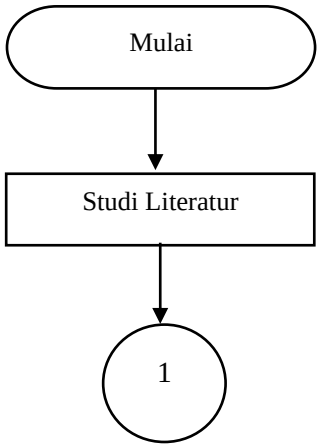
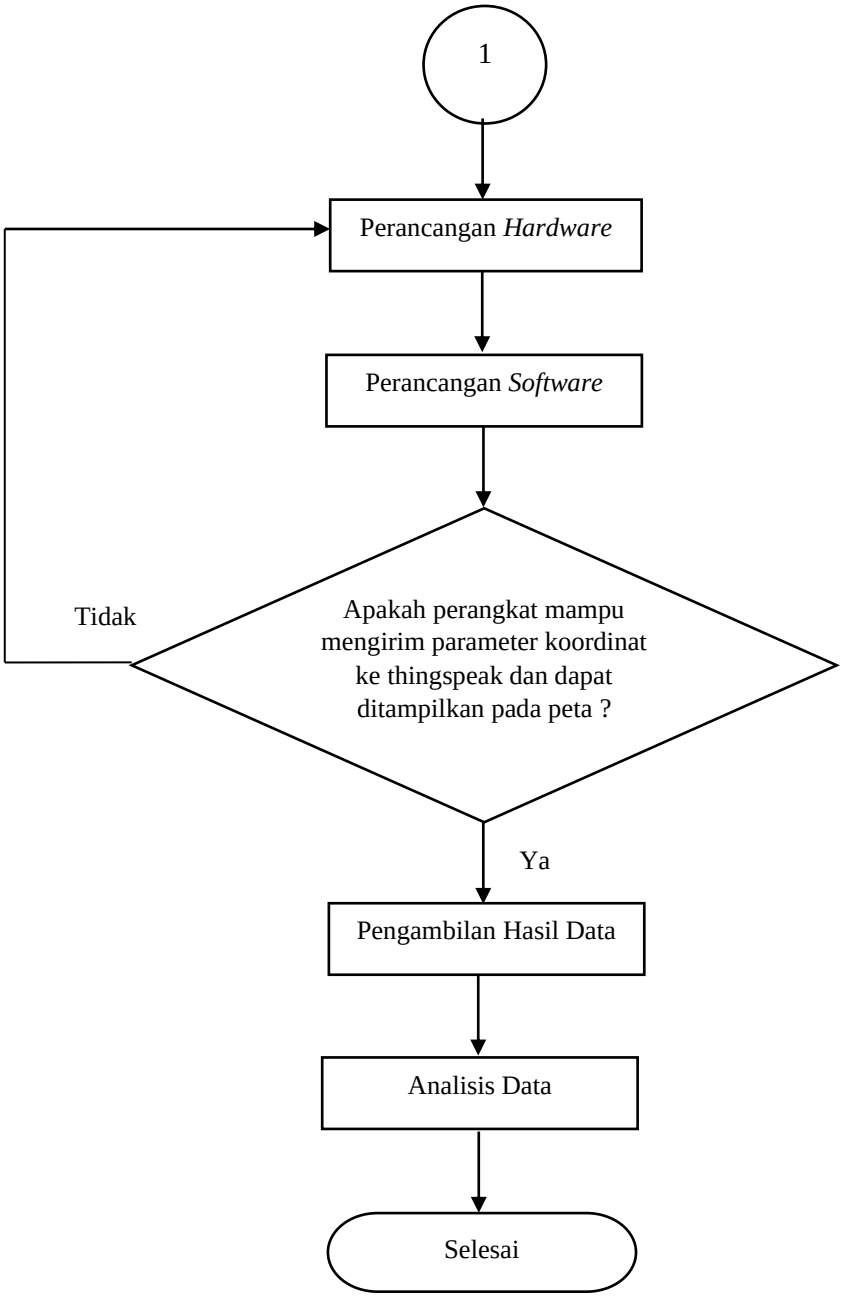
BAB 3
METODE PENELITIAN

Pada penelitian ini penulis melakukan penelitian tentang rancang bangun alat *monitoring* angkutan umum yang menggunakan metode *GPS Tracking*. GPS *Tracker* adalah alat yang dapat digunakan untuk memantau atau melacak kendaraan dengan menggunakan satelit *GPS* dalam bentuk titik kordinat yang dapat diamati secara *real-time* melalui peta digital. Berikut ini adalah metodologi penelitian yang dibutuhkan untuk Skripsi ini.

3.1 ALUR PENELITIAN

Penelitian ini dirancang dalam berbagai tahap. Dimulai dengan mencari studi literatur, membuat rancangan *hardware*, kemudian melakukan perancangan *software*, menguji sesuai parameter, dan kemudian membuat hasil data dari pengujian sistem dan menganalisis data. Alur penelitian diperlukan dalam perancangan penelitian agar rencana dapat dilaksanakan. Gambar 3.1 menunjukkan bentuk alur penelitian, yang dapat menjelaskan proses perancangan penelitian.





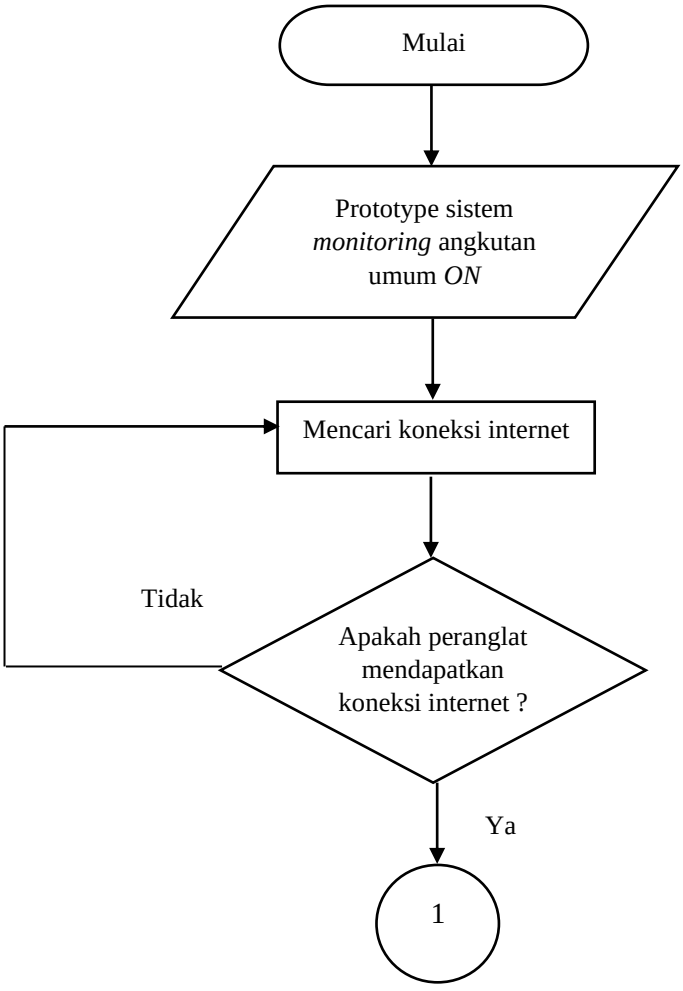
Gambar 3. 1 *Flow chart* Alur Penelitian

