BAB 5 PERANCANGAN DAN IMPLEMENTASI

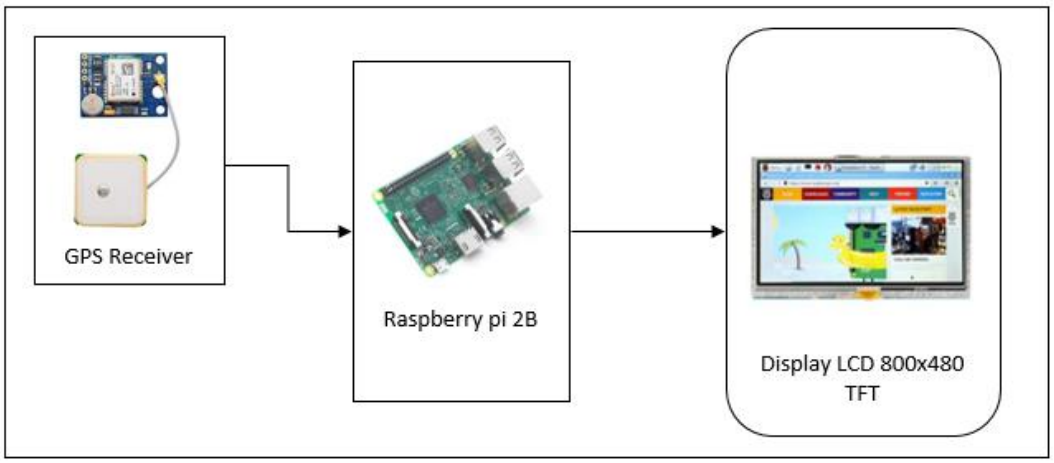
Bab ini membahas perancangan dan implementasi yang dilakukan pada penelitian ini. Perancangan membahas tentang persiapan dan instalasi sehingga sistem yang dirancang siap diimplementasikan. Implementasi membahas mengenai penerapan dari sistem sesuai dengan perancangan yang telah dilakukan sebelumnya.

5.1 Perancangan

Pada bagian ini berisi perancangan secara perangkat keras dan perangkat lunak yang nantinya akan diterapkan pada penelitian " Rancang Bangun GPS *Track Back* pada Rekaman Rute Pendakian Menggunakan Sistem *Embedded* ". Perancangan dilakukan meliputi, perancangan perangkat keras dan perancangan perangkat lunak.

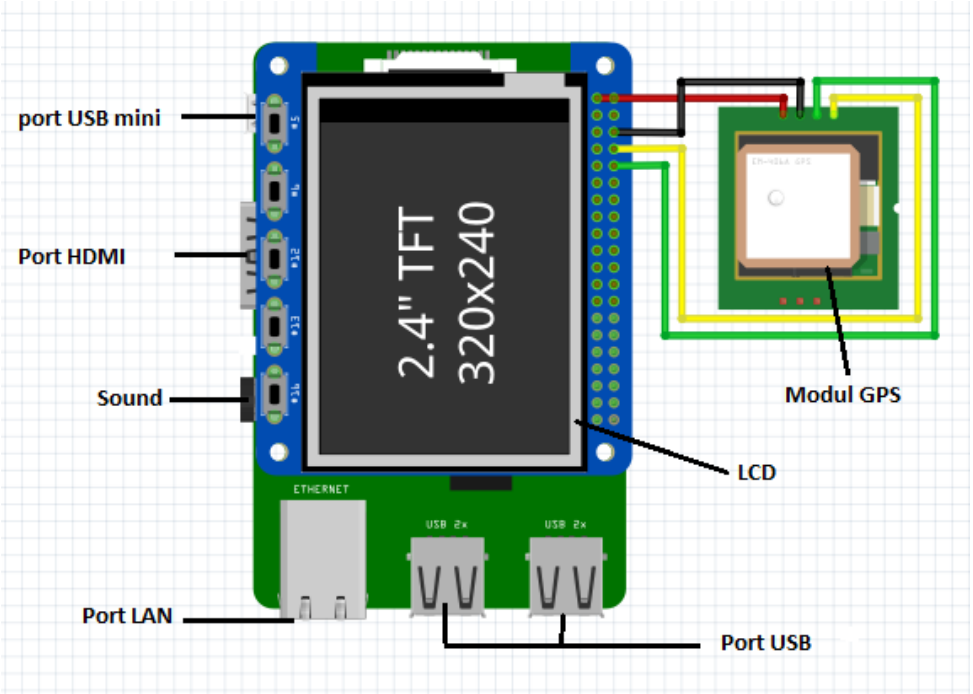
5.1.1 Perancangan Perangkat Keras

Perangkat keras yang digunakan pada sistem ini adalah *Raspberry PI 2B* dan berfungsi sebagai pengolah data masukan dari sensor. Perangkat yang digunakan sebagai input adalah modul GPS U-blox Neo-6m. Modul GPS U-blox Neo-6m digunakan untuk mengambil data GPS *Latitude* dan *Longitude* yang digunakan untuk menentukan titik kordinat lokasi. Kemudian untuk menampilkan hasil pengolahan data digunakan LCD *display* 800 x 480 TFT. Perancangan perangkat keras digambarkan dalam diagram blok sebagai berikut :



Gambar 5.1 Diagram Blok Perancangan perangkat keras

Diagram blok perancangan sistem menjadi acuan dalam perancangan sistem yang dibuat. Komponen penunjang perancangan sistem berupa modul GPS, Raspberry Pi 2B*, Display LCD 800 X 480 TFT* dirangkai menjadi satu sehingga dapat saling terhubung dan bekerja sesuai dengan fungsinya dan digambarkan sebgai berikut :



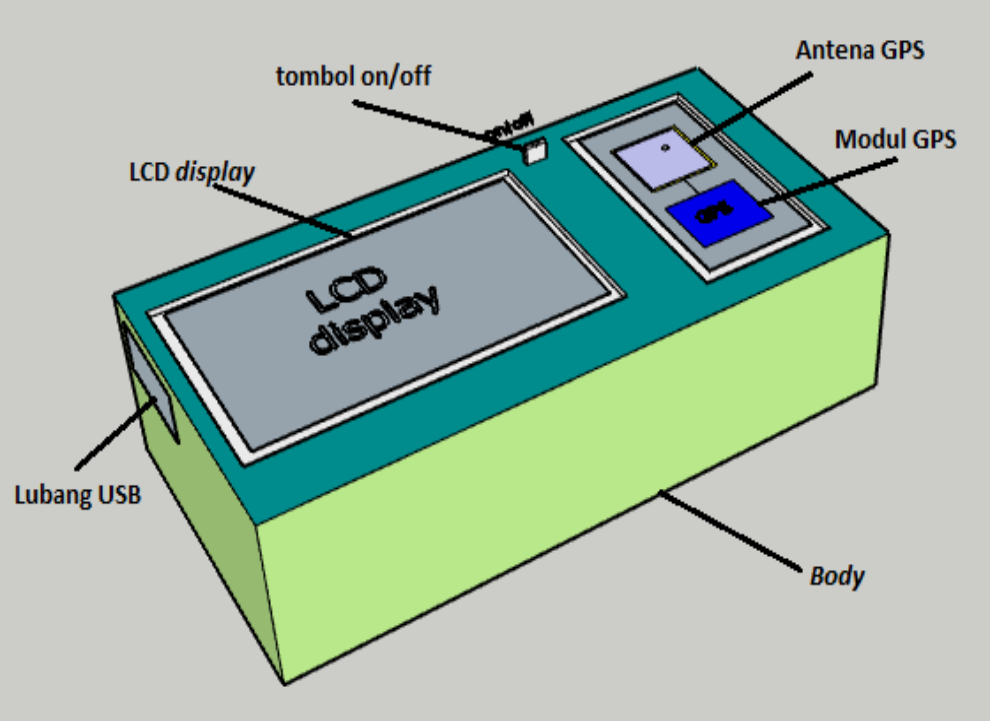
Gambar 5.2 Perancangan perangkat keras

Untuk dapat bekerja sesuai dengan fungsi, seluruh komponen perangkat keras harus saling terhubung dengan spesifikasi port pin dari masing-masing komponen. Seluruh modul yang digunakan dihubungkan dengan Raspberry Pi 2B sebagai pusat pengolah data utama pada alat. Modul Ublox GPS Neo 6M dan Display LCD dihubungkan pada pin GPIO Raspberry Pi 2B menggunakan kabel sesuai dengan spesifikasi pin komunikasi masing – masing modul. Berikut adalah tabel komunikasi antar pin :

