Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled "**Self Balancing Bot"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/ Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2015 to June 2016 under the supervision of Dr. Vivek Sehgal (Associate Professor, Computer science and Engineering)**.**The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shivam Gupta (121265) ....

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

Dr. Vivek Sehgal
Associate Professor
Computer Science & Engineering
Dated :)

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our project guide Dr. Vivek Sehgal who helped us in conceptualizing the project and actual building of procedures used to complete the project. We would also like to thank our Head of department for providing us this golden opportunity to work on a project like this, which helped us in doing a lot of research and we came to know about so many things.

At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

Thanking you,

Shivam Gupta (121265)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S No.	Abbreviations	Description
1	IDE	Integrated Development Environment
2	ADT	Android Development Tools
3	SPI	Serial Peripheral Interface
4	EEPROM	Electrically Erasable Programmable Read-Only Memory
5	PID	Proportional-Integral-Derivative

ABSTRACT

This report presents a development self-balancing bot using PID controller. The platform has been designed using mobile robot kits including IMU and two servos, and controlled by an open source microcontroller with PID. An Arduino microcontroller, and a two-degree of freedom (axis) accelerometer and gyroscope or MPU-6050 have been used to create the controlled platform. The controller has been designed to maintain the platform at an initially selected angle when the support structure orientation changes. To simplify things a little bit, I take a simple assumption; the robot's movement should be confined on one axis (e.g. only move forward and backward) and thus both wheels will move at the same speed in the same direction. Under this assumption the mathematics become much simpler as we only need to worry about sensor readings on a single plane. If we want to allow the robot to move sidewise, then you will have to control each wheel independently. The general idea remains the same with a less complexity since the falling direction of the robot is still restricted to a single axis.

CHAPTER 1
INTRODUCTION

1.1Project Background

Designing a mobile robot with special capabilities has become a trend these days for a variety of universal human consumption. It also fits well with the needs and nature of the human lifestyle. Different forms and uses, mobile robots have been designed and are now in the market worldwide.

A mobile robot comprises of three main parts including sensors, logical processing unit and actuator. In this project, a robot that can maintain an upright and balanced position on a platform is designed and developed. The robot consists of Inertial Measurement Units (IMU) sensors, microprocessor and motors. The design is designed and the resulting parameters are used and burned into Arduino UNO controller. The main purpose of the controller is to fuse the wheel encoder, gyroscope and accelerometer sensors to estimate the attitude of the platform and then to use this information to drive there action wheel in the direction to maintain an upright and balanced position platform.

If the platform system itself is not balanced, which means it keeps falling off away from the vertical axis, then a gyro chip is needed to provide the angle position of the inverted pendulum or robot base and input into the controller, which the program itself is a balancing algorithm. The PID controller will then provide a type of feedback signal through PWM control to turn the motor servo clockwise or anticlockwise, thus balancing the platform. These two measurements are summed and fed-back to the actuator which produces the counter torque required to balance the platform robot. 2

In this project, the PID will be used because it is relatively easy to implement yet practical. Besides that, PID controller has only three adjustable parameters that can be determined from several techniques. Previous research has shown that PID controller has shown good results in terms of response time and accuracy when the parameters i.e, Kp , Ki and Kd, are properly tuned.

To make a self-balancing robot, it is essential to solve the inverted pendulum problem or an inverted pendulum on cart. While the calculation and expressions are very complex, the goal is quite simple: the goal of the project is to adjust the wheels' position so that the inclination angle remains stable within a pre-determined value (e.g. the angle when the robot is not outside the premeasured angel boundary). When the robot starts to fall in one direction, the wheels should move in the inclined direction with a speed proportional to angle and acceleration of falling to correct the inclination angle. So I get an idea that when the deviation from equilibrium is small, we should move "gently" and when the deviation is large we should move more quickly.

To simplify things a little bit, I take a simple assumption; the robot's movement should be confined on one axis (e.g. only move forward and backward) and thus both wheels will move at the same speed in the same direction. Under this assumption the mathematics become much simpler as we only need to worry about sensor readings on a single plane. If we want to allow the robot to move sidewise, then you will have to control each wheel independently. The general idea remains the same with a less complexity since the falling direction of the robot is still restricted to a single axis.

