Pi *Camera*, Pada Arduino Uno pin D7 (PWM) dihubungkan ke pin data pada ESC, untuk catu daya pada motor DC *brushless* yaitu kabel *power* untuk motor DC *brushless* pada ESC dihubungkan langsung ke regulator 11V karena membutuhkan daya yang besar. dibawah adalah skematik rangkaian motor DC *brushless* dengan Arduino Uno.



Gambar 3.20 Skema Rangkaian Arduino Uno dan Motor DC

3.3.6 Perancangan Akhir

Perancangan akhir merupakan satu kesatuan perancangan-perancangan perangkat keras diatas, yang bertujuan agar alat yang dibuat dapat bekerja dengan baik dan maksimal, tidak hanya dari segi kualitas akan tetapi dari segi estetika ataupun kerapihan. Setiap bagian dari perancangan alat yang telah selesai perlu dilakukan verifikasi dan pengujian kembali.

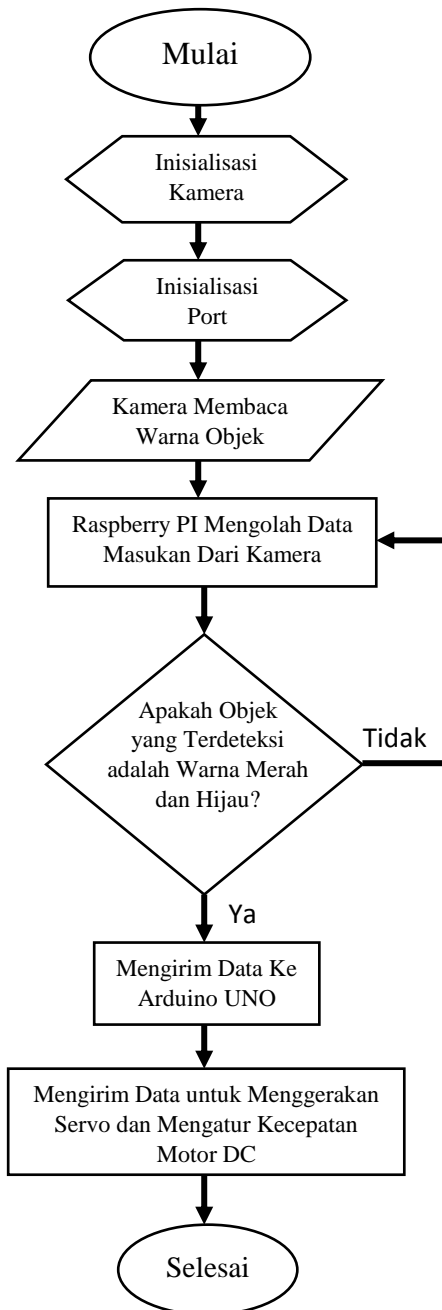
Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah alat tersebut bekerja sesuai dengan yang diharapkan. Sebagai contoh apakah rangkaian regulator tersebut sudah terhubung dengan benar dan tegangan keluaran sudah sesuai dengan apa yang ada di *data sheet* komponen itu, apakah desain kapal mengganggu kinerja kamera dalam melakukan pengambilan gambar, apakah jalur kabel pada setiap komponen elektronika sudah terhubung dengan benar dan tidak mengalami *short circuit*. Berikut adalah gambar akhir perancangan kapal cepat tanpa awak dengan kendali *autonomous* mulai dari komputer, mikrokontroler dan badan kapal.



Gambar 3.21 Perancangan Akhir Kapal Cepat Tanpa Awak

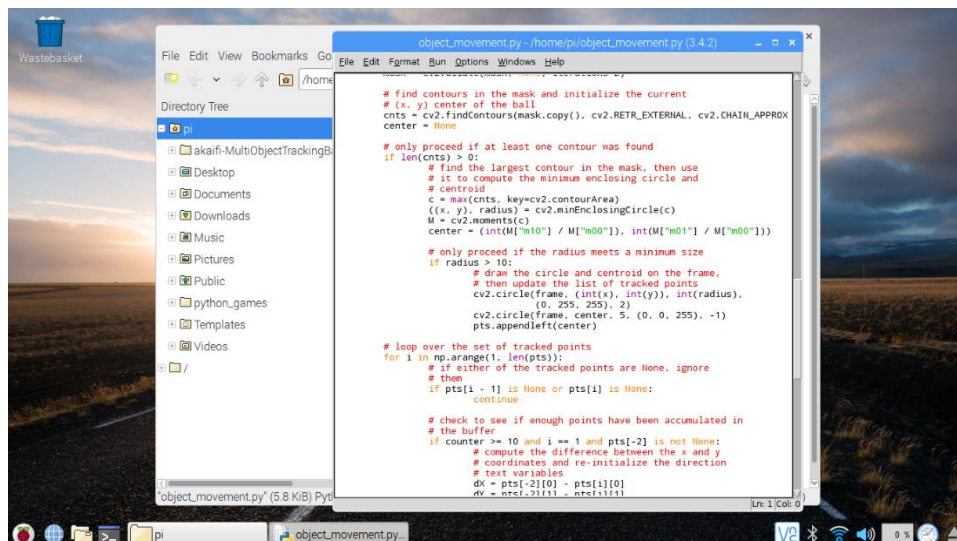
3.4 Perancangan Perangkat Lunak

Untuk memudahkan dalam pembuatan alur program maka dibuatlah *flowchart* untuk memudahkan pemahaman pada program yang dibuat seperti gambar dibawah.



