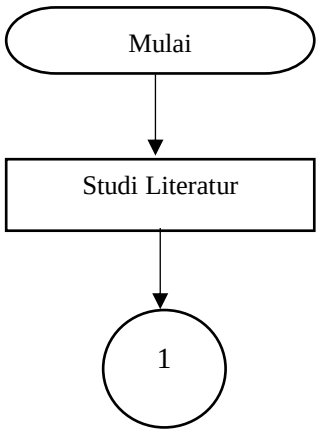
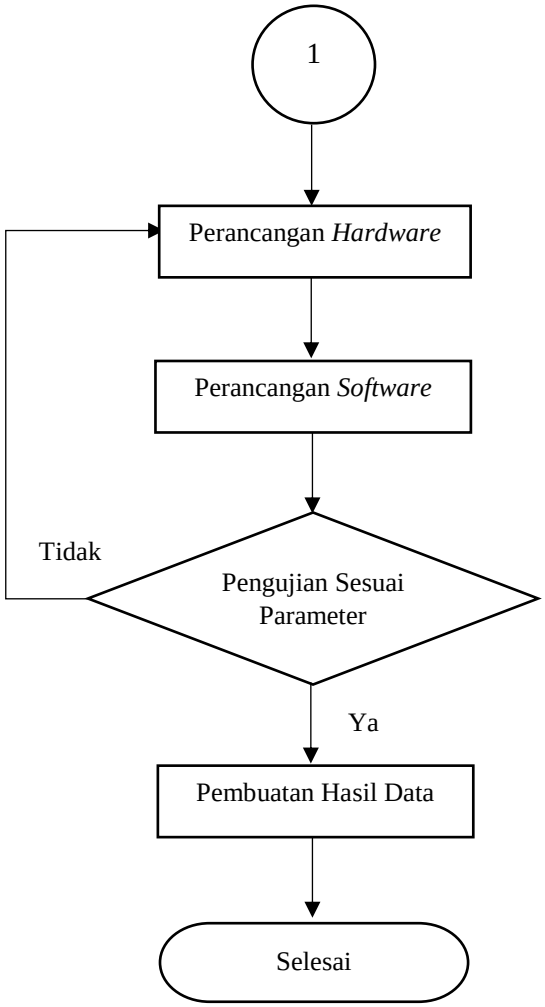
BAB 3
METODE PENELITIAN

Pada penelitian ini penulis merencanakan untuk membuat rancang bangun alat *monitoring* angkutan umum dengan metode GPS *tracking*. GPS *Tracker* adalah sebuah alat untuk memantau atau melacak dan penentu lokasi sebuah kendaraan menggunakan satelit GPS secara akurat dalam bentuk titik kordinat yang dapat kita amati secara *realtime* melalui peta *digital*. Diperlukan metodologi penelitian yang digunakann pada tugas akhir ini adalah sebagai berikut.

3.1 ALUR PENELITIAN

Perancangan suatu penelitian dilakukan dalam berbagai tahap yaitu dimulai dari pencarian studi literatur, melakukan perancangan *hardware*, melakukan perancangan *software*, melakukan pengujian sesuai parameter, dan yang terakhir adalah tahap pembuatan hasil data dari hasil pengujian sistem. Dalam sebuah perancangan suatu penelitian diperlukan adanya alur penelitian agar dalam melakukan perancangan dapat berjalan sesuai dengan rencana yang telah disusun seperti diatas. Salah satu bentuk dari alur penelitian adalah *flowchart*, jika dilihat secara singkat *flowchart* dapat menjelaskan proses perancangan pada penelitian yang akan dibuat seperti pada gambar 3.1.



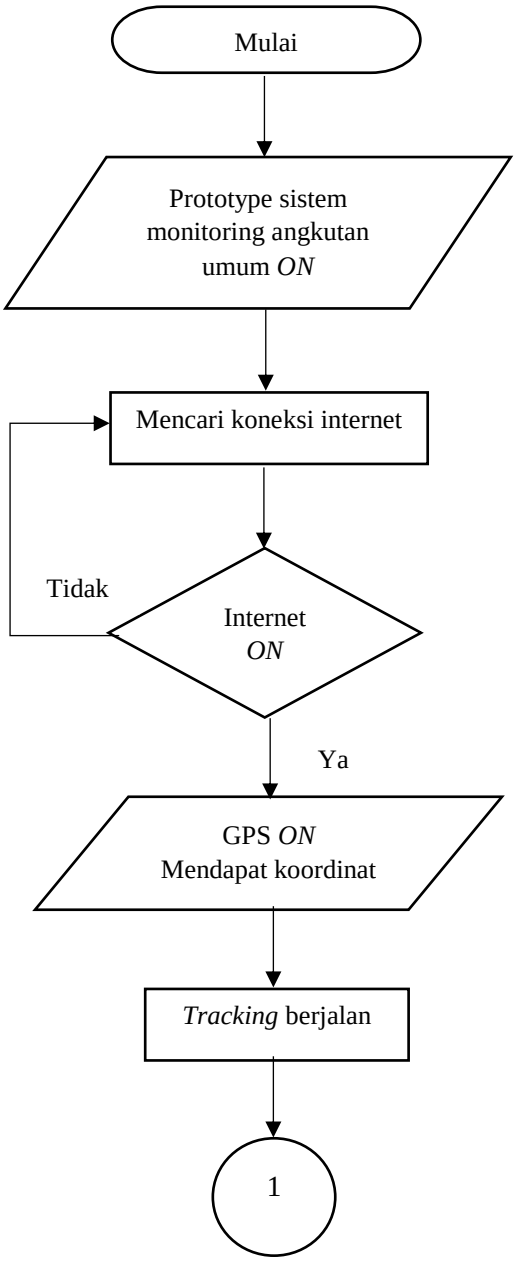


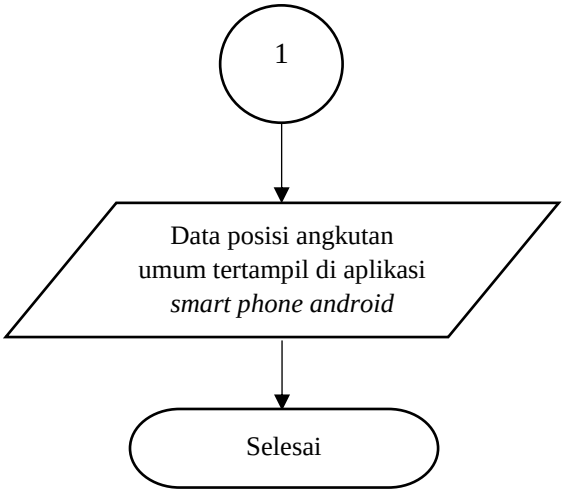
Gambar 3.1. *Flow chart* Alur Penelitian

Sesuai dengan *flow chart* alur penelitian pada gambar 3.1 dimulai dari pencarian studi literatur yang dilakukan dengan membandingkan kajian teori dari perancangan sebelumnya, selain itu studi literatur dilakukan dengan membaca buku-buku, jurnal ilmiah dan beberapa artikel dari *internet* yang dapat menunjang dari cara kerja dan sistem setiap perangkat yang digunakan. Pada blok diagram perancangan *hardware* merupakan proses pengumpulan alat dan bahan yang terdiri dari perangkat modul SIM808 untuk menerima data GPS dan mendukung konektivitas GPRS yang digunakan sekaligus sebagai data masukkan untuk *Arduino UNO* R3, perangkat *mikrokontroler Arduino UNO* R3 sebagai pengolah data masukkan dari perangkat modul SIM808. Pada blok diagram perancangan *software* merupakan proses pembuatan aplikasi yang digunakan pada perancangan Tugas Akhir ini dengan menggunakan *App Invertor* secara online yang

menampilkan data lokasi keberadaan perangkat sistem monitoring angkutan umum yang dikirim dari *Thingspeak*. Setelah perancangan *hardware* dan *software* untuk tiap-tiap perangkat maka selanjutnya adalah melakukan pengujian sesuai dengan parameter, jika pada pengujian tersebut tidak sesuai dengan parameter atau terdapat kesalahan maka akan dilakukan perancangan *hardware* dan *software* kembali hingga pengujian tersebut berhasil dan apabila pada pengujian tersebut sesuai dengan parameter maka akan langsung dibuat hasil data berdasarkan pada pengujian tersebut.

