BAB IV

HASIL DAN UJI COBA

IV.1. Tampilan Hasil

Dalam bab ini akan dijelaskan dan ditampilkan bagaimana hasil dari pengujian rancangan alat yang dibuat beserta pembahasan tentang pergerakan dan cara kerja perangkat robot penyeimbang menggunakan sensor jarak berbasis android. Adapun hasil dari pengujian yang dilakukan adalah perangkat elektronik yang dibuat atau dirancang dan diprogram dengan menggunakan aplikasi Arduino IDE.

IV.2. Pelaksanaan Pengujian Rangkaian

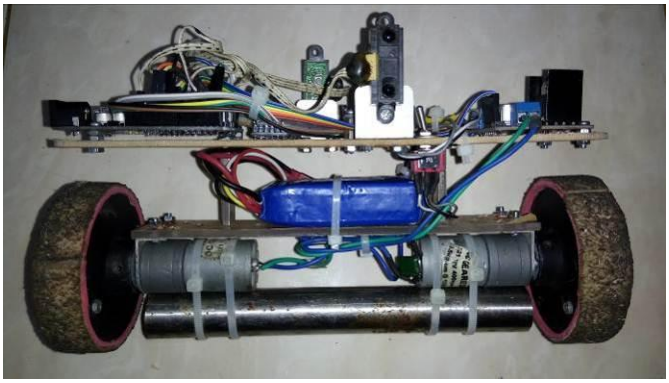
Sebelum melakukan pengujian, beberapa hal yang harus diperhatikan dan dipersiapkan adalah sebagai berikut :

1. Perangkat robot penyeimbang menggunakan sensor jarak dalam keadaan siap diuji, tidak ada *trouble* pada saat pengujian.
2. Sebelum pengujian perangkat, hubungkan *power* (baterai lipo) dengan perangkat dan menghidupkan tombol *power on/off*.
3. Hasil pengujian dianalisa dan dibandngkan dengan perangkat pembanding, seperti menghitung jarak sensor dengan penggaris, menghitung tegangan menggunakan multimeter dan lain sebagainya.
4. Hasil pengujian dipaparkan dalam bentuk tabel dan grafik, dianalisa dan dijelaskan secara terperinci.

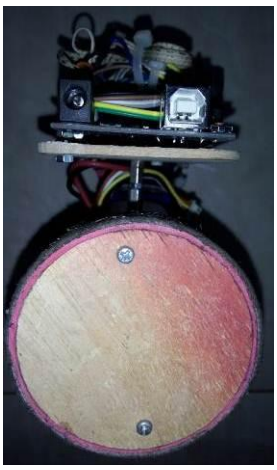
IV.3. Tampilan Hasil Perangkat

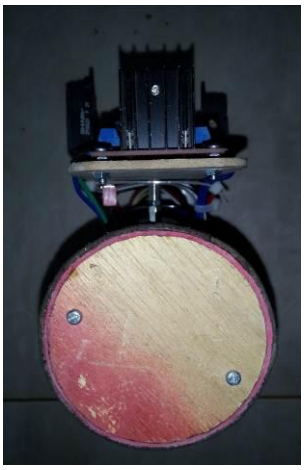
Berikut adalah tampilan hasil perancangan robot penyeimbang menggunakan sensor jarak berbasis android, ditunjukan oleh gambar di bawah ini:

A. Sisi Depan

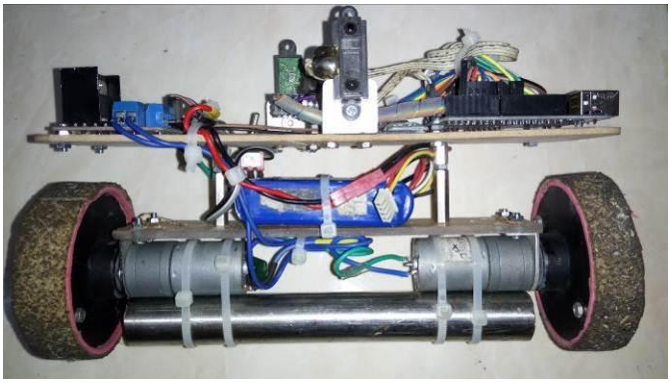


B. Sisi Samping

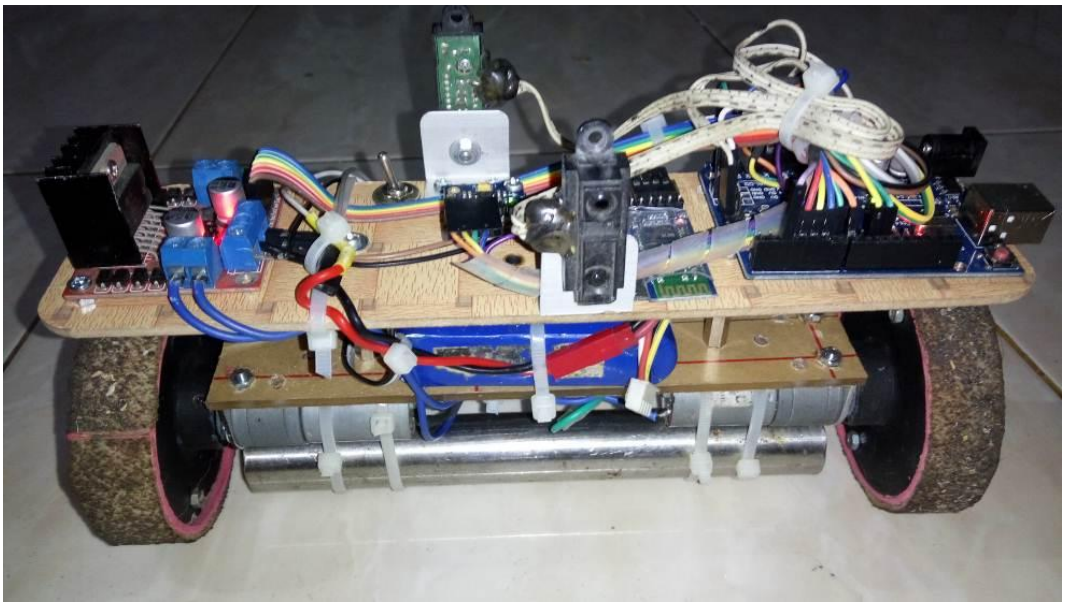




C. Sisi Belakang



D. Hasil Keseluruhan



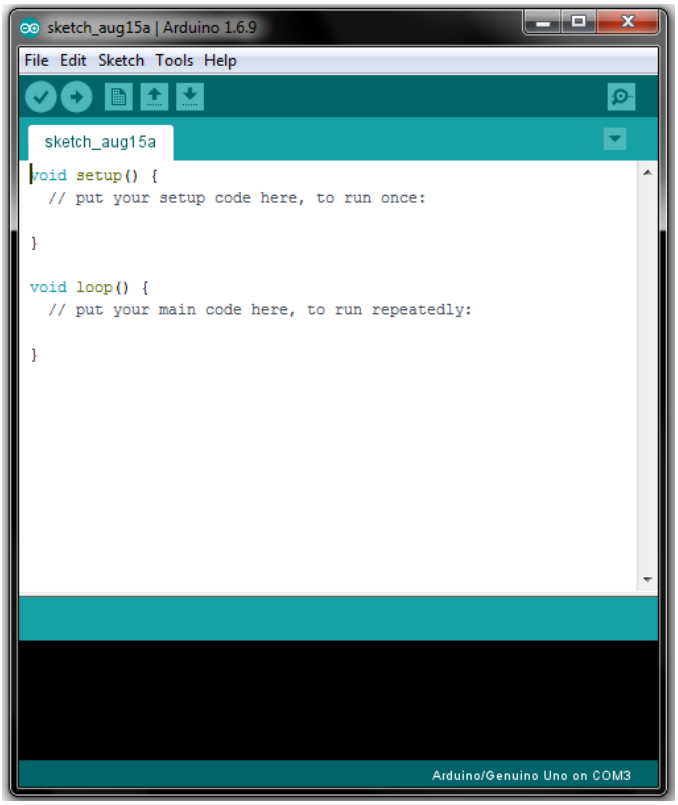
Gambar IV.1. Perangkat Keseluruhan

IV.4. Pengujian *Software*

Untuk mengetahui apakah rangkaian pada perangkat telah bekerja dengan baik, maka dilakukan pengujian dengan memberikan program perintah pada mikrokontroler dengan melakukan penginputan data dari komputer ke dalam mikrokontroler. Sebelum dilakukannya proses *download* program, hubungkan terlebih dahulu antara komputer melalui kabel USB dengan rangkaian mikrokontroler.

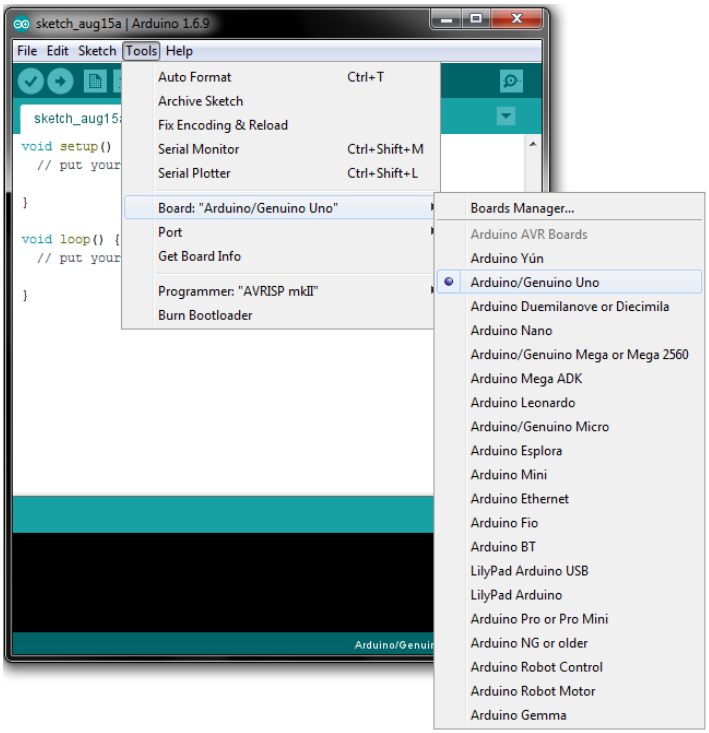
Dalam proses instalasi ini menggunakan aplikasi *Arduino 1.6.9*. Untuk melakukan instalasi ini dapat dilakukan dengan beberapa langkah antara lain :

a. Langkah pertama yang dilakukan adalah dengan mengklik *icon* Arduino. Setelah program melakukan *load* maka akan terlihat bentuk tampilan seperti gambar IV.2 berikut.

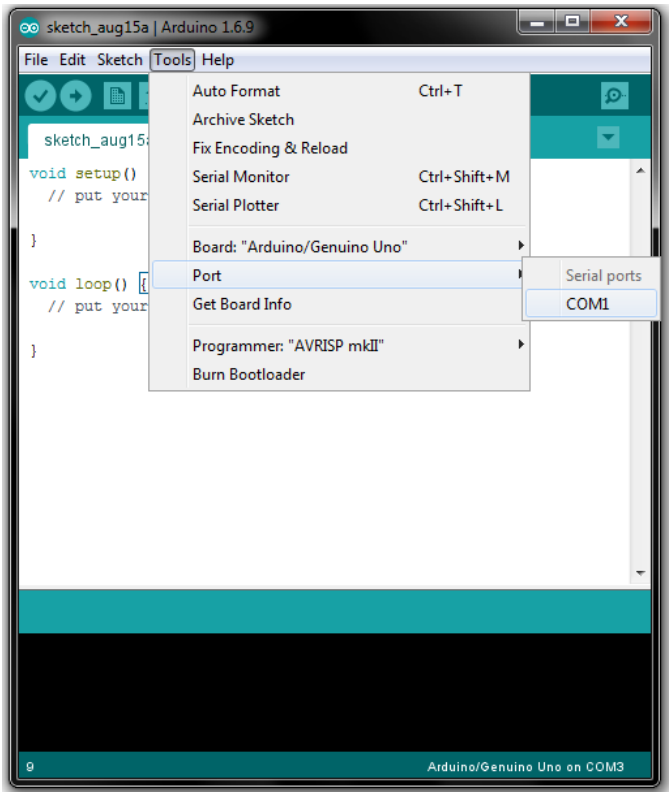


Gambar IV.2. Tampilan Arduino 1.6.9

b. Selanjutnya yang dilakukan sebelum menginstal program terhadap mikrokontroler adalah melakukan pengaturan (*setting*) pada perangkat yang diperlukan dan mengetikkan program sesuai dengan yang dibutuhkan. Pengaturan pertama adalah pemilihan *broad* arduino yang digunakan pada *software* sesuai dengan perangkat yaitu Arduino UNO, seperti pada gambar IV.3. Pengaturan kedua adalah pemilihan *port USB* yang digunakan perangkat, seperti pada gambar IV.4. berikut :