Gambar 3.22 *Flowchart* Perancangan Perangkat Lunak

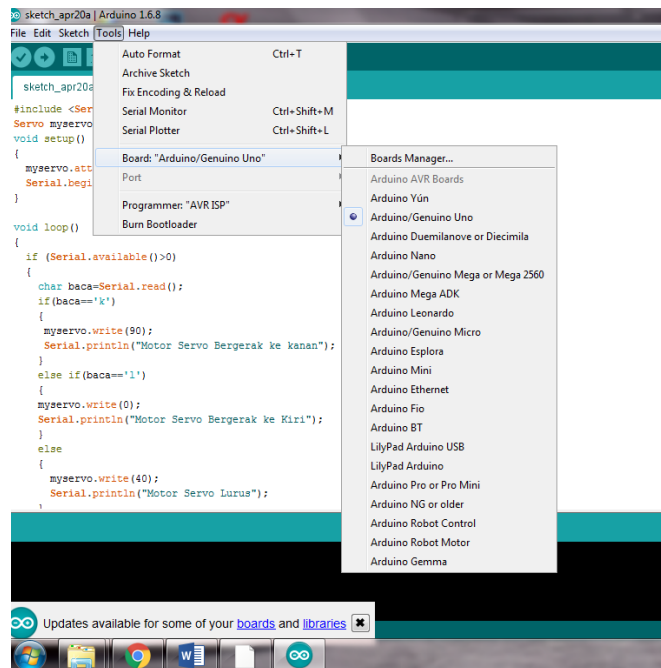
Perancangan ini menggunakan 1 masukan (*input*) dan 2 keluaran (*output*) yang diproses oleh komputer Raspberry Pi 3. *Input* tersebut adalah hasil pemindaian citra oleh Raspberry Pi Camera. Sedangkan *output* dari mikrokontroler berupa sinyal data PWM untuk menggerakkan servo dan mengatur kecepatan motor DC *brushless* pada ESC. Perancangan perangkat lunak disini menggunakan pustaka OpenCV dengan Bahasa pemrograman Python pada Raspberry Pi 3 yaitu dimana program ini bertugas untuk mengolah citra yang ditangkap oleh Raspberry Pi Camera menjadi pendeteksian warna pada objek sehingga hanya warna tertentu saja yang dipindai. Sebelum melakukan pemrograman, Raspberry Pi 3 harus dipasangkan pustaka OpenCV dan Bahasa pemrograman Python lalu meng-*install* beberapa pustaka yang dibutuhkan untuk pemrograman ini seperti, pustaka untuk menjalankan kamera Raspberry Pi camera, pustaka untuk fungsi-fungsi pemrograman pada Python dan pustaka untuk mengirimkan data serial UART pada mikrokontroler.



Gambar 3.23 Pemrograman pada Raspberry Pi 3 menggunakan Python 3.2

Setelah dilakukan pemrograman pada Raspberry Pi 3 maka pemrograman berlanjut pada mikrokontroler, yang mana fungsi dari pada pemrograman mikrokontroler disini yaitu untuk menggerakkan fungsi dari servo dan mengatur kecepatan putaran motor DC *brushless*, pemrograman mikrokontroler disini

menggunakan *software* Arduino IDE dengan bahasa pemrograman C. untuk melakukan pemrograman dilakukan terlebih dahulu memilih jenis Arduino dan *port* yang akan digunakan pada *menu tools*.



Gambar 3.24 *Menu Tools* untuk Memilih *Board* atau *Port* pada Arduino

Setelah dilakukan pemilihan *board* Arduino dan *port* yang akan digunakan seperti pada gambar diatas, maka dapat langsung dilakukan pemrograman untuk *input* dan *output* pada perancangan ini. Secara garis besar cara kerja alat ini dimulai pada pembacaan objek oleh Raspberry Pi *Camera*, data yang diterima oleh Raspberry Pi 3 yang berbentuk citra kemudian diproses untuk pendeteksian warna, ketika warna yang sudah *setting* tersebut (sebagai contoh warna hijau), maka Raspberry Pi 3 hanya akan mendeteksi warna hijau saja.

kemudian ditentukan kordinat pixel pada ruang pembacaan kamera, kamera disini menggunakan resolusi 320 x 240dpi. Jika ada bola yang terdeteksi menuju pixel yang menjadi batasannya, maka Raspberry Pi 3 mengirimkan data *char* ke Arduino dengan komunikasi serial untuk menggerakkan servo agar menjauh dari