3.1.1 *FLOW CHART* ALUR SISTEM GPS





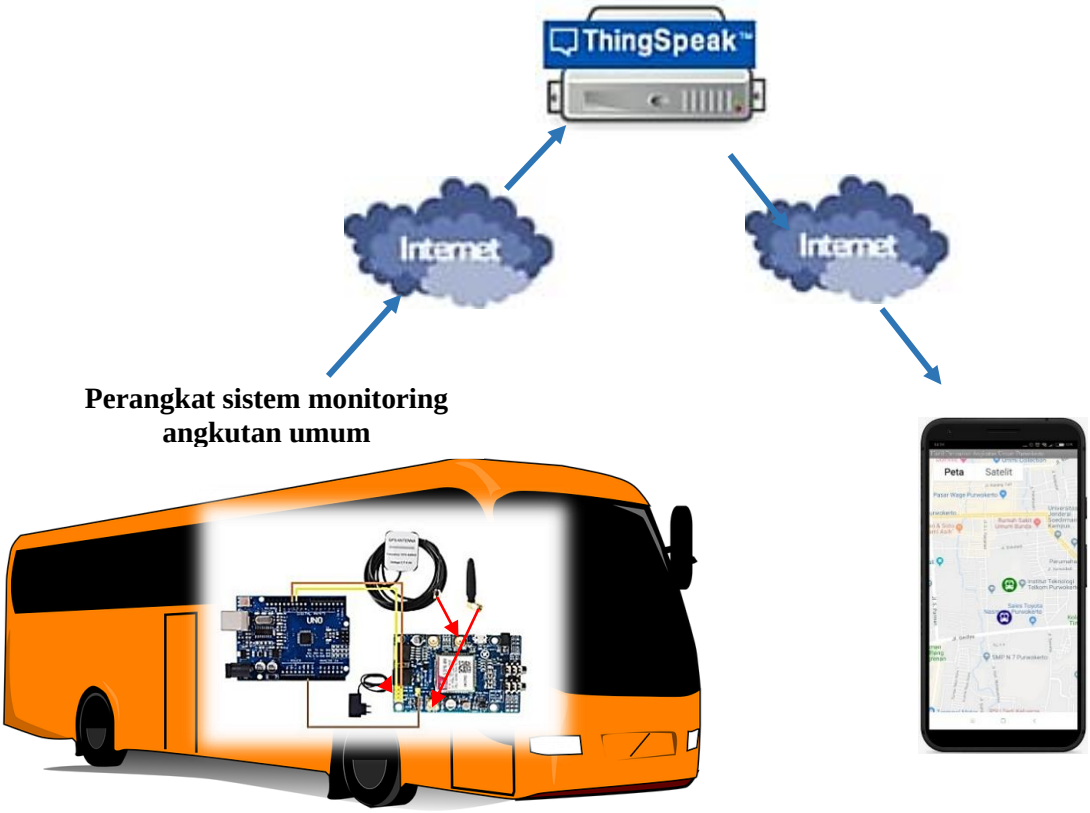
Gambar 3.2. *Flow chart* Alur Sistem GPS *Tracking*

Berdasarkan *flow chart* alur sistem GPS *tracking* diatas diawali dari kondisi sistem monitoring yang telah terpasang di angkutan umum telah aktif, kemudian sistem akan mencari koneksi internet, jika sistem monitoring angkutan umum tersebut tidak mendapatkan koneksi internet maka sistem monitoring tersebut akan kembali mencari koneksi internet, tetapi jika telah mendapatkan koneksi internet maka GPS yang sudah aktif dan mendapatkan koordinat selama *tracking* berjalan akan ditampilkan pada aplikasi *smart phone android* di sisi *user*.

3.2 PERANGKAT YANG DIGUNAKAN

Dalam penelitian ini perangkat yang digunakan meliputi peralatan perangkat keras untuk perancangan *prototype* dan perangkat lunak. Modul SIM808 pada blok diagram sistem pada gambar 3.3 berfungsi untuk menerima data GPS dan mendukung konektivitas GPRS yang digunakan sekaligus sebagai data masukkan untuk *Arduino UNO* R3. Perangkat *mikrokontroler Arduino UNO* R3 sebagai pengolah data masukkan dari perangkat modul SIM808.

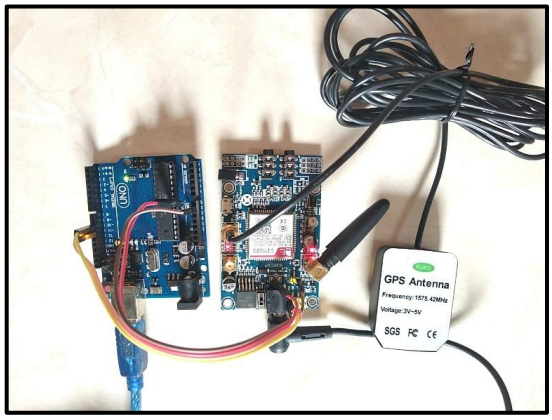
Data dari prototype yang terpasang pada angkutan umum tersebut kemudian dikirim ke *server* database *Thingspeak* dan dapat di akses melalui aplikasi *android* yang telah dibuat melalui *App Inventor*. Keluaran dari aplikasi *smartphone android* disisi *user* tersebut yaitu berupa *marker* lokasi keberadaan angkutan umu



Gambar 3.3. Blok Sistem Monitoring Angkutan Umum

3.2.1 PERANCANGAN ALAT

Dalam perancangan untuk membangun sebuah rancang bangun sistem *monitoring* angkutan umum menggunakan metode GPS digunakan beberapa peralatan yang di gunakan untuk menunjangnnya kegiatan penelitian ini yang akan membentuk perangkat sistem monitoring angkutan umum seperti pada gambar 3.4 dibawah ini, yaitu:

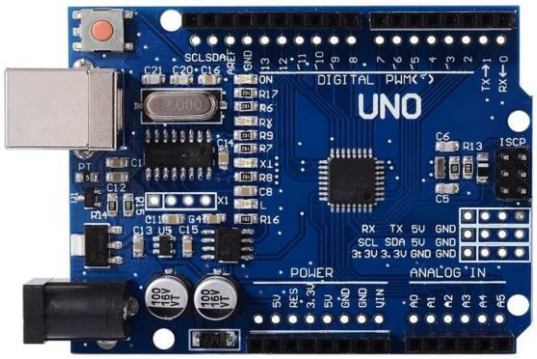


Gambar 3.4. Perangkat Sistem Monitoring Angkutan Umum

A. *Arduino UNO*

Pada sistem ini *Arduino UNO* berfungsi sebagai processor yang mengolah masukan dari sensor yang digunakan. *Arduino UNO* mempunyai 14 pin masukan atau keluaran *digital* (6 pin bisa dipakai sebagai keluaran PWM), 6 *input analog*, jack listrik tombol *reset*, 16 MHz osilatorkristal, dan koneksi USB. Pin-pin tersebut berisi semua yang akan digunakan untuk mendukung *mikrokontroler* dengan menghubungkan ke komputer dengan kabel USB atau sumber tegangan bisa didapat dari adaptor AC-DC ataupun bisa menggunakan baterai untuk menggunakannya. [13]

Pada sistem monitoring angkutan umum menggunakan pin 10 dan 11 sebagai pengolah data masukkan dan keluaran dan juga pin GND atau pin *Ground*.



