TROLI BELANJA YANG CERDAS

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Ahli Madya
Program Studi Mekatronika

Disusun oleh:

1. Nama: Achileus Arles Arkontardo
   NIM: 181113047
2. Nama: Robby Christiawan Edhy
   NIM: 181113052

PROGRAM STUDI MEKATRONIKA
FAKULTAS VOKASI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2021

SMART SHOPPING TROLLEY

FINAL PROJECT

Presented as partial fulfilment of the requirements
to obtain the *Ahli Madya* degree
in Mechatronic Engineering

By:

1. Name: Achileus Arles Arkontardo
   NIM: 181113047
2. Name: Robby Christiawan Edhy
   NIM: 181113052

MECHATRONICS STUDY PROGRAM
VOCATIONAL FACULTY
SANATA DHARMA UNIVERSITY
YOGYAKARTA
2021

HALAMAN PERSETUJUAN

TUGAS AKHIR

TROLI BELANJA YANG CERDAS

Disusun oleh:

1. Nama: Achileus Arles Arkontardo
NIM: 181113047
2. Nama: Robby Christiawan Edhy
NIM: 181113052

Telah disetujui oleh:

Pembimbing **Tanggal : 8 Juli 2021**

Sukma Meganova Effendi, S.T., M.Tr.T.

HALAMAN PENGESAHAN

TUGAS AKHIR

TROLI BELANJA YANG CERDAS

Dipersiapkan dan ditulis oleh:

1. Nama: Achileus Arles Arkontardo
NIM: 181113047
2. Nama: Robby Christiawan Edhy
NIM: 181113052

Telah dipertahankan di depan Dewan Penguji
pada tanggal 15 Juli 2021
dan dinyatakan memenuhi syarat

Susunan Dewan Penguji:

Ketua	**:**	**Pippie Arbiyanti, S.T., M. Eng.**	**……**
Sekretaris	**:**	**Dian Artanto, S.T., M.Eng.**	**……**
Anggota	**:**	**Sukma Meganova Effendi, S.T., M.Tr.T.**	**……**

Yogyakarta, 15 Juli 2021
Fakultas Vokasi
Universitas Sanata Dharma
Dekan

Eko Aris Budi Cahyono, S.T., M.Eng.

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka, sebagaimana layaknya karya ilmiah.

Yogyakarta, 18 September 2021

Penulis,

Achileus Arles Arkontardo

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama : Achileus Arles Arkontardo
Nomor Mahasiswa : 181113047

Demi pengembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul: **TROLI BELANJA YANG CERDAS / *SMART SHOPPING TROLLEY*** beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan kepada Perpustakaan Universitas Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain, mengelolanya dalam bentuk pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya maupun memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Atas kemajuan teknologi informasi, saya tidak berkeberatan jika nama, tanda tangan, gambar atau *image* yang ada di dalam karya ilmiah saya terindeks oleh mesin pencari (*search engine*), misalnya *google*.

Demikian pernyataan ini yang saya buat dengan sebenarnya.
Dibuat di Yogyakarta
Pada tanggal: 18 September 2021

(Achileus Arles Arkontardo)

ABSTRAK

Penggunaan teknologi otomasi saat ini semakin dibutuhkan di berbagai kegiatan, salah satunya dalam kegiatan berbelanja. Dalam berbelanja, troli menjadi salah satu alat yang dapat memudahkan para pembeli dalam membawa barang belanjaan. Penggunaan troli biasanya membutuhkan tenaga manusia sebagai tenaga penggerak. Hal ini memungkinkan troli menjadi sarana penyebaran virus dan bakteri. Selain itu, ruang gerak pengguna troli juga menjadi terbatas karena pengguna harus mendorong troli tersebut. Oleh karena itu, dibutuhkan troli otomatis yang dapat bergerak mengikuti penggunanya. Troli otomatis dirancang untuk dapat membaca arah pergerakan dan jarak pengguna dari troli. Pendeteksian dilakukan dengan pemanfaatan sensor GPS *Tracking* dan sensor Ultrasonik.

Berkaitan dengan hal tersebut, penulis merumuskan dua pertanyaan yang nantinya akan dibahas dan diselesaikan dalam tugas akhir ini. Pertanyaan yang pertama adalah bagaimana merancang dan membuat sistem pembacaan data GPS dan deteksi objek pada troli? Pertanyaan yang kedua adalah bagaimana merancang kendali Blynk yang dapat mengendalikan gerakan roda-roda troli sehingga dapat mengikuti pengguna?

Untuk menjawab pertanyaan-pertanyaan tersebut, penulis merancang mekanik proyek ini memanfaatkan bentuk dasar dari troli yang sudah ada di pasaran dengan menambahkan beberapa komponen. Komponen-komponen tersebut seperti kotak yang berisi rangkaian sistem kendali, elektrik dan sumber tegangan yang berupa aki motor. Penulis juga menambahkan sensor ultrasonik dan tombol *emergency* sebagai sistem pengaman bila terjadi keadaan yang tidak diinginkan. Troli ini diperkirakan mampu membawa beban hingga 15kg.

Sebagai kesimpulan, sistem pembacaan dan pengolahan data GPS berjalan dengan semestinya. Dengan memanfaatkan aplikasi Blynk sebagai pengendali troli, gerakan pada roda-roda troli yang terhubung dengan motor listrik juga dapat dikendalikan. Dengan kata lain, sistem troli yang dibuat dapat berjalan sesuai dengan yang direncanakan.

Kata kunci: troli, GPS, Blynk.

ABSTRACT

The use of automation technology is now increasingly needed in various activities, one of which is shopping activities. In shopping, a trolley is one of the tools that can facilitate buyers in carrying groceries. The use of a trolley usually requires humans to operate it. This allows the trolley to be a means of spreading viruses and bacteria. In addition, the move for trolley users is also limited because users have to push the trolley. Therefore, an automatic trolley is needed that can move following its users. The automatic trolley is designed to be able to read the direction of movement and distance of the user from the trolley. Detection is done by utilizing GPS Tracking sensors and Ultrasonic sensors.

In this regard, the authors formulate two questions that will be discussed and resolved in this final project. The first question is how to design and make a GPS data reading and object detection system on the trolley? The second question is how to design a Blynk control that can control the movement of the trolley wheels so that it can follow the user?

To answer these questions, the author designed the mechanics of this project utilizing the basic form of a trolley that already exists in the market by adding several components. These components include a box containing the control system circuit, electricity, and a voltage source in the form of a motor battery. The author also adds ultrasonic sensors and an emergency button as a safety system in case of unwanted circumstances. This trolley is estimated to be able to carry loads of up to 15kg.

In conclusion, the GPS data reading and processing system runs properly. By utilizing the Blynk application as the trolley controller, the movement of the trolley wheels connected to the electric motor can also be controlled. In other words, the trolley system made can run as planned.

Keywords: trolley, GPS, Blynk.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Teknologi Otomasi merupakan pemanfaatan penggunaan mesin, sistem kontrol, dan teknoligi informasi untuk optimasi produksi produk serta pengiriman barang dan jasa. Penggunaan teknologi otomasi semakin dibutuhkan dewasa ini di berbagai kegiatan, salah satunya adalah kegiatan berbelanja.

Troli merupakan alat bantu manusia untuk memindahkan barang dari satu tempat ke tempat yang lain. Penggunaan alat ini memudahkan manusia untuk membawa dan memindahkan barang dalam jumlah yang banyak. Troli pada umumnya membutuhkan tenaga manusia sebagai tenaga penggerak dan mengharuskan penggunanya mendorong troli tersebut. Hal ini memungkinkan troli menjadi sarana penyebaran virus dan bakteri akibat dari penggunaanya yang bergantian antara satu orang dengan orang yang lain, selain itu penggunaan troli yang mengharuskan manusia untuk mendorongnya membatasi penggunaan serta aktifitas tangan untuk melakukan kegiatan yang lain seperti menggendong anak, mengambil barang, ataupun menggunakan gadget. Oleh karena itu, dibutuhkan troli otomatis yang dapat bergerak mengikuti penggunanya.

