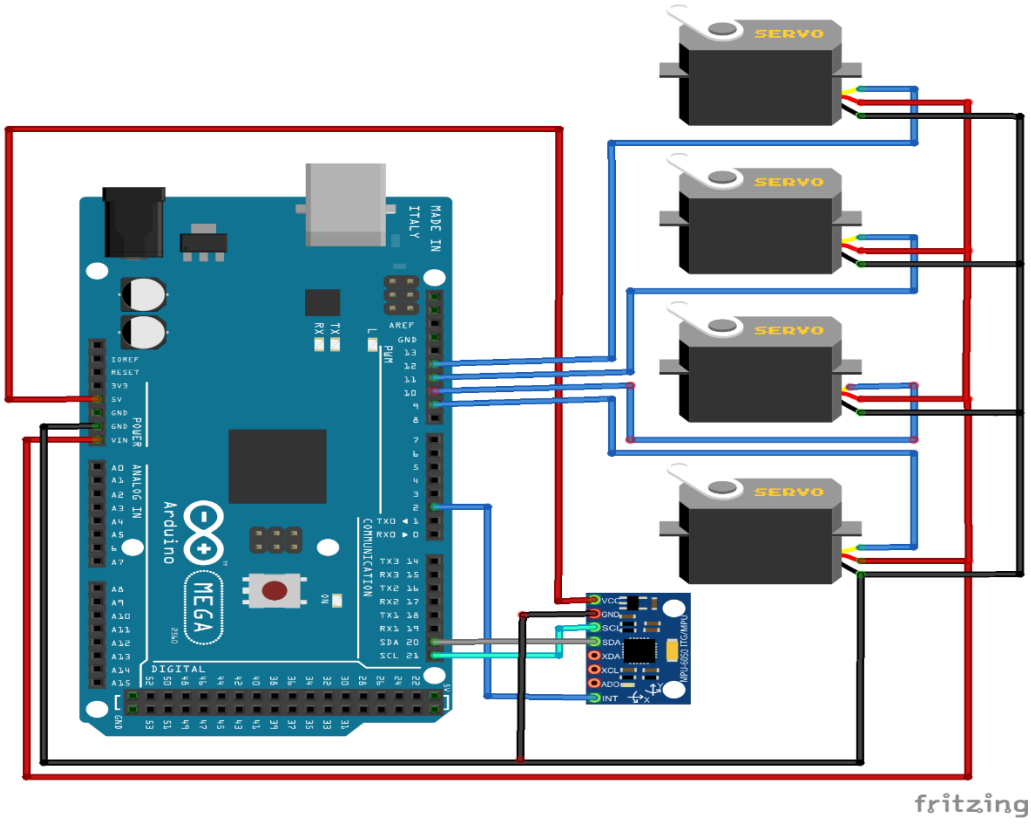
BAB IV
HASIL DAN PEMBAHASAN

Bab ini akan membahas hasil dari perancangan dan implementasi mikrokontroller arduino 2560 untuk pengendalian gerakan body stabiliser control pada model kendaraan roda empat.

4.1 Diagram Sirkuit Pemrograman

Dalam setiap suatu perancangan pemrograman pasti terdapat diagram sirkuit, karena merupakan suatu gambaran atau petunjuk tentang komponen apa saja yang ada didalam suatu rangkaian beserta susunannya. Begitu juga pada perancangan kali ini terdapat diagram sirkuit pemrograman yang berfungsi untuk mempermudah dalam proses perangkaian. Diagram sirkuit pemrograman yang digunakan dapat dilihat pada gambar 4.1 berikut.



Gambar 4.1 Diagram sirkuit pemrograman

Diagram sirkuit pemrograman diatas terdiri dari rangkaian antara Arduino Mega 2560, motor servo dan sensor MPU6050. Penjelasan dari rangkaian diagram sirkuit pemrograman pada gambar 4.1 tersebut yaitu :

Pada keluaran MPU6050 VCC terhubung ke 5V Arduino Mega 2560, SCL terhubung dengan SCL pin 21 Arduino Mega 2560, SDA terhubung dengan SDA pin 20 Arduino Mega 2560, INT terhubung dengan pin 2 Arduino Mega 2560, ground (GND) terhubung ke ground (GND) Arduino Mega 2560 dan terhubung dengan ground masing-masing servo (kabel warna coklat). Jadi terdapat 5 output keluaran dari MPU6050.

Kemudian pada tiap-tiap servo terbagi menjadi tiga keluaran yaitu kabel warna jingga servo terhubung pada pin 9 sampai 12 pada Arduino Mega 2560, kabel warna merah servo terhubung dengan VIN pada Arduino Mega 2560 dan kabel warna coklat servo terhubung dengan ground (GND) pada MPU6050 dan pada Arduino Mega 2560. Jika terdapat salah satu rangkaian yang tidak sesuai dengan diagram sirkuit pemrograman maka komponen hardware dapat rusak atau sistem tidak berjalan.

