System Design of a Two wheeler self balanced vehicle

A Thesis submitted to the
Dept. of Electrical & Electronic Engineering, BRAC University
in partial fulfillment of the requirements for the
Bachelor of Science degree in Electrical & Electronic Engineering

Sheikh Mohibul Islam Rumi
I.S.M. Shanamul Islam
Mehdi Fakid Hossain

30th April 2014

Declaration

We do hereby declare that the thesis titled “System Design and Circuit Implementation of a self balanced two wheeler vehicle” submitted to the Department of Electrical and Electronics Engineering of BRAC University in partial fulfillment of the Bachelor of Science in Electrical and Electronics Engineering. This is our original work and was not submitted elsewhere for the award of any other degree or any other publication.

Date:

Supervisor:

_____

Dr. Md Khalilur Rhaman

_____

Sheikh Mohibul Islam Rumi

Student ID: 10121053

_____

I.S.M. Shanamul Islam

Student ID: 10121088

_____

Mehdi Fakid Hossain

Student ID: 10121099

Acknowledgement

We are deeply indebted to our supervisor Dr. Md Khalilur Rhaman, Associate Professor, BRAC University for his help, guidance, motivating suggestions and encouragement throughout the course of this work. Moreover, we would like to thank Maisun Ibn Monowar, Md. Ismail Kiron, and Md Waheduzzaman for their help on arduino and android programming. At the same time we would like to thank Monir Hassan for helping us in building the chassis for experiment. Besides, we are grateful to faculty member Risul Karim for his amazing cooperation and assistance.

Additionally, we would like to thank our parents and friends for encouraging and supporting us.

Abstract

The project is about the designing of a two wheeler self balanced car. The two wheeler vehicle would be able to balance itself and can be stabilized against any impact and in zero velocity as well. We used two heavy rotating disks with hub motors at the chassis to compensate the tilt of the vehicle and get it stabilized. An android device is used to measure the tilt angle of the chassis using orientation sensor. The data then is sent to a bluetooth receiver that is connected with an arduino. An android application is developed which takes the angle of tilt of the vehicle as data input from the phone and sends a control signal to the arduino accordingly. Using the signals the vehicle is balanced by controlling the motor from the arduino which determines the tilt direction of the rotating disks. This vehicle is designed to provide the safety that two wheeler vehicle does not have during an impact. Our aim is to design a safe, cost effective and fuel efficient vehicle.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Motivation

Motorbike is a very popular transport around the globe. It has been very popular due to its energy efficiency, compact design, convenience and attractive look. Many youngsters consider it as fashionable ride while people in the developing country often use it as a low priced vehicle with better fuel efficiency.

However, despite of the features and popularity motorbike has lack of safety and is very risky. Therefore, motorbike accidents are fatal. An injury is must while death is more frequent scenario.

The major lacking in motorbike addressing the safety features are the passenger's body is exposed during ride time which allows the passengers to get off the vehicle and exposes him to impact with roadside elements and the chance of damage is limitless.

On the other hand many people does not consider it as a transport as it does not have the comfort features like the car while two wheel vehicle can save energy and space.

We designed a vehicle which has both the cabin comfort of a car and the compactness of a motorbike. It has the safety features which can minimize the damage during an impact. The vehicle also has luxury features and still cost effective.

1.2 Project Outline

The project is about designing a self balanced two wheeler vehicle and developing the circuits. The vehicle balances on its own when it has its engine on while there is stand to keep it standing when the engine is not running. The vehicle will be balancing on the using counterforce to it when it tilts beyond the accepted angle.

1.3 Project Objective

The objective of building a self balanced two wheeler vehicle is mainly to ensure safety of the rider. We have considered the scenario of our country, Bangladesh in this manner. Enormous numbers of people become victim of fatal accidents. Moreover, the cars in the cities are increasing day by day but the roads are not increasing. So if a vehicle that can serve like a car and just takes the small amount of place like a motorbike whether for parking or running on roads, would be a better solution for people. With the cabin the rider is safe from impact of thrust and with the self balancing property of the vehicle; the rider is safe from falling. We are trying to make a compact size vehicle for low power consumption too.

1.4 Outline of this thesis

The thesis report is organized in such a way that the reader can understand the basic theories and components of the self balancing vehicle. They will also be able to understand the detailed process and progress of how the project went on. The report also describes how the working principle is implemented firstly manually and then automatically. The batteries we used, the motors we have inplemented, the control flow block diagram, the algorithm everything is described here in brief. The limitations and problems faced has also been discussed in the report. Finally the results and discussions are described at the end.

Chapter 2

Literature Review

2.1 Stability Analysis of a Two-wheeler during Curve Negotiation under Braking:

Two-wheelers have some different dynamic characteristics. They are statically unstable but the roll instability disappears as the forward speed increases. A simplified model has been considered to analyze the effects of forward speed and braking force on the roll instability during cornering of a two-wheeler. It helps to understand some important concepts about a two-wheeler negotiating a turn under applied braking force.

One method of establishing the proper lean is counter-steering in which handlebar is turned counter to the desired turn and thus developing a centrifugal torque that leans the two-wheeler appropriately [3].

2.2 Two-wheeler model and Equations:

2.2.1Two-wheeler model for curve negotiation without braking:

For the purpose of the analysis a simple two wheeler is considered as shown in fig.1. The coordinates used to analyze the two-wheeler is shown in fig.2 (a). The inertial system is with the axes uvw and origin O [3].

The system uvw is fixed to inertial frame and the system xyz has its origin at contact point of rear wheel with the xy plane. The x-axis is in the direction of the contact line of the rear wheel with xy plane and z- axis is vertical and y-axis is perpendicular to x and

positive on the left side of the two-wheeler. The roll angle Φ of rear frame is positive when leaning to the right and the steer angle δ is positive for steering left. The angle between axes x and u, is ψ indicating the orientation of rear wheel plane [3].

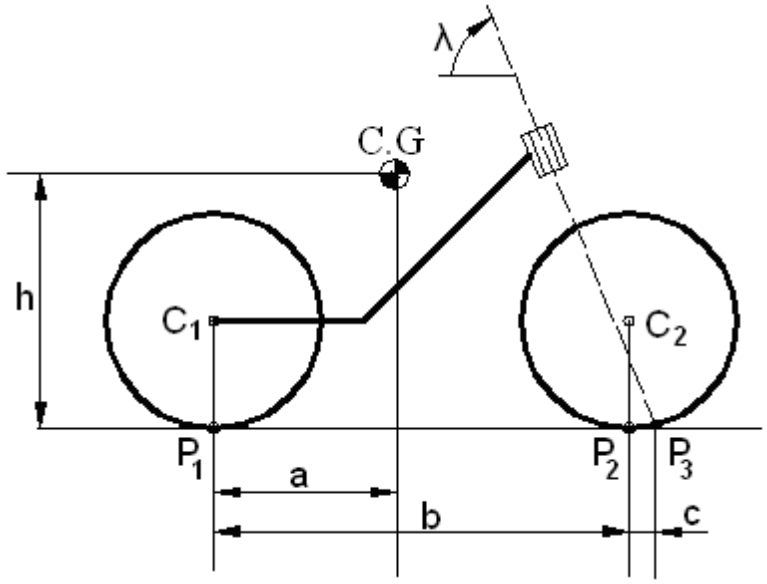

Fig. 2.1: Parameters defining a simple two-wheeler.

Due to tilt in steer axis by an angle λ, the effective

front angle is

$\delta f = \delta \sin\lambda$

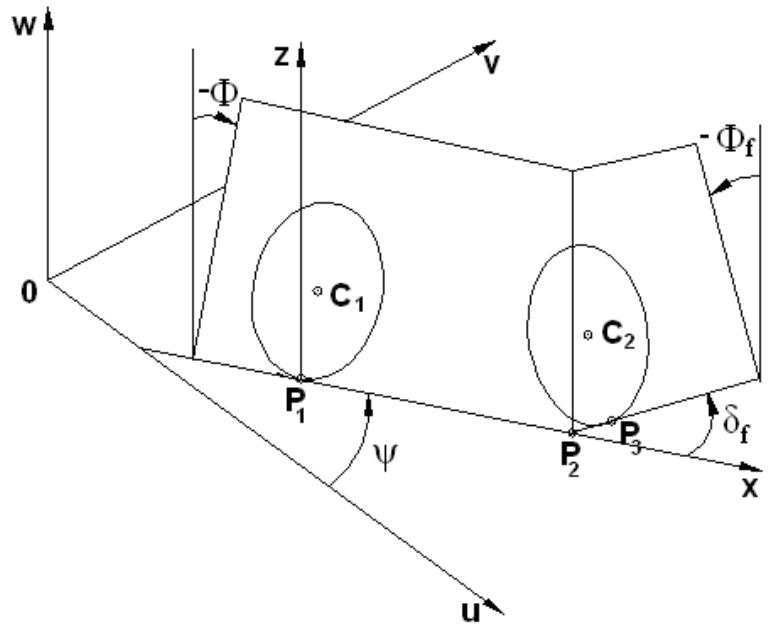

Fig. 2.2: Coordinate system.