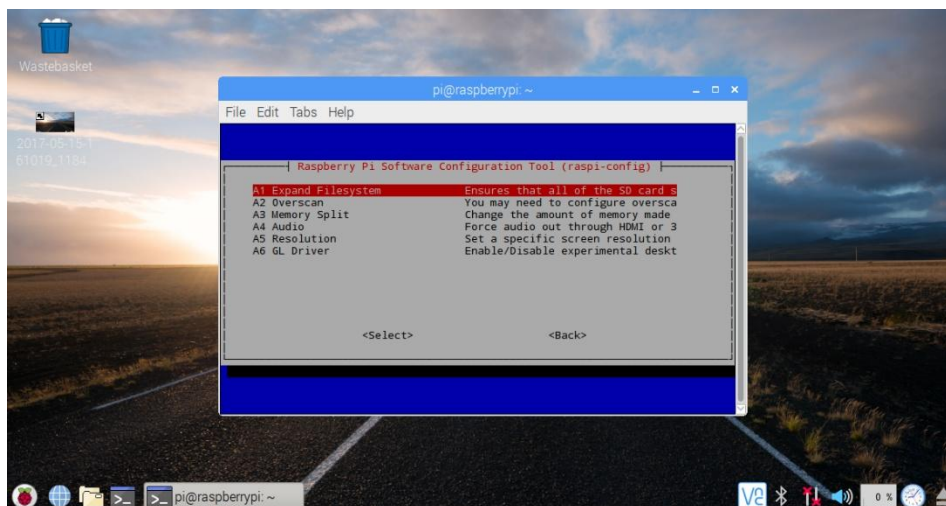
bola yang terdeteksi, untuk bergerak ke arah kanan atau kiri sehingga berada pada jalurnya kembali.

3.4.1 Penginstalan pustaka OpenCV pada Raspberry Pi 3

Untuk dapat memfungsikan program pengolahan citra, komputer Raspberry Pi 3 sebelumnya harus melakukan instalasi pustaka OpenCV dan bahasa pemrograman Python, karena pada sistem ini menggunakan OpenCV sebagai pustaka untuk pemrograman pengolahan citra dan bahasa yang digunakan adalah bahasa Python. Untuk langkah-langkah instalasi pustaka OpenCV dan Python akan dijelaskan sebagai berikut :

1. Langkah pertama, Perluasan *Filesystem*

Sebelum melakukan instalasi yaitu melakukan perluasan *filesystem* untuk memasukan ruang yang tersedia pada SD *card*, Setelah Melakukan Perluasan pada *filesystem* berikutnya Raspberry Pi 3 dinyalakan ulang (*reboot*).

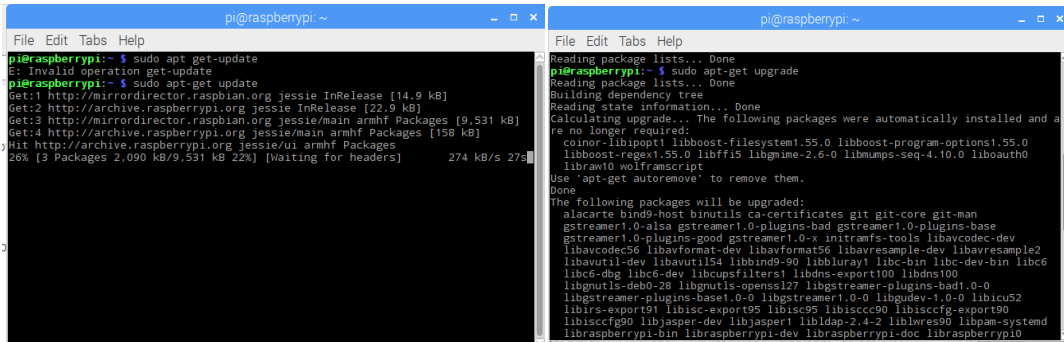


Gambar 3.25 Memperluas *Filesystem* pada Raspberry Pi 3

2. Langkah Ke-Dua, Menginstal *dependencies*

Pada awal penginstalan di langkah kedua, yaitu dilakukan *update* dan *upgrade* pada *package* Raspberry Pi 3, penginstalan beberapa alat dari pengembang, termasuk Cmake, yang dapat membantu untuk mengkonfigurasi

proses penginstalan OpenCV. Pada langkah ini juga dibutuhkan penginstalan beberapa paket *image I/O*, *video I/O*, *library* GTK untuk fungsi *highgui*, dan penginstalan dependensi tambahan untuk dapat mengoptimalkan lebih jauh pada operasi *matriks*, serta penginstalan kedua *file header* Python 2.7 dan Python 3



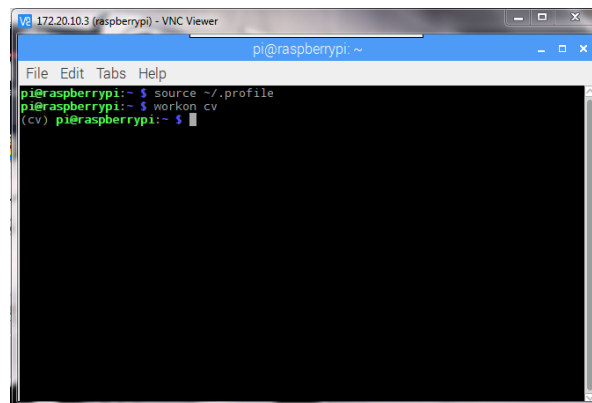
Gambar 3.26 Update dan Upgrade Package pada Raspberry Pi 3

3. Langkah Ke-Tiga, Pengunduhan OpenCV Source Code

Setelah selesai memasang dependensi, langkah berikutnya mengunduh arsip OpenCV dari penyimpanan resmi OpenCV, Untuk instalasi OpenCV 3 yang lengkap, maka perlukan juga untuk mengunduh *file* repository pada *opencv_contrib*.

