PROYEK AKHIR – VE 180626

RANCANG BANGUN *QUADCOPTER* MENGGUNAKAN STM32F1 GUNA PELACAKAN KORBAN BENCANA ALAM DENGAN GPS

Ilham Dwiki Ramadhan
NRP. 10311600000047

Dosen Pembimbing
Ir. Joko Susila, MT.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2020

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

FINAL PROJECT – VE 180626

QUADCOPTER DESIGN USING STM32F1 TO TRACKING NATURAL DISASTER VICTIMS WITH GPS

Ilham Dwiki Ramadhan
NRP. 10311600000047

Supervisor
Ir. Joko Susila, MT.

DEPARTEMENT OF ELECTRICAL AUTOMATION ENGINEERING
Vocations Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2020

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

PERNYATAAN KEASLIAN PROYEK AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Proyek Akhir saya dengan judul **"Rancang Bangun Quadcopter Menggunakan STM32F1 Guna Pelacakan Korban Bencana Alam dengan GPS"** adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya orang lain.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 20 Januari 2020

Ilham Dwiki Ramadhan
NRP. 10311600000047

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

RANCANG BANGUN QUADCOPTER MENGGUNAKAN STM32F1 GUNA PELACAKAN KORBAN BENCANA ALAM DENGAN GPS

LEMBAR PENGESAHAN
PROYEK AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Ahli Madya
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing

Ir. Joko Susila, M.T.
NIP. 19660606199102001

SURABAYA
JANUARI, 2020

RANCANG BANGUN QUADCOPTER MENGGUNAKAN STM32F1 GUNA PELACAKAN KORBAN BENCANA ALAM DENGAN GPS

Nama : Ilham Dwiki Ramadhan
Pembimbing : Ir. Joko Susila, MT.

ABSTRAK

Semakin sering terjadi bencana alam pada saat ini diperlukan kesiagaan dari tim penyelamat untuk bekerja seefisien mungkin dalam menemukan korban yang masih hidup, terutama korban yang berada di daerah yang susah dijangkau oleh manusia. Agar korban yang masih hidup mudah ditemukan dan dievakuasi maka dibuatlah alat yang berguna untuk memudahkan pencarian korban tersebut seperti quadcopter. *Quadcopter* adalah robot penjelajah udara *Unmanned Aerial Vehicle (UAV)* yang termasuk kategori *UAV micro* dan banyak digunakan oleh beberapa lembaga atau instansi. Robot *quadcopter* merupakan *UAV* yang memiliki ciri khusus yang mudah dikenali yaitu memiliki empat buah baling-baling motor yang digunakan sebagai penggeraknya.

Pada proyek akhir ini *quadcopter* menggunakan STM32F1 sebagai flight controller dan dilengkapi dengan GPS, barometer dan gyro. Cara kerja alat ini yaitu dengan memasang kamera external pada *quadcopter* yang terhubung dengan PC atau laptop untuk pemantauan visual secara real time , ketika pengguna berhasil menangkap visual korban, pengguna dapat melakukan *tracking* lokasi korban dengan data berfungsi untuk menstabilkan quadcopter ketika terbang , sedangkan barometer berfungsi untuk mengukur ketinggian terbang quadcopter.

STM32F1 dapat digunakan sebagai flight controller atau pusat kendali yang dapat mengoperasikan GPS, Gyro dan barometer. STM32F1 diprogram menggunakan *software* Arduino IDE.Modul GPS ublox M8N yang dipakai dapat membaca data latitude dan longtitude. GPS Ublox M8N juga dilengkapi dengan kompas HMC5883L.
Kata kunci : STM32F1, *Quadcopter, GPS*

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

QUADCOPTER DESIGN USING STM32F1 TO TRACKING NATURAL DISASTER VICTIMS WITH GPS

Name **: Ilham Dwiki Ramadhan**
Supervisor **: Ir. Joko Susila, MT.**

ABSTRACT

More frequent natural disasters at this time are required preparedness of the rescue team to work as efficiently as possible in finding survivors, especially victims who are in areas that are difficult to reach by humans. So that survivors are easily found and evacuated, then a tool is made that is useful to facilitate the search for victims such as quadcopter. Quadcopter is an Unmanned Aerial Vehicle (UAV) aerial robot that belongs to the UAV micro category and is widely used by several agencies or agencies. Quadcopter robot is a UAV that has a special characteristic that is easily recognized, which has four motor propellers that are used as its driving force.

In this final project the quadcopter uses STM32F1 as a flight controller and is equipped with GPS, barometer and gyro. The way this tool works is by installing an external camera on a quadcopter connected to a PC or laptop for visual monitoring in real time, when the user successfully captures the victim's visuals, the user can track the location of the victim with data functioning to stabilize the quadcopter when flying, while the barometer functions to measure the height of a quadcopter fly.

STM32F1 can be used as a flight controller or control center that can operate GPS, Gyro and barometer. STM32F1 is programmed using Arduino IDE software. The ublox M8N GPS module used can read latitude and longtitude data Ublox M8N also built in HMC5883L compass.

Keywords: Quadcopter, STM32F1, GPS data

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

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang selalu memberikan rahmat dan hidayah-Nya sehingga Proyek Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam semoga selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Proyek Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma 3 di Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember Surabaya dengan judul:

RANCANG BANGUN QUADCOPTER MENGGUNAKAN STM32F1 GUNA PELACAKAN KORBAN BENCANA ALAM DENGAN GPS

Dalam Proyek Akhir ini rancang bangun quadcopter menggunakan STM32F1 sebagai flight controllernya dan dilengkapi dengan GPS guna mendapat lokasi berdasarkan latitude dan longtitude,maka dari itu pengguna dapat terbantu dalam pelacakan para korban bencana alam.

Penulis mengucapkan terima kasih kepada Ibu dan Bapak penulis yang memberikan berbagai bentuk doa serta dukungan tulus tiada henti, Bapak Ir. Joko Susila, MT. atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaikannya Proyek Akhir ini, Penulis juga mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Proyek Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Proyek Akhir ini. Akhir kata, semoga Proyek Akhir ini dapat bermanfaat dalam pengembangan keilmuan di kemudian hari.

Surabaya,20 Januari 2020

Ilham Dwiki Ramadhan

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

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

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

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini teknologi berkembang dengan cepat. Dari waktu ke waktu semakin banyak bermunculan alat yang dibuat untuk mempermudah pekerjaan karena manusia membutuhkan mobilitas tinggi dalam melakukan pekerjaan serta otomasisasi sehingga mendapat kemudahan dari teknologi tersebut.

Semakin pesatnya perkembangan teknologi di era industri modern sekarang ini, berbagai macam teknologi banyak bermunculan mulai dari teknologi baru, sampai teknologi yang merupakan perkembangan dari teknologi sebelumnya. Pada jaman sekarang berbagai macam bencana alam sering terjadi mulai bencana besar yang menelan banyak korban jiwa seperti gempa bumi sampai bencana kecil seperti banjir, bencana-bencana tersebut memporak porandakan rumah serta lingkungan di sekitarnya sehingga jika ingin mengevakuasi para korban membutuhkan waktu yang lama untuk mencari keberadaan korban yang terselip di puing bangunan yang hanya dapat dilihat jika dipantau dari udara. Berdasarkan permasalahan tersebut, maka dibutuhkan teknologi seperti quadcopter. Quadcopter merupakan robot penjelajah udara Unmanned Aerial Vehicle (UAV) yang termasuk kategori UAV mikro dan banyak digunakan oleh beberapa lembaga atau instansi. Robot quadcopter merupakan UAV yang memiliki ciri khusus yang mudah dikenali yaitu memiliki empat buah baling-baling motor yang digunakan sebagai penggeraknya.

Quadcopter memiliki beberapa kelebihan yang menjadikannya cocok untuk melakukan pekerjaan tertentu. Bentuknya yang kecil membuat quadcopter cukup leluasa untuk bergerak di tempat-tempat yang sulit. Sistem penggerak pada quadcopter menggunakan empat buah motor yang merupakan sinkronisasi antara dua buah motor yang berputar searah jarum jam dan dua buah motor yang berputar berlawanan dengan arah jarum jam. Dengan sistem tersebut quadcopter memiliki tantangan dalam segi kontrol yang menarik kalangan industri maupun universitas untuk mengembangkan sistem tersebut. Salah satunya adalah dengan menggunakan teknologi GPS (Global Positioning System) sebuah quadcopter diharapkan mampu

mempertahankan posisi pada koordinat yang ditentukan, sehingga dengan adanya teknologi tersebut quadcopter dapat diimplementasikan untuk berbagai bidang antara lain, inspeksi pada aerial photography, pemetaan, dan sebagai satelit.

Kelebihan dari penggunaan quadcopter sendiri adalah simple, tidak perlu membutuhkan bahan bakar seperti helicopter dan portable atau dapat dibawa kemanapun. Untuk biaya produksi sendiri jauh lebih terjangkau dari pada memakai helicopter, cukup dengan menambahkan action cam pada quadcopter , dan hasil dari monitoring action cam tersebut dapat diihat di laptop atau handphone pengguna.

Mengacu pada latar belakang tersebut penulis mempunyai gagasan untuk membuat sebuah quadcopter menggunakan STM32F1 sebagai flight controllernya juga sebagai pengatur kerja sistem gyro, barometer ,dan GPS.

1.2 Permasalahan

Perumusan masalah yang dibahas dalam pengerjaan Proyek Akhir ini adalah sebagai berikut:

1. Pemantauan terhadap korban bencana alam melalui udara masih memgandalkan helicopter yang tidak dapat langsung datang ketika dibutuhkan.
2. Penggunaan helicopter tidak praktis dan efisien
3. Letak lokasi bencana alam terjadi berada di tempat terpencil ,sehingga membutuhkan alat yang portable.

1.3 Tujuan

Tujuan yang dicapai dalam penyelesaian pada Proyek Akhir ini adalah sebagai berikut

1. Merealisasikan quadcopter yang dilengkapi dengan GPS guna pelacakan posisi korban bencana alam berbasis STM32F1.
2. Cara menyajikan dan pengiriman data berupa data GPS,tekanan udara, tegangan baterai quadcopter melalui alat telemetry.

1.4 Sistematika Laporan

Untuk pembahasan lebih lanjut, laporan Proyek Akhir ini disusun dengan sistematika sebagai berikut:

Bab I PENDAHULUAN

Membahas tentang latar belakang, perumusan masalah, batasan masalah, maksud dan tujuan, sistematika laporan, metodologi, serta relevansi Proyek Akhir yang dibuat.

Bab II TEORI PENUNJANG

Menjelaskan teori yang berisi teori-teori dasar yang dijadikan landasan dan mendukung dalam perencanaan dan pembuatan alat yang dibuat.

Bab III PERENCANAAN DAN IMPLEMENTASI

Membahas perencanaan dan pembuatan tentang perencanaan dan pembuatan *hardware* yang meliputi desain mekanik dan perancangan s*oftware* yang meliputi program yang akan digunakan untuk menjalankan alat tersebut.

Bab IV PENGUJIAN DAN ANALISA

Membahas pengujian alat dan menganalisa data yang didapat dari pengujian tersebut serta membahas tentang pengukuran, pengujian, dan penganalisaan terhadap alat.

Bab V PENUTUP

Berisi penutup yang menjelaskan tentang kesimpulan yang didapat dari Proyek Akhir ini dan saran-saran untuk pengembangan alat ini lebih lanjut.

1.5 Relevansi

Membantu pengguna dalam pencarian korban bencana alam terutama pada lokasi terpencil yang susah dilewati, dengan pantauan melalui udara memudahkan proses monitoring. Mengetahui lokasi latitude dan longtitude korban dengan pelacakan menggunakan sistem GPS.

1.6 Batasan Masalah

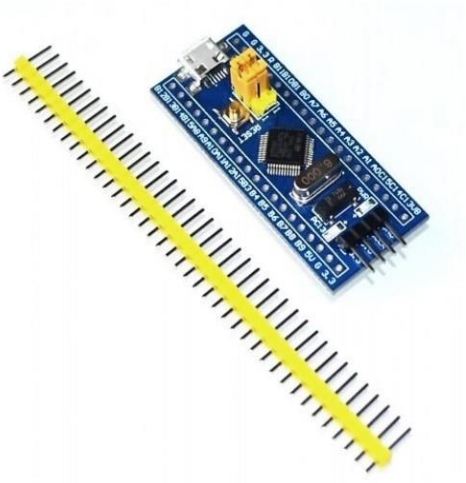
Dalam pembuatan alat untuk proyek akhir ini, alat mempunyai batasan – batasan masalah sebagai berikut ini.