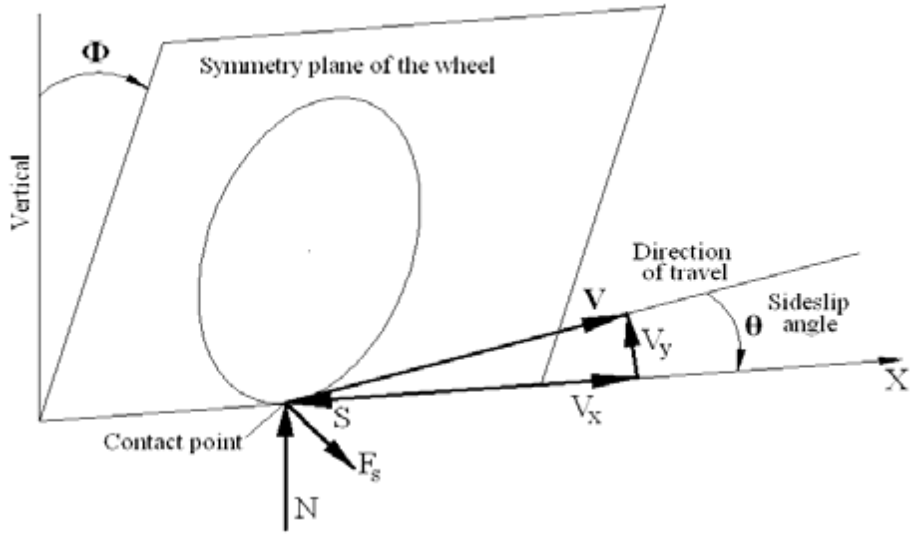

Fig. 2.3: Tyre forces and sideslip angle.

The effective front fork roll angle is given by

$$\varphi f = \varphi - \delta \cos\lambda \qquad (2)$$

Normal reaction force on front wheel excluding dynamic and centrifugal effect

$$N_f = amg/b \qquad (3)$$

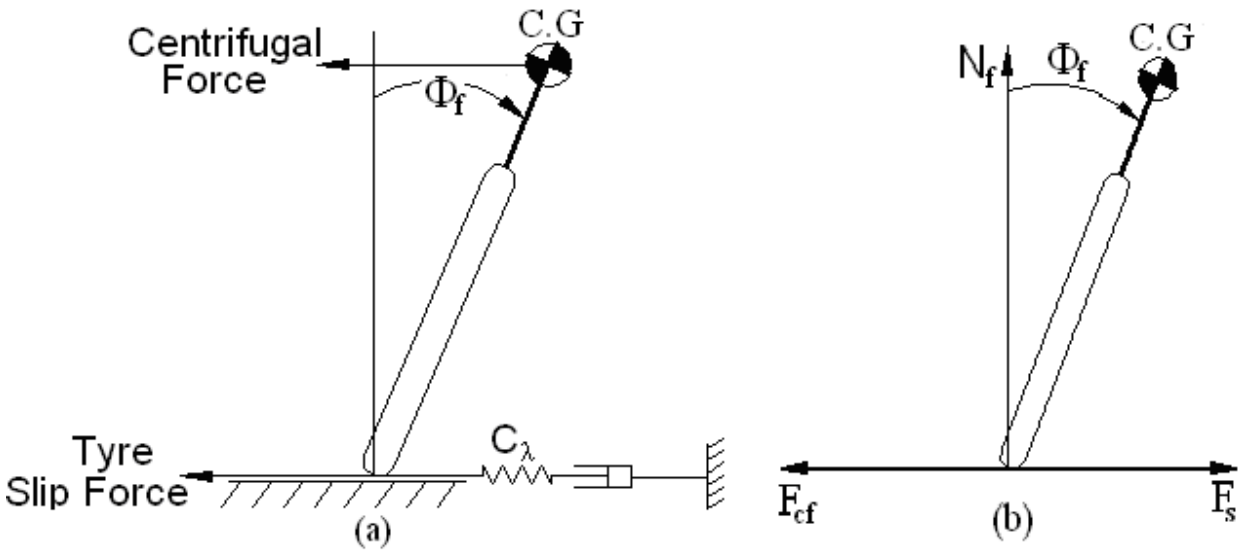

Fig. 2.4: (a) Tire model, (b) Reaction forces on front wheel.

Effective centrifugal force on front wheel

$$F_{cf} = \frac{amv^2}{b \times r}$$

$$F_{cf} = \frac{amv^2}{b^2}\delta_f \qquad (4)$$

$$F_{cf} = \frac{amv^2 \sin\lambda}{b^2}\delta$$

Lateral force resisting sideslip

$$F_s = C_\lambda \times \theta + C_\phi \times \phi_f \qquad (5)$$

The static torque balance equation

$$T - [F_{cf} + (C_\lambda \times \theta + C_\phi \times \phi_f)]c \times \sin\lambda = 0 \qquad (6)$$

Substituting equations (1) - (5) in equation (6)

$$\delta = \frac{1}{K_\lambda c \sin\lambda} T - \frac{C_\phi}{K_\lambda}\phi - \frac{C_\lambda}{K_\lambda}\theta \qquad (7)$$

Angular momentum balance for the two-wheeler frame [2]

$$J\ddot{\phi} - mgh\phi = \frac{Dv\sin\lambda}{b}\dot{\delta} + \frac{m(v^2 h - acg)\sin\lambda}{b}\delta \qquad (8)$$

Substituting δ from equation (7) in main frame equation (8)

$$J\ddot{\phi} + \frac{DvC_\phi \sin\lambda}{bK_\lambda}\dot{\phi} + [K_v C_\phi \sin\lambda - mgh]\phi = \frac{Dv}{bcK_\lambda}\dot{T} + \frac{K_v}{c}T - [K_v C_\lambda \sin\lambda]\theta \qquad (9)$$

Where,

$$K_\lambda = \frac{amv^2 \sin\lambda - b^2 C_\phi \cos\lambda}{b^2}$$

$$K_v = \frac{m(v^2 - acg)}{bK_\lambda}$$

$$D = mah$$

$$J = mh^2$$

The above equation reveals that torque at the handle bar has a great influence on roll angle as well as stability of the two-wheeler. [3]

2.2.2 Two-wheeler model for curve negotiation with braking:

For stability analysis of the two-wheeler during cornering with braking the model considered is shown in fig.3. The key purpose of the model is to discuss the balancing problem for two-wheeler during cornering under braking. The distribution of forces during braking is shown in fig.4.

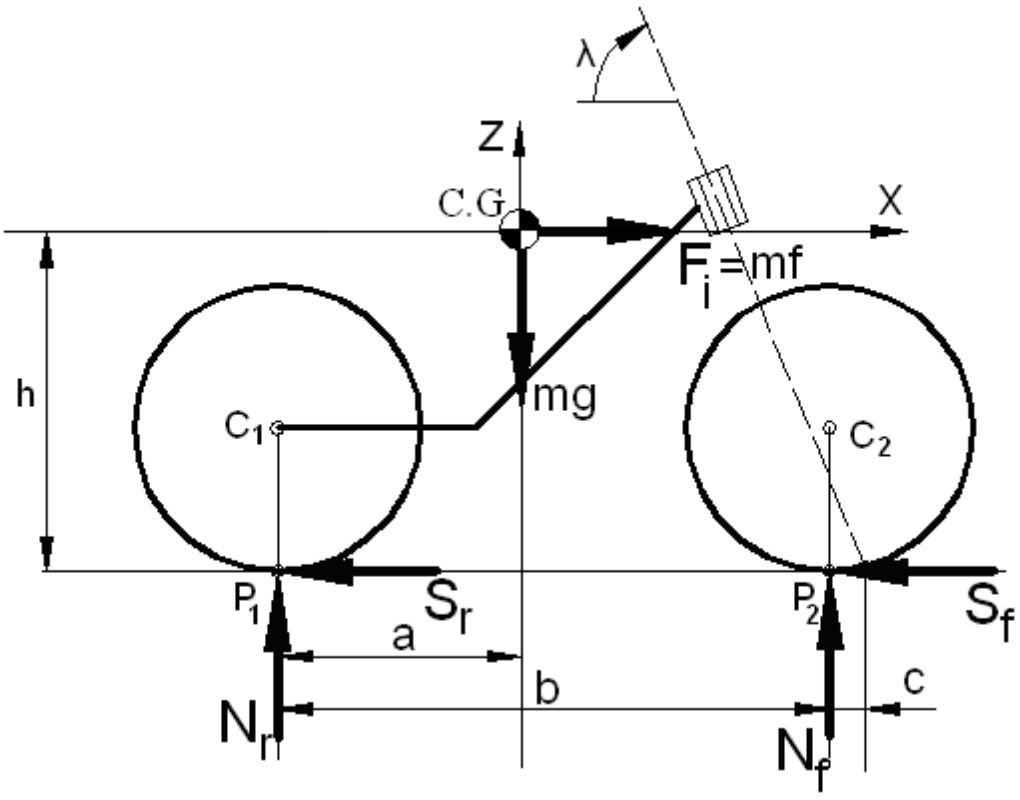

Fig. 2.5: Forces acting on the two-wheeler during braking.

Vertical reaction force due to braking on front and rear wheel. [3]

Vertical reaction force due to braking on front and rear wheel

$$N_{fb} = \frac{amg}{b} + \frac{mfh}{b}$$
$$N_{rb} = \frac{(b-a)mg}{b} - \frac{mfh}{b} \qquad (10)$$

Substituting δ in main frame equation (8) from equation (7)

$$J\ddot{\phi} + \frac{DvC_{\phi}\sin\lambda}{bK_{\lambda}}\dot{\phi} + \left[K_1 C_{\phi} - mgh\right]\times\phi = \frac{Dv}{bcK_{\lambda}}\dot{T} + \frac{K_1}{c\sin\lambda}T - [K_1 C_{\lambda}]\theta \qquad (11)$$

Where,

$$K_1 = \frac{m\sin\lambda}{bK_{\lambda}}\left[v^2 h - acg - \frac{2Dav^2\sin\lambda}{b^2 K_{\lambda}}\right]$$

$$f = \frac{F_i}{m}$$

Equation (11) is derived considering deceleration due to braking and therefore, velocity as a variable with time. Breaking force largely affects the stability of the two-wheeler which can be controlled suitably applying torque on the handle bar. Thus equations (9) and (11) provide qualitative explanation for stabilization [3].