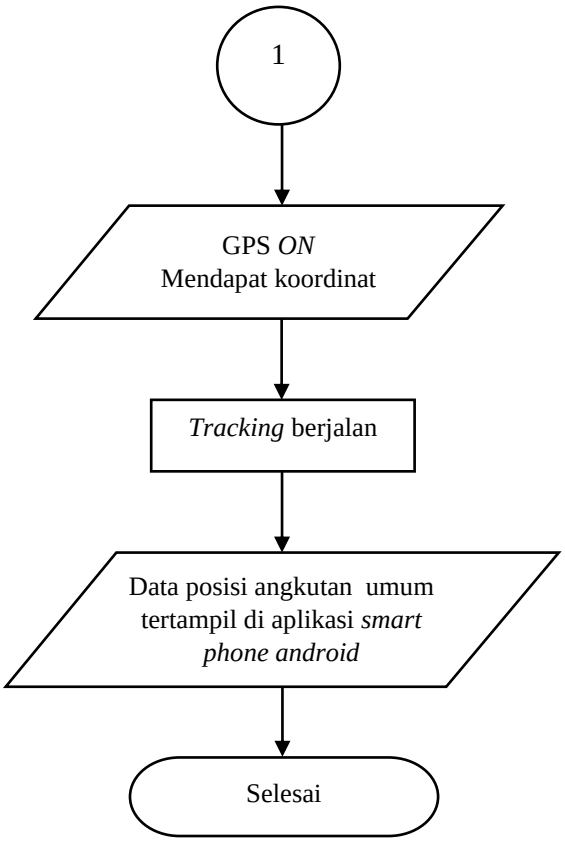
Seperti yang ditunjukkan oleh *flowchart* alur penelitian pada gambar 3.1, diawala dari pencarian studi literatur yang dengan membandingkan kajian teori dari perancangan/peneliti sebelumnya, selain itu studi literatur dilakukan dengan cara membaca beberapa artikel dari *internet*, buku, dan jurnal ilmiah yang dapat membantu memahami sistem dan cara kerja setiap perangkat. Dalam blok diagram perancangan *hardware* adalah proses pengumpulan alat dan bahan untuk menerima data *GPS* dan mendukung konektivitas GPRS untuk data masukkan

untuk *Arduino UNO* R3 yaitu pada perangkan A perangkat modul SIM808 dan perangkat pembanding B yaitu menggunakan modul *GPS* NEO 6M dengan SIM800L, perangkat *mikrokontroler Arduino UNO* R3 erfungsi sebagai pengolah data masukkan dari perangkat *GPS* NEO.

Blok diagram perancangan *software* menunjukkan proses pembuatan aplikasi yang digunakan dalam perancangan skripsi ini. Proses ini dilakukan dengan menggunakan *App Invertor* secara *online*, yang menampilkan data posisi perangkat sistem *monitoring* angkutan umum dari *server Thingspeak* . Setelah perancangan *hardware* dan *software* setiap perangkat selesai, maka selanjutnya yaitu dilakukan pengujian sesuai dengan parameter, jika pada pengujian tersebut terdapat ketidak sesuaian dengan parameter atau terdapat kesalahan maka perancangan *hardware* dan *software* akan diulang hingga pengujian berhasil. Namun kika hasilnya sesuai dengan parameter, hasil data langsung dibuat dan dilakukan analisis berdasarkan hasil data pengujian.

3.1.1 *FLOW CHART* ALUR SISTEM *GPS*





Gambar 3. 2 Flow chart Alur Sistem GPS *Tracking*

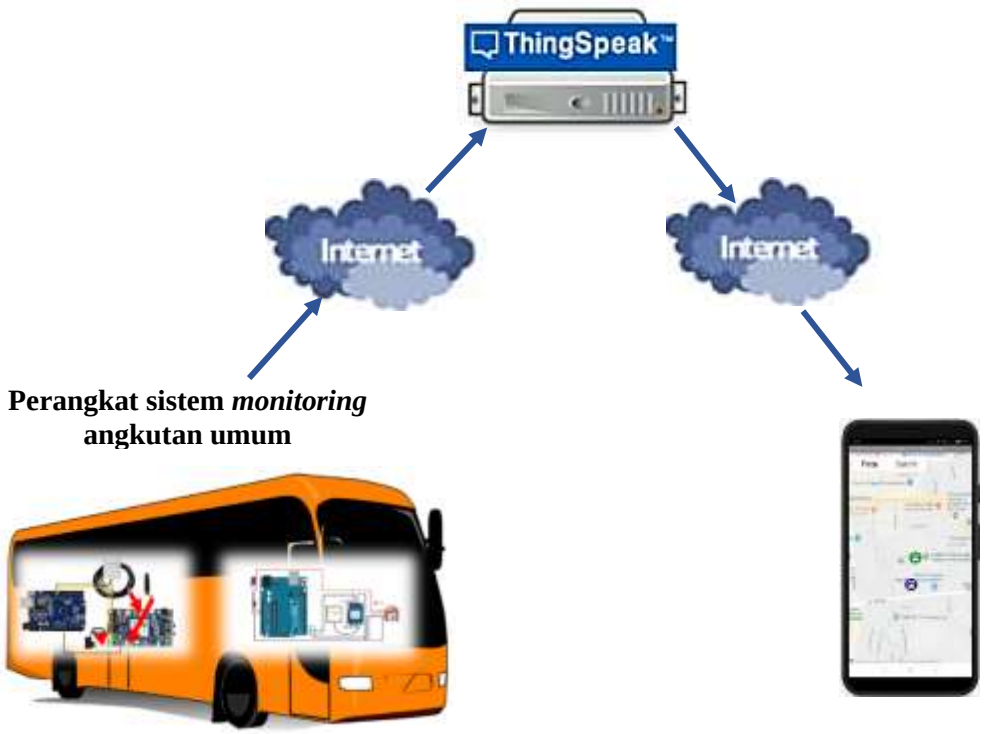
Gambar 3.2 menunjukkan diagram aliran alur sistem *GPS* Tracking, dimulai dengan kondisi sistem *monitoring* yang terpasang di angkutan umum telah aktif, kemudian sistem tersebut akan mencari koneksi *internet*. Jika tidak tersedianya koneksi *internet*, maka sistem *monitoring* akan kembali mencari koneksi *internet*. Jika tersedia koneksi *internet*, maka *GPS* yang telah aktif dan telah mendapatkan koordinat *Tracking* berjalan akan ditampilkan di bagian user aplikasi *smartphone* Android.

3.2 PERANGKAT YANG DIGUNAKAN

Pada penelitian ini perangkat yang digunakan meliputi peralatan perangkat keras yang digunakan untuk perancangan *prototype* dan juga perangkat lunak. Terdapat 2 perangkat keras *system* yang terpasang pada angkutan umum untuk pembanding keakurasian *system GPS* dalam menangkap dan mengirim

koordinat. Berdasarkan blok diagram sistem yang ditunjukkan pada gambar 3.3 perangkat keras *system* yang pertama(A) terdiri dari modul SIM808 berfungsi untuk menerima data *GPS* dan mendukung konektivitas GPRS sekaligus berfungsi sebagai data masukkan untuk *Arduino UNO* R3. Selain itu pada blok diagram sistem pada gambar 3.3 terdapat perangkat keras *system* pembanding(B) terdiri dari modul *GPS Ublox Neo* 6M berfungsi untuk menerima data *GPS* dan terhubung ke modul SIM 800L yang mendukung konektivitas GPRS yang digunakan sekaligus sebagai data masukkan untuk *Arduino UNO* R3. Kedua perangkat *system* tersebut terhubung ke perangkat *mikrokontroler Arduino UNO* R3 sebagai pengolah data input untuk modul SIM808 dan modul SIM 800L.

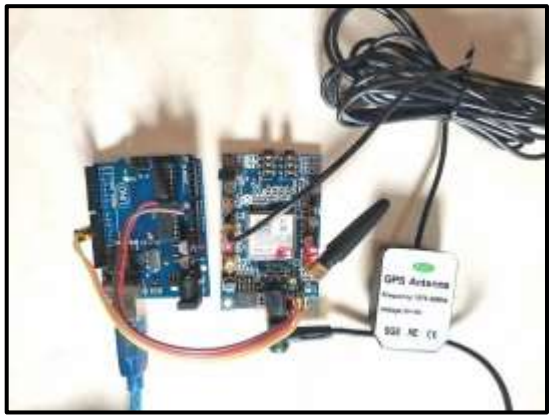
Setelah prototype dipasang di angkutan umum, data koordinat dikirim ke *server* database *Thingspeak* . Dari sisi pengguna dapat mengakses melalui aplikasi Android yang dibuat menggunakan *App Inventor*. Aplikasi tersebut akan menampilkan marker lokasi keberadaan angkutan umum.