1. Pemantauan harus dibantu dengan kamera external yang dipasang pada quadcopter seperti action cam sehingga hasil visual dapat di monitoring melalui gadget secara real time.
2. Latitude dan longtitude hasil pelacakan sistem GPS tidak akurat 100%.
3. Modul RF pada telemetry sering hilang koneksi

BAB II
TEORI PENUNJANG

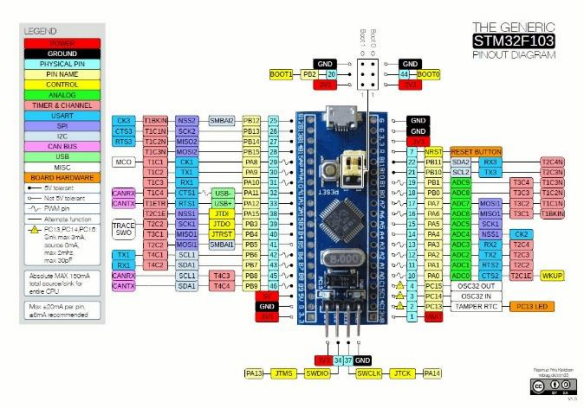
2.1 STM32F103C8T6

mikrokontroler berbasis inti prosesor 32 bit RISC ARM Cortex-M7, Cortex-M4F, Cortex-M3, Cortex-M0+, dan Cortex-M0 dari STMicrelectronics. Mikrokontroler ini mempunyai frekuensi clock tinggi, umumnya berada pada kisaran 72MHz atau lebih. Dibandingkan dengan Arduino uno yang hanya memilik clock sebesar 16MHz tentu memiliki perbedaan yang besar ketika melakukan transfer data [1]



Gambar 2. 1 STM32F103C8T6
(Sumber https://embeddednesia.com/v1/berkenalan-dengan-stm32-mikrokontroler-32-bit)

STM32F103C8T6 memiliki 44 pin ,untuk pin Tx adalah PA9 sedangkan Rx adalah PA10 seperti pada Tabel 2.1 ,STM32F1 dapat deprogram secara serial menggunakan STLink atau FTDI tool.Pinout dari minimum system STM2F103C8T6 dapat dilihat pada Gambar 2.2



Gambar 2. 2 Pinout STM32F103C8T6
(Sumber https://embeddednesia.com/v1/berkenalan-dengan-stm32-mikrokontroler-32-bit)

Tabel 2. 1 Pin Rx dan Tx pada STM32F1

Pin	Nama Pin	Keterangan
PA9	*TX*	Pengirim data
PA10	RX	Penerima data

2.2 GPS Ublox M8N

Ublox M8N menyediakan kinerja terbaik dan integrasi RF termudah dari semua konfigurasi ublox M8. Ublox M8N merupakan generasi baru Ublox GPS yang mampu akurasi pada jarak 0.6 hingga 0.9 meter. Ublox M8N juga memiliki built dalam kompas dengan tingkat refresh 10GHz sehingga kinerja lebih baik.[2]

Semakin banyak satelit yang diperoleh maka akurasi posisi kita akan semakin tinggi. Untuk mendapatkan sinyal tersebut, perangkat GPS harus berada di ruang terbuka. Apabila perangkat GPS kita berada dalam ruangan atau kanopi yang lebat dan daerah kita dikelilingi oleh gedung tinggi maka sinyal yang diperoleh akan semakin berkurang

Gambar 2. 3 GPS Ublox M8N
(Sumber http://buaya-instrument.com/gps-neo-m8n-for-apm-bonus-stand-1107990006.html)

2.3 MPU 6050

Sensor MPU6050 adalah sensor mampu membaca kemiringan sudut berdasarkan data dari sensor accelerometer dan sensor gyroscope.Sensor MPU-6050 berisi sebuah MEMS Accelerometer dan sebuah MEMS Gyro yang saling terintegrasi. Sensor ini sangat akurat dengan fasilitas hardware internal 16 bit ADC untuk setiap kanalnya. Sensor ini akan menangkap nilai kanal axis X, Y dan Z bersamaa dalam satu waktu.Untuk komunikasi, sensor ini menggunakan komunikasi I2C yang artinya hanya memerlukam dua kabel data dan dua lagi untuk power [3]

Gambar 2. 4 MPU 6050
(Sumber https://tutorkeren.com/artikel/cara-menggunakan-accelerometer-gyro-mpu6050-arduino-dengan-motor-servo.html)

2.4 Motor Brushless

Motor brushless adalah motor yg umumnya memiliki tiga buah coil yang masing-masing memiliki satu kabel untuk masuk ke ESC atau baterai, sehingga jika dilihat dari luar brushless motor selalu memiliki tiga buah kabel. Di motor brushless ini, yang berputar bukanlah bagian coil-nya, karena coil brushless menempel di chasing motornya. Motor brushless direct current (BLDC) adalah motor yang tidak menggunakan sikat atau brush untuk pergantian medan magnet (komutasi) tetapi dilakukan secara komutasi elektronis. Perbedaan utama antara motor DC magnet permanen (DC-MP) dengan motor brushless DC adalah terletak pada pembangkitan medan magnet untuk menghasilkan gaya gerak. Jika pada motor DC-MP medan magnet yang dikontrol berada di rotor dan medan magnet tetap berada di stator. Sebaliknya, motor brushless menggunakan pembangkitan medan magnet stator untuk mengontrol geraknya sedang medan magnet tetap berada di rotor. [4]

Gambar 2. 5 Brushless Motor
(Sumber https://kleenrc.com/products/turnigy-d2836-8-1100kv-brushless-outrunner-motor-1)

2.5 *Electronic Speed Control*(ESC)

Electronic Speed Controller yang digunakan adalah berjenis brushless, terdiri atas susunan MOSFET (Metal Oxide Semiconductor Field Effect Transistor) untuk mengendalikan kecepatan motor brushless. ESC bekerja secara cepat untuk menghidupkan atau mematikan pulsa ke motor, sehingga respon kendali motor cepat. Selain

itu ESC yang digunakan telah berbasis mikroprosessor, sehingga dapat diprogram sesuai dengan kebutuhan

Gambar 2. 6 Electronic Speed Controller
(Sumber https://sea.banggood.com/SimonK-30A-2-3S-Procedure-Brushless-ESC-For-RC-Model-p-919129.html?cur_warehouse=CN)

2.6 MS5611

Sebuah sensor barometer yang berfungsi untuk mengukur tekanan. Tekanan adalah ekspresi dari gaya yang dibutuhkan untuk menghentikan cairan dari perluasan,dan biasanya dinyatakan dalam gaya per satuan luas. Sebuah sensor tekanan biasanya bertindak sebagai transduser, yaitu menghasilkan sinyal sebagai fungsi dari tekanan yang dikenakan. Untuk mengubah data tekanan udara menjadi ketinggian dapat menggunakan rumus :

$$P_{atm} = \rho g h$$

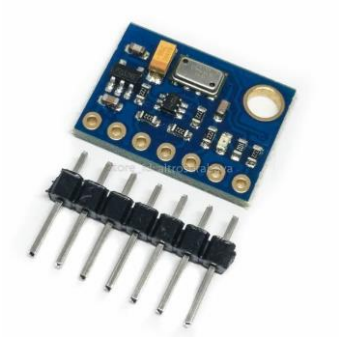
Dimana :

P = tekanan atmosfer

g = percepatan gravitasi

h= ketinggian/kedalaman

Bentuk asli dari sensor barometer MS5611dapat dilihat pada Gambar 2.7



Gambar 2. 7 MS5611
(Sumber https://beetrona.com/product/gy63-ms5611-high-resolution-atmospheric-pressure-gy-63/)

2.7 FTDI FT232RL

Gambar 2. 8 FTDI 232RL
(Sumber http://www.jogjarobotika.com/usb-converter-module/251-ft232rl-usb-to-serial-module-usb-to-ttl.html)

Pin pada board ini cocok dengan kabel FTDI yang umum digunakan untuk memprogram board Arduino juga dapat digunakan pada aplikasi umum yang menggunakan koneksi serial. Perbedaan utama dari board ini dengan yang sejenis adalah digunakannya pin DTR dan bukannya pin RTS seperti pada kabel FTDI. Pin DTR ini

memungkinkan board Arduino yang akan di-program untuk melakukan auto-reset setiap kali dimasukkannya sketch (program) kedalam Arduino. Ini adalah fitur yang bagus untuk dimiliki, sehingga tidak perlu lagi melakukan reset secara manual pada saat memasukkan sketch. Board ini akan me-reset secara otomatis board Arduino apapun yang menyediakan reset pin pada konektornya.Pin dengan label BLK dan GRN pada board ini sama dengan warna yang digunakan pada kabel FTDI. Warna hitam pada kabel FTDI adalah GND dan warna hijau adalah DTR. Gunakan warna hitam dan hijau ini untuk mencocokkan posisi FTDI basic board ini dengan board Arduino yang ingin di-program.[5]

2.8 APC220

APC 220 merupakan modul komunikasi semi duplex transcieve, modul ini dapat mengirimkan data serial melalui perantara media udara. Device tersebut melakukan proses penumpangan data serial digital ke frekuensi pembawa dengan frekuensi yang lebih tinggi untuk kemudian dipancarkan ke udara oleh pemancar.Modul APC220 dapat dilihat pada Gambar 2.9

Gambar 2. 9 APC220
(Sumber https://indonesian.alibaba.com/product-detail/apc220-wireless-serial-data-module-with-usb-adapter-kit-for-arduinos-60674334720.html)

APC 220 Wireless data transceiver dapat mengirimkan dan menerima data serial melalui media udara, dengan rentang frekuensi yang dapat digunakan dari 418 MHz – 455 MHz dengan kecepatan praktis karena dari segi ukuran cukup kecil dan penggunaan pin nya cukup mudah. Modul tersebut bekerja dengan supply antara 3.5 VDC sampai 5 VDC. Dalam satu modul bisa digunakan sebagai pengirim sekaligus penerima dalam waktu yang berbeda.data serial yang dipancarkan melalui RF diumpamakan ke modul APC oleh mikrokontroler secara serial. Begitu pula data yang diterima, akan di ambil oleh mikrokontroler secara serial. Untuk spesifikasi APC220 dapat dilihat pada Tabel 2.2 di bawah ini

Tabel 2. 2 Spesifikasi APC220

No	Spesifikasi	Keterangan
1	Range kerja	418 Mhz – 455 Hz
2	Catu daya	3.5 – 5 V
3	Jenis komunikasi	UART / TTL
4	Jangkauan	1000 – 1200 m
5	Arus daya Transmitter	35 Ma
6	Arus daya Receiver	32 Ma
7	Output daya	20 mW
8	Sensitivitas	-117 db @1200bps

2.9 Arduino Uno

Arduino Uno adalah sebuah board yang menggunakan mikrokontroler ATmega328. Arduino Uno memiliki 14 pin digital (6 pin dapat digunakan sebagai output PWM), 6 input analog, sebuah 16 MHz osilato kristal, sebuah koneksi USB, sebuah konektor sumber tegangan, sebuah header ICSP, dan sebuah tombol reset. Arduino Uno memuat segala hal yang dibutuhkan untuk mendukung sebuahmikrokontroler.[6]

Hanya dengan menghubungkannya ke sebuah komputer melalui USB atau memberikan tegangan DC dari baterai atau adaptor AC ke DC sudah dapat membuanya bekerja. Arduino Uno

menggunakan ATmega16U2 yang diprogram sebagai USB to serial converter untuk komunikasi serial ke komputer melalui port USB. "Uno" berarti satu di Italia dan diberi nama untuk menandai peluncuran Arduino 1.0. Versi 1.0 menjadi versi referensi Arduino ke depannya. Arduino Uno R3 adalah revisi terbaru dari serangkaian board Arduino, dan model referensi untuk platform Arduino. Tampak atas dari arduino uno dapat dilihat pada Gambar 2.10

Gambar 2. 10 Arduino Uno
(Sumber https://ilearning.me/sample-page-162/arduino/pengertian-arduino-uno/)

2.9.1 Pin Masukan dan Keluaran Arduino Uno

Masing-masing dari 14 pin digital arduino uno dapat digunakan sebagai masukan atau keluaran menggunakan fungsi pinMode(), digitalWrite() dan digitalRead(). Setiap pin beroperasi pada tegangan 5 volt. Setiap pin mampu menerima atau menghasilkan arus maksimum sebasar 40 mA dan memiliki 10 resistor pull-up internal (diputus secara default) sebesar 20-30 KOhm. Sebagai tambahan, beberapa pin masukan digital memiliki kegunaan khusus yaitu:

1. Komunikasi serial: pin 0 (RX) dan pin 1 (TX), digunakan untuk menerima (RX) dan mengirim (TX) data secara serial.
2. External Interrupt: pin 2 dan pin 3, pin ini dapat dikonfigurasi untuk memicu sebuah interrupt pada nilai rendah, sisi naik atau turun, atau pada saat terjadi perubahan nilai.

3. Pulse-width modulation (PWM): pin 3, 5, 6, 9, 10 dan 11, menyediakan keluaran PWM 8-bit dangan menggunakan fungsi analogWrite(). 4
4. Serial Peripheral Interface (SPI): pin 10 (SS), 11 (MOSI), 12 (MISO) dan 13 (SCK), pin ini mendukung komunikasi SPI dengan menggunakan SPI library.
5. LED: pin 13, terdapat built-in LED yang terhubung ke pin digital 13. Ketika pin bernilai High maka LED menyala, sebaliknya ketika pin bernilai Low maka LED akan padam.

Arduino Uno memiliki 6 masukan analog yang diberi label A0 sampai A5, setiap pin menyediakan resolusi sebanyak 10 bit (1024 nilai yang berbeda). Secara default pin mengukur nilai tegangan dari ground (0V) hingga 5V, walaupun begitu dimungkinkan untuk mengganti nilai batas atas dengan menggunakan pin AREF dan fungsi analogReference(). Sebagai tambahan beberapa pin masukan analog memiliki fungsi khusus yaitu pin A4 (SDA) dan pin A5 (SCL) yang digunakan untuk komunikasi Two Wire Interface (TWI) atau Inter Integrated Circuit (I2C) dengan menggunakan Wire library.