CHAPTER 3

Architecture

3.1 Chassis:

The chassis has been built using iron plates. It has been designed like a ladder to keep space at the middle for two rotating disks and two wheels at two ends. The rotating disk along with the hub motors are set up in a certain distance from each other so that they can move easily during tilt of the structure. The structure has 6 inches clearance from the ground. The hub motors tilts were observed when the frame was tilted. As the wheels are spinning carrying a fair amount of weight so if they can be tilted in the other direction to the impact direction then this can counter the impact force and this might be enough for the frame to fall down if the weight ratio of the frame and wheels are right.

To tilt the wheel forcefully to desired direction a motor is integrated with the hub motors axis. RPM motor was needed with high power output to serve this purpose. In this case a side glass lifting motor of car has been selected. The motor worked according to this assumption. When it was biased the tilt of the rotating hub wheels it could be observed that it gives a great force in the particular direction. The rotating wheel was forced to the opposite direction of the frame's fall and observed it can counter the falling force and can pull back the whole frame. However, the frame cannot be stabilized as the operation could not be done in a synchronous manner as it was operated manually. The weight distribution in the structure also matters in this issue. Weight has to be equal in both right and left side. The glass lifting motor that has been attached here increases the weight on

that side. So some additional weight on the left had to be put just to balance out the structure.

Figure 3.1: Chassis

3.2 Proposed Chassis Structure:

The design of the chassis is kept very simple a. The rotating disks are accommodated at round space in the middle of the chassis. They are quite apart from each other to rotate and tilt according to the self balancing principle. It helps to stabilize the vehicle perfectly. At the same time enough space is allotted to connect two wheels at both of the ends. The wheels are connected to the chassis using axles. This chassis is mainly built with steel. The hub motors are attached firmly with their mount in the hub motor chambers. The batteries or other components can be kept above the hub motor chamber. For example, on the chassis above the hub motor chamber there will another compartment for engine. The compartment for the rider and the luggage will be above the main chassis. The chassis will consist of batteries, wires, gyroscope and engine. The automatic gear system will also be controlled from here. The main motors, electrical components will be accommodated here.

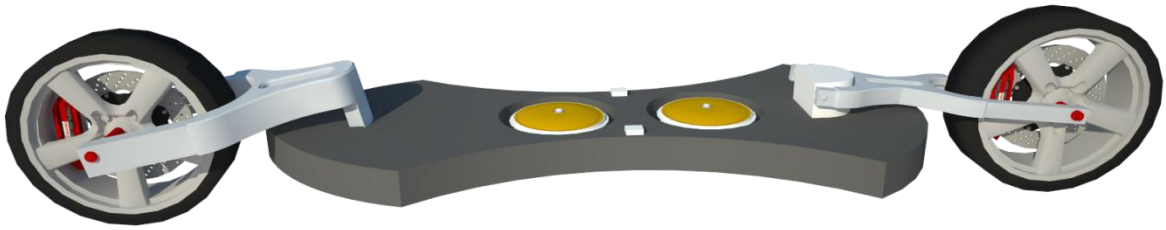

Figure: 3.2 proposed chassis structure

The hub motors have been kept in such a way they can move according to the tilting angle as mentioned earlier.

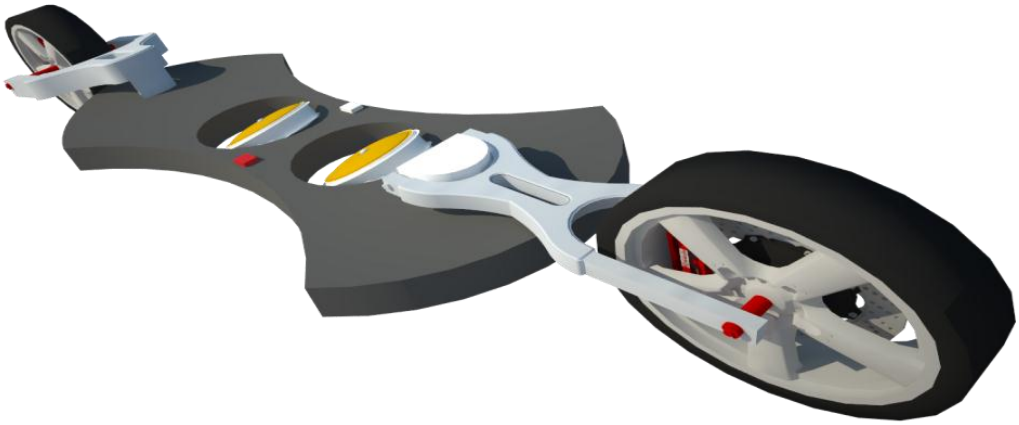

Figure 3.3: Chassis tilted left

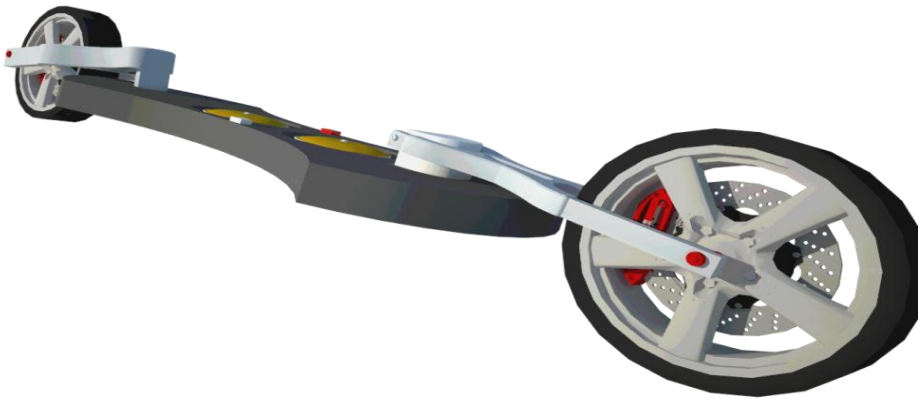

Figure 3.4: Chassis tilted right

The chassis was let to be tilted either ways. The wheels are designed a bit wider than the usual two wheel vehicles to have a better stability. Wheels are wide enough to keep the vehicle standing still when it is not rolling and the engine is off without any help of the self balancing technology

3.3 Design:

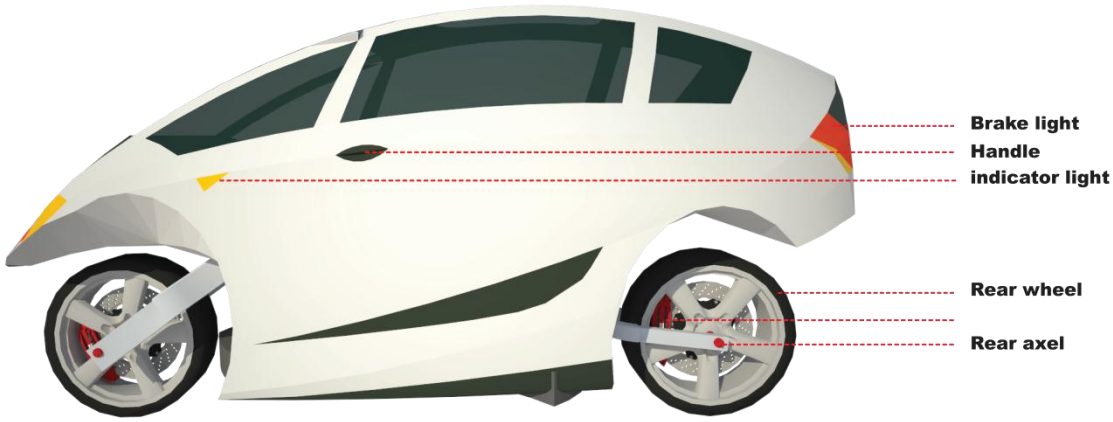

Figure 3.5: Side View

The car is designed in such a way what a single person can easily accommodate in it easily. It is given the aerodynamic shape in order to enable it achieving great speed. It has one door on its each side. There are indicator lights placed on the side of the car. It also has a stand. If the car is not started, then it stands on it. As soon as it starts, the hub motors take the responsibility to balance it. It is designed to have two threshold angles while it is in motion. The angles depend on the speed. The motor will balance the car according to the threshold angle.

The car has safety features in it such as seat belts and airbags. There will be a air conditioner to cool the cabin as it is a must feature to any car. The dash board includes an odometer, rpm meter and and speedo meter it self. There is a panel for display from rear camera as well as media source. This is a car driven by electrical system.

Figure 3.6: Rear View

Rear part has brake lights and indicatior lights on each side. There rear part of the body is extended to the length of the rear wheel. It provides some additional space for accommodating necessaries. The car is designed such a way that it needs less space for parking andin the road too. The gear system of the car is fully automated and operated

internally. GPS system will be integrated for the purpose of navigation and can also be very useful to prevent theft.