Troli yang dibuat harus mampu untuk membaca arah pergerakan dan jarak pengguna dari troli. Pembacaan arah pergerakan dan jarak ini ditujukan agar troli dapat mengikuti pergerakan pengguna dan kecepatan gerak dapat di kontrol. Pendeteksian dilakukan dengan pemanfaatan sensor GPS *Tracking* dan sensor Ultrasonik.

Metode penggerak roda troli yang diaplikasikan pada topik kali ini adalah *Differential Drive Robot/Differential Wheeled Robot*. Metode ini menggunakan dua buah roda penggerak yang independen yang dihasilkan dari dua buah aktuator. Metode ini memanfaatkan kecepatan dan arah putaran roda yang berbeda untuk menggerakkan troli.

1.2 RUMUSAN MASALAH

Permasalahan yang akan diselesaikan dalam tugas akhir ini adalah:

1. Bagaimana merancang dan membuat sistem pembacaan data GPS dan deteksi objek pada troli?
2. Bagaimana merancang kendali Blynk yang dapat mengendalikan gerakan roda-roda troli sehingga dapat mengikuti pengguna?

1.3 BATASAN MASALAH

Adapun beberapa batasan masalah dari topik tugas akhir ini adalah:

1. Beban angkut maksimal troli adalah 15kg.
2. Pengguna harus mengunduh dan menginstall aplikasi Blynk.

1.4 REFERENSI RANCANGAN

Adapun beberapa referensi yang kami gunakan dalam pembuatan sistem ini adalah :

a. 1 https://youtu.be/7wA84azO-HM

b.

https://www.hackster.io/hackershack/make-an-autonomous-follow-me-cooler-7ca8bc

c.  https://youtu.be/PbP8IbX3c-A

1.5 SOLUSI TERPILIH

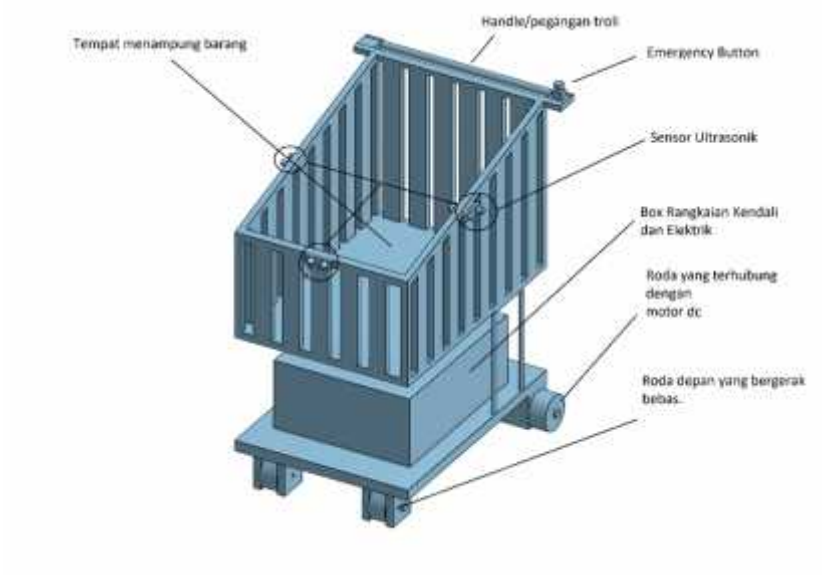
Berdasarkan referensi yang telah kami dapatkan, untuk pembuatan alat ini kami memilih penggabungan 2 referensi dimana bentuk dari sistem yang kami buat mengikuti dari referensi pertama dan ketiga dimana bentuk sistem memodifikasi bentuk troli yang sudah ada. Sedangkan untuk skema pengendalian dan pemrograman kami memilih referensi kedua.

BAB II
PERANCANGAN ALAT

2.1 DESKRIPSI ALAT

Troli belanja minimarket dengan kapasitas beban angkut maksimal sebesar 15kg, digunakan untuk pusat perbelanjaan dengan skala kecil hingga menengah. Troli digerakkan dengan sumber tegangan 24V yang dilengkapi sistem sensor untuk mendeteksi lokasi dan jarak troli dengan rak ataupun orang disamping agar troli tidak menabrak objek di dekatnya. Menggunakan sistem kendali bluetooth yang terkoneksi pada aplikasi Blynk di ponsel pintar sehingga troli terbebas dari sentuhan langsung dengan pengguna.

2.2 PERANCANGAN MEKANIK

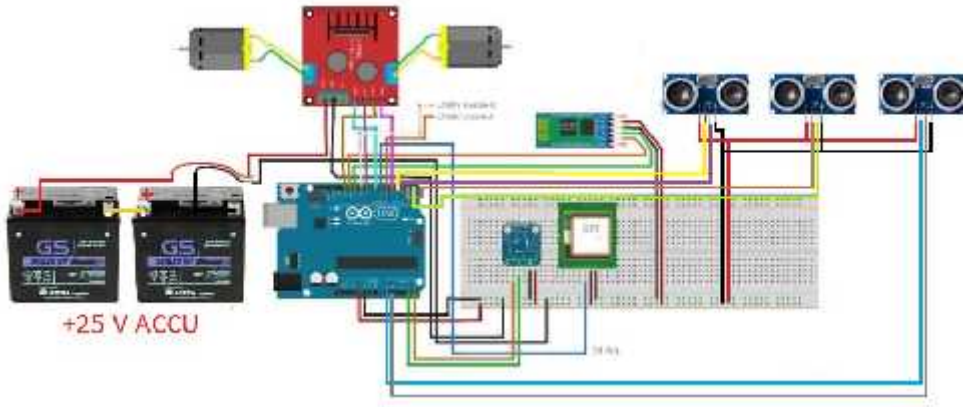


Gambar 2.1 Rancangan Troli Belanja yang Pintar

Dalam perancangan mekanik proyek ini memanfaatkan bentuk dasar dari troli yang sudah ada di pasaran, sehingga pada bagian mekanik tidak banyak perubahan yang dilakukan. Dengan memanfaatkan bentuk dasar dari troli yang sudah ada, penulis menambah beberapa komponen. Komponen-komponen tersebut seperti kotak yang berisi rangkaian sistem kendali, elektrik dan sumber tegangan yang berupa aki motor. Penulis juga menambahkan sensor ultrasonik

dan tombol *emergency* sebagai sistem pengaman bila terjadi keadaan yang tidak diinginkan. Dengan desain mekanik yang seperti di gambar diperkirakan mampu membawa beban hingga 15kg.

2.3 PERANCANGAN ELEKTRIK



Gambar 2.2 Wiring Sistem

Wiring Arduino → Driver Motor

Arduino	L298N
PIN 5	EN A
PIN 9	EN B
PIN 12	IN1
PIN 4	IN2
PIN 7	IN3
PIN 8	IN4
PIN 5V	VCC
PIN GND	GND

Wiring Arduino → GPS Module

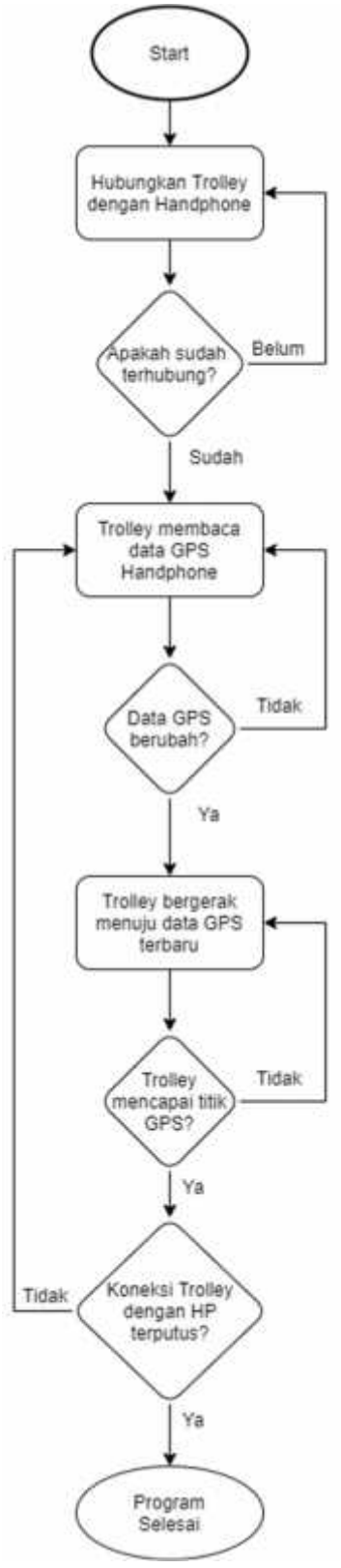
ARDUINO	UBLOX-NEO6M
PIN 6	TX
PIN +5V	VCC
PIN GND	GND

Wiring Arduino→Ultrasonik

ARDUINO	HC-SR04
PIN 11, 2, A0	TRIG
PIN 10, 3, A1	ECHO
PIN VCC	VCC
PIN GND	GND

Wiring Arduino→Module Bluetooth		Wiring Arduino → Module Kompas	
ARDUINO	HC-05	ARDUINO	HMC5833I
PIN 0	TX	PIN A5	SCL
PIN 1	RX	PIN A4	SDA
PIN 5V	VCC	PIN +5V	VCC
PIN GND	GND	PIN GND	GND

