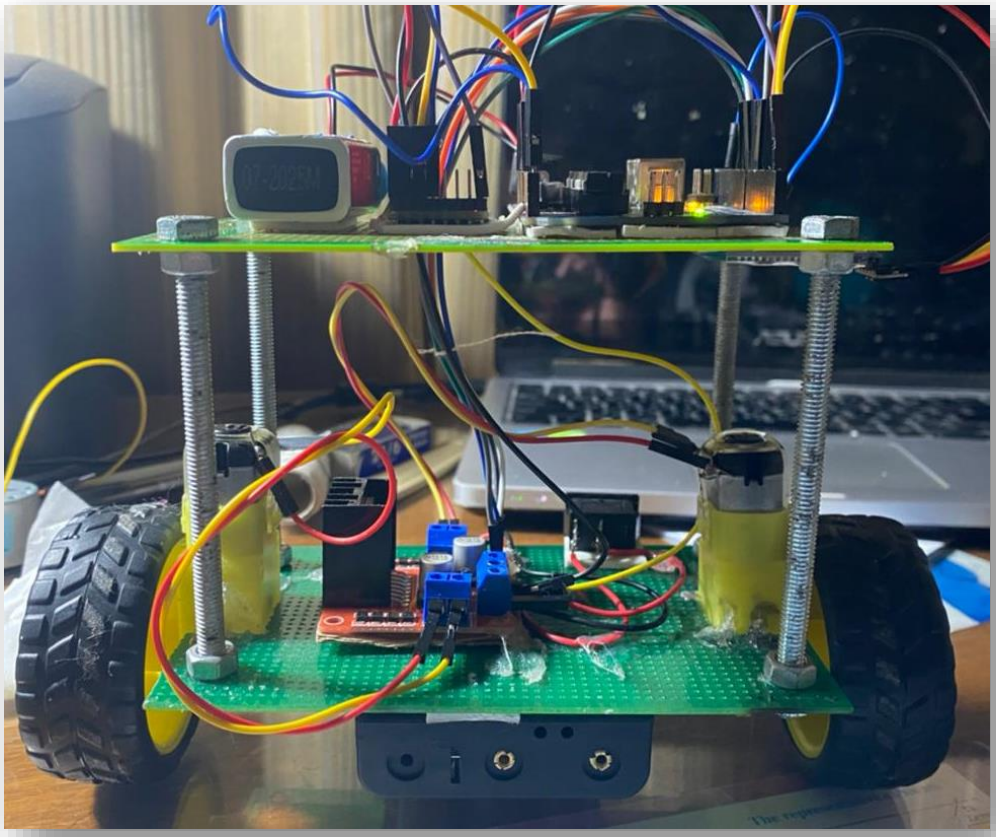
YR4-MECHANICAL ENGINEERING
SEMESTER 1 (21/22)

Self Balancing Robot

Applied Microcontroller Prog.MECH83H

Module leader: Dr. Mahmoud Magdy
Module TA: Eng. Omar El-mergawy

Student Name: Kareem Nabil Hamed
Student ID : 167042



Contents

Refrence vidoes :

https://drive.google.com/drive/folders/1gYx0iPUm8mC_k_86rmCqq3NL6ZC79C_e?usp=sharing

Table of figures

❖ Introduction

Design and technique in conducting selfbalancing robot is the same as balancing inverted pendulum. Inverted pendullam may be basic problem and concept in dynamics life. Inverted pendullam is used wordly to test control algorithms (PID variable, neural networks and genetic algorithms)( ). Rockets or Missile are a simple example for inverted pendullam. Where their center of gravity should always be located in the center. This phonemina can be explained using self balance robot.

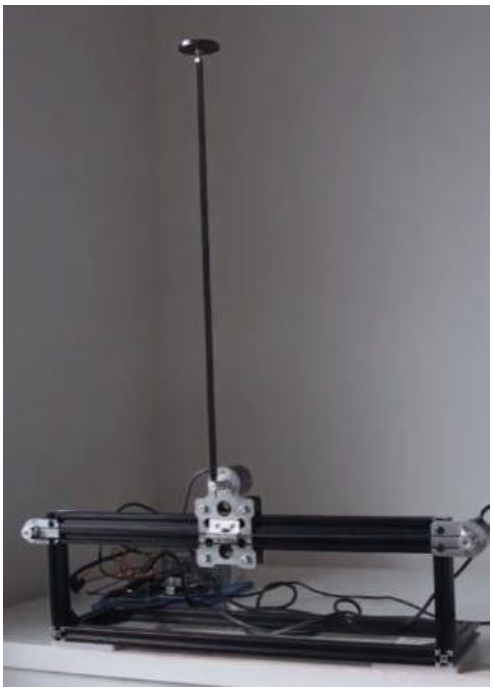


Figure 1 Ref 2Inverted Pendulam

❖ Background

Self balance robot is the solution for inverted pendulum problem(). The purse of the loops is to control the wheels position, such that the angle of inclination should always be stable at fixed angle which is the anglewhen the robot is vertical or fixed. In case of the robot is falling in any direction, the wheels should move faster in the same direction of falling to decrease the angle of inclination.

For example, if the robot is falling forward, the motors will rotates in the same direction of falling to make the robot vertical in no time. Same happens when mentioning the rate of deviation. If the robot is deflecting by small angle, the motors will rotate is small speed to reach equibrium position and vise versa. Handling the robot to not fall in one direction may also cause problem when holding it, it might start falling in the other direction due to the converted mass from the other direction. Here comes the function of PID controllers.

PID (proportional, Integral, derivative) respectively are controller variables used to control any system or applications to a certain function as (speed, room temperature or self balance robot). PID is a controlled loop to handle and to control process vaules. These values should be very accureate ans stable to atcheieve the desired output.

❖ P controller

This controller is proportional to the state value of the error. It case the error is big and positive, the p controller will control the output value to be large and positive. Using P controller only will result in errors that is set between actual values and setpoint. P controller needs error to establish proportional act. It case of zero error, there will be no corrective responed.

❖ I controller

I controller states for the prevoius value which intergartes these values over time to produce the I term. Such as, if the error in the P controller, the intergral interfernce to minimize this error by adding the I controller effect, which is collect from the prevoius loops of the errors. In case of the error vaniches, the I controller get to increase again.

❖ D controller

D controller is used to esimate the upcoming error that may accure. Depending on the current rate of deflection. D controller aims to minimize the error by understanding the rate of which error changes.

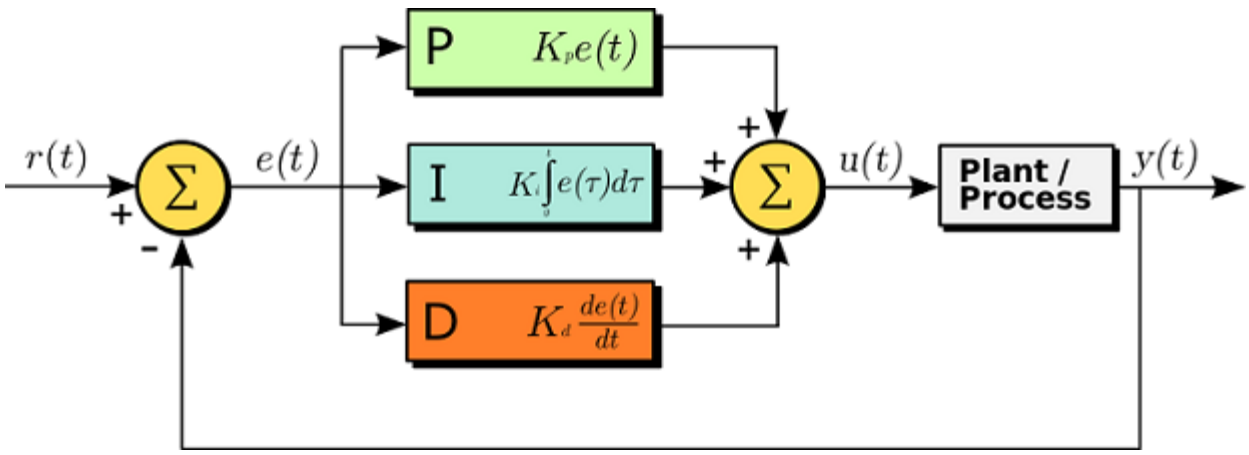


Figure 2 Ref3. PID controller

❖ Description of the project

Self balance robot is derived from inverted pendullam. This aime to stay stable, in possion which its mass of gravity applied in the center achieving balance. This same concept applied for the self blanance robot. The balancing of the robot is etbalished using MPU6050 which measure the accelermotor and the gyroscope. MPU give order or signal to the Arduino board which recieves this data. Accrourding to the data, the Micro controller which is in this case Arduino UNO, will send signals to the motors to move and reach vertical motion (balance position) . the MPU will keep recieving signal and sending signals to maintane all the time balancing through the controlled controllers PID.

This Robot consists of many electrical components.

1. Microcontroller (Arduino UNO)
2. Mpu 6050
3. H-bridge (motor driver).
4. 2 DC motors (12V)
5. Ultra sonic senser
6. Bluetooth module
7. 2 separites power supplies and charger.
8. Body or fram for the robot.
9.

1. Microcontroller (Arduino UNO)

The Arduino UNO is a microcontroller board based on the ATmega328( ). Arduino UNO has 14 digital input/output pins. 6 digital pins can be used as Pulse width modulation (PWM). These pins are (3, 5,6,9,10,11).
PWM pins are digital rectangular wave. These wave have constant frequency, but have variance in time of the signal, these variance can be changes from 0% to 100%.