Gambar IV.3. Pengaturan dan Pemilihan *Board* Arduino



Gambar IV.4. Pengaturan *Port USB* pada *Software* Arduino 1.6.9

c. Setelah pengaturan selesai, proses berikutnya adalah penulisan *listing* program. Berikut adalah *listing* program dari perangkat robot penyeimbang menggunakan sensor jarak berbasis android:

```
//*******************************************************
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
#endif

#define LOG_INPUT 0
#define MANUAL_TUNING 0
#define LOG_PID_CONSTANTS 0 //MANUAL_TUNING must be 1
#define MOVE_BACK_FORTH 0
#define MIN_ABS_SPEED 30

#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX

//MPU
MPU6050 mpu;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 =
error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

//PID
#if MANUAL_TUNING
  double kp , ki, kd;
  double prevKp, prevKi, prevKd;
```

```
#endif
double originalSetpoint = 172.2;
double setpoint = originalSetpoint;
double movingAngleOffset = 1;
double input, output;
int moveState=0; //0 = balance; 1 = back; 2 = forth
char perintah;

#if MANUAL_TUNING
  PID pid(&input, &output, &setpoint, 0, 0, 0, DIRECT);
#else
  PID pid(&input, &output, &setpoint, 20, 80, 1.3, DIRECT);
#endif

//MOTOR CONTROLLER
int ENA = 3;
int IN1 = 4;
int IN2 = 8;
int IN3 = 5;
int IN4 = 7;
int ENB = 6;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, 0.6, 1);

//timers
long time1Hz = 0;
long time5Hz = 0;

volatile bool mpuInterrupt = false;      // indicates whether MPU interrupt pin has
gone high
void dmpDataReady()
{
   mpuInterrupt = true;
}

void setup()
{

   //motorController.move(150,150,10);
   //delay(1000);
   //motorController.move(0,0,10);
   ///delay(1000);
   //motorController.move(-150,-150,10);
   ///delay(1000);
   //motorController.move(0,0,10);
   //delay(1000);
   // join I2C bus (I2Cdev library doesn't do this automatically)
```

```
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (115200 chosen because it is required for Teapot Demo output, but it's
    // really up to you depending on your project)
    Serial.begin(115200);
    mySerial.begin(9600);
    while (!Serial); // wait for Leonardo enumeration, others continue immediately

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(682);
    mpu.setYGyroOffset(524);
    mpu.setZGyroOffset(-271);
    mpu.setZAccelOffset(25816); // 1688 factory default for my test chip

    // make sure it worked (returns 0 if so)
    if (devStatus == 0)
    {
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection (Arduino external interrupt
0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
```

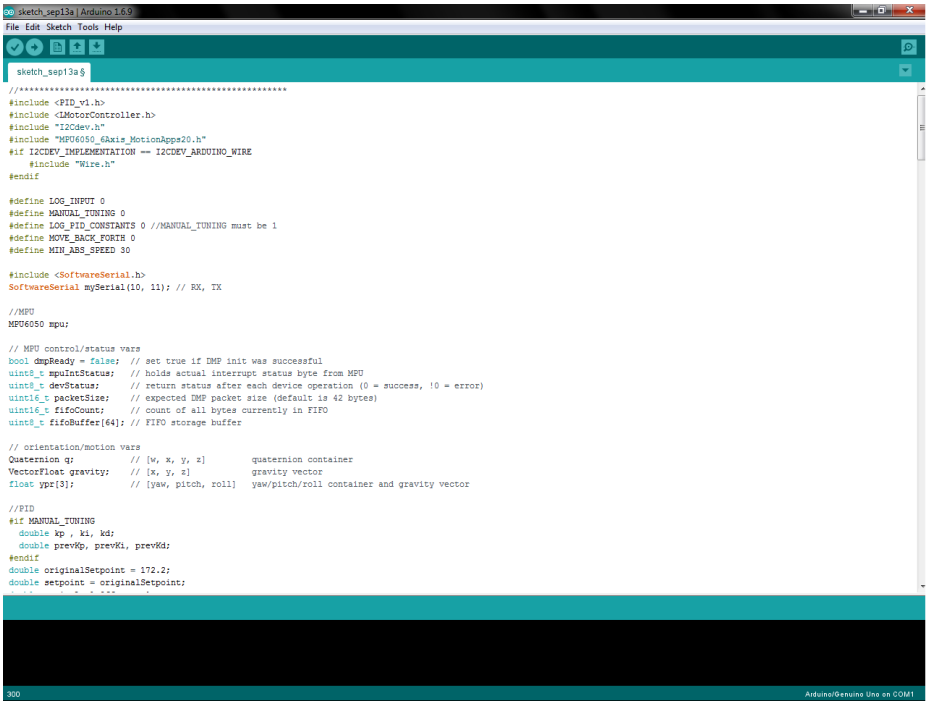
```
        // set our DMP Ready flag so the main loop() function knows it's okay to use
it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();

        //setup PID
        pid.SetMode(AUTOMATIC);
        pid.SetSampleTime(10);
        pid.SetOutputLimits(-255, 255);
    }
    else
    {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }
}

void loop()
{
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize)
    {
        //no mpu data - performing PID calculations and output to motors
        pid.Compute();
        motorController.move(output, MIN_ABS_SPEED);

        unsigned long currentMillis = millis();
        if (currentMillis - time1Hz >= 1000)
        {
            loopAt1Hz();
            time1Hz = currentMillis;
        }

        if (currentMillis - time5Hz >= 5000)
        {
```

```
            loopAt5Hz();
            time5Hz = currentMillis;
        }
    }

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024)
    {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt (this should happen frequently)
    }
    else if (mpuIntStatus & 0x02)
    {
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;

        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        #if LOG_INPUT
            Serial.print("ypr\t");
            Serial.print(ypr[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[2] * 180/M_PI);
        #endif
```

```
        input = ypr[1] * 180/M_PI + 180;

        if (mySerial.available()) {
    perintah = mySerial.read();
    if(perintah=='F'){
    Serial.println("F");
    motorController.move(150,150,10);  delay(50);
    }
    else if(perintah=='B'){
    Serial.println("B");
    motorController.move(-150,-150,10);   delay(50);
    }
    else if(perintah=='L'){
    Serial.println("L");
    motorController.move(-150,150,10);  delay(50);
    }
    else if(perintah=='R'){
    Serial.println("R");
    motorController.move(150,-150,10); delay(50);
    }
    else if(perintah=='S'){
    Serial.println("S");
    mpu.resetFIFO();
    }
    }
   }
}

void loopAt1Hz()
{
#if MANUAL_TUNING
    setPIDTuningValues();
#endif
}

void loopAt5Hz()
{
    #if MOVE_BACK_FORTH
        moveBackForth();
    #endif
}

//move back and forth
void moveBackForth()
{
    moveState++;
```

```
  if (moveState > 2) moveState = 0;
  if (moveState == 0)
    setpoint = originalSetpoint;
  else if (moveState == 1)
    setpoint = originalSetpoint - movingAngleOffset;
  else
    setpoint = originalSetpoint + movingAngleOffset;
}

//PID Tuning (3 potentiometers)
#if MANUAL_TUNING
void setPIDTuningValues()
{
  readPIDTuningValues();
  if (kp != prevKp || ki != prevKi || kd != prevKd)
  {
#if LOG_PID_CONSTANTS
    Serial.print(kp);Serial.print(",          ");Serial.print(ki);Serial.print(",
");Serial.println(kd);
#endif
    pid.SetTunings(kp, ki, kd);
    prevKp = kp; prevKi = ki; prevKd = kd;
  }
}

void readPIDTuningValues()
{
  int potKp = analogRead(A0);
  int potKi = analogRead(A1);
  int potKd = analogRead(A2);

  kp = map(potKp, 0, 1023, 0, 25000) / 100.0; //0 - 250
  ki = map(potKi, 0, 1023, 0, 100000) / 100.0; //0 - 1000
  kd = map(potKd, 0, 1023, 0, 500) / 100.0; //0 - 5
}
#endif

//*******************************************************
```

d. Proses berikutnya adalah melakukan *Verify/Compile* program dan *Upload* program, dengan memilih menu *Sketch -> Upload* pada *software* Arduino 1.6.9, seperti pada gambar di bawah berikut ini :