Gambar 3. 3 Blok Sistem *Monitoring* Angkutan Umum

3.2.1 PERANCANGAN ALAT

Gambar 3.4 dan 3.5 menunjukkan beberapa peralatan yang digunakan untuk mendukung kegiatan penelitian ini. Peralatan ini akan membentuk perangkat sistem *monitoring* angkutan umum yang dirancang dengan metode *GPS*, yaitu:



Gambar 3. 4 Perangkat Sistem *Monitoring* Angkutan Umum A

Gambar 3. 5 Perangkat Sistem *Monitoring* Angkutan Umum B

3.2.1.1 *ARDUINO UNO*

Arduino UNO digunakan sebagai prosesor untuk mengelola masukan dari sensor yang digunakan. Dalam sistem *monitoring* angkutan umum, pin 2, 3, 8, dan 9 berfungsi untuk pengolah data *input* dan *output*, serta pin 5V dan pin GND atau ground. [9]

Tabel 3. 1 Keterangan Fungsi Bagian *Board Arduino UNO* yang digunakan pada perangkat *system monitoring* angkutan umum A *[9]*

Bagian *Board*	**Fungsi**
Pin 10 dan 11	digunakan sebagai input atau *output*, dan dapat diatur melalui suatu program.
USB	Digunakan sebagai: a. Memuat program ke dalam *board* dari komputer. b. Membuat komunikasi serial antara *board* dan komputer. c. Memberikan daya ke *board* .
GND	Pin *Ground*

Tabel 3. 2 Keterangan Fungsi Bagian *Board Arduino UNO* yang digunakan pada perangkat *system monitoring* angkutan umum B *[9]*

Bagian *Board*	**Fungsi**
Pin 2, 3, 8 dan 9	Digunakan sebagai input atau *output*, dapat diatur oleh program. Pin 2 terhubung ke TX Modul SIM 800L dan Pin 3 terhubung ke RX Modul SIM 800L, sedangkan Pin 8 terhubung ke TX Modul *GPS* Neo 6M dan Pin 9 terhubung ke RX Modul *GPS* Neo 6M.
USB	berfungsi sebagai: d. Memuat program ke dalam *board* dari komputer. e. Membuat komunikasi serial antara *board* dan komputer. f. Memberikan daya ke *board* .
5V	Pin untuk daya *mikrokontroler* dan komponen lainnya di *board* . Dimana Pin 5V terhubung ke VCC Modul SIM 800L dan VCC Modul *GPS* Neo 6M.
GND	Pin *Ground* terhubung ke Pin GND Modul SIM 800L dan GND Modul *GPS* Neo 6M.

3.2.1.2 MODUL SIM808

Pada sistem ini, modul SIM808 berfungsi untuk berkomunikasi melalui jaringan seluler GPRS (*Internet*), dan modul SIM808 memiliki sensor lokasi A-*GPS* (*indoor* atau *outdoor*) yang dapat berkomunikasi dengan satelit baik di dalam gedung maupun di area terbuka. Dengan demikian, modul SIM808 digunakan dalam sistem ini untuk menghitung nilai *latitude* dan *longitude*. Modul SIM808 terhubung ke *Arduino* melalui pin Rx dan Tx, dan juga melalui pin *LI-ion-*. [12]

Dengan menggunakan modul SIM808 yang memiliki fungsi *GPS*, variabel koordinat dapat dilacak dengan lancar di mana pun dan kapan pun dalam jangkauan sinyal.

Tabel 3. 3 Fungsi pin pada Modul SIM808 yang digunakan *[12]*

Pin	Fungsi
RX (atau RXD)	Terhubung ke pin 10 papan *Arduino* untuk menerima pin input data.
TX (atau TXD)	Mengirimkan pin *output* data. Terhubung ke papan *Arduino* atau pin 11
LI-ion -	pin negatif untuk baterai Li-Ion 3.5-4V DC
POWKEY	tombol mulai. tekan untuk memulai modul

3.2.1.3 LAPTOP

Laptop digunakan dalam penelitian ini untuk mengkonfigurasi mikro pengendali *Arduino UNO* dan modul *GPS*, dan untuk memantau data yang dikirimkan dari *Arduino UNO*. Dengan alat ini dapat membuat listing pemrograman, mengambil sekaligus mengolah hasil pengujian sistem dan perangkat, melihat hasil data dari *server Thingspeak* , dan membuat aplikasi untuk *smartphone* Android dengan *App Inventor*.

3.2.1.4 MODUL *GPS UBLOX NEO* 6M

Sebagai penerima *GPS*, modul *GPS Ublox Neo* 6M dapat mendeteksi lokasi dengan mengumpulkan dan memproses sinyal dari satelit navigasi. [5]

Tabel 3. 4 Fungsi pin pada Modul *GPS Ublox Neo* 6M yang digunakan *[5]*

Pin	Fungsi
RX (atau RXD)	Terhubung ke papan *Arduino* atau pin 9. Menerima pin *input* data.
TX (atau TXD)	Terhubung ke papan *Arduino* atau pin 10. Mengirimkan pin *output* data.
GND	Terhubung ke papan *Arduino* atau pin GND, pin *ground*.
VCC	Terhubung ke papan *Arduino* atau pin VCC. Tegangan Suplai.

3.2.1.5 MODUL SIM 800L

SIM 800L digunakan untuk operasi dalam jaringan seluler GPRS, atau *Internet*, dan dapat mengirim data koordinat ke *server*. [17]

Tabel 3. 5 Fungsi pin pada Modul SIM 800L yang digunakan *[17]*

Pin	Fungsi
RX	Terhubung ke papan *Arduino* atau pin 3. Menerima pin *input* data.
TX	Terhubung ke papan *Arduino* atau pin 2. Mengirimkan pin *output* data.
GND	Terhubung ke papan *Arduino* atau pin GND. pin *ground*.
VCC	Terhubung ke papan *Arduino* atau pin VCC. Tegangan Suplai.

3.2.1.6 *SMART PHONE ANDROID*

Dalam penelitian ini, *smart phone android* digunakan oleh pengguna untuk memantau angkutan umum melalui aplikasi yang diatur untuk memantau angkutan umum..

3.2.2 PERANGKAT LUNAK UNTUK PENELITAN

3.2.2.1 *ARDUINO IDE*

Pada penelitian ini menggunakan *Arduino IDE* untuk mengkonfigurasi program yang selanjutnya akan di masukan ke *Arduino*.

1. Program pada *Arduino* pada perangkat Sistem *Monitoring* Angkutan Umum A:

```
#include <SoftwareSerial.h> //library softwareserial untuk modul sim808
SoftwareSerial mySerial(10,11);   //port koneksi serial RX TX modul sim808
#define DEBUG true //debug bervalue true
String latitude, longitude, altitude, timegps, speedknot, c, state ; /*variabel latitude,
longitude, altitude, timegps, speedknot, c, state dalam bentuk string*/
```

Gambar 3. 6 Bagian *header*

Gambar 3.6 menunjukkan prosesor pengarah yang meminta compiler untuk memasukkan kode dari header file iostream.h ke dalam program. Fungsi cout memerlukan file iostream.h yang bernama <*Software*Serial.h> untuk menampilkan kode header yang melibatkan *Android SoftwareSerial* pada header baris berikutnya dari serial perangkat lunak untuk modul *GPS*/*GSM*. Pin *Arduino UNO* 10 terhubung ke TX modul SIM808, sementara pin UNO *Arduino* 11 terhubung ke RX modul SIM808. Selain itu, terdapat data string untuk *latitude*, *longitude*, altitude, time*GPS*, speedknot, c, dan state dalam program.

```
void setup()//fungsi yang hanya dilakukan sekali eksekusi diawal
{
  mySerial.begin(9600); //kecepatan transfer data dari modul sim808 ke arduino
  Serial.begin(9600);   //kecepatan transfer data dari arduino ke serial
  delay(5);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CGATT?");//memeriksa apakah perangkat telah terpasang ke GPRS.0Detch,1Attch
  delay(1000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIPSHUT");//shut packet data protocol context
  delay(1000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIPSTATUS");// mengembalikan status koneksi saat ini
  delay(2000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIPMUX=0"); //membuat koneksi multi-IP (0=single connection)
  delay(2000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CSTT=\"internet\"");//mengatur jaringan , telkomsel APN = internet
  delay(1000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIICR");//memunculkan koneksi wireless
  delay(3000);
```