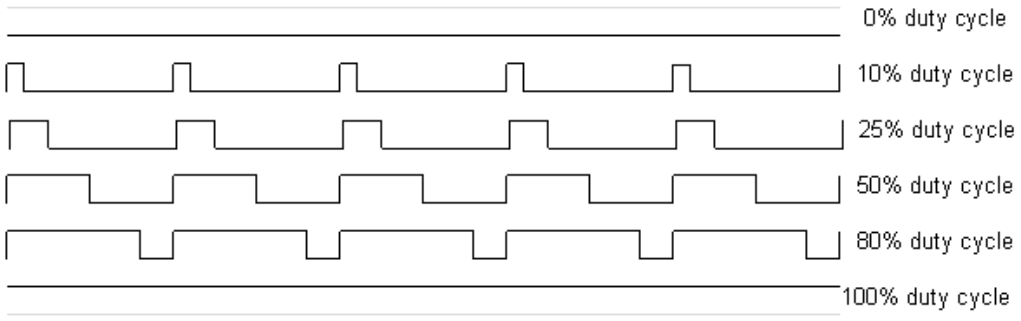


Figure 3 PWM Wave (Arduino.cc)

Arduino UNO contains also 6 analoge pins. Analoge pins works the same as digital pins. It is used to to have varaince of value between 0 to 1, which is infinte numbers. Analoge pins can take any values between 0-1023, also it can work on inputs value from 0 to 255.

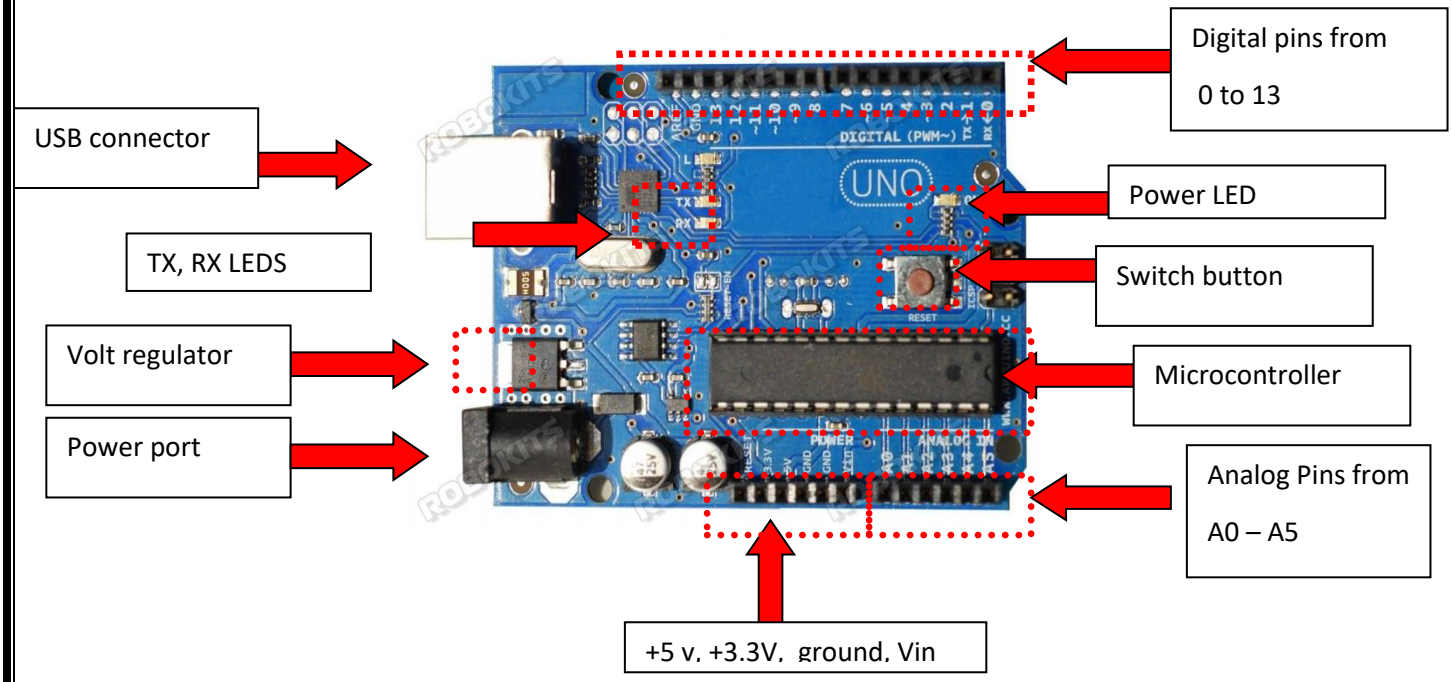


Figure 4 Ardiuno UNO

2. MPU6050

MPU 6050 is electrical component that measure 6 DOF (3-axis accelerometer& 3-aix gyroscope) and tempurature all these in one single chip, MPU contains Digital Motion Processor (DMP).

DMP software proccesses measurers 6-axis motion using algorithms. MPU can be connected to external sensor through ports (XDA, XCL). This connection in esbalished through I^2C bus interferance.

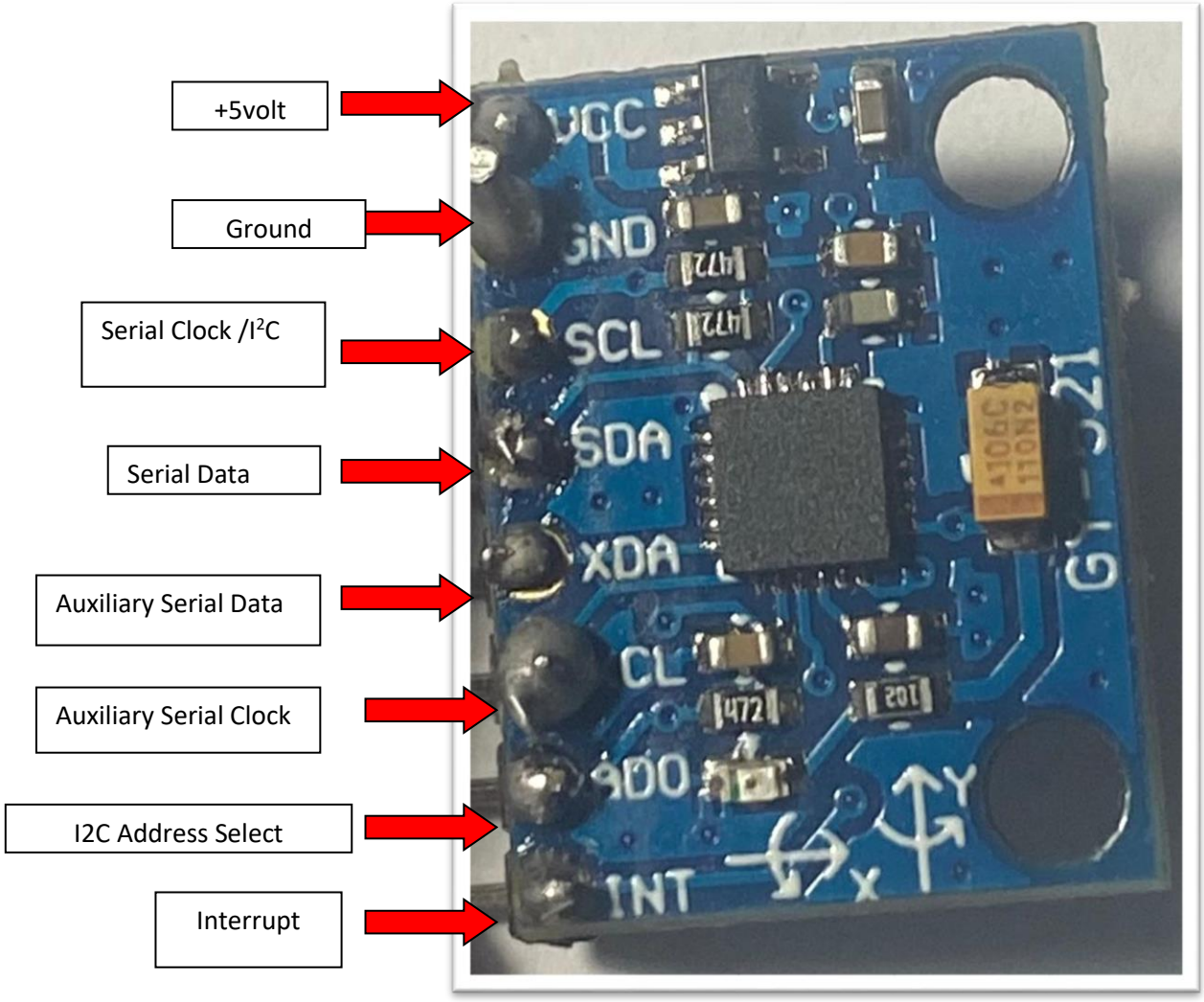


Figure 5 MPU6050 (6DOF)

3. H-bridge (Motor driver)

H-bridge is an electric componante that controlles the motor. It switches the polarity of volt wish is applied to the motor. DC motor have two terminals, which any one of them is positive and the other is negative. H-bridge switch the polarity of both terminals,which allows the motor even to move forward or backward without even changing the termianls of the motor.