Gambar 3.5 *Arduino UNO* [13]

Tabel 3.1 Keterangan Fungsi Bagian *Board Arduino UNO* yang digunakan

Bagian *Board*	**Fungsi**
Pin 10 dan 11	Digunakan sebagai masukan atau keluaran, dapat diatur oleh program
USB	berfungsi sebagai: a. Memuat program dari komputer ke dalam *board*. b. Komunikasi serial antara *board* dan *computer*. c. Memberi daya listrik ke *board*.
GND	Pin *Ground*

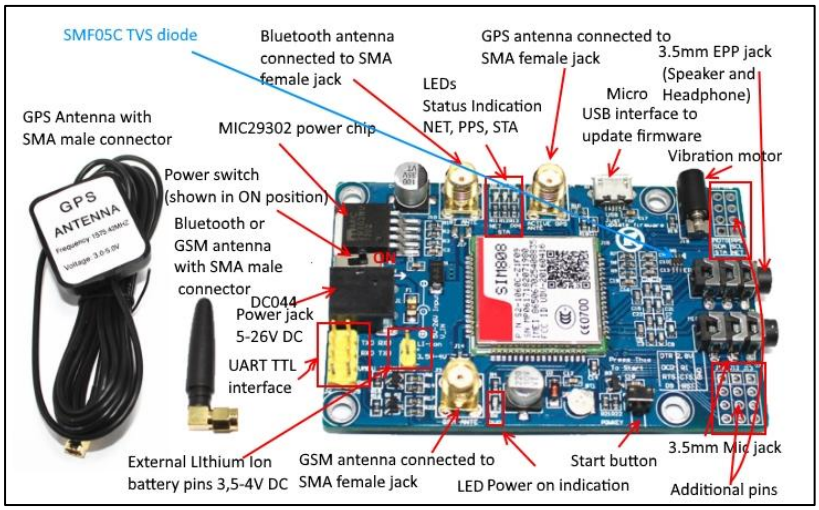
Deskripsi *Arduino Uno*:

Tabel 3.2 Deskripsi *Arduino Uno*

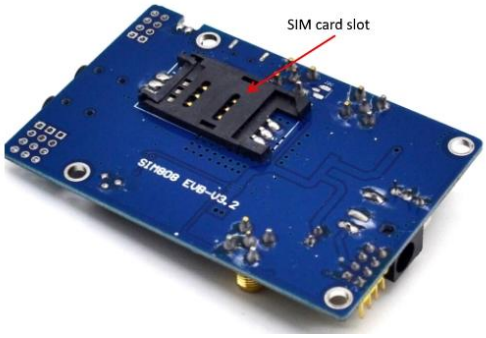
Mikrokontroller	**Atmega328**
Operasi *Voltage*	5V
Input Voltage	7-12 V (Rekomendasi)
Input Voltage	6-20 V (*limits*)
Arus	50 mA
I/O	14 pin (6 pin untuk PWM)
Flash Memory	32KB
EEPROM	1 KB
Bootloader	SRAM 2 KB
Kecepatan	16 Mhz

B. Modul SIM808

Modul SIM808 pada sistem ini digunakan untuk komunikasi secara jaringan seluler GPRS (*Internet*), selain itu modul SIM808 juga terdapat sensor lokasi A-GPS (*indoor*/*outdoor*) yang bisa bekomunikasi dengan satelit di dalam gedung ataupun di area terbuka sehingga pada sistem ini modul SIM808 juga digunakan untuk mendapatkan nilai koordinat *latitude* dan *longitude*. Modul SIM808 dihubungkan dengan *Arduino* melalui pin Rx dan Tx, juga pada pin LI-ion -. [23]



Gambar 3.6. Modul SIM808 [18]



Gambar 3.7 Tampak belakang dari Modul SIM808 [18]

Penggunaan modul SIM808 yang memiliki fungsi GPS akan memungkinkan variabel koordinat dilacak secara lancar di lokasi manapun dan kapan saja dengan jangkauan sinyal.

Tabel 3.3 fungsi pin pada Modul SIM808 yang digunakan [18]

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

Tabel 3.3 fungsi pin pada Modul SIM808 yang digunakan (lanjutan) [18]

Pin	Fungsi
PPS	*Output speaker* atau *jack headphone*. *Speaker* atau *headphone* terhubung.
NET	Status jaringan.
STA	*Power on* status

C. Laptop

Pada penelitian ini Laptop digunakan untuk melakukan konfigurasi terhadap mikro pengendali *Arduino Uno* dan modul SIM808, serta memantau data yang dikirimkan dari *Arduino Uno*. Perangkat ini difungsikan untuk membuat *listing* pemrograman, pengambilan sekaligus mengolah hasil data dari pengujian sistem dan perangkat, dan melihat hasil data dari server *Thingspeak*, dan juga dugunakan untuk pembuatan aplikasi *smartphone android*.

D. *Smart Phone Android*

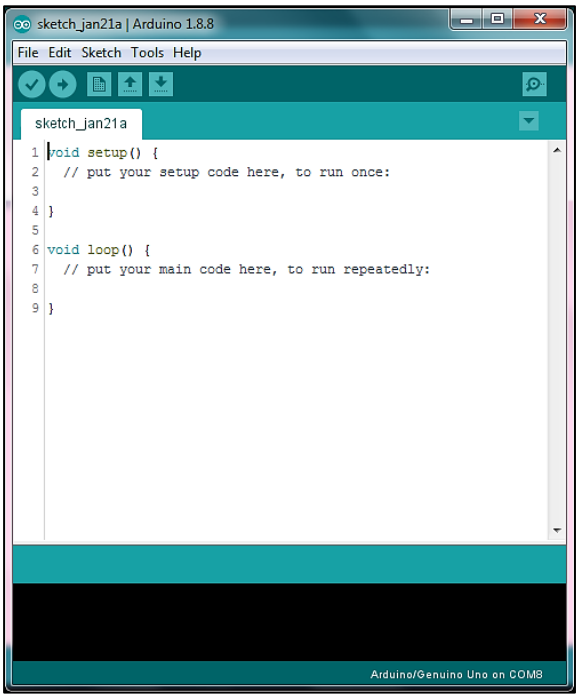
Pada penelitian ini *Smart Phone* digunakan disisi pengguna untuk melakukan monitoring angkutan umum. Proses monitoring angkutan umum tersebut melalui sebuah aplikasi yang telah diatur untuk monitoring angutan umum.