Tabel 5.1 Komunikasi pin GPIO

No	Raspberry	Display LCD	Ublox GPS Neo 6M
1	VCC	VCC	VCC
2	GND	GND	GND
3	RX	-	TX
4	TX	-	RX
5	SDA	-	-
6	SCL	SCL	-
7	HDMI	HDMI	-
8	MISO	MO	-
9	MOSI	MI	-
10	IRQ	IRQ	-

11	CS	CS	-

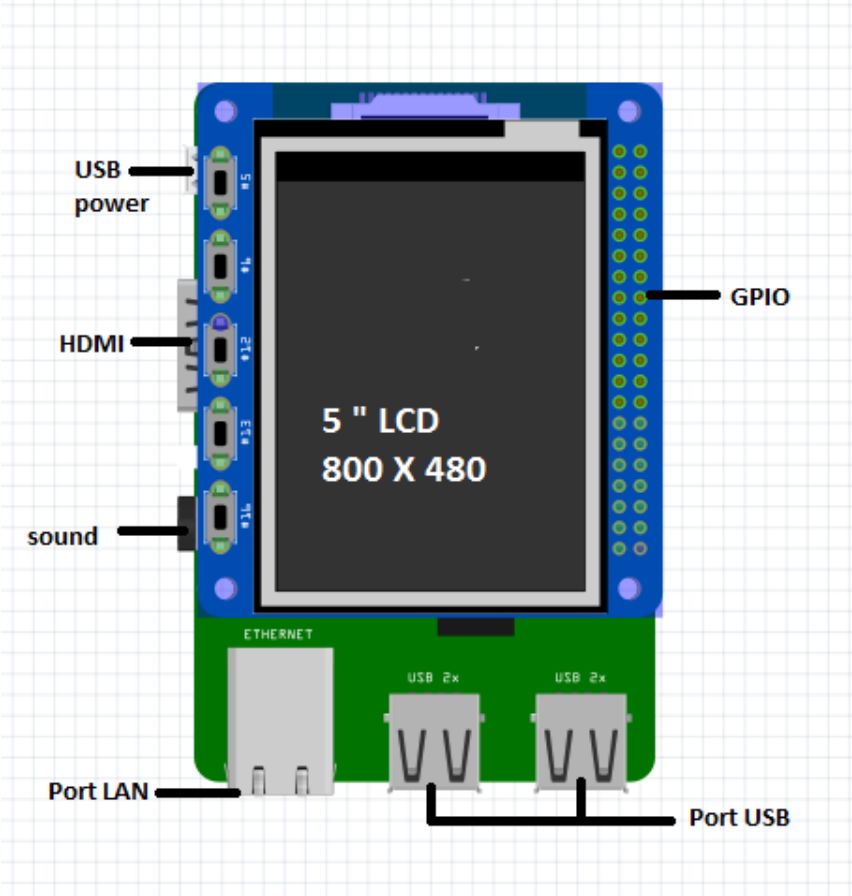


Gambar 5.3 Desain Kemasan Sistem

Untuk memudahkan sistem yang dirancang agar lebih mudah dibawa dan digunakan maka dibuat desain kemasan dari sistem yang digukanakan sebagai wadah dari rangkaian sistem tanpa mempengaruhi kinerja sistem.

a) Perancangan Rangkaian *Display* LCD

Pada perancangan rangkaian *display* digunakan *display* LCD 800 X 480 *touch screen*. LCD akan dirangkai dengan Raspberry Pi 2B dengan menggunkan komunikasi HDMI. Pada *display* LCD yang akan digunakan terdapat beberapa pin komunikasi diantranya 13 x 2 pin GPIO, 1 buah konektor HDMI dan, 1 buah konektor USB power. Desain perancangan *display* LCD dengan Raspberry Pi 2B seperti gambar 5.4 berikut.

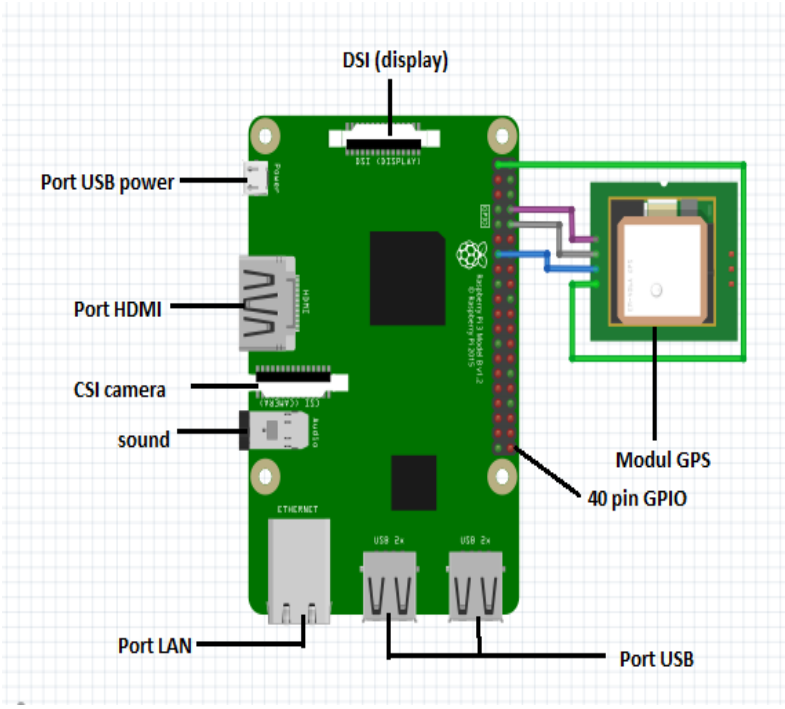


Gambar 5.4 Desain perancangan *display* LCD

Port HDMI pada LCD akan dihubungkan dengan port HDMI pada Raspberry Pi 2B dan pin GPIO pada LCD akan dihubungkan dengan pin GPIO pada Raspberry Pi 2B. fasilitas *Touch Screen* pada LCD dapat berfungsi dengan menghubungkan pin GPIO TPInterrupt(IRQ) 21, TPChipsellect(CS) 26, MI(MOSI) 19, MO(MISO) 21, SCLK 23, VCC 5V 2, GND 25. Dengan desain perancangan pada *display* LCD diharapkan LCD yang digunakan dapat berfungsi sesuai dengan harapan yang diingnkan.

b) Perancangan Rangkaian Modul GPS

Pada perancangan rangkaian modul GPS akan digunakan modul GPS *Ublox Neo 6 M* sebagai sensor pengolah data GPS. Modul GPS Ublox Neo 6 M memiliki 4 pin komunikasi UART yang dapat diimplementasikan dengan Raspberry Pi 2B. Modul GPS yang akan digunkan dihubungkan dengan Raspberry Pi 2B menggunkan pin VCC, RX,TX dan, GND. Gambar 5.5 merupakan desain perancangan rankaian modul GPS dengan Raspberry Pi 2B sebgai berikut.