4.2 Analisa Kode Pemrograman

Pada kode pemrograman setiap kalimat memiliki makna atau kegunaan tersendiri, seperti dijelaskan dibawah ini.

```
#include <MPU6050_tockn.h>

#include <Wire.h>

#include <Servo.h>
```

Kode program di atas merupakan library untuk sensor MPU6050 dan motor servo, tanpa adanya library, kode pemrograman tidak dapat digunakan.

```
Servo myservo1;  // create servo object to control a servo
```

```
Servo myservo2;  // create servo object to control a servo
Servo myservo3;  // create servo object to control a servo
Servo myservo4;  // create servo object to control a servo
```

Pemberian variabel nama dari setiap motor servo pada kode pemrograman, seperti tertera pada kode pemrograman di atas. Dari kode pemrograman di atas di diketahui jika variabel nama motor servo yaitu: my servo 1, my servo 2, my servo 3 dan my servo 4. Pemberian variabel nama bertujuan agar posisi servo yang terpasang pada papan *platform* sesuai dengan rangkaian yang telah dibuat dan sebagai pembantu ketika menggunakan fungsi library motor servo.

```
int sudut_servo = 0;
int sudut_servo_1 = 0;
int sudut_servo_2 = 0;
int sudut_servo_3 = 0;
int sudut_servo_4 = 0;
```

Kode pemrograman di atas merupakan variabel-variabel sudut yang akan digunakan pada motor servo, tipe variabel yang digunakan adalah int *(integer)* dan nilainya sebesar nol saat pertama kali dijalankan. Pemberian nilai nol pada saat pertama kali dijalankan agar posisi sudut motor servo tepat pada 0°, sehingga kerataan papan *platform* bagian atas dapat tercapai dan memudahkan dalam mengamati data perubahan sudut yang terjadi.

```
int sumbu_y;
int sumbu_x;
int sumbu_z;
MPU6050 mpu6050(Wire);
```

Hampir sama dengan kode pemrograman sebelumnya, pada kode pemrograman di atas merupakan variabel-variabel sumbu yang terdapat pada MPU6050. Tipe variabel yang digunakan sama seperti tipe variabel sudut servo yaitu int *(integer).* Penggunaan varabel *integer* dikarenakan variabel yang sering digunakan dan dapat menyimpan data sebesar 2 bytes (16 bit). Kode pemrograman tersebut berguna sebagai awalan sebelum menggunakan fungsi library dari MPU 6050.

```
void setup() {
  Serial.begin(9600);
  Wire.begin();

  mpu6050.calcGyroOffsets(true);
  myservo1.attach(9);
  myservo2.attach(10);
  myservo3.attach(11);
  myservo4.attach(12);
}
```

Kode pemrograman di atas merupakan bentuk *setting* awal dari sensor dan pin motor servo yang digunakan pada Arduino Mega 2560 R3. Pin yang digunakan adalah pin 9, pin 10, pin 11 dan pin 12. Penggunaan fungsi *void setup* bertujuan agar kode pemrograman yang dibuat hanya sekali dibaca atau dijalankan oleh Arduino, dikarenakan sebagai kode perintah untuk menentukan fungsi pada sebuah pin.

```
void loop() {
mpu6050.update();
```

```
sumbu_y=mpu6050.getAngleY();
sumbu_x=mpu6050.getAngleX();
```

Kemudian kode pemrograman di atas merupakan pemberian nama dan tipe sensor. Variabel ini akan membantu dalam menggunakan fungsi library dan sebagai pengambilan data sumbu x dan y. Pada sumbu x dan y didalamnya dapat berisikan nilai positif atau negatif, sehingga gerakan yang terjadi pada papan *platform* berupa gerak *pitching* dan *rolling*.

