SELF BALANCING ROBOT USING ARDUINO

A Report on Mini project work carried out

&

Submitted in partial fulfilment of the requirements
for qualifying the course

Mini Project Work (U14EC609)

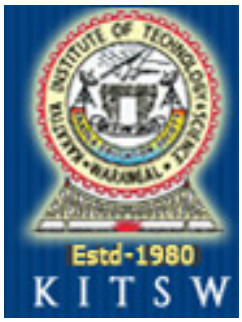
in

VI-Semester of ECE

by

CH.V.S BHARGAVA REDDY
B15EC065

Under the guidance of
A.Srinivas
Assistant Professor,
Dept of ECE

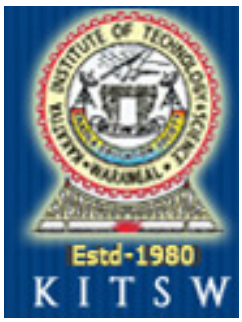


DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE:: WARANGAL – 15 (T.S)
(An Autonomous Institute under Kakatiya University, Warangal)
April, 2017-18

DEPARTMENT OF ELECTRONICS & COMMUNICATIONENGINEERING

KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE:: WARANGAL – 15 (T.S)

(An Autonomous Institute under Kakatiya University, Warangal)



CERTIFICATE

This is to certify that **CH.V.S BHARGAVA REDDY** bearing **Roll No. B15EC065** has successfully completed Mini Project work entitled **"SELF BALANCING ROBOT USING ARDUINO"** in complete fulfilment for qualifying the Mini project work course in VI-semester of Electronics and Communication Engineering during this academic year **2017-18.**

Internal Guide :

A.Srinivas
Assistant.Professor ,
Dept. of ECE

Head of the Department:

Dr. G. Raghotham Reddy
Professor,
Dept. of ECE

DECLARATION

I declare that this Project entitled **"SELF BALNCING ROBOT USING ARDUINO"** is original and bonafide work of my own and is submitted in fulfilment for qualifying the Mini project work course(U14EC609) in VI-semester to the Department of Electronics & Communication Engineering, **KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE**, Warangal, and has not been copied from any earlier or other reports. The conclusion and results in this report are based on my own work.

CH.V.S BHARGAVA REDDY
B15EC065

ACKNOWLEDGEMENT

I express our immense pleasure with a profound feeling of reverence and gratitude to **Sri.A.Srinivas,** Asst. Professor in Electronics and communication Engineering Department, for his inspiring and valuable guidance throughout this project and as well as being my Project Guide.

I wish to express our sincere thanks to **Mr.V.Raju ,** Asst. Professor, for his inspiration and being our project coordinator.

I wish to express our gratitude to **Dr.G.RAGOTHAM REDDY,** Head of Department of Electronics and Communication Engineering and all the staff members of Electronics and Communication Engineering Department for their encouragement and support.

I place my sincere thanks to **Dr.P.VENKATESHWAR RAO**, **Principal** of Kakatiya Institute of Technology & Science, Warangal and **Dr.Y.MANOHAR**, Director, Kakatiya Institute of Technology & Science, Warangal for providing all the facilities required for completing this project work.

ABSTRACT

This project will undertake the construction and implementation of a two-wheeled robot that is capable of balancing itself. The structural, mechanical, and electronic components of the bot will be assembled in a manner that produces an inherently unstable platform that is highly susceptible to tipping in one axis.

The wheels of the robot are capable of independent rotation in two directions, each driven by a servo motor. Information about the angle of the device relative to the ground (i.e. tilt) will be obtained from sensors on the device. The precise type of sensor that will be used is yet to be specified. The tilt sensor may be an accelerometer, gyroscopic sensor, or IR sensor (to measure distance to the ground). Information from the sensors will be fed back to the Z8, which will process the feedback using a crude proportional, integral, derivative (PID) algorithm to generate compensating position control signals to the servo motors in order to balance the device.