Gambar 5.5 Perancangan rangkaian modul GPS

Desain perancangan diatas dijelaskan bagaimana modul GPS yang akan digunakan dapat berfungsi dengan Raspberry Pi 2B. Modul GPS Ublox Neo 6 M bekerja pada tengan 3.3 Volt. Kabel berwarna hijau sebagai VCC akan dihubungkan dengan pin GPIO 3.3 Volt pada Raspberry Pi 2B. Sedangkan, kabel warna ungu merupakan RX akan dihubungkan ke pin GPIO TX dan kabel warna abu – abu sebagai TX akan dihubungkan dengan pin GPIO RX pada Raspberry Pi 2B. Kemudian kabel berwarna biru sebagai GND akan dihubungkan dengan pin GPIO GND pada Raspberry Pi 2B. Dengan desain perancangan rangkaian modul GPS diharapkan modul GPS yang digunakan dapat bekerja dengan Raspberry Pi 2B.

Tabel 5.2 Pin komunikasi modul GPS

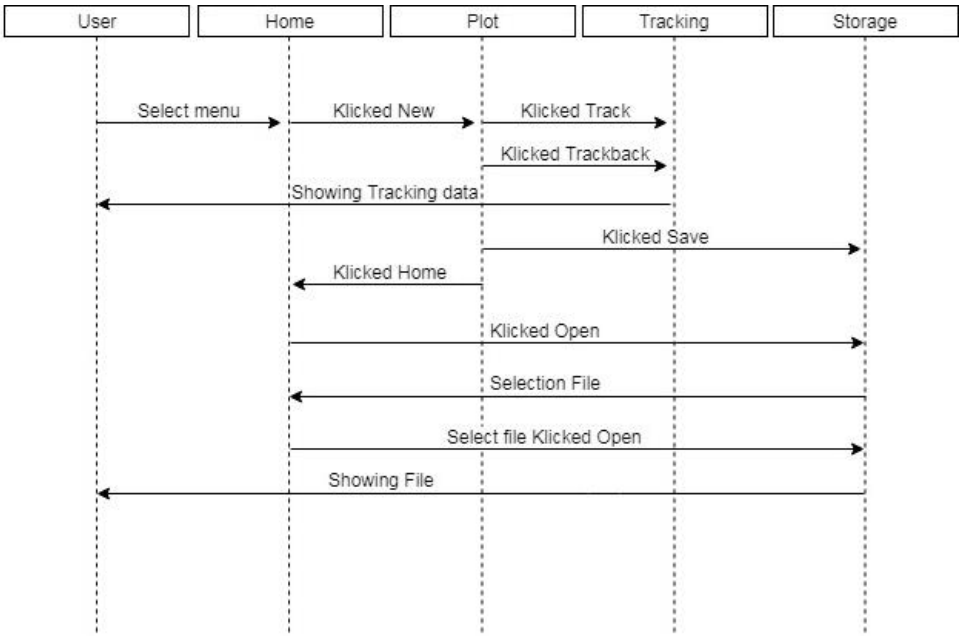
No	Pin GPIO	Pin Modul GPS
1	VCC 3.3 V	VCC
2	GND	GND
3	TX	RX
4	RX	TX

5.1.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak akan membahas perancangan sistem yang dibangun untuk menjalankan perekaman rute pendakian pada alat dengan menggunakan perangkat lunak yang telah dibahas pada bab sebelumnya. Perancangan perangkat lunak bertujuan merancang proses yang akan dikerjakan oleh sistem guna mendapatkan input dari perangkat keras modul GPS yang digunakan untuk diproses oleh Raspberry pi 2B berupa data rekaman rute dan

mendapatkan output yang dapat ditampilkan pada LCD *display* sesuai dengan yang diharapkan. Perancangan yang dilakukan akan dijadikan acuan dalam pembangunan sistem dengan menggunakan bahasa pemrograman Python dan memanfaatkan beberapa *library* penunjang yang digunakan untuk menampilkan data hasil rekaman.

Pada rancang bangun GPS *back track* pada rekaman rute pendakian menggunakan sistem *embedded* diperlukan sebuah sistem yang dapat mengolah data GPS berupa nilai koordinat posisi yang dirubah dalam bentuk garis rute pendakian dan menampilkannya pada LCD *display*. Dalam perancangan sistem ini akan dirancang sebuah *user interface* yang berjalan pada alat beserta sistem pemrosesan alat yang digambarkan dalam sebuah diagram *activity*. Pada gambar 5.6 digambarkan alur aktifitas sistem yang akan berjalan.

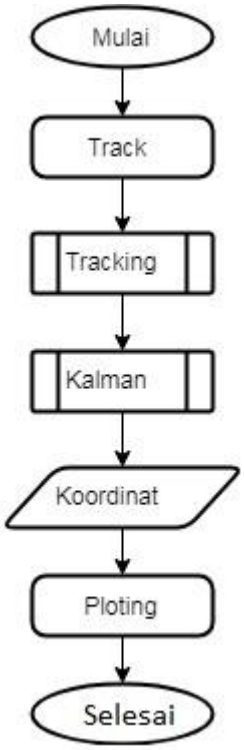


Gambar 5.6 Diagram *Aktivity* Sistem

Pada diagram *activity,* sistem yang akan berjalan pada alat dirancang memiliki tampilan *user interface* yang mempunyai dua frame. Pada frame pertama akan digunakan sebagai halaman *Home* yang menampilkan tombol *NEW, OPEN* dan *EXIT*. Kemudian pada frame kedua sebagai halaman graph yang menampilkan oprasi utama pada alat dalam melakukan perakaman rute yaitu display *Ploting*, tombol *Track, Trackback, Save, Home* dan *Exit*. Alat yang dirancang, menggunakan LCD *display* TFT 800x480 yang dibekali dengan teknologi layar sentuh sehingga dapat mempermudah pengguna dalam melakukan proses perekaman dengan sistetem yang berjalan pada alat. Alur dari proses perekaman diawali pada sistem *Standby* dapat memilih menu pada halaman *HOME, click New* untuk masuk pada halaman graph. Pada halaman graph terdapat beberapa pilihan menu seperti yang telah disebutkan diatas. *Click Track* atau *Trackback* untuk memberikan perintah sensor GPS melakukan *Tracking* data koordinat, sensor GPS akan terus melakukan pengambilan data koordinat hingga pengguna menghentikannya dengan cara *click Stop*. Data koordinat yang peroleh dari sensor akan diproses dengan kalman filter

untuk dilakukan penyaringan data koordinat. Hasil dari proses penyaringan berupa data koordinat update yang kemudian disimpan dalam folder file *temporarry* dan kemudian akan di*Ploting* oleh sistem menjadi tampilan garis rute. Data koordinat selain di*Ploting* menjadi garis koordinat, data juga akan disimpan kedalam folder file penyimpanan jika pengguna dengan *click Save*. Kemudian alur untuk proses membuka kembali file data hasil perekaman, pengguna dapat mnggunakan fungsi *button click OPEN*. Kemudian pengguna dapat memilih file hasil rekaman yang telah disimpan untuk dibuka kembali dan ditampilkan pada LCD *display.*