3.2.2 PERANGKAT LUNAK UNTUK PENELITAN

A. *Arduino* IDE

Arduino IDE pada penelitian ini dugunakan untuk melakukan konfigurasi sebuah program yang nantinya akan di masukan ke *Arduino*. Program tersebut bersifat *open-source* dalam lingkup Bahasa pemrograman

Bahasa *Arduino* dan dapat bekerja pada sistem oprasi *Windows* 7 x64 yang digunakan dalam penelitian ini. [13]



Gambar 3.8 Tampilan awal *Arduino Ide* 1.8.8

```
#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 3.9 Bagian *header*

Gambar 3.9 merupakan sebuah prosesor pengarah yang mengatakan kepada compiler untuk meletakan kode dari *header file iostream.h* kedalam program. Fungsi *cout* memerlukan file *iostream.h* berupa <SoftwareSerial.h> menunjukan kode *header* yang melibatkan *library SoftwareSerial* yang kemudian ditunjukan pada *header* baris berikutnya serial perangkat lunak untuk modul GPS/GSM dimana 10 merupakan pin *Arduino Uno* yang terhubung ke TX modul SIM808, sedangkan 11 merupakan pin *Arduino Uno* yang terhubung ke RX modul SIM808. Selanjutnya pada program terdapat data *string* untuk variable *latitude*, *longitude*, *altitude*, *timegps*, *speedknot*, c, dan *state*.

```
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 3.10 *Void setup*

Gambar 3.10 diatas menunjukan fungsi dari *setup*() dipanggil ketika sketsa dimulai. Pada struktur ini kecepatan transfer data dari modul SIM808 ke Arduino yaitu memiliki baudrate 9600, dan serial *port* yang terbuka yaitu pada *baudrate* 9600 untuk menampilkan di serial monitor. Selanjutnya dapat menggunakan fungsi *sendData* untuk mencetak perintah membuka GPS. Kemudian dilakukan perintah AT *command* untuk melakukan konfigurasi jaringan yang digunakan, dan juga perintah AT *command* untuk mengaktifkan konektifitas *wireless*.

```
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);
```

(a)

```
  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);
}
```

(b)

Gambar 3.11 *Void GetGPS*

Pada gambar 3.11 (a) dan (b) merupakan fungsi untuk mengaktifkan GPS SIM808 yang nantinya akan diletakan pada *void loop*(). Pada *void GetGPS*() dilakukan perintah AT *command* untuk mengaktifkan dan mendapatkan data dari SIM808, data tersebut selanjutnya akan dilakukan teknik parsing atau pembagian *array* tiap masing-masing data dari *state*, *timegps*, *latitude*, *longitude*, *altitude*, dan *speedknot*.

```
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 3.12 *Void ConnectServer*

Pada gambar 3.12 merupakan program dari *void ConnectServer*(), dimana pada program ini terdapat beberapa perintah AT *Command* yang digunakan agar data terhubung ke server dengan protokol TCP.

```
void Field() //fungsi untuk mengirim data ke thingspeak
{
  Serial.println(mySerial.readString());
  String str = "GET https://api.thingspeak.com/update?api_key=APIKEYTHINGSPEAK&field1=" + String (latitude)+"&field2="+String(longitude);
  mySerial.println(str);
  delay(2000);