4. Langkah Ke-Empat, Pemasangan Python 2.3 dan Python 3

Sebelum dapat memulai *compiling* OpenCV pada Raspberry PI 3, pertama dibutuhkan penginstalan ‘pip’, pada langkah ke-empat ini disarankan agar untuk menginstal ‘*virtualenv*’ dan ‘*virtualenvwrapper*’ akan tetapi tidak diharuskan untuk menginstall OpenCV menggunakan ‘*virtualenv*’ dan ‘*virtualenvwrapper*’. Jika penginstalan ini berhasil maka akan muncul tampilan (cv) pada *terminal* seperti gambar dibawah ini :

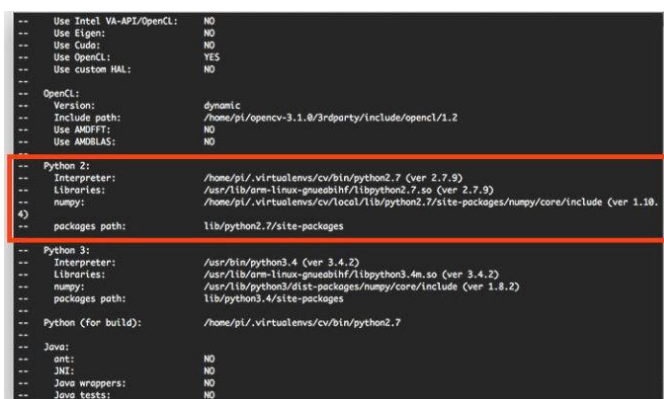


Gambar 3.27 Tampilan OpenCV pada terminal di Raspberry PI 3

Dengan asumsi bahwa penginstalan OpenCV ini telah berhasil, dan seharusnya berada di *virtual cv environment*. Satu ketergantungan pada Python ini adalah NumPy, yaitu paket Python yang digunakan untuk pemrosesan numerik, maka dari itu penginstalan selanjutnya adalah penginstalan NumPy supaya pada pemrosesan numerik dapat berjalan.

5. Langkah Ke-Lima, Penginstalan OpenCV

Pada langkah ini siap untuk meng-*compile* dan menginstal OpenCV, periksa kembali apakah sudah berada di *virtual cv environment* dengan memeriksa *prompt* (tertera teks (cv) pada terminal), jika tidak ada, cukup jalankan kode '\$ workon cv' pada terminal. Jika berhasil meng-*compile* OpenCV maka akan muncul tampilan seperti ini :



Gambar 3.28 Memastikan Bahwa Python 2.3 akan Digunakan saat Mengkompilasi OpenCV 3 untuk Raspbian Jessie

```

-- Xine: NO
-- gPhoto2: NO
--
-- Parallel framework: pthreads
--
-- Other third-party libraries:
-- Use IPP: NO
-- Use VA: NO
-- Use Intel VA-API/OpenCL: NO
-- Use Eigen: NO
-- Use Cuda: NO
-- Use OpenCL: YES
-- Use custom HAL: NO
--
-- OpenCL:
-- Version: dynamic
-- Include path: /home/pi/opencv-3.1.0/3rdparty/include/opencv/1.2
-- Use AMDFFT: NO
-- Use AMDBLAS: NO
--
-- Python 2:
-- Interpreter: /usr/bin/python2.7 (ver 2.7.9)
-- Libraries: /usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.9)
-- numpy: /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.8.2)
-- packages path: lib/python2.7/dist-packages
--
-- Python 3:
-- Interpreter: /home/pi/.virtualenvs/cv/bin/python3.4 (ver 3.4.2)
-- Libraries: /usr/lib/arm-linux-gnueabihf/libpython3.4m.so (ver 3.4.2)
-- numpy: /home/pi/.virtualenvs/cv/lib/python3.4/site-packages/numpy/core/include (ver 1.10.4)
-- packages path: lib/python3.4/site-packages
--

```

Gambar 3.29 Memastikan Bahwa Python 3 akan Digunakan saat Mengkompilasi OpenCV 3 untuk Raspbian Jessie

```

[ 98%] Built target tutorial_spp_demo
Scanning dependencies of target tutorial_planar_tracking
[ 98%] Building CXX object samples/cpp/Makefiles/tutorial_planar_tracking.dir/tutorial_code/features2D/AKAZE_tracking/plana
r_tracking.cpp.o
Linking CXX executable ../bin/cpp-tutorial-objectDetection
Linking CXX executable ../bin/cpp-tutorial-objectDetection2
Linking CXX executable ../bin/cpp-tutorial-non_linear_svm
[ 98%] Built target tutorial_objectDetection
[ 98%] Built target tutorial_objectDetection2
Scanning dependencies of target tutorial_pointPolygonTest_demo
Scanning dependencies of target tutorial_video_input_psnr_ssim
[ 98%] [ 98%] Building CXX object samples/cpp/Makefiles/tutorial_pointPolygonTest_demo.dir/tutorial_code/ShapeDescriptors/p
ointPolygonTest_demo.cpp.o
Built target tutorial_non_linear_svm
[ 98%] Building CXX object samples/cpp/Makefiles/tutorial_video_input_psnr_ssim.dir/tutorial_code/HighGUI/video-input-psnr-
ssim/video-input-psnr-sim.cpp.o
Scanning dependencies of target tutorial_video_write
[100%] Building CXX object samples/cpp/Makefiles/tutorial_video_write.dir/tutorial_code/HighGUI/video-write/video-write.cpp
.o
Linking CXX executable ../bin/cpp-tutorial-pointPolygonTest_demo
Linking CXX executable ../bin/cpp-tutorial-video-write
[100%] Built target tutorial_pointPolygonTest_demo
[100%] Built target tutorial_video_write
Linking CXX executable ../bin/cpp-tutorial-video-input-psnr-sim
Linking CXX executable ../bin/cpp-tutorial-planar_tracking
[100%] Built target tutorial_video_input_psnr_ssim
[100%] Built target tutorial_planar_tracking

real    72m12.945s
user    228m31.480s
sys     6m20.110s
cv) pi@raspberrypi:~/opencv-3.1.0/build $