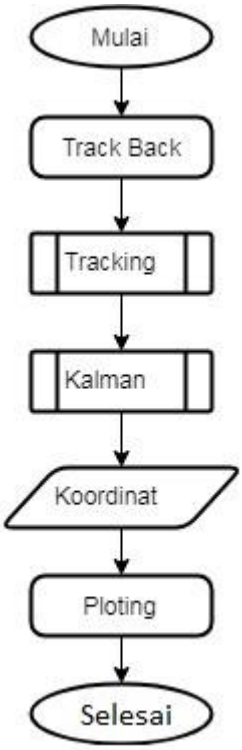
a) Perancangan *Ploting* Rute Dengan Fungsi *Track*



Gambar 5.7 *Flowchart Ploting* Rute *Track*

Gambar 5.7 merupakan gambaran proses *Ploting* rute dengan menggunakan fungsi *track.* Pada saat melakukan perekaman rute yang akan dilalui pengguna akan merekam rute perjalanan awal dengan menggunkan fungsi *track*. Fungsi *track* digunakan untuk melakukan *Ploting* data koordinat dalam bentuk garis rute yang menjadi acuan awal perjalan. Data yang diperoleh dari sensor GPS diolah dengan menggunakan kalman filter utuk mendapatkan nilai koordinat yang telah diupdate oleh kalman filter. Kemudian data disimpan dalam file temporer yang kemudian dibaca olehh sistem untuk di*Ploting* menjadi garis rute dengan warna biru.

b) Perancangan *Ploting* Rute Dengan Fungsi *Trackback*

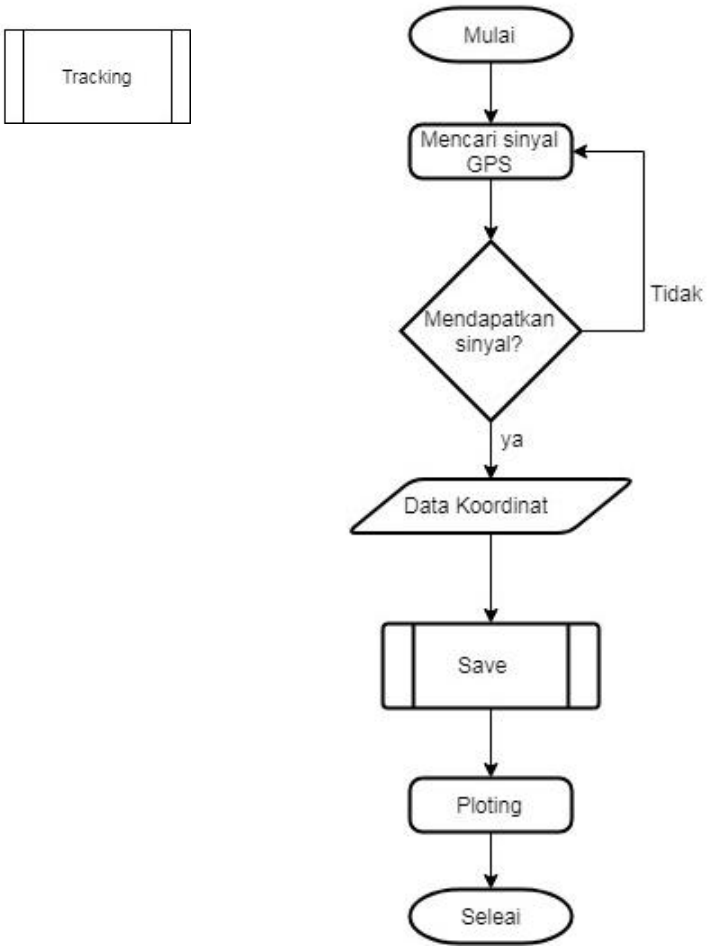


Gambar 5.8 *Flowchart Ploting* Rute *Trackback*

Gambar 5.8 menggambarkan perancangan *Ploting* rute dengan menggunakan fungsi *trackback* yang dapat melakukan *Ploting* data koordinat untuk menuntun pengguna kembali ke rute awal ketika rute yang dilewati menyesatkan pengguna atau menemui jalan buntu. Sehingga pada sistem dirancang sebuah fungsi yang dapat melakukan tracking data GPS dan kemudian diolah menjadi garis rute yang ditampilkan pada LCD *display*. Serupa dengan proses *Ploting* rute awal, pada sistem dirancang fungsi *trackback* yang digunakan untuk mengolah data GPS menjadi garis rute perjalanan dengan warna garis orange.

c) Sub-Program *Tracking* Data GPS

Tracking Data GPS adalah perancangan yang digunakan untuk proses pengambilan data GPS dari satelit dalam menentukan koordinat *Latitude, Longitude*. Proses pengambilan data dilakukan oleh modul GPS Ublox Neo 6M dengan menerima sinyal dari satelit yang mengorbit pada bumi. Jika modul mendapatkan sinyal dari satelit maka sistem dapat menentukan koordinat *Latidute* dan *Longitude*. Apabila tidak mendapatkan sinyal dari satelit, sistem akan mengulangi proses pencarian sinyal hingga mendapatkannya. Setelah modul GPS mendapatkan sinyal dari satelit data GPS (*Latitude, Longitude*) kemudian akan diproses untuk disimpan sebagai file rekaman dan pembuatan rute.

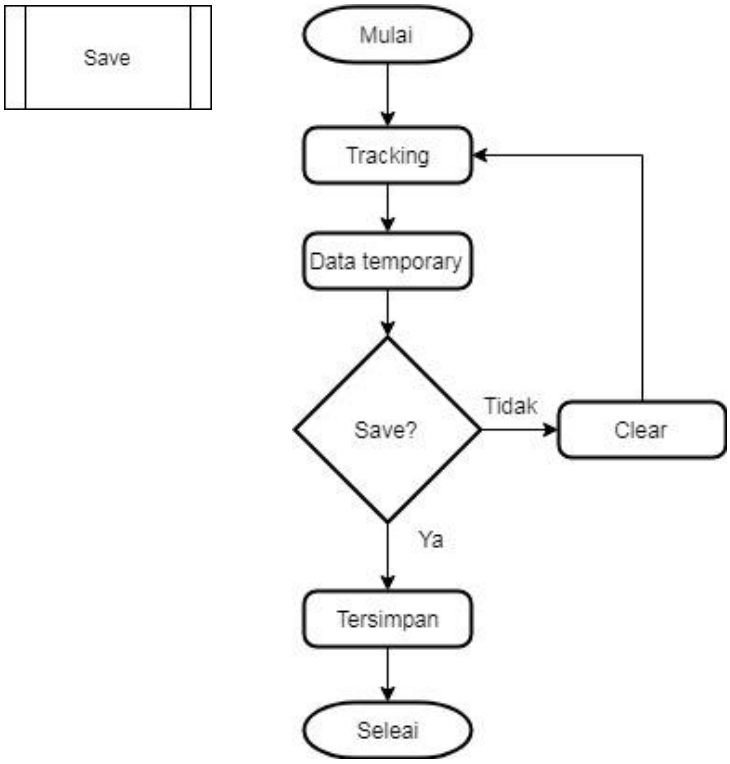


Gambar 5.9 *Flowchart* Sub-Program *Tracking* Data GPS

Pada sistem yang dirancang akan ditambah dengan algoritma kalman filter untuk mendapatkan nilai akurasi dari data yang diproses. Kalman filter akan mengestimasi nilai waktu awal (Prediksi) pembacaan data GPS untuk mendapatkan nilai waktu saat ini pembacaan sensor yang digunakan sebagai nilai koreksi sebagai hasil filter (*update*) untuk olah sistem. Data koordinat kemudian di*Ploting* dalam grafik berupa garis hasil perjalanan yang dibedakan dengan warna biru untuk *Track* dan warna orange untuk garis *Trackback*.