2.4 PERANCANGAN KENDALI



Gambar 2.3 Rancangan Kendali Troli

BAB III
HASIL ALAT

3.1 SPESIFIKASI ALAT

Ttroli barang otomatis berkapasitas 15kg

Panjang : 74cm

Lebar : 45cm

Tinggi : 84cm

Dilengkapi dengan 3 buah sensor ultrasonik yang bekerja pada tegangan 5V

Bekerja dengan tegangan sumber 24V

Terdapat tombol Emergency Stop yang dapat digunakan untuk menghentikan gerak troli apabila terjadi hal yang darurat.

Gambar 3.1 Hasil Alat

3.2 KOMPONEN-KOMPONEN ALAT

Daftar komponen mekanik:

No	Komponen Mekanik	Spesifikasi
1	Roda 	- Fungsi: sebagai roda penggerak - Diameter 100 mm - Jumlah 4 buah
2	Troli 	- Fungsi: sebagai kerangka dasar - Dimensi : - Jumlah 1 buah

Tabel 3.1 Daftar Komponen Mekanik

Daftar komponen elektrik:

No	Komponen Elektrik	Spesifikasi
1	Sensor ultrasonik 	- Fungsi: pendeteksi jarak - Tipe: HC-SR04 - Jumlah 3 buah
2	Motor DC 24 V 	- Fungsi: penggerak roda - Tipe: PG28 24Vdc 800 RPM - Jumlah 2 buah
3	Driver motor	- Tipe : L298N - Fungsi : Pengendali arah putaran motor

No	Komponen Elektrik	Spesifikasi
		- Jumlah 2 buah
4	Sensor Kompas 	- Tipe : HMC5833I - Jumlah 1 buah
5	Modul Bluetooth 	- Tipe : HC05 - Fungsi : sebagai jalur komunikasi antara Troli dengan aplikasi Blynk - Jumlah 1 buah
6	Arduino 	- Tipe : Arduino UNO - Fungsi : Sistem kendali pada troli - Jumlah 1 buah
7	GPS Modul 	- Tipe : U-Blox NEO-6M-V2 - Fungsi : membaca posisi Troli - Jumlah 1 buah

No	Komponen Elektrik	Spesifikasi
8	Kabel Jumper 	- Tipe : JST SH 2mm 3pin - Fungsi : Konektor Motor Dc dengan Driver motor - Jumlah 2 buah
9	Kabel Jumper 	- Tipe : Female to Female - Jumlah 10 buah - Tipe : Male to Female - Jumlah 10 buah - Tipe : Male to Male - Jumlah 10 buah
10	Aki Motor 	- Tipe : Aki 12Vdc - Fungsi : sebagai sumber tegangan - Jumlah 2 buah
11	Emergency Button 	- Fungsi : Pemutus aliran arus dan tegangan darurat - Jumkah 1 buah
12	Modul Regulator Step Down 	- Tipe : LM2596 - Fungsi : menurunkan tegangan dari 24V ke 12V dan 5V - Jumlah 3 buah

Tabel 3.2 Daftar Komponen Elektrik

3.3 CARA KERJA ALAT

Ketika Troli dihubungkan dengan *smartphone* dan tombol ON pada aplikasi ditekan, data lokasi GPS pada *smartphone* akan di baca. Kemudian data GPS tersebut dikirimkan ke troli melalui perantara *bluetooth*. Ketika data lokasi GPS telah diterima troli, data tersebut akan di bandingkan dengan data GPS yang dideteksi oleh sensor GPS dan kompas yang terdapat di troli. Apabila terdapat perbedaan data, maka mikrokontroler akan memberikan perintah berupa output untuk menggerakkan motor listrik menuju kearah data GPS yang terdeteksi di *smartphone*. Ketika data GPS *smartphone* telah tercapai maka troli akan berhenti dan program akan melakukan scanning kembali untuk melihat apakah data GPS *smartphone* berubah atau tetap sama.

Apabila saat troli memberikan perintah untuk menuju ke data GPS *smartphone*, sensor ultrasonik mendeteksi bahwa ada objek yang terdeteksi. Maka mikrokontroler akan memberikan perintah untuk menghentikan gerak troli untuk sementara(dalam hal ini troli berhenti sekitar 2 detik) kemudian mikrokontroler akan memberikan output motor listrik untuk bergerak menjauhi objek yang terdeteksi. Ketika sensor ultrasonik sebelah kiri mendeteksi objek, maka troli akan diberi perintah untuk bergerak ke kanan demikian juga untuk sensor ultrasonik sebelah kanan, troli akan diberi perintah untuk bergerak ke kiri. Untuk sensor ultrasonik bagian depan, apabila sensor tersebut mendeteksi adanya objek, troli akan diberi perintah untuk berhenti sesaat hingga objek yang terdeteksi menjauh atau sudah tidak terdeteksi lagi.

Untuk prosedur penggunaan dapat dilihat pada *manual book.*

BAB IV
PENUTUP

4.1 KESIMPULAN

Berdasarkan keseluruhan sistem yang berjalan pada troli kami, maka dapat disimpulkan bahwa:

1. Sistem pembacaan dan pengolahan data GPS berjalan dengan semestinya. Dengan memanfaatkan sensor GPS pada *smartphone*, data GPS diambil dan kemudian dkirim ke troli yang nantinya data tersebut akan dibandingkan dengan data yang dibaca sensor GPS dan kompas yang terdapat di troli.
2. Dengan memanfaatkan aplikasi Blynk sebagai pengendali troli, gerakan pada roda-roda troli yang terhubung dengan motor listrik dapat dikendalikan. Jika data GPS pada troli belum sesuai dengan data GPS pada *smartphone*, mikrokontroler troli akan memberi perintah output pada motor listrik untuk bergerak maju atau mundur sesuai lokasi tujuan yaitu lokasi data GPS *smartphone*.

Dengan demikian dapat diambil kesimpulan bahwa sistem troli yang dibuat dapat berjalan sesuai dengan yang direncanakan, akan tetapi sistem ini memiliki kelemahan yaitu troli hanya bisa digunakan diluar ruangan dimana pembacaan data GPS dapat maksimal.

4.2 PROSPEK PENGEMBANGAN ALAT

Berdasarkan sistem yang sudah jadi dan beberapa fitur yang sudah dimiliki alat ini, masih terdapat beberapa batasan masalah yang mana dapat diperbaiki sebagai bentuk pengembangan alat yang lebih baik lagi. Diantaranya adalah:

1. Penambahan beban maksimal yang mampu dibawa oleh troli.
2. Penambahan fitur baru seperti scan harga barang sehingga waktu sampai di kasir hanya tinggal bayar dan penambahan fitur bayar online.
3. Pemakaian sensor Vission dan Ultrasonik sebagai parameter pembacaan jarak dan lokasi.
4. Penggantian sumber tegangan dari aki ke baterai agar bisa dilakukan pengisian daya tanpa mengganti sumber tegangan nya.
5. Penggantian sistem kendali dari otomasi menggunakan GPS dan bluetooth menjadi sistem kendali joystick agar lebih interaktif terhadap pengguna.