INDEX

LIST OF FIGURES:

LIST OF TABLES:

CHAPTER 1

INTRODUCTION

To make a robot which can balance itself on two wheels. There will be only one axle connecting the two wheels and a platform will be mounted on that .There will be a another platform above it. The platform will not remain stable itself. Our job will be to balance the platform using distance sensors and to maintain it horizontal. At first we have decided to just balance the robot on its two wheels .If the platform inclines then microcontroller(in this case it is Arduino) will send signals to motors such that motors would move forward or backward depending on the inclination direction and extent. So if the platform tilts forward then motors will run forward and viceversa to keep the platform horizontal. For this we will need to code the Arduino in order to perform job according to this.

Technique used in making the self balancing robot is same as the principle used in balancing of the Inverted Pendulum.Balancing of the inverted pendulum is a classic problem in dynamics and control theory and is widely used as a benchmark for testing control algorithms (PID controllers, neural networks, fuzzy control, genetic algorithms, etc.). Variations on this problem include multiple links, allowing the motion of the cart to be commanded while maintaining the pendulum, and balancing the cart-pendulum system on a see-saw. The inverted pendulum is related to rocket or missile guidance, where the center of gravity is located behind the center of drag causing aerodynamic instability. The understanding of a similar problem can be shown by simple robotics in the form of a balancing cart. Balancing an upturned broomstick on the end of one's finger is a simple demonstration, and the problem is solved in the technology of the Segway PT, a self-balancing transportation device.

Making a self-balancing robot is essentially solving the classic inverted pendulum problem. The goal of the control loop is to adjust the wheels' position so that the inclination angle remains stable at a pre-determined value (e.g. the angle when the robot is balanced). When the robot starts to fall in one direction, the wheels

should move in the falling direction to correct the inclination angle. When the deviation from equilibrium is small, wheels move gently and when the deviation is large wheels move more quickly .

1.1 PROJECT OVERVIEW

Technique used in making the self balancing robot is same as the principle used in balancing of the Inverted Pendulum. Balancing of the inverted pendulum is a classic problem in dynamics and control theory and is widely used as a benchmark for testing control algorithms (PID controllers, neural networks, fuzzy control, genetic algorithms, etc.). Variations on this problem include multiple links, allowing the motion of the cart to be commanded while maintaining the pendulum, and balancing the cart-pendulum system on a see-saw. The inverted pendulum is related to rocket or missile guidance, where the center of gravity is located behind the center of drag causing aerodynamic instability. The understanding of a similar problem can be shown by simple robotics in the form of a balancing cart. Balancing an upturned broomstick on the end of one's finger is a simple demonstration, and the problem is solved in the technology of the segwet PT, a self-balancing transportation device.

This report documents the design and implementation of a self-balancing robot, which is an unstable system; the basic model is that of an inverted pendulum balancing on two wheels. This work details the derivation of the model of the system and lays out the framework of the robot's control system. It also shows the full implementation of a control system stabilizing the robot. The robot is built using Lego Mindstorm, an educational product first released by Lego in 1998. The Lego Mindstorm kit is equipped with a 32-bit ARM7 microprocessor with a bootloader modified to run the nxtOSEK real-time operating system.