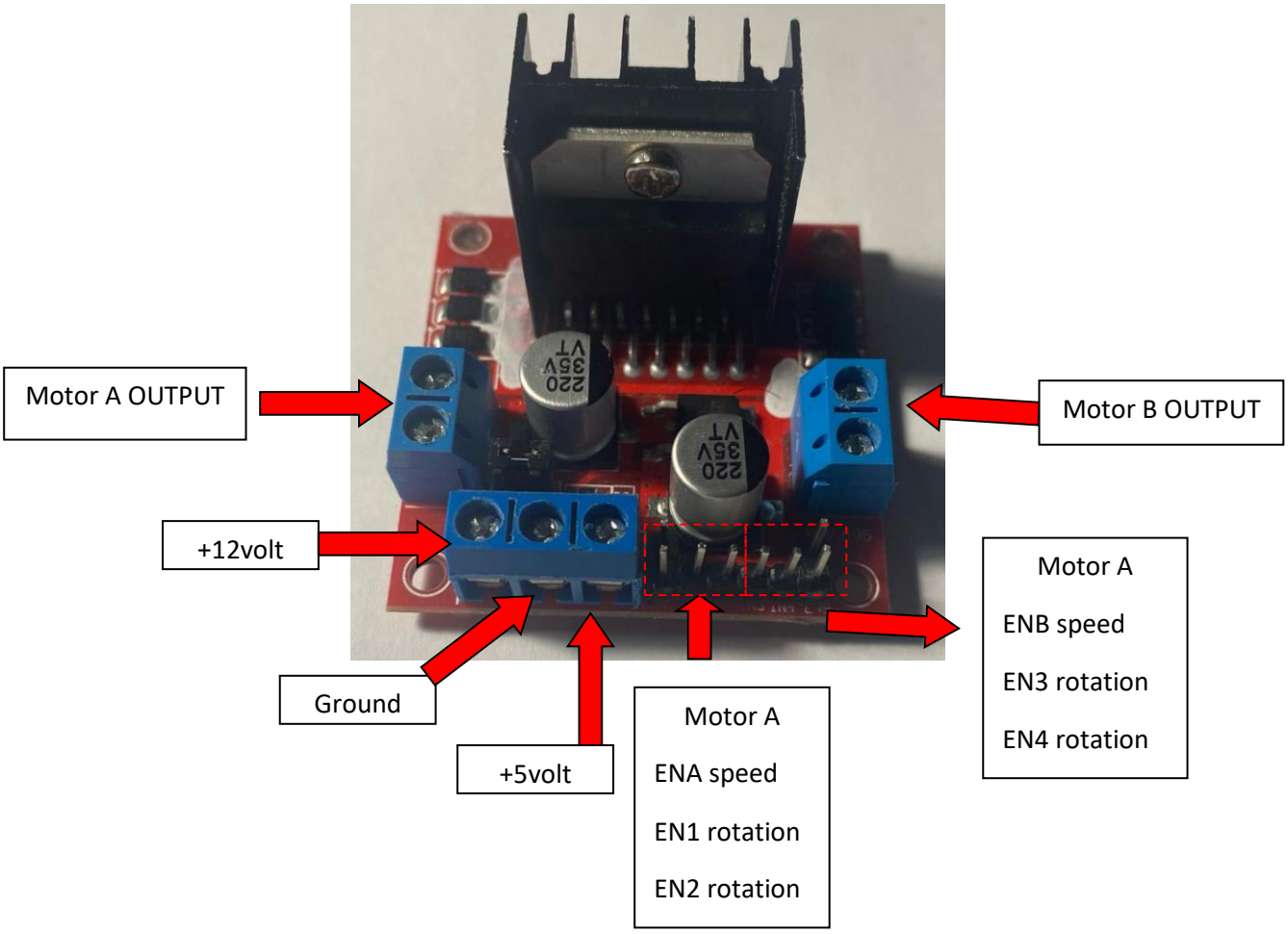


Figure 6 H-bridge

❖ **H-Bridge Connection**

Pin Name	**Connection**
IN1 & IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 & IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	PWM signal for Motor A
ENB	PWM signal for Motor B
OUT1 & OUT2	Output pins of Motor A
OUT3 & OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground

DC Motors (12 volt)

DC 12 volt motor are also known as Battery operated (BO) motors. BO motors are geared motor which have a good weight to torque ratio. This motor with maximum current can give torque up to 800 gf.cm.

Bo motors are capable to work with or without lubrication.

Figure 7ref 4- DC gear motor

5. Ultra sonic sensor

ultrasonic Sensor is an electric component used to measure distance using sound waves. Ultra sonic sensor have two transducers. The first one sends sound wave and called trigger, and the second one received the sound wave after hitting an object and called echo.

Ultrasonic sensor working mechanism is to send wave of frequency above human hearing range. This ultrasonic sensor have two transducers unlike other who have one transducer which do both sending and receiving. The sensor measure the distance by measuring the time taken between sending and receiving the pulse.

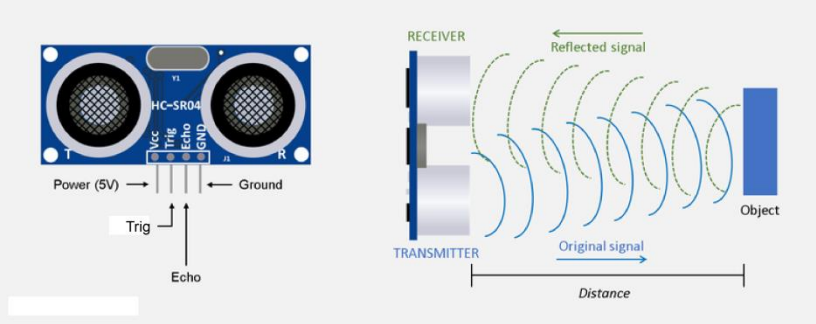


Figure 8 Ref 4- Ultrasonic sensor

- **Speed of sound at room tempurature is approximite 29.1us/cm.**

6. Bluetooth module HC-05

Bluetooth module is a device used to communicate devices together. It is widely used now days in every single application. HC-05 have a range of 100 meters connection depending on the atmosphere and geographic. The Bluetooth module can communicate with microcontroller using serial port.

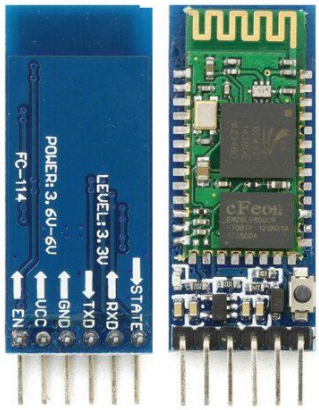


Figure 9 Ref 4- HC-05

7. Power Supply

This robot is power by two separate power supplies. 9-volt alkaline battery to power up the Arduino and 12-volt lithium ion battery to power the dc motors.

7.1 9-volt Alkaline

The 9-volt Alkaline battery produces 300 mAh which is enough to power up the Arduino. It is connected using male plug port.

Figure 10 ref 4 9Volt Alkaline battery

7.2 12 volt lithium ion rechargeable battery (18650)

The DC motor are powered with much stronger batteries. The Li-ion batteries comes with 3.7 volt and varies current mAh starting from (1500-3500).

Li-ion batteries are connected in series to produce 3.7 X 3 =12 volt.

Output ampere is 1500 mAh

Figure 11 Ref 4 - Lithiom ion rechargable battery

8 Frame of the robot

Self-balance robot does not focus on only electric components as such as mechanical design. Robot should be light as possible, so that the robot could keep balancing easily. The frame was built from PCB board, which is very rigid and very light. The board dimensions are 14 X 9 cm. robots levels are connected together using 10 cm threaded nail and bolts.

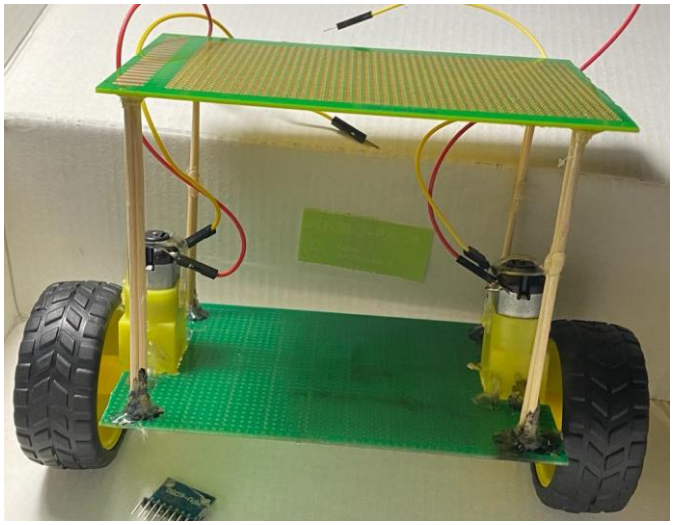
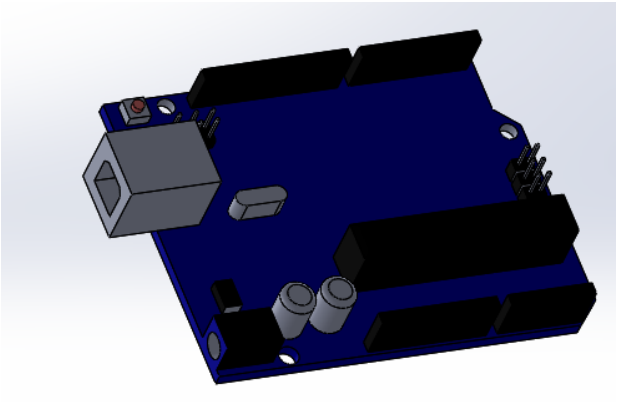