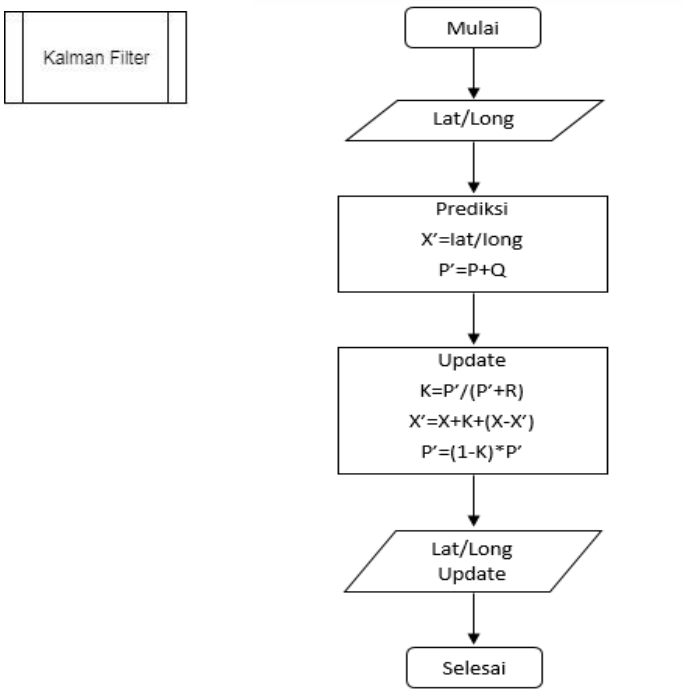
d) Sub-program *Save data*

Pada perancangan *Save data*, sistem akan menyimpan data yang diperoleh pada proses *tracking* sesuai dengan keinginan pengguna. Data hasi *tracking* akan sisimpan dalam sebuah file temporarry. Data dapat disimpan ketika pengguana menekan tombol *Save* pada tampilan grap. Jika pengguna tidak menyimpan data maka, data akan dibersihkan pada saat pengguna kembali melakukan proses *tracking*, data lama yang tidak disimpan akan dihapus dan digantikan dengan data hasil *tracking* yang baru. Alur penyimpanan data sperti gamabar berikut.



Gambar 5.10 *Flowchart* Sub-program *Save* Data

e) Sub-Program Kalman Filter



Gambar 5.11 *Flowchart* Sub-Program Kalman Filter

Pada perancangan sistem ini digunakan kalman filter sebagai penyaring data GPS. Dalam pangambilan data dari modul GPS yang digunakan, data yang diterima digunakan sebagai input pada proses pengolahan data, termasuk data yang

mengandung gangguan (*noise*). Gangguan tersebut terjadi akibat pengaruh perangkat elektronik lain di sekitar penggunaan sistem. Untuk meredam data gangguan, yang dijadikan input harus disaring terlebih dahulu dengan menggunakan kalman filter. Kalman filter akan mengestimasi keadaan suatu proses yang terjadi sebelum saat ini, dan yang akan datang. Proses penyaringan data kalman filter memiliki dua proses utama. Pertama kalman filter akan mengestimasi nilai koordinat sebelumnya (X) untuk mendapatkan nilai koordinat saat ini atau proses prediksi. Kemudian nilai koordinat sebelumnya (X') akan digunakan sebagai informasi untuk mendapatkan nilai saat ini atau proses *Update*. Kedua proses ini akan terus berjalan setiap sistem mendapatkan nilai input data koordinat dari GPS.

5.2 Implementasi

5.2.1 Implementasi Desain sistem GPS *Track Back*

Gambar 5.12 Implementasi Desain GPS *Track Back*

Pada implementasi perangkat keras secara keseluruhan yang telah dirancang, terdapat rangkaian komponen yang akan dipasang di dalam wadah yang sudah didesain untuk memperoleh bentuk secara tampilan dari alat yang dirancang. Hasil implementasi dari desain wadah yang dibuat adalah seperti gambar 5.9 berikut. Didalam wadah yang dibuat terdapat rangkaian keseluruhan komponen yang sudah digabungkan antar komponen yang meliputi sebagai berikut :

1. *Display LCD 800 x 480*
2. Modul *GPS U-blox Neo 6 m*
3. Kabel USB tipe B mini
4. Raspberry Pi 2B
5. Power Bank

a) Implementasi Display LCD 800 x 480

Pada implementasi Display LCD 800 x 480 pada sistem dijelaskan rangkaian perangkat keras untuk dapat berjalan sesuai dengan fungsi yang diharapkan sebagai berikut :

Gambar 5.13 Implementasi Rangkaian Display LCD 800 x 480

Rangakian *display* LCD yang digunakan pada sistem menggunakan XPT2046 *Touch Controller* 5*inch* HDMI *Display* dengan resolusi 800 x 480 *Pixel* yang dapat diimplementasikan dengan Raspberry Pi 2B. LCD yang digunakan dibekali dengan 13 x 2 pin GPIO, port HDMI, port Mini USB, Backlight on/off, dan TFT *touch screen*. Untuk dapat bekerja pada *Raspberry Pi 2B*, LCD dihubungkan dengan *connector* HDMI, pin VCC, GND, MOSI, MISO, IRQ, CS, SCLK untuk mengaktifkan fungsi *touch screen.* sedangkan untuk menampilkan *display* menggunakan sambungan HDMI.

b) Implementasi Modul GPS Ublox Neo 6M

Pada implementasi perangkat keras digunakan module GPS *Neo* 6M sebagai komponen pengambil data *Longitude* dan *Latitude.* Module GPS Ublox *Neo* 6M dibekali dengan pin VCC, GND, RX dan TX. Module dapat bekerja pada tegangan 3,5 V pada *Raspberry Pi 2B*. Sebagai komunikasi untuk dapat mengirimkan data pin RX – TX pada modul disambungkan bersilang dengan GPIO RX – TX pada *Raspberry Pi 2B*. RX diartikan sebagai *Receiver* sedangkan TX diartikan sebagai *Transmitter*. Berikut adalah gambar implementasi perangkat keras modul Ublox Neo 6M :

Gambar 5.14 Implementasi Modul Ublox Neo 6M

Untuk dapat menjalankan modul GPS pada *Raspberry Pi 2B* dibutuhkan beberapa konfigurasi melalui *comand line*terminal pada sistem oprasi *Raspbian*

jessy. Langkah pertama adalah dengan menonaktifkan port *serial* pada *Raspberry Pi 2B* . kemudian masuk kedalam *cmdline* untuk mengubah baudrate dengan mengetikakan `sudo nano /boot/cmdline.txt` dan hapus `console=ttyAMA0,115200` kemudian save. Setelah konfigurasi dilakukan, instal `sudo apt-get install gpsd gpsd-clients`. Setelah instalasi selesai, kemudian yang awalnya baudrete 115200 di hapus, jalankan baudrete 9600 dengan cara mengetikkan `stty -F /dev/ttyAMA0 9600`, kemudian aktifkan gpsd.sock dengan `sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock` dan gps siap digunakan dengan perintah *cgps –s.*