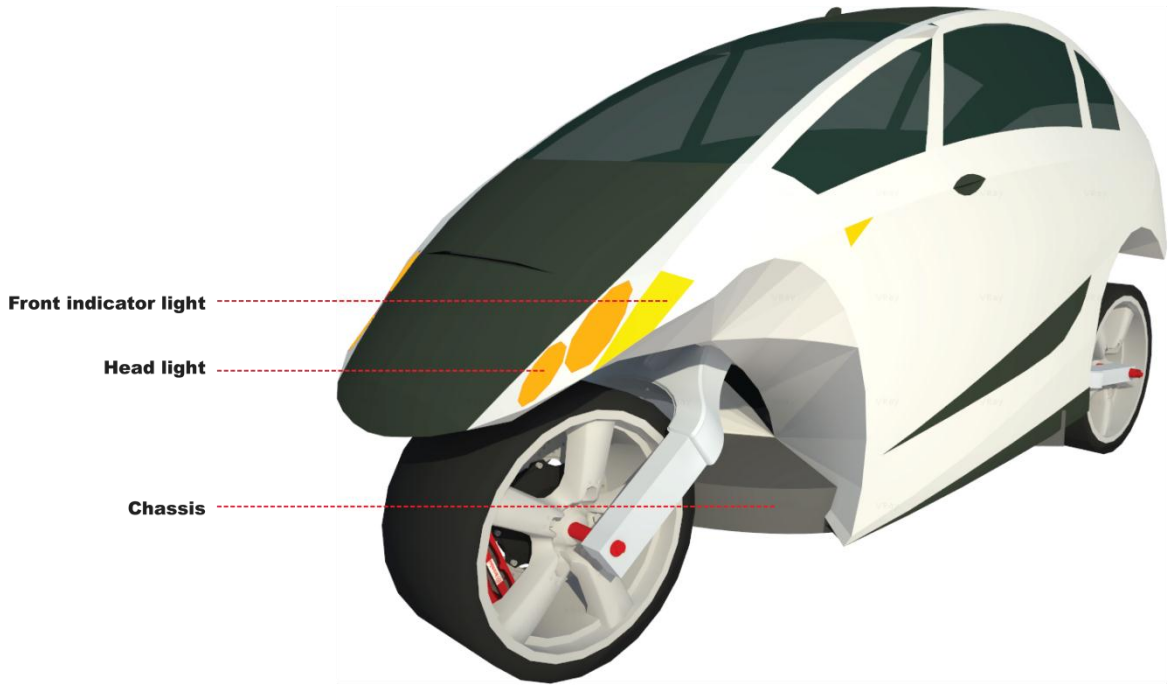

Figure 3.7: Front View

In the front side of the car two headlamps are placed on each side along with an indicator and the front axle holds the front wheel. The car's body is extended and curved towards ground which actually covers the front wheel and works like mud guards. It gives an attractive look too.

CHAPTER 4

Implementation

4.1 Introduction

This two wheeler self balancing vehicle is made up with the components such as batteries, a gyro sensor, circular iron disc, hub motor with controller, arduino, an android mobile phone and a HC-05 bluetooth module that connects the phone with the arduino. Different components required by the vehicle system for this project will be discussed. The aspects that are going to be covered are mechanical implementation, electrical implementation, control, power and communication of the system here in this chapter.

4.2 Mechanical implementation:

4.2.1 Hub Motor:

Hub motor is also called “Wheel Motor”, “Wheel Hub Motor”, and “In Wheel Motor”. This is an electric motor incorporated with the hub of a wheel that drives the wheel directly. Hub motor electromagnetic fields are supplied to the stationary windings of the motor. The outer part of the motor follows, or tries to follow, those fields, turning the

attached wheel. In a brushed motor, energy is transferred by brushes contacting the rotating shaft of the motor. Energy is transferred in a brushless motor electronically, eliminating physical contact between stationary and moving parts.

Electric motors have their greatest torque at startup, making them ideal for vehicles as they need the most torque at startup too. Their greatest torque occurs as the rotor first begins to turn, which is why electric motors do not require a transmission.

Two hub motors has been used in this vehicle so that these motors can help to stabilize the force during tilt.

Figure 4.1: Hub Motor

4.2.2 Hub Motor Controller:

Hub motor controller is needed to connect the hub motor with batteries through wires.

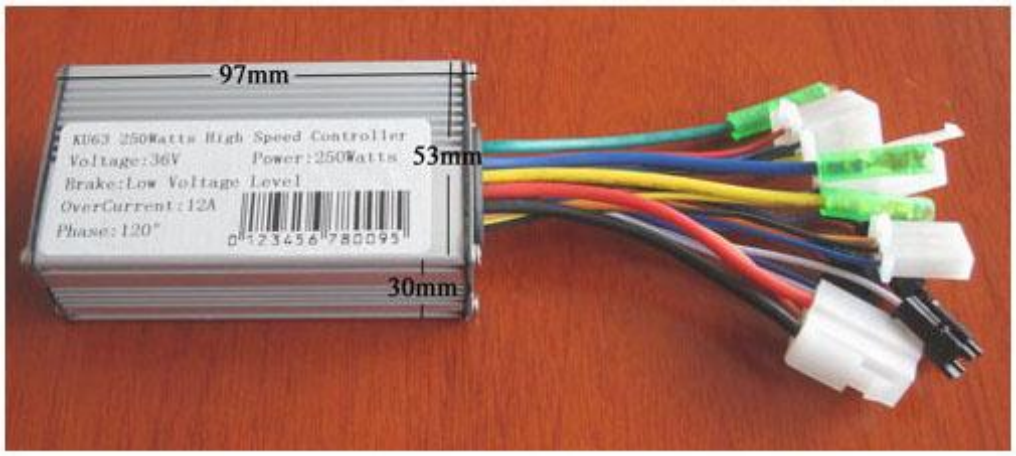

Figure 4.2: Hub Motor Controller

Two motor controller was used to drive two hub motors.

4.2.3 Additional weight:

For stabilizing weight, 10.64kg of weight was provided with the hub motors. The weight is made of irons. This weight helps the hub motors to provide required force for stabilization. It provides addition momentum generated from the rotating disks. The irons are round and attached under the hub motors. Finally the weight of each hub motor including the iron becomes 12.44kg.

4.2.4 Wiper Motor:

The wipers combine two mechanical technologies to perform their task:

- A combination electric motor and worm gear reduction provides power to the wipers.
- A neat linkage converts the rotational output of the motor into the back-and-forth motion of the wipers.

It takes a lot of force to accelerate the wiper blades back and forth across the windshield so quickly. In order to generate this type of force, a worm gear is used on the output of a small electric motor. The worm gear reduction can multiply the torque of the motor by about 50 times, while slowing the output speed of the electric motor by 50 times as well. The output of the gear reduction operates a linkage that moves the wipers back and forth. Inside the motor or gear assembly is an electronic circuit that senses when the wipers are in their down position. The circuit maintains power to the wipers until they are parked at the bottom of the windshield then cuts the power to the motor. This circuit also parks the wipers between wipes when they are on their intermittent setting.

Wiper motor is used here to solve the balancing problem. When the vehicle is tilted left to a certain angle then a signal comes and starts the motor. The motor is expected to force the vehicle to right, thus it balances the vehicle. Again if the vehicle is tilted right to a certain angle, another signal comes and starts the motor. The motor gives a force to the left balancing the vehicle. If the vehicle is straight and not tilted beyond the threshold angle, the motor does not start. Wiper motor works as a bidirectional motor and mainly is responsible for changing the orientation of the two rotating hub motors.

Figure 4.3: Wiper motor

4.3 Electrical Implementation:

With the above motor driver circuit, the motors were operated using two relays. And it was able to pull the chassis towards right when it was tilted left and pull the chassis to left when it was tilted right. Here a 12 volt DC battery is used to control the wiper motor circuit. Motor direction is mentioned in the circuit diagram by DIRSW1 and Q1 is for speed control. Diode prevents any inverse current.

But it could not get the chassis still and stable in the upright position. Moreover due to the massive weight we have put up with the hub motors there was a large amount of current drawn by the wiper motor and as the circuit could not take the amount of load it burnt out after a couple of experiments.