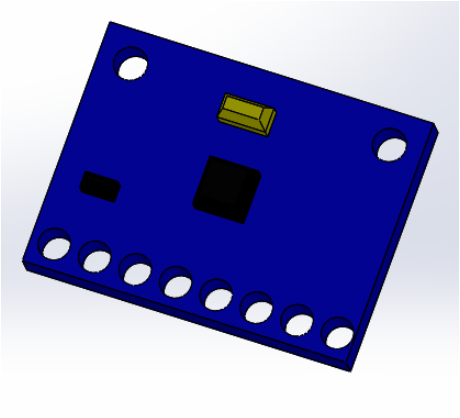
Figure 12 robot frame

- Mechanical drawing

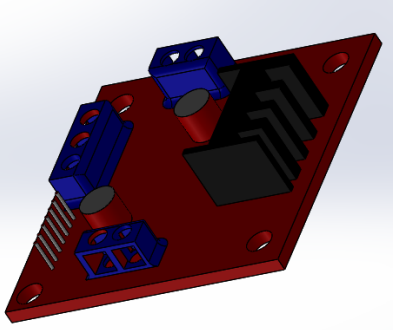
1 Arduino Uno



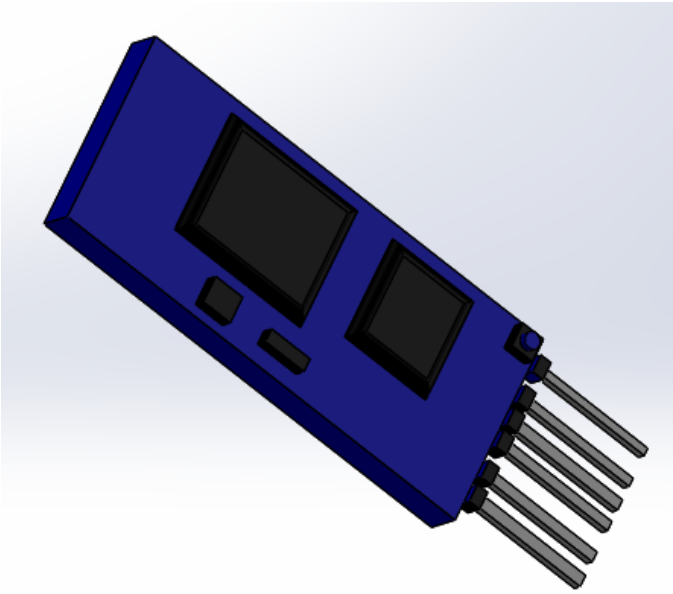
2 MPU 6050



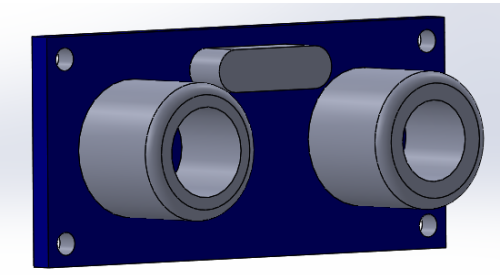
3. H-Bridge



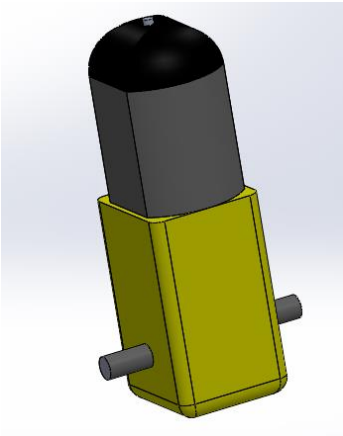
4. Bluetooth Module HC-05



5. Ultrasonic Sensor

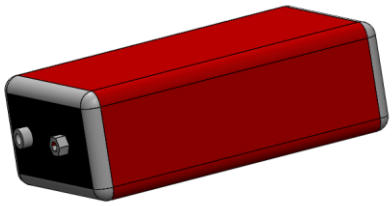


6. DC gear Motor

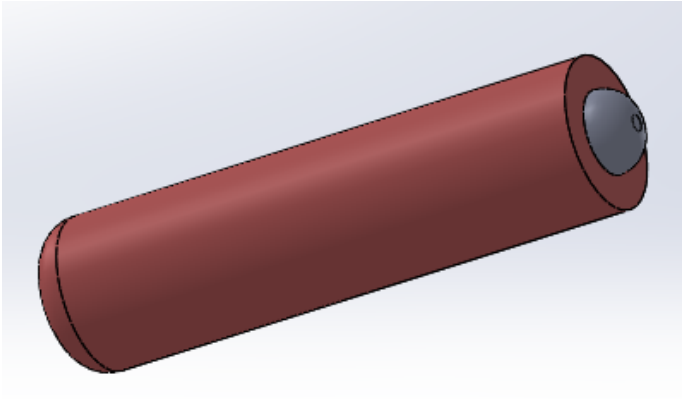


7. Power source

7.1 9 volt

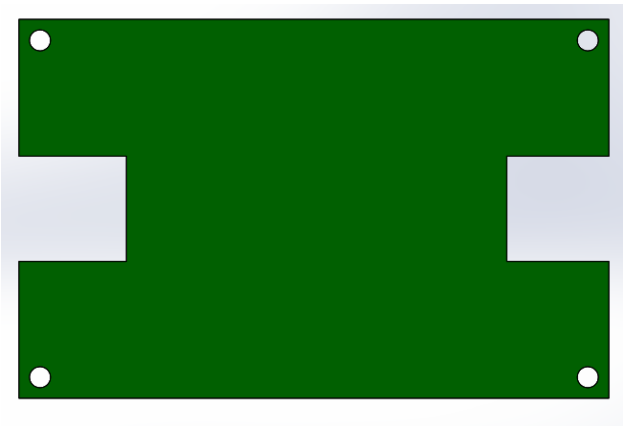


