SELF BALANCING ROBOT

A Project Report

Submitted by

Mr. GOTHE SHAHID SHAMS
Mr. FOUJE SALMAN AMIR
Mr. ANTULEY ABDUL MAJJID
Mr. SHAIKH MOHD JUNAID

in partial fulfillment for the award of the degree

of

B.E

IN

ELECTRONICS & TELECOMMUNICATION

At

ANJUMAN-I-ISLAM'S

KALSEKAR TECHNICAL CAMPUS

PANVEL

2014-2015

CERTIFICATE

This is to certify that the project entitled **"SELF BALANCING ROBOT"** is the bonafide work carried out by students of B.E., KALSEKAR Technical Campus, Panvel, during the year 2014-2015, in complete fulfillment of the requirements for the award of the Degree of B.E EXTC and that the project has not formed the basis for the award previously of any degree, diploma, associateship, fellowship or any other similar title.

(Prof.Bandanawaz kotiyal)
Internal guide

(Prof.Mujib tamboli)
HOD

(External)

Dissertation Approval for Bachelor Of Engineering

This Dissertation entitled "**Self Balancing Robot"**by **Gothe shahid ,Fouje salman,Antuley A.majjid,Shaikh M.junaid.** are approved for the degree of ***Bachelor of Engineering in Electronics and Telecommunication Engineering***.

Examiners

1. -----------------------------------

2. -----------------------------------

Supervisors /Guide's

1. --------------------------

2. ----------------------------------

-

Head of Department

Principal

Date : ---------------------------

Place : New Panvel

DECLARATION

We hereby declare that the project entitled **“SELF BALANCING ROBOT”** submitted for the B.E. Degree is our original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Signature of the Students:

Mr. GOTHE SHAHID SHAMS

Mr. FOUJE SALMAN AMIR

Mr. ANTULEY ABDUL MAJJID

Mr. SHAIKH MOHD JUNAID

Place: New Panvel
Date:

ACKNOWLEDGEMENT

We take this opportunity to express our deep sense of gratitude to our guide **Prof. Bandanawaz Kotiyal** for his valuable guidance and inspiration in spite of his busy schedule. He devoted himself in completing our task with the admirable excellence. He has taken keep and personal interest in giving us constant encouragement and timely suggestions also to our HOD **Prof. Mujib Tamboli** for cheerful encouragement and notable guidance.

Our special thanks to our electronics & telecommunication staff, who gave precious guidelines for our paper "**SELF BALANCING ROBOT**" and supporting staff members of electronics & telecommunication department for their valuable help in our paper.

We also express our heart full thanks to our beloved Director **Dr. Abdul Razzak Honnutagi** has provided the facilities for paper, we are also thankful to our friends**.**

ABSTRACT

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.

LIST OF FIGURES

TABLE OF CONTENT

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 .This is because as soon as the robot accelerates in a particular direction we can find that the torque due to the pseudo force in the non-inertial frame of accelerating robot opposes the torque due to gravity and if it is high enough (obviously depends on acceleration) it can bring the robot platform back to the horizontal level.

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

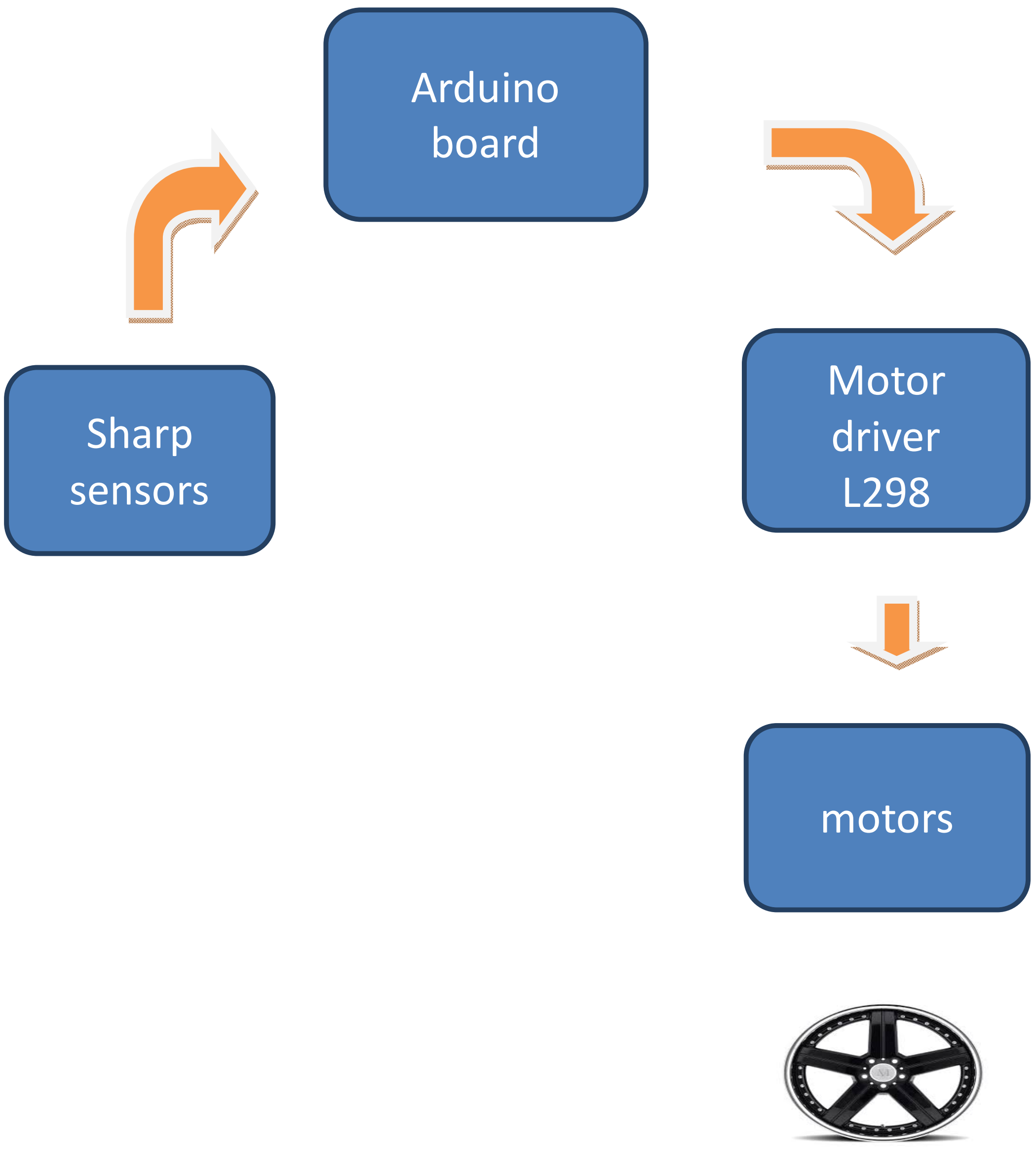


Figure 1.2 :system working

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.