DAFTAR PUSTAKA

https://www.blibli.com/p/emergency-stop-reset-putar-hanyoung/pc--MTA-8784650

https://www.astraotoshop.com/aki-gs-astra-mf-gtz-5s-untuk-motor-honda-supra-x-honda-vario-honda-beat-fi-yamaha-mio-yamaha-vixion.html

https://shopee.co.id/LM2596-DC-DC-Step-Down-Voltage-Regulator-LM-2596-Penurun-Tegangan-i.1937297.864052937

https://elediy.com/products/grove-universal-4-pin-to-beaglebone-blue-6-pin-female-jst-sh-conversion-cable-10-pcs-pack

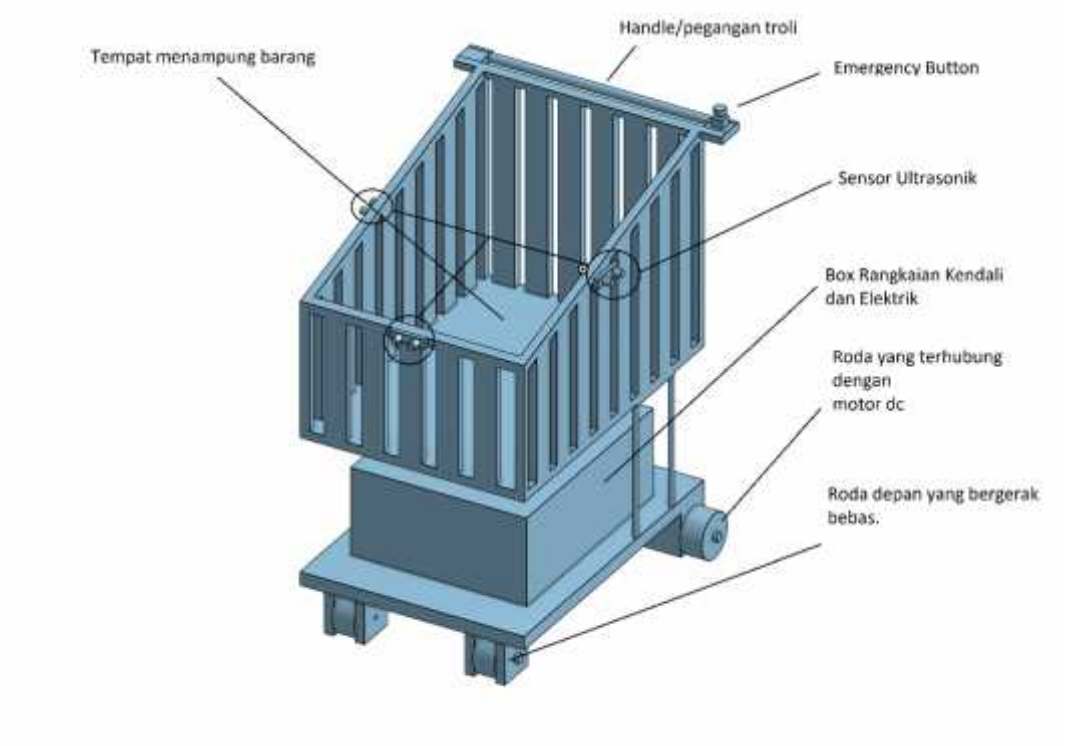
https://digiwarestore.com/id/jump-wire/kabel-jumper-female-to-female-dupont-10cm-331260.html

https://www.hackster.io/hackershack/make-an-autonomous-follow-me-cooler-7ca8bc

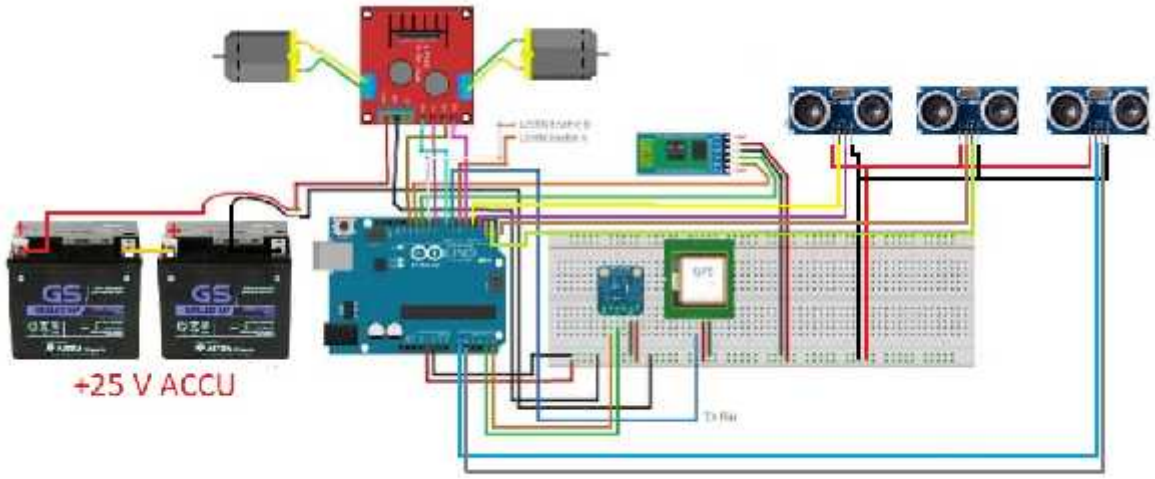
DAFTAR LAMPIRAN

1. Gambar rancangan mekanik alat
2. Gambar skema rangkaian elektronik
3. Program/ladder diagram
4. Data sheet komponen yang digunakan