2.9.2 Catu Daya

Arduino uno dapat diberi daya melalui koneksi USB (Universal Serial Bus) atau melalui power supply eksternal. Jika arduino uno dihubungkan ke kedua sumber daya tersebut secara bersamaan maka arduino uno akan memilih salah satu sumber daya secara otomatis untuk digunakan. Power supply eksternal (yang bukan melalui USB) dapat berasal dari adaptor AC ke DC atau baterai. Adaptor dapat dihubungkan ke soket power pada arduino uno. Jika menggunakan baterai, ujung kabel yang dibubungkan ke baterai dimasukkan kedalam pin GND dan Vin yang berada pada konektor power. Arduino uno dapat beroperasi pada tegangan 6 sampai 20 volt.

Tegangan rekomendasi yang diberikan ke arduino uno berkisar antara 7-12 volt. Pin-pin catu daya adalah sebagai berikut:

1. Vin adalah pin untuk mengalirkan sumber tegangan ke arduino uno ketika menggunakan sumber daya eksternal (selain dari koneksi USB atau sumber daya yang teregulasi lainnya). Sumber tegangan juga dapat disediakan melalui

pin ini jika sumber daya yang digunakan untuk arduino uno dialirkan melalui soket power.

2. 5V adalah pin yang menyediakan tegangan teregulasi sebesar 5 volt berasal dari regulator tegangan pada arduino uno.
3. 3V3 adalah pin yang meyediakan tegangan teregulasi sebesar 3,3 volt berasal dari regulator tegangan pada arduino uno.
4. GND adalah pin ground.

2.10 Buzzer 5V

Buzzer adalah sebuah komponen elektronika yang berfungsi untuk mengubah getaran listrik menjadi getaran suara. Pada dasarnya prinsip kerja buzzer hampir sama dengan loud speaker, jadi buzzer juga terdiri dari kumparan yang terpasang pada diafragma dan kemudian kumparan tersebut dialiri arus sehingga menjadi elektromagnet, kumparan tadi akan tertarik ke dalam atau keluar, tergantung dari arah arus dan polaritas magnetnya, karena kumparan dipasang pada diafragma maka setiap gerakan kumparan akan menggerakkan diafragma secara bolak-balik sehingga membuat udara bergetar yang akan menghasilkan suara. Buzzer biasa digunakan sebagai indikator bahwa proses telah selesai atau terjadi suatu kesalahan pada sebuah alat (alarm).

Gambar 2. 11 Buzzer 5V
(Sumber https://www.ajifahreza.com/2017/04/menggunakan-buzzer-komponen-suara.html)

2.11 LCD Keypad Shield

Liquid crystal display (LCD) adalah komponen yang dapat menampilkan tulisan dengan memanfaatkan kristal cair, salah satu jenisnya adalah LCD keypad shield yang memiliki dua baris setiap baris

terdiri dari enam belas karakter atau memakai LCD berukuran 16x2.[7] Cara menggunakan LCD keypad shield adalah dipasang diatas pin Arduino uno. Gambar LCD keypad shield dapat dilihat pada Gambar 2.12 dan Gambar 2.13..

Gambar 2. 12 LCD Keypad Shield
(Sumber http://www.labelektronika.com/2015/06/cara-memprogram-lcd-karakter-keypad-shield.html)

LCD ini memiliki 16 *pin* dengan fungsi *pin* masing – masing diperlihatkan pada Tabel 2.3.

.

Tabel 2. 3 Pin pada LCD display

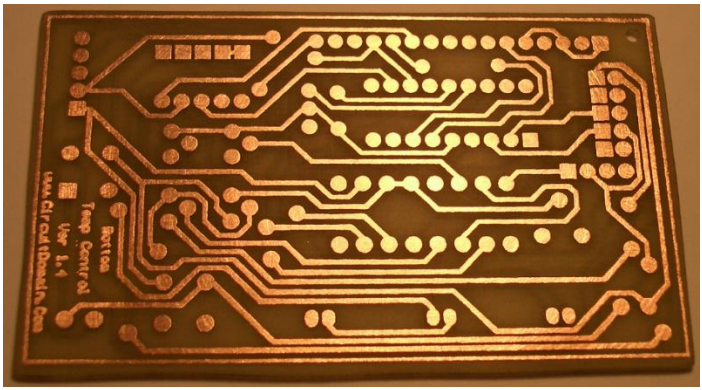
No. Pin	Nama Pin	I/O	Keterangan
1	*GROUND*	*Power*	Catu daya, ground (0V)
2	VCC	*Power*	Catu daya positif
			Pengatur kontras. Menurut datasheet, pin ini perlu dihubungkan dengan pin VSS melalui resistor 5kΩ. Namun, dalam praktik, resistor yang digunakan sekitar 2,2kΩ.
3	CONTR	*Power*	*Register Select*
4	RS	*Input*	RS=*HIGH*: untuk mengirim data
			RS=*LOW*:Untuk mengirim instruksi
5	R/W	*Input*	*Read/Write control bus*
			R/W=*HIGH*: mode untuk membaca data

2.12 Printed Circuit Board

Dalam kehidupan ini tidak terlepas dari penggunaan barang elektronik seperti televisi, handphone, komputer, radio dan peralatan elektronik lainnya. Didalam peralatan tersebut terdapat banyak komponen-komponen elektronika yang membentuk satu rangkaian sehingga menjadi sistem yang dibuat untuk tujuan tertentu. Komponen-komponen tersebut biasanya disusun dan dipasang pada papan rangkaian yang disebut PCB (*Printed Circuit Board*).

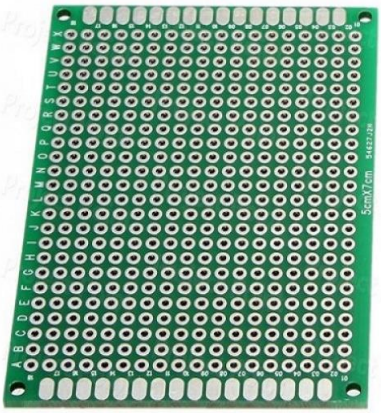
Printed Circuit Board disingkat PCB adalah sebuah papan komponen-komponen elektronika yang tersusun membentuk rangkaian elektronik atau tempat rangkaian yang menghubungkan komponen elektronik yang satu dengan lainnya tanpa menngunakan kabel. Disebut papan sirkuit karena diproduksi secara massal dengan cara mencetak.

Ada tiga tipe PCB yang sering digunakan yaitu *single side*, *double side* dan *multi layer*. *Single side* artinya papan PCB tersebut hanya mempunyai satu sisi dilapisi oleh lempeng tembaga. *Double side* artinya papan PCB tersebut mempunyai dua sisi yang dilapisi oleh lempeng tembaga dan lapisan fiber-nya ada diantara dua lapisan tembaga tersebut, sehingga dapat membuat jalur di layer atas maupun layer bawah. *Multi layer* terdiri dari beberapa lapis tembaga yang bersifat konduktor yang disusun secara bergantian , contoh gambar PCB dapat dilihat pada Gambar 2.13 di bawah ini.



Gambar 2. 13 Printed Circuit Board

Pada proyek akhir ini digunakan Printed Circuit Board jenis dot matrix atau biasa disebut dengan PCB lubang, dengan demikian komponen-komponen yang dibutuhkan dapat langsung dipasang dan disolder, kelebihan Printed Circuit Board jenis ini adalah kita dapat langsung meletakan komponen yang kita butuhkan dan langsung melakukan *soldering* tanpa kita desain terlebih dahulu menggunakan software desain PCB. Namun karena hal tersebut membuat penggunaan Printed Circuit Board jenis ini membutuhkan ketelitian lebih dalam hal pemasangan maupun soldering nya , juga dalam hal pemakaian timah dan kabel jumper dapat dikatakan lebih boros ketimbang Printed Circuit Board yang telah didesain terlebih dahulu. Printed Circuit Board jenis dot matrix dapat dilihat pada Gambar 2.14 di bawah ini.



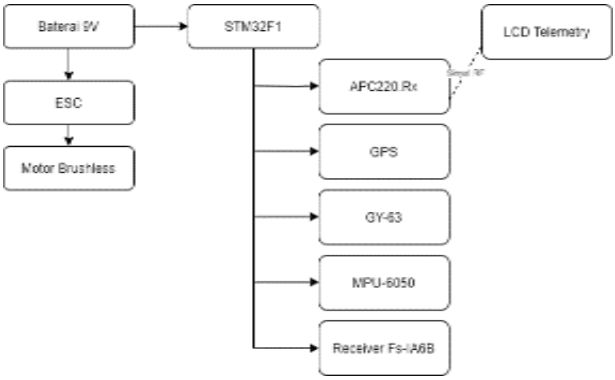
Gambar 2. 14 PCB Dot Matrix

BAB III
PERENCANAAN DAN IMPLEMENTASI

Bab ini membahas perancangan dari perangkat keras dan perangkat lunak Rancang Bangun *Quadcopter* Menggunakan STM32F1 Guna Pelacakan Korban Bencana Alam dengan GPS. Bab ini berfokus pada blok diagram sistem pada flight controller maupun pada alat telemetry, perancangan mekanik, perancangan perangkat elektrik dan perancangan software

3.1 Blok Diagram Sistem Keseluruhan

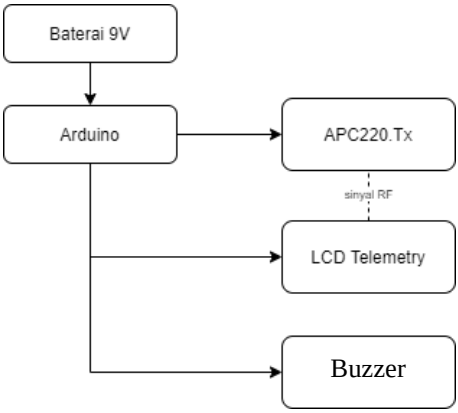
Blok Diagram sistem pada proyek akhir ini terbagi menjadi dua yaitu diagram fungsional sistem pada flight controller dan diagram fungsional sistem pada alat telemetry



Gambar 3. 1 Diagram Fungsional Sistem Flight Controller

Berdasarkan sistem 3.1 alur kerja sistem flight controller yaitu baterai lipo 11.1v mensupply Electronic Speed Controller (ESC) dan STM32F1 ,kemudian STM32F1 membaca hasil kecepatan sudut dari sumbu x , y ,dan z melalui MPU 6050, membaca hasil koordinat latitude dan longtitude dari GPS Ublox M8N, serta membaca hasil tekanan udara melalui GY 63 atau MSC5611. Sinyal PPM dari receiver masuk pada pin A0 STM32F1. Kemudian data GPS,gyro, dan

barometer tersebut dikomunikasikan pada LCD telemetry melalui sinyal RF (Radio Frekuensi) menggunakan APC220 dengan sambungan pin Rx.



Gambar 3. 2 Diagram Fungsional Sistem Telemetry

Pada diagram fungsional sistem telemetry diketahui bahwa baterai 9v mensupply Arduino, kemudian Arduino memprogram buzzer , LCD , dan APC220 dengan sambungan pin Tx mendapat hasil pengukuran dari modul di flight controller yang kemudian ditampilkan pada LCD alat telemetry

3.2 Perancangan Perangkat Mekanik

Dalam perancangan perangkat Mekanik pada proyek akhir ini dibagi menjadi dua tahap perancangan yang dilakukan sebagai berikut:

- Perancangan Frame quadcopter
- Pemilihan propeller quadcopter

3.2.1 *Perancangan Frame Quadcopter*

Pada perancangan frame quadcopter, dipilih frame F450 yang biasa dipakai jika ingin membuat diy quadcopter dan dapat didapatkan di toko online, frame 450 terdiri dari board power distribution , board polos untuk meletakan controller dan frame arm sebanyak 4 buah , kemudian dirancang dan hasilnya dapat dilihat pada Gambar 3.3

Gambar 3.3 Hasil Perancangan Frame F450

3.2.2 Pemilihan Propeller Quadcopter

Pada proyek akhir ini dipilih propeller berukuran 10 x 4.5 yang didapat dari toko online, dipilihnya propeller ukuran tersebut karena sesuai dengan besar dan berat dari frame quadcopter F450, propeller berukuran 10x4.5 dapat dilihat pada Gambar 3.4 .