1.2Problem Statements

Control systems are often designed to improve stability, speed of response, steady-state error, or prevent oscillations. Many researchers wants to produce a mathematical equation that is able to determine the position of a very accurate motor position, thus the steady state error should be zero. DC motor systems have played an important role in the improvement and development of the industrial revolution. Therefore, the development of a more efficient control strategy that can be used for the control of a DC servomotor system and a well defined mathematical model that can be used for off line simulation are essential for this type of systems. Servomotor systems are known to have nonlinear parameters and dynamic factors, so to make the systems easy to control, conventional control methods such as PID controllers are very convenient. Also, the dynamics of the servomotor and outside factors add more complexity to the analysis of the system, for example when the load attached to the control system changes.

Due to these parameters and factors, this study will apply the PID controller to make the steady-state error, due to continuous disturbance, to be zero. Accordingly this project will review the principles of PID that is used to control the servo movement that depends on the angle captured by the IMU. This project uses the PID to compensate the robot body inclination to stabilize the platform. Among other performance requirements are to reach the final position of the motor position very quickly without excessive overshoot. In this case, focusing on systems that have a finish time of 10 ms and the overshoot is smaller than 25%.

1.3Project Objectives

The aim of this project is to implement PID controller to a mobile robot to maintain its flatness on a moving platform. The objectives of this project are as follows:

a) To design and develop a mobile robot and a flat platform.

b) To design a PID controller to maintain the robot flatness

c) To simulate the controller using Matlab and analyse its performance.

d) To integrate the controller into the mobile robot.

1.4 Project Scopes

The scopes of study are as follows:

a) Using mobile platform kits available in the market.

b) Using sensor fusion to measure the tilts in the X-axis
and Y-axis.

c) Using arduino Uno as the mainboard of the mobile robot.

1.5 Organization of Report

As an overview, the structure of this report is organized as follows:

Chapter 1 describes a general introduction of the project, problem statement project aims and project scope.

Chapter 2 provides details literature review that includes an introduction to some basic concepts and a survey of existing works in the areas of developing an algorithm for solving PID controller to maintain flatness platform. This chapter explains in detail all the researches, studies, theories and gathering that have been make throughout the project.

Chapter 3 discusses the methodology of the project which provides a detailed description of the design to develop a mobile robot using PID controller to maintain the platform flatness. It also discusses about the hardware which has been used in the project.

Chapter 4 discusses about the result and analysis.

Chapter 5 concludes the project and gives suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Previous Case Study

Conducting initial review research is very critical in understanding self balancing platform control techniques. The review of research about related literature conducted in this project summarizes some of topics related to the techniques used for the balancing of platform based on Dc motor position. Comparisons between the present project and the related topics of existing information will also be discussed. The methodologies and the techniques used by other researchers around the globe on the balancing platform topic will also be reviewed.

- **<u>An Azure-driven service for Arduino based PID controllers</u>** designed for control and monitoring of grills and other cooking devices. The "Cloud Cooker" system is a hobbyist project to interface cooking devices to the cloud using a Microsoft Mobile App in an Azure App Service. The system will include Arduino-based controller devices, web services, web apps, and mobile/universal applications.

- Meena et al. (2011) **proposed a design for a servo motor controller in discrete-time system to obtain the transfer function of the PID controller design**. MATLAB / Simulink has been used to confirm the effectiveness of this new design method, which provides a simple and powerful way to design a speed controller for servo motor. It also extracted a DC servo motor mathematical model and equations and there were three different motion controllers that were designed and simulated to control the velocity of the motor.

- Popescu et al. (2011) **did a comparison between PID and Fuzzy controllers used in mobile robot control**. There is a significant problem for fuzzy controllers in which computing time is longer than the PID because a lot of complex operations such as requiring fuzzification, inference, and defuzzification.

- Masakazu et al. (2005) **proposed a tuning method for PID controller that considers changes in system characteristics.** It is about the concept of using the optimization of PID controller tuning, depending on the obstacles on the control input derivatives and considering model uncertainties caused by changes in the system dynamics. Partial model matching method was used to evaluate performance and control while the reference referred to interference and repression compared to the tracking properties.

- Arpit et al. (2012) **proposed a performance comparison of PID and Fuzzy logic controller using different defuzzification techniques for positioning control of dc motors.** The result of the fine-tuned PID controller gives relatively less overshoot and settling time with no steady state error. The fuzzy logic controller with different defuzzification techniques gives zero % overshoot and lesser settling time.