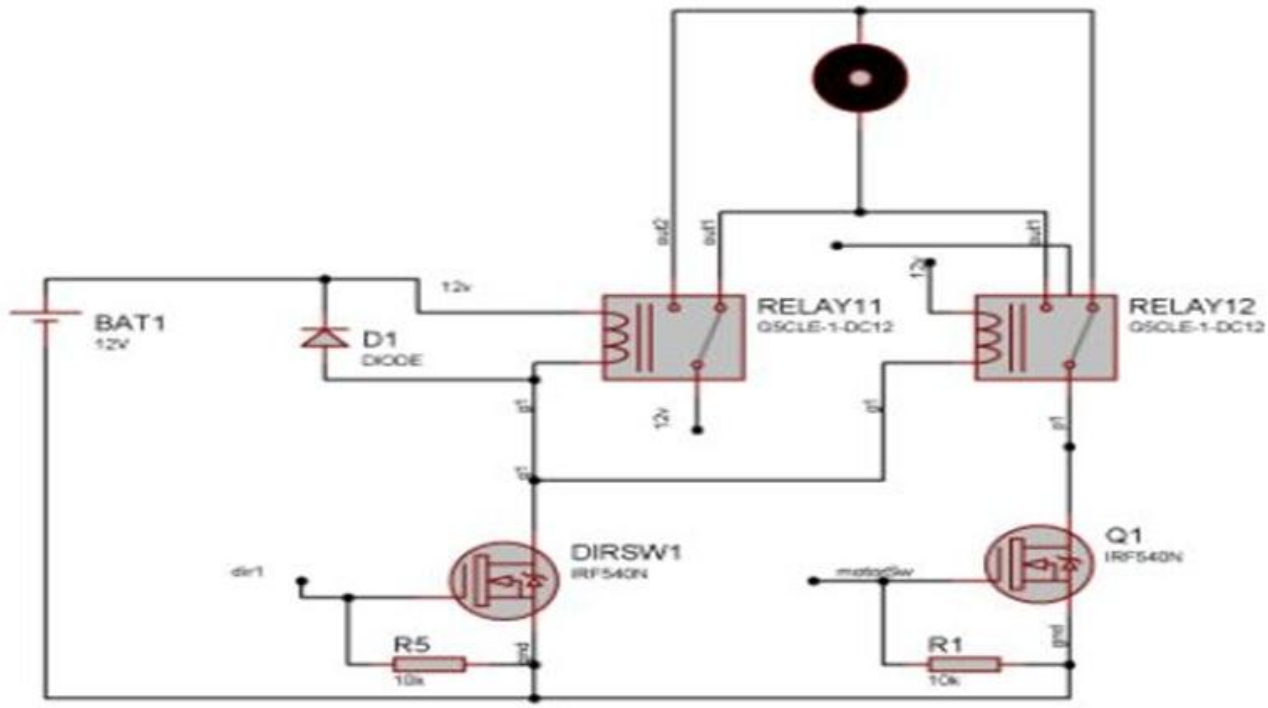

Figure 4.4: Electrical Circuit for automated control

4.4 Control Implementation:

Initially the chassis was controlled through manual switching. Then this control system was made automated by using the arduino and android phone.

The control flow block diagram shows that first the data from an android phone is sent through the Bluetooth kit to arduino. The arduino then drives the motor driver circuit to control the wiper motor and thus the wiper motor controls the orientation of the hub motor to stabilize the chassis.

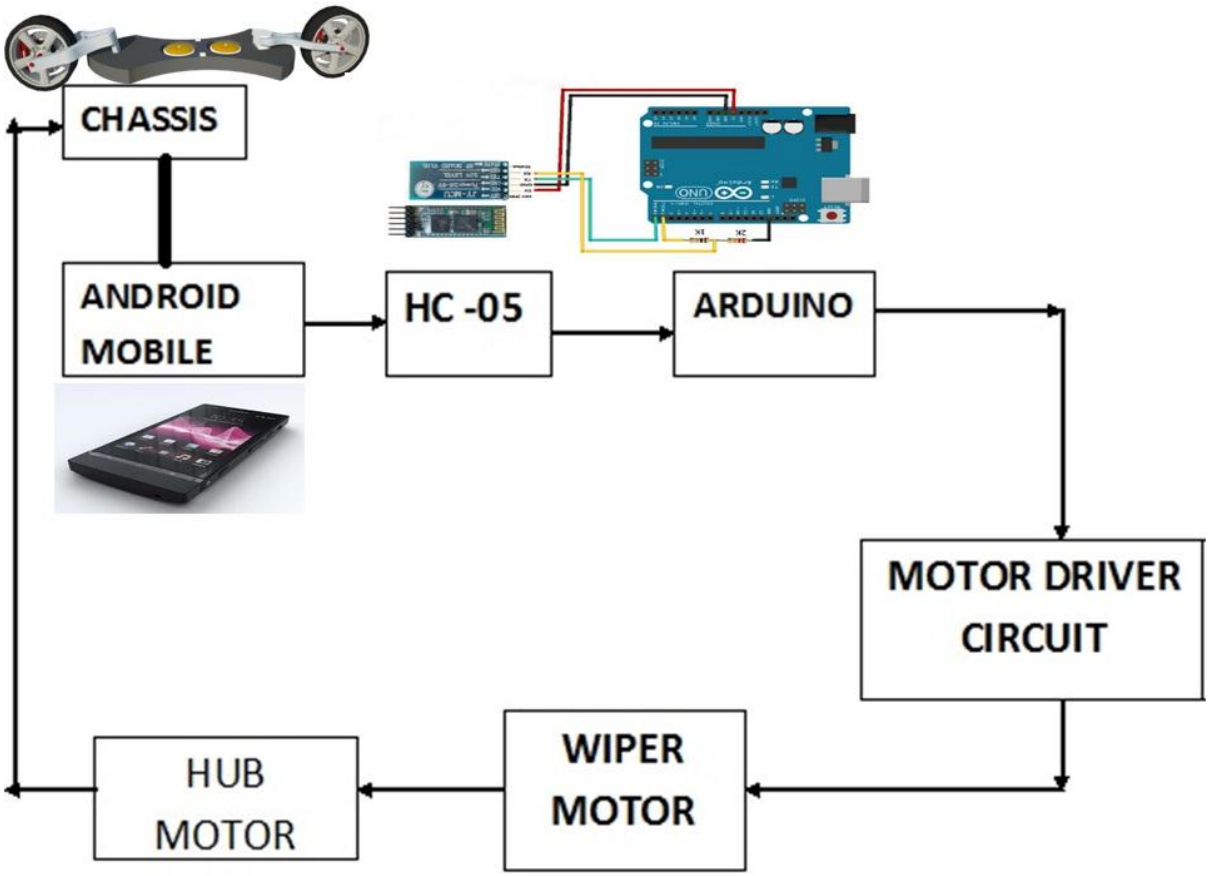

Figure 4.5: Control Block diagram

Walton Primo NX android mobile was paired with arduino mega successfully and communication was made successfully between these two devices. Then the android mobile was configured to send control signal to arduino which turns on pin number 8, 9, 10 of the arduino for the orientation angle equal or less than -35 degree, orientation angle -34 to 34 degree and orientation angle equal 35 degree or beyond respectively. And only one pin gets enable at an instance while one pin is on the rest are off.

These pins was used to switch the wiper motor in both directions which triggers the change of orientation of the two hub motors in our desired direction. So the arduino was programmed to rotate the wiper motor in forward direction when the pin number 8 is on and run the wiper motor reverse when pin number 10 is on. This tilts the hub motors backward and forward respectively. And when number 9 pin is on the motor does not run.

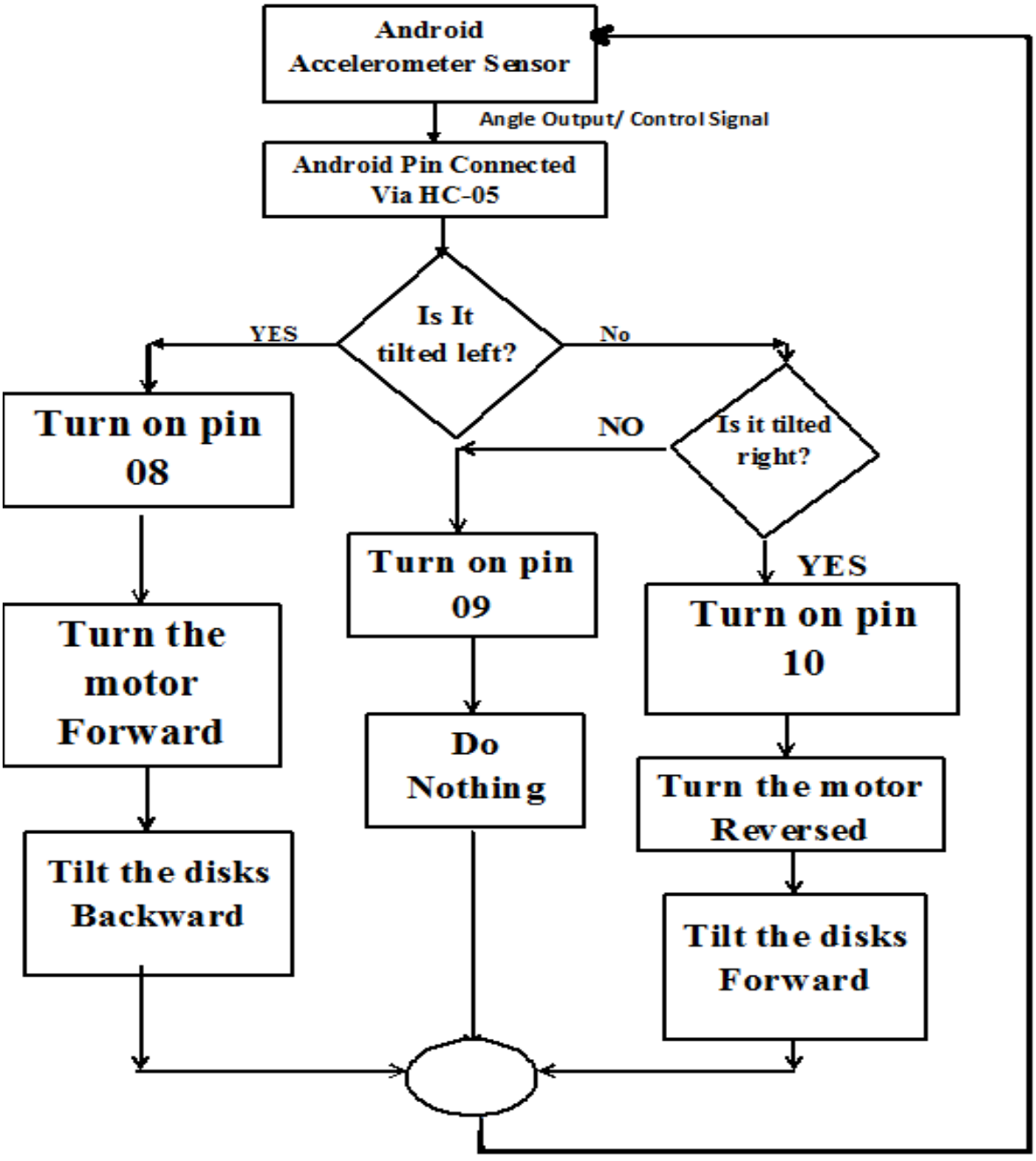

Figure: 4.6: Control Flow diagram 1

4.5 Power:

Hub motor specifications:

Name: Q8512 Font driving E- bike hub motor

Hub motor ratings: 24V

Power: 200-250W

Fork size: 100mm

RPM: 330

Hub motor Controller Rating:

Applied voltage: 36V

Current: 15Amp (each controller)

Wiper Motor Rating:

Wiper motor operating voltage was 12V.