```

Gambar 3.30 OpenCV 3 sudah sepenuhnya berhasil diinstall

6. Langkah Ke-enam, Pengetesan OpenCV 3

Pertama-tama pastikan bahwa instalasi OpenCV bekerja dengan baik, dengan cara, buka terminal baru, jalankan perintah *source* dan *workon* dengan kode sebagai berikut :

```

$ source ~/.profile

$ workon cv

$ python

>>> import cv2

>>> cv2.__version__

```

Maka akan tampil dengan tampilan seperti ini pada terminal,

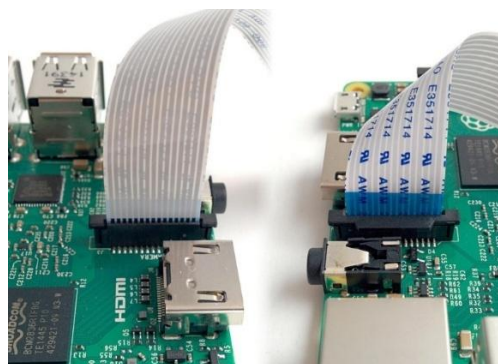
```
pi@raspberrypi:~$ source ~/.profile
pi@raspberrypi:~$ workon cv
(cv) pi@raspberrypi:~$ python
Python 2.7.9 (default, Mar 8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license()" for more information.
>>> import cv2
>>> cv2.__version__
'3.1.0'
>>> |
```

Gambar 3.31 Mengkonfirmasi Bahwa OpenCV 3 Sudah Berhasil di *Install* Pada Raspberry Pi 3 yang Berjalan Pada Raspbian Jessie

3.4.2 Pengaturan Modul Raspberry Pi Camera V2 pada Raspberry Pi 3

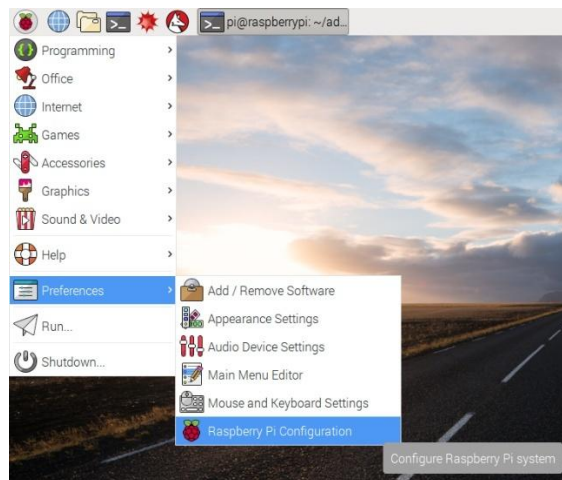
Untuk dapat melakukan pengambilan gambar atau video menggunakan kamera Raspberry Pi Camera V2 maka harus dilakukan instalasi pada kamera tersebut dengan langkah-langkah sebagai berikut :

1. Langkah Pertama, hubungkan kabel konektor pada kamera ke *socket* CSI-2 pada Raspberry Pi 3.



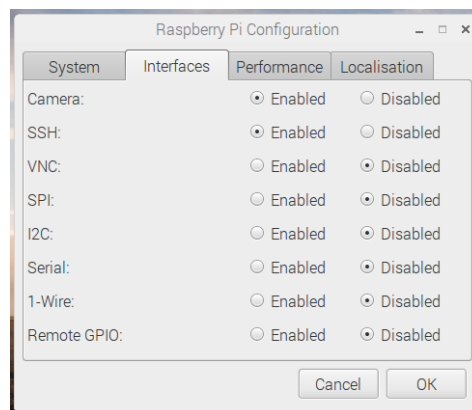
Gambar 3.32 Menghubungkan Kabel *Ribbon* yang ada pada Raspberry Pi Camera V2 ke *socket* CSI-2 Pada Raspberry Pi 3

2. Langkah Ke-Dua, nyalakan komputer Raspberry Pi 3
3. Langkah Ke-Tiga, buka Raspberry Pi *Configuration Tool* dari menu utama