The motor driver is powered by 12V power supply. The PWM sent to the motor driver has low voltage=0V and high voltage=5V. the motor driver IC L298 (or L293D) scales it to the range low voltage=0V and high voltage=12V (motor driver is used because motors need 12 V to run. Also they need high current . Microcontroller output ports can not supply 12 V and high current). Then this PWM is applied across the motor .so ,if the robot tilts such that height of sensor 1 decreases , then accordingly outputPWM will be such that (outputPWM will be positive) motors will run to accelerate the robot to sensor 1 side. Because of this this acceleration robot will experience a torque (due to pseudo force in the non inertial frame) that will oppose the torque due to gravity and it will bring the robot platform back to the horizontal level.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.

The motor driver is powered by 12V power supply. The PWM sent to the motor driver has low voltage=0V and high voltage=5V. The motor driver IC L298 (or L293D) scales it to the range low voltage=0V and high voltage=12V (motor driver is used because motors need 12 V to run. Also they need high current . Microcontroller output ports can not supply 12 V and high current). Then this PWM is applied across the motor .so ,if the robot tilts such that height of

sensor 1 decreases , then accordingly output PWM will be such that (output PWM will be positive) motors will run to accelerate the robot to sensor 1 side. Code here makes it more clear. Because of this acceleration robot will experience a torque (due to pseudo force in the non inertial frame) that will oppose the torque due to gravity and it will bring the robot platform back to the horizontal level.

2. SURVEY

2.1 Existing System

The concept of balancing robot is based on the inverted pendulum model. This model has been widely used by researches around the world in controlling a system not only in designing wheeled robot but other types of robot as well such as legged robots. Researches at the Industrial Electronics Laboratory at the Swiss Federal Institute of Technology have built a prototype two wheel robot in which the control is based on a Digital Signal Processor. A linear state space controller using information from a gyroscope and motor encoder sensors is being implemented to make this system stabilize. (Grasser et al.2002). Another two wheeled robot called ‘SEGWAY HT’ is available commercially (Dean Kamen ,2001) . It is invented by Dean Kamen who has design more than 150 systems which includes climate control systems and helicopter design. An extra feature this robot has is that it is able to balance while a user is standing on top of and navigate the terrain with it. However, this uses five gyroscopes and a few other tilt sensors to keep it balanced. Next is the small scale robot, Nbot which is similar to JOE is built by David. P Ander son. (Anderson, David.P)

This robot uses a commercially available inertial sensor and position information from motor encoder to balance the system. This robot has won the NASA cool robot of the week in the year 2003. Steven Hassenplug used a more innovative approach to construct a balancing robot (Steve Hassenplug, 2002). The chassis of the body is constructed by using the LEGO Mindstorms robotics kit. The balancing method of controlling the system is unique with two Electro-Optical Proximity Detector sensors is used to provide the tilt angle information for the controller. This omits the conventional use of gyroscope that has been used by previous robot researchers. Louis Brennan, an Irish-Australian inventor, was one of the first to patent a gyroscopic stabilizing vehicle. In 1903, Brennan patented a gyroscopically balanced monorail system that he designed for military use; he successfully demonstrated the apparatus in 1909. By mounting one or more gyrostats (a modified gyroscope) along the body, the monorail balanced itself when its equilibrium was disturbed. Brennan feared that the gyrostats would fail in use,

causing total system failure; thus, he prevented the monorail from being mass-produced. More recently, a group from Columbia University manufactured a modernized version of Brennan’s monorail. Unfortunately, the group was unable to create a working model. The electronic component of the model continuously overheated during operation, causing the motor to burn out. The electronic segment was improperly modeled, which led to the mechanism's inability to perform

2.2 Problem Description

I've made some research around and i've come to the conclusion that it should be feasible. The most difficult part is probably going to be the programming, but i'm a computer programmer (quite skilled, btw :P) and i've already have some ideas on how to build it. I worked on some videogames in the past and i have a background on 3D math and physics that should help a lot. I don't know if it will work, in the worst case i'll use the parts to make a standard non balancing robot .

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. This is because as soon as the robot accelerates in a particular direction we can find that the torque due to the pseudo force in the non-inertial frame of accelerating robot opposes the torque due to gravity and if it is high enough (obviously depends on acceleration) it can bring the robot platform back to the horizontal level[7.6]

- Inadequate acceleration (and so pseudo force)
- Center of gravity much above the wheel axle. What is the soultion?
- To make the response fast we should use fast responce giving motors.
- To increase the acceleration we should increase motors rpm from 60rpm to higher rpm (say 150 rpm)
- We can lower the centre of gravity by shifting the circuit and battery to the bottom. .

- So finally , we need to replace simple 60 rpm dc motors by 150 rpm Johnson Motors giving fairly fast response.

2.3 Proposed System

We decided to complete our project on three levels. 1) Mechanical 2) Electrical 3) Coding. While working on the mechanical level we first built mechanical chassis of the robot. This job included simply joining of cut acrylic sheets by using screws and L-clamps and attaching motors & tires. Then working on the electrical level we made a simple L293D/L298 motor driver circuit. That included IC 7805, 7809 also. We have used Arduino Duemilanove ATmega 328 microcontroller board. IC 7809 was needed for powering this board (constantly at 9 Volts). Sensors used by us(sharp sensors GP2D120) needed 5V supply. So we needed IC 7805 that generates constant 5V output voltage.