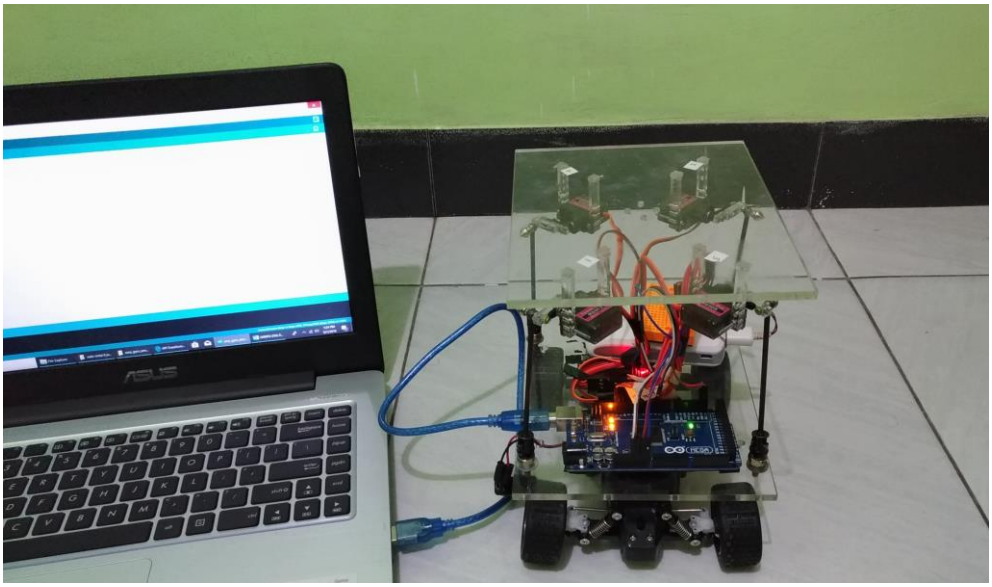
```
Serial.print(sudut_x);
  Serial.print("\t");
  Serial.println(sudut_y);

  sudut_servo_1=map(sudut_x,20,-20,0,180);
  sudut_servo_2=map(sudut_x,-20,20,0,180);
  sudut_servo_3=map(sudut_y,20,-20,0,180);
  sudut_servo_4=map(sudut_y,-20,20,0,180);
  myservo1.write(sudut_servo_1);
  myservo2.write(sudut_servo_2);
  myservo3.write(sudut_servo_3);
  myservo4.write(sudut_servo_4);

}
```

Terakhir merupakan kode pemrograman sebagai inti dari pengendalian servo. Setelah mengetahui nilai dari besar sumbu x dan y, maka digunakanlah perintah ‘map’. Posisi sudut -20 derajat sampai 20 derajat merupakan sudut gerakan pada sensor yang mewakili putaran motor servo dari 0 derajat sampai 180 derajat.

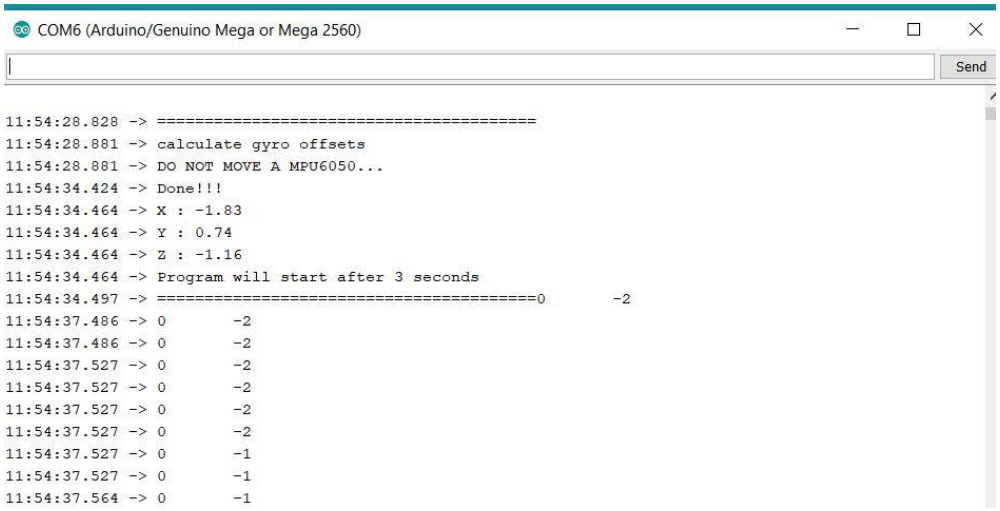
Respon yang terjadi ketika menjalankan perintah ini yaitu gerakan dari *platform* bergerak *pitching* dan *rolling*. Penggunaan fungsi *void loop* bertujuan agar kode pemrograman setelah fungsi *void setup* dapat dibaca atau dijalankan terus menerus oleh Arduino sehingga papan platform dapat bergerak tanpa berhenti kecuali dengan memutus sumber arus yang ada.

4.3 Karakteristik Gerakan *Platform*

Seperti dilihat pada gambar 4.2 yaitu pada pada posisi menyiapkan membutuhkan waktu 3 detik hingga papan *platform* pada posisi 0°. Waktu yang dibutuhkan untuk menyiapkan dapat dilihat pada serial monitor pada menu tools Arduino IDE seperti pada gambar 4.3.