7.2 Lithium ion battery

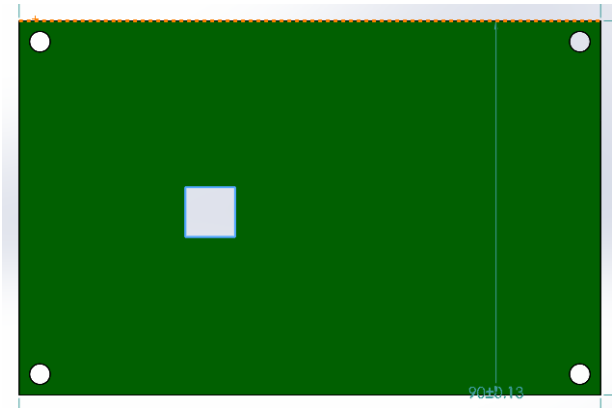


8. Frame of the robot

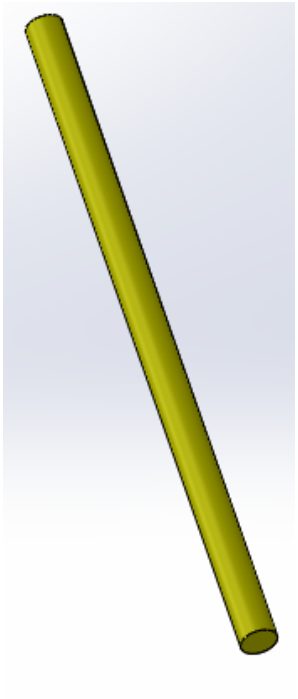
8.1 Lower Frame



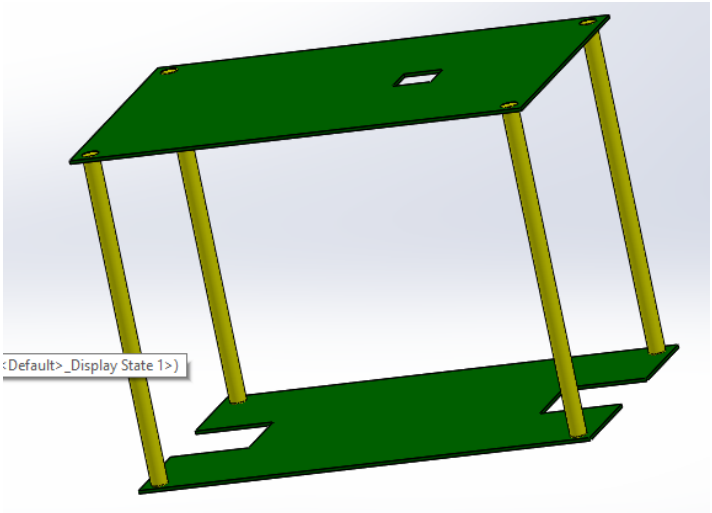
8.2 Upper frame



8.3 Connection

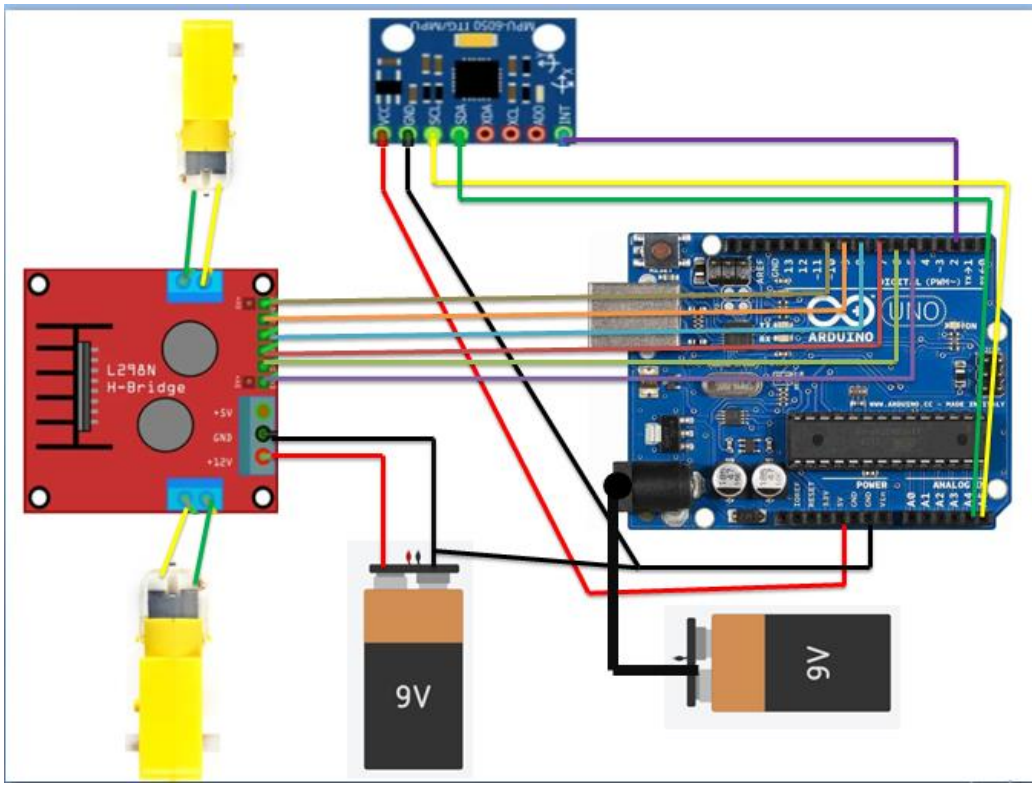


8.4 Final Frame assembly

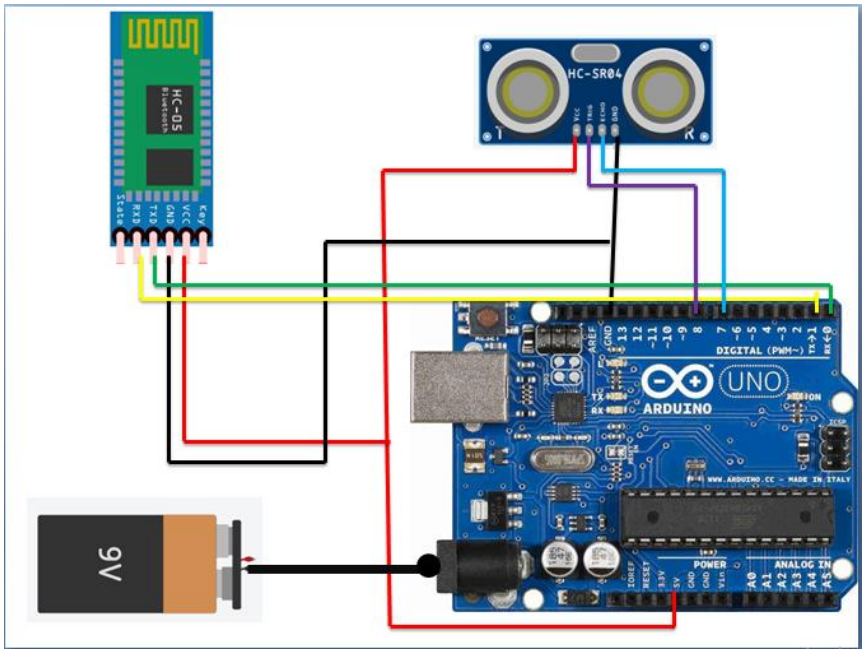


❖ Electrical Drawing for Self balancing robot

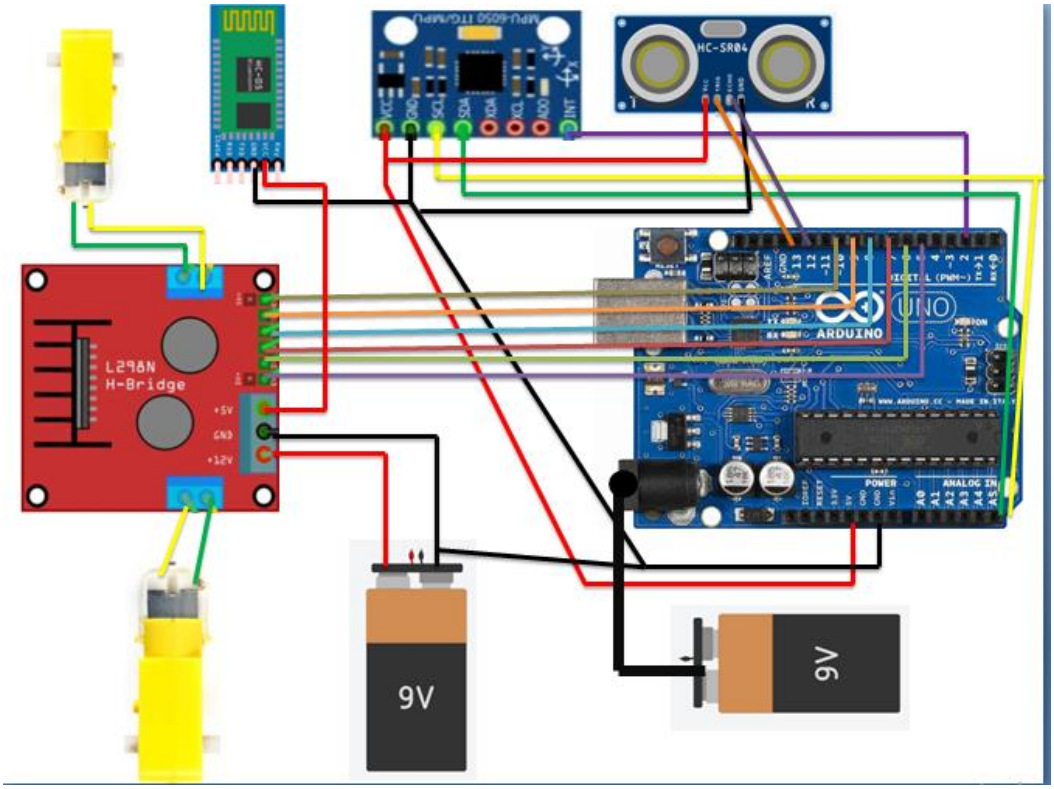
1. Self balance circuit



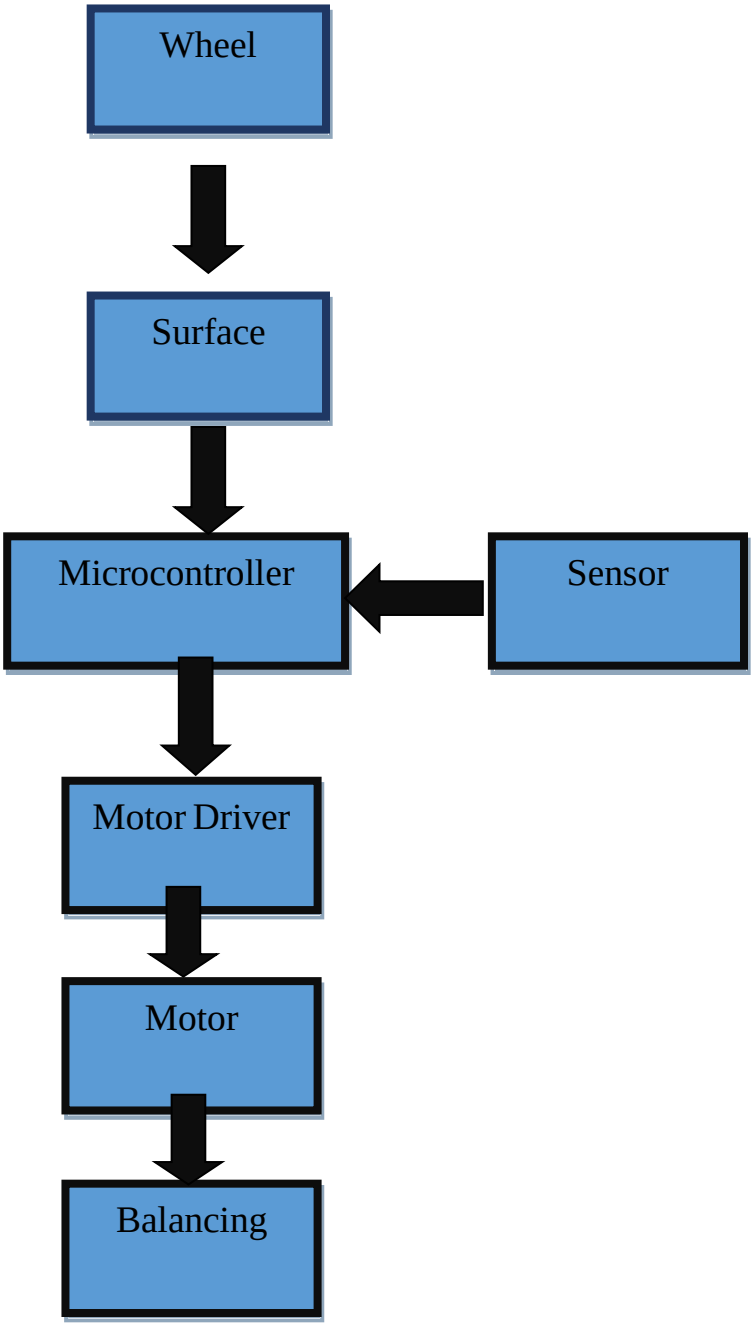
2. Ultrasonic Sensor using Bluetooth Module