This 5V is also used for enabling the motor driver and also as logical high. Working on the coding level we developed a code such that arduino would give proper PWM and PWM direction to the motor driver by analyzing the input data sent by the sensors. The value of the PWM signal and its direction was totally based on the difference between the analog values given by the sensors to arduino. More the difference more the PWM . Direction of PWM application was determined by simply testing whether the output PWM (that code generate using PID control mechanism) is positive or negative.

3. IMPLEMENTATION

1 Ultrasonic sensors HC-SR04

2 Arduino atmega2560

3 Li-ion battery,

4 120 rpm 12 V Johnson Motors,

5 L293D, L298N

6 IC 7805,7809,

7 Two wheels,

8 And other necessary stuff for robot making(eg,acrylic sheets, L-clamps, connecting wires, connectors, PCB, double sided tape).

3.1 L298N as Motor driver

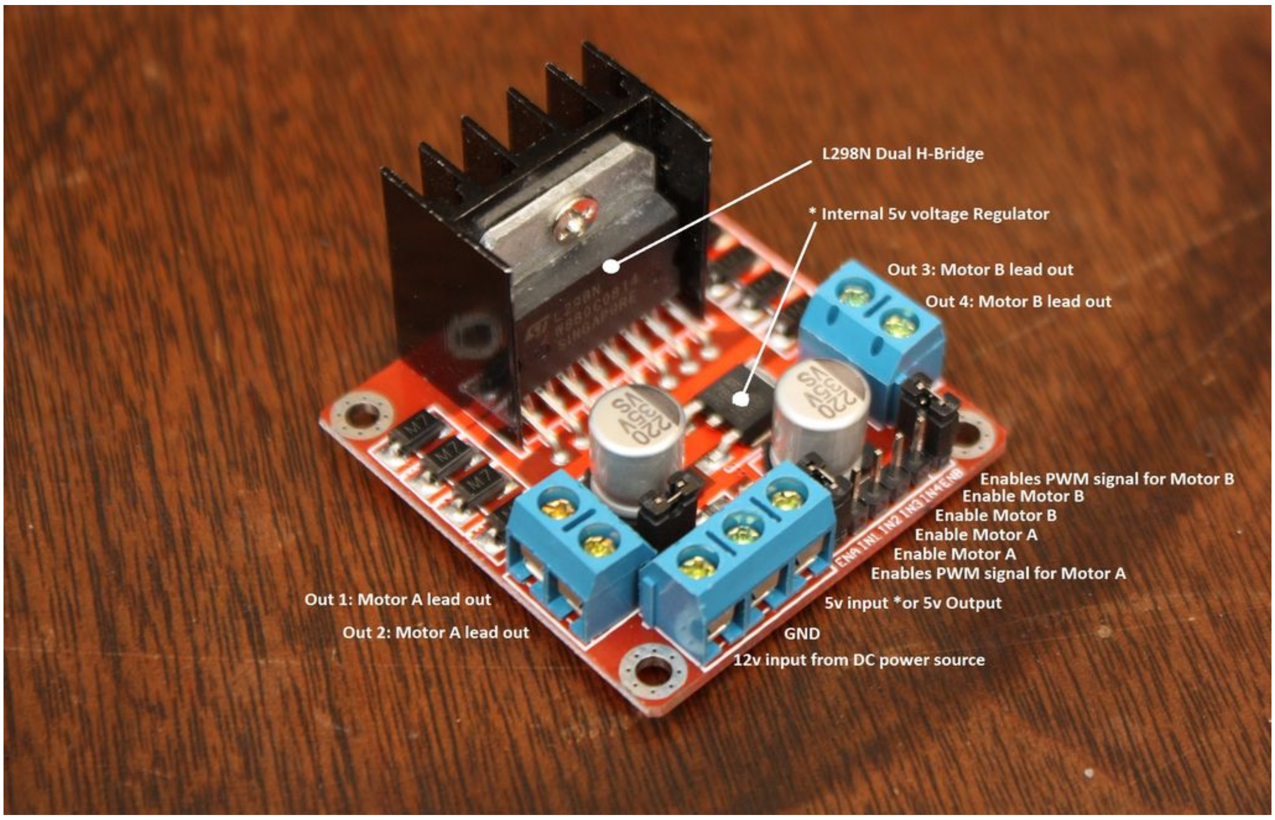


Figure 3.1 :motor driver

Featuring Unitrode L293 and L293D

_ Wide Supply-Voltage Range: 4.5 V to 36 V

_ Separate Input-Logic Supply

_ Internal ESD Protection

_ Thermal Shutdown

_ High-Noise-Immunity Inputs

_ Functionally Similar to SGS L293 and SGS L293D

_ Output Current 1 A Per Channel (600 mA for L293D)