Gambar 3. 7 Bagian *Void Setup*

Ketika sketsa dimulai, fungsi Setup() dipanggil, seperti yang ditunjukkan pada Gambar 3.7. Pada struktur ini, kecepatan transfer data dari modul SIM808 ke *Arduino* yaitu memiliki *baudrate* 9600, dan serial *port* yang terbuka memiliki *baudrate* 9600 untuk menampilkan data pada serial monitor. Untuk mencetak perintah membuka *GPS* dapat menggunakan fungsi *sendData*. Perintah *AT Command* kemudian digunakan untuk melakukan konfigurasi jaringan yang digunakan dan mengaktifkan konektifitas jaringan..

```
void GetGPS() // fungsi untuk mengaktifkan modul gps
{
  Serial.println(mySerial.readString());
  mySerial.println(" AT+CGPSPWR=1"); //buka GPS
  delay(4500 / 1.1);

  mySerial.println("AT+CGNSPWR=1");
  delay(100);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CGNSSEQ=RMC"); //Tentukan kalimat NMEA terakhir yang diuraikan
  delay(100);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CGPSSTATUS?"); //status gps saat ini
  delay(100);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CGNSINF"); //mengirim informasi lokasi GPS saat ini
  delay(50);

  if (mySerial.available() > 0)
  {
    while (mySerial.available() > 0)
    {
      c = (mySerial.readString());
      delay(500);
      c.remove(100);
```

Gambar 3. 8 Bagian *Void GetGPS*

```
  delay(50);

  if (mySerial.available() > 0)
  {
    while (mySerial.available() > 0)
    {
      c = (mySerial.readString());
      delay(500);
      c.remove(100);
    }
  }
  //pembagian array dengan proses parsing data
  state = c.substring(25, 26); //state = menerima data - 1, tidak menerima - 0
  timegps = c.substring(27, 41);
  latitude = c.substring(46, 55);
  longitude = c.substring(56, 66);
  altitude = c.substring(67, 74);
  speedknot = c.substring(75, 79);

  Serial.println("State      :" + state); //memasukan data state dalam variabel state
  Serial.println("Time       :" + timegps); //memasukan data time dalam variabel time
  Serial.println("Latitude   :" + latitude); //memasukan data latitude dalam variabel latitude
  Serial.println("Longitude  :" + longitude); //memasukan data longitude dalam variabel longitude
  Serial.println("Altitude   :" + altitude); //memasukan data altitude dalam variabel altitude
  Serial.println("Speed      :" + speedknot + " knot"); //memasukan data speedknot dalam variabel speed
  Serial.println(" ");
  delay(500);
}
```

Gambar 3. 9 Bagian *Void GetGPS*

Gambar 3.8 dan 3.9 menunjukkan fungsi untuk mengaktifkan *GPS* SIM808, yang kemudian akan diletakan pada *loop* Void(). Pada Void Get*GPS*(), perintah *AT Command* digunakan untuk mengaktifkan dan mendapatkan data dari SIM808. Setiap masing-masing data dari *state*, *timeGPS*, *latitude*, *longitude*, *altitude*, dan *speedknot* kemudian dibagi menjadi array dengan teknik parsing atau pembagian array.

```
void ConnectServer() //fungsi untuk konek ke server
{
  Serial.println(mySerial.readString());
  mySerial.println("AT+CIFSR");//dapatkan alamat IP lokal
  delay(2000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIPSPRT=0");//memulai koneksi TCP atau UDP
  delay(3000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");//memunculkan koneksi ke thingspeak
  delay(6000);

  Serial.println(mySerial.readString());
  mySerial.println("AT+CIPSEND");//mengirim data ke server
  delay(4000);
}
```

Gambar 3. 10 Bagian *Void ConnectServer*

Program Void Connect*Server*() ditampilkan pada gambar 3.10. Program ini menggunakan beberapa perintah *AT Command* untuk menghubungkan data ke *server* dengan protokol TCP.

Gambar 3. 11* Bagian *Void Field

Gambar 3.11 terdapat fungsi untuk memanggil *Void Field*(), yang membantu program mengirim data *latitude* dan *longitude* ke *Field* 1 dan 2 *Thingspeak* .

```
void loop() //fungsi perulangan
{
  GetGPS();//memanggil fungsi GetGPS
  ConnectServer();//memanggil fungsi ConnectServer
  Field(); //memanggil fungsi Field
}
```

Gambar 3. 12 Bagian *Void loop*

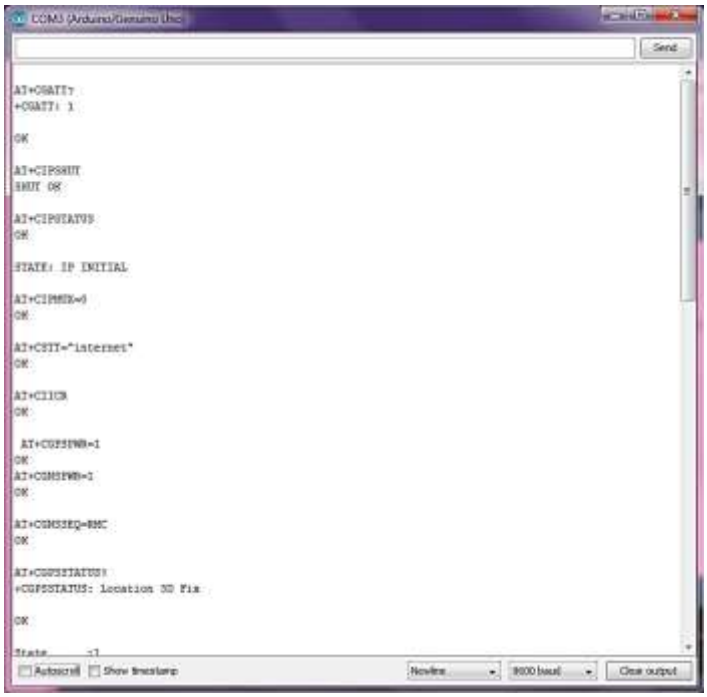
Gambar 3.12 menunjukkan fungsi *loop*() program yang berguna untuk melaksanakan atau mengeksekusi perintah program yang telah dibuat berulang-ulang. Fungsi ini menerima perintah perulangan dari fungsi *GetGPS*() untuk mendapatkan data *latitude* dan *longitude* posisi sistem *monitoring* angkutan umum. Kemudian, fungsi *ConnectServer*() digunakan untuk terhubung ke *server Thingspeak* , dan fungsi *Field*() digunakan untuk mengimpor data.