- In a paper titled **'Attitude Estimation Using Low Cost Accelerometer and Gyroscope'** written by Young SooSuh (2003), it shows two different sensors which are the accelerometer and gyroscope that exhibit poor results when used separately to determine the attitude which is referred as the pitch angle or roll angle. However, the gyroscope can combine with accelerometer to determine the pitch or roll angle with much better result with the use of Kalman filter.

- Tomislav et al (2012**) proposed self-balancing mobile robot tilter.** It provides a summary of work done in the field of electronic, mechanical design, software design, system characterization and control theory. Robotic system model and simulation results of various control methods required for the stabilization of the system were studied. Dynamic effects become increasingly important in assessing performance limits in robotic. The processes where the project was carried out including design and production of certain parts of the integration section, electronic, mechanical and software.

- V.J. Van Doren (2009) **suggested a two wheeled robot to perform the balancing and control of mobile robots**. In this project the Proportional, Integral, Derivative (PID) has been implemented to control the flatness of a mobile robot platform. PID has proven to be popular among the control engineering community.

As stated by the author of article Vance J. VanDoren (2009), “For more than 60 years after the introduction of Proportional-Integral-Derivative controllers, remain the workhorse of industrial process control”.

CHAPTER-3

SYSTEM DEVELOPMENT

3.1 Project Methodology

This project is done in three phases. The first phase is to understand and design the mobile robot as well as to implement the theories in the real hardware. The second phase is to understand the PID controller and its characteristic and to design and implement the controller into the robot. The last phase is to analyze the controller performance using M-file in Matlab and compare it with the response of the real hardware.

.

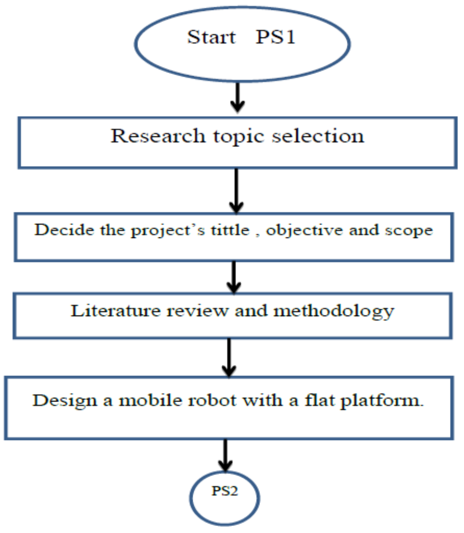

Start PS1
Research topic selection
Decide the project's tittle , objective and scope
Literature review and methodology
Design a mobile robot with a flat platform.
PS2