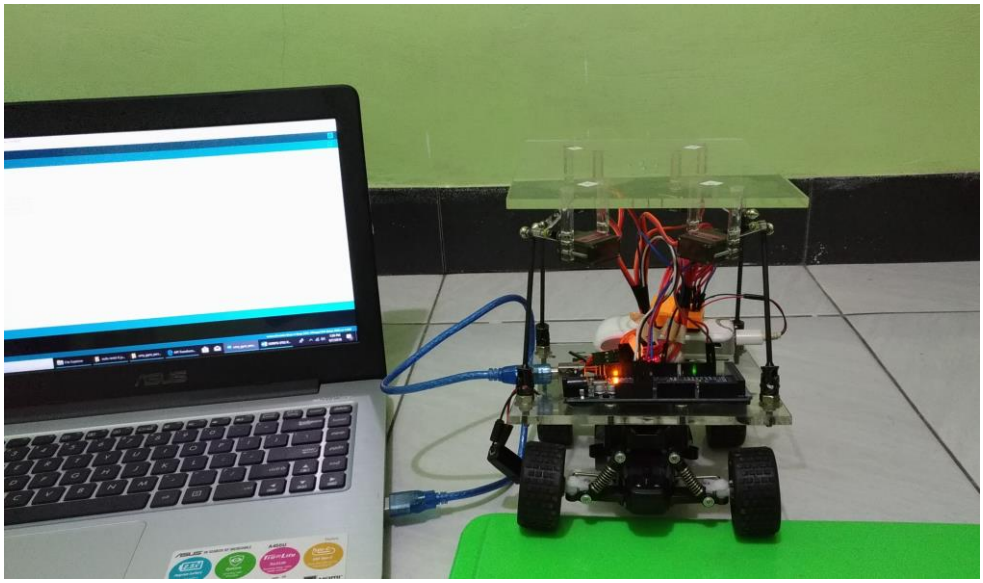
Gambar 4.2 Platform pada posisi menyiapkan



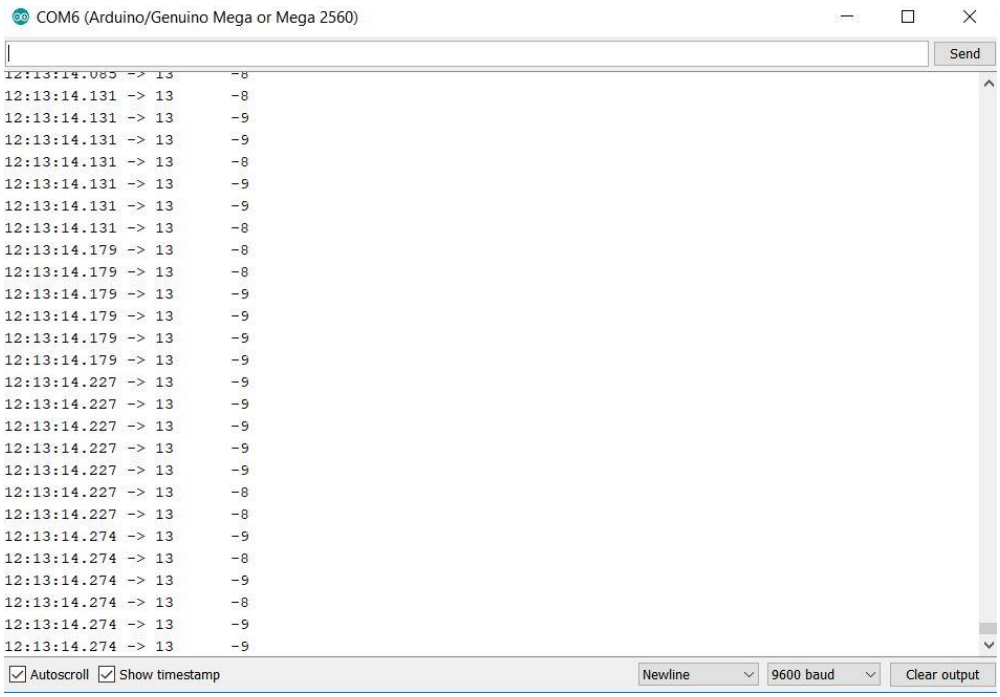
Gambar 4.3 Tampilan serial monitor ketika posisi menyiapkan

Pada posisi *platform* dengan kemiringan mengangguk *(pitching)* seperti ditunjukkan gambar 4.4, posisi sudut yang di peroleh adalah 13° dan dapat dicapai dalam kecepatan 2 detik pada ketinggian hambatan jalan sebesar 45 mm. Sedangkan posisi sudut 10° dapat di capai dalam kecepatan 1 detik pada ketinggian hambatan jalan sebesar 30 mm .Untuk mencapai kecepatan kemiringan 13° dan 10° pada waktu 2 detik dan 1 detik digunakan kode pemrograman seperti dibawah ini.