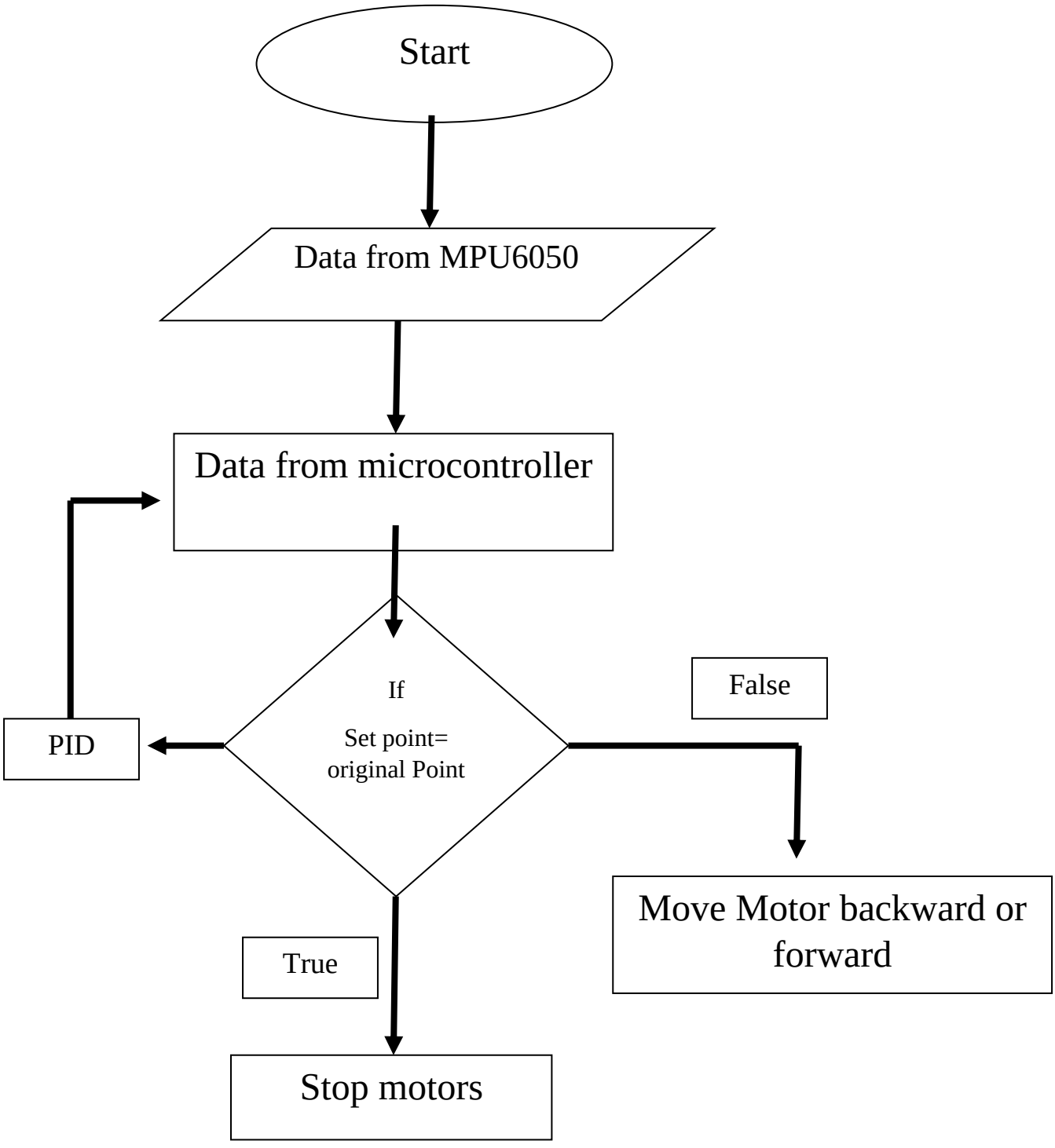
3. Merge Self-balancing robot



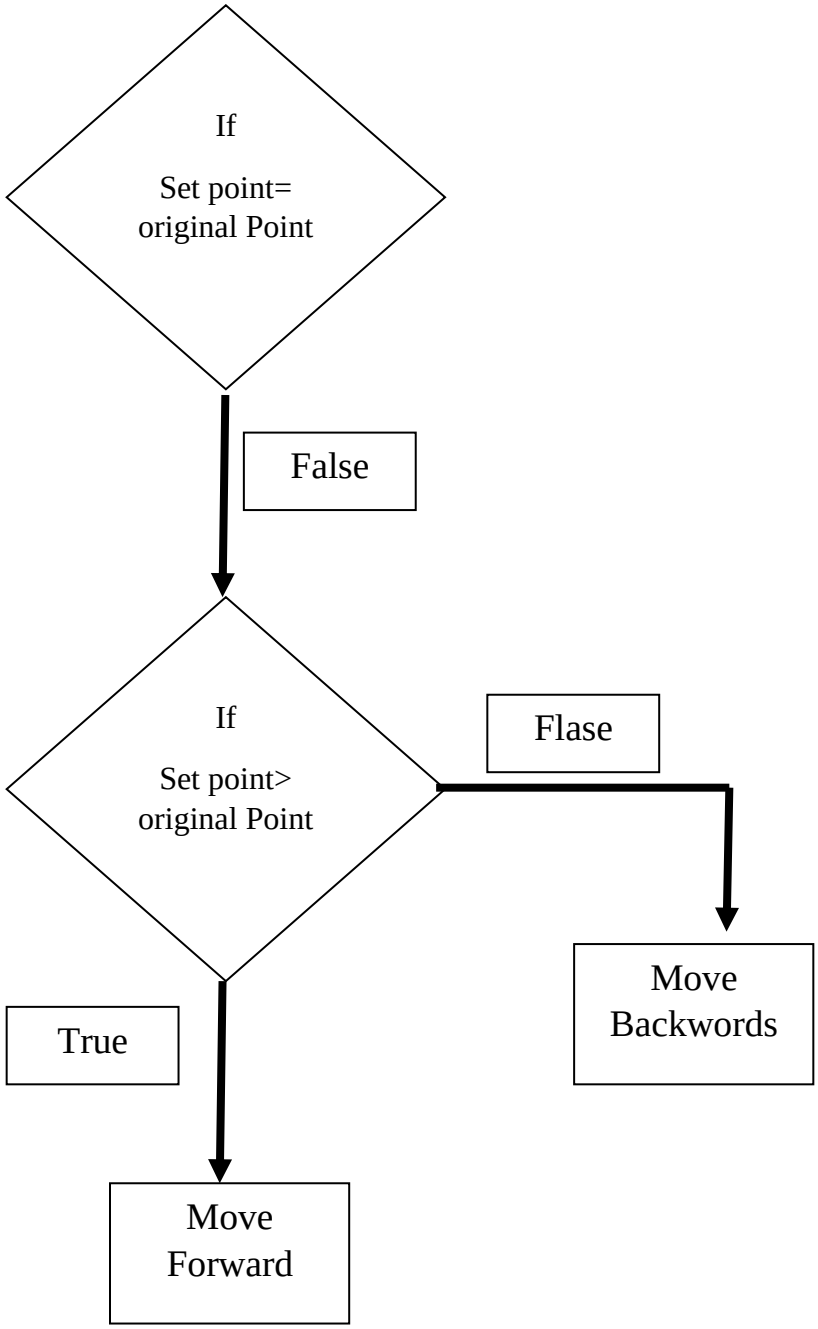
5. Block Diagram



5.1 Flow chart diagram



5.3 Direction Flow chart



❖ Structure analysis

Component	Weight (grams)
Arduino battery	30
Arduino UNO	25
Upper Board	25
MPU 6050	20
Nails and bolts	50X4=200
Total Weight of upper level	**300 gram**
H-Bridge	35
Motor Batteries	(35X3)+10 = 115
Lower Board	28
Switch Button	10
Bluetooth Module	20
Wires	60
Total Weight of lower Level	**268gram**
Total weight	**568 gram**

❖ Assumptions

i. Neglect weight of motors and wheels
ii. Weight is equal distributed
iii. Weight acts at the Mid-point of the Robot. at 7.5 cm from both motors

Maximum Torque Applied by each Motor is 800gm.cm

Maximum Torque Applied by both Motor is 1600gm.cm

Assume weight Unknown

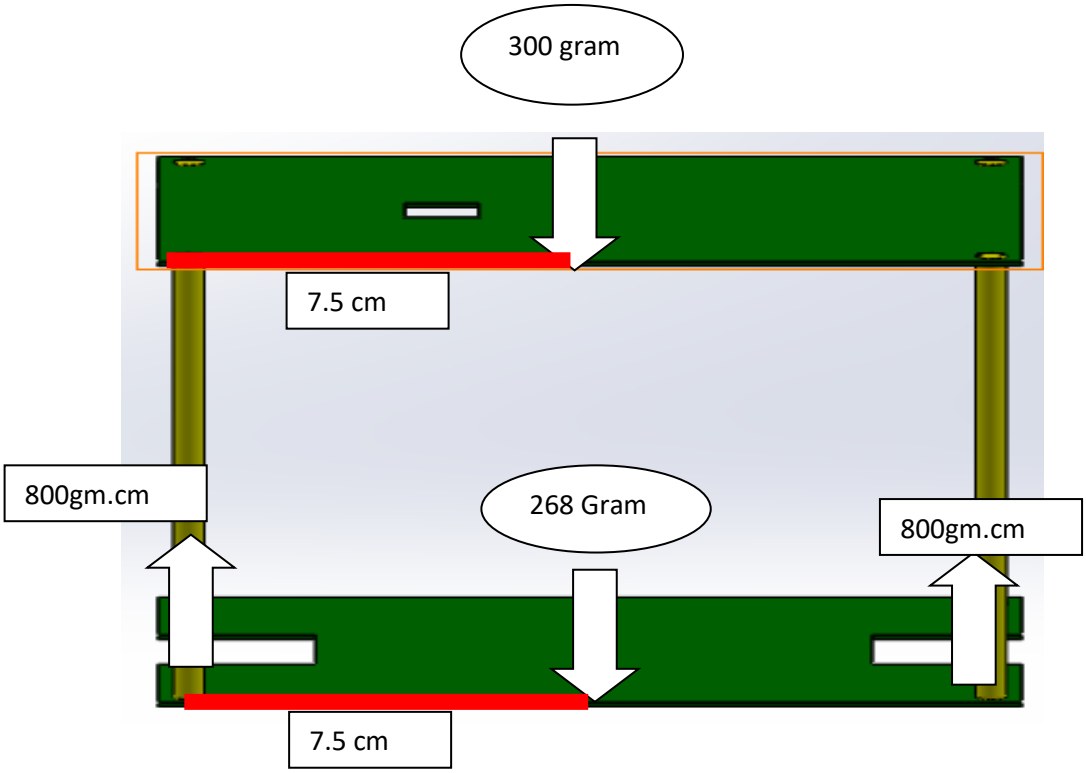
$$W_{th} \text{ X } 7.5 <= (800X2) \text{ X } 7.5$$