5.2.2 Implementasi Perangkat Lunak

Pada Implementasi perangkat lunak dilakukan pembuatan program untuk melakukan pengambilan data kordinat (*Latitude, Longitude*), proses plot data GPS menjadi rekaman rute dengan pembahasan berdasarkan fungsi utama pada sistem yang dirancang. Dalam rancang bangun GPS track back pada rekaman rute pendakian ini yang menjadi pokok pembahasan adalah bagaimana sistem yang dirancang dapat melakukan proses perekaman rute berdasarkan data GPS. Pada implementasi proses perekaman yang menjadi bagian utama yang mempunyai kendali melakukan proses perekaman adalah fungsi *Track* dan *TrackBack* untuk menampilkan hasil perekaman rute pendakian.

a) Implementasi Library yang Digunakan

Pada sub bab implementasi sistem perekam rute pendakian, digunakan beberapa *library* dalam pembuatan program. *Library* yang digunakan antara lain *tkinter yang* digunakan untuk pembuatan *user interface* atau tampilan yang dapat menampilkan proses perekaman dengan menggunakan bahasa pemrograman *Python* pada *Raspberry Pi 2B*. kemudian digunakan *librarry matplotlib* yang digunakan untuk mengolah data GPS yang didapatkan untuk dirubah menjadi tampilan grafik koordinat dan.

Tabel 5.3 *Library*

No	Librarry
1	`import tkinter as tk`
2	`import matplotlib`

b) Implementasi Button Click Track

Pada bagian implementasi program *Track* ini dibuat proses fungsi *button click* dengan nama *Track* yang digunakan untuk melakukan perjalanan awal pada saat melakukan perekaman sebagai acuan rute kembali *(Back Track)*. *Button click Track* dibuat untuk proses pengambilan data GPS berupa nilai koordinat *Latitude* dan *Longitude* dari sensor GPS yang digunakan untuk selanjutnya diplot menjadi garis kordinat perjalanan yang dapat dilihat dalam tampilan LCD *display*. Berikut kode program fungsi *Track* untuk pengambilan data GPS :

Tabel 5.4 Fungsi *Button Click Track*

NO	Kode Program Button Click Track
1	. . . . .
2	track_button = tk.Button(buttonframe, text="Track",
3	height=2, width=12, relief=tk.SOLID, bd=1,
4	command=self.track_clicked)
5	track_button.pack(side=tk.TOP, pady=4)
6	
7	def track_clicked(self):
8	global gps_stats, track_button
9	self.tr = threading.Thread(target=self.run_tracking)
10	self.tr.start()
12	track_button.configure(text="Stop", bg=conf.BLUE,
11	fg="white", command=self.stop_tracking)
12	gps_stats.config(text="Tracking Started",
13	fg=conf.BLUE)

Pada tabel diatas dibuat sebuah fungsi *button click Track* yang akan menjalankan proses pengambilan data GPS dan proses perekaman rute apabila fungsi *button click* diaktifkan dengan cara disentuh pada layar LCD. *Button click Track* akan mengarahkan kedalam fungsi *track_clicked*. Didalam fungsi ini terdapat beberapa konfigurasi yang akan merubah tampilan *button click Track* setelah ditekan, tampilan yang awalnya berupa *text="Track"* akan berubah menjadi *text="Stop"* dengan warna biru sebagai tanda fungsi *button click Track* yang dibuat berfungsi dan dapat digunakan.

c) Implementasi Proses *Run Tracking*

Pada bagian pengambilan data GPS dan perekaman dalam bentuk tampilan rute yang dilewati berdasarkan data koordinat GPS diperlukan sebuah fungsi yang dapat mengaktifkan sensor GPS untuk menangkap sinyal satelit untuk mendapatkan data GPS dan sekaligus melakukan perekaman pada sistem. Berikut tabel program fungsi *run_tracking* :

Tabel 5.5 Fungsi *Run Tracking*

NO	Kode Program Tracking
1	# -------------------------
2	# -- Run tracking process --
3	# -------------------------
4	def run_tracking(self):
5	global gps_stats, track_done
6	try:
7	self.running = True
8	self.gpsd = gps(mode=WATCH_ENABLE)
10	self.track_data = open(conf.TRACK_TMP_FILE, "w+")
11	while self.running:
12	self.gpsd.next()
13	latitude = self.gpsd.fix.latitude
14	longitude = self.gpsd.fix.longitude
15	if not math.isnan(latitude) and not
16	math.isnan(longitude) and latitude != 0.0 and
17	longitude != 0.0:

```
18                  # Data format
19                  data =
20                  "{:.6f},{:.6f}".format(float(latitude),float(
21                  longitude))
22                  # Write data
23                  self.track_data.write(str(data) + '\n')
24                  self.track_data.flush()
25                  os.fsync(self.track_data)
26                  print(data)
27                  gps_stats.config(text="Run Tracking",
28                  fg=conf.BLUE)
29              else:
30                  print "No Signal..."
31                  gps_stats.config(text="No Signal",
32                  fg=conf.RED)
33              time.sleep(2)
34
35      except (KeyboardInterrupt, SystemExit):
36          if self.running:
37              self.running = False
38              self.tr.join()
39              track_done = True
40          print "Tracking Stopped"
```

Dari program fungsi *run_tracking* ini dibuat alur proses perekaman mulai dari awal sensor GPS aktif. Fungsi *run tracking* akan melakukan menjalankan tugasnya ketika *button click Track* aktif. Proses pertama yang dilkaukan oleh fungsi *run tacking* adalah mengambil data dari modul GPS dengan mengaktifkan *mode watch enable* dan kemudian membuka file baru untuk merekam data GPS yang didapat oleh modul GPS. Fungsi yang dibuat hanya mengambil data koordinat *Latitude* dan *Longitude* yang memiliki nilai selain 0, dengan format hingga 6 angka dibelakang koma {0.6F} format tersebut berdasarkan nilai DMS (*Degree Minute Secon).* Data koordinat yang didapat, selain disimpan dalam file temporary akan langsung diplot kedalam grafik dengan menggunakan fungsi *flush*. Dengan fungsi *flush* pengguna dapat melihat proses perekaman rute pada tampilan LCD *display* yang *diupdate* setiap 2 detik. Untuk mengetahui GPS mendapatkan sinyal dari satelit dan proses perekaman terus berjalan, dibuat sebuah pesan notifikasi *"run Tracking"* dengan warna biru yang muncul di bagian kanan bawah. Apabila GPS gagal mendapatkan data koordinat dari satelit untuk direkam maka, muncul pesan notifikasi *"No Signal"* dengan warna merah sebagai tanda gps sedang tidak mendapatkan data koordinat untuk direkam. Proses perekaman dan *Ploting* data koordinat GPS akan terus berulang hingga dihentikan dengan cara mengaktifkan *button click Stop.* Data koordinat GPS yang telah didapat dan ditulis dalam file *temporary* dapat disimpan secara permanen oleh pengguna diakhir proses perekaman.

d) Implementasi Button Click TrackBack

Pada sub bab implementasi program *TrackBack* dibuat *button click* dengan nama *TrackBack*. Setelah proses perekaman rute perjalanan dengan menggunakan fungsi *track* selasai, dapat dilakukan proses perekamna berikutnya dengan fungsi *button click TrackBack. Button click TrackBack* dibuat untuk proses

pengambilan data GPS berupa nilai koordinat *Latitude* dan *Longitude* dari sensor GPS yang kemudian data tersebut digunakan untuk proses perekaman perjalanan kembali (*Back Track*) yang selanjutnya diplot menjadi garis rute berdasrkan kordinat perjalanan yang dapat dilihat dalam tampilan LCD *display* dalam tampilan grafik. Berikut kode program fungsi *TrackBack* untuk pengambilan data GPS perjalanan kembali (*Back Track)*:

Tabel 5.6 Fungsi *Button Click TrackBack*

NO	Kode Program Button Click Track Back
1	`# TrackBack button`
2	`trackback_button = tk.Button(buttonframe,`
3	`  text="TrackBack", height=2, width=12, relief=tk.SOLID,`
4	`  bd=1,state="disabled", command=self.trackback_clicked)`
5	`trackback_button.pack(side=tk.TOP, pady=4)`
6	
7	`def trackback_clicked(self):`
8	`  global gps_stats, trackback_button`
9	`  self.tb =`
10	`  threading.Thread(target=self.run_trackback)`
11	`  self.tb.start()`
12	`  trackback_button.configure(text="Stop",`
13	`  bg=conf.ORANGE, fg="white",`
14	`  command=self.stop_trackingback)`
15	`  gps_stats.config(text="TrackBack Started",`
16	`  fg=conf.ORANGE)`

Pada tabel diatas dibuat sebuah fungsi *button click TrackBack* yang akan menjalankan proses pengambilan data GPS dan proses perekaman rute kembali dengan cara menyentuh tombol *TrackBack* pada LCD *display* untuk mengaktifkan fungsi *button click TrackBack*. *Button click TrackBack* akan mengarahkan kefungsi *trackback_clicked*. Didalam fungsi ini terdapat beberapa konfigurasi yang akan merubah tampilan *button click TrackBack* yang mualanya berupa *text="TrackBack"* setelah ditekan atau diaktifkan akan berubah menjadi tampilan dengan format *text="Stop"* dengan warna oranye sebagai tanda fungsi *button click TrackBack* yang dibuat berfungsi dan dapat digunakan.

e) Implementasi fungsi Run Tracking Back

Pada bagian pengambilan data GPS dan perekaman dalam bentuk tampilan rute yang dilewati berdasarkan data koordinat GPS diperlukan sebuah fungsi yang dapat mengaktifkan sensor GPS untuk menangkap sinyal satelit untuk mendapatkan data GPS dan sekaligus melakukan perekaman pada sistem. Berikut tabel program fungsi *run_trackingback*:

Tabel 5.7 Fungsi *Run Track Back*

No	Kode Program Track Back
1	`  # ------------------------------`
2	`  # -- Run trackingback process --`
3	`  # ------------------------------`
4	`  def run_trackback(self):`

```
5           global gps_stats, trackback_done
6           try:
7               self.running = True
8               print ""
9               print "TrackingBack Started"
10              print "----------------"
11              self.trackback_data =
12              open(conf.TRACKBACK_TMP_FILE, "w+")
13              while self.running:
14                  self.gpsd.next()
15                  latitude  = self.gpsd.fix.latitude
16                  longitude = self.gpsd.fix.longitude
17                    if not math.isnan(latitude) and not
18                     math.isnan(longitude) and latitude != 0.0
19                     and longitude != 0.0:
20                       data =
21                        "{:.6f},{:.6f}".format(float(latitude),f
22                        loat(longitude))
23                       self.trackback_data.write(str(data) +
24                       '\n')
25                       self.trackback_data.flush()
26                       os.fsync(self.trackback_data)
27                       print data
28                       gps_stats.config(text="Run TrackBack",
29                        fg=conf.ORANGE)
30                  else:
31                      print "No Signal..."
32                      gps_stats.config(text="No Signal",
33                       fg=conf.RED)
34                  time.sleep(2)
35          except (KeyboardInterrupt, SystemExit):
36              if self.running:
37                  self.running = False
38                  self.tb.join()
39                  trackback_done = True
40              print "Tracking Stopped"
```

Dari tabel fungsi *run_trackingback* ini dibuat proses yang memberikan perintah untuk melakukan pengambilan data GPS. Fungsi *run_trackingback* akan menjalankan tugasnya ketika *button click Track* diaktifkan dengan cara disentuh pada LCD *display* yang memiliki kemampuan *touch screen*. Proses pertama yang dilkaukan oleh fungsi *run tacking back* adalah mengambil data dari modul GPS dengan mengaktifkan *mode= watch enable* dan kemudian membuka file *temporary* baru untuk menyimpan data koordinat yang diperoleh dari modul GPS dan nantinya akan disimpan sebagai file hasil perekaman data GPS. Fungsi yang dibuat hanya mengambil data koordinat *Latitude* dan *Longitude* yang memiliki nilai selain 0, dengan format hingga 6 angka dibelakang koma {0.6F} format tersebut berdasarkan nilai DMS (*Degree Minute Secon).* Data koordinat yang didapat selain disimpan dalam file temporary juga akan langsung diplot kedalam grafik dengan menggunakan fungsi *flush*. Dengan fungsi *flush* pengguna dapat melihat proses perekaman rute pada tampilan LCD *display* yang *diupdate* setiap 2 detik. Untuk mengetahui GPS mendapatkan sinyal dari satelit dan proses perekaman terus berjalan, dibuat sebuah pesan notifikasi *"run Trackback"* dengan

warna orange yang muncul dibagian kanan bawah. Apabila GPS gagal mendapatkan data koordinat dari satelit untuk direkam maka, muncul pesan notifikasi *"No Signal"* dengan warna merah sebagai tanda gps sedang tidak mendapatkan data koordinat untuk direkam. Proses perekaman dan *Ploting* data koordinat GPS akan terus berulang hingga dihentikan dengan cara mengaktifkan *button click Stop.* Data koordinat GPS yang telah didapat dan ditulis dalam file *temporary* dapat disimpan secara permanen oleh pengguna diakhir proses perekaman.

f) Implementasi Fungsi *Ploting* Data

Data GPS yang telah didapat melalui fungsi *track* dan *trackback* diatas kemudian akan diproses pada fungsi plot untuk dirubah menjadi grafik. Pada grafik yang terbaca pengguna dapat meliahat rute perjalanan yang sedang direkam dalam bentuk rute hasil *Ploting* dari koordinat *Latitude* dan *Longitude*. Berikut tabel proses *Ploting* dari sistem :

Tabel 5.8 Fungsi *Ploting* Data

NO	Kode Program Plot data
1	`def animate_a(i):`
2	`    # Check if file exists, if not create files`
3	`    if not os.path.exists(conf.TRACK_TMP_FILE) and not`
4	`     os.path.exists(conf.TRACKBACK_TMP_FILE):`
5	`        # Create track data files`
6	`        trackData = open(conf.TRACK_TMP_FILE, "w")`
7	`        trackData.write()`
8	`        trackData.close()`
9	`        # Create trackback data files`
10	`        trackbackData = open(conf.TRACKBACK_TMP_FILE,`
11	`        "w")`
12	`        trackbackData.write()`
13	`        trackbackData.close()`
14	
15	`    trackData = open(conf.TRACK_TMP_FILE, "r").read()`
16	`    trackbackData`
17	`     =open(conf.TRACKBACK_TMP_FILE,"r").read()`
18	`    trackList = trackData.split('\n')`
19	`    trackbackList = trackbackData.split('\n')`
20	`    tr_xList = []`
21	`    tr_yList = []`
22	`    for eachLine in trackList:`
23	`        if len(eachLine) > 1:`
24	`            lat, long = eachLine.split(',')`
25	`            tr_xList.append(float(long))`
26	`            tr_yList.append(float(lat))`
27	`    tb_xList = []`
28	`    tb_yList = []`
29	`    for eachLine in trackbackList:`
30	`        if len(eachLine) > 1:`
31	`            lat, long = eachLine.split(',')`
32	`            tb_xList.append(float(long))`
33	`            tb_yList.append(float(lat))`
34	`    plot_a.clear()`
35	`  tracking_line, = plot_a.plot(tr_xList, tr_yList,`
36	`   conf.BLUE, lw=2)`

```
37    trackback_line, = plot_a.plot(tb_xList, tb_yList,
36     conf.ORANGE, lw=2)
```