1.2 SYSTEM WORKING

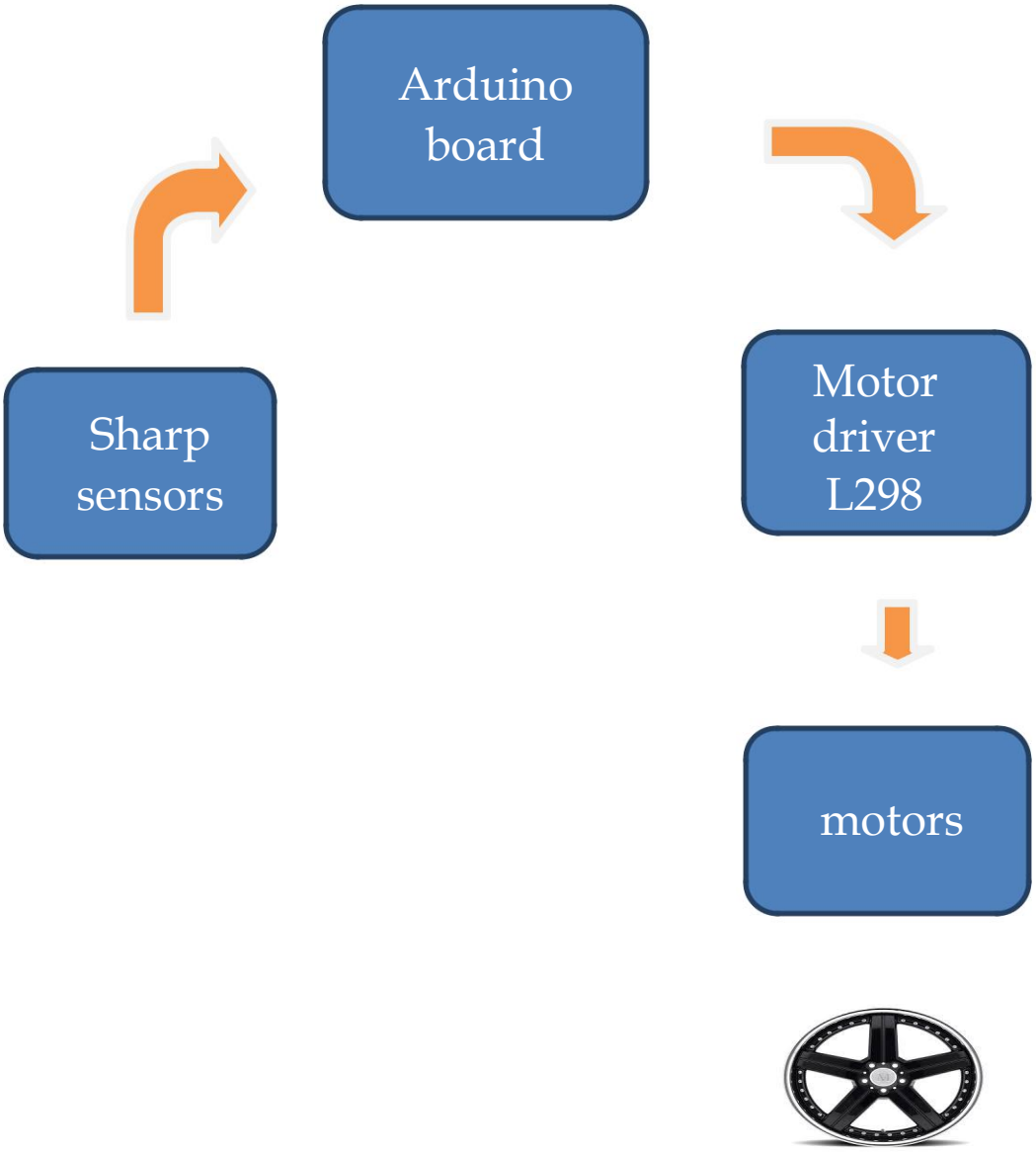


Figure 1.2 :system working

Sensors send the analog input voltages to the arduino. These analog voltages lie in the range from 0V to 5V. Arduino reads these analog inputs and converts them to digital using

inbuilt ADC (analog to digital converter). Then it calculates the difference between these values (sensor1 value - sensor2 value). Using this difference , prev_difference (difference at previous loop execution time)and PID control mechanism Arduino microcontroller determines the value of output PWM that should be fed to the motor driver. The code here makes it more clear.

To increase the effect of D (differential) we have used a time measuring function (millis()) in the code. This time measuring function measures time in milliseconds. Modified code is here. This modification was necessary because loop execution time was so small to detect any considerable change in the difference value. the modified part of code measures difference (bet sensor1 value and sensor2 value) regularly after some fixed interval (eg 10 milli seconds). so the change in difference(i.e., difference- prev difference) is considerable. We have not used I (integration) here in the PID control as we found P+D sufficient to balance the robot.