Gambar 3.4 Propeller ukuran 10 x 4.5

3.3 Perancangan *Elektronik.*

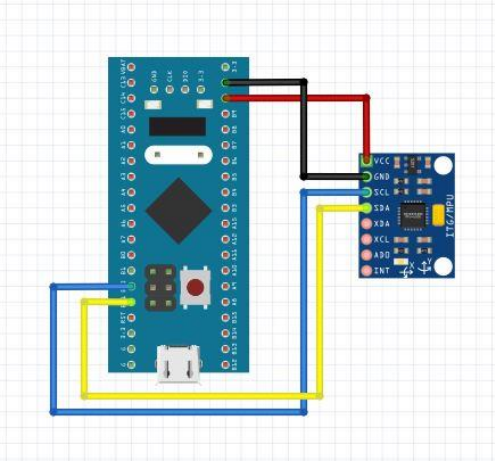
Dalam perancangan elektrik pada proyek akhiri ini terbagi menjadi delapan macam wiring. Spesifikasi dari masing-masing wiring tersebut adalah :

1. Wiring STM32F1 dan MPU 6050
2. Wiring STM32F1 dan GPS Ublox M8N

3. Wiring STM32F1 dan GY 63 / MS5611
4. Wiring STM32F1 dan FTDI
5. Wiring STM32F1 dan APC220 RX
6. Wiring STM32F1 dan receiver FS IA6B
7. Wiring STM32F1 dan ESC + brushless motor
8. Wiring rangkaian telemetry

3.3.1 Wiring STM32F1 dan MPU 6050

Wiring STM32F1 dan MPU 6050 dapat dilihat pada Gambar 3.5



Gambar 3.5 Wiring STM32F1 dan MPU 6050

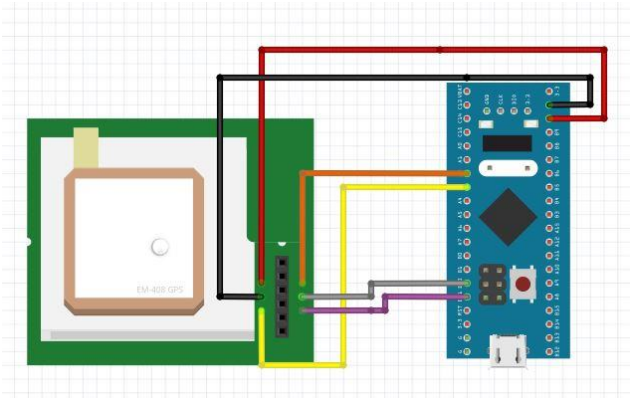
MPU 6050 gyro + accelerometer digunakan untuk membaca kecepatan sudut dari sumbu x , y , dan z serta menstabilkan quadcopter ketika terbang, sambungan pin MPU 6050 ke STM32F1 dapat dilihat pada tabel 3.1

Tabel 3. 1 Sambungan Pin MPU 6050 ke STM32F1

Pin STM32F1	Pin MPU 6050
5V	VCC
GND	GND
B10	SCL
B11	SDA

3.3.2 Wiring STM32F1 dan GPS Ublox M8N

Wiring STM32F1 dan GPS Ublox M8N dapat dilihat pada Gambar 3.6



Gambar 3.6 Wiring STM32F1 dan GPS Ublox M8N

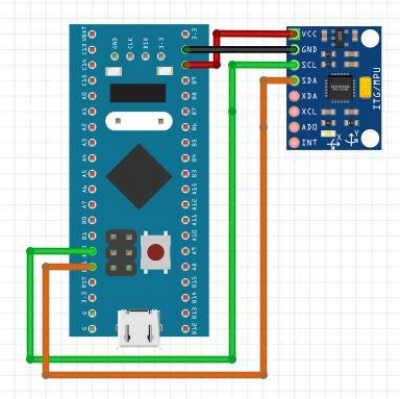
GPS Ublox M8N digunakan untuk melacak posisi koordinat Latitude dan longtitude melalui satelit yang dikirimkan ke modul. Kemudian data latitude dan longtitude yang didapat dimasukan ke google maps untuk dilihat lokasi asli berdasarkan data terebut,sambungan pin GPS Ublox M8N ke STM32F1 dapat dilihat pada Tabel 3.2

Tabel 3. 2 Sambungan Pin GPS Ublox M8N ke STM32F1

Pin GPS Ublox M8N	Pin STM32F1
VCC	5V
GND	GND
TX	A3
RX	A2
SCL	B10
SDA	B11

3.3.3 Wiring STM32F1 dan MS5611

Wiring STM32F1 dan MS5611 dapat dilihat pada Gambar 3.7



Gambar 3.7 Wiring STM32F1 dan MS5611

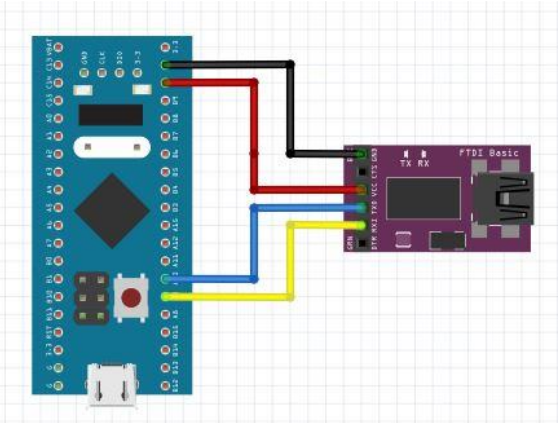
MS5611 adalah modul barometer yang digunakan untuk menghasilkan output atmospheric pressure (tekanan udara), modul GY-63 sangat cocok untuk mengukur relative altitude atau ketinggian relatif. Cukup dengan algoritma atau rumus konversi tekanan ke ketinggian. Sambungan pin MS5611 ke STM32F1 dapat dilihat pada Tabel 3.3

Tabel 3. 3 Sambungan Pin MS5611 ke STM32F1

Pin STM32F1	Pin MS5611
3.3V	VCC
GND	GND
B10	SCL
B11	SDA

3.3.4 *Wiring* STM32F1 dan FTDI

Wiring STM32F1 dan FTDI Tool dapat dilihat pada Gambar 3.8



Gambar 3.8 Wiring STM32F1 dan FTDI Tool

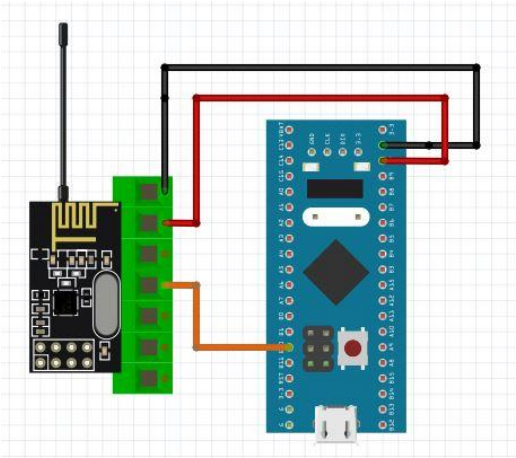
FTDI Tool berfungsi untuk mengkonversi USB ke data serial. Dalam proyek akhir ini FTDI Tool digunakan untuk memprogram STM32F1 secara serial. Sambungan pin FTDI ke STM32F1 dapat dilihat pada Tabel 3.4

Tabel 3. 4 Sambungan Pin FTDI ke Pin STM32F1

Pin STM32F1	Pin FTDI
5V	VCC
GND	GND
A10	TX
A9	RX

3.3.5 *Wiring* STM32F1 dan APC220

Wiring STM32F1 dan APC220 dapat dilihat pada Gambar 3.9



Gambar 3.9 Wiring STM32F1 dan APC220

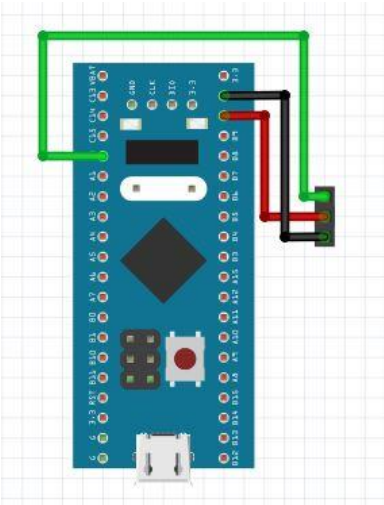
APC 220 Wireless dgunakan untuk data transceiver mengirimkan dan menerima data serial melalui media udara, dengan rentang frekuensi yang dapat digunakan dari 418 MHz – 455 MHz dengan kecepatan praktis. data serial yang dipancarkan melalui RF diumpamakan ke modul APC oleh mikrokontroler secara serial. Begitu pula data yang diterima, akan diambil oleh STM32F1 secara serial. Sambungan pin APC220 ke STM32F1 dapat dilihat pada Tabel 3.5

Tabel 3. 5 Sambungan Pin APC220 ke Pin STM32F1

Pin STM32F1	Pin APC220
5V	VCC
GND	GND
B0	RX

3.3.6 *Wiring* STM32F1 dan Receiver FS-IA6B

Wiring STM32F1 dan APC220 dapat dilihat pada Gambar 3.10



Gambar 3. 10 *Wiring* STM32F1 dan Receiver FS IA6B

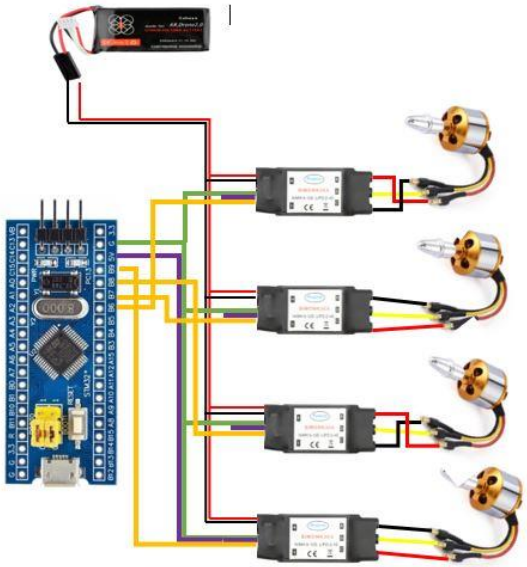
FS IA6B berfungsi receiver dari transmitter FS-i6 untuk mengontrol kerja ESC dan motor brushless pada quadcopter menggunakan remote. PPM signal dari FS IA6B disambungkan ke pin A0 STM32F1.Sambungan pin receiver ke STM32F1 dapat dilihat pada Tabel 3.6

Tabel 3. 6 Sambungan Pin Receiver FS-IA6B ke Pin STM32F1

Pin STM32F1	Pin Receiver FS-IA6B
5V	VCC
GND	GND
A0	PPM

3.3.7 *Wiring* STM32F1 dan ESC + Brushless Motor

Wiring STM32F1 dan APC220 dapat dilihat pada Gambar 3.11



Gambar 3.11 *Wiring* STM32F1 dan ESC + Brushless Motor

ESC singkatan dari Electronic Speed Controller. ESC adalah motor controller elektronik yang berguna mengontrol kecepatan, arah dan mungkin pengereman motor. Untuk mengubah arah putar motor seperti yang digunakan seperti arah CW (Clock Wise) atau searah jarum jam maupun arah CCW (Counter Clock Wise) atau berlawanan jarum jam dapat dengan cara mengganti wiring di bagian motor ke ESC.Sambungan STM32F1 ke ESC dan arah putar motor dapat dilihat pada Tabel 3.7

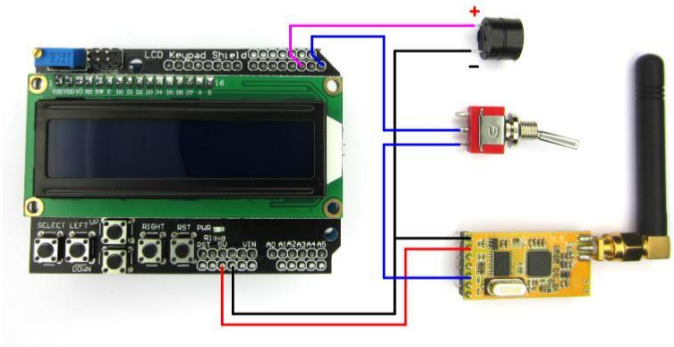
Tabel 3. 7 Sambungan STM32F1 ke ESC dan Arah Putar Motor

Motor Brushless	STM32F1	Posisi Motor	Arah Putar Motor
1	B6	Kanan depan	Counterclockwise
2	B7	Kanan belakang	Clockwise

3	B8	Kiri belakang	Counterclockwise
4	B9	Kiri depan	Clockwise

3.3.8 *Wiring* Rangkaian Telemetry

Rangkaian Telemetry terdiri dari Arduino Uno yang dipasang dengan LCD Shield, kemudian APC220, dan buzzer 5v. Wiring rangkaian telemetry dapat dilihat pada Gambar 3.12.