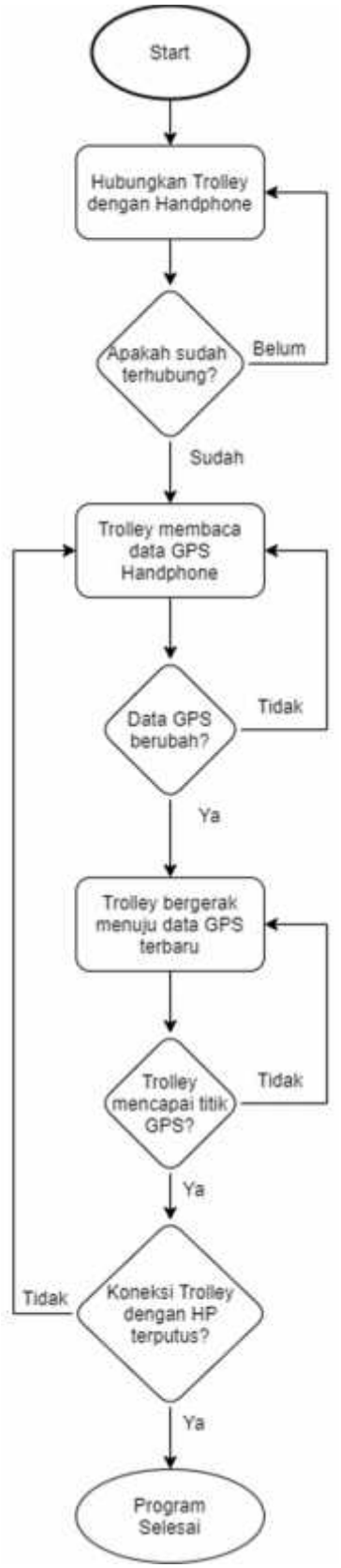
Lampiran 1 Rancangan Mekanik Alat



Lampiran 2 Gambar Skema Elektronik dan Elektrik



Lampiran 3 Program dan Ladder Diagram



Trolley_coba_1

```
#define BLYNK_USE_DIRECT_CONNECT

// Imports

#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_HMC5883_U.h>

#include <SoftwareSerial.h>

#include <BlynkSimpleSerialBLE.h>

#include "./TinyGPS.h"// Use local version of this library

#include "./PenginstallanPort.h"

// GPS

TinyGPS gps;

// Master Enable

bool enabled = false;

// Serial components

SoftwareSerial bluetoothSerial(BLUETOOTH_TX_PIN, BLUETOOTH_RX_PIN);

SoftwareSerial nss(GPS_TX_PIN, 255);          // TXD to digital pin 6

/* Compass */

Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);

GeoLoc checkGPS() {

 Serial.println("Reading onboard GPS: ");

 bool newdata = false;

 unsigned long start = millis();

 while (millis() - start < GPS_UPDATE_INTERVAL) {

  if (feedgps())
```

```
    newdata = true;
  }
  if (newdata) {
   return gpsdump(gps);
  }
  GeoLoc coolerLoc;
  coolerLoc.lat = 0.0;
  coolerLoc.lon = 0.0;
  return coolerLoc;
}
// Get and process GPS data
GeoLoc gpsdump(TinyGPS &gps) {
  float flat, flon;
  unsigned long age;

  gps.f_get_position(&flat, &flon, &age);
  GeoLoc coolerLoc;
  coolerLoc.lat = flat;
  coolerLoc.lon = flon;
  Serial.print(coolerLoc.lat, 7); Serial.print(", "); Serial.println(coolerLoc.lon, 7);
  return coolerLoc;
}
bool feedgps() {
  while (nss.available()) {
   if (gps.encode(nss.read()))
```

```
    return true;
  }
  return false;
}
// Killswitch Hook
BLYNK_WRITE(V1) {
  enabled = !enabled;
  //Stop the wheels
  stop();
}
// GPS Streaming
BLYNK_WRITE(V2) {
  GpsParam gps(param);
  Serial.println("Received remote GPS: ");
  Serial.print(gps.getLat(), 7); Serial.print(", "); Serial.println(gps.getLon(), 7);
  GeoLoc phoneLoc;
  phoneLoc.lat = gps.getLat();
  phoneLoc.lon = gps.getLon();
  driveTo(phoneLoc, GPS_STREAM_TIMEOUT);
}
void displayCompassDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
```

```
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" uT");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}
#ifndef DEGTORAD
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.295779513082320876f
#endif
float geoBearing(struct GeoLoc &a, struct GeoLoc &b) {
  float y = sin(b.lon-a.lon) * cos(b.lat);
  float x = cos(a.lat)*sin(b.lat) - sin(a.lat)*cos(b.lat)*cos(b.lon-a.lon);
  return atan2(y, x) * RADTODEG;
}
float geoDistance(struct GeoLoc &a, struct GeoLoc &b) {
  const float R = 6371000; // km
  float p1 = a.lat * DEGTORAD;
  float p2 = b.lat * DEGTORAD;
  float dp = (b.lat-a.lat) * DEGTORAD;
  float dl = (b.lon-a.lon) * DEGTORAD;
```

```
  float x = sin(dp/2) * sin(dp/2) + cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
  float y = 2 * atan2(sqrt(x), sqrt(1-x));
  return R * y;
}
float geoHeading() {
  sensors_event_t event;
  mag.getEvent(&event);
  float heading = atan2(event.magnetic.y, event.magnetic.x);
  // Offset
  heading -= DECLINATION_ANGLE;
  heading -= COMPASS_OFFSET;
  if(heading < 0)
    heading += 2*PI;
  if(heading > 2*PI)
    heading -= 2*PI;
  // Convert radians to degrees for readability.
  float headingDegrees = heading * 180/M_PI;
  // Map to -180 - 180
  while (headingDegrees < -180) headingDegrees += 360;
  while (headingDegrees >  180) headingDegrees -= 360;
  return headingDegrees;
}
void setSpeedMotorA(int speed) {
  digitalWrite(MOTOR_A_IN_1_PIN, LOW);
  digitalWrite(MOTOR_A_IN_2_PIN, HIGH);
```

```
  analogWrite(MOTOR_A_EN_PIN, speed + MOTOR_A_OFFSET);
}
void setSpeedMotorB(int speed) {
  digitalWrite(MOTOR_B_IN_1_PIN, LOW);
  digitalWrite(MOTOR_B_IN_2_PIN, HIGH);
  analogWrite(MOTOR_B_EN_PIN, speed + MOTOR_B_OFFSET);
}
void setSpeed(int speed)
{
  // turn on motor A
  setSpeedMotorA(speed);
  // turn on motor B
  setSpeedMotorB(speed);
}
void stop() {
  // now turn off motors
  digitalWrite(MOTOR_A_IN_1_PIN, LOW);
  digitalWrite(MOTOR_A_IN_2_PIN, LOW);
  digitalWrite(MOTOR_B_IN_1_PIN, LOW);
  digitalWrite(MOTOR_B_IN_2_PIN, LOW);
}
void drive(int distance, float turn) {
  int fullSpeed = 230;
  int stopSpeed = 0;
  // drive to location
```

```
int s = fullSpeed;
if ( distance < 8 ) {
  int wouldBeSpeed = s - stopSpeed;
  wouldBeSpeed *= distance / 8.0f;
  s = stopSpeed + wouldBeSpeed;
}
int autoThrottle = constrain(s, stopSpeed, fullSpeed);
autoThrottle = 230;
float t = turn;
while (t < -180) t += 360;
while (t >  180) t -= 360;
Serial.print("turn: ");
Serial.println(t);
Serial.print("original: ");
Serial.println(turn);
float t_modifier = (180.0 - abs(t)) / 180.0;
float autoSteerA = 1;
float autoSteerB = 1;
if (t < 0) {
  autoSteerB = t_modifier;
} else if (t > 0){
  autoSteerA = t_modifier;
}
Serial.print("steerA: "); Serial.println(autoSteerA);
Serial.print("steerB: "); Serial.println(autoSteerB);
```

```
  int speedA = (int) (((float) autoThrottle) * autoSteerA);
  int speedB = (int) (((float) autoThrottle) * autoSteerB);
  setSpeedMotorA(speedA);
  setSpeedMotorB(speedB);
}
void driveTo(struct GeoLoc &loc, int timeout) {
  nss.listen();
  GeoLoc coolerLoc = checkGPS();
  bluetoothSerial.listen();
  if (coolerLoc.lat != 0 && coolerLoc.lon != 0 && enabled) {
    float d = 0;
    //Start move loop here
    do {
      nss.listen();
      coolerLoc = checkGPS();
      bluetoothSerial.listen();
      d = geoDistance(coolerLoc, loc);
      float t = geoBearing(coolerLoc, loc) - geoHeading();
      Serial.print("Distance: ");
      Serial.println(geoDistance(coolerLoc, loc));
      Serial.print("Bearing: ");
      Serial.println(geoBearing(coolerLoc, loc));
      Serial.print("heading: ");
      Serial.println(geoHeading());
      drive(d, t);
```

```
    timeout -= 1;
   } while (d > 3.0 && enabled && timeout>0);
   stop();
  }
}
void setupCompass() {
 if(!mag.begin())
 {
  Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
  while(1);
 }
 displayCompassDetails();
}
void setup()
{
 // Compass
 setupCompass();
 // Motor pins
 pinMode(MOTOR_A_EN_PIN, OUTPUT);
 pinMode(MOTOR_B_EN_PIN, OUTPUT);
 pinMode(MOTOR_A_IN_1_PIN, OUTPUT);
 pinMode(MOTOR_A_IN_2_PIN, OUTPUT);
 pinMode(MOTOR_B_IN_1_PIN, OUTPUT);
 pinMode(MOTOR_B_IN_2_PIN, OUTPUT);
 pinMode(LED_BUILTIN, OUTPUT);
```

```
  digitalWrite(LED_BUILTIN, HIGH);
  //Debugging via serial
  Serial.begin(4800);
  //GPS
  nss.begin(9600);
  //Bluetooth
  bluetoothSerial.begin(9600);
  Blynk.begin(bluetoothSerial, auth);
}
// Testing
void testDriveNorth() {
  float heading = geoHeading();
  int testDist = 10;
  Serial.println(heading);

  while(!(heading < 5 && heading > -5)) {
    drive(testDist, heading);
    heading = geoHeading();
    Serial.println(heading);
    delay(500);
  }
  stop();
}
void loop()
{
```

```
Blynk.run();}
```