3.1.1 DESCRIPTION

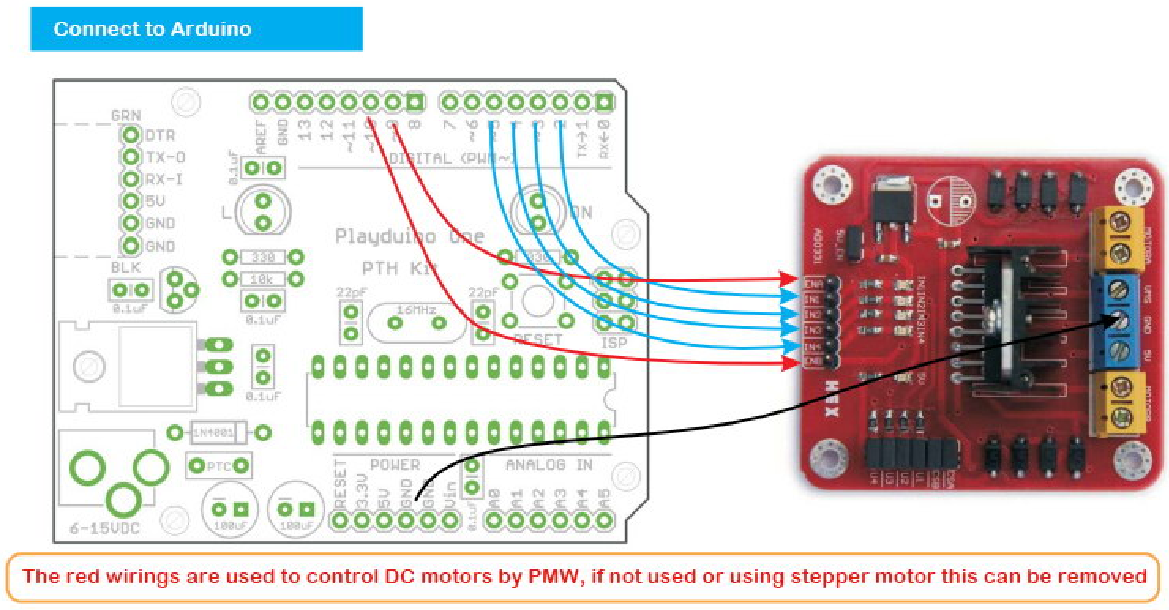
Motor driver IC (L293d or L293D) has input pins, output pins, enable pin , logic supply voltage pin , Main supply voltage pin. L298N is a 15 pin motor driver IC that can drive two motors at a time. It allows high current (2A per motor) to flow through it. So it is useful for driving motors which need 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. When the enable input is low, those drivers are disabled, and their outputs areoff and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or brid reversible drive suitable for solenoid or motor applications.

3.1.2 L298N connection with arduino:


Connect to Arduino

The red wirings are used to control DC motors by PMW, if not used or using stepper motor this can be removed

Figure 3.1.2 :l298n connection with arduino

3.2 ARDUINO ATMEGA 2560

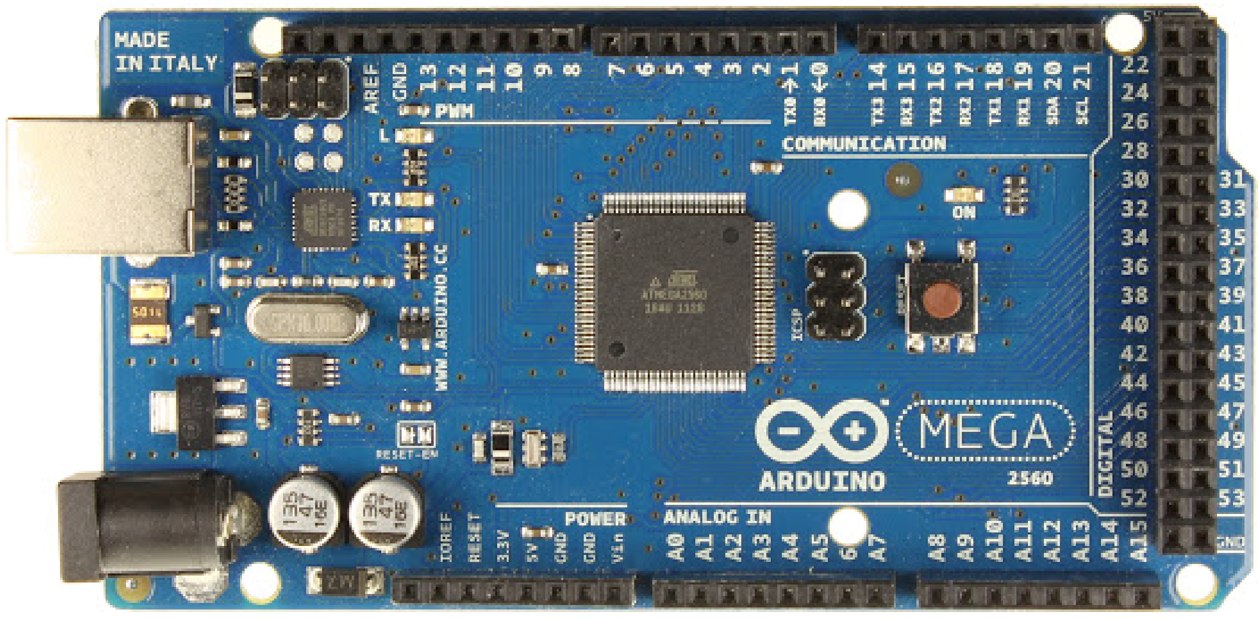


Figure 3.2:arduino atmega 2560

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. The power pins are as follows:

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-series driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

3.2.1 ARDIUNO PIN DESCRIPTION

VCC

Digital supply voltage.

GND

Ground.

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and sourcecapability. As inputs, Port B pins that are externally pulled low will source current if the pull-upresistors are activated. The Port B pins are tri-stated when a reset condition becomes active,even if the clock is not running.Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.If the Internal Calibrated RC Oscillator is used as chip clock source, PB7...6 is used asTOSC2...1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set. The various special features of Port B are elaborated in and ”System Clock and Clock Options”

Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5...0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/RESET

If the RSTDISBL Fuse is programmed,PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of PortC. If the RSTDISBL Fuse is unprogramed PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length of will generate a Reset, even if the clock is not running.

The minimum pulse length is given in Table 28-12 on page 323. Shorter pulses are not guaranteed to generate a Reset.The various special features of Port C are elaborated in ”Alternate Functions of Port C”

Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-u p resistors are activated. The Port D pins are tri-stated when a reset condition becomes active,even if the clock is not running.

AVcc

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6...4 use digital supply voltage, VCC.

AREF

AREF is the analog reference pin for the A/D Converter.

ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter.These pins are powered from the analog supply and serve as 10-bit ADC channels.