Gambar 3.12 *Wiring* Rangkaian Telemetry

Rangkaian Telemetry berfungsi untuk melihat data hasil modul yang terprogram oleh STM32F1 seperti data latitude dan longtitude GPS Ublox M8N, data kecepatan sudut dan arah sudut x, y , dan z dari MPU 6050 , data tekanan udara dari modul MS5611 dan data voltage dari baterai lipo. Untuk detail sambungan rangkaian telemetry dapat dilihat pada Tabel 3.8 di bawah ini

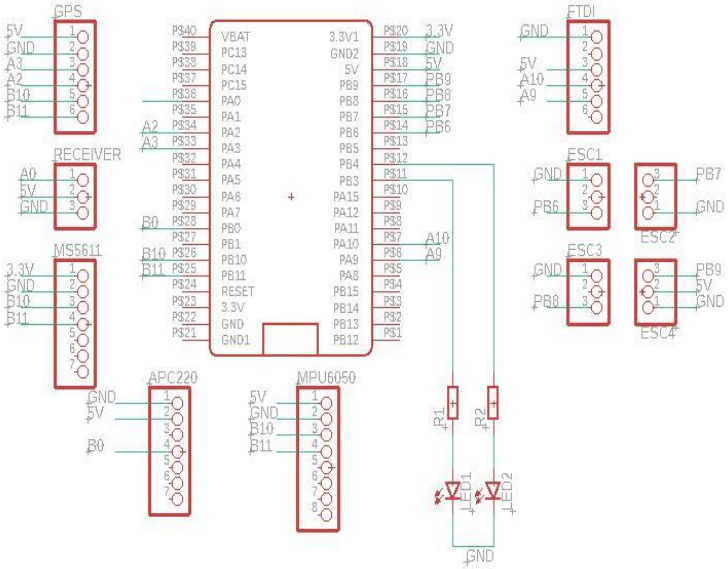
Tabel 3. 8 Sambungan Rangkaian Telemetry

Arduino LCD Shield	APC220	Buzzer 5V
Pin 2	-	Pin positif

GND	GND	Pin Negatif
Pin 0	TX	-
5V	VCC	-

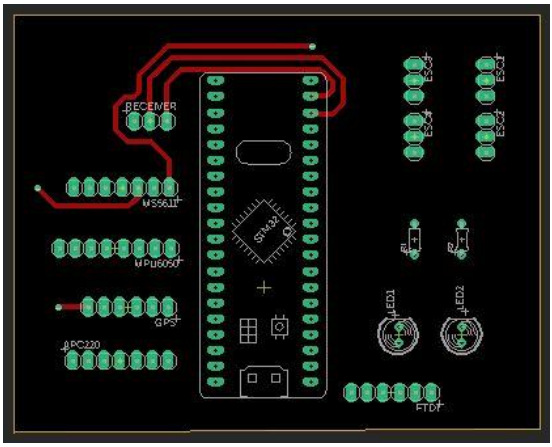
3.4 Desain PCB rangkaian pada *software Eagle*

Untuk desain PCB pada proyek akhir ini menggunakan *software Eagle* , pertama harus membuat skematik rangkaian terlebih dahulu ,berikut adalah skematik rangkaian flight controller pada software eagle yang ditunjukkan Gambar 3.13

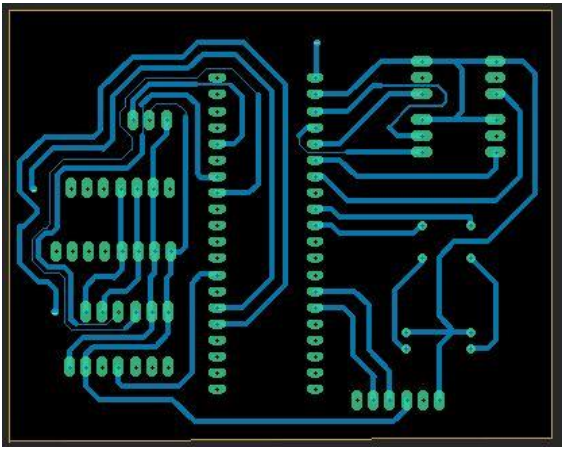


Gambar 3.13 Skematik Rangkaian pada *Software Eagle*

Setelah membuat skematik , kemudian *swith to board* dan dirouting , setelah proses peroutingan board akan nampak seperti Gambar 3.14 dan Gambar 3.15



Gambar 3. 14 Layout Board Layer Depan



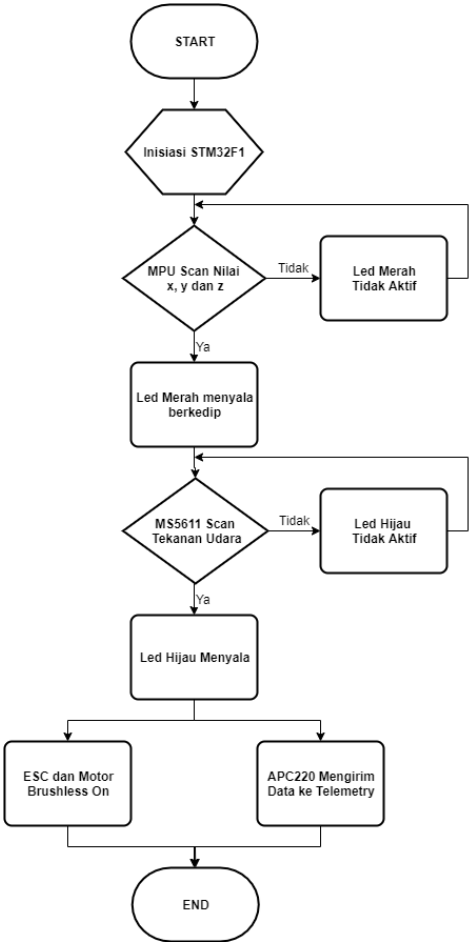
Gambar 3. 15 Layout Board Layer Belakang

3.5 Perancangan Software

Pada Sub bab ini menjelaskan dua macam flowchart yaitu:

1. sistem pada flight controller
2. sistem pada alat telemetry

3.5.1 Flowchart Pada Flight Controller



Gambar 3. 16 Flowchart Flight Controller

Flowchart cara kerja Flight Controller adalah :

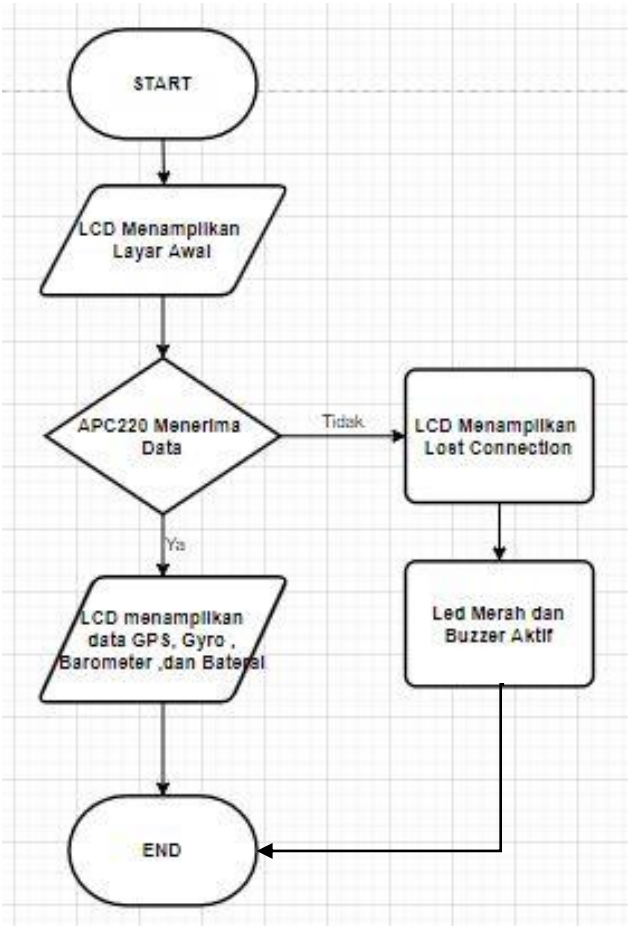
1. Start
   Langkah pertama untuk mengoperasikan alat yaitu dengan memberikan tegangan pada rangkaian

2. Inisiasi STM32F1
   Inisiasi STM32F1 agar dapat menjalankan sensor-sensor yang terhubung
3. MPU 6050 scan nilai x,y,dan z
   Setelah sistem STM32F1 aktif , akan melakukan fungsi sebagai control semua input dan output. STM32F1 mengaktifkan MPU 6050 dan membaca sumbu x, y ,dan z
4. Led Merah On
   Jika MPU 6050 berhasil kalibrasi led merah akan menyala berkedip
5. MS5611 scan tekanan udara
   Jika MPU 6050 berhasil kalibrasi maka MS5611 akan memulai kalibrasi
6. Led hijau On
   Jika led hijau menyala maka MS5611 berhasil dan esc+ motor siap menyala
7. ESC dan motor Brushless On
   Setelah MPU 6050 dan MS5611 berhasil kalibrasi , ESC dan motor dapat menyala ketika throttle transmitter digeser ke kiri
8. APC220 mengirim data ke telemetry
   Ketika MPU 6050 dan MS5611 telah melakukan kalibrasi , data MPU 6050, M5611 dan GPS akan tampil di LCD telemetry

3.5.2 Flowchart Pada Telemetry

1. Start
   Langkah pertama untuk mengoperasikan alat yaitu dengan memberikan tegangan pada rangkaian
2. LCD Menampilkan Tampilan Awal
   LCD menampilkan Tampilan awal dimana data sensor seperti Longtitude dan latitude GPS, data tekanan udara, dan data Baterai belum tampil
3. APC220 Menerima data
   APC220 akan menerima data yang terpasang pada Flight Controller seperti data GPS, MPU6050 dan data MS5611
4. LCD menampilkan data sensor dan baterai
   Jika APC220 berhasil menerima data, maka akan muncul data GPS, data barometer ,dan baterai

5. Led merah dan buzzer aktif
   Jika terjadi lost koneksi antara APC220 maka led merah dan buzzer akan menyala , serta LCD menampilkan tulisan Lost Connection



Gambar 3. 17 Flowchart Telemetry

BAB IV
PENGUJIAN DAN ANALISIS

Setelah melakukan perancangan Quadcopter beserta komponen-komponen nya, pada bab ini akan mebahas penujian dari sistem sistem yang dirancang. Bab ini bertuajuan untuk mendapatkan data yang kemudian dilakukan analisa masing masing pengujian, . Berikutnya adalah urutan pengujian yang dilakukan.

1. Pengujian dan Analisa pada Flight Controller
2. Pengujian dan Analisa pada Alat Telemetry

4.1 Pengujian Dan Analisa pada Flight Controller

Pada sub bab ini membahas semua komponen yang digunakan dalam board Flight Controller , dimana masing masing komponen dapat melakukan perintah sesuai dengan sistem keseluruhan yang di jelaskan pada *flowchart* system flight controller pada Gambar 3.16.

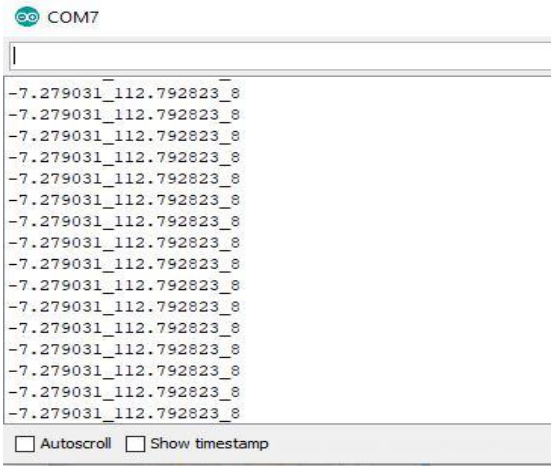
4.1.1 Pengujian STM32F1

STM32F1 digunakan untuk mengendalikan *input* dan *output* pada alat, sehingga STM32F1 memerlukan *supply* tegangan yang sesuai.

Pengukuran tegangan *input* pada mikrokontroler Mega menggunakan *Multimeter* analog adalah 2.3V. Dari pengukuran tegangan *input* tersebut menunjukkan bahwa hasil pengukuran sesuai dengan *datasheet* , STM32F1 membutuhkan tegangan operasional sebesar 2,3 – 3.4V. STM32F1 berfungsi sebagai pusat kendali *input*/*output* pada board flight controller.

4.1.2 Pengujian Sistem Posisi GPS Ublox M8N

Pengujian ini dilakukan untuk memastikan bahwa GPS telah bekerja dengan baik dan mengetahui tingkat akurasi dari sensor GPS Ublox M8N. pada pengujian ini , dibandingkan data GPS yang diterima sensor Ublox M8N dan data GPS yang diterima oleh Laptop. Contoh hasil pengujian dapat diliat pada Gambar 4.1 dan 4.2 yang memiliki perbedaan hasil titik koordinat latitude dan longtitude



Gambar 4. 1 Contoh hasil data GPS Ublox M8N

Gambar 4. 2 Contoh hasil data GPS Laptop

Tabel 4. 1 Perbandingan Data Hasil GPS Ublox M8N dan Laptop

Data GPS Ublox M8N		Data GPS Laptop	
Latitude	Longtitude	Latitude	Longtitude
-7.279031	112.792823	-7.278821	112.792443

Dari Tabel 4.1 didapatkan contoh perbandingan data hasil pengujian GPS Ublox M8N dengan GPS laptop. Hasil yang didapat dari pembacaan berupa data koordinat Latitude dan Longtitude. Untuk

mendapat jarak perbedaan GPS yaitu dengan menggunakan teorema Euclid, rumus yang digunakan yaitu sebagai berikut :

Jarak = $\sqrt{(Lat_1 - Lat_2)^2 + (Long_1 - Long_2)^2}$

Berdasarkan rumus diatas didapatkan perbedaan jarak sebesar :

Jarak = $\sqrt{0.00021^2 + -0.000523^2}$
= $\sqrt{4.41 - 2.735}$
= $\sqrt{1.675}$
= 1.29 m

Perbedaan data GPS yang didapatkan tidak terlalu jauh, dibuktikan dengan data GPS pada tabel 4.1, dimana perbedaan jarak sebesar 1.29m

Dengan menggunakan rumus diatas , didapatkan perbedaan jarak dalam meter dari beberapa lokasi yang ditrack yaitu seperti pada tabel 4.2 dibawah ini :