PenginstallanPort.h

```
// Blynk Auth

char auth[] = "QbQevz0RvaLDW-rIHsH1AFfy0VsCBS6K";

// Pin variables

#define GPS_TX_PIN 6

#define BLUETOOTH_TX_PIN 10

#define BLUETOOTH_RX_PIN 11

#define MOTOR_A_EN_PIN 5

#define MOTOR_B_EN_PIN 9

#define MOTOR_A_IN_1_PIN 7

#define MOTOR_A_IN_2_PIN 8

#define MOTOR_B_IN_1_PIN 12

#define MOTOR_B_IN_2_PIN 4

// If one motor tends to spin faster than the other, add offset

#define MOTOR_A_OFFSET 20

#define MOTOR_B_OFFSET 0

#define DECLINATION_ANGLE 0.1222f

#define COMPASS_OFFSET 0.0f

#define GPS_UPDATE_INTERVAL 1000

#define GPS_STREAM_TIMEOUT 18

#define GPS_WAYPOINT_TIMEOUT 45

struct GeoLoc {

 float lat;

 float lon;

};
```

TinyGPS.cpp

```
#include "Arduino.h"

#include "TinyGPS.h"

#define _GPRMC_TERM   "GPRMC"

#define _GPGGA_TERM   "GPGGA"

TinyGPS::TinyGPS()

:  _time(GPS_INVALID_TIME)

,  _date(GPS_INVALID_DATE)

,  _latitude(GPS_INVALID_ANGLE)

,  _longitude(GPS_INVALID_ANGLE)

,  _altitude(GPS_INVALID_ALTITUDE)

,  _speed(GPS_INVALID_SPEED)

,  _course(GPS_INVALID_ANGLE)

,  _last_time_fix(GPS_INVALID_FIX_TIME)

,  _last_position_fix(GPS_INVALID_FIX_TIME)

,  _parity(0)

,  _is_checksum_term(false)

,  _sentence_type(_GPS_SENTENCE_OTHER)

,  _term_number(0)

,  _term_offset(0)

,  _gps_data_good(false)

#ifndef _GPS_NO_STATS

,  _encoded_characters(0)

,  _good_sentences(0)
```

```
, _failed_checksum(0)
#endif
{
 _term[0] = '\0';
}
bool TinyGPS::encode(char c)
{
 bool valid_sentence = false;

 ++_encoded_characters;
 switch(c)
 {
 case ',': // term terminators
  _parity ^= c;
 case '\r':
 case '\n':
 case '*':
  if (_term_offset < sizeof(_term))
  {
   _term[_term_offset] = 0;
   valid_sentence = term_complete();
  }
  ++_term_number;
  _term_offset = 0;
  _is_checksum_term = c == '*';
```

```
    return valid_sentence;
  case '$': // sentence begin
    _term_number = _term_offset = 0;
    _parity = 0;
    _sentence_type = _GPS_SENTENCE_OTHER;
    _is_checksum_term = false;
    _gps_data_good = false;
    return valid_sentence;
  }
  // ordinary characters
  if (_term_offset < sizeof(_term) - 1)
    _term[_term_offset++] = c;
  if (!_is_checksum_term)
    _parity ^= c;
  return valid_sentence;
}

#ifndef _GPS_NO_STATS
void TinyGPS::stats(unsigned long *chars, unsigned short *sentences, unsigned short
*failed_cs)
{
  if (chars) *chars = _encoded_characters;
  if (sentences) *sentences = _good_sentences;
  if (failed_cs) *failed_cs = _failed_checksum;
}
#endif

int TinyGPS::from_hex(char a)
```

```
{
  if (a >= 'A' && a <= 'F')
    return a - 'A' + 10;
  else if (a >= 'a' && a <= 'f')
    return a - 'a' + 10;
  else
    return a - '0';
}
unsigned long TinyGPS::parse_decimal()
{
  char *p = _term;
  bool isneg = *p == '-';
  if (isneg) ++p;
  unsigned long ret = 100UL * gpsatol(p);
  while (gpsisdigit(*p)) ++p;
  if (*p == '.')
  {
    if (gpsisdigit(p[1]))
    {
      ret += 10 * (p[1] - '0');
      if (gpsisdigit(p[2]))
        ret += p[2] - '0';
    }
  }
  return isneg ? -ret : ret;
```

```
}
unsigned long TinyGPS::parse_degrees()
{
  char *p;
  unsigned long left = gpsatol(_term);
  unsigned long tenk_minutes = (left % 100UL) * 10000UL;
  for (p=_term; gpsisdigit(*p); ++p);
  if (*p == '.')
  {
    unsigned long mult = 1000;
    while (gpsisdigit(*++p))
    {
      tenk_minutes += mult * (*p - '0');
      mult /= 10;
    }
  }
  return (left / 100) * 100000 + tenk_minutes / 6;
}
bool TinyGPS::term_complete()
{
  if (_is_checksum_term)
  {
    byte checksum = 16 * from_hex(_term[0]) + from_hex(_term[1]);
    if (checksum == _parity)
    {
```

```
    if (_gps_data_good)
    {
#ifndef _GPS_NO_STATS
      ++_good_sentences;
#endif
      _last_time_fix = _new_time_fix;
      _last_position_fix = _new_position_fix;

      switch(_sentence_type)
      {
      case _GPS_SENTENCE_GPRMC:
        _time     = _new_time;
        _date     = _new_date;
        _latitude  = _new_latitude;
        _longitude = _new_longitude;
        _speed    = _new_speed;
        _course   = _new_course;
        break;
      case _GPS_SENTENCE_GPGGA:
        _altitude  = _new_altitude;
        _time     = _new_time;
        _latitude  = _new_latitude;
        _longitude = _new_longitude;
        break;
      }
```

```
      return true;
    }
   }
#ifndef _GPS_NO_STATS
   else
    ++_failed_checksum;
#endif
   return false;
 }
 // the first term determines the sentence type
 if (_term_number == 0)
 {
  if (!gpsstrcmp(_term, _GPRMC_TERM))
    _sentence_type = _GPS_SENTENCE_GPRMC;
  else if (!gpsstrcmp(_term, _GPGGA_TERM))
    _sentence_type = _GPS_SENTENCE_GPGGA;
  else
    _sentence_type = _GPS_SENTENCE_OTHER;
  return false;
 }

 if (_sentence_type != _GPS_SENTENCE_OTHER && _term[0])
 switch((_sentence_type == _GPS_SENTENCE_GPGGA ? 200 : 100) + _term_number)
 {
```

```
case 101: // Time in both sentences
case 201:
 _new_time = parse_decimal();
 _new_time_fix = millis();
 break;
case 102: // GPRMC validity
 _gps_data_good = _term[0] == 'A';
 break;
case 103: // Latitude
case 202:
 _new_latitude = parse_degrees();
 _new_position_fix = millis();
 break;
case 104: // N/S
case 203:
 if (_term[0] == 'S')
  _new_latitude = -_new_latitude;
 break;
case 105: // Longitude
case 204:
 _new_longitude = parse_degrees();
 break;
case 106: // E/W
case 205:
 if (_term[0] == 'W')
```

```
      _new_longitude = -_new_longitude;
    break;
  case 107: // Speed (GPRMC)
    _new_speed = parse_decimal();
    break;
  case 108: // Course (GPRMC)
    _new_course = parse_decimal();
    break;
  case 109: // Date (GPRMC)
    _new_date = gpsatol(_term);
    break;
  case 206: // Fix data (GPGGA)
    _gps_data_good = _term[0] > '0';
    break;
  case 209: // Altitude (GPGGA)
    _new_altitude = parse_decimal();
    break;
 }

 return false;
}

long TinyGPS::gpsatol(const char *str)
{
 long ret = 0;
```

```
  while (gpsisdigit(*str))
    ret = 10 * ret + *str++ - '0';
  return ret;
}

int TinyGPS::gpsstrcmp(const char *str1, const char *str2)
{
  while (*str1 && *str1 == *str2)
    ++str1, ++str2;
  return *str1;
}
/* static */
float TinyGPS::distance_between (float lat1, float long1, float lat2, float long2)
{
  float delta = radians(long1-long2);
  float sdlong = sin(delta);
  float cdlong = cos(delta);
  lat1 = radians(lat1);
  lat2 = radians(lat2);
  float slat1 = sin(lat1);
  float clat1 = cos(lat1);
  float slat2 = sin(lat2);
  float clat2 = cos(lat2);
  delta = (clat1 * slat2) - (slat1 * clat2 * cdlong);
  delta = sq(delta);
```

```
  delta += sq(clat2 * sdlong);
  delta = sqrt(delta);
  float denom = (slat1 * slat2) + (clat1 * clat2 * cdlong);
  delta = atan2(delta, denom);
  return delta * 6372795;
}
```