  mySerial.println((char)26);
  delay(4000);
  Serial.println(mySerial.readString());
  mySerial.println();
}
```

Gambar 3.13 *Void Field*

Gambar 3.13 menggunakan fungsi pemanggilan *void Field*(), dimana pada program tersebut berguna untuk mengirim data *latitude* dan *longitude* menuju *field* 1 dan *field* 2 yang terdapat pada *Thingspeak.*

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

Gambar 3.14 *Void loop*

Fungsi *loop*() yang ditunjukan pada gambar 3.14 pada program tersebut berguna untuk melaksanakan atau mengeksekusi perintah program yang telah dibuat secara berulang-ulang. Pada fungsi tersebut terjadi perintah perulangan dari pemanggilan dari fungsi *Get*GPS() untuk mendapatkan data *latitude* dan *longitude* posisi sistem monitoring angkutan umum, kemudian masuk ke fungsi *ConnectServer*() untuk terhubung ke server *Thingspeak* dan selanjutnya masuk ke fungsi *Field*() untuk mengirim data *latitude* dan *longitude* tersebut kedalam *field Thingspeak.*

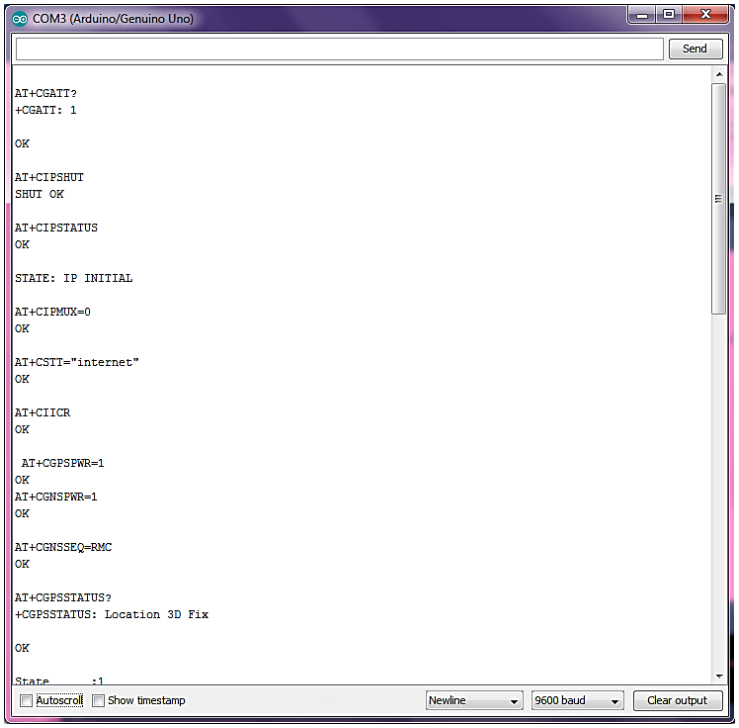
Untuk penjelasan dari masing-masing perintah AT *Command* yang digunakan pada program dapat dilihat pada tabel 3.4 dibawah ini :

Tabel 3.4 Perintah AT *Command* yang digunakan

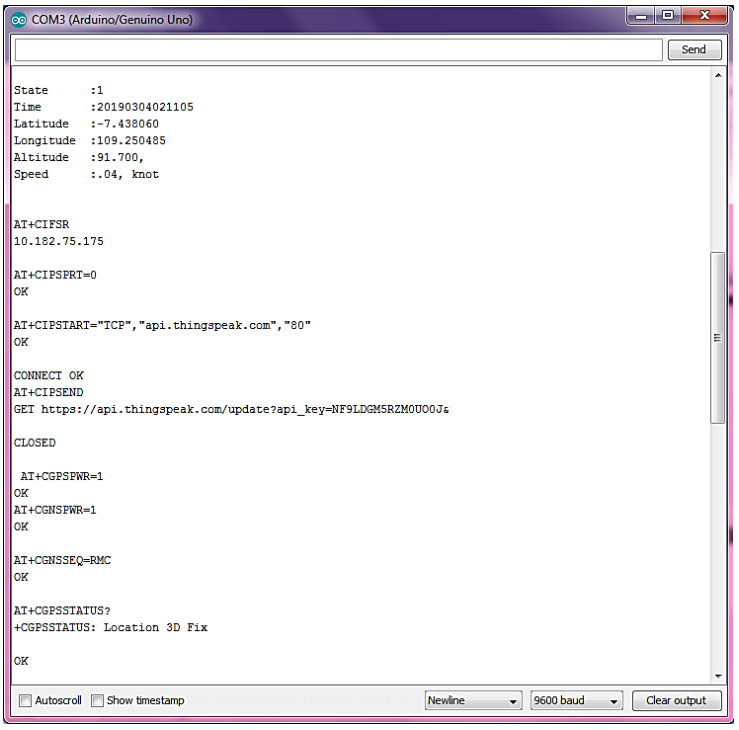
Perintah AT *Command*	**Penjelasan**
AT	Memulai perintah
AT+CGATT?	Memeriksa apakah perangkat telah terpasang ke GPRS.0Detch,1Attch
AT+CIPSHUT	*Shut packet* data *protocol context*

Tabel 3.4 Perintah AT *Command* yang digunakan (lanjutan)

Perintah AT *Command*	Penjelasan
AT+CIPSTATUS	Mengembalikan status koneksi saat ini
AT+CIPMUX=0	Membuat koneksi multi-IP (0=*single connection*)
AT+CSTT=\"internet\"	Mengatur jaringan , telkomsel APN = internet
AT+CIICR	Memunculkan koneksi *wireless*
AT+CGPSPWR=1	Buka GPS
AT+CGNSSEQ=RMC	Tentukan kalimat NMEA terakhir yang diuraikan
AT+CGPSSTATUS?	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



(a)



(b)

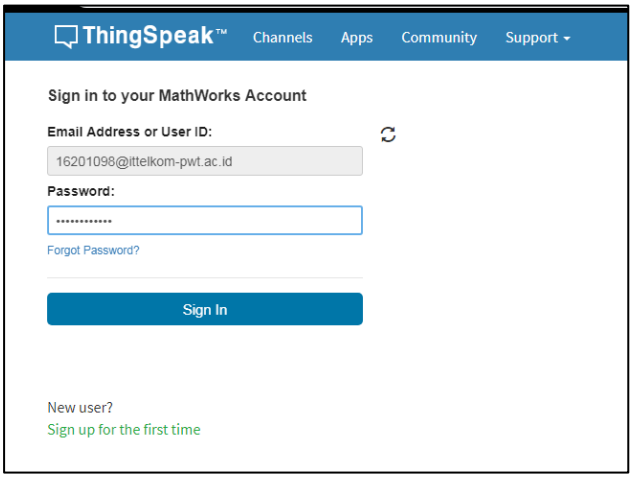
Gambar 3.15 Tampilan pada serial monitor

Pada saat ditampilkan di serial monitor dan telah di atur pengaturan *baud rate* pada 9600 maka akan muncul tampilan seperti gambar 3.15. Pada

gambar tersebut dapat terlihat aktifitas AT *command* yang telah terdapat pada program dan juga muncul data *state*, time, *latitude*, *longitude*, altitude, dan *speed* dari sistem monitoring angkutan umum.

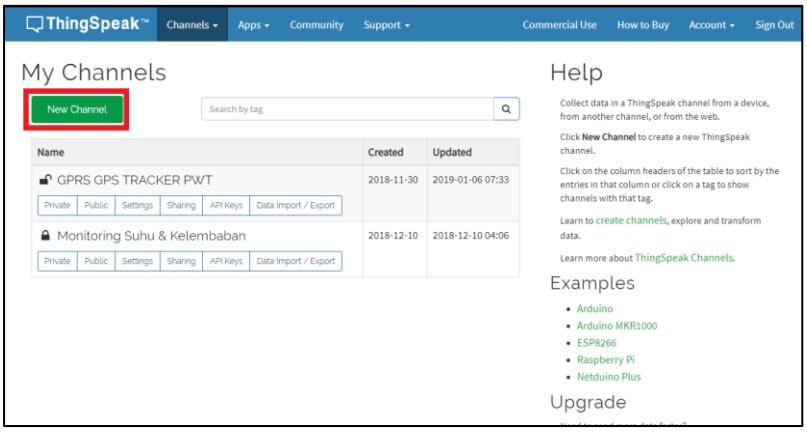
B. *Thingspeak*

Pada penelitian ini menggunakan *Thingspeak* yang merupakan salah satu layanan berbasis *cloud computing* yang digunakan untuk menyimpan data secara *online*. Layanan ini merupakan layanan *platform* analisis *Internet of Things* (IoT) berbasis *cloud* yang memungkinkan pengguna dapat mengumpulkan, memvisualisasikan, dan menganalisis aliran data. *Thingspeak* memberikan visualisasi instan terhadap data yang di-posting dari perangkat pengguna. Fitur *Thingspeak* diantaranya yaitu mengumpulkan data dalam bentuk *private channel*, mendukung *restful* dan MQTT API, analisis dan visualisasi berbasis MATLAB, *event scheduling*, mendukung *alert*, integrasi aplikasi dan dukungan komunitas *global*. *Thingspeak* dapat bekerja pada perangkat *Arduino*.