$$W_{th} <= \frac{(800x2)\ X7.5}{7.5} = 1600 \text{ gram}$$

Since Actual, weight is less than the theoretical weight,

Motors can overcome the weight force and achieve balance

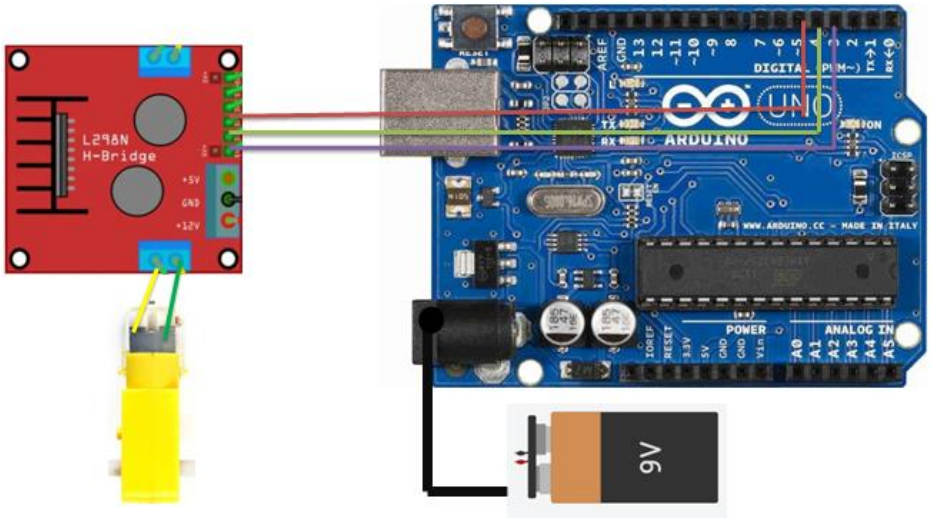
❖ Free Body Diagram



❖ Simulations Results

1- H-bridge controlling motor direction and speed

1.1. Circuit



1.2. Code

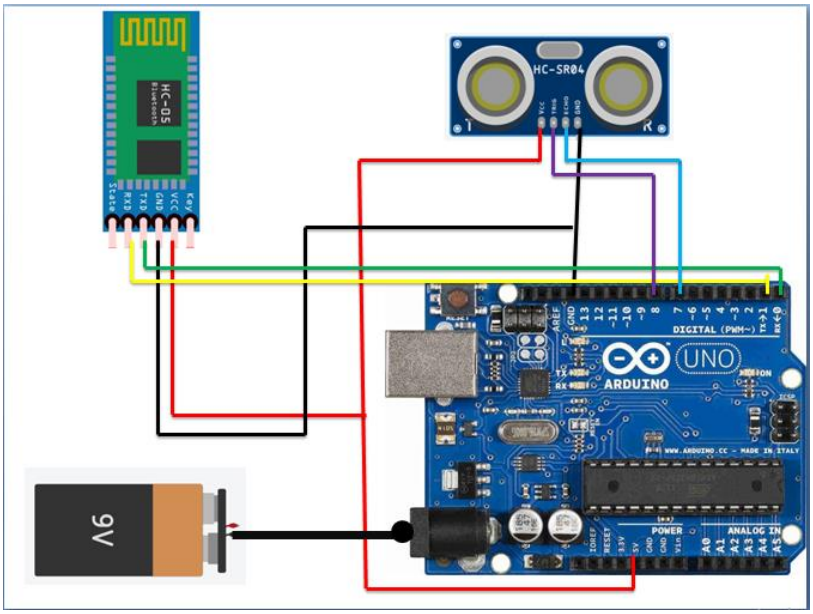
Driver_Motor_L298N §

```
int enA = 3;
int in_1 = 4;
int in_2 = 5;

void setup() {

  pinMode(enA,OUTPUT);
  pinMode(in_1,OUTPUT);
  pinMode(in_2,OUTPUT);

}

void loop() {
  Kareem_1();  //first loop to do something
  delay(2000);
  Kareem_2();   //Second loop to do something
  delay(2000);
  Kareem_3();  //Third loop to do something
  delay(2000);
  Kareem_4();    //Forth loop to do something
  delay(2000);
  }

void Kareem_1(){
  analogWrite(enA,20);  /// speed equal 20 , in direction 1, in_1 HIGH
  digitalWrite(in_1,HIGH);
  digitalWrite(in_2,LOW);
}
```

```
void Kareem_1(){
  analogWrite(enA,20);  /// speed equal 20 , in direction 1, in_1 HIGH
  digitalWrite(in_1,HIGH);
  digitalWrite(in_2,LOW);
}

void Kareem_2(){
  analogWrite(enA,30);   /// speed equal 30 , in direction 1, in_1 HIGH
  digitalWrite(in_1,HIGH);
  digitalWrite(in_2,LOW);
}

void Kareem_3(){
  analogWrite(enA,20);  /// speed equal 20 , in direction 2, in_2 HIGH
  digitalWrite(in_1,LOW);
  digitalWrite(in_2,HIGH);
}

void Kareem_4(){     ////speed equal 30, in direction 2, in_2 HIGH
  analogWrite(enA,30);
  digitalWrite(in_1,LOW);
  digitalWrite(in_2,HIGH);
}
```

2. Ultrasonic and Bluetooth Module to send distance on mobile

2.1 Circuit



2.2 Code

```
int triggerPin = 4; //triggering on pin 4
int echoPin = 13;    //echo on pin 13
int LED = 4; //led pin
int info = 0;//variable for the information comming from the bluetooth module
int state = 0;//simple variable for displaying the state

void setup() {

  Serial.begin(9600);  // we can see the distance on the serial monitor first

  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);

}

void loop(){ //here we combine both codes to work in the loop
  bluetooth();  // separet loops down
  sensor();
}


void bluetooth() {         //loop for the bluetooth code named void bluetooth
  if(Serial.available() > 0){  //if there is any information comming from the serial lines...
    info = Serial.read();
    state = 0;   // store it into the "info" variable
  }
```

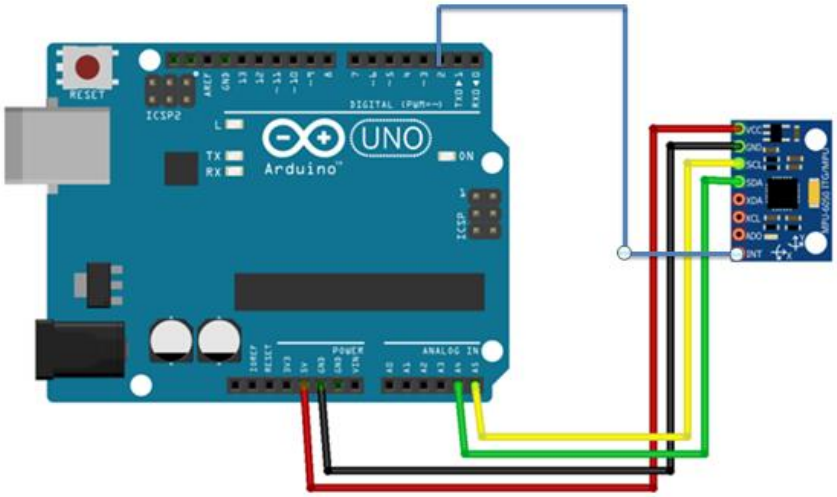
```
void bluetooth() {          //loop for the bluetooth code named void bluetooth
  if(Serial.available() > 0){  //if there is any information comming from the serial lines...
    info = Serial.read();
    state = 0;   // store it into the "info" variable
  }
  if(info == '1'){               //if it gets the number 1(stored in the info variable...
    digitalWrite(LED, HIGH);    //it's gonna turn the led on(the on board one)
    if(state == 0){
      Serial.println("good to go");  //^^that will prevent the arduino from sending all the time, only when you change the state
      state = 1;
    }
   }
  else if(info == '0'){
    digitalWrite(LED, LOW);      //else, it's going to turn it off
    if(state == 0){
      Serial.println("LED OFF");//display that the LED is off
      state = 1;
     }
  }
}

void sensor() { //loop for the sensor code is named void sensor

  int duration, distance;    //constants duration and distance

  digitalWrite(triggerPin, HIGH); //triggering the wave
  delay(10);
  digitalWrite(triggerPin, LOW);
```

distance_mes_code§

```
      Serial.println("LED OFF");//display that the LED is off
      state = 1;
     }
   }
  }

  void sensor() { //loop for the sensor code is named void sensor

    int duration, distance;    //constants duration and distance

    digitalWrite(triggerPin, HIGH); //triggering the wave
    delay(10);
    digitalWrite(triggerPin, LOW);

    duration = pulseIn(echoPin, HIGH); // function for listening and waiting for the wave
    distance = (duration/2) / 29.1; //transforming the number to cm 29.1 speed of sound in room temp mircosecond/cm

    Serial.print(distance);    //printing the distance
    Serial.print("cm");       //print unit
    Serial.println(" ");      //printing to a new line

    //adding for mesuring distance where the led will turn off, even if we tell it to turn off when we chose so in the app

    if(distance <= 15){  //if we get too close, the LED will turn on
      digitalWrite(LED, HIGH);
      Serial.println("TOO CLOSE!!!");

    }
  }
```

3 MPU 6050 Offset calibration

3.1 Circuit

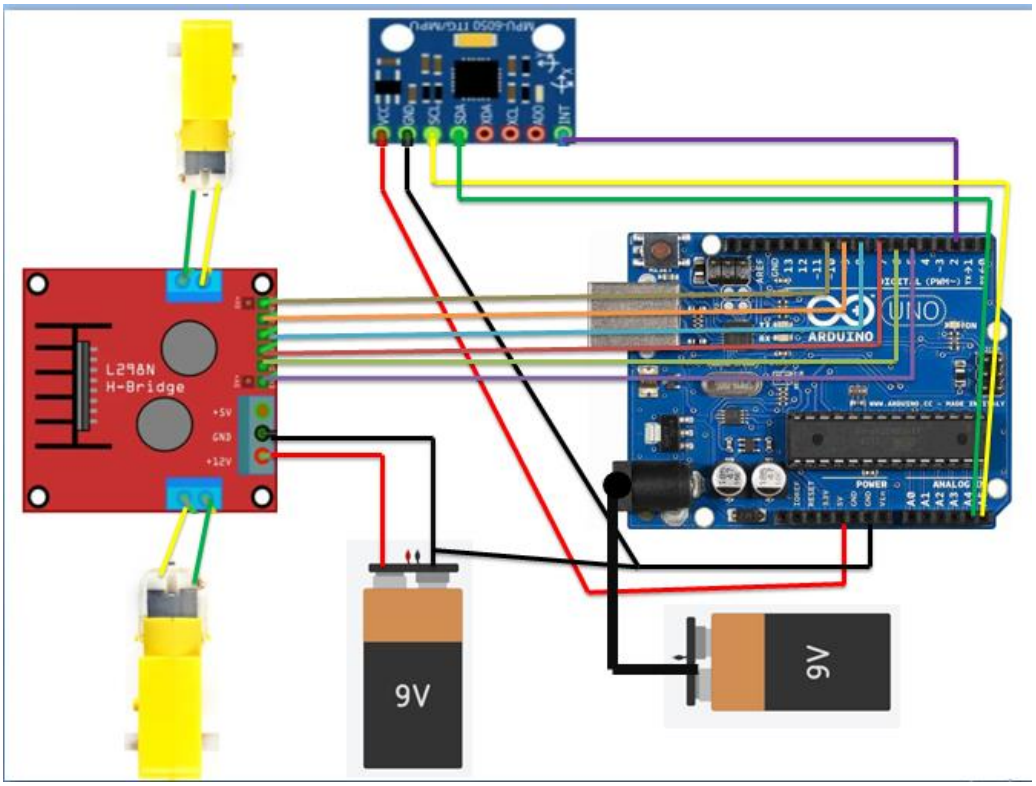


3.2 Code

```
#include "Wire.h"  ///include wire for I2c Address
#include <MPU6050_light.h>   //include mpu
MPU6050 mpu(Wire);   // connect manualy
unsigned long timer = 0;   //set int timer zero
void setup() {

  Serial.begin(9600);   // send data
  Wire.begin(); // connect
byte status = mpu.begin();  //set status to begin
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);   // print actual state
  while (status != 0) { } // wait if could not connect to MPU6050
Serial.println(F("Calculating, do not move MPU6050"));
  delay(1000);
  mpu.calcOffsets(); // gyro and accelero
  Serial.println("Done");
}
void loop() {
  mpu.update();
if ((millis() - timer) > 10) { // print every 10ms
    Serial.print("X : ");
    Serial.print(mpu.getAngleX());  //print yaw
    Serial.print("\tY : ");
    Serial.print(mpu.getAngleY());  //print pitch
    Serial.print("\tZ : ");
    Serial.println(mpu.getAngleZ());  //print roll
    timer = millis();
  }
```