Tabel 3.4 berisi penjelasan tentang perintah AT yang digunakan pada program :

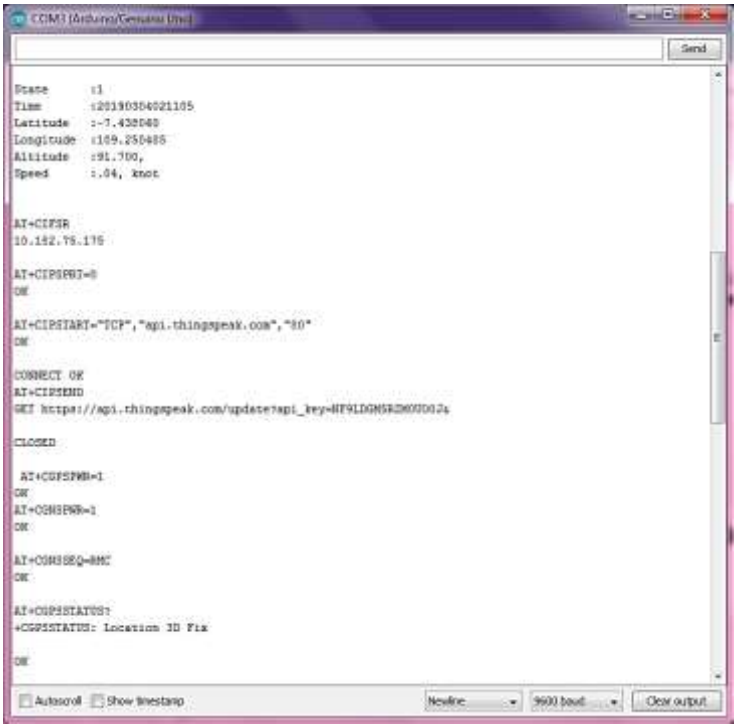
Tabel 3. 6 Perintah *AT Command* yang digunakan

Perintah *AT Command*	**Penjelasan**
AT	Memulai perintah
AT+CGATT?	Memeriksa apakah perangkat telah terpasang ke *GPRS.0Detch,1Attch*
AT+CIPSHUT	*Shut packet* data *protocol context*
AT+CIPSTATUS	Mengembalikan status koneksi saat ini
AT+CIPMUX=0	Membuat koneksi multi-IP (0=*single connection*)

Perintah *AT Command*	Penjelasan
AT+CSTT=\"*internet*\"	Mengatur jaringan , telkomsel APN = *internet*
AT+CIICR	Memunculkan koneksi *wireless*
AT+C*GPS*PWR=1	Buka *GPS*
AT+CGNSSEQ=RMC	Tentukan kalimat NMEA terakhir yang diuraikan
AT+C*GPS*STATUS?	Status *GPS* saat ini
AT+CGNSINF	mengirim informasi lokasi *GPS* saat ini
AT+CIFSR	Dapatkan alamat IP lokal
AT+CIPSPRT=0	Memulai koneksi TCP atau UDP
AT+CIPSTART=\"TCP\",\"api. *Thingspeak* .com\",\"80\"	Memunculkan koneksi ke *Thingspeak*
AT+CIPSEND	Mengirim data ke *server*



Gambar 3. 13 Display pada *serial monitor*



Gambar 3. 14 Display pada serial monitor

Ketika tampilan pada serial monitor dan pengaturan *baud rate*nya diatur pada 9600 maka akan muncul tampilan yang ditunjukkan pada gambar 3.13 dan 3.14. Pada gambar 3.13 dan 3.14 dapat menampilkan aktifitas *AT Command* yang telah terdapat pada program dan juga muncul data *state*, time, *latitude*, *longitude*, altitude, dan *speed* dari perangkat sistem *monitoring* angkutan umum.

2. Program pada *Arduino* pada perangkat Sistem *Monitoring* Angkutan Umum B:

```
#include <SoftwareSerial.h> //library softwareserial untuk modul sim800L
#include <AltSoftSerial.h> //library AltSoftSerial untuk modul sim800L
#include "TinyGPS++.h" //library GPS untuk menjalankan modul GPS ublox neo-6m
TinyGPSPlus gps;
AltSoftSerial mygps;


SoftwareSerial SIM800L(2,3);//Serial SIM800L RXPin, TXPin pin
String Write_API_key = "L4AB8XF604F4DC39";  //Thingspeak  API Key
String apn = "telkomsel";
String Latitude;
String Longitude;
```

Gambar 3. 15 Bagian *header*

Gambar 3.15 menunjukkan prosesor pengarah yang meminta compiler untuk memasukkan kode dari header file *iostream.h* ke dalam program.

Fungsi *cout* membutuhkan file *iostream.h* yang berupa <*Software*Serial.h>, <Alt*Software*Serial>, dan "Tiny*GPS*++.h" yang menunjukan kode *header* melibatkan *Android*. Kemudian serial perangkat lunak untuk modul SIM800L ditunjukkan pada header baris berikutnya dimana pin 2 *Arduino UNO* terhubung ke RX modul SIM800L dan pin 3 *Arduino UNO* terhubung ke TX modul SIM800L. Selain itu, pada program terdapat string data untuk variable api key *Thingspeak* , apn, *latitude*, dan *longitude* .

```
void setup() //fungsi yang hanya dilakukan sekali eksekusi diawal
{
  Serial.begin(115200);
  SIM800L.begin(115200);
  mygps.begin(9600);
  Serial.println("SIM800L GPRS Test");
  delay(2000);
}
```

Gambar 3. 16 Bagian *Void Setup*

Gambar 3.16 menunjukkan bahwa fungsi Setup() dipanggil ketika sketsa dimulai dan dilakukan hanya sekali ketika eksekusi dimulai. Pada struktur ini, kecepatan transfer data dari modul SIM800L ke *Arduino* adalah 115200 baud, dan kecepatan transfer data dari modul *GPS* Neo ke *Arduino* adalah 9600 baud, yang dapat ditampilkan di serial monitor.

Gambar 3. 17 Bagian *Void loop*

Untuk melaksanakan atau mengeksekusi perintah program yang sudah dibuat berulang-ulang maka fungsi *loop*() digunakan pada program, seperti yang ditunjukkan pada gambar 3.17. Fungsi ini melakukan perintah perulangan dari pemanggilan ketika *GPS* tersedia, dan beberapa perintah AT

digunakan untuk menghubungkan data ke *server* dengan protokol TCP. Selanjutnya, program mengirim data *latitude* dan *longitude* ke *Field* 3 dan *Field* 4 menggunakan protokol TCP.

```
void ReadSensor()
{
  Latitude = String(gps.location.lat(),6);
  Longitude = String(gps.location.lng(),6);
  delay(500);
  Serial.println();
  Serial.print("Lat = ");Serial.print(Latitude);
  Serial.print("\t");
  Serial.print("Lng = ");Serial.print(Longitude);

}
```

Gambar 3. 18 Bagian *ReadSensor()*

Pada gambar 3.18 merupakan fungsi *Void ReadSensor*() yaitu untuk pembacaan sensor *GPS* dari modul *GPS* Neo 6M yang nantinya akan diletakan pada *Void loop*().

```
void SetupModule()
{
  if (SIM800L.available())Serial.write(SIM800L.read());
  SIM800L.println("AT"); delay(1000);
  SIM800L.println("AT+CPIN?"); delay(1000);
  SIM800L.println("AT+CREG?"); delay(1000);
  SIM800L.println("AT+CGATT?"); delay(1000);
  SIM800L.println("AT+CIPSHUT");delay(1000);
  SIM800L.println("AT+CIPSTATUS"); delay(2000);
  SIM800L.println("AT+CIPMUX=0");  delay(2000);

 //setting the APN,
  SIM800L.println("AT+CSTT=\""+apn+"\"");delay(1000);
  ShowSerialData();
  SIM800L.println("AT+CIICR");    delay(2000);
  ShowSerialData();

  //get local IP adress
  SIM800L.println("AT+CIFSR");    delay(2000);
  ShowSerialData();

  SIM800L.println("AT+CIPSPRT=0");delay(2000);
  ShowSerialData();
}
```

Gambar 3. 19 Bagian *Void Setup*

Gambar 3.19 menunjukan fungsi pengaturan perangkat SIM800L dengan perintah *AT Command* yang memungkinkan untuk melakukan konfigurasi jaringan, dan juga perintah *AT Command* pengaktifan konektifitas nirkabel.