CHAPTER 2

(i) Arduino UNO:

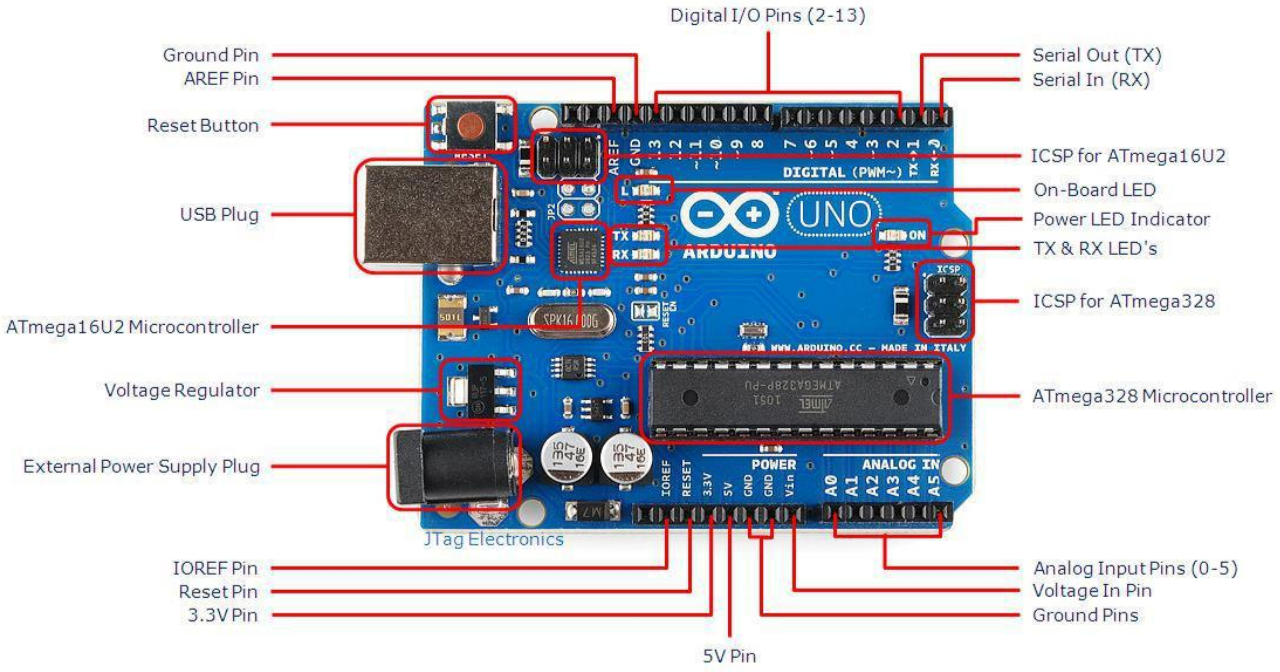


Fig :2.1

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC -to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were

the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

Technical Specifications:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Table 2.1

Documentation of Arduino:

Programming

The Arduino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK5 00 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

Memory:

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output:

See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical.

Each of the 14 digital pins on the Uno can be used as an inpu t or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20 -50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication:

Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

(ii) Mpu 6050

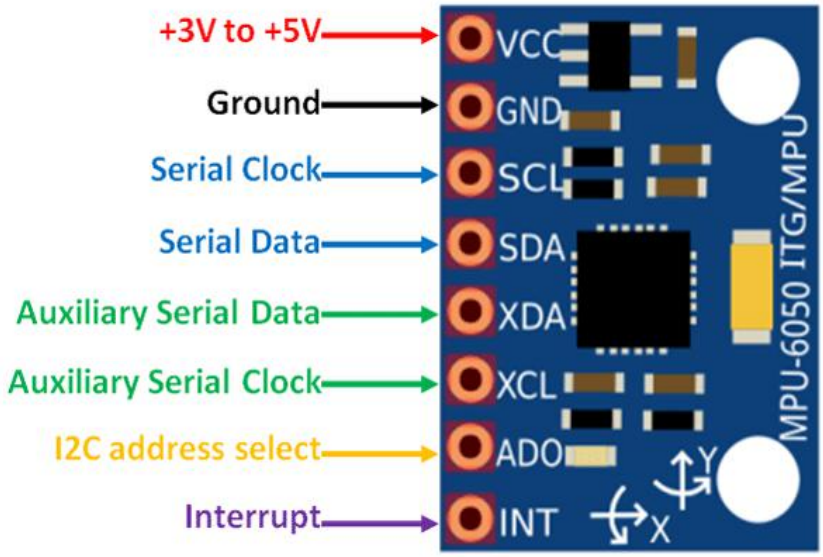


fig 2.2

This sensor combines a 3-axis MEMS gyroscope and a 3-axis MEMS accelerometer, is capable of processing complex 9-axis algorithms, captures the x, y, & z channel at the same time, and is very accurate due to the 16-bits analog to digital conversion hardware for each channel. The MPU-6050 also removes the cross-axis alignment problems that can occur on discrete parts.