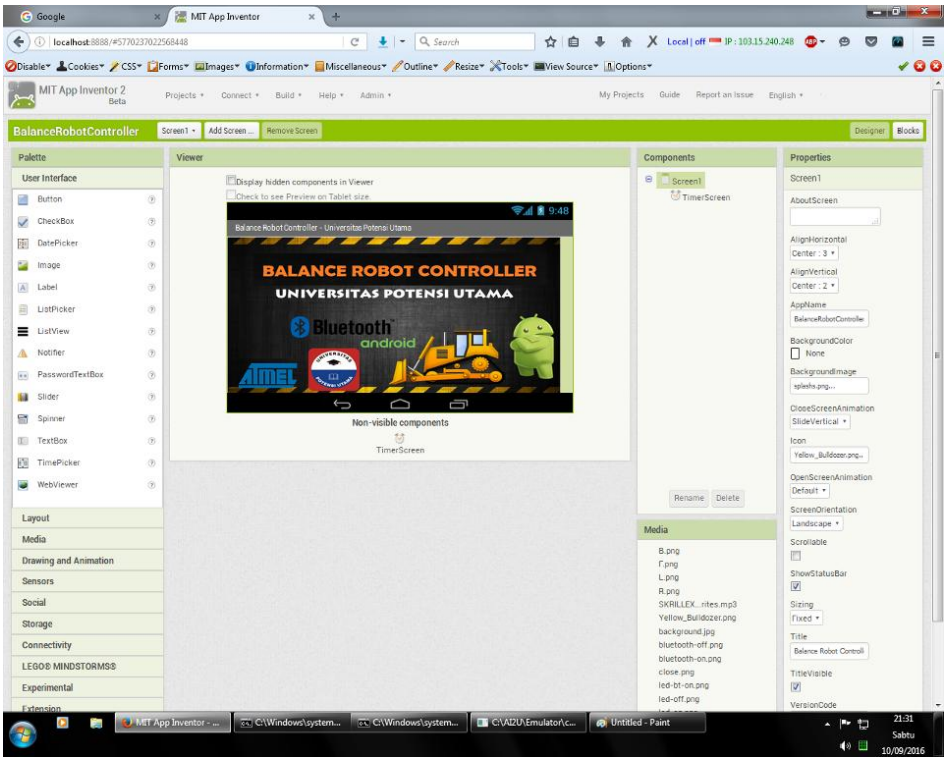


Gambar IV.5. Proses *Upload* Program *Software* Arduino 1.6.9

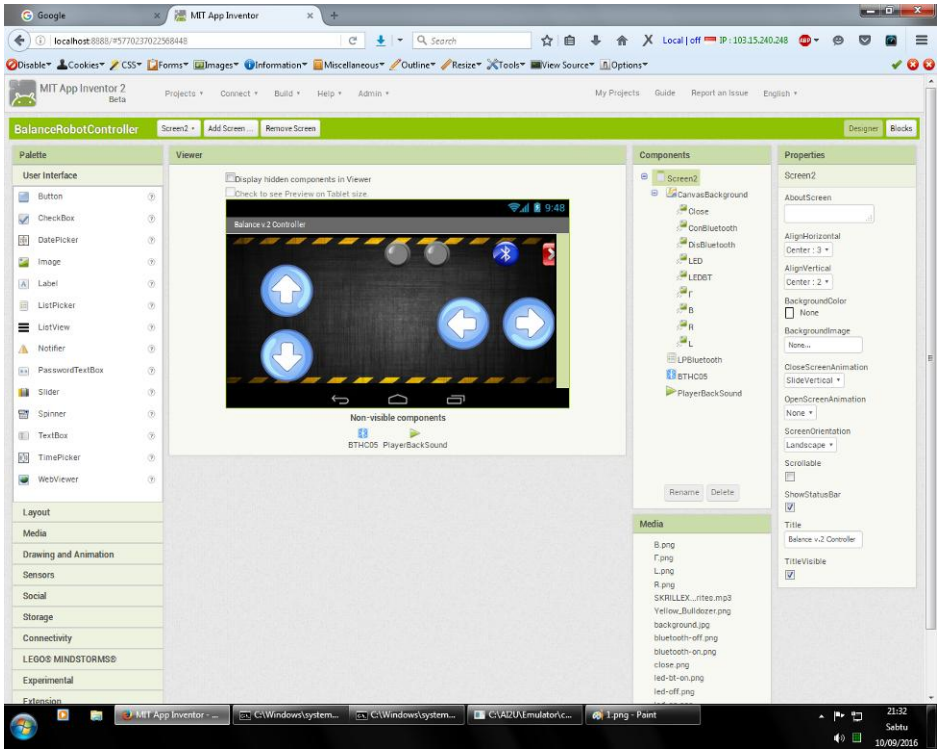
e. Setelah proses *upload* program selesai terhadap rangkaian mikrokontroler, maka dapat dilihat kinerja dari perangkat berjalan sesuai dengan program yang diperintahkan dengan melakukan pengujian perangkat secara *hardware*.

Pengujian *software* berikutnya adalah perancangan aplikasi pengendali robot penyeimbang menggunakan sensor jarak berbasis android. Perancangan menggunakan *software app inventor 2 ultimate*. Berikut adalah hasil dari perancangan dari aplikasi ditampilkan pada gambar berikut :