In our project we are using arduino atmega 2560. The pin diagram of arduino board is shown below:

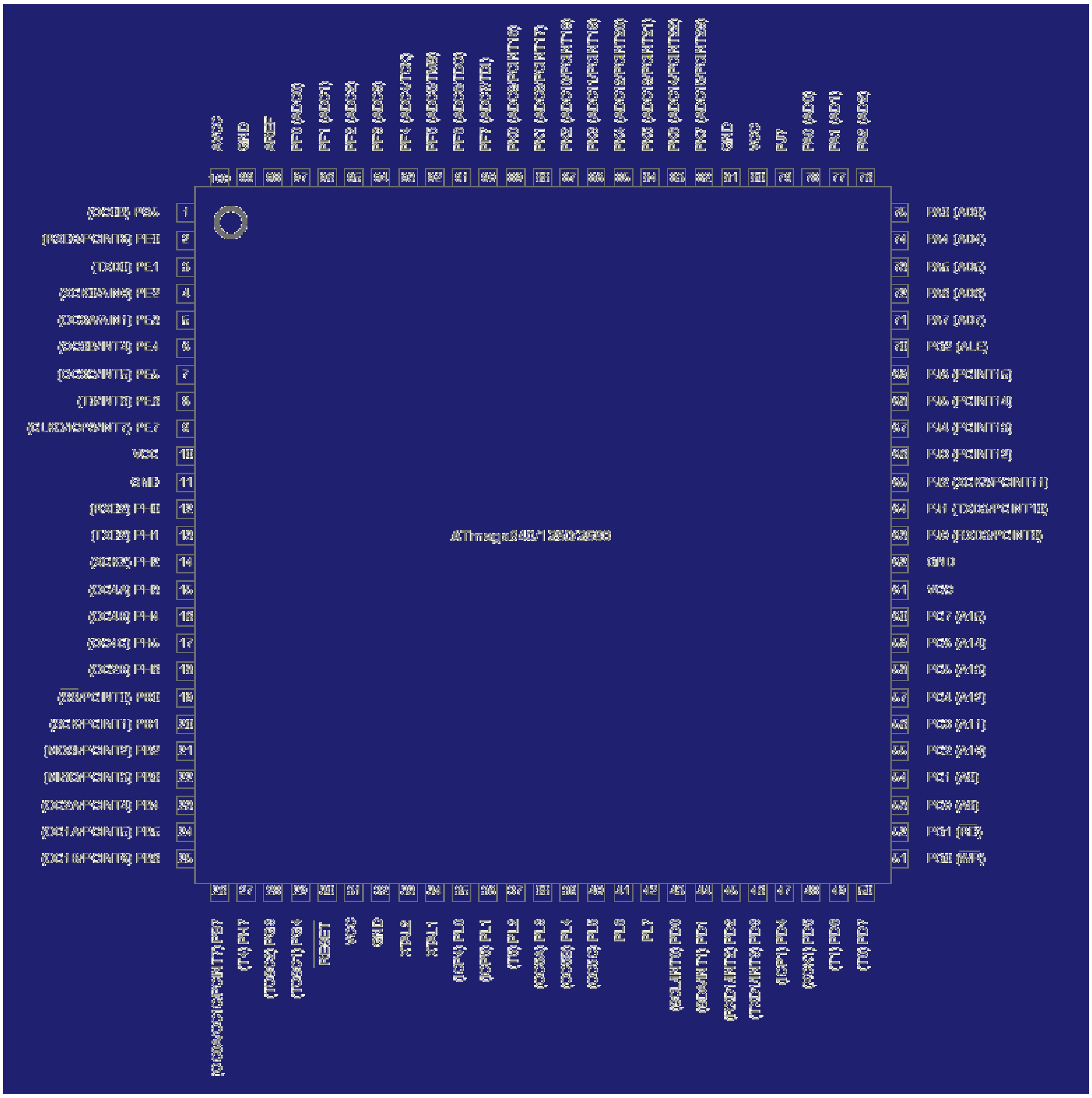


Figure 3.2.1:arduino pin description

3.2.2 ARDUINO SPECIFICATION

Microcontroller ATmega2560

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limits) 6-20V

Digital I/O Pins 54 (of which 14 provide PWM output)

Analog Input Pins 16

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 256 KB of which 8 KB used by bootloader

SRAM 8 KB

EEPROM 4 KB

Clock Speed 16 MHz

- **Why Arduino ?**

As we all know that Arduino community is Open Source which means open hardware and open software and also our Institute has taken the initiative to promote Open Source Community In collaboration with Indian Institute of Technology – Bombay (IIT-B) Open Software we are all aware of that we can make changes in the programming of the software and use it as per our requirements or can be modified depending on the application, in other words we can tailor made a software according to the use of our application without any. But what is Open Hardware, hopefully listing it first time. And how can the Hardware can be open source as we are purchasing for it. We all also have the misconception that the Open Source things are free of cost. But over here the hardware is open source because the company has made the Schematic file .sch, the Board file .brd, and all the other related documents and sources available freely on the official website that also in the most common formats like EAGLE PCB.

The Arduino Mega2560 has a number of facilities for communicating with a computer, another arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The arduino software 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 ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Mega's digital pins. The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials. The Atmega2560 on the Arduino Mega comes preburned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the boot loader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