Raw Values:

For reading the raw values from the accelerometer and gyro registers, the sleep mode has to be disabled. The sensor values can also be programmed to be placed in the 1024byte FIFO buffer, which can then be read by a Chineduino Uno Rev3 [Product Link]. If the MPU-6050 places data in the FIFO buffer, it signals the Uno-R3 using the interrupt signal so Uno-R3 knows that there is data in the FIFO buffer waiting to be read.

Control I2C Device:

The MPU-6050 acts as a slave to the Arduino with the SDA and SCL pins connected to the I2C-bus. Beside the normal I2C-bus, it has it's own I2C controller, to be a master on a second(sub) I2C-bus. It uses the pins AUX_DA (XDA) and AUX_CL (XCL) for that second (sub)-I2C-bus which allows it to control, for example, a magnetometer. The values of the magnetometer can be passed on to the Uno-R3.

Digital Motion Processor:

This sensor has a Digital Motion Processor(DMP), also called a Digital Motion Processing Unit. This DMP can be programmed with firmware and is able to do complex calculations with the sensor values. For this DMP, InvenSense has a discouragement policy, by not supplying enough information how to program the DMP. The DMP can do fast calculations directy on the chip, and is even able to do calculations with the sensor values of another chip, connected to the second(sub) I2C-bus. This reduces the load for the microcontroller.

Features:

- Standard I2C communications protocol
- Built-in 16bit ADC
- 16-bit data output
- Tri-Axis Gyro with a sensitivity up to 131 LSBs/dps and a full-scale range of ± 250, 500, 1000, 2000°/s (dps)
- Tri-Axis Accelerometer with a programmable full scale range of ± 2, ± 4, ± 8, ± 16g
- Digital-output temperature sensor
- Pin pinch 0.1 inch

(iii) Motor Shield

3.1.1 DESCRIPTION

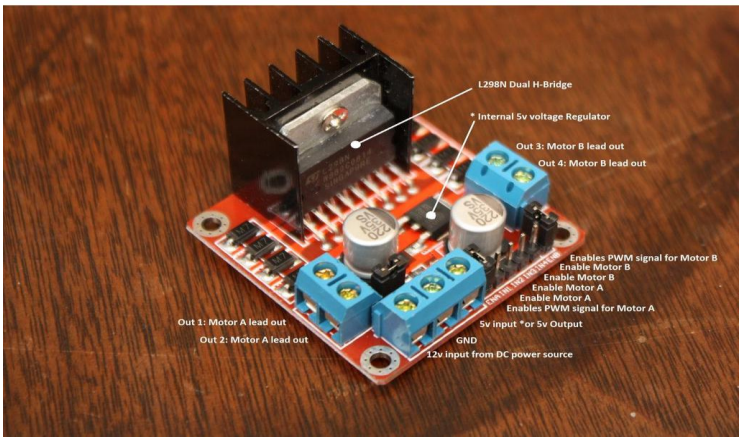
Motor driver IC (L293d or L293D) has input pins, output pins, enable pin , logic supply voltage pin , Main supply voltage pin. L298N is a 15 pin motor driver IC that can drive two motors at a time. It allows high current (2A per motor) to flow through it. So it is useful for driving motors which need high current to run. Pin connections for the IC L298N are given in the above diagram. Input pins are connected to the micro controller (in this case it is Aruino). Arduino applies PWM across these pins . Output pins are connected to the motors. Enable pin and logic supply pin are given the voltage that is defined as logic high(here it it 5 V). Main supply voltage pin is joined to the main power supply (max 46V for L298N, here it is 12 V).