Gambar 3.33 Masuk ke Konfigurasi Raspberry Pi 3

4. Pastikan perangkat lunak kamera diaktifkan



Gambar 3.34 Mengaktifkan Perangkat Lunak Kamera

5. Jika Perangkat Lunak Tersebut Berada pada Posisi Nonaktif (*Disable*) maka ubah pada posisi Aktif (*Enable*)
6. Pengetesan Kamera apakah berfungsi atau tidak, dengan cara membuka *software* Python 3 dan ketikkan kode berikut :

```
from picamera import PiCamera
from time import sleep
```

```
camera = PiCamera()
```

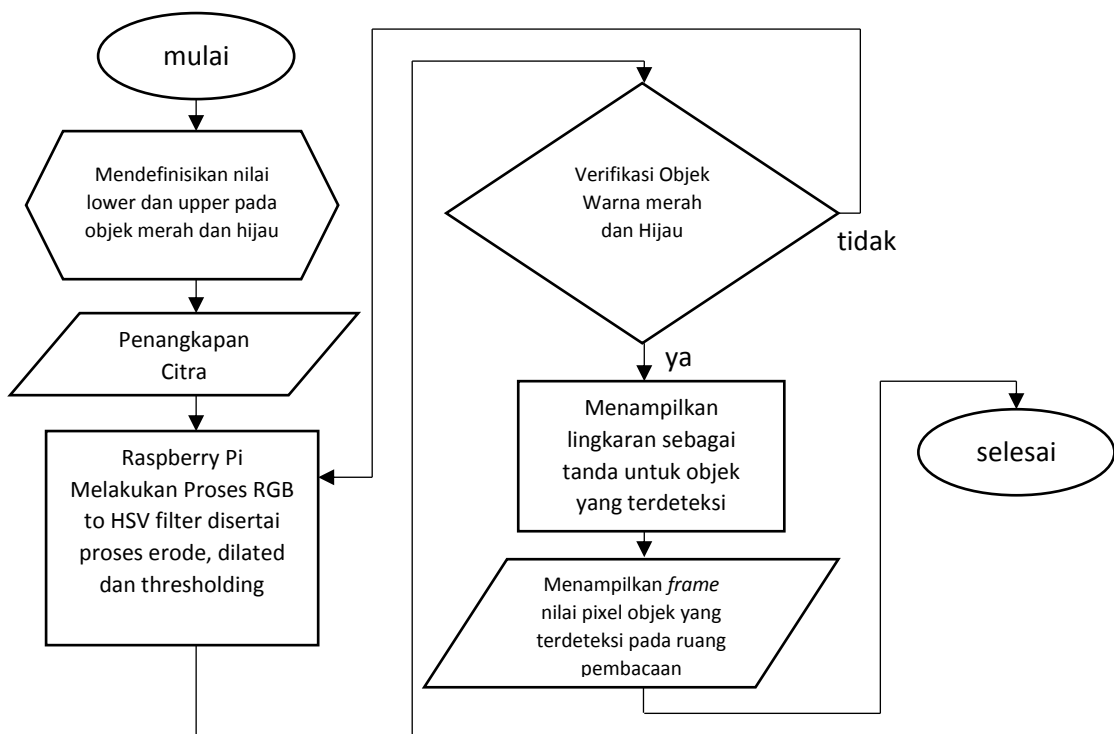


```
camera.start_preview()  
sleep(10)  
camera.stop_preview()
```

ketika pengetikan kode selesai maka tekan ctrl + S pada keyboard untuk mengaktifkan fungsi penyimpanan, lalu tekan F5 pada keyboard untuk menjalankan program diatas.

3.4.3 Algoritma untuk Melakukan Pemindaian Objek Terhadap Citra

Sebelum melakukan pembuatan program untuk dapat melakukan pemindaian objek terhadap citra, maka dibuatlah *flowchart* untuk dapat memudahkan pemahaman pada program yang akan dibuat seperti gambar dibawah.



Gambar 3.35 Blok Diagram Pemindaian Objek Terhadap Citra

3.4.3.1 Penjelasan Blok Diagram Pemindian Objek Terhadap Citra

1) Mendefinisikan Nilai Lower dan Upper pada Objek Merah dan Hijau

Awal mula untuk memulai pemindaian citra, terlebih dahulu untuk mendefinisikan nilai ambang bawah dan ambang atas (*lower and upper*) HSV pada objek yang akan dipindai, yaitu warna merah dan warna hijau. Nilai tersebut disisipkan pada program berikut:

```
# define the lower and upper boundaries of the "green"
# ball in the HSV color space
greenLower = (29, 86, 6)
greenUpper = (64, 255, 255)
pts = deque(maxlen=args["buffer"])
counter = 0
(dx, dy) = (0, 0)
direction = ""

# define the lower and upper boundaries of the "red"
# ball in the HSV color space
redLower = (136, 87, 111)
redUpper = (180, 255, 255)
pts = deque(maxlen=args["buffer"])
counter = 0
(dx1, dy1) = (0, 0)
direction1 = ""
```

Untuk dapat mengetahui nilai ambang atas dan ambang bawah sehingga dapat memindai objek merah dan hijau, dibutuhkan software yang dapat mengkalkulasi nilai RGB menjadi HSV. Nilai HSV yang didapat dimasukan ke dalam program ini. Nilai HSV disini dapat diatur sesuai keadaan pencahayaan disekitar.