4.6 Communication:

An android application was developed, which is named as **Balancer**. Firstly the orientation sensor of the mobile was accessed and to see if the sensor is well calibrated or not we assigned red color for tilt of the device 35 degree or more towards left, blue color for tilt of the device 35 degree or more towards right and green for any other cases. Later on the threshold was set to 15 degree.

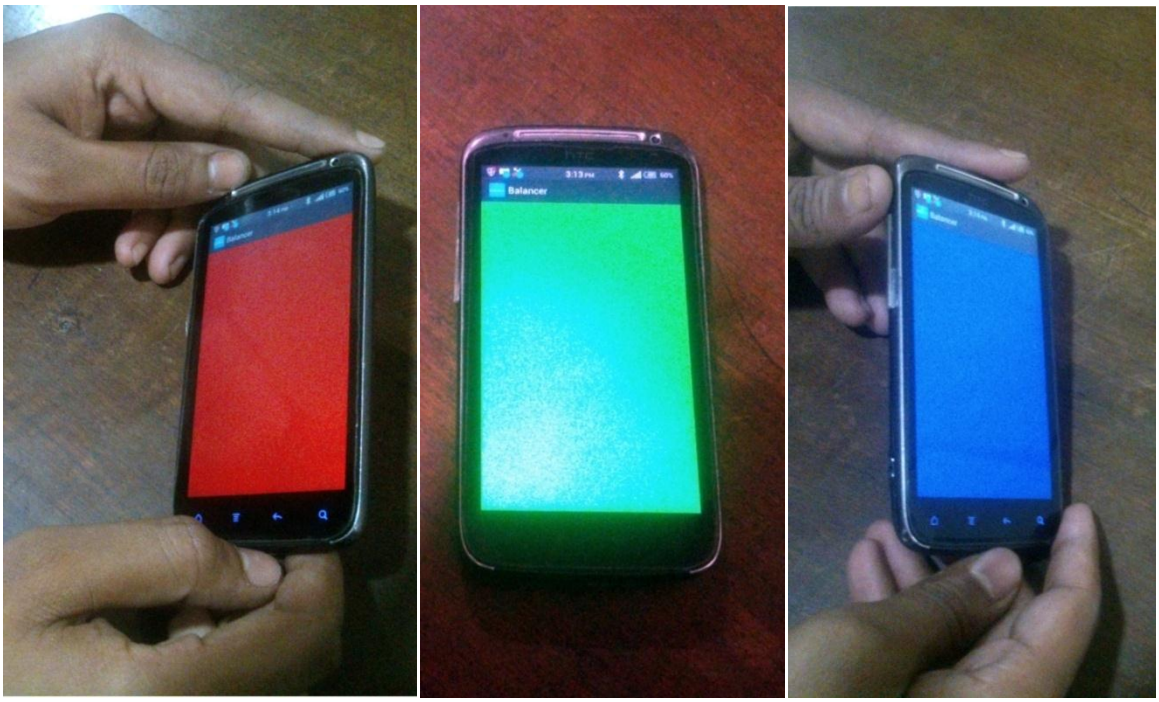

Figure 4.7: Android application showing tilt angle

4.6.1 Interfacing:

After being sure that mobile's orientation sensor is working perfectly, interfacing part was emphasized. A connection was established from the android to the arduino. Bluetooth medium has been found the most simple solution for this objective and as the Bluetooth receiver at the arduino end we used HC-05 bluetooth kit for arduino.

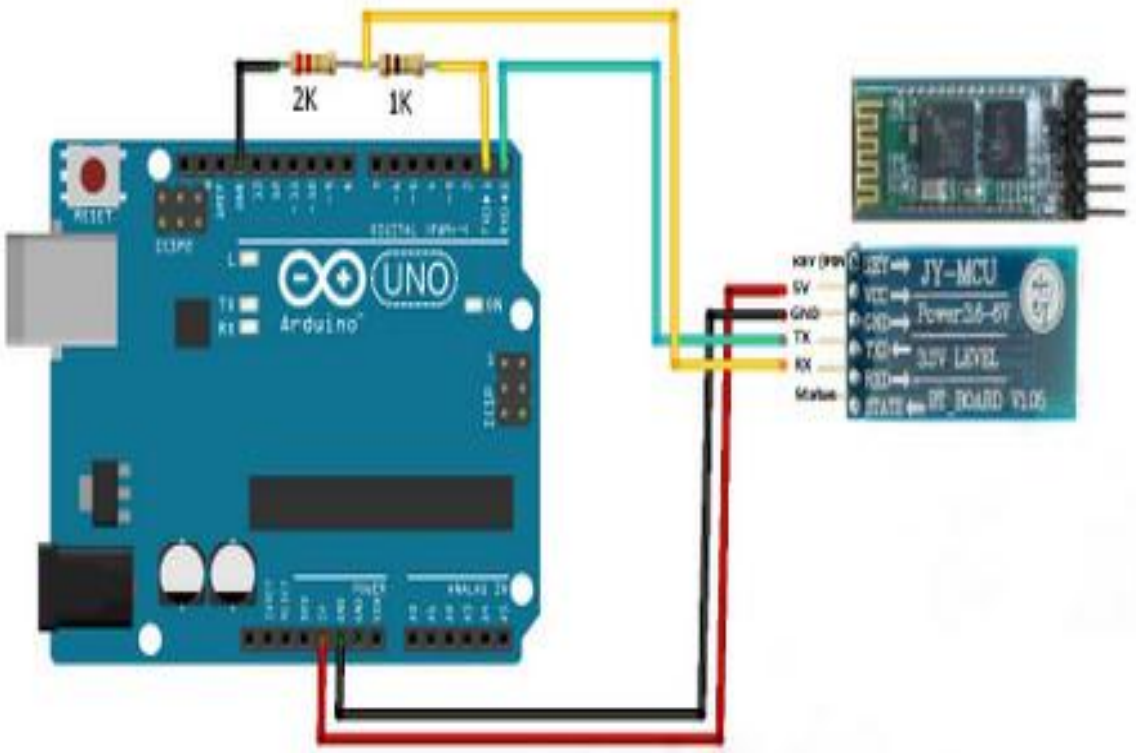

Figure4.8: Interfacing HC-05

Thus the android phone was interfaced with the arduino via Bluetooth module HC-05. The connection diagram is exactly as shown in the figure.

CHAPTER 5

Experimental Result

Initially experiment was conducted to determine whether the structure can be balanced on its own or not. Research found that with high RPM hub motor the structure can be stabilized. Then the concern was switching it manually after letting the structure lean. With the help of the wiper motor we could balance the chassis.

For manual control of switching the motor, we had to act really fast during the tilt to balance the structure. With perfect weight distribution, chassis can be balanced manually.

During the experiment that was done several times, one wiper motor got damaged and it had to be replaced with another one. One hub motor controller was also needed to be repaired because of short circuit. But finally, the chassis was managed to balance by rapid switching.

After successful manual switching, it was aimed to balance the chassis automatically. So the need of gyro sensor arose. And the built in orientation sensor of android was used.

The threshold angle in both sides was 35 degree that was measured by the orientation sensor of the android phone. The phone sent data according to the angle of the both sides to the arduino and the arduino operated the motor accordingly. It was observed that the arduino drives the motor successfully with the motor driver circuit but after sometimes due to excessive current, the circuit gets damaged.

CHAPTER 6

Discussion

Initially the concern was to operate the self balancing system with the gyroscope. A gyro sensor was designed to be with the chassis and the angle had to be sent to an arduino and arduino program is to decide which direction to change the orientation of the hub motors by what degree of tilt. Groove 3 axis digital gyroscope (ITG 3200) was considered as gyro sensor and interfaced with arduino.

However the ITG 3200 provides angular velocity as output which needed to be integrated with the sampling time frame to get the angular displacement. And after calibrating 800 times the gyro sensor was unable to provide any stable output.

Figure 5.1: Groove 3 axis Digital Gyro

This situation leads to develop an android application that gets the angle from built in orientation sensor because other gyroscopes are not available in this country.

After successfully interfacing the arduino with the android device, the next challenge was to drive the wiper motor with the motor driver circuit. The experiments were quite a bit successful but after sometimes the circuit gets burnt because of excessive current drawn by the wiper motor.

The circuit was tried to be rebuilt using a single MOSFET, then connecting several MOSFETs in parallel connection. However, still the circuit got damaged due to large amount of current.