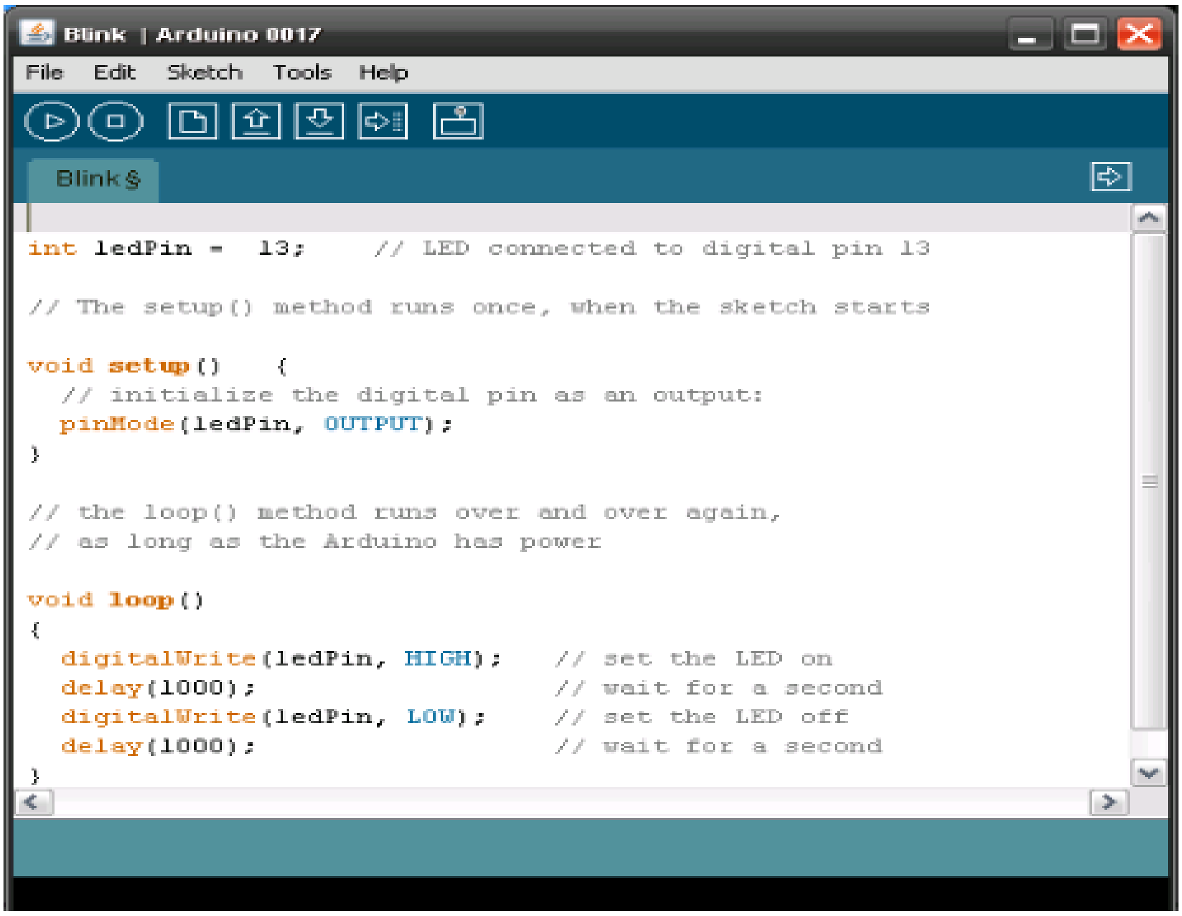
Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS Xor Linux, it resets each time a connection is made to it from software (via USB). For the following half-secondor so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e.anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins.

3.2.3 HOW TO USE ARDUINO

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).Arduino is a cross-platform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions.

Arduino is a cross-platform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.



```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(1000);                  // wait for a second
}
```

Figure 3.2.3:arduino screen

Now you're actually ready to "burn" your first program on the arduino board. Toselect "blink led", the physical translation of the well known programming "helloworld", select

File>Sketchbook>Arduino-0017>Examples>Digital>Blink Once you have your skecth you'll see something very close to the screenshot on the right. In **Tools>Board** select MEGA Now you have to go to **Tools>SerialPort** and select the right serial port, the one arduino is attached to.

3.3 ULTRASONIC SENSOR HC-SR04

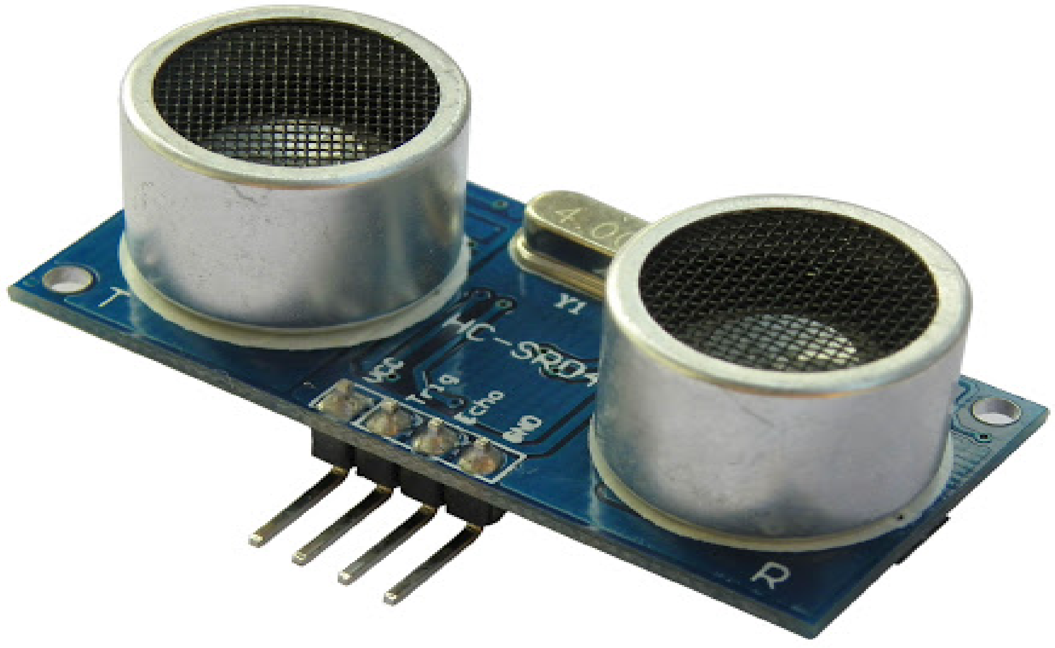


Figure 3.3:ultrasonic sensor

- **Specifications:**

power supply :5V DC

quiescent current : <2mA

effectual angle: <15°

ranging distance : 2cm – 500 cm

resolution : 0.3 cm

- **Product Features:**

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit.

- **The basic principle of work:**

(1) Using I/O trigger for at least 10us high level signal.
(2) The Module automatically sends eight 40 kHz and detect whether there is apulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning. Test distance = (high level time×velocity of sound (340M/S) / 2.