2) Penangkapan Citra

Kamera yang digunakan iyalah jenis kamera Raspberry Pi *Camera*, kamera disini melakukan proses penangkapan citra, citra yang di tangkap oleh kamera kemudian dijadikan *input* untuk proses selanjutnya. Untuk dapat menggunakan kamera pada raspberry, dibutuhkan program dan *library* yang dapat menjalankan kamera ini. Dengan *library* yang dibutuhkan sebagai berikut:

```
from picamera.array import PiRGBArray
from picamera import PiCamera
```

Program yang digunakan adalah sebagai berikut:

```
if not args.get("video", False):
    camera = PiCamera()
    camera.resolution = (320, 240)
    camera.framerate = 35
    rawCapture = PiRGBArray(camera, size=(320, 240))
    time.sleep(0.1)

else:
    camera = cv2.VideoCapture(args["video"])

for image in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    frame = image.array
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)
    if key == ord("q"):
        break
```

Program diatas berfungsi mengaktifkan kamera untuk *mode* video, resolusi kamera yang digunakan yaitu 320x420dpi. program ini hanya untuk menjalankan kamera Raspberry Pi *Camera* tidak untuk kamera jenis lain seperti *webcam* atau sebagainya, dibutuhkan program lain apabila menggunakan kamera berjenis *webcam*.

3) Raspberry Pi Melakukan Proses RGB to HSV filter Disertai Proses *Erosions, Dilations dan Thresholding*

Setelah kamera melakukan penangkapan citra, proses berikutnya melakukan penyaringan terhadap citra yang masih memiliki warna RGB menjadi warna HSV, dimana HSV ini berfungsi tidak hanya mengolah warna akan tetapi juga pencahayaan. Dibutuhkan program sebagai berikut untuk dapat melakukan penyaringan warna:

```
# construct a mask for the color "green", then perform
# a series of dilations and erosions to remove any small
# blobs left in the mask
red = cv2.inRange(hsv, redLower, redUpper)
red = cv2.erode(red, None, iterations=2)
red = cv2.dilate(red, None, iterations=2)

# construct a mask for the color "green", then perform
# a series of dilations and erosions to remove any small
# blobs left in the mask
green = cv2.inRange(hsv, greenLower, greenUpper)
green = cv2.erode(green, None, iterations=2)
green = cv2.dilate(green, None, iterations=2)
```

Program diatas tidak hanya untuk memfilter warna RGB menjadi HSV, akan tetapi sudah terdapat proses untuk melakukan pengikisan dan pelebaran serta *thresholding (erosions, dalations, and thresholding)*.

4) Verifikasi Objek Warna merah dan Hijau

Setelah melakukan proses penyaringan dan seleksi warna merah dan hijau, selanjutnya dilakukan verivikasi, apakah objek yang dipindai iyalah benar berwarna merah dan hijau seperti nilai HSV yang sudah ditentukan sebelumnya, jika warna sesuai dengan nilai HSV yang sudah ditentukan maka proses akan dilanjutkan untuk membuat tanda bahwa objek tersebut iyalah benar objek yang berwarna hijau dan merah, jika objek yang dipindai bukan berwarna hijau dan merah sesuai dengan nilai HSV yang sudah ditentukan, maka kembali lagi pada proses penyaringan dan seleksi.

5) Menampilkan Lingkaran Sebagai Tanda untuk Objek yang Terdeteksi

Setelah berhasil memindai objek warna merah dan hijau, selanjutnya dilakukan penambahan tanda berupa lingkaran yang mengikuti objek warna tersebut, supaya dapa dapat mengetahui bahwasannya objek yang dipindai iyalah objek berwarna merah dan hijau. Untuk dapat membuat tanda yang mengikuti objek warna yang telah dipindai, dibutuhkan program seperti berikut:

```
# find contours in the mask and initialize the current
# (x, y) center of the ball
cnts = cv2.findContours(red.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
center = None

cnts1 = cv2.findContours(green.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
center = None

# only proceed if at least one contour was found
if len(cnts) > 0:
    # find the largest contour in the mask, then use
    # it to compute the minimum enclosing circle and
    # centroid
    c = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)
    M = cv2.moments(c)
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

    # only proceed if the radius meets a minimum size
    if radius > 10:
        # draw the circle and centroid on the frame,
        # then update the list of tracked points
        cv2.circle(frame, (int(x), int(y)), int(radius),
                   (0, 255, 255), 2)
        cv2.circle(frame, center, 5, (0, 0, 255), -1)
        pts.appendleft(center)
        print(int(x), int(y))
```

```

show the movement deltas and the direction of movement on
the frame
cv2.putText(frame, direction, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 0, 255), 3)
cv2.putText(frame, "X, Y: {}".format(center), (10, frame.shape[0]- 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

```

Program diatas berupa pembentukan lingkaran serta titik tengah ruang pembacaan, supaya didapat kordinat *pixel*-nya, sehingga apabila objek yang terpindai berada pada kanan *frame* maka akan secara otomatis akan memunculkan nilai pada *pixel* yang menjadi wilayah objek yang telah dipindai, dengan kordinat X = lebar *frame* Y = tinggi *frame*.