A. Tampilan Awal

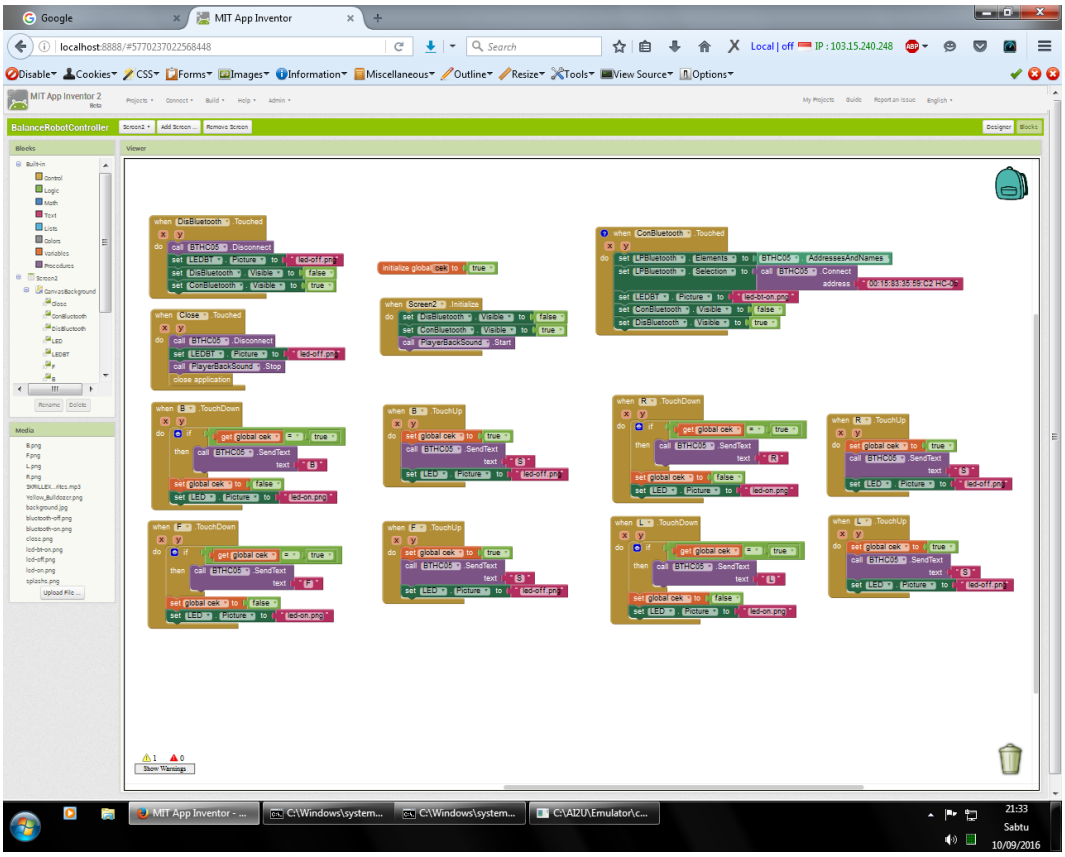


B. Tampilan Utama



Gambar IV.6. Perancangan *Software* Aplikasi Pengendali

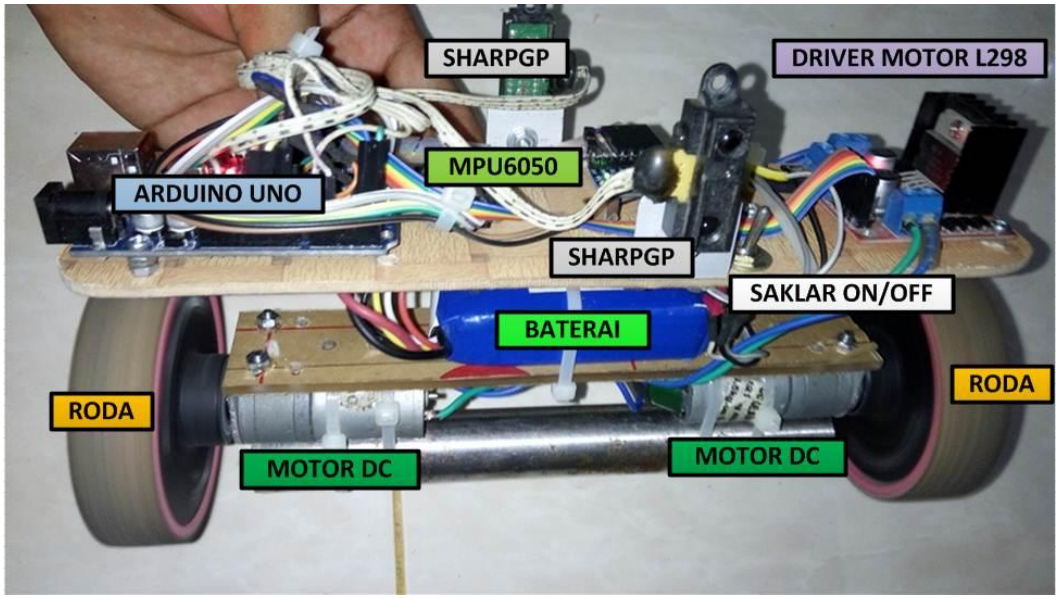
Untuk *listing* program aplikasi pengendali, berikut adalah gambar dari blok dari aplikasi aplikasi pengendali pada *smartphone* android, ditunjukan pada gambar di bawah ini :



Gambar IV.7. Blok Program *Software* Aplikasi Pengendali

IV.5. Pengujian *Hardware*

Setelah semua rangkaian yang telah selesai dirancang pada perangkat robot penyeimbang menggunakan sensor jarak berbasis android, kemudian dilakukan penyatuan semua rangkaian yang telah selesai dengan mekanik. Berikut adalah gambar hasil dari perancangan sistem mekanik dan eletronik ditunjukan oleh gambar IV.8 Berikut :



Gambar IV.8. *Hardware* Mekanik dan Elektronik

IV.6. Uji Coba Perangkat

Setelah semua komponen terpasang dan program selesai disusun, maka langkah berikutnya adalah melakukan pengujian alat. Pengujian ini dilakukan secara bertahap dari rangkaian ke rangkaian berikutnya.

IV.6.1. Pengujian Rangkaian Mikrokontroler Arduino Uno

Untuk mengetahui apakah rangkaian mikrokontroler Arduino Uno telah bekerja dengan baik, maka dilakukan pengujian. Pengujian bagian ini dilakukan dengan memberikan program pada mikrokontroler Arduino Uno. Program sederhana yang digunakan adalah *blink led* pada Pin 13 Arduino. Berikut adalah listing program dari *blink led* :

```
void setup() {
// initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off
  delay(1000);              // wait for a second
}
```

Jika setelah *upload* program dilakukan dan led berkedip setiap 1000 milisekon (1 detik) maka arduino dalam keadaan baik.

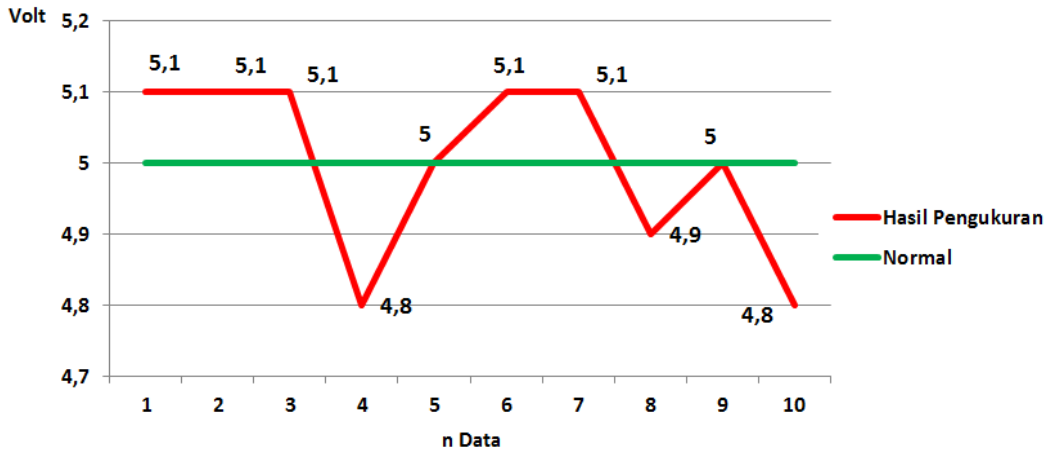
IV.6.3. Pengujian Rangkaian Regulator Tegangan

Pengujian ini dilakukan untuk mengukur tegangan yang dihasilkan dari regulator tegangan LM7805. Regulator mengubah tegangan dari baterai menjadi 5VDC untuk tegangan kerja perangkat keseluruhan. Pengukuran dilakukan menggunakan multimeter atau *voltmeter*. Berikut adalah hasil dari pengukuran tegangan, ditunjukan pada tabel IV.1 :

Tabel IV.1. Hasil Pengujian Regulator Tegangan 5VDC

No. Pengujian	Hasil Pengukuran (Volt)	Error (Volt)
1	5,1	0,1
2	5,1	0,1
3	5,1	0,1
4	4,8	0,2
5	5	0
6	5,1	0,1
7	5,1	0,1
8	4,9	0,1
9	5	0
10	4,8	0,2
Σ Error		**1,0**
Rata – Rata Error		**0,1**