In our motor driver circuit we joined the current sensing pins directly to ground. Any resistance joined between this pin and the ground will limit the current flowing through the

output pins of the IC depending upon the value of the resistance and therefore protect it from burning.But as our motors do not take much more current than 2A for normal application,we did not need any resistance. It is imp to mention it here because we wasted our 2 days just to determine whether to join current sense pin directly to ground or not and to determine what resistance should be used. There are its (current sensing pin's) other uses also (like controlling using feedback)

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo- Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs.

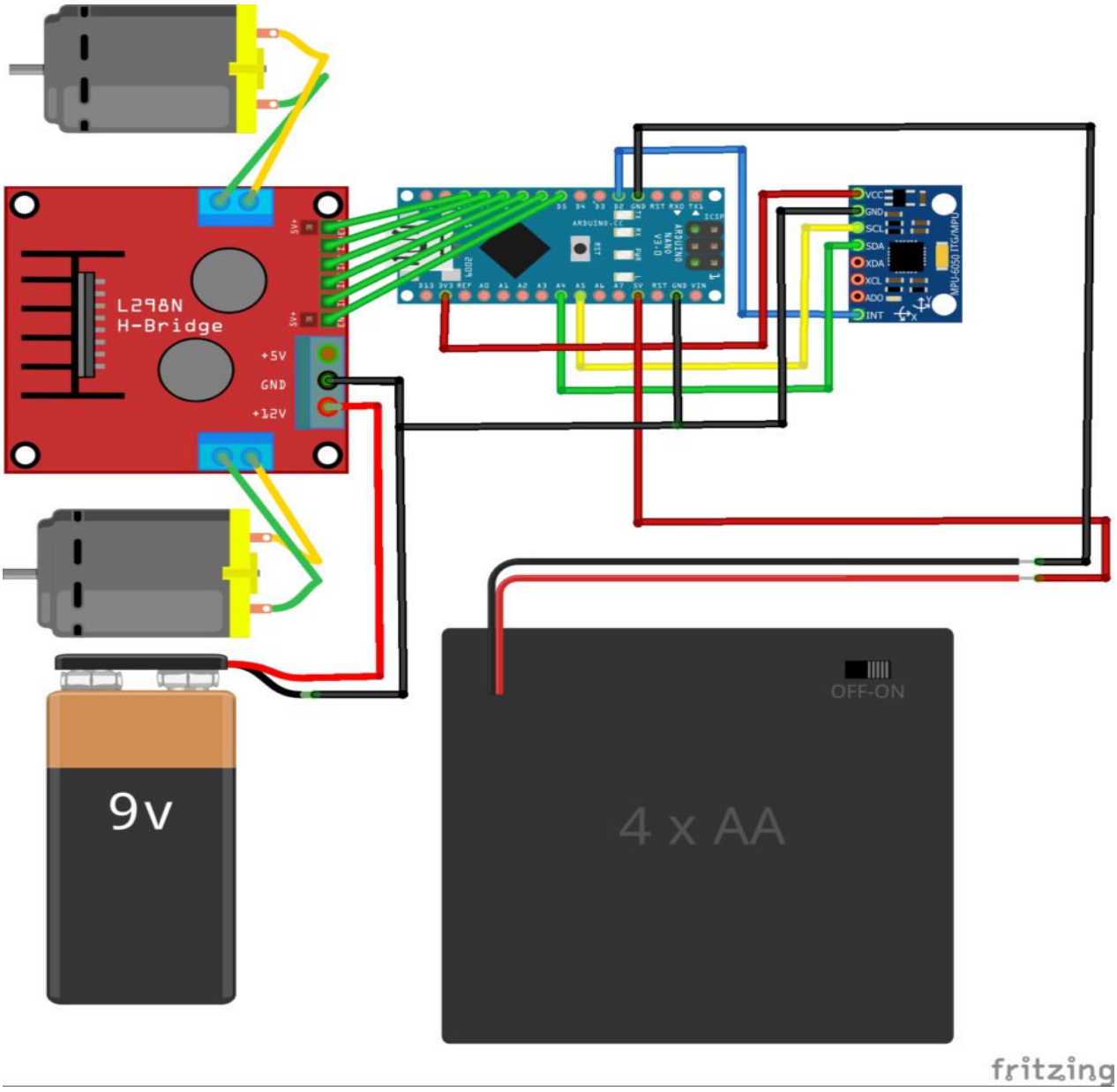


CHAPTER 3

WORKING PRINCIPLE

- First the information about the tilted angle is given by the Accelerometer(MPU-6050 which is present at the handle of the frame) to the Arduino.
- This information is a digital value whose range is between -16384 to +16384
- Now the Arduino maps the information between -180 to +180.
- Now the angles information are fed to Servo motors so that they compensate the angular rotation
- Now we give some delay (5 milli seconds) so that the motors can work as they are slow devices.