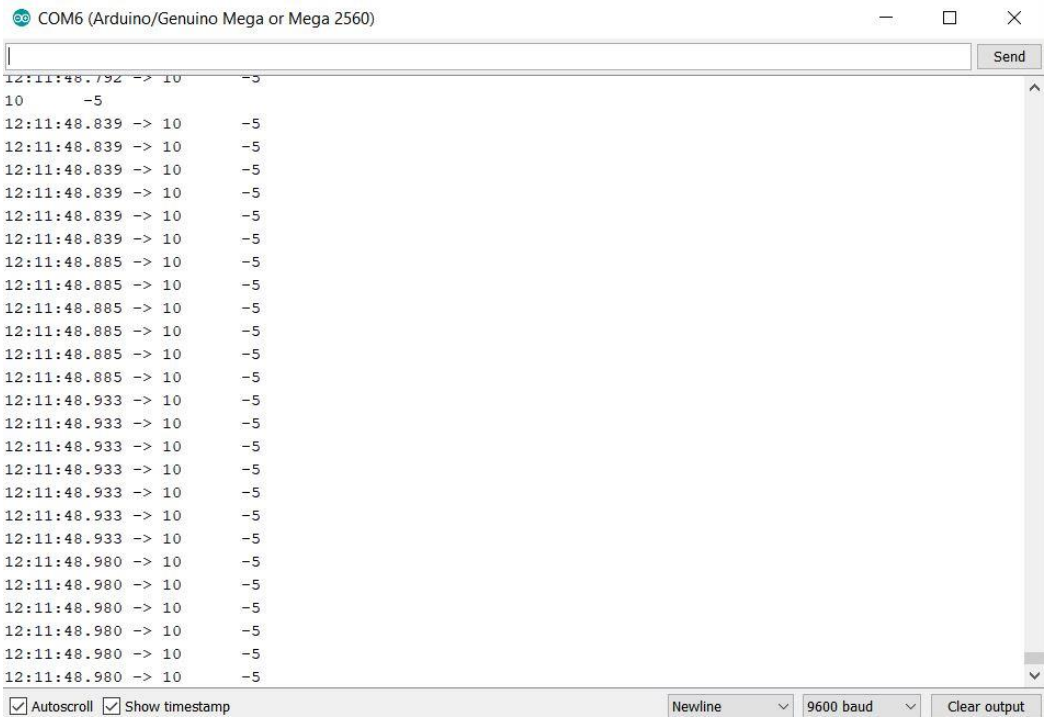
```
sudut_servo_1=map(sudut_x,20,-20,0,180);

sudut_servo_2=map(sudut_x,-20,20,0,180);

sudut_servo_3=map(sudut_y,20,-20,0,180);

sudut_servo_4=map(sudut_y,-20,20,0,180);
```



Gambar 4.4 Posisi *platform* dengan kemiringan *pitching* pada sudut 13° dan ketinggian hambatan jalan 45 mm.



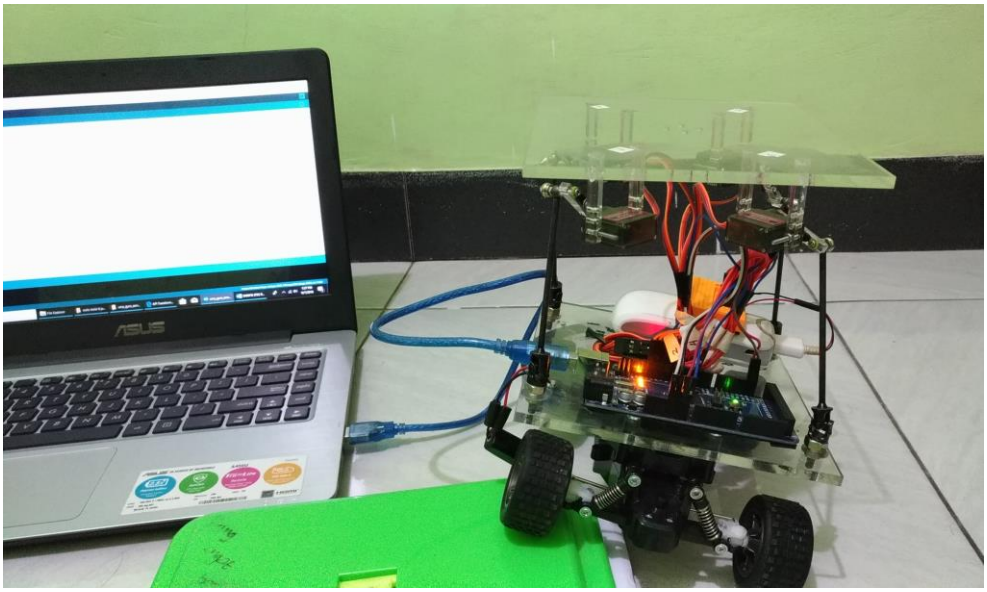
Gambar 4.5 Tampilan serial monitor ketika posisi *pitching* pada tinggi hambatan jalan 45 mm.