Berdasarkan data dari tabel di atas, disimpulkan bahwa *error* dari tegangan normal dengan tegangan regulator 5VDC memiliki total selisih *error* ± 1 Volt pada 10 kali pengujian (n) atau rata – rata *error* sebesar 0,1 Volt. Berdasarkan tabel diatas, dapat digambarkan pada grafik di bawah ini :



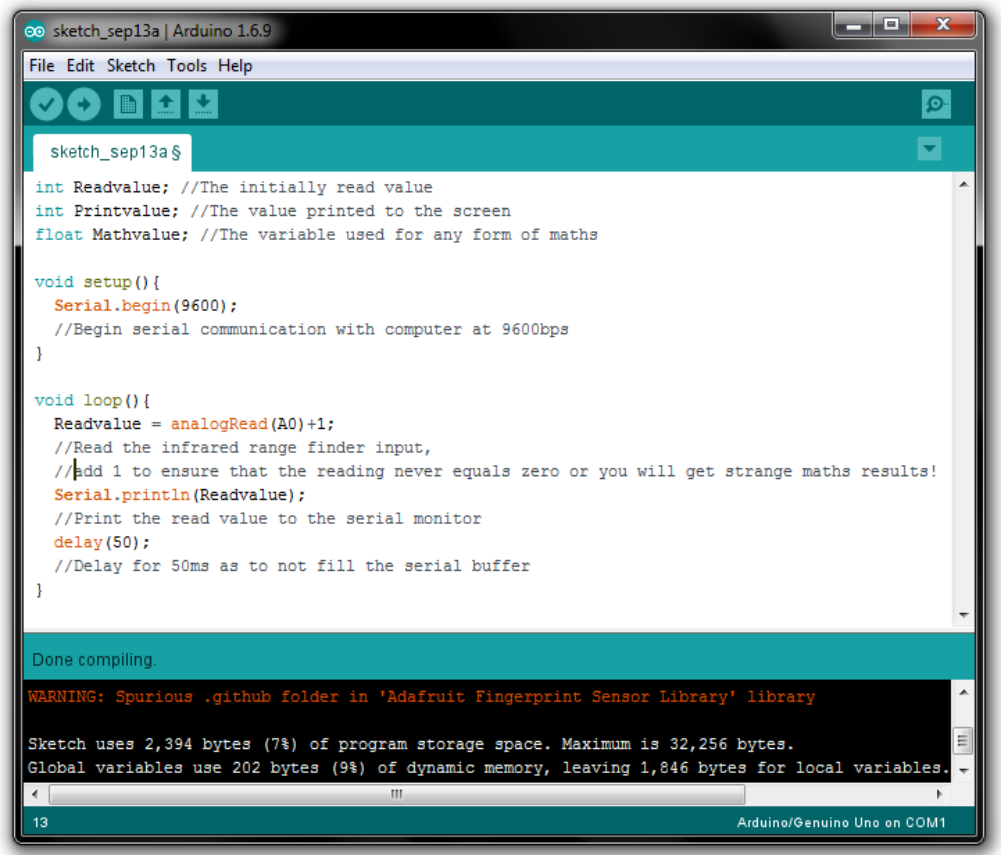
Gambar IV.9. Grafik Hasil Perbandingan Tegangan Normal dengan Regulator 5VDC

Terlihat hasil kedua pengujian hampir sama dengan selisih nilai pengukuran yang kecil. Kesimpulan dari pengujian ini adalah regulator tegangan 5VDC dapat dinyatakan berkerja dengan baik dan sesuai jika dibandingkan dengan tegangan kerja normal yaitu 5VDC.

IV.6.3. Pengujian Rangkaian Sensor Jarak Sharpgp GP11

SRF04 sebagai sensor jarak yang mengirimkan data jarak terhadap objek. Pengujian ini dilakukan dengan mencoba mengirimkan data jarak ke mikrokontroler. Data pembanding yang digunakan adalah penggaris dengan pengambilan data 10 cm – 50 cm. Berikut *listing* program untuk menampilkan data sensor jarak Sharpgp GP11 dalam cm :

```
int Readvalue; //The initially read value
int Printvalue; //The value printed to the screen
float Mathvalue; //The variable used for any form of maths

void setup(){
  Serial.begin(9600);
  //Begin serial communication with computer at 9600bps
}

void loop(){
  Readvalue = analogRead(A0)+1;
  //Read the infrared range finder input,
  //add 1 to ensure that the reading never equals zero or you will get strange
maths results!
  Serial.println(Readvalue);
  //Print the read value to the serial monitor
  delay(50);
  //Delay for 50ms as to not fill the serial buffer
}
```



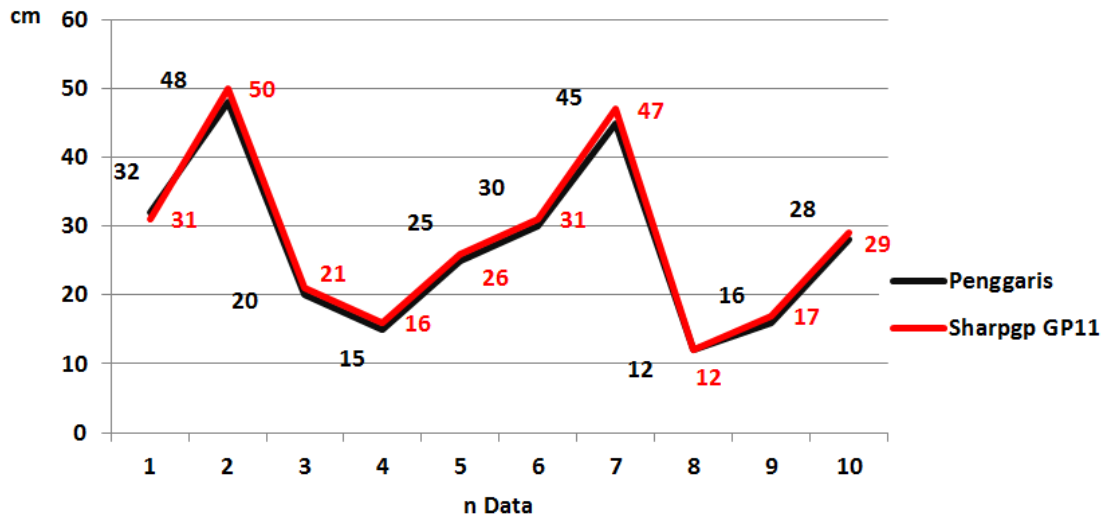
Gambar IV.10. Listing Program Pengujian Sharpgp GP11

Berikut adalah hasil dari pengukuran sensor jarak terhadap penggaris, ditunjukan pada tabel IV.2 :

Tabel IV.2. Hasil Pengujian Sensor Sharpgp GP11 Terhadap Penggaris

No. Pengujian	Hasil Pengukuran (cm)		Error (cm)
	Penggaris	Sharpgp GP11	
1	32	31	1
2	48	50	2
3	20	21	1
4	15	16	1
5	25	26	1
6	30	31	1
7	45	47	2
8	12	12	1
9	16	17	1
10	28	29	1
Σ Error			**12**
Rata – Rata Error			**1,2**