TinyGPS.h

```
#ifndef TinyGPS_h

#define TinyGPS_h

#include "Arduino.h"

#define _GPS_VERSION 10 // software version of this library

#define _GPS_MPH_PER_KNOT 1.15077945

#define _GPS_MPS_PER_KNOT 0.51444444

#define _GPS_KMPH_PER_KNOT 1.852

#define _GPS_MILES_PER_METER 0.00062137112

#define _GPS_KM_PER_METER 0.001

class TinyGPS

{

  public:

    TinyGPS();

    bool encode(char c); // process one character received from GPS

    TinyGPS &operator << (char c) {encode(c); return *this;}

    inline void get_position(long *latitude, long *longitude, unsigned long *fix_age = 0)

    {

      if (latitude) *latitude = _latitude;

      if (longitude) *longitude = _longitude;

      if (fix_age) *fix_age = _last_position_fix == GPS_INVALID_FIX_TIME ?

        GPS_INVALID_AGE : millis() - _last_position_fix;

    }

    inline void get_datetime(unsigned long *date, unsigned long *time, unsigned long
*fix_age = 0)

    {
```

```
    if (date) *date = _date;
    if (time) *time = _time;
    if (fix_age) *fix_age = _last_time_fix == GPS_INVALID_FIX_TIME ?
      GPS_INVALID_AGE : millis() - _last_time_fix;
  }
  inline long altitude() { return _altitude; }
  inline unsigned long course() { return _course; }
  unsigned long speed() { return _speed; }
#ifndef _GPS_NO_STATS
  void stats(unsigned long *chars, unsigned short *good_sentences, unsigned short
*failed_cs);
#endif
  inline void f_get_position(float *latitude, float *longitude, unsigned long *fix_age = 0)
  {
    long lat, lon;
    get_position(&lat, &lon, fix_age);
    *latitude = lat / 100000.0;
    *longitude = lon / 100000.0;
  }
  inline void crack_datetime(int *year, byte *month, byte *day,
    byte *hour, byte *minute, byte *second, byte *hundredths = 0, unsigned long
*fix_age = 0)
  {
    unsigned long date, time;
    get_datetime(&date, &time, fix_age);
    if (year)
```

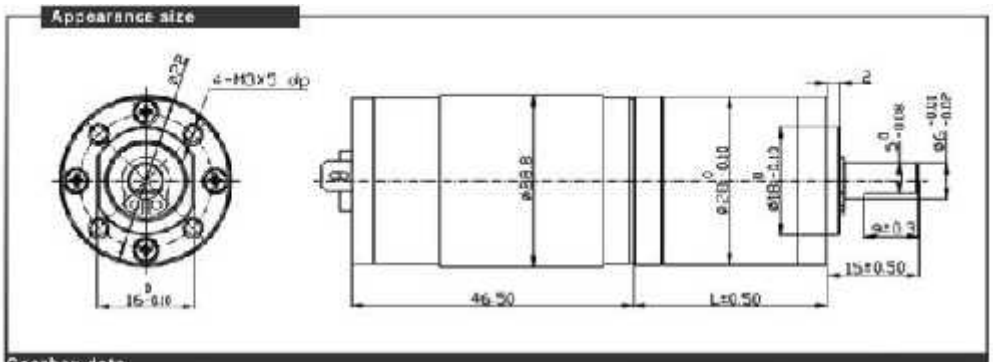
```
    {
      *year = date % 100;
      *year += *year > 80 ? 1900 : 2000;
    }
    if (month) *month = (date / 100) % 100;
    if (day) *day = date / 10000;
    if (hour) *hour = time / 1000000;
    if (minute) *minute = (time / 10000) % 100;
    if (second) *second = (time / 100) % 100;
    if (hundredths) *hundredths = time % 100;
  }
  inline float f_altitude()    { return altitude() / 100.0; }
  inline float f_course()      { return course() / 100.0; }
  inline float f_speed_knots() { return speed() / 100.0; }
  inline float f_speed_mph()   { return _GPS_MPH_PER_KNOT * f_speed_knots(); }
  inline float f_speed_mps()   { return _GPS_MPS_PER_KNOT * f_speed_knots(); }
  inline float f_speed_kmph()  { return _GPS_KMPH_PER_KNOT * f_speed_knots(); }
  static int library_version() { return _GPS_VERSION; }
  enum {GPS_INVALID_AGE = 0xFFFFFFFF, GPS_INVALID_ANGLE = 999999999,
GPS_INVALID_ALTITUDE = 999999999, GPS_INVALID_DATE = 0,
    GPS_INVALID_TIME = 0xFFFFFFFF, GPS_INVALID_SPEED = 999999999,
GPS_INVALID_FIX_TIME = 0xFFFFFFFF};
  static float distance_between (float lat1, float long1, float lat2, float long2);
private:
  enum {_GPS_SENTENCE_GPGGA, _GPS_SENTENCE_GPRMC,
_GPS_SENTENCE_OTHER};
```

```
    unsigned long _time, _new_time;
    unsigned long _date, _new_date;
    long _latitude, _new_latitude;
    long _longitude, _new_longitude;
    long _altitude, _new_altitude;
    unsigned long  _speed, _new_speed;
    unsigned long  _course, _new_course;
    unsigned long _last_time_fix, _new_time_fix;
    unsigned long _last_position_fix, _new_position_fix;
    byte _parity;
    bool _is_checksum_term;
    char _term[15];
    byte _sentence_type;
    byte _term_number;
    byte _term_offset;
    bool _gps_data_good;
#ifndef _GPS_NO_STATS
    unsigned long _encoded_characters;
    unsigned short _good_sentences;
    unsigned short _failed_checksum;
    unsigned short _passed_checksum;
#endif
    int from_hex(char a);
    unsigned long parse_decimal();
    unsigned long parse_degrees();
```

```
    bool term_complete();
    bool gpsisdigit(char c) { return c >= '0' && c <= '9'; }
    long gpsatol(const char *str);
    int gpsstrcmp(const char *str1, const char *str2);
};
#undef int
#undef char
#undef long
#undef byte
#undef float
#undef abs
#undef round
#endif
```

Lampiran 4 Datasheet Komponen

Motor DC PG28



Gearbox data

Number of stages	1 stage reduction	2 stage reduction	3 stage reduction	4 stage reduction	5 stage reduction
Reduction ratio	4 4.75	16 19 22.5	64, 76, 90, 107,	256, 304, 361, 428, 509,	1024, 1216, 14 , 1715, 2036, 2418,
Gearbox length L mm	22	27.1	32.2	37.3	42.4
Max. Running torque	2.0Kg f • cm	3.0Kg f • cm	4.0Kg f • cm	6.0Kg f • cm	10Kgf • cm
Max. Gear breaking torque	6.0Kg f • cm	9.0Kg f • cm	12Kgf • cm	18Kgf • cm	30Kgf • cm
Gearing efficiency	90%	81%	73%	65%	59%

HC-SR04

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound(340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

HMC5833I

HMC5883L

Characteristics	Conditions*	Min	Typ	Max	Units
Reflow Classification	MSL 3, 260 °C Peak Temperature				
Package Size	Length and Width	2.85	3.00	3.15	mm
Package Height Package Weight		0.8	0.9 18	1.0	mm mg

Absolute Maximum Ratings (* Tested at 25°C except stated otherwise.)