Establishing a wired connection for this communication purpose was considered initially but a Bluetooth module was much more convenient in the circuit implementation and also for limited time span.

CHAPTER 7

Conclusion

This is a two wheeler vehicle has many more safety features than motorcycles which make it more reliable. Safety is one of our top priorities with this vehicle. However, the most important safety feature is our gyro stability system. This will keep the vehicle upright even in a collision, preventing the vehicle from flipping or rolling. On the other hand it will also protect the passenger from rain, wind, dust as it is a covered vehicle. It will also be more comfortable than any other motorbikes at the same time will require a very small space of parking. The idea of two wheeler self balanced vehicle is new. The vehicle is designed considering cost effectiveness and fuel efficiency factors.

From the thesis project some certain observation are provided:

- The force experienced due to the tilt of rotating wheels depends on the RPM of the wheel, the weight of the wheels and the angle of tilt.
- The higher the RPM, the bigger the counterforce. That means the counterforce is much larger when the RPM of the hub motor is larger.
- The direction of rotating wheel tilt determines the force direction of when spinning is in a particular direction.
- Weight attached to hub motor helps to stabilize the balancing.
- The more the tilt angle, the more force is needed to stabilizes the chassis.

This vehicle needs much more development for future works. Firstly two powerful hub motors are needed for balancing it. The RPM should be over a thousand with attached weight of at least 10-15kg or more. Then the weight should be distributed equally in the chassis. If the weight is more on the right, the vehicle will be tilted on that side more. This is applicable for the left side too. So this unequal distribution of weight can be a problem during balancing.

Again, a calibrated gyroscope is needed to get the tilt angle. A Wired connection with the gyroscope to arduino can be much more convenient for the set up. A PCB can be very helpful too for the motor driving circuit as it reduces wires and thus the reliability of the connection increases.

7.1 Limitation:

The project is so far has managed to discover the principle of self balancing. It is managed to design an algorithm for to keep the frame straight all time. However, the frame is not quite able to balance fully on its own. It creates force to a certain direction on the change of orientation of hub motors. But the hub motors are a bit overweighed after adding some additional weight. This large weight along with the force due to the change of orientation of rotating disk creates such a momentum with get the frame to the other side rather than keeping it straight. Switching fast does not help, as the initial impact force is enough to create the big momentum which pushes the frame beyond our expected limit. The disks are tilted to the other direction to compensate this event. However, till the total frame is leaned close to our threshold angle it generates greater momentum to diminish the force provided by the rotating disks and fell on the other side.

The whole system worked perfectly. However, the wiper motor draws much current due to its high torque and the amount of load it has to handle. So the motor driver circuit burns immediately after providing few outputs.

The system is designed to stable the car while standing still. The dynamics are totally different when the wheels of the car are rolling. The higher the speed of the car the lesser the threshold angle would be.

7.2 Future Scope:

The car is designed to provide safety as well as energy efficiency. It is designed to accommodate a single passenger due to its compact size. The thesis emphasized on the self-stabilization technology.

There are scopes to design its power distribution system and the driving motor can be emphasized later on. We recommend having DC power source for driving the car for energy efficiency as well as fuel can be a secondary power source. This might be a hybrid drive system [4] or fuel tank can be a reserved energy for emergency situation. The car can be designed to transport two passengers rather than one. The passenger can be accommodated behind the driver.

7.3 Appendix:

Arduino code for balancer:

Balance Duino

```
/*
 PROJECT: ArduDroid
 PROGRAMMER: Hazim Bitar (techbitar at gmail dot com)
 DATE: Oct 31, 2013
 FILE: ardudroid.ino
 LICENSE: Public domain
*/

#define START_CMD_CHAR '*'
#define END_CMD_CHAR '#'
#define DIV_CMD_CHAR '|'
#define CMD_DIGITALWRITE 10
#define CMD_ANALOGWRITE 11
#define CMD_TEXT 12
#define CMD_READ_ARDUDROID 13
#define MAX_COMMAND 20  // max command number code. used for error checking.
#define MIN_COMMAND 10  // minimum command number code. used for error checking.
#define IN_STRING_LENGHT 40
#define MAX_ANALOGWRITE 255
#define PIN_HIGH 3
#define PIN_LOW 2

String inText;

void setup() {
```

```
  Serial.begin(9600);
  Serial.println("ArduDroid 0.12 Alpha by TechBitar (2013)");
  Serial.flush();
}

void loop()
{
  Serial.flush();
  int ard_command = 0;
  int pin_num = 0;
  int pin_value = 0;

  char get_char = ' ';  //read serial

  // wait for incoming data
  if (Serial.available() < 1) return; // if serial empty, return to loop().
  String data = "";
  while (Serial.available())  {
     char c = Serial.read();  //gets one byte from serial buffer
     delay(5);
     if (c == END_CMD_CHAR) { // if we the complete string has been read
       // add your code here
       break;
     }
     else {
       if (c !=  DIV_CMD_CHAR) {
         data += c;
         delay(5);
       }
     }
   }
  Serial.println(data);
```

```
  //================================
  if (data == "8")
  {
    digitalWrite(8, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);
  }
  else if (data == "9")
  {
    digitalWrite(8, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(10, LOW);
  }
  else if (data == "10")
  {
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    digitalWrite(10, HIGH);
  }
  //================================
  // parse incoming command start flag
  /*
  get_char = Serial.read();
  Serial.println("Data Found");
  Serial.println(get_char);
  if (get_char != START_CMD_CHAR) return; // if no command start flag, return
to loop().

  // parse incoming command type
  ard_command = Serial.parseInt(); // read the command
```

```
// parse incoming pin# and value
pin_num = Serial.parseInt(); // read the pin
pin_value = Serial.parseInt();  // read the value

// 1) GET TEXT COMMAND FROM ARDUDROID
if (ard_command == CMD_TEXT){
  inText =""; //clears variable for new input
  while (Serial.available())  {
    char c = Serial.read();  //gets one byte from serial buffer
    delay(5);
    if (c == END_CMD_CHAR) { // if we the complete string has been read
      // add your code here
      break;
    }
    else {
      if (c !=  DIV_CMD_CHAR) {
        inText += c;
        delay(5);
      }
    }
  }
}
*/
// 2) GET digitalWrite DATA FROM ARDUDROID

if (ard_command == CMD_DIGITALWRITE){
  if (pin_value == PIN_LOW) pin_value = LOW;
  else if
  (
    pin_value == PIN_HIGH
  )
  pin_value = HIGH;
```

```
    else
    {
    return;
    } // error in pin value. return.
    set_digitalwrite( pin_num,  pin_value);  // Uncomment this function if you wish
to use
    return;  // return from start of loop()
  }

  // 3) GET analogWrite DATA FROM ARDUDROID
  if (ard_command == CMD_ANALOGWRITE) {
    analogWrite(  pin_num, pin_value );
    // add your code here
    return;  // Done. return to loop();
  }

  // 4) SEND DATA TO ARDUDROID
  if (ard_command == CMD_READ_ARDUDROID) {
    // char send_to_android[] = "Place your text here." ;
    // Serial.println(send_to_android);   // Example: Sending text
    Serial.print(" Analog 0 = ");
    Serial.println(analogRead(A0));  // Example: Read and send Analog pin value to
Arduino
    return;  // Done. return to loop();
  }
}

// 2a) select the requested pin# for DigitalWrite action
void set_digitalwrite(int pin_num, int pin_value)
{

  Serial.println("num: ");
```

```
Serial.println(pin_num);

//====================
pin_value = 1;
//====================

switch (pin_num) {
case 13:
  pinMode(13, OUTPUT);
  digitalWrite(13, pin_value);
  // add your code here
  break;
case 12:
  pinMode(12, OUTPUT);
  digitalWrite(12, pin_value);
  // add your code here
  break;
case 11:
  pinMode(11, OUTPUT);
  digitalWrite(11, pin_value);
  // add your code here
  break;
case 10:
  pinMode(10, OUTPUT);
  digitalWrite(10, pin_value);
  // add your code here
  break;
case 9:
  pinMode(9, OUTPUT);
  digitalWrite(9, pin_value);
  // add your code here
  break;
```

```
case 8:
  pinMode(8, OUTPUT);
  digitalWrite(8, pin_value);
  // add your code here
  break;
case 7:
  pinMode(7, OUTPUT);
  digitalWrite(7, pin_value);
  // add your code here
  break;
case 6:
  pinMode(6, OUTPUT);
  digitalWrite(6, pin_value);
  // add your code here
  break;
case 5:
  pinMode(5, OUTPUT);
  digitalWrite(5, pin_value);
  // add your code here
  break;
case 4:
  pinMode(4, OUTPUT);
  digitalWrite(4, pin_value);
  // add your code here
  break;
case 3:
  pinMode(3, OUTPUT);
  digitalWrite(3, pin_value);
  // add your code here
  break;
case 2:
  pinMode(2, OUTPUT);
```

```
      digitalWrite(2, pin_value);
      // add your code here
      break;
      // default:
      // if nothing else matches, do the default
      // default is optional
    }
  }
```