Berdasarkan data dari tabel di atas, disimpulkan bahwa *error* dari perhitungan jarak penggaris dengan Sharpgp GP11 memiliki total selisih *error* ± 12 cm pada 10 kali pengujian (n) atau rata – rata *error* sebesar 1,2 cm. Berdasarkan tabel diatas, dapat digambarkan pada grafik di bawah ini :



Gambar IV.11. Grafik Hasil Perbandingan Sensor Jarak Sharpgp GP11 Terhadap Penggaris

IV.6.4. Pengujian Transmisi Data Bluetooth HC-05

Pengukuran jarak transmisi bertujuan untuk mengetahui seberapa jauh *bluetooth* dapat berhubungan dan mampu membawa perintah dari *smartphone* ke mikrokontroler. jarak jangkau maksimum *bluetooth* adalah 10 meter. Pengujian dilakukan dengan dua metode yaitu pengujian jarak tanpa halangan dan pengujian jarak dengan banyak halangan untuk menghambat transmisi data, seperti lemari dan perabotan rumah lainnya. Tabel IV.3. berikut adalah tabel dari hasil pengujian tanpa halangan.

Tabel IV.3. Hasil Pengujian Jarak Transmisi Tanpa Halangan

Jarak	Pengujian I	Pengujian II	Pengujian III	Pengujian IV	Pengujian V
1 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
2 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
3 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
4 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
5 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
6 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim

7 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
8 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
9 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
10 Meter	Tidak Terkirim	Terkirim	Tidak Terkirim	Terkirim	Terkirim
Error	**1%**				

Pengujian jarak tanpa halangan dilakukan sebanyak 5 kali tahap pengujian dengan jarak maksimum sejauh 10 meter. Dari hasil pengujian, pada saat perangkat mencapai jarak jangkau maksimum 10 meter, perintah yang dikirimkan *smartphone* tidak diterima perangkat, dari 50 kali pengirimkan perintah terdapat 2 kali gagal diterima atau sebanyak 1%.

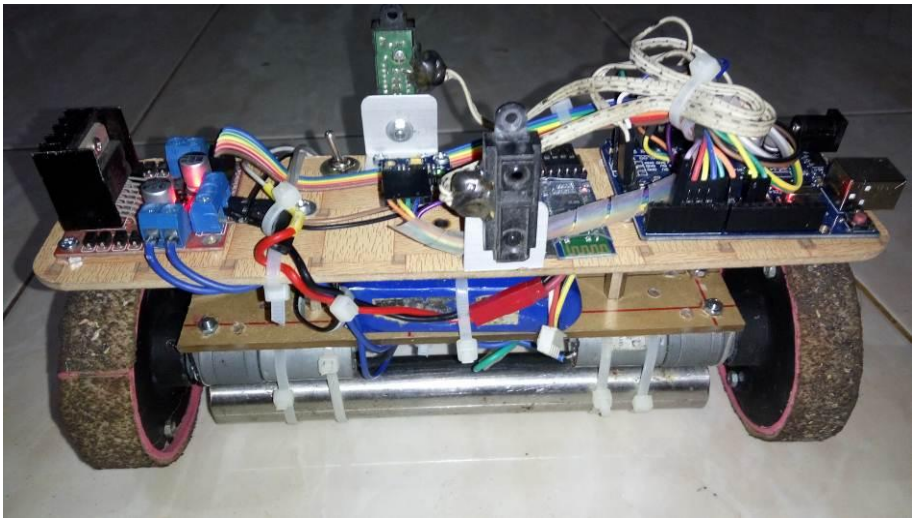
Tabel IV.4. Hasil Pengujian Jarak Transmisi Dengan Halangan

Jarak	**Pengujian I**	**Pengujian II**	**Pengujian III**	**Pengujian IV**	**Pengujian V**
1 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
2 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
3 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
4 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
5 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
6 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
7 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
8 Meter	Terkirim	Terkirim	Terkirim	Terkirim	Terkirim
9 Meter	Tidak Terkirim	Terkirim	Tidak Terkirim	Tidak Terkirim	Terkirim
10 Meter	Terkirim	Terkirim	Tidak Terkirim	Tidak Terkirim	Terkirim
Error	**2.5%**				

Sedangkan pengujian jarak dengan halangan dilakukan sebanyak 5 kali tahap pengujian dengan jarak maksimum sejauh 10 meter, terdapat perintah yang dikirimkan *smartphone* tidak diterima perangkat sebanyak 5 kali gagal diterima atau sebanyak 2.5%.

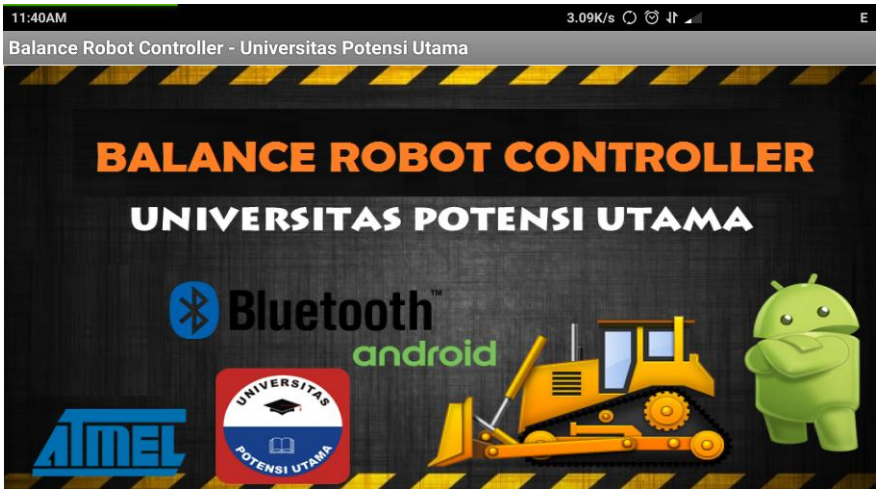
IV.6.5. Pengujian Analisa Perangkat Keseluruhan

Pengujian ini dilakukan untuk mengetahui apakah perancangan perancangan robot penyeimbang menggunakan sensor jarak berbasis android bekerja sesuai dengan logika program. Sebelum pengujian dilakukan, perangkat dalam telah menyala dan keadaan robot dalam keadaan normal. Keadaan normal yang dimaksud adalah keadaan dimana robot menyeimbangkan diri. Berikut adalah gambar dari kondisi robot ketika dalam kondisi normal :