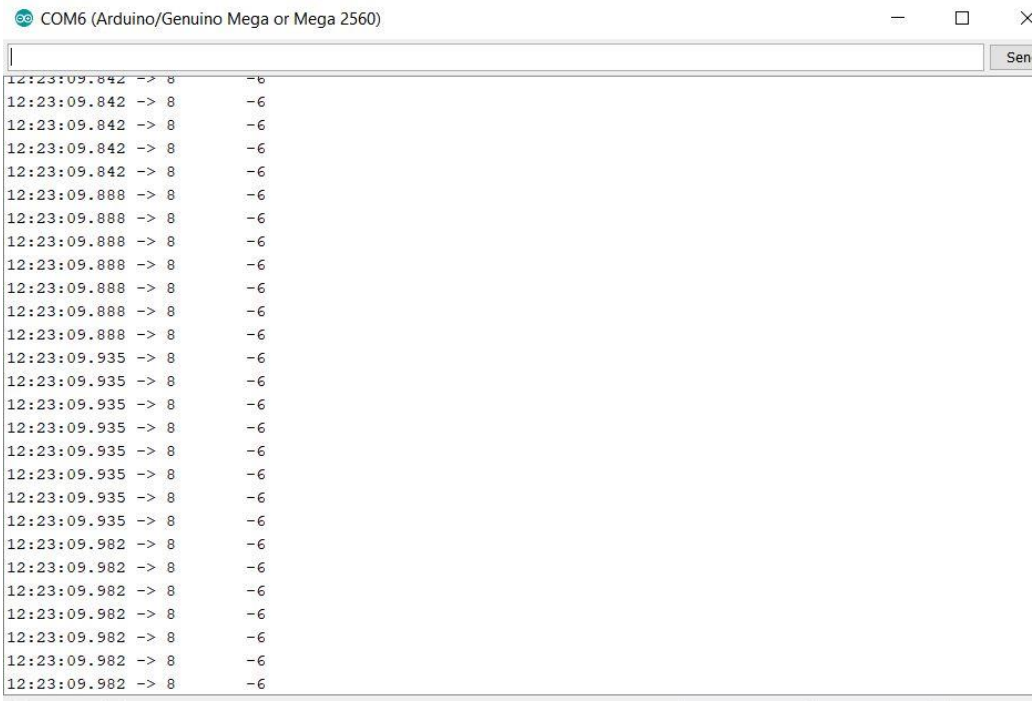
Gambar 4.6 Tampilan serial monitor ketika posisi *pitching* pada tinggi hambatan jalan 30 mm.

Sedangkan pada posisi *platform* dengan kemiringan diagonal roda kanan depan terangkat seperti terlihat pada gambar 4.7 posisi sudut yang di peroleh adalah 8° dan dapat dicapai dalam kecepatan 2 detik pada ketinggian hambatan jalan sebesar 45 mm. Sedangkan posisi sudut 4° dapat di capai dalam kecepatan 1 detik pada ketinggian hambatan jalan sebesar 30 mm .Untuk mencapai kecepatan kemiringan 8° dan 4° pada waktu 2 detik dan 1 detik digunakan kode pemrograman seperti dibawah ini.

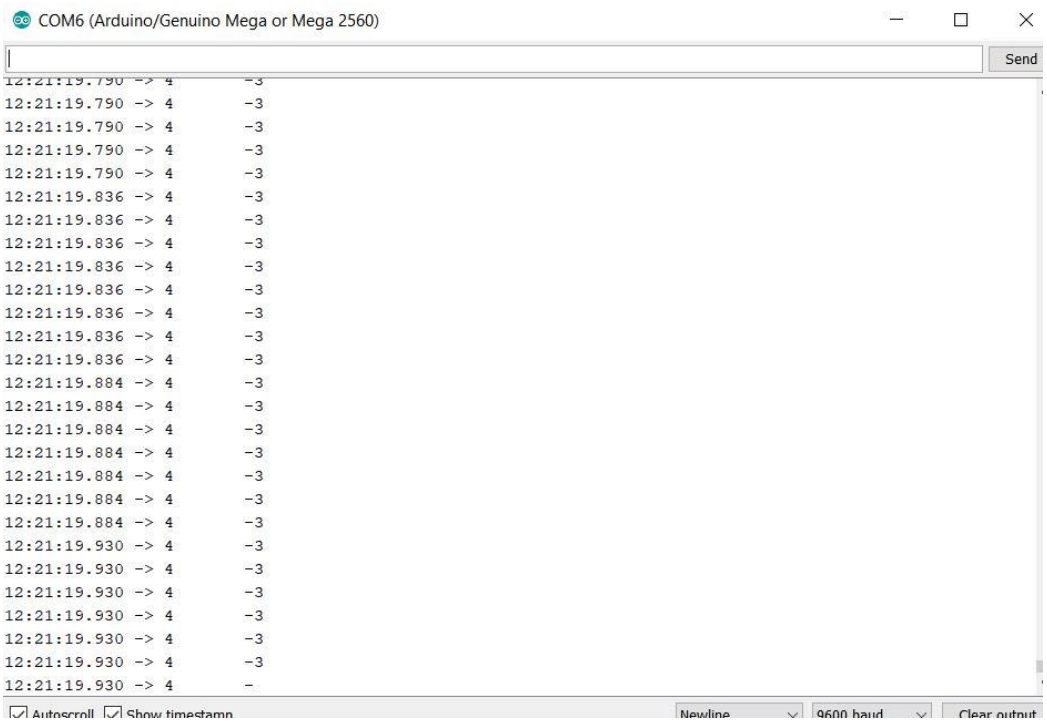
```
sudut_servo_1=map(sudut_x,20,-20,0,180);

sudut_servo_2=map(sudut_x,-20,20,0,180);

sudut_servo_3=map(sudut_y,20,-20,0,180);

sudut_servo_4=map(sudut_y,-20,20,0,180);
```