4. Self balance robot

4.1 Circuit Diagram



4.2 Code

```
#include <PID_v1.h>   // libray for pid controller values
#include <LMotorController.h>  // library for controlling the motors(h-bridge)
#include "I2Cdev.h"    // libray for transforming data
#include "MPU6050_6Axis_MotionApps20.h"   // library for mpu

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE   //connect manually I2C with wire
#include "Wire.h"  // library for comunicating with mpu6050 I2C // Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implem
#endif

#define MIN_ABS_SPEED 20  //constant speed for moving

MPU6050 mpu;  // name of my mpu

// MPU control functions
bool dmpReady = false; // set true if DMP init was successful//  Digital motion Processor  //used incase of true or false , each carry 1
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU //Interrupt functionality is configured via the Interrupt Configura
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error) // unsigned int 8 bits
uint16_t packetSize; // expected DMP packet size (default is 42 bytes for ardiuno uno)
uint16_t fifoCount; // count of all bytes currently in FIFO, FIFO IS known as first in first out , read data in sequence (standard)
uint8_t fifoBuffer[64]; // FIFO storage or size of buffer bytes

// orientation/motion vars (axis)  // will be used to calc. offset in equation down
Quaternion q; // [w, x, y, z] quaternion container  //used to calc ypr
VectorFloat gravity; // [x, y, z] gravity vector  // used to calc ypr
```

```
VectorFloat gravity; // [x, y, z] gravity vector  // used to calc ypr
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector

//PID
double originalSetpoint=173;  // constant, when the robot is perfectly Vertical
double setpoint = originalSetpoint; // return to original point in case of falling
double movingAngleOffset = 0.06; //limitation for angle deviation
double input, output;

//adjust pid values based on the design
double Kp = 15;   // value get them by using trial and error
double Kd = 1.8; //value get them by using trial and error
double Ki = 50;   // value get them by using trial and error
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);  //If my Input is above the Setpoint, the output should be inc

double motorSpeedFactorLeft = 0.6;   // left motor factor  from 0 to 1
double motorSpeedFactorRight =0.6;  // right motor factor from 0 to 1

//H-bridge set up pin
int ENA = 5;  //motor 1 speed
int IN1 = 6; // motor 1 direction
int IN2 = 7; // motor 1 direction
int IN3 = 8; // motor 2 direction
int IN4 = 9; // motor 2 direction
int ENB = 10; // motor 2 speed
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, motorSpeedFactorRight);

volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high , ///its value can be changed by something
void dmpDataReady()  //  void for data mangment plat form for handlling mpu data
{
mpuInterrupt = true;      //set true, which can not be false for ever
}

////////////////////////// Ultrasonic//////////////////////////////
//int triggerPin = 4; //triggering on pin 7
//int echoPin = 13;    //echo on pin 8
//int LED = 0; //led pin
//int info = 0;//variable for the information comming from the bluetooth module
//int state = 0;//simple variable for displaying the state


void setup()
{
// join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
Wire.begin();              //set I2c as master , and mpu will be the slave
```

```
Serial.begin(9600);          //to print offset on mobile and moniter
TWBR = 24; // 400kHz for fast mode-I2C clock, 16/ 16+(2.24).1 = 0.4 MHz,  Data sheet , prescaler =1
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
Fastwire::setup(400, true);  // 400 KHZ for fast mode
#endif

mpu.initialize();

devStatus = mpu.dmpInitialize();       //devstatus = unsigned int 8bits

// supply your own gyro offsets here, using mpu tutorial
mpu.setXGyroOffset(190);
mpu.setYGyroOffset(55);
mpu.setZGyroOffset(-65);
mpu.setZAccelOffset(1788);


if (devStatus == 0)  // make sure it worked (returns 0 if so)
{

mpu.setDMPEnabled(true);  // turn on the DMP, now that it's ready

// enable Arduino interrupt detection
attachInterrupt(0, dmpDataReady, RISING);    // Rising, trigger when the pin goes from low to high



mpuIntStatus = mpu.getIntStatus();  // define both

// set our DMP Ready flag so the main loop() function knows it's okay to use it
dmpReady = true;     //switch from false to true

// get expected DMP packet size for later comparison
packetSize = mpu.dmpGetFIFOPacketSize();  // force to get dmpfifo , number of bytes in the FIFO

//setup PID
pid.SetMode(AUTOMATIC);   // Make PID act automatic
pid.SetSampleTime(10);
pid.SetOutputLimits(-255, 255);  // pid control for the motors limitation
}
else
{

Serial.print(F("DMP Initialization failed (code "));
Serial.print(devStatus);
Serial.println(F(")"));
}
////////////////////////////////ultrasonic////////////////////////////////////
  //we'll start serial comunication, so we can see the distance on the serial monitor
```

```
 // pinMode(triggerPin, OUTPUT); //defining pins
 // pinMode(echoPin, INPUT);
 // pinMode(LED, OUTPUT);    //defining LED pin
  //digitalWrite(LED, LOW);  //once the programm starts, turn off the led.
}


void loop()
{bluetooth();
Serial.println(input);   // print set point on Serial monitor

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

// wait for MPU interrupt or extra packet(s) available
while (!mpuInterrupt && fifoCount < packetSize) // if the robot not falling wait for interupt
{
//no mpu data - performing PID calculations and output to motors
pid.Compute(); // return to PID controllers , then move
motorController.move(output, MIN_ABS_SPEED);  //  move the motors is case of falling with consistant speed.

}

// reset interrupt flag and get INT_STATUS byte



mpuInterrupt = false;  // turn false agian,  , then true, on and on
mpuIntStatus = mpu.getIntStatus(); // define both


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

// check for overflow (this should never happen unless our code is too inefficient)
if ((mpuIntStatus & 0x10) || fifoCount == 1024) //The MPU-60X0 contains a 1024-byte FIFO re
{
// reset so we can continue cleanly
mpu.resetFIFO();


// 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
```

```
// 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);
input = ypr[1] * 180/M_PI + 180;   /// input offset value after calculated

}
////////////////////////////////////////////////Bluetooth ////////////////////////////////

}
void bluetooth() { //loop from the bluetooth code is renamed to "bluetooth" void
  if(Serial.available() > 0){  //if there is any information comming from the serial lines...
int    info = Serial.read();
int    state = 0;   //...than store it into the "info" variabl
  }
}

//void sensor() { //loop from the sensor code is renamed to the "sensor" void

//  int duration, distance;    //constants duration and distance
```

```
int    state = 0;   //...than store it into the "info" variabl
  }
}

//void sensor() { //loop from the sensor code is renamed to the "sensor" void

//  int duration, distance;    //constants duration and distance

  //digitalWrite(triggerPin, HIGH); //triggering the wave
  //delayMicroseconds(10);
 // digitalWrite(triggerPin, LOW);
 // delayMicroseconds(10);

 // duration = pulseIn(echoPin, HIGH); // function for sending and waiting for the sound wave
 // distance = (duration/2) / 29.1; //transforming to cm  &&&&&  29.1 is the speed of sound in room temp.

 // Serial.print(distance);    //printing the distance
 // Serial.print("cm");       //print unit
 // Serial.println(" ");
```

❖ Datasheet for self-balance

Mpu6050

6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	I²C FAST-MODE					
f_{SCL}, SCL Clock Frequency				400	kHz	

Figure 13 Ref1. I2C fast mode frequency 400KHZ

7.11 Auxiliary I²C Serial Interface

The MPU-60X0 has an auxiliary I²C bus for communicating to an off-chip 3-Axis digital output magnetometer or other sensors. This bus has two operating modes:

- I²C Master Mode: The MPU-60X0 acts as a master to any external sensors connected to the auxiliary I²C bus
- Pass-Through Mode: The MPU-60X0 directly connects the primary and auxiliary I²C buses together, allowing the system processor to directly communicate with any external sensors.

Figure 14 Ref 1. I2C modes

7.17 FIFO

The MPU-60X0 contains a 1024-byte FIFO register that is accessible via the Serial Interface. The FIFO configuration register determines which data is written into the FIFO. Possible choices include gyro data, accelerometer data, temperature readings, auxiliary sensor readings, and FSYNC input. A FIFO counter keeps track of how many bytes of valid data are contained in the FIFO. The FIFO register supports burst reads. The interrupt function may be used to determine when new data is available.

For further information regarding the FIFO, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.

7.18 Interrupts

Interrupt functionality is configured via the Interrupt Configuration register. Items that are configurable include the INT pin configuration, the interrupt latching and clearing method, and triggers for the interrupt. Items that can trigger an interrupt are (1) Clock generator locked to new reference oscillator (used when switching clock

Figure 15 Ref1. FIFO maximum bytes 1024

Table of Interrupt Sources

Interrupt Name	Module
FIFO Overflow	FIFO
Data Ready	Sensor Registers
I^2C Master errors: Lost Arbitration, NACKs	I^2C Master
I^2C Slave 4	I^2C Master

Figure 16 Ref1. Interupt states of mpu6050

Reference

1- InvenSense Inc. (n.d.). *MPU-6050 Product Specification*. TDK | Attracting Tomorrow.

https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

(n.d.). TDK |

2- Franklin et al. [1998], Franklin et al. [1991], Ogata [2002], Golnaraghi and Kuo [2010], Goodwin et al. [2000]
And Astrom and Murray [2008].

4- https://ram-e-shop.com/

5- https://store.fut-electronics.com/