6) Menampilkan *frame* nilai *pixel* objek yang terdeteksi pada ruang pembacaan

Ketika semua proses sudah dilakukan, tahap berikutnya menampilkan *frame* pada tampilan komputer, dengan keluaran berbentuk tampilan video citra yang berhasil dipindai. Program dan gambar sebagai berikut:

```

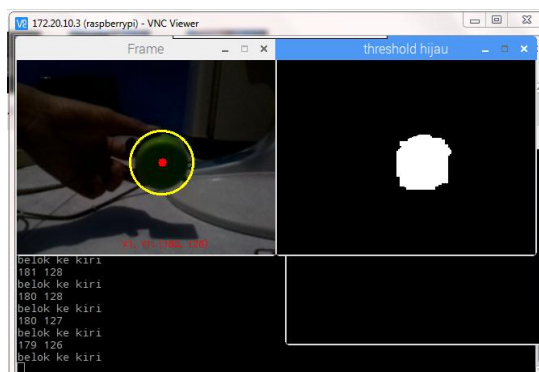
# show the frame to our screen and increment the frame counter
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
counter += 1

# if the 'q' key is pressed, stop the loop
if key == ord("q"):
    break

# cleanup the camera and close any open windows
camera.close()
cv2.destroyAllWindows()

```

Untuk menutup *frame* hanya menekan tombol “q” pada *keyboard* dan akan menghentikan semua proses diatas.



Gambar 3.36 Objek Warna Hijau Berhasil Terdeteksi

3.4.3.2 Algoritma Untuk Mengirimkan Data Umpan Balik pada Raspberry Pi

Untuk mengendalikan kapal cepat ini, masih membutuhkan beberapa program untuk dapat mengendalikan motor dan servo. nilai kordinat yang didapat pada pembacaan citra dijadikan umpan balik bagi mikrokontroler. Komunikasi diperlukan untuk menghubungkan antara komputer dengan mikrokontroler, dengan cara mengaktifkan fungsi komunikasi serial pada Raspberry Pi. Dengan *library* yang dibutuhkan sebagai berikut:

```
#mengaktifkan komunikasi serial ke arduino
arduino = serial.Serial('/dev/ttyS0',baudrate=9600,timeout=3.0)
```

Setelah komunikasi serial sudah aktif, maka dilanjutkan pada algoritma yang akan dikirimkan ke mikrokontroler, dengan program sebagai berikut:

```
#objek merah
if int(x) >= 10 and int(y) >= 60 :
    arduino.write('R'.encode('ascii'))
    print('belok ke kanan')
else :
    arduino.write('F'.encode('ascii'))
    print('lurus')

#objek Hijau
if int(x1) <= 200 and int(y1) >= 60 :
    arduino.write('L'.encode('ascii'))
    print('belok ke kiri')
else :
    arduino.write('F'.encode('ascii'))
    print('lurus')
```

Terdapat dua program yang berbeda diatas, yaitu program untuk objek berwarna merah dan program untuk objek berwarna hijau. Penjelasan dari program untuk objek warna merah, apabila objek berwarna merah melewati kordinat “X” pada ruang pembacaan lebih dari sama dengan 10 dan kordinat “Y” pada ruang pembacaan lebih dari sama dengan 60, maka Raspberry Pi akan mengirimkan data *char* “L” ke Arduino. begitupun juga dengan objek warna hijau, apabila objek berwarna hijau melewati kordinat “X” pada ruang pembacaan kurang dari sama dengan 200 dan kordinat “Y” pada ruang pembacaan lebih dari sama dengan 60 maka Raspberry Pi akan mengirimkan data *char* “R” ke Arduino. Jika kedua objek yang dideteksi tidak melewati batas yang sudah ditentukan maka Raspberry Pi akan

mengirimkan *char* “F” ke Arduino. Fungsi dari masing-masing *char* yang dikirimkan berbeda-beda, tetapi dengan tujuan yang sama yaitu untuk menggerakkan servo dan motor.

3.4.4 Algoritma untuk Menggerakkan Motor dan Servo Pada Mikrokontroler

Mikrokontroler bertugas untuk menggerakkan motor dan servo, sehingga kapal dapat melaju dengan baik. Masukan yang diperoleh mikrokontroler yaitu data *char* komunikasi serial dari Raspberry Pi, data tersebut dipergunakan sebagai isyarat untuk membangkitkan sinyal PWM ke motor dan servo, dengan keluaran motor berputar dan servo menggerakkan *fins*.

```
if (Serial.available()>0)
{
  int baca=Serial.read();

  if(baca=='L')
  {
    brushless.write(65);
    finsservo.write(60);
    Serial.println("Motor Servo Bergerak ke kiri");
  }

  else if(baca=='R')
  {
    brushless.write(65);
    finsservo.write(135);
    Serial.println("Motor Servo Bergerak ke Kanan");
  }

  else
  {
    finsservo.write(90);
    brushless.write(65);
    Serial.println("Motor Servo Lurus");
  }
}
```

Penjelasan pada program diatas, yaitu jika mikrokontroler mendapatkan masukan data *char* “L” maka servo akan menggerakkan *fins* ke kiri, dan jika mendapat masukan “R” maka servo akan menggerakkan *fins* ke kanan. Apabila *char* yang diterima selain *char* “L” dan “R” maka servo akan menggerakkan *fins* lurus, kecepatan motor dibuat sama, karena nilai PWM “65” sudah pada kecepatan rpm terendah.