Figure 1- Flow chart of process methodology PS1

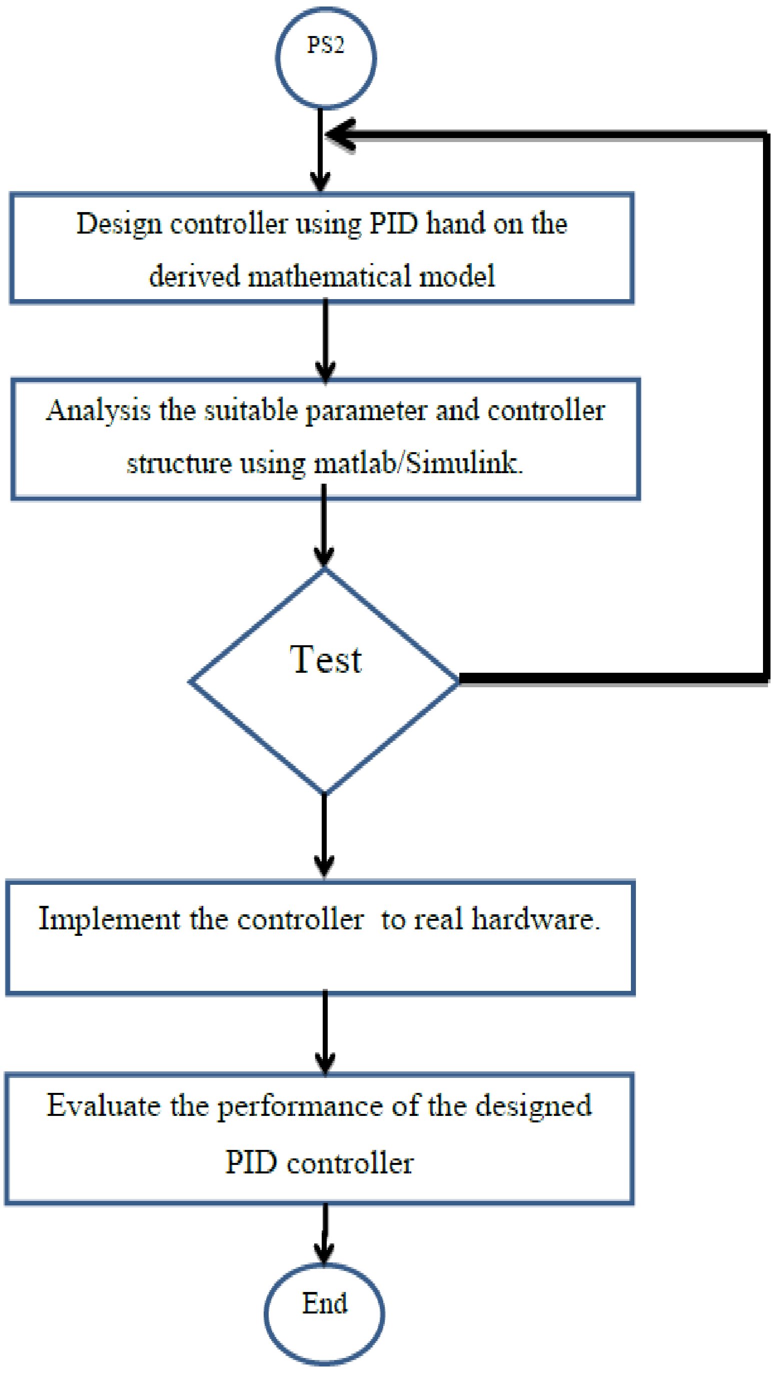

PS2
Design controller using PID hand on the derived mathematical model
Analysis the suitable parameter and controller structure using matlab/Simulink.
Test
Implement the controller to real hardware.
Evaluate the performance of the designed PID controller
End