3.3.1 Electrical Parameter

Working Voltage DC 5 V
Working Current 15mAWorking Frequency 40Hz
Max Range 4m
Min Range 2cm
Measuring Angle 15 degree
Trigger Input Signal 10uS TTL pulse
Echo Output Signal Input TTL lever signal and the range in
proportion
Dimension 45*20*15mm

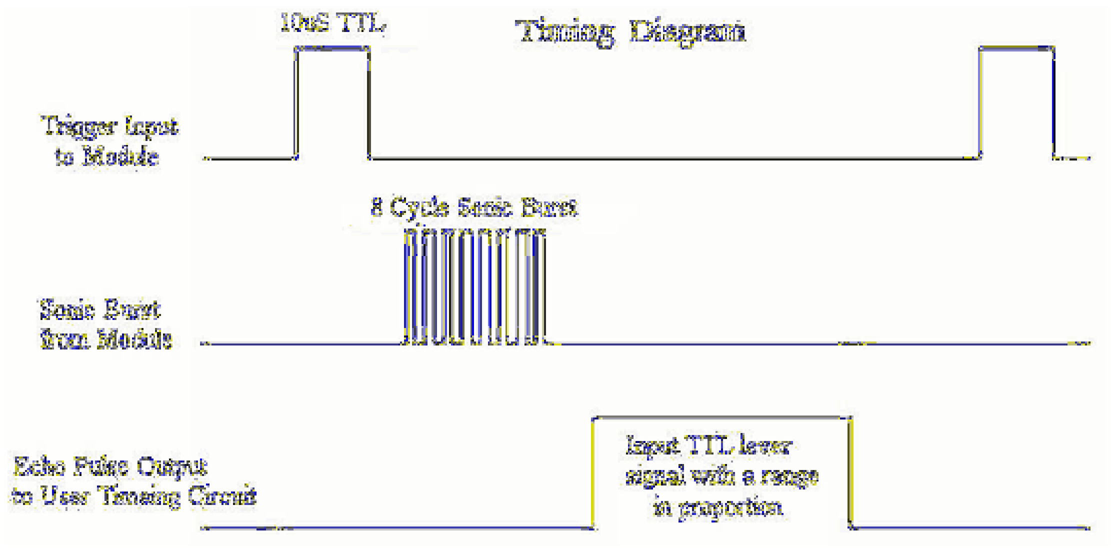

10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timing Circuit
Input TTL lever signal with a range in proportion

Figure 3.3.1:input output trigger pulse

A short ultrasonic pulse is transmitted at the time 0, reflected by an object. The senor receives this signal and converts it to an electric signal. The next pulse can be transmitted when the echo is faded away. This time period is called cycle period. The recommend cycle period should be no less than 50ms. If a 10µs width trigger pulse is sent to the signal pin, the Ultrasonic module will output eight 40kHz ultrasonic signal and detect the echo back. The measured distance is proportional to the echo pulse width and can be calculated by the formula above. If no obstacle is detected, the output pin will give a 38ms high level signal

3.3.2 Ultrasonic sensors connection with arduino:

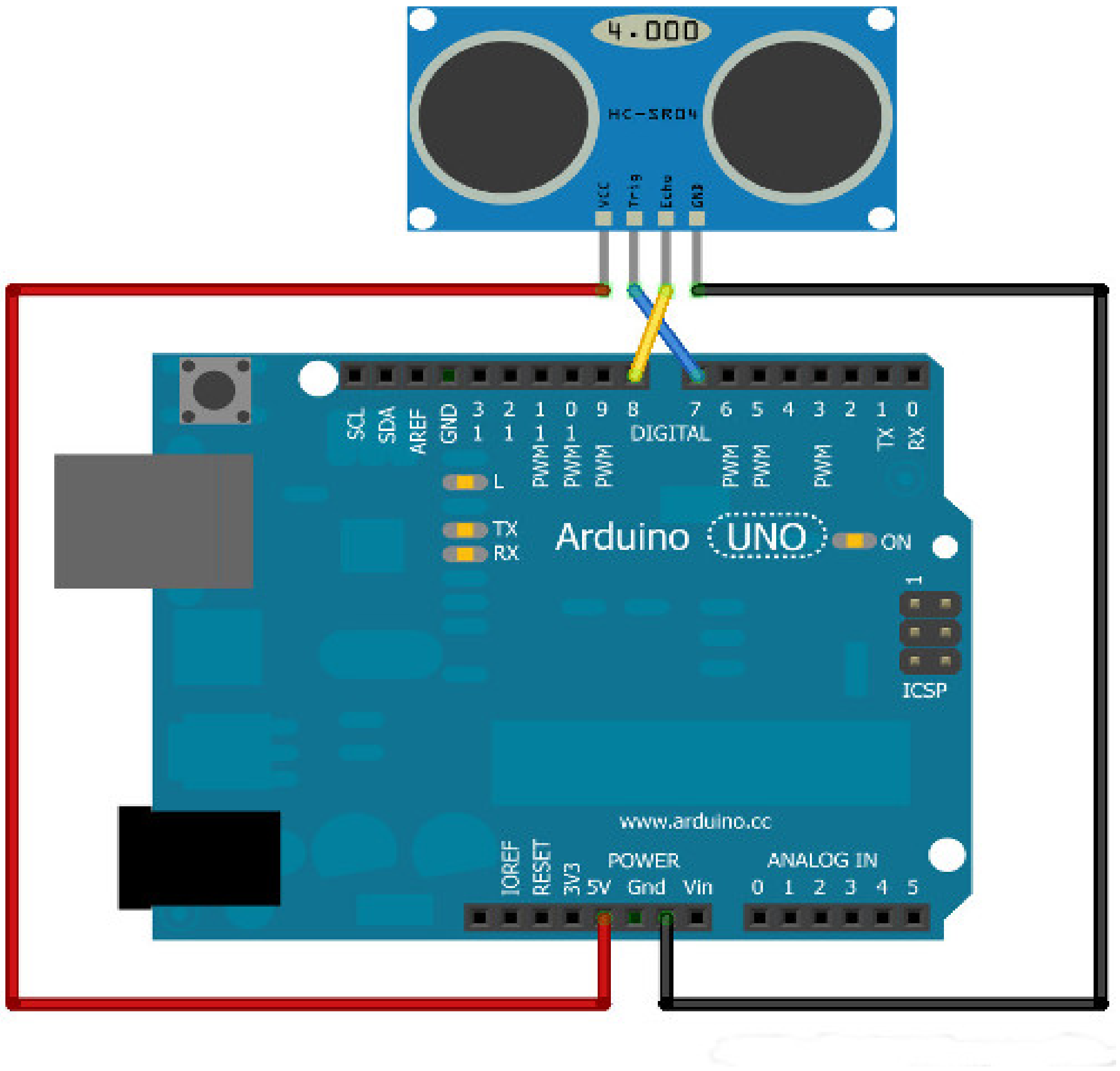


Figure 3.3.2:ultrasonic sensor with arduino

3.4 Lithium-ion battery

Li-ion 18650 Pack
3.7V 6600mAh
Caution
Do not dispose of in fire.
Do not heat, deform, solder, disassemblenor modify.
Do not insert batteries with different connetor. If leaked liquid in eyes, wash them with clean water and consult doctor immediately.