Tabel 4. 2 Perbandingan Data GPS beberapa lokasi

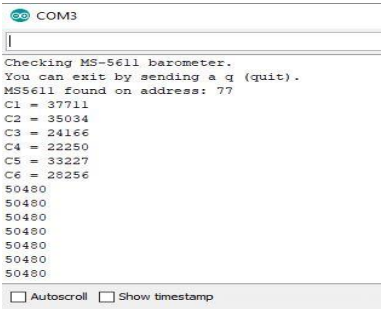
Lokasi	GPS Ublox M8N		GPS Laptop		Jarak (m)
	Latitude	Longtitude	Latitude	Longtitude	
Kontainer DTEO	-7.279059	112.792755	-7.279086	112.792758	1.32
PPNS gedung M	-7.279158	112.793098	-7.279043	112.793225	1.46
Teknik Lingkungan	-7.279345	112.792510	-7.289702	112.792721	1.57
Kantin Teknik Sipil	-7.280192	112.793991	-7.280011	112.793184	1.58
Departemen PWK	-7.279805	112.794235	-7.279698	112.794081	2.03
Mushola PWK	-7.279840	112.793960	-7.279642	112.793907	1.68
Parkir PPNS Kolam Uji	-7.279275	112.793327	-7.279160	112.793489	1.34
Mushola Amanah	-7.278626	112.792510	-7.278568	112.792532	1.21
Tek. Kimia Industri	-7.279069	112.792449	-7.278794	112.792570	1.53

Statistika Bisnis	-7.278294	112.792724	-7.278332	112.792748	1.22
Kantin Manajemen Bisnis	-7.278513	112.792396	-7.278510	112.792348	0.96
Lab Instrument	-7.278464	112.792449	-7.278255	112.792467	1.34
Himpunan Instrument	-7.278374	112.792576	-7.278330	112.792345	1.24
Elektro Otomasi	-7.279052	112.792167	-7.278763	112.792141	1.41
Manajemen Bisnis	-7.277821	112.792694	-7.277834	112.792620	0.94
			Rata-rata		1.39

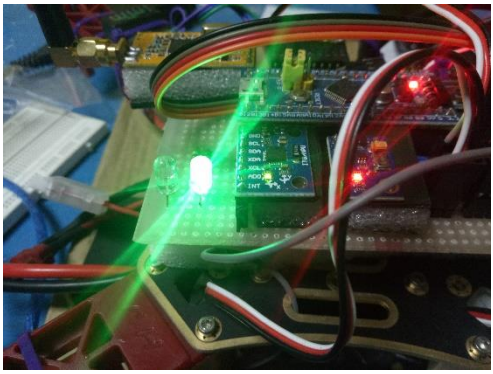
Dari data tabel di atas , dapat diketahui bahwa rata-rata perbedaan data latitude dan logtitude yang dihasilkan GPS ublox M8N dan GPS Laptop sebesar 1.39 meter.

4.1.3 Pengujian MS5611

Pada pengujian pertama, MS5611 dikalibrasi sehingga didapat data berupa tekanan udara yang dapat dilihat pada Gambar 4.3. Data tersebut ditampilkan pada serial monitor software Arduino IDE, posisi sensor MS5611 dalam keadaan tegak, tidak miring ke kanan kiri atau ke depan belakang. Setelah MPU 6050 melakukan kalibrasi, sensor MS5611 melakukan kalibrasi yang ditandai dengan indikator led hijau menyala berkedip seperti pada Gambar 4.4



Gambar 4. 3 Data MS5611



Gambar 4. 4 Indikator Led Hijau Menyala

4.1.4 Pengujian Kompas pada GPS Ublox M8N

GPS Ublox M8N telah dilengkapi dengan kompas HMC5883L, pengujian kompas pada proyek akhir ini dibandingkan perbedaan data derajat mata angin utara, selatan, timur, dan barat yang dihasilkan oleh kompas GPS Ublox M8N dengan kompas pada handphone.

Gambar 4. 5 Utara Hasil Data Kompas Handphone

Gambar 4. 6 Utara Hasil Data Kompas HMC5883L

Gambar 4. 7 Selatan Hasil Data Kompas Handphone

Gambar 4. 8 Selatan Hasil Data Kompas HMC5883L

Gambar 4. 9 Timur Hasil Data Kompas Handphone

Gambar 4. 10 Timur Hasil Data GPS HMC5883L

Gambar 4. 11 Barat Hasil Data GPS Handphone

Gambar 4. 12 Barat Hasil Data GPS HMC5883L

Dari kedelapan data gambar di atas , didapat data dalam tabel 4.3 berikut

Tabel 4. 3 Perbandingan Data Kompas Handphone dan HMC5883L

Arah Mata Angin	Kompas Handphone	Kompas HMC5883L	% *error*
Utara	360°	359°	0.2%
Selatan	180°	176°	2%
Barat	270°	236°	1.2%
Timur	90°	112°	2.4 %
		Rata-Rata	1.45%

Dari data pada Tabel 4.3 diatas didapatkan perbedaan yang disebut dengan % error atau %kesalahan, %error didapatkan dengan menggunakan rumus

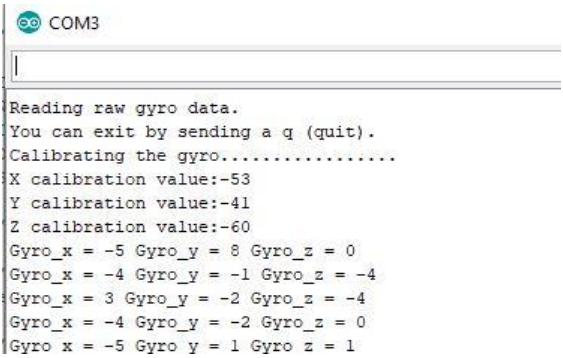
$$\frac{Nilai\ Kompas\ HMC5883L - Nilai\ Kompas\ Handphone}{Nilai\ Kompas\ Handphone}\ x\ 100\%$$

Persentase error terbesar didapatkan pada arah mata angin timur yaitu sebesar 2.4% dan persentase error terkecil didapatkan pada arah mata angin Utara yaitu sebesar 0.2% .

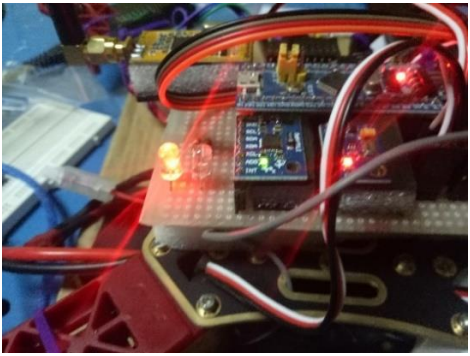
Dari data %error keseluruhan didapatkan rata-rata %error data yang dihasilkan oleh kompas HMC5883L sebesar 1.45%.

4.1.5 Pengujian MPU 6050

Pada pengujian pertama, MPU 6050 dikalibrasi sehingga didapat 2 data yaitu data gyro dan accelerometer , data gyro dapat dilihat pada Gambar 4.13 Kedua data tersebut dilihat pada serial monitor software Arduino IDE, posisi sensor MPU 6050 dalam keadaan tegak, tidak miring ke kanan kiri atau ke depan belakang. Sebelum quadcopter take off, MPU 6050 akan melakukan kalibrasi yang ditandai dengan indikator led merah menyala berkedip seperti pada Gambar 4.15



Gambar 4. 13 Data Gyro

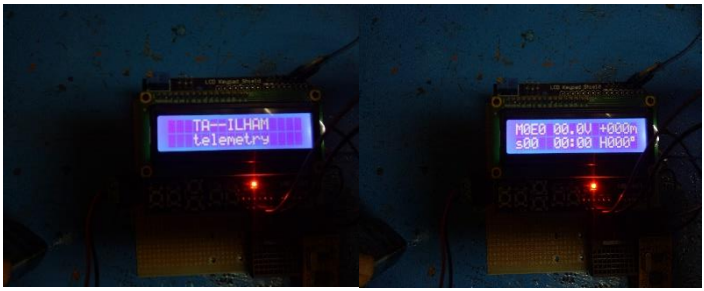


Gambar 4. 14 Indikator Led Merah Menyala

4.2 Pengujian dan Analisa pada Alat Telemetry

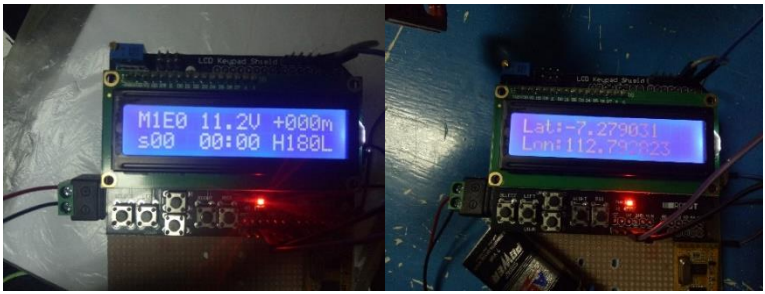
Pada pengujian ini membahas semua komponen yang digunakan dalam alat telemetry,dimana masing masing komponen dapat melakukan perintah sesuai dengan sistem keseluruhan yang di jelaskan pada *flowchart* system telemetry pada Gambar 3.14.

LCD pada alat Telemetry mempunyai tampilan awal seperti Gambar 4.15 dibawah ini



Gambar 4. 15 Tampilan Awal LCD Telemetry

Setelah Data diterima oleh modul APC220 yang terpasang pada alat telemetry , maka tampilan LCD akan berubah sesuai dengan Data yang diterima , seperti data daya Baterai Lipo , data Kompas, data Latitude Longtitude GPS , dan data ketinggian drone dapat dilihat pada Gambar 4.16



Gambar 4. 16 Data pada Telemetry

4.3 Hasil Perancangan Quadcopter

Hasil perancangan *quadcopter* secara keseluruhan harus sesuai dengan desain yang dibuat baik secara dimensi maupun peletakan komponen yang dibutuhkan sistem , sehingga setiap sistem di dalam *quadcopter* dapat bekerja dengan baik. Berikut adalah gambar hasil perancangan quadcopter jika dilihat dari sisi atas dan bawah.

Gambar 4. 17 Hasil *Quadcopter* Tampak Atas

Gambar 4. 18 Hasil Quadcopter Tampak Bawah

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

BAB V
PENUTUP

Dari hasil yang telah didapatkan selama proses dan pembuatan serta proses analisis data untuk Proyek Akhir ini, maka dapat diambil beberapa kesimpulan dan saran yang berguna untuk perbaikan dan pengembangan agar nantinya bisa bermanfaat.

5.1 Kesimpulan

Dari hasil penelitian dan pembahasan dapat disimpulkan bahwa:

1. Flight Controller pada quadcopter dapat dioperasikan dengan STM32F1 sebagai pusat kendali rangkaian dan diprogram menggunakan *software* Arduino IDE.
2. Data koordinat latitude dan longtitude GPS yang didapat tidak sepenuhnya presisi, terdapat selisih jarak dengan titik koordinat asli dengan rata-rata selisih sebesar 1.39 meter
3. Data Kompas dari HMC5833L memiliki rata-rata persentase error sebesar 1.45% jika dibandingan dengan hasil kompas pada handphone.
4. Data pada telemetry berhasil diterima meskipun relatif lambat dalam penerimaan dari transmitter.
5. APC220 sering lost koneksi ketika telemetry menyala dalam beberapa menit.

5.2 Saran

Dari hasil penelitian dan pembahasan dapat diketahui tambahan saran yaitu :

1. Modul wireless untuk pengiriman data telemetry dapat diganti dengan yang lebih baik agar pengiriman data tidak terganggu atau terputus .
2. Pada saat pengambilan data latitude dan longtitude GPS sebaiknya dilakukan di tempat terbuka dan tidak ada logam yang dapat mengganggu sinyal GPS.
3. Pada saat pengambilan data kompas sebaiknya dilakukan pada saat angin tidak bertiup kencang yang menyebabkan kompas susah menentukan arah

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

DAFTAR PUSTAKA

[1] Hardana S.Kom, *Dasar-dasar Mikrokontroler Arsitektur ARM ST Microelectronics*. Dewi Pustaka,Mojokerto, 2018.

[2] Wishnu E, *GPS pada Android*. Jasakom,Jakarta, 2012.

[3] Capt. H. R. Soebekti, S.M,Mar ,"Pedoman Gyro (teori) Otomat Pengemudian", *Buku Ajar* , Akademi Maritim Djadajat, Jakarta, 2018.

[4] Carosena Meola dan Simone BChang Liang Xiaoccardi, "*Permanent Magnet Brushless DC Motor Drives and Controls", Singapore*, 2012.

[5] Handayani Saptaji , *Mudah Belajar Mikrokontroler dengan Arduino*, Widya Media, Yogyakarta,2015.

[6] Istiyanto, J, E, *Pengantar Elektronika & Instrumentasi (Pendekatan Project Arduino dan Android) Edisi Pertama*, Andi, Yogyakarta, 2014.

[7] Fajar Wicaksono, "*Mudah Belajar Mikrokontroller Arduino* ", Informatika, Bandung, 2017.