Circuit diagram:



Chapter 4

Arduino Code

```
#include<Wire.h>
const int MPU_addr=0x68;  // I2C address of the MPU-6050
int16_t AcX;
int rf=3,rb=5,lf=6,lb=9;
void setup(){
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.begin(9600);
  pinMode(rf,OUTPUT);
  pinMode(rb,OUTPUT);
  pinMode(lf,OUTPUT);
  pinMode(lb,OUTPUT);
}
void loop(){
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C
(ACCEL_XOUT_L)
  Serial.print("AcX = "); Serial.print(AcX);
  AcX=map(AcX,-16000,16000,-255,255);
  if(AcX>=0 && AcX<=120)
  {
    digitalWrite(rf,HIGH);
    digitalWrite(rb,LOW);
    digitalWrite(lf,HIGH);
    digitalWrite(lb,LOW);
  }
  if(AcX<0)
  {
    digitalWrite(rb,HIGH);
```

```
    digitalWrite(rf,LOW);
    digitalWrite(lb,HIGH);
    digitalWrite(lf,LOW);
  }

  delay(3);
}
```

Chapter 5

CONCLUSION

Researchers could build on what is researched until now. There are a few experiments that are unaccomplished. That is the main drawback that hampers the overall project as concreteresult its unable to attain. Therefore appropriate conclusions are not able to achieve. The problem with the oscillation still remains with the system and future work has to be done to achieve a stable solution.

FUTURE WORK:

The stabilization provided by the reaction wheel is limited be the torque provided by the reaction wheel motor. Subsequent plan is to use a rotating disc and its gyroscopic precession for balancing. This would provide a more stable design capable of providing higher restoring torque .In such a case particular attention should be paid to any rotary axes, their alignment, and how they are fixed to the model, to the position and alignment of brackets, and to the mounting and fastening of any flexible couplings. In addition to this, fuzzy logic controller can also be implemented to provide flexibility and accuracy in control.

Chapter 6

REFERENCE

[1]Brennan, L. (1905) U.S. Patent No. 796, 893. Washington, D.C.: U.S. Patent and Trademark Office.

[2]Carter, De Rubis, Guiterrez, Schoellig, Stolar. "Gyroscopically Balanced Monorail System Final Report" (2005) Columbia University.

[3]E. Ferreira, S. Tsai, C. Paredis, and H. Brown "Advanced Robotics, Vol. 14, No. 6, June, 2000, pp. 459 - 475.

[4]C.H. Ross, J. C. Hung, "Stabilization of an Unmanned Bicycle," Proc. IEEE Region III Convention, 1968, pp. 17.4.1-17.4.8.

[5]Gallaspy, J. "Gyroscopic Stabilization of an Unmanned Bicycle." Ph.D. Thesis, Auburn University (2000).

[6] Anderson, D.P, 'Nbot, a two wheel balancing robot',<http://www.geology.smu.edu/~dpa-www/robo/nbot>

[7] Steve Hassenplug, 2002, 'Steve's Legway', <http://www.teamhassenplug.org/robots/legway/>

[8] Dean Kamen ,2001,<http://www.segway.com>

[9] John Green,David Krakauer, March 2003, New iMEMS Angular Rate Sensing Gyroscope,<http://www.analog.com/library/analogDialogue/archives/37-03/gyro.html>

[10] PeterHemsley,32-bitsignedintegermathsforPICS, <http://www.piclist.com/techref/microchip/math/32bmath-ph.htm>

[11] Mosfets and Mosfet's drivers, <http://homepages.which.net/~paul.hills/SpeedControl/Mosfets.html>