Figure 3.4:li-on battery

. One does not need to wait until these packs are "dead" before charging them.Lithium Ion battery packs are rechargeable battery packs offering longer storage life than NiMH (Nickel Metal Hydride) battery chemistry.Lithium Ion technology batteries do not sustain memory loss or what is commonly referred to as "memory effect". This means they can be charged and recharged even before they have become fully discharged.

4. SOFTWARE IMPLEMENTATION

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials. The ATmega2560 on the Arduino Mega comes preburned 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 STK500 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.

The ATmega2560 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmeg2560 is loaded with a DFU boot loader, which can be activated by: On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground,making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows)or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use theISP header with an external programmer (overwriting the DFU bootloader). See this user contributed tutorial for more information.

- **Automatic (Software) Reset**

Rather than requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment.This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via

USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

- **USB Over-current Protection**

The Arduino Mega2560 has a resettable poly-fuse that protects your computer's USB ports from shorts and over-current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

- **Physical Characteristics and Shield Compatibility**

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to5, the power header, and ICSP header are all in equivalent locations. Further the main UART(serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3respectively).

4.1 ARDUINO CODING

The control algorithm that was used to maintain it balance on the autonomous self-balancing two wheel robot was the PID controller. The proportional, integral, and derivative (PID) controller, is well known as a three term controller.

The input to the controller is the error from the system. The Kp, Ki, and Kd are referred as the proportional, integral, and derivative constants (the three terms get multiplied by these constants respectively). The closed loop control system is also referred to as a negative feedback system. The basic idea of a negative feedback system is that it measures the process output *y* from a sensor. The measured process output gets subtracted from the reference *set-point* value to produce an *error*. The error is then fed into the PID controller, where the error gets managed in three ways. The error will be used on the PID controller to execute the proportional term, integral term for reduction of steady state errors, and the derivative term to handle overshoots. After the PID algorithm processes the error, the controller produces a control signal *u*. The PID control signal then gets fed into the process under control. The process under PID control is the two wheeled robot. The PID control signal will try to drive the process to the desired reference setpoint value. In the case of the two wheel robot, the desired set-point value is the zero degree vertical position.

The PID control algorithm can be modelled in a mathematical representation.PID is used to calculate the 'correction' term : [7.4]

Correction = Kp*error + Ki* ∫error + Kd* d/dt(error);
Kp , Ki and Kd are constants which are set experimentally.

If only the first term had been used to calculate the correction, the robot would have reacted in the same way as in the classical line following algorithm. The second term forces the robot to move towards the mean position faster. The third term resists sudden change in deviation.

The integral term is simply the summation of all previous deviations. Call this integral- ‘total error’. The derivative is the difference between the current deviation and the previous deviation. Following is the code for evaluating the correction.

These lines should run in each iteration :

```
correction = Kp*deviation + Ki*totalerror + Kd*(deviation - previousdeviation);
totalerror += correction;
previousdeviation = deviation;
```

4.2 PROGRAM

```
int led1 = 13;
int led2 = 3;
int m1 = 8;
int m2 = 9;
int m3 = 10;
int m4 = 11;
int trigPin1 = 4;
int echoPin1 = 5;
int trigPin2 = 6;
int echoPin2 = 7;
void setup()
{
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
}

void loop()
{
  long duration1, duration2, distance1, distance2;

  digitalWrite(trigPin1, LOW)
  delay (2);
  digitalWrite(trigPin1, HIGH);
  delay (10);
  digitalWrite(trigPin1, LOW);
  duration1 = pulseIn(echoPin1, HIGH);
  distance1 = (duration1/2) / 29.1;

  delay(0);

  digitalWrite(trigPin2, LOW);
  delay (2);
  digitalWrite(trigPin2, HIGH);
  delay (10);
  digitalWrite(trigPin2, LOW);
 duration2 = pulseIn(echoPin2, HIGH);
 distance2 = (duration2/2) / 29.1;

  if (distance2 < distance1)
  {
    digitalWrite(led1, HIGH);
    digitalWrite(m1, HIGH);
    digitalWrite(m2, LOW);
    digitalWrite(m3, HIGH);
    digitalWrite(m4, LOW);
  }
 // else if(distance1 < 10)
//  {
//    digitalWrite(led2,HIGH);
//  }
  if(distance2 > distance1)
  {
   digitalWrite(led1, LOW);
    digitalWrite(m2, HIGH);
    digitalWrite(m1, LOW);
    digitalWrite(m4, HIGH);
    digitalWrite(m3, LOW);
  }

 /* else
  {
     //digitalWrite(led1, LOW);
     digitalWrite(m2, LOW);
     digitalWrite(m1, LOW);
     digitalWrite(m4, LOW);
     digitalWrite(m3, LOW);
   }*/

}
```

5. APPLICATION

1 Self-balancing unicycle, a self-powered unicycle that balances itself in three dimensions.

2 Monocycle Is The World's First Self Balancing Bike: The Monocycle is a perfect city bike that can be used for short distance travels and can be parked in the least available of spaces. Moreover, the bicycle comes with a strong electric motor to support the riders, so they need not push too hard to speed up the bike.

3 Electric self-balance bike scooter/with Remote switch/balance bike: it's a self balancing electric bike which balance itself on its two wheels.

6. CONCLUSIONS

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.

7. FUTURE IMPROVEMENT

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.

8. REFERENCES

[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] Peter Hemsley, 32-bit signed integer maths for PICS, <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>