[8] Vechian, *"Control Quadcopter With Stereo Camera and Self-Balancing System. Dissertation", Electronics Engineering, Universiti Tun Hussein Onn Malaysia.2012.*

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

LAMPIRAN A

Lampiran Program Telemetry

```
#include <LiquidCrystal.h>
#include <EEPROM.h>


uint8_t receive_buffer[50], receive_buffer_counter, receive_byte_previous, receive_start_detect;
uint8_t check_byte, temp_byte;
uint8_t error, alarm_sound, flight_mode;
uint8_t telemetry_lost;
uint8_t alarm_silence;
uint8_t start, flight_timer_start;
uint8_t hours,minutes, seconds;
uint8_t heading_lock;
uint8_t number_used_sats;
uint8_t fix_type;

int8_t page, previous_page;

uint32_t last_receive, next_sound, flight_timer, flight_timer_previous, flight_timer_from_start, flight_time_from_eeprom;
uint32_t hours_flight_time, minutes_flight_time, seconds_flight_time;
int32_t l_lat_gps, l_lon_gps;

int16_t temperature, button_push, button_store, roll_angle, pitch_angle;
int16_t altitude_meters, max_altitude_meters, max_altitude_from_eeprom;
uint16_t key_press_timer;
int16_t takeoff_throttle;
uint16_t actual_compass_heading;

float battery_voltage, adjustable_setting_1, adjustable_setting_2, adjustable_setting_3;

byte led;
```

```
void setup() {
  .
  TCCR2A = 0;
  TCCR2B = 0;
  TIMSK2 |= (1 << OCIE2A);
  TCCR2B |= (1 << CS22|1 << CS21|1 << CS20);
  OCR2A = 249;
  TCCR2A |= (1 << WGM21);


  Serial.begin(9600);
  pinMode(2, OUTPUT);

  lcd.begin(16, 2);
  lcd.setCursor(3, 0);
  lcd.print("TA--ILHAM");
  lcd.setCursor(3, 1);
  lcd.print("telemetry");

  digitalWrite(2, HIGH);
  delay(10);
  digitalWrite(2, LOW);
  delay(50);
  digitalWrite(2, HIGH);
  delay(10);
  digitalWrite(2, LOW);
  delay(2500);

  button_store = -1;
  lcd.clear();
  telemetry_lost = 2;
  flight_time_from_eeprom = (uint32_t)EEPROM.read(0x00)<< 24 |
(uint32_t)EEPROM.read(0x01)<< 16 |
(uint32_t)EEPROM.read(0x02)<< 8 | (uint32_t)EEPROM.read(0x03);
  max_altitude_from_eeprom = EEPROM.read(0x04)<< 8 |
EEPROM.read(0x05);
}
```

```
void loop() {
  if(key_press_timer > 0){
    key_press_timer = 0;
    button_store = -1;


  }

  if(button_store != -1){
    while(analogRea
  Serial.begin(9600);
  pinMode(2, OUTPUT);

  lcd.begin(16, 2);
  lcd.setCursor(3, 0);
  lcd.print("TA--ILHAM");
  lcd.setCursor(3, 1);
  lcd.print("telemetry");

  digitalWrite(2, HIGH);
  delay(10);
  digitalWrite(2, LOW);
  delay(50);
  digitalWrite(2, HIGH);
  delay(10);
  digitalWrite(2, LOW);
  delay(2500);

d(0) < 1000){
      delay(10);
      if(key_press_timer < 200)key_press_timer ++;
      if(key_press_timer == 200){
        digitalWrite(2, HIGH);
        delay(10);
        digitalWrite(2, LOW);
        delay(50);
        digitalWrite(2, HIGH);
        delay(10);
        digitalWrite(2, LOW);
```

```
      delay(500);


    }
   }
   if(key_press_timer < 200){
    digitalWrite(2, HIGH);
    delay(10);
    digitalWrite(2, LOW);
   }


 }

 if((telemetry_lost == 1 || alarm_sound == 1) && button_store != -1){
  if(alarm_sound == 1)alarm_sound = 2;
  if(telemetry_lost == 1)telemetry_lost = 2;
  page = 0;
  lcd.clear();
  button_store = -1;
 }

 if(button_store != -1 && key_press_timer < 200){
  if(button_store < 300 && button_store > 200)page--;
  if(button_store < 150 && button_store > 50)page
  if(button_store < 700 && button_store > 600)page=0;
  if(page > 6)page = 6;
  button_store = -1;


 }

 if(start > 1 && flight_timer_start == 0){
   flight_time_from_eeprom = (uint32_t)EEPROM.read(0x00)<< 24 |
(uint32_t)EEPROM.read(0x01)<<                16                |
(uint32_t)EEPROM.read(0x02)<< 8 | (uint32_t)EEPROM.read(0x03);
   flight_timer_from_start = millis();
   flight_timer = millis() - flight_timer_previous;
```

```
    flight_timer_start = 1;
  }

  if(start == 0 && flight_timer_start == 1){
    if(max_altitude_meters > max_altitude_from_eeprom){
      max_altitude_from_eeprom = max_altitude_meters;
      EEPROM.write(0x04, max_altitude_from_eeprom >> 8);
      EEPROM.write(0x05, max_altitude_from_eeprom);
    }



    flight_time_from_eeprom += (millis() -
flight_timer_from_start)/1000;
    EEPROM.write(0x00, flight_time_from_eeprom >> 24);
    EEPROM.write(0x01, flight_time_from_eeprom >> 16);
    EEPROM.write(0x02, flight_time_from_eeprom >> 8);
    EEPROM.write(0x03, flight_time_from_eeprom);



    flight_timer_previous = millis() - flight_timer;
    flight_timer_start = 0;
  }

  while(Serial.available()){
    receive_buffer[receive_buffer_counter] = Serial.read();
//Load them in the received_buffer array.
    //Search for the start signature in the received data stream.
    if(receive_byte_previous == 'J' &&
receive_buffer[receive_buffer_counter] == 'B'){
      receive_buffer_counter = 0;
      receive_start_detect ++;                    .
      if(receive_start_detect >= 2)get_data();
    }
    else{
      receive_byte_previous = receive_buffer[receive_buffer_counter];
      receive_buffer_counter ++;
```

```
    if(receive_buffer_counter > 48)receive_buffer_counter = 0;
   }
  }

  if(start > 1){
   minutes = (millis() - flight_timer)/60000;
   seconds = ((millis() - flight_timer)-minutes*60000)/1000;
  }

  if(page != previous_page){
   previous_page = page;
   lcd.clear();
  }

  if(page == 0){
   lcd.setCursor(0, 0);
   lcd.print("M");
   lcd.print(flight_mode);

   lcd.setCursor(5, 0);
   if(battery_voltage < 10)lcd.print("0");
   lcd.print(battery_voltage,1);
   lcd.print("V ");

   lcd.setCursor(11, 0);
   if(altitude_meters < 0)lcd.print("-");
   else lcd.print("+");
   if(altitude_meters < 100)lcd.print("0");
   if(altitude_meters < 10)lcd.print("0");
   lcd.print(abs(altitude_meters));
   lcd.print("m");

   lcd.setCursor(2, 0);
   lcd.print("E");
   lcd.print(error);

   lcd.setCursor(0, 1);
   if(fix_type == 3)lcd.print("S");
   else lcd.print("s");
```

```
  if(number_used_sats < 10)lcd.print("0");
  lcd.print(number_used_sats);

  lcd.setCursor(5, 1);
  if(minutes < 10)lcd.print("0");
  lcd.print(minutes);
  lcd.print(":");
  if(seconds < 10)lcd.print("0");
  lcd.print(seconds);

  lcd.setCursor(11, 1);
  lcd.print("H");
  if(actual_compass_heading < 100)lcd.print("0");
  if(actual_compass_heading < 10)lcd.print("0");
  lcd.print(actual_compass_heading);
  if(heading_lock)lcd.print("L");
  else lcd.print((char)223);
}

if(page == 1){
  lcd.setCursor(0, 0);
  lcd.print("Lat:");
  lcd.print(l_lat_gps);
  lcd.setCursor(0, 1);
  lcd.print("Lon:");
  lcd.print(l_lon_gps);
}

if(page == 2){
  if(key_press_timer == 200){
    button_store = -1;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Reset max alt?");
    lcd.setCursor(0, 1);
    lcd.print("Select = yes");
    while(button_store == -1)delay(10);
    if(button_store < 700 && button_store > 600){
      while(analogRead(0) < 1000)delay(10);
```

```
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Max alt is reset");
      EEPROM.write(0x04, 0x00);
      EEPROM.write(0x05, 0x00);
      max_altitude_from_eeprom = EEPROM.read(0x04)<< 8 |
EEPROM.read(0x05);
      delay(2000);
     }
     lcd.clear();
   }
   else{
     lcd.setCursor(0, 0);
     lcd.print("Max altitude:");
     lcd.setCursor(0, 1);
     if(max_altitude_meters < 100)lcd.print("0");
     if(max_altitude_meters < 10)lcd.print("0");
     lcd.print(max_altitude_meters);
     lcd.print("m     mem");
     if(max_altitude_from_eeprom < 100)lcd.print("0");
     if(max_altitude_from_eeprom < 10)lcd.print("0");
     lcd.print(max_altitude_from_eeprom);
     lcd.print("m");
   }
 }

 if(page == 3){
   lcd.setCursor(0, 0);
   lcd.print("roll: ");
   if(roll_angle >= 0)lcd.print("+");
   else lcd.print("-");
   if(roll_angle < 10 && roll_angle > -10)lcd.print("0");
   lcd.print(abs(roll_angle));
   lcd.setCursor(0, 1);
   lcd.print("pitch:");
   if(pitch_angle >= 0)lcd.print("+");
   else lcd.print("-");
   if(pitch_angle < 10 && pitch_angle > -10)lcd.print("0");
   lcd.print(abs(pitch_angle));
```

```
  }

  if(page == 4){
    if(key_press_timer == 200){
      button_store = -1;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Reset timer?");
      lcd.setCursor(0, 1);
      lcd.print("Select = yes");
      while(button_store == -1)delay(10);
      if(button_store < 700 && button_store > 600){
        while(analogRead(0) < 1000)delay(10);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Timer is reset");
        EEPROM.write(0x00, 0x00);
        EEPROM.write(0x01, 0x00);
        EEPROM.write(0x02, 0x00);
        EEPROM.write(0x03, 0x00);
        flight_time_from_eeprom = (uint32_t)EEPROM.read(0x00)<< 24
| (uint32_t)EEPROM.read(0x01)<< 16 |
(uint32_t)EEPROM.read(0x02)<< 8 | EEPROM.read(0x03);
        delay(2000);
      }
      //telemetry_lost = 2;
      lcd.clear();
    }
    else{
      lcd.setCursor(0, 0);
      lcd.print("Tot flight time");
      lcd.setCursor(0, 1);
      hours_flight_time = flight_time_from_eeprom/3600;
      minutes_flight_time = (flight_time_from_eeprom -
(hours_flight_time*3600))/60;
      seconds_flight_time = flight_time_from_eeprom -
(hours_flight_time*3600) - (minutes_flight_time*60);
      if(hours_flight_time < 10)lcd.print("0");
      lcd.print(hours_flight_time);
```

```
    lcd.print(":");
    if(minutes_flight_time < 10)lcd.print("0");
    lcd.print(minutes_flight_time);
    lcd.print(":");
    if(seconds_flight_time < 10)lcd.print("0");
    lcd.print(seconds_flight_time);
   }
  }

  if(page == 5){
   lcd.setCursor(0, 0);
   lcd.print("1:");
   if(adjustable_setting_1 < 10)lcd.print("0");
   lcd.print(adjustable_setting_1);
   lcd.setCursor(8, 0);
   lcd.print("3:");
   if(adjustable_setting_3 < 10)lcd.print("0");
   lcd.print(adjustable_setting_3);
   lcd.setCursor(0, 1);
   lcd.print("2:");
   if(adjustable_setting_2 < 10)lcd.print("0");
   lcd.print(adjustable_setting_2);

  }

  if(page == 6){
   lcd.setCursor(0, 0);
   lcd.print("Take-off thr:");
   lcd.setCursor(0, 1);
   lcd.print(takeoff_throttle);
  }

  if(page == 100){
   lcd.setCursor(0, 0);
   lcd.print(" Lost telemetry");
   lcd.setCursor(0, 1);
   lcd.print("   connection!");
   delay(1000);
  }
```

```
  if(page > 100){
    lcd.setCursor(0, 0);
    lcd.print("Error:");
    lcd.print(error);
    lcd.setCursor(0, 1);
    if(error == 1)lcd.print("Battery LOW!");
    if(error == 5)lcd.print("Loop time exc.");
    if(error == 6)lcd.print("Take-off error");
    if(error == 10)lcd.print("Take-off thr.");

  }

  if(last_receive + 3000 < millis() && receive_start_detect &&
telemetry_lost == 0 && key_press_timer < 200){
    telemetry_lost = 1;
    lcd.clear();
    receive_start_detect = 0;
    page = 100;
  }

  if(error && alarm_sound == 0){
    alarm_sound = 1;
    page = 100 + error;
  }

  if(error == 0 && alarm_sound){
    alarm_sound = 0;
    page = 0;
  }

  if((telemetry_lost == 1 || alarm_sound == 1) && next_sound <
millis()){
    digitalWrite(2, HIGH);
    delay(10);
    digitalWrite(2, LOW);
    delay(50);
    digitalWrite(2, HIGH);
    delay(10);
    digitalWrite(2, LOW);
```

```
    next_sound = millis() + 1000;
  }
}


ISR(TIMER2_COMPA_vect){
  if(button_store == -1)button_store = analogRead(0);
  if(button_store > 1000)button_store = -1;
}


.
void get_data(void){
  check_byte = 0;                                                    .
  for(temp_byte=0;temp_byte <= 30; temp_byte ++)check_byte ^= receive_buffer[temp_byte];  //Calculate the check_byte.
  if(check_byte == receive_buffer[31]){
    if(telemetry_lost > 0){
      telemetry_lost = 0;
      page = 0;
    }
    last_receive = millis();
    receive_start_detect = 1;
    //In the following lines the different variables are restored from the valid data stream.
    //The name of the variables are the same as in the YMFC-32 flight controller program.
    error = receive_buffer[0];
    flight_mode = receive_buffer[1];
    battery_voltage = (float)receive_buffer[2]/10.0;
    temperature = receive_buffer[3] | receive_buffer[4] << 8;
    roll_angle = receive_buffer[5] - 100;
    pitch_angle = receive_buffer[6] - 100;
    start = receive_buffer[7];
    altitude_meters = (receive_buffer[8] | receive_buffer[9] << 8) - 1000;
    if(altitude_meters > max_altitude_meters)max_altitude_meters = altitude_meters;
    takeoff_throttle = receive_buffer[10] | receive_buffer[11] << 8;
    actual_compass_heading = receive_buffer[12] | receive_buffer[13] << 8;
```

```
    heading_lock = receive_buffer[14];
    number_used_sats = receive_buffer[15];
    fix_type = receive_buffer[16];
    l_lat_gps = (int32_t)receive_buffer[17] | (int32_t)receive_buffer[18]
<< 8 | (int32_t)receive_buffer[19] << 16 | (int32_t)receive_buffer[20]
<< 24;
    l_lon_gps = (int32_t)receive_buffer[21] | (int32_t)receive_buffer[22]
<< 8 | (int32_t)receive_buffer[23] << 16 | (int32_t)receive_buffer[24]
<< 24;
    adjustable_setting_1 = (float)(receive_buffer[25] |
receive_buffer[26] << 8)/100.0;
    adjustable_setting_2 = (float)(receive_buffer[27] |
receive_buffer[28] << 8)/100.0;
    adjustable_setting_3 = (float)(receive_buffer[29] |
receive_buffer[30] << 8)/100.0;
   }
}

  if(page > 100){
    lcd.setCursor(0, 0);
    lcd.print("Error:");
    lcd.print(error);
    lcd.setCursor(0, 1);
    if(error == 1)lcd.print("Battery LOW!");
    if(error == 5)lcd.print("Loop time exc.");
    if(error == 6)lcd.print("Take-off error");
    if(error == 10)lcd.print("Take-off thr.");

  }

  if(last_receive + 3000 < millis() && receive_start_detect &&
telemetry_lost == 0 && key_press_timer < 200){
    telemetry_lost = 1;
    lcd.clear();
    receive_start_detect = 0;
    page = 100;
  }

  if(error && alarm_sound == 0){
```

```
    alarm_sound = 1;
    page = 100 + error;
  }

  if(error == 0 && alarm_sound){
    alarm_sound = 0;
    page = 0;
  }

  if((telemetry_lost == 1 || alarm_sound == 1) && next_sound <
millis()){
    digitalWrite(2, HIGH);
    delay(10);
    digitalWrite(2, LOW);
    delay(50);
    digitalWrite(2, HIGH);
    delay(10);
    digitalWrite(2, LOW);
```