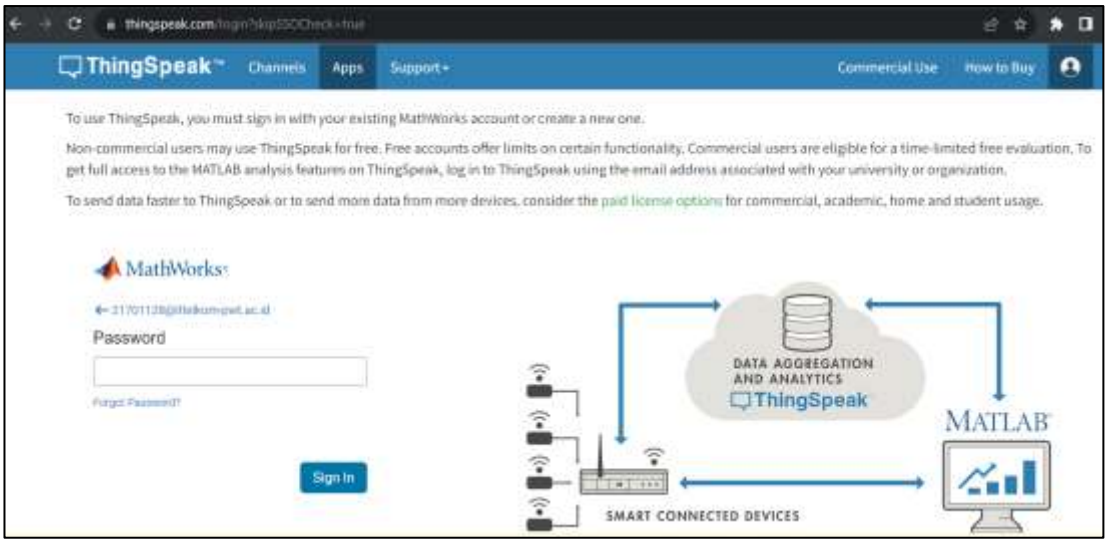
```
void ShowSerialData()
{
  while(SIM800L.available()!=0)
  Serial.write(SIM800L.read());
  delay(2000);
}
```

Gambar 3. 20 Bagian *Void Field*

Gambar 3.20 menggunakan fungsi menampilkan data serial pembacaan dari SIM800L.

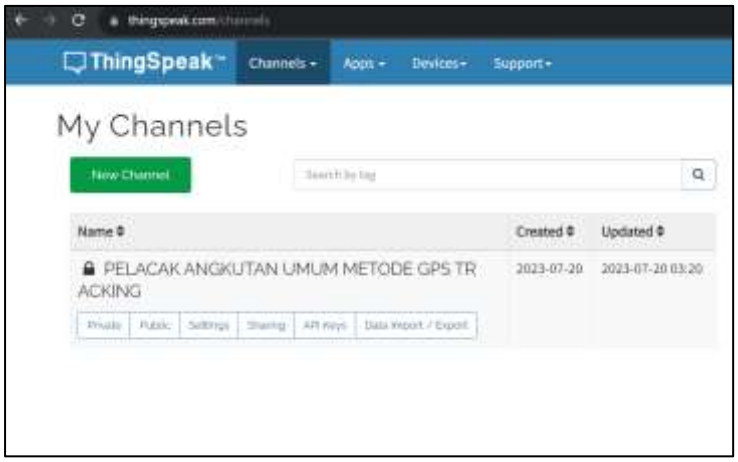
3.2.2.2 *THINGSPEAK*

Penelitian ini menggunakan *Thingspeak* yang merupakan salah satu layanan berbasis *cloud computing* sebagai *server* database untuk menyimpan data secara *online*. *Thingspeak* dapat bekerja pada perangkat *Arduino*.



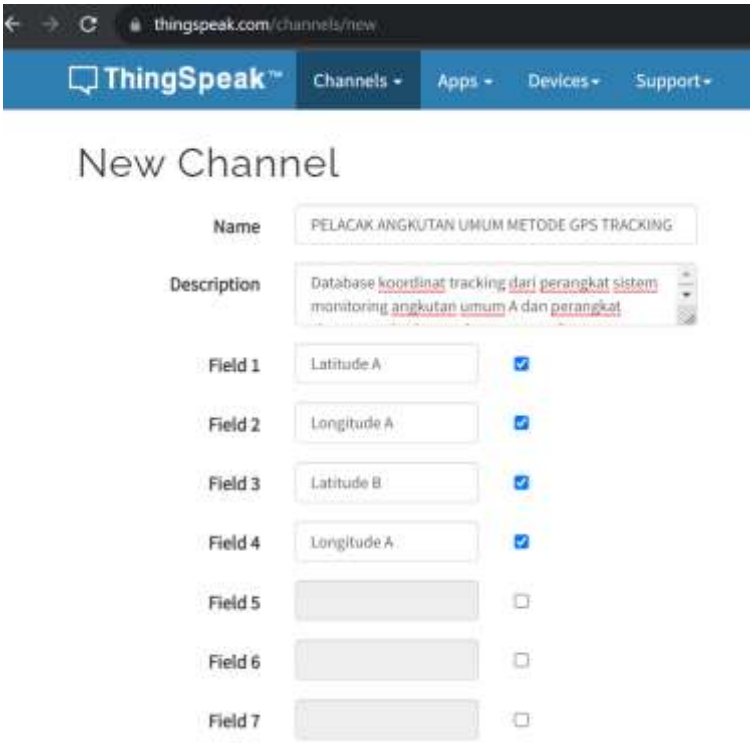
Gambar 3. 21 *Sign in to Thingspeak*

Seperti yang ditunjukkan pada gambar 3.21, sebelum menggunakan *Thingspeak* harus masuk dengan akun *Thingspeak* yang telah terdaftar sebelumnya dan mematuhi syarat dan ketentuan yang berlaku untuk layanan *Thingspeak* tersebut.



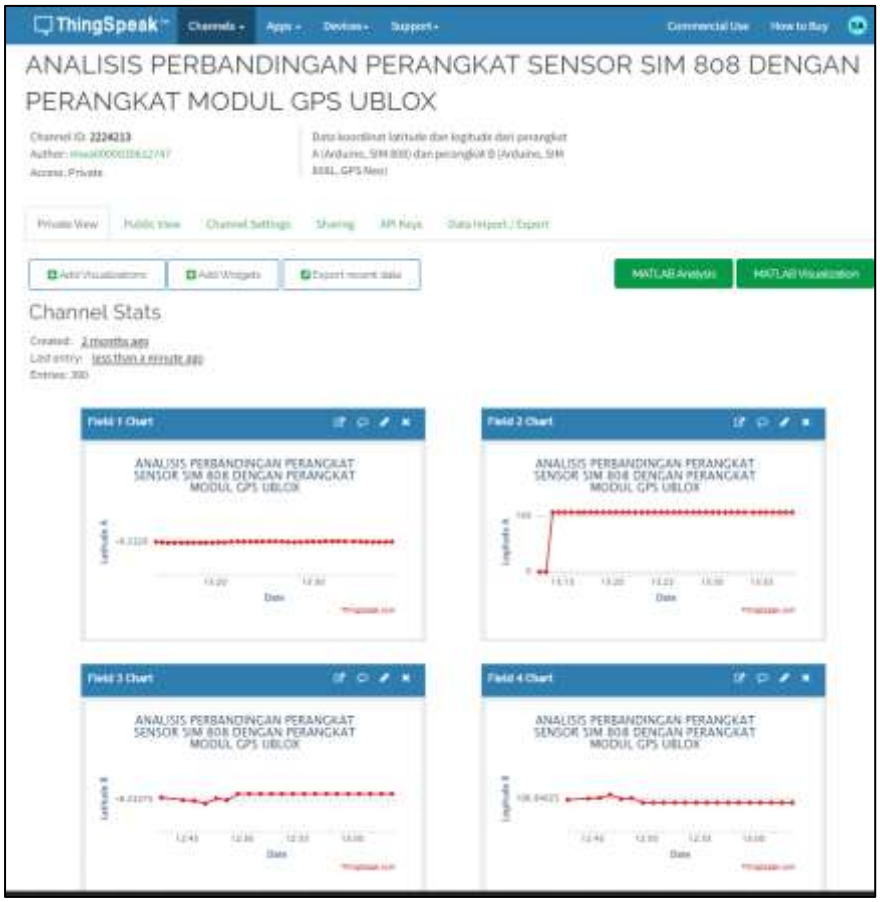
Gambar 3. 22 *New channel*

Untuk membuat channel atau tampilan baru, pilih menu My Channel, seperti yang ditunjukkan pada gambar 3.22.