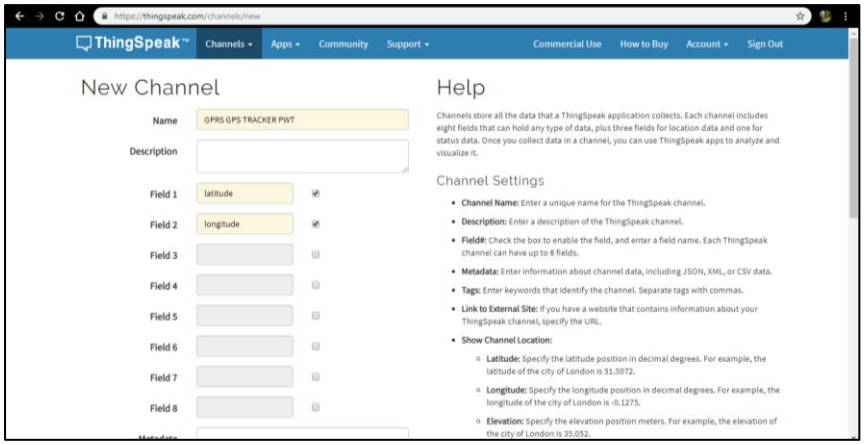
Gambar 3.16 *Sign in to Thingspeak*

Sebelum menggunakan *thingspeak* terlebih dahulu masuk dengan akun *thingspeak* yang telah terdaftar sebelumnya dengan syarat dan ketentuan yang berlaku pada layanan *thingspeak* tersebut seperti yang ditunjukan pada gambar 3.16.



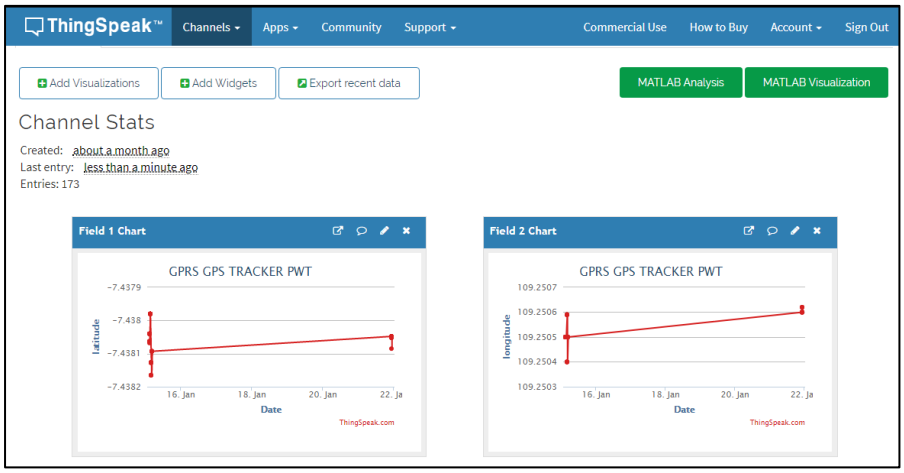
Gambar 3.17 *New channel*

Pada menu *my channel* pilih *new channel* yang berfungsi untuk membuat *channel* atau tampilan baru seperti pada gambar 3.17.



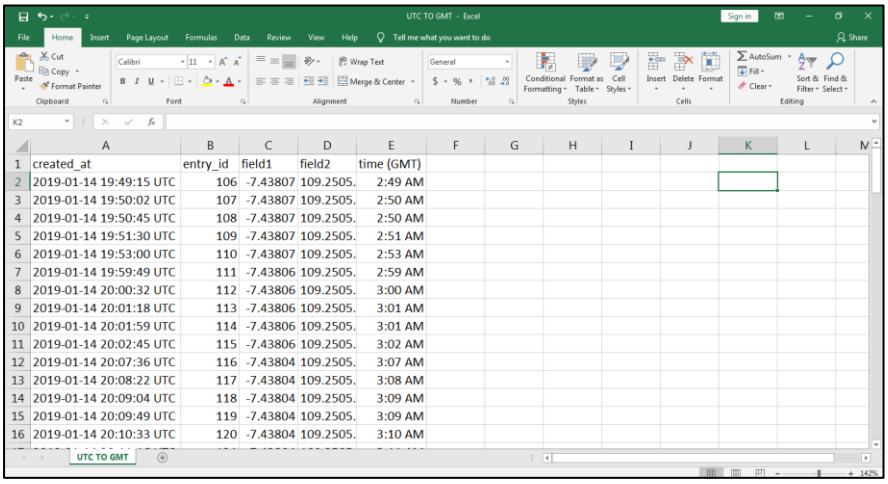
Gambar 3.18 *New channel* (2)

Gambar 3.18 menunjukan tampilan saat akan membuat *channel* baru pada *thingspeak*. Pada penelitian ini menggunakan "GPRS GPS TRACKER PWT" pada bagian kolom nama. Kemudian membutuhkan dua buah *field*, *field* 1 untuk data *latitude* dan *field* 2 untuk data *longitude*. Untuk menyimpan pengaturan *channel* baru tersebut dilakukan dengan cara memilih tombol *save channel*.



Gambar 3.19 Tampilan *field* 1 dan *field* 2 pada *thingspeak*

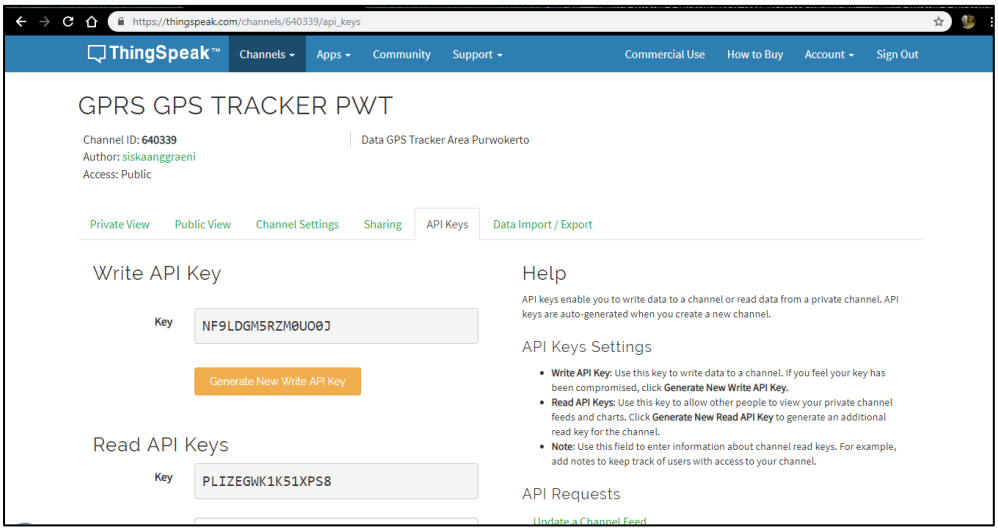
Pada gambar 3.19 Merupakan tampilan grafik data *latitude* dan *longitude*, dimana pada *field* 1 merupakan data *latitude* dan pada *field* 2 merupakan data dari *longitude* yang diterima oleh *thingspeak* dari sistem monitoring angkutan umum secara *real time* dan secara otomatis akan tersimpan. Grafik tersebut menunjukan hasil perubahan posisi sistem monitoring angkutan umum yang telah dilakukan sebelumnya.



	created_at	entry_id	field1	field2	time (GMT)
2	2019-01-14 19:49:15 UTC	106	-7.43807	109.2505.	2:49 AM
3	2019-01-14 19:50:02 UTC	107	-7.43807	109.2505.	2:50 AM
4	2019-01-14 19:50:45 UTC	108	-7.43807	109.2505.	2:50 AM
5	2019-01-14 19:51:30 UTC	109	-7.43807	109.2505.	2:51 AM
6	2019-01-14 19:53:00 UTC	110	-7.43807	109.2505.	2:53 AM
7	2019-01-14 19:59:49 UTC	111	-7.43806	109.2505.	2:59 AM
8	2019-01-14 20:00:32 UTC	112	-7.43806	109.2505.	3:00 AM
9	2019-01-14 20:01:18 UTC	113	-7.43806	109.2505.	3:01 AM
10	2019-01-14 20:01:59 UTC	114	-7.43806	109.2505.	3:01 AM
11	2019-01-14 20:02:45 UTC	115	-7.43806	109.2505.	3:02 AM
12	2019-01-14 20:07:36 UTC	116	-7.43804	109.2505.	3:07 AM
13	2019-01-14 20:08:22 UTC	117	-7.43804	109.2505.	3:08 AM
14	2019-01-14 20:09:04 UTC	118	-7.43804	109.2505.	3:09 AM
15	2019-01-14 20:09:49 UTC	119	-7.43804	109.2505.	3:09 AM
16	2019-01-14 20:10:33 UTC	120	-7.43804	109.2505.	3:10 AM

Gambar 3.20 Tampilan data *excel* dari *thingspeak*

Hasil dari grafik pada gambar dapat di*export* kedalam bentuk format *excel* seperti ditunjukan pada gambar 3.20 yang terdapat deskripsi waktu, tanggal, *latitude* dan *longitude* dari sistem monitoring angkutan umum.

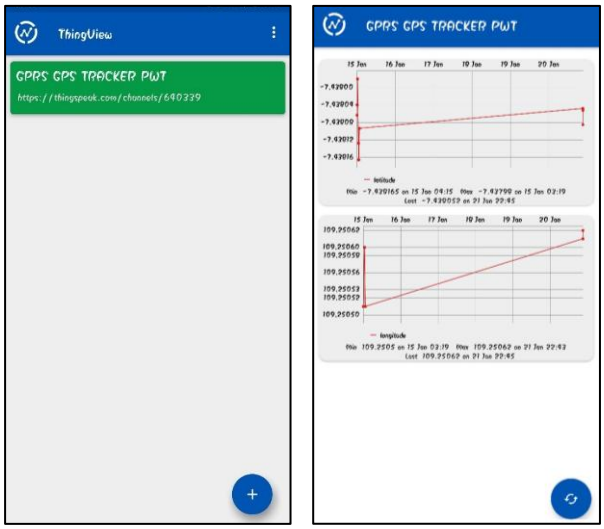


Gambar 3.21 *API Keys*