LAMPIRAN B

Lampiran Program Flight Controller

```
#include <Wire.h>

HardWire HWire(2, I2C_FAST_MODE);
TwoWire HWire (2, I2C_FAST_MODE);


int16_t = signed 16 bit integer
uint8_t disable_throttle, flip32;
uint8_t error;
uint32_t loop_timer;
float angle_roll_acc, angle_pitch_acc, angle_pitch, angle_roll;
float battery_voltage;
int16_t loop_counter;
uint8_t data, start, warning;
int16_t acc_axis[4], gyro_axis[4], temperature;
int32_t gyro_axis_cal[4], acc_axis_cal[4];


int32_t cal_int;
int32_t channel_1_start, channel_1;
int32_t channel_2_start, channel_2;
int32_t channel_3_start, channel_3;
int32_t channel_4_start, channel_4;
int32_t channel_5_start, channel_5;
int32_t channel_6_start, channel_6;
int32_t measured_time, measured_time_start;
uint8_t channel_select_counter;


uint16_t C[7];
uint8_t barometer_counter, temperature_counter;
int64_t OFF, OFF_C2, SENS, SENS_C1, P;
uint32_t raw_pressure, raw_temperature, temp;
```

```
float actual_pressure, actual_pressure_slow, actual_pressure_fast,
actual_pressure_diff;
float ground_pressure, altutude_hold_pressure;
int32_t dT, dT_C5;



int16_t compass_x, compass_y, compass_z;

uint8_t gyro_address = 0x68;
uint8_t MS5611_address = 0x77;
uint8_t compass_address = 0x1E;


void setup() {
  pinMode(4, INPUT_ANALOG);

.
  afio_cfg_debug_ports(AFIO_DEBUG_SW_ONLY);

  pinMode(PB3, INPUT);
  pinMode(PB4, INPUT);
  if (digitalRead(PB3) && digitalRead(PB4))flip32 = 1;
  else flip32 = 0;
  flip32 = 0;


  pinMode(PB3, OUTPUT
  pinMode(PB4, OUTPUT);

  green_led(LOW);
  red_led(LOW);

  Serial.begin(57600);
  delay(100);
  timer_setup
  delay(50);

  HWire.begin();
```

```
  HWire.beginTransmission(gyro_address);
  HWire.write(0x6B
  HWire.write(0x00);
  HWire.endTransmission();


  HWire.beginTransmission(gyro_address);
  HWire.write(0x1B);
  HWire.write(0x08);
  HWire.endTransmission();


  HWire.beginTransmission(gyro_address);
  HWire.write(0x1C);
  HWire.write(0x10);
  HWire.endTransmission();


  HWire.beginTransmission(gyro_address);
  HWire.write(0x1A);
  HWire.write(0x03);
  HWire.endTransmission();



  print_intro();
}

void loop() {
  delay(10);

  if (Serial.available() > 0) {
    data = Serial.read();                                .
    delay(100);
    while (Serial.available() > 0)loop_counter = Serial.read();

    disable_throttle = 1;
  }
```

```
if (!disable_throttle) {
  TIMER4_BASE->CCR1 = channel_3;
  TIMER4_BASE->CCR2 = channel_3;
  TIMER4_BASE->CCR3 = channel_3;
  TIMER4_BASE->CCR4 = channel_3;
}


else {
  TIMER4_BASE->CCR2 = 1000;
  TIMER4_BASE->CCR3 = 1000;
  TIMER4_BASE->CCR4 = 1000;
}



if (data == 'a') {
  Serial.println(F("Reading receiver input pulses."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
  reading_receiver_signals();
}



if (data == 'b') {
  Serial.println(F("Starting the I2C scanner."));
  i2c_scanner();
}



if (data == 'c') {
  Serial.println(F("Reading raw gyro data."));
  Serial.println(F("You can exit by sending a q (quit)."));
  read_gyro_values();
}
```

```
if (data == 'd') {
  Serial.println(F("Reading the raw accelerometer data."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
  read_gyro_values();
}



if (data == 'e') {
  Serial.println(F("Reading the IMU angles."));
  Serial.println(F("You can exit by sending a q (quit)."));
  check_imu_angles();
}



if (data == 'f') {
  Serial.println(F("Test the LEDs."));
  test_leds();
}



if (data == 'g') {
  Serial.println(F("Reading the battery voltage."));
  Serial.println(F("You can exit by sending a q (quit)."));
  check_battery_voltage();
}


if (data == 'h') {
  Serial.println(F("Checking MS-5611 barometer."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
```

```
  check_barometer();
 }


 if (data == 'i') {
  Serial.println(F("Checking raw GPS data."));
  check_gps();
 }


 if (data == 'j') {
  Serial.println(F("Checking HMC5883L compass."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
  check_compass();
 }

 if (data == '1') {
  Serial.println(F("Check motor 1 (front right, counter clockwise
direction)."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
  check_motor_vibrations();
 }


 if (data == '2') {
  Serial.println(F("Check motor 2 (rear right, clockwise direction)."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
  check_motor_vibrations();
 }


 if (data == '3') {
  Serial.println(F("Check motor 3 (rear left, counter clockwise
direction)."));
  Serial.println(F("You can exit by sending a q (quit)."));
  delay(2500);
```

```
    check_motor_vibrations();
   }


   if (data == '4') {
    Serial.println(F("Check motor 4 (front lefft, clockwise direction)."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
   }


   if (data == '5') {
    Serial.println(F("Check motor all motors."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
   }
}


void gyro_signalen(void) {
  //Read the MPU-6050 data.
  HWire.beginTransmission(gyro_address);
  HWire.write(0x3B);                                .
  HWire.endTransmission();

  acc_axis[1] = HWire.read() << 8 | HWire.read();
  acc_axis[2] = HWire.read() << 8 | HWire.read();
  acc_axis[3] = HWire.read() << 8 | HWire.read();


 low and high byte to the temperature variable.
  gyro_axis[1] = HWire.read() << 8 | HWire.read();
  gyro_axis[2] = HWire.read() << 8 | HWire.read();
  gyro_axis[3] = HWire.read() << 8 | HWire.read();
  gyro_axis[2] *= -1;
  gyro_axis[3] *= -1;
```

```
  if (data == 'j') {
   Serial.println(F("Checking HMC5883L compass."));
   Serial.println(F("You can exit by sending a q (quit)."));
   delay(2500);
   check_compass();
  }

  if (data == '1') {
   Serial.println(F("Check motor 1 (front right, counter clockwise direction)."));
   Serial.println(F("You can exit by sending a q (quit)."));
   delay(2500);
   check_motor_vibrations();
  }


  if (data == '2') {
   Serial.println(F("Check motor 2 (rear right, clockwise direction)."));
   Serial.println(F("You can exit by sending a q (quit)."));
   delay(2500);
   check_motor_vibrations();
  }


  if (data == '3') {
   Serial.println(F("Check motor 3 (rear left, counter clockwise direction)."));
   Serial.println(F("You can exit by sending a q (quit)."));
   delay(2500);
 if (cal_int >= 2000) {

   gyro_axis[1] -= gyro_axis_cal[1];
   gyro_axis[2] -= gyro_axis_cal[2];
   gyro_axis[3] -= gyro_axis_cal[3];
  }
}
  void red_led(int8_t level) {
   if (flip32)digitalWrite(PB4, !level);
   else digitalWrite(PB4, level);
```

```
  }
  void green_led(int8_t level) {
    if (flip32)digitalWrite(PB3, !level);
    else digitalWrite(PB3, level);
  }
    digitalWrite(Relay, LOW);

    digitalWrite(LED_G, HIGH);

    delay(3000);   if (data == '1') {
    Serial.println(F("Check motor 1 (front right, counter clockwise
direction)."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
  }


  if (data == '2') {
    Serial.println(F("Check motor 2 (rear right, clockwise direction)."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
  }


  if (data == '3') {
    Serial.println(F("Check motor 3 (rear left, counter clockwise
direction)."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
 if (cal_int >= 2000) {

    gyro_axis[1] -= gyro_axis_cal[1];
    gyro_axis[2] -= gyro_axis_cal[2];
    gyro_axis[3] -= gyro_axis_cal[3];
  } println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
```

```
  }


  if (data == '4') {
    Serial.println(F("Check motor 4 (front lefft, clockwise direction)."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
  }


  if (data == '5') {
    Serial.println(F("Check motor all motors."));
    Serial.println(F("You can exit by sending a q (quit)."));
    delay(2500);
    check_motor_vibrations();
  }
}


void gyro_signalen(void) {
  //Read the MPU-6050 data.
  HWire.beginTransmission(gyro_address);
  HWire.write(0x3B);                                    .
  HWire.endTransmission();

  acc_axis[1] = HWire.read() << 8 | HWire.read();
  acc_axis[2] = HWire.read() << 8 | HWire.read();
  acc_axis[3] = HWire.read() << 8 | HWire.read();
}
```

DAFTAR RIWAYAT PENULIS

Nama	:	Ilham Dwiki Ramadhan
TTL	:	Malang, 31 Desember 1998
Jenis Kelamin	:	Laki-laki
Agama	:	Islam
Alamat	:	Plaosan Permai D-40 , Blimbing Malang
Telp/HP	:	0856-0413-0696
E-mail	:	ilhamdwikira14@gmail.com

RIWAYAT PENDIDIKAN

1. 2004 – 2010 : SDN Blimbing 3 Malang
2. 2010 – 2013 : SMP Negeri 8 Malang
3. 2013 – 2016 : SMA Negeri 4 Malang
4. 2016 – 2019 : Departemen Teknik Elektro Otomasi - Fakultas Vokasi - Institut Teknologi Sepuluh Nopember (ITS)

PENGALAMAN KERJA

1. Magang di PT. Cinovasi Rekaprima Surabaya

PENGALAMAN ORGANISASI

1. Staff Kesejahteraan Mahasiswa HIMAD3TEKTRO
2. Staff ITS Expo
3. IRIS Team Divisi Elektronik