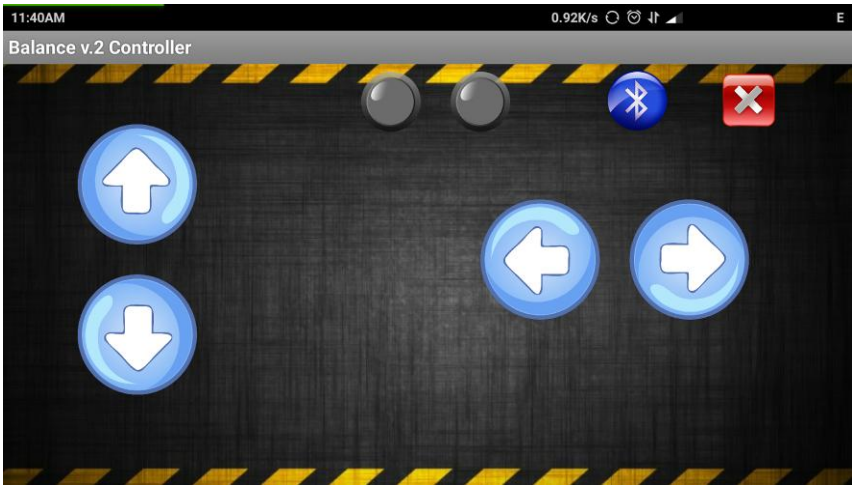
Gambar IV.12. Kondisi Normal Robot Penyeimbang

Pengujian berikutnya adalah mencoba koneksi *bluetooth* perangkat robot dengan perangkat *smartphone* android. Proses awal adalah menjalankan aplikasi seperti gambar berikut :



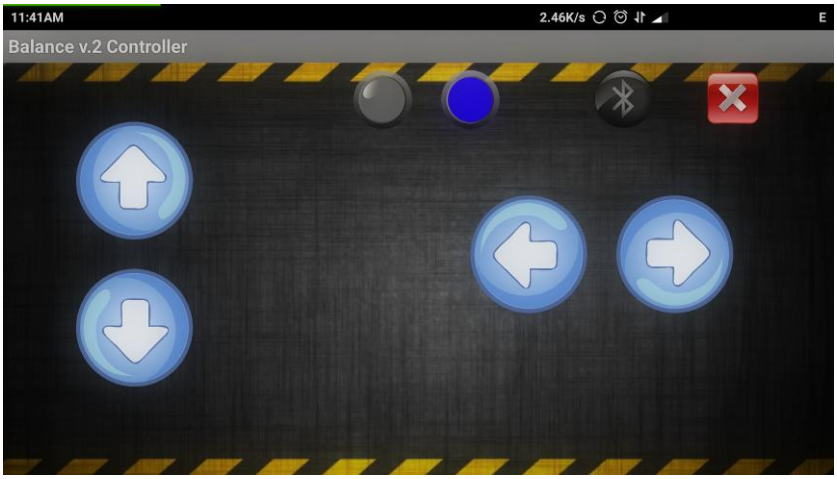
Gambar IV.13. Kondisi Smartphone Menjalankan Aplikasi

Koneksi dapat dilakukan dengan menekan tombol “  ” pada aplikasi. Perangkat akan otomatis terkoneksi. Berikut adalah gambar proses melakukan koneksi terhadap perangkat robot.



Gambar IV.14. Kondisi Smartphone Melakukan Koneksi.

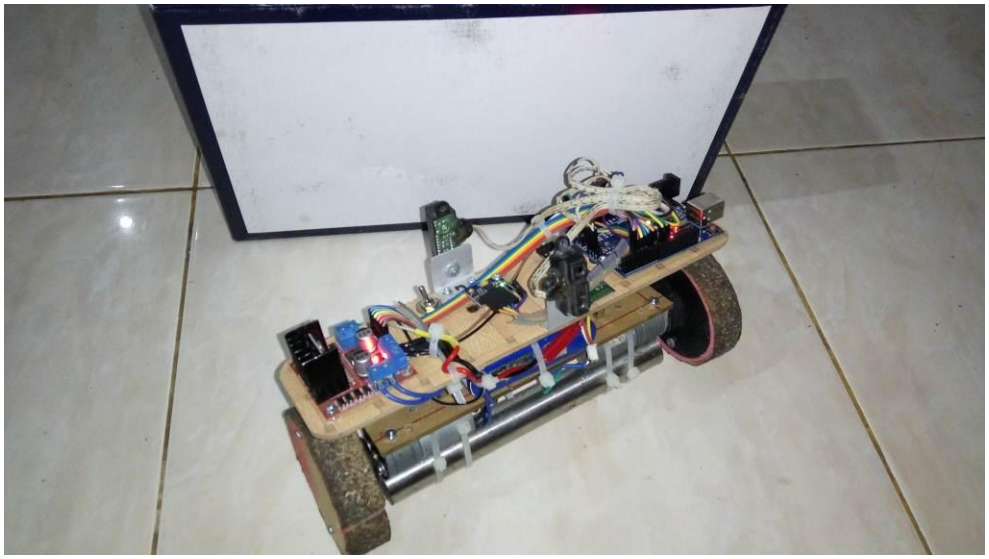
Jika koneksi berhasil, perangkat akan menampilkan indikator menyala. Berikut adalah gambar dari koneksi berhasil.



Gambar IV.15. Kondisi Smartphone Berhasil Melakukan Koneksi.

Robot penyeimbang akan mengikuti pergerakan sesuai dengan data yang dikirimkan oleh aplikasi android. Robot juga merespon objek yang terdeteksi pada sensor jarak posisi depan ataupun sensor jarak belakang. Berikut adalah gambar hasil pengujian bahwa robot bergerak mengikuti perintah dari pengguna dan mendeteksi objek sekitar.

Gambar IV.16. Robot Bergerak Menyesuaikan Perintah (1).



Gambar IV.17. Robot Bergerak Menyesuaikan Perintah (2).

Untuk melakukan diskoneksi, tekan kembali tombol “  ”. Aplikasi akan berhenti jika pengguna menekan tombol “  ”.

IV.7. Kelebihan dan Kekurangan

Pada perancangan robot penyeimbang menggunakan sensor jarak berbasis android ini masih kurang sempurna. Perakitan dan pembuatan perangkat ini masih memiliki beberapa kelebihan dan kekurangan, diantaranya:

a. Kelebihan

Adapun beberapa kelebihan yang dimiliki perancangan robot penyeimbang menggunakan sensor jarak berbasis android ini, antara lain :

1. Dengan adanya perangkat ini, maka kita dapat mengendalikan robot bergerak maju, mundur, kanan dan kiri sesuai dengan keinginan pengguna dalam keadaan seimbang dan menghindari objek yang terdeteksi oleh sensor jarak.

2. Robot tetap dapat menyeimbangkan diri walaupun diberikan dorongan dari luar.
3. Perangkat robot *balancing* bekerja menggunakan baterai lipo 11.1V 1A dan regulator tegangan 5VDC.

b. Kekurangan

Adapun beberapa kekurangan yang dimiliki perancangan robot penyeimbang menggunakan sensor jarak berbasis android antara lain:

1. Perancangan mekanik robot cukup berat, sehingga konsumsi tegangan menjadi besar.
2. Respon robot terhadap data yang dikirimkan aplikasi terdapat *delay* (tunda) yang menyebabkan sering terjadi kehilangan data dan robot bergerak dalam kondisi *error*.
3. Sensor jarak hanya dapat mendeteksi objek pada posisi depan dan belakang dari perangkat (robot).
4. Aplikasi yang dirancang telah disesuaikan dengan *module bluetooth* HC-05, sehingga aplikasi tidak dapat digunakan untuk mengontrol robot lainnya.
5. Sumber daya yang dibutuhkan untuk mengendalikan motor DC cukup besar dan terjadi pemanasan berlebih pada regulator tegangan.