Data GPS yang diperoleh dari hasil *tracking* dan *trackback* selanjutnya diproses untuk ditampilakan pada LCD *display*. Proses diawali dengan sistem akan memastikan file *temporary Track* dan *TrackBack* dalam keadaan tidak menerima input data GPS. Jika dalam file sudah dipastikan kosong maka, file *temporry Track* dan *trackback* akan menerima data GPS untuk ditulis. File yang pertama kali terisi adalah file *Track* yang digunakan sebagai acuan rute awal pendakian sebelum melakukan perjalanan kembali (*Back Track*). Sistem akan membaca data GPS dalam file *temporarry* secara berurutan dimulai dari file *temporarry track* kemudian *trackback*. Data koordinat GPS *latitude* dan *longitude* yang berada didalam file kemudian dibaca oleh sistem untuk dirubah menjadi grafik yang menggambarkan rute perjalanan yang sedang direkam secara berurutan sesuai masukan data yang diterima dari modul GPS. Garis rute yang dibuat dari hasil perjalan dalam grafik dibedakan menjadi dua garis warna yaitu grais rute untuk hasil *Track* dibuat dengan warna biru sedangkan untuk *TrackBack* dibuat dengan warna orenge. Tujuan dari perbedaan warna agar menjadi pembeda antara grafik rute hasil *Track* dan *TrackBack*.

g) Implementasi Kalman Filter

Pada rancang bangun GPS *Back Track* pada rekaman rute pendakian ini digunakan algoritma kalaman filter sebagai algoritma filter data untuk meminimalisir nilai gangguan pada data GPS yang digunakan sebagai data masukan proses perekaman. Kalman filer memiliki dua proses untuk dapat memprediksi nilai dari sensor GPS. Proses prediksi merupakan langkah pertama untuk mendapatkan nilai prediksi kesalahan sensor yang akan menjadi umpan balik dalam proses *measurement update*. Hasil dari implementasi kalman filter dijadikan sebagai perbandingan hasil rekaman yang dilakukan tanpa menggunkan filter data. Implementasi kalman filter pada sistem dijabarkan pada tabel 5.9 berikut :

Tabel 5.9 Implementasi Kalman Filter

NO	Kode Program Kalman Filter
1	`# inisialisasi kalman filter latitude`
2	`Qlat = 0 #inisialisasi Q latitude`
3	`Rlat = 1 #inisialisasi R latitude`
4	`xhatlat = 0 #inisialisasi xhat latitude`
5	`Plat = 0.00001 #inisialisasi P latitude`
6	
7	`# inisialisasi kalman filter longitude`
8	`Qlong = 0 #inisialisasi Q longitude`
9	`Rlong = 1 #inisialisasi R longitude`
10	`xhatlong = 0 #inisialisasi xhat longitude`
11	`Plong = 0.00001 #inisialisasi P longitude`
12	
13	`while self.running:`
14	`# gpsd func`
15	`self.gpsd.next()`
16	`# Get fix latitude and longitude`

```
17              latitude  = self.gpsd.fix.latitude
18              longitude = self.gpsd.fix.longitude
19              # Kalman filtering for latitude
20              xhatlatbef = xhatlat # X(k-1) latitude
21              Platbef = Plat + Q #P(k-1)+Q latitudr
22
23              Klat = Platbef / (Platbef + Rlat) #K latitude
24              xhatlat  =  xhatlatbef  +  Klat  +  (latitude-
25               xhatlatbef) #xhat latitude update
26              Plat = (1-Klat)*Platbef #P latitude baru update
27              # Kalman filtering for longitude
28              xhatlongbef = xhatlong # X(k-1) longitude
29              Plongbef = Plong + Q #P(k-1)+Q longitude
30
31              Klong = Plongbef / (Plongbef + Rlong) #K longitude
32              xhatlong  =  xhatlongbef  +  Klong  +  (longitude-
33                xhatlongbef) #xhat latitude update
34              Plong  =  (1-Klong)*Plongbef  #P  latitude  baru
35               update
```

Dalam penelitian ini digunakan Kalman filter yang berfungsi sebagai estimator rata – rata error data GPS dengan membandingkan hasil pengukuran data waktu sebelumnya dan data pengukuran yang sedang berlangsung. Kalman filter menggunkan konsep dengan memprediksi (*Time Update)* untuk mendapatkan nilai estimasi dari waktu sebelumnya untuk nilai yang akan datang (*Measurement Update)* kemudian nilai yang sudah didapatkan akan menjadi *correct* dan diperbarui untuk digunakan kembali sebgai umppan balik nilai estimasi untuk *time update* dengan harapan akan mendapat nilai yang lebi akurat. Dalam proses prediksi data (*Time Upodate*) digunakan persamaan sebagai berikut.

$$Xhat = Xhat(k-1)$$

Dan

$$Pbef = P + Q$$

Keterangan :
Xhat = nilai awal data GPS (*latitude, Longitude*)
P*bef* = *covariance* sebelum di*update*
Q = estimmasi error

Persamaan diatas digunakan untuk menentukan nilai X dan P yang akan digunakan sebagai umpan balik dalam proses selanjutnya (*measurement update*). Dalam proses perhitungan diatas nilai sensor (*Xhat)* akan di kalikan dengan hasil *index matrix* (k) dikurangi 1. Serupa dengan *Xhat*, untuk mendapatkan nilai P juga akan diproses sama dengan penentuan nilai *Xhat*. Kemudian setelah prediksi nilai awal *Xhat* dan P didapat, nilai tersebuat akan digunakan sebagai umpan balik untuk mendapatkan nilai akhir hasil filter dengan melalui proses *measurement Update* dengan menggunakan persamaan sebagai berikut.

$$\mathrm{K} = \frac{\mathrm{P}bef}{(\mathrm{P}bef + \mathrm{R})}$$

$$\mathrm{X} = \mathrm{X}hat + \mathrm{K} + (lat\ or\ long - \mathrm{X}hatbef)$$

$$\mathrm{P} = (1 - \mathrm{K}lat\ or\ long) \times \mathrm{P}bef$$

Keterangan :
K = Kalman Gain
R = perkiraan *error*

Untuk mendapatkan hasil *measurement update* langkah pertama dengan menentukan nilai kalman gain. Kalman gain digunakan sebagai estimator data GPS yang akan menentukan nilai hasil prediksi. Untuk menentukan nilai kalman gain perlukan dua hal. Pertama digunakan nilai kovarian dari sensor sebagai kesalahan data asli *Phat* (*Phat*) yang digunakan sebgai informasi untuk memperbarui data berikutnya. Kedua diperlukan hasil kovarian P*hat + R* sebagia kesalahan dalam perkiraan. Dalam perhitungan kalman gain nilai data sensor (*Phat*) akan dibandingkan dengan hasil penjumlahan nilai prediksi dengan dengan estimasi kesalahan (*Phat + R)*. Nilai kalman gain yang didapat kemudian digunakan untuk menentukan nilai akhir yang akan diupdate menjadi input pada alat.