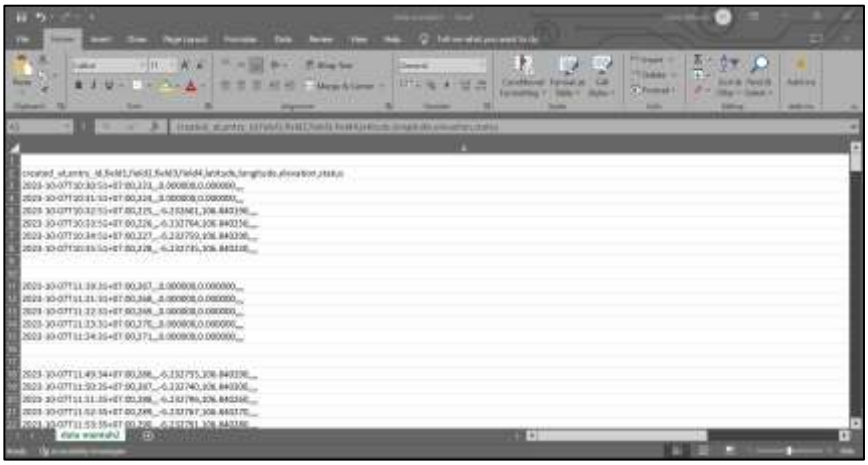
Gambar 3. 23 *New channel* (2)

Gambar 3.23 menunjukkan bagaimana membuat channel baru pada *Thingspeak* . Pada bagian kolom nama penelitian ini digunakan “PELACAK ANGKUTAN UMUM METODE *GPS* TRACKING”. Kemudian diperlukan dua kolom atai *Field* yaitu kolom 1 untuk data *latitude* dan kolom 2 untuk data *longitude*. Untuk menyimpan pengaturan channel baru, pilih tombol “*save channel*”.



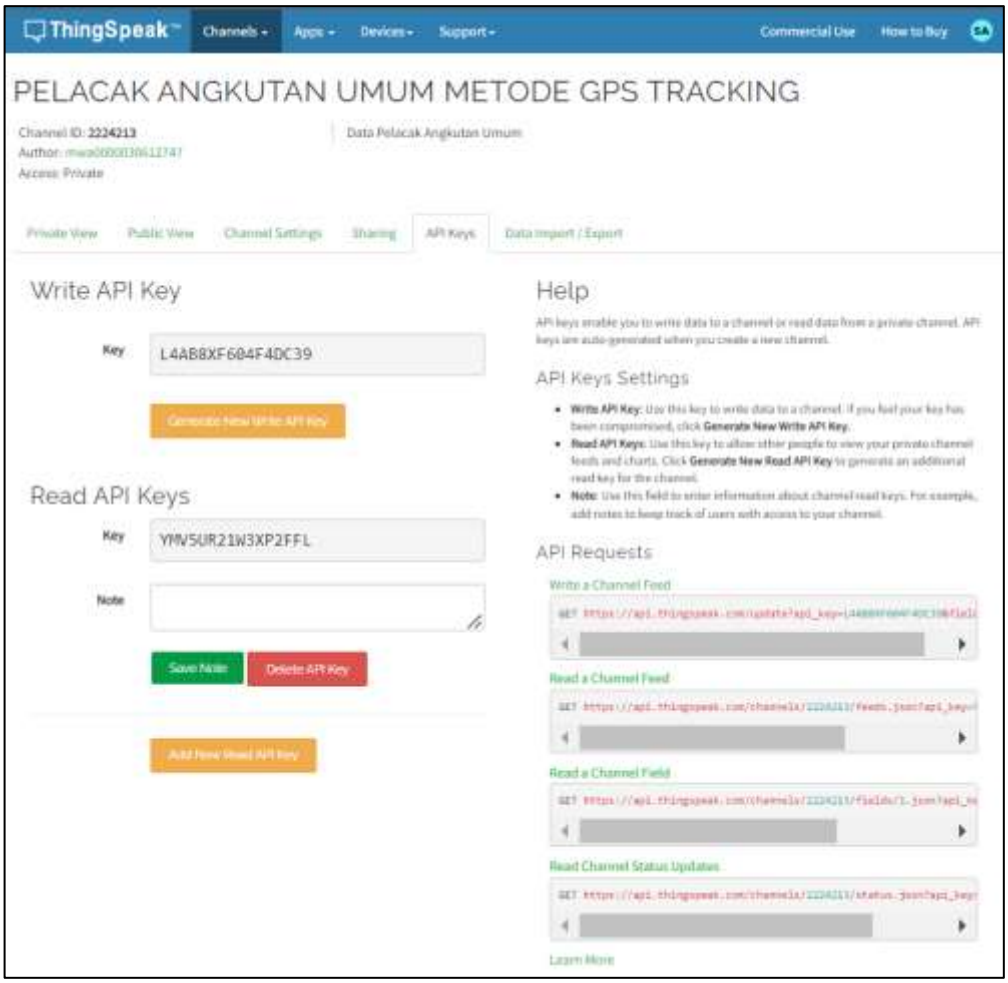
Gambar 3. 24 Tampilan *Field* pada *Thingspeak*

Gambar 3.24 menampilkan grafik data *latitude* dan *longitude*, dengan *Field* 1 menunjukkan data *latitude* dan *Field* 2 menunjukkan data *longitude*, yang diterima *Thingspeak* dari sistem *monitoring* angkutan umum secara *real-time* dan secara otomatis disimpan. Grafik ini menunjukkan hasil dari perubahan posisi sistem sebelumnya.



Gambar 3. 25 Tampilan data excel dari *Thingspeak*

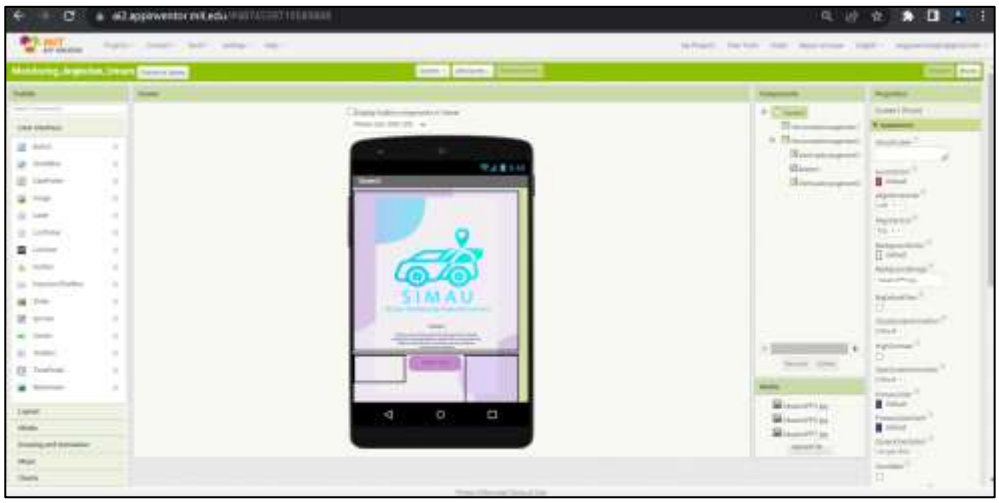
Seperti yang ditunjukkan pada gambar 3.25, yang menunjukkan deskripsi waktu, tanggal, *latitude*, dan *longitude* sistem *monitoring* angkutan umum, hasil dari grafik dapat diexport ke dalam format Excel.



Gambar 3. 26 *API Keys*

Untuk mendapatkan *Key* akses *Write* data dengan menggunakan menu API Keys, copy kunci pada bagian *Write* API Key. Sedangkan *Key Read API Keys* digunakan untuk membaca data. Seperti yang ditunjukkan pada gambar 3.26, aplikasi Android akan menggunakan Channel ID untuk menampilkan data.

3.2.2.3 *APP INVENTOR*



Gambar 3. 27 Tampilan *designer screen 1*

Gambar 3. 28 Tampilan *block screen 1*

Gambar 3.27 menunjukkan tampilan awal aplikasi Sistem *Monitoring* Angkutan Umum (SIMAU), yang digunakan untuk melacak angkutan umum. Pada tampilan awal aplikasi SIMAU terdapat tombol “MULAI”, dan ketika tombol “MULAI” diklik, akan diberikan perintah yang sama seperti yang ada di block screen 1 pada Gambar 3.28, yaitu untuk membuka layar dengan nama SIMAU.