Characteristics	Min	Max	Units
Supply Voltage VDD	-0.3	4.8	Volts
Supply Voltage VDDIO	0.3	4.8	Volts

PIN CONFIGURATIONS

Pin	Name	Description
1	SCL	Serial Clock – I²C Master/Slave Clock
2	VDD	Power Supply (2.16V to 3.6V)
3	NC	Not to be Connected
4	S1	Tie to VDDIO
5	NC	Not to be Connected
6	NC	Not to be Connected
7	NC	Not to be Connected
8	SETP	Set/Reset Strap Positive – S/R Capacitor (C2) Connection
9	GND	Supply Ground
10	C1	Reservoir Capacitor (C1) Connection
11	GND	Supply Ground
12	SETC	S/R Capacitor (C2) Connection – Driver Side
13	VDDIO	IO Power Supply (1.71V to VDD)
14	NC	Not to be Connected
15	DRDY	Data Ready, Interrupt Pin. Internally pulled high. Optional connection. Low for 250 usec when data is placed in the data output registers.
16	SDA	Serial Data – I²C Master/Slave Data

Table 1: Pin Configurations

HC-05

HC-05

Bluetooth to Serial Port Module

Overview

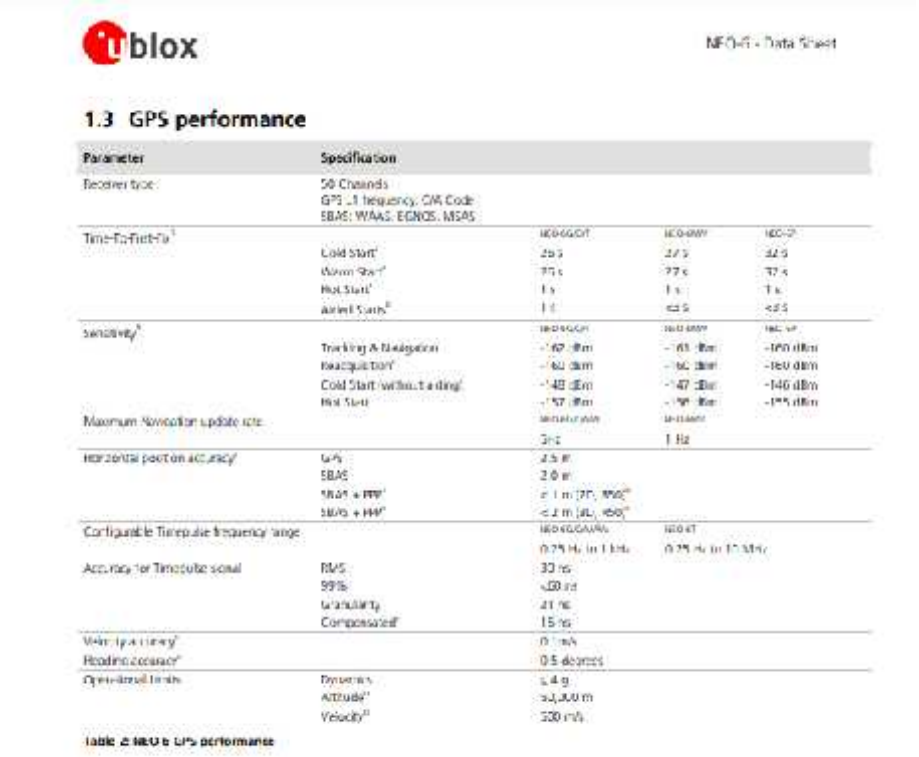
HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation, 1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

U-Blox NEO-6M-V2



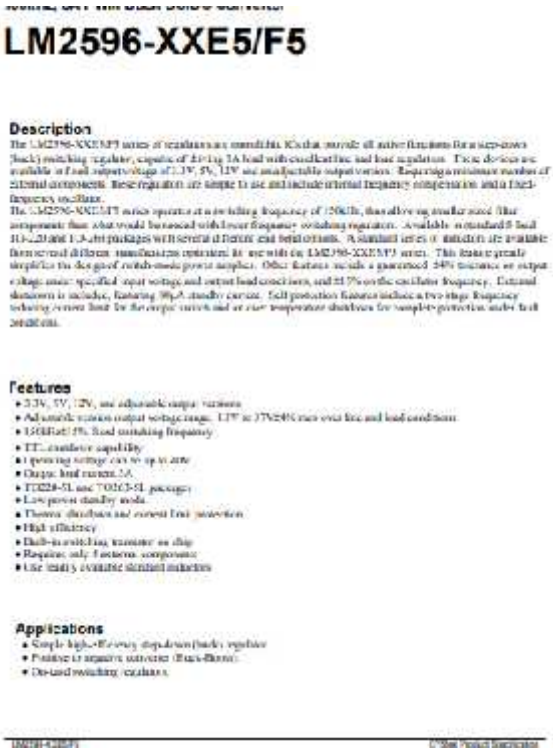
u-blox NEO-6 - Data Sheet

1.3 GPS performance

Parameter	Specification		
Receiver type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS		
Time-To-First-Fix[1]		NEO-6Q/7Q/8Q	NEO-6M/7M	NEO-6T
	Cold Start[2]	29 s	27 s	32 s
	Warm Start[2]	28 s	28 s	28 s
	Hot Start[2]	1 s	1 s	1 s
	Aided Starts[3]	1 s	<3 s	<3 s
Sensitivity[4]		NEO-6Q/7Q/8Q	NEO-6M/7M	NEO-6T
	Tracking & Navigation	-161 dBm	-161 dBm	-160 dBm
	Reacquisition[5]	-160 dBm	-160 dBm	-160 dBm
	Cold Start (without aiding)[6]	-148 dBm	-147 dBm	-146 dBm
	Hot Start	-157 dBm	-156 dBm	-155 dBm
Maximum Navigation update rate		NEO-6Q/6M/7Q/7M/8Q	NEO-6T	
		5 Hz	1 Hz	
Horizontal position accuracy[7]	GPS	2.5 m		
	SBAS	2.0 m		
	SBAS + PPP[8]	< 1 m (2D, 95%)[9]		
	SBAS + PPP[8]	< 2 m (3D, 95%)[9]		
Configurable Timepulse frequency range		NEO-6Q/6M/7Q/7M/8Q	NEO-6T	
		0.25 Hz to 1 kHz	0.25 Hz to 10 MHz	
Accuracy for Timepulse signal	RMS	30 ns		
	99%	<60 ns		
	Granularity	21 ns		
	Compensated[10]	15 ns		
Velocity[7]		0 m/s		
Heading accuracy[7]		0.5 degrees		
Operational limits	Dynamics	≤ 4 g		
	Altitude[11]	50,000 m		
	Velocity[11]	500 m/s		

Table 2: NEO-6 GPS performance

LM2596



LM2596-XXE5/F5

Description

The LM2596-XXE5/F5 series of regulators are monolithic IC's that provide all the active functions for a step-down (buck) switching regulator, capable of driving 3A load with excellent line and load regulation. These devices are available in fixed output voltage of 3.3V, 5V, 12V and an adjustable output version. Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation and a fixed-frequency oscillator.

The LM2596-XXE5/F5 series operates at a switching frequency of 150kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators. Available in standard 5 lead TO-220 and TO-263 packages with several different lead bend options, and a standard series of inductors are available from several different manufacturers optimized for use with the LM2596-XXE5/F5 series. This feature greatly simplifies the design of switch-mode power supplies. Other features include a guaranteed ±4% tolerance on output voltage under specified input voltage and output load conditions, and ±15% on the oscillator frequency. External shutdown is included, featuring 80μA standby current. Self-protection features include a two stage frequency reducing current limit for the output switch and an over temperature shutdown for complete protection under fault conditions.

Features

- 3.3V, 5V, 12V, and adjustable output versions
- Adjustable version output voltage range, 1.23V to 37V±4% max over line and load conditions
- 150kHz±15% fixed switching frequency
- TTL shutdown capability
- Operating voltage can be up to 40V
- Output load current 3A
- TO220-5L and TO263-5L packages
- Low power standby mode
- Thermal shutdown and current limit protection
- High efficiency
- Built-in switching transistor on chip
- Requires only 4 external components
- Use readily available standard inductors

Applications

- Simple high-efficiency step-down (buck) regulator
- Positive to negative converter (Buck-Boost)
- Distributed switching regulators