Android Code:

```
package com.level11corp.balancer;

import java.io.IOException;
import java.util.ArrayList;
import android.app.ListActivity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.AsyncTask;
```

```
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends ListActivity {
        TextView et;
        RelativeLayout rl;
        final static String TAG = "arup";
        SensorManager sensorManager;
        int orientationSensor;
        float rollAngle;
        private boolean connected=false;
        //public final static String UUID = "3606f360-e4df-11e0-9572-0800200c9a66";
        public final static String UUID = "00001101-0000-1000-8000-00805F9B34FB";
        BluetoothAdapter bluetoothAdapter;
        BroadcastReceiver discoverDevicesReceiver;
        BroadcastReceiver discoveryFinishedReceiver;
//      // ---store all the discovered devices---
        ArrayList<BluetoothDevice> discoveredDevices;
        ArrayList<String> discoveredDevicesNames;
//      // ---thread for running the server socket---
        ServerThread serverThread;
//      // ---thread for connecting to the client socket---
```

```
ConnectToServerThread connectToServerThread;
@SuppressWarnings("deprecation")
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et = (TextView) findViewById(R.id.et);
        rl = (RelativeLayout) findViewById(R.id.rl);

        // ---init the ArrayList objects and bluetooth adapter---
        discoveredDevices = new ArrayList<BluetoothDevice>();
        discoveredDevicesNames = new ArrayList<String>();
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        bluetoothAdapter.enable();
        try{
                Thread.sleep(3000);
        }catch(InterruptedException ie){
                Log.d(TAG,""+ie);
        }
        DiscoveringDevices();
        // ---start the socket server---
        serverThread = new ServerThread(bluetoothAdapter);
        serverThread.start();
        sensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
        orientationSensor = Sensor.TYPE_ORIENTATION;
        sensorManager.registerListener(sensorEventListener,
                        sensorManager.getDefaultSensor(orientationSensor),
                        SensorManager.SENSOR_DELAY_NORMAL);
```

```
        }

        final SensorEventListener sensorEventListener = new SensorEventListener() {
                @SuppressWarnings("deprecation")
                public void onSensorChanged(SensorEvent sensorEvent) {
                        if (sensorEvent.sensor.getType() ==
Sensor.TYPE_ORIENTATION) {

                                rollAngle = sensorEvent.values[2];
                                Log.d(TAG, "Roll: " + String.valueOf(rollAngle));
                                int rollvalue = (int) rollAngle;
                                if (rollvalue > 35) {
                                        et.setText("Roll: LeftDown. Angle: " + rollAngle);
                                        rl.setBackgroundColor(android.graphics.Color.RED);
                                        SendMessage("8");
                                }
                                if (rollvalue < -35) {
                                        et.setText("Roll: RightDown. Angle: " + rollAngle);

        rl.setBackgroundColor(android.graphics.Color.BLUE);
                                        SendMessage("9");
                                }
                                if ((rollvalue > -15) && (rollvalue < 15)) {
                                        et.setText("Roll: Normal");
                                rl.setBackgroundColor(android.graphics.Color.GREEN);
                                        SendMessage("10");
                                        }
                        }
                }
```

```
        public void onAccuracyChanged(Sensor senor, int accuracy) {
        }
};
int cancel = 0;
public void onBackPressed() {
        cancel++;
        Toast.makeText(getApplicationContext(), "press again to stop",
                        Toast.LENGTH_SHORT).show();
        final Thread t = new Thread() {

                @Override
                public void run() {
                        // TODO Auto-generated method stub
                        try {
                                sleep(1500);
                        } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } finally {
                                cancel = 0;
                        }
                }
        };
        t.start();
        if (cancel == 2) {
                onDestroy();
                sensorManager.unregisterListener(sensorEventListener);
                finish();
        }
```

```
};
//        // ---used to discover other bluetooth devices---
private void DiscoveringDevices() {
        if (discoverDevicesReceiver == null) {
                discoverDevicesReceiver = new BroadcastReceiver() {
                        // ---fired when a new device is discovered---
                        @Override
                        public void onReceive(Context context, Intent intent) {
                                String action = intent.getAction();
                                // ---a device is discovered---
                                if
(BluetoothDevice.ACTION_FOUND.equals(action)) {
                                        // ---get the BluetoothDevice object from
                                        // the Intent---
                                        BluetoothDevice device = intent

        .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                                        // ---add the name and address to an array
                                        // adapter to show in a ListView---
                                        // ---only add if the device is not already
                                        // in the list---
                                        if (!discoveredDevices.contains(device)) {
                                                // ---add the device---
                                                discoveredDevices.add(device);
                                        // ---add the name of the device; used for
                                                // ListView---
                                        discoveredDevicesNames.add(device.getName());
                                                // ---display the items in the ListView--
                                                setListAdapter(new
ArrayAdapter<String>(
```

```
                                        getBaseContext(),
                                    android.R.layout.simple_list_item_1,
                                        discoveredDevicesNames));
                                }
                            }
                        };
                }
                if (discoveryFinishedReceiver == null) {
                        discoveryFinishedReceiver = new BroadcastReceiver() {
                            // ---fired when the discovery is done---
                            @Override
                            public void onReceive(Context context, Intent intent) {
                                // ---enable the listview when discovery is over;
                                // about 120 seconds---
                                getListView().setEnabled(true);
                                Toast.makeText(
                                        getBaseContext(),
                                        "Discovery completed. Select ur
arduino "
                                                + "to start",
Toast.LENGTH_LONG)
                                        .show();
                                unregisterReceiver(discoveryFinishedReceiver);
                            }
                        };
                }
                // ---register the broadcast receivers---
                IntentFilter filter1 = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
```

```
		IntentFilter filter2 = new IntentFilter(
				BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
		registerReceiver(discoverDevicesReceiver, filter1);
		registerReceiver(discoveryFinishedReceiver, filter2);		// ---disable the listview when discover is in progress---
		getListView().setEnabled(true);
		Toast.makeText(getBaseContext(),
				"Discovery in progress...please wait...",
Toast.LENGTH_LONG)
				.show();
		bluetoothAdapter.startDiscovery();
	}
//	// ---used for updating the UI on the main activity---
	static Handler UIupdater = new Handler() {
		@Override
		public void handleMessage(Message msg) {
			int numOfBytesReceived = msg.arg1;
			byte[] buffer = (byte[]) msg.obj;
	// for now this handler has no use. but may need later if arduino sends back any data which needs to retrieve.
		}
	};
//	@Override
	public void onDestroy() {
		super.onDestroy();
		// ---cancel discovery of other bluetooth devices
		bluetoothAdapter.cancelDiscovery();
		bluetoothAdapter.disable();
		// ---unregister the broadcast receiver for
		// discovering devices---
```

```
            if (discoverDevicesReceiver != null) {
                try {
                    unregisterReceiver(discoverDevicesReceiver);
                } catch (Exception e) {
                }
            }
            // ---if you are currently connected to someone...---
            if (connectToServerThread != null) {
                try {
                    // ---close the connection---
                    connectToServerThread.bluetoothSocket.close();
                } catch (IOException e) {
                    Log.d("MainActivity", e.getLocalizedMessage());
                }
            }
            // ---stop the thread running---
            if (serverThread != null)
                serverThread.cancel();
        }
//      // ---when a client is tapped in the ListView---
        public void onListItemClick(ListView parent, View v, int position, long id) {
            // ---if you want to communicate with another arduino---
            if (connectToServerThread != null) {
                try {
                    // ---close the connection first---
                    connectToServerThread.bluetoothSocket.close();
                    bluetoothAdapter.cancelDiscovery();
                } catch (IOException e) {
```

```
                    Log.d("MainActivity", e.getLocalizedMessage());
}
        }
        // ---connect to the selected Bluetooth device---
        BluetoothDevice deviceSelected = discoveredDevices.get(position);
        connectToServerThread = new ConnectToServerThread(deviceSelected,
                    bluetoothAdapter);
        connectToServerThread.start();
                                try{
                    Thread.sleep(3500);
                }catch(InterruptedException e){
                    Log.d(TAG, e.toString());
                }finally{
                    connected= true;

                }
        Toast.makeText(
                    this,
                    "You have connected to "
                                + discoveredDevices.get(position).getName(),
                    Toast.LENGTH_SHORT).show();

                //      parent.setVisibility(View.INVISIBLE);
}
private class WriteTask extends AsyncTask<String, Void, Void> {
        protected Void doInBackground(String... args) {
                try {
                    connectToServerThread.commsThread.write(args[0]);
                } catch (Exception e) {
```

```
                    Log.d("MainActivity", e.getLocalizedMessage());
                }
                return null;
            }
        }
        // ---send a message to the connected socket client---
        public void SendMessage(String str) {
            if(connected){
            if (connectToServerThread != null) {
                if(str!=null){
                new WriteTask().execute(str);
                }else{
                    Toast.makeText(getApplicationContext(), "nothing in the str
", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "Select a client first",
Toast.LENGTH_SHORT)
                        .show();
            }
            }
    }
}
```

References:

1. http://www.techbitar.com/uploads/2/0/3/1/20316977/5040796_orig.jpg [access date: 23.04.14]
2. http://auto.howstuffworks.com/wiper1.htm [access date: 18.04.14]
3. "Stability Analysis of a Two-wheeler during Curve Negotiation under Braking" By M Ghosh , S Mukhopadhyay
4. http://www.toyota-global.com/innovation/environmental_technology/hybrid [access date: 10.04.14]
5. www.litmotors.com [access date: 13.04.14]
6. http://www.robotshop.com/en/pmod-high-performance-gyro-sensor.html [access Date: 19.04.14]
7. http://t1.gstatic.com/images?q=tbn:ANd9GcSBljAJzutwWFk6AEdJaGJy8AR5WmKuACKZpUhn3d0JXLNdZUL [access date: 20.04.14]