Gambar 4.7 Posisi *platform* dengan kemiringan diagonal pada sudut 8° dan ketinggian hambatan jalan 45 mm.



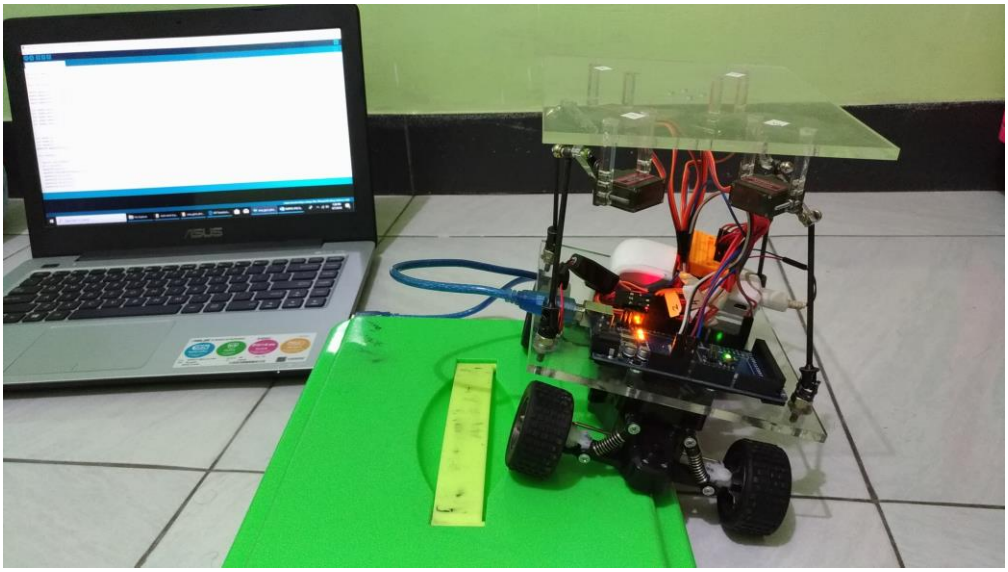
Gambar 4.8 Tampilan serial monitor ketika posisi kemiringan diagonal pada tinggi hambatan jalan 45 mm.



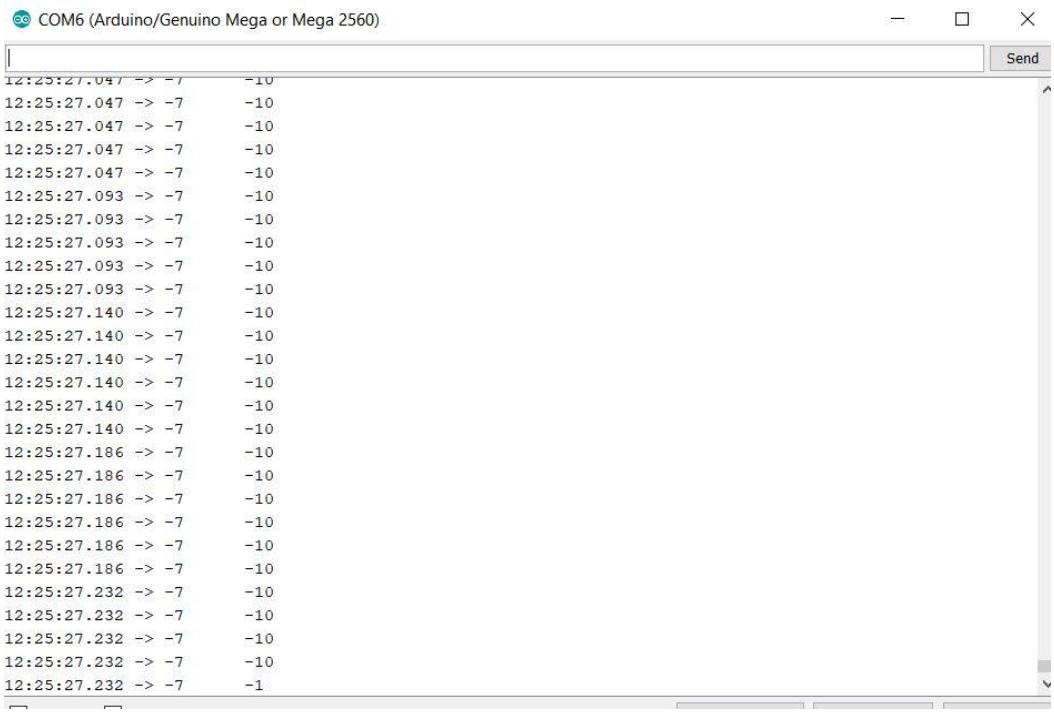
Gambar 4.9 Tampilan serial monitor ketika posisi kemiringan diagonal pada tinggi hambatan jalan 30 mm.