Gambar 3. 29 Tampilan *designer* SIMAU

Gambar 3. 30 Tampilan *block* SIMAU

Gambar 3.29 menunjukkan tampilan desain SIMAU, yang menampilkan menu untuk pilihan pencarian angkutan umum. Setiap tombol melakukan fungsi yang sama seperti yang ditunjukkan pada tampilan blok SIMAU, misalnya, jika tombol 1 diklik "VIEW TRANSPORTATION" akan membuka layar dengan nama "*Monitoring*_Transportasi_Umum", dan jika tombol 2 diklik "HELP" akan membuka layar dengan nama "HELP".

Gambar 3. 31 Tampilan *designer screen web*

Gambar 3.27 menunjukkan tampilan layar desain dengan nama "*monitoring*angkutanumum". Pada layar " *monitoring*angkutanumum ", *web* view akan diarahkan ke url peta untuk menampilkan posisi angkutan umum. Untuk melihat peta harus mengubah setting *Java*script yang terdapat di lampiran 2.

Gambar 3. 32 Tampilan *designer screen* bantuan

Jika Anda mengklik tombol "HELP" di layar SIMAU, menu akan menampilkan informasi tentang aplikasi SIMAU dan instruksi untuk memberikan masukan melalui kontak email, seperti yang ditunjukkan pada gambar 3.32.

3.3 SKEMA PENGUJIAN AKURASI PERANGKAT SISTEM *MONITORING* A DAN B

Pengujian akurasi dilakukan dengan membandingkan titik koordinat sistem A dan B dengan koordinat sebenarnya menggunakan *Google* Earth. Pengujian ini akan dilakukan 30 kali pengambilan data koordinat titik lokasi perangkat yang berbeda yaitu 15 kali pada perangkat *system monitoring* A kemudian 15 kali selanjutnyaa pada perangkat *system monitoring* B. Kemudian hasil dari pengambilan data koordinat akan dilakukan pengukuran selisih jarak koordinat dengan koordinat sebenarnya menggunakan *ruler* pada *Google earth*. Pengujian ini bertujuan untuk mengetahui perbandingan seberapa akurat titik koordinat yang diperoleh dari perangkat antenna *GPS* pada modul sistem A dan sistem B.

Berikut merupakan perhitungan pada pengujian akurasi perangkat sistem *monitoring* angkutan umum :

Selisih Jarak = Panjang ruler dari titik pin koordinat perangkat *system monitoring* sampai dengan titik pin koordinat sebenarnya.

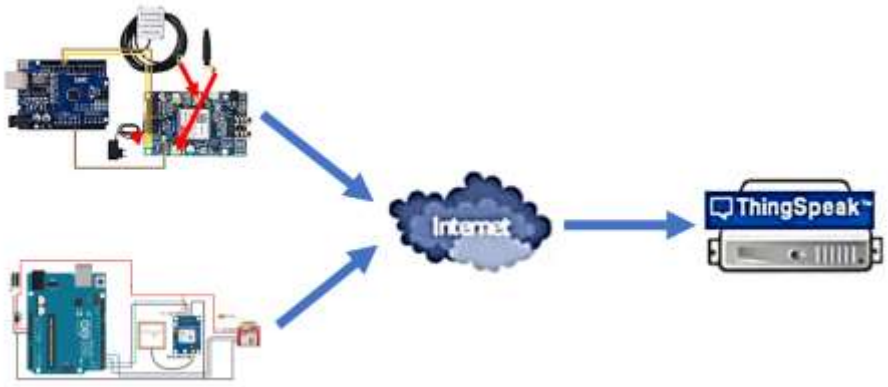
$$\text{Rata-rata Selisih Jarak} = \frac{\text{Jumlah data selisih jarak}}{\text{Banyak data selisih jarak}}$$

$$\text{Simpangan Baku} = \sqrt{\frac{\sum_{t=1}^{n}(Selisih\ jarak - rata\ rata\ selisih\ jarak)^2}{\text{Banyak data selisih jarak} - 1}}$$

Root Mean Squared error = $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_i - F_i)^2}{n}}$

$$= \sqrt{\frac{\sum_{t=1}^{n}(Selisih\ jarak)^2}{\text{Banyak data selisih jarak}}} \text{ atau } \sqrt{MSE}$$

3.4 SKEMA PENGUJIAN PARAMETER QOS (*QUALITY OF SERVICE*)



Gambar 3. 33 Bagan untuk pengujian parameter QoS (*Quality of Service*)

Dalam penelitian ini, perangkat sistem A dan sistem B berfungsi sebagai sisi pengirim dan *server Thingspeak* berfungsi sebagai sisi penerima. Gambar 3.33 menunjukkan bagan pengujian parameter QoS yang akan dilakukan, yang mencakup pengujian *Packet Loss* dan *delay* berdasarkan dari data yang diterima di sisi penerima atau *server Thingspeak* . Tujuan dari pengujian ini adalah untuk membandingkan perbedaan nilai parameter QoS dari modul sistem A dan sistem B.

3.4.1 SKEMA PENGUJIAN *PACKET LOSS*

Pengujian Packet Loss digunakan untuk menghitung jumlah total paket pengiriman data koordinat yang hilang dari sisi tx ke sisi rx. Pada penelitian ini, pengujian ini dilakukan selama lima menit dan dilakukan sebanyak tiga puluh kali. Setiap pengujian ditentukan jumlah total *Packet Loss* yang didapatkan dengan menyesuaikan antara data yang dikirim dari tx yang dapat dilihat pada serial monitor dan data yang diterima di sisi rx yang dapat diambil dari CSV *file server Thingspeak* . Apabila data yang telah diterima dari sisi rx telah sesuai dengan data yang dikirim dari sisi tx, maka menunjukan bahwa tidak ada data *loss* atau data yang hilang, tetapi jika data yang telah diterima di sisi rx lebih sedikit dari data yang dikirim dari sisi tx, maka menunjukan bahwa terdapat data yang hilang.

Berikut merupakan perhitungan pada pengujian *packet loss* perangkat sistem *monitoring* angkutan umum :

Presentase *Packet Loss* $= 100\% - ((\left(\frac{\text{Jumlah data yang diterima}}{\text{Banyak data yang dikirim}}\right) x\ 100\%))$

Rata-rata *Packet Loss* $= \frac{\text{Jumlah data Presentase } Packet\ Loss}{\text{Banyak data Presentase } Packet\ Loss}$

3.4.2 SKEMA PENGUJIAN *DELAY*

Pengujian *delay* dilakukan agar mengetahui berapa waktu tunda suatu paket. Pengujian *delay* pada penelitian ini yaitu data *delay* dari selisih waktu data awal yang diterima rx ke data terima rx yang selanjutnya. Pada penelitian ini, pengiriman data dari sisi tx ke sisi rx diuji selama lima menit dan dilakukan sebanyak tiga puluh kali. Data *delay* diambil dari data sisi rx, yaitu dari file CSV di *server Thingspeak* . Dari tiap pengujian dilakukan perhitungan nilai *delay* tiap data yang diterima dan juga dilakukan perhitungan nilai rata-rata *delay* di tiap pengujian, kemudian diambil data nilai rata-rata *delay* secara keseluruhan.

Berikut merupakan perhitungan pada pengujian *delay* perangkat sistem *monitoring* angkutan umum :

Delay tiap paket pengiriman = waktu terima data koordinat kedua - waktu terima data koordinat awal

Rata-rata *delay* pengiriman $= \frac{\text{Jumlah data } Delay \text{ tiap paket pengiriman}}{\text{Banyak data } Delay \text{ tiap paket pengiriman}}$

Rata-rata *delay* keseluruhan $= \frac{\text{Jumlah dataRata}-\text{rata } delay \text{ pengiriman}}{\text{Banyak data Rata}-\text{rata } delay \text{ pengiriman}}$

Simpangan Baku $= \sqrt{\frac{\sum_{t=1}^{n}(rata-rata\ delay\ pengiriman)^2}{\text{Banyak data r}ata-rata\ delay\ pengiriman}}$