Untuk mendapatkan *Key* akses *Write* data melalui menu *API Keys* kemudian *copy key* pada bagian *Write API Key*. Sedangkan *Key Read API Keys* digunakan untuk membaca data. *Channel ID* akan digunakan untuk menampilkan data melalui aplikasi *android* seperti yang ditunjukan pada gambar 3.21.

C. Thingview

Thingview merupakan aplikasi android untuk memvisualisasikan saluran *Thingspeak* dengan cara yang lebih simpel, yaitu dengan cara memasukan *channel* ID, seperti pada gambar 3.22 menunjukan tampilan awal yang mengarah pada *channel* GPRS GPS TRACKER PWT. Tampilan dibawah ini merupakan tampilan *channel* GPRS GPS TRACKER PWT yang terdapat dua buah *field*. *Field* yang muncul pada *channel Thingview* sesuai dengan *channel* pada *Thingspeak*, dimana terdapat *field latitude* dan *field longitude* yang datanya diambil dari data pada *Thingspeak* dengan menggungakan *channel* ID API *Key channel* tersebut.

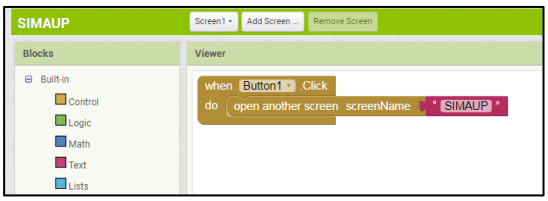


Gambar 3.22 *Thingview*

D. *App Inventor*

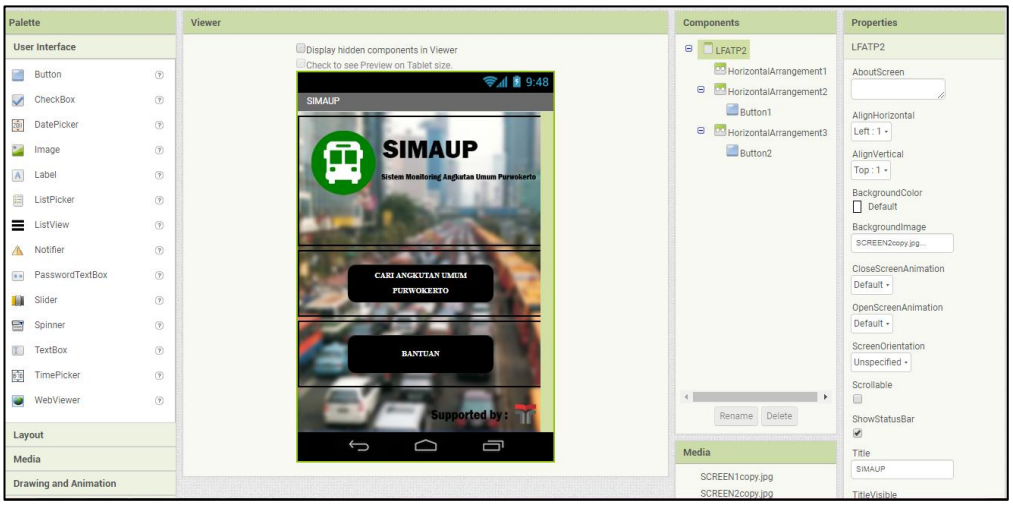
App Inventor digunakan untuk pembuatan aplikasi di *Android*. *App Inventor* ini merupakan alat pengembangan yang mudah digunakan oleh siapa saja dengan menggunakan pendekatan blok. Adapun desain layar dilakukan dengan pendekatan "*click* & *drag*". [22]

Gambar 3.23 Tampilan *designer screen* 1

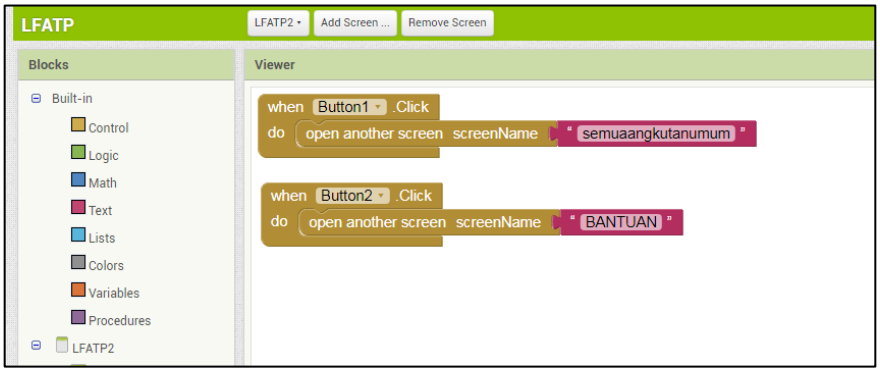


Gambar 3.24 Tampilan *block screen* 1

Pada gambar 3.23 merupakan desain tampilan awal dari aplikasi SIMAUP atau Sistem Monitoring Angkutan Umum Purwokerto yang merupakan aplikasi yang digunakan untuk melakukan monitoring angkutan umum di wilayah purwokerto, dimana pada tampilan awal dari aplikasi SIMAUP tersebut terdapat tombol 'MULAI'. Saat tombol 'MULAI' diklik maka akan dilakukan perintah sesuai *block screen* 1 pada gambar 3.24, yaitu perintah untuk membuka ke *screen* dengan nama SIMAUP.



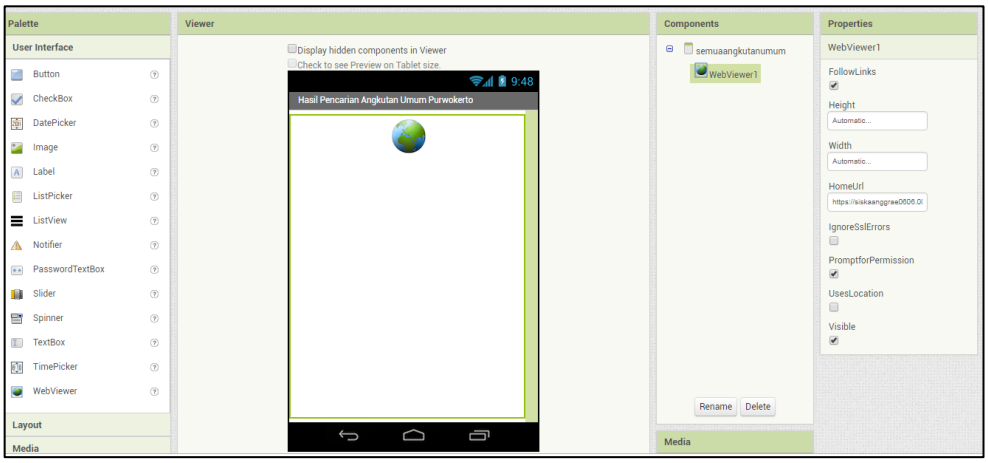
Gambar 3.25 Tampilan *designer* SIMAUP



Gambar 3.26 Tampilan *block* SIMAUP

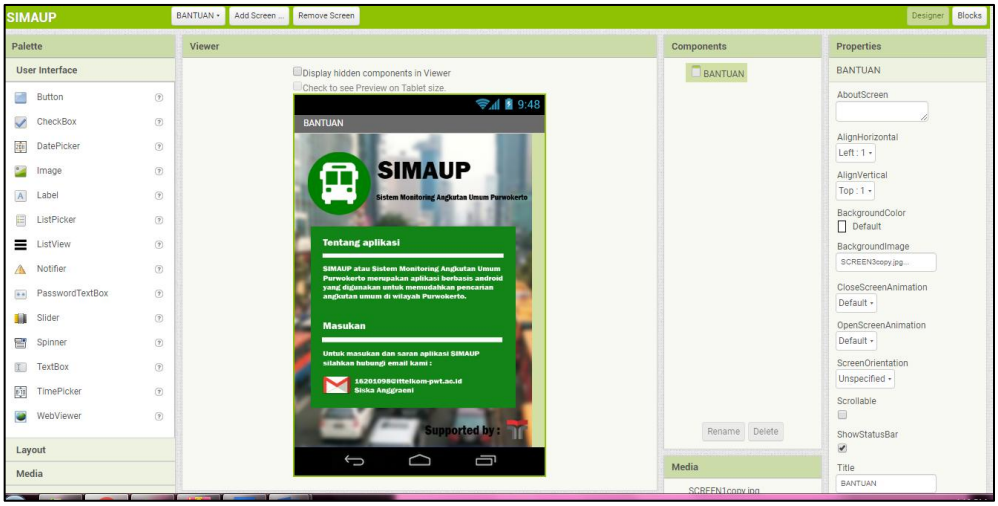
Pada gambar 3.25 merupakan tampilan *designer* SIMAUP, dimana pada tampilan tersebut merupakan menu untuk pilihan pencarian angkutan umum. Tiap tombol akan melakukan perintah sesuai pada tampilan *block* SIMAUP yang ditunjukan pada gambar 3.26 yaitu jika button 1 'CARI ANGKUTAN UMUM PURWOKERTO' di klik maka akan membuka *screen*

dengan nama “semuaangkutanumum”, selanjutnya jika button 2 ‘BANTUAN’ di klik maka akan membuka *screen* dengan nama BANTUAN.



Gambar 3.27 Tampilan *designer screen* web

Pada gambar 3.27 merupakan tampilan *designer screen* dengan nama angkutanumumpurwokerto, dimana pada desain tersebut menggunakan *user interface* berupa *web view*. Pada *screen* “semuaangkutanumum”, *web view* akan diarahkan ke url peta untuk tampilan posisi angkutan keseluruhan yang ada di area purwokerto. Dimana untuk menampilkan peta tersebut akan menuju web url dengan settingan *javascript* yang tertera pada lampiran 2.



Gambar 3.28 Tampilan *designer screen* bantuan

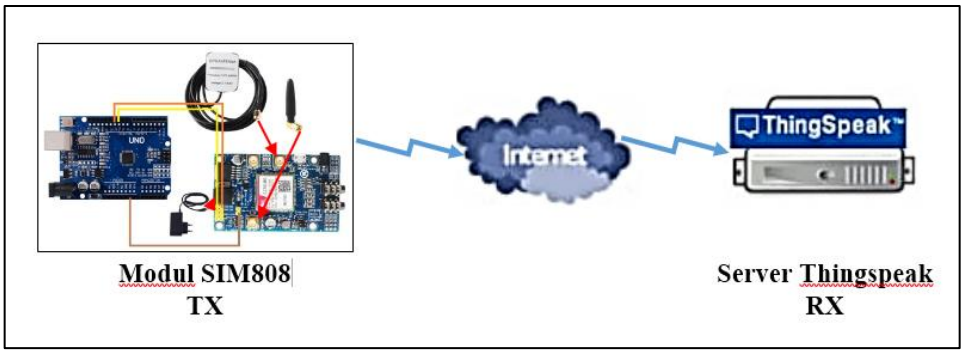