Pada posisi *platform* dengan kemiringan kesamping *(rolling)* seperti ditunjukkan gambar 4.10, posisi sudut yang di peroleh adalah 7° dan dapat dicapai dalam kecepatan 2 detik pada ketinggian hambatan jalan sebesar 45 mm. Sedangkan posisi sudut 4° dapat di capai dalam kecepatan 1 detik pada ketinggian hambatan jalan sebesar 30 mm .Untuk mencapai kecepatan kemiringan 7° dan 4° pada waktu 2 detik dan 1 detik digunakan kode pemrograman seperti dibawah ini.

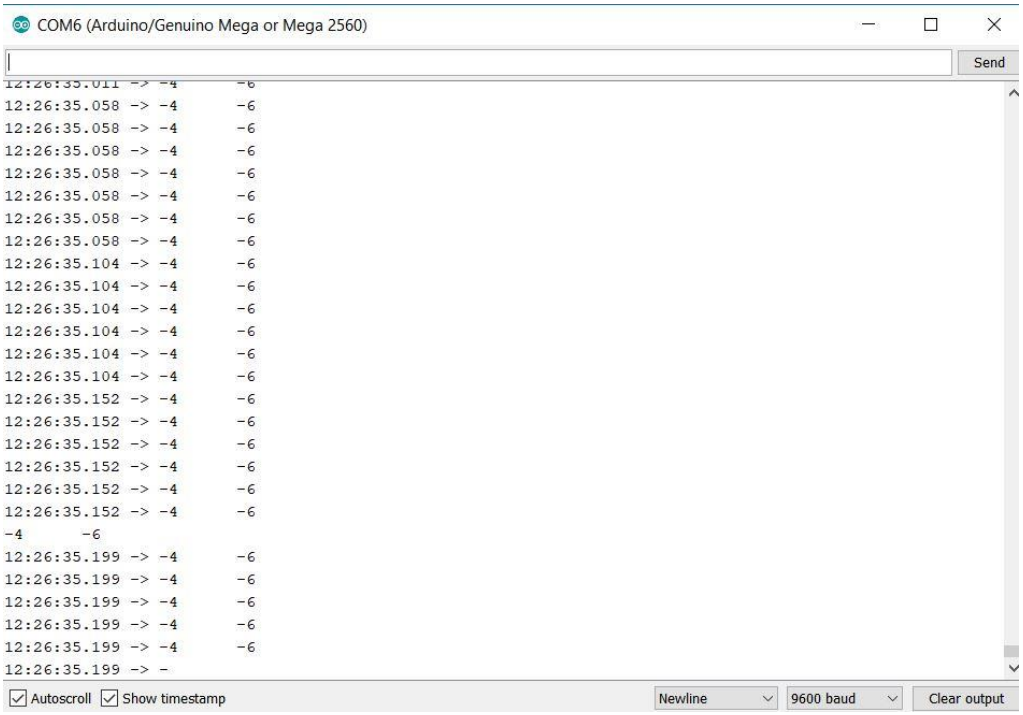
```
sudut_servo_1=map(sudut_x,20,-20,0,180);

sudut_servo_2=map(sudut_x,-20,20,0,180);

sudut_servo_3=map(sudut_y,20,-20,0,180);

sudut_servo_4=map(sudut_y,-20,20,0,180);
```



Gambar 4.10 Posisi *platform* dengan kemiringan *rolling* pada sudut 7° dan ketinggian hambatan jalan 45 mm.



Gambar 4.11 Tampilan serial monitor ketika posisi *rolling* pada tinggi hambatan jalan 45 mm.



Gambar 4.12 Tampilan serial monitor ketika posisi *rolling* pada tinggi hambatan jalan 30 mm.

Pada saat prototipe dijalankan respon gerakan pada platform terpengaruh oleh kecepatan laju kendaraan, dikarenakan gerakan komponen penggerak RC yang terlalu cepat. Hal ini menyebabkan data pada serial monitor tidak stabil.