Pada gambar 3.28 diatas merupakan tampilan jika tombol ‘BANTUAN’ pada *screen* SIMAUP diklik. Pada menu tersebut menunjukan

keterangan mengenai aplikasi SIMAUP, dan juga terdapat keterangan untuk memberi masukan melalui kontak email.

3.3 SKEMA PENGUJIAN AKURASI PERANGKAT GPS MODUL SIM808 PADA KONDISI *INDOOR* DAN *OUTDOOR*

Pengujian akurasi dilakukan dengan cara membandingkan salah satu titik koordinat dengan perangkat GPS modul SIM808 dengan koordinat sebenarnya yaitu menggunakan GARMIN GPS MAP64s. Pengujian ini dilakukan 30 kali pengambilan data koordinat pada titik lokasi yang berbeda yaitu 15 kali pada kondisi *indoor* atau didalam ruangan dan 15 kali berikutnya pada kondisi *outdoor* atau diluar ruangan. Selanjutnya hasil pengambilan titik koordinat tersebut dilakukan pengukuran selisih jarak dari koordinat dengan perangkat GPS modul SIM808 dengan koordinat sebenarnya. Menggunakan *ruler* pada *google earth*. Pengujian ini bertujuan untuk mengetahui seberapa akurat titik koordinat yang didapatkan perangkat antenna GPS pada modul SIM808.

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



Gambar 3.29 Bagan untuk pengujian parameter QoS (*Quality of Service*)

Dari gambar 3.29 merupakan bagan untuk pengujian parameter QoS pada penelitian ini, dimana modul SIM808 sebagai sisi *transmitter* dan server *Thingspeak* sebagai sisi *receiver*. Pengujian parameter QoS yang akan dilakukan yaitu meliputi pengujian *packet loss* dan *delay* berdasarkan data yang diterima disisi *receiver* atau server *thingspeak*.

3.4.1 SKEMA PENGUJIAN *PACKET LOSS*

Pengujian *packet loss* digunakan untuk mengetahui jumlah total *packet* pengiriman data koordinat yang hilang dari sisi tx ke sisi rx. Proses pengujian *packet loss* pada penelitian ini dilakukan dengan pengujian pengiriman data dari tx ke rx selama 5 menit dan dilakukan sebanyak 30 kali pengujian. Dari tiap pengujian ditentukan jumlah total *packet loss* yang dapat diketahui 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* pada server *thingspeak*. Jika data yang diterima di sisi rx sesuai dengan data yang dikirim dari sisi tx menunjukan bahwa tidak adanya data yang hilang, tetapi jika data yang diterima di sisi rx lebih sedikit dari data yang dikirim dari sisi tx menunjukan bahwa adanya data yang hilang.

3.4.2 SKEMA PENGUJIAN *DELAY*

Pengujian *delay* dilakukan untuk mengetahui waktu tunda suatu paket. Pada pengujian *delay* penelitian ini merupakan data *delay* dari selisih waktu data awal yang diterima di sisi rx ke data terima selanjutnya. Proses pengujian *delay* pada penelitian ini dilakukan dengan pengujian pengiriman data dari sisi tx ke sisi rx selama 5 menit dan dilakukan sebanyak 30 kali pengujian. Data *delay* diambil dari data yang diterima di sisi rx yaitu pada CSV *file* di server *thingspeak*. Dari tiap pengujian ditentukan nilai *delay* tiap data yang diterima dan dilakukan perhitungan rata-rata *delay* di tiap pengujian, selanjutnya diambil nilai data rata-rata *delay* secara keseluruhan.