Figure 2- Flow chart of process methodology PS2

3.2 HARDWARE

3.2.1 ARDUINO UNO

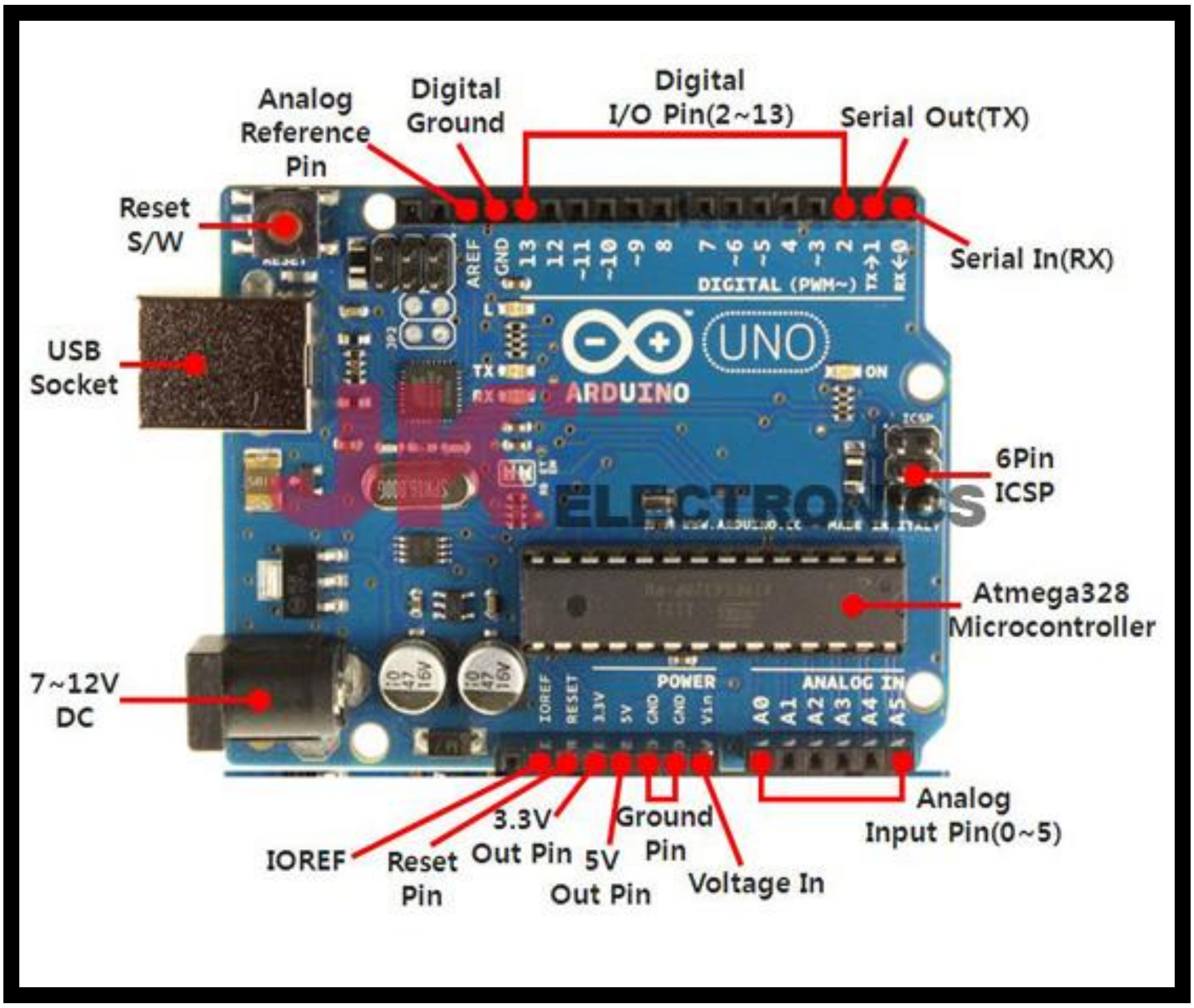


FIGURE 3: Arduino Uno

The 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 worrying 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.

Programming

The 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 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 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 ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, 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 rese ing 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.

Warnings

The Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. 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.

Differences with other boards

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Power

The Uno board 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 from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows.

- **Vin**. The input voltage to the Uno board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V**.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3**. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND**. Ground pins.
- **IOREF**. This pin on the Uno board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

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 input 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

The Uno has a number of facilities for communicating with a computer, another Uno 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 Software Serial 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.

Technical specs

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
Length	68.6 mm
Width	53.4 mm
Weight	25 g

PIN DIAGRAM

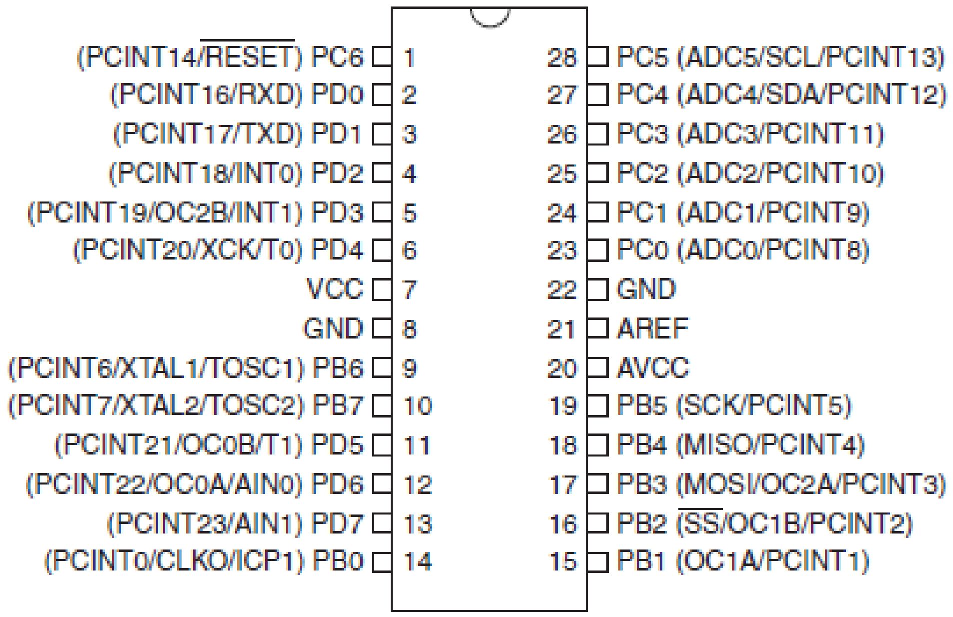


FIGURE 4: Atmel Atmega 328 microcontroller

3.2.2 L298 Dual H-Bridge Motor Driver

Double H driver module uses ST L298N dual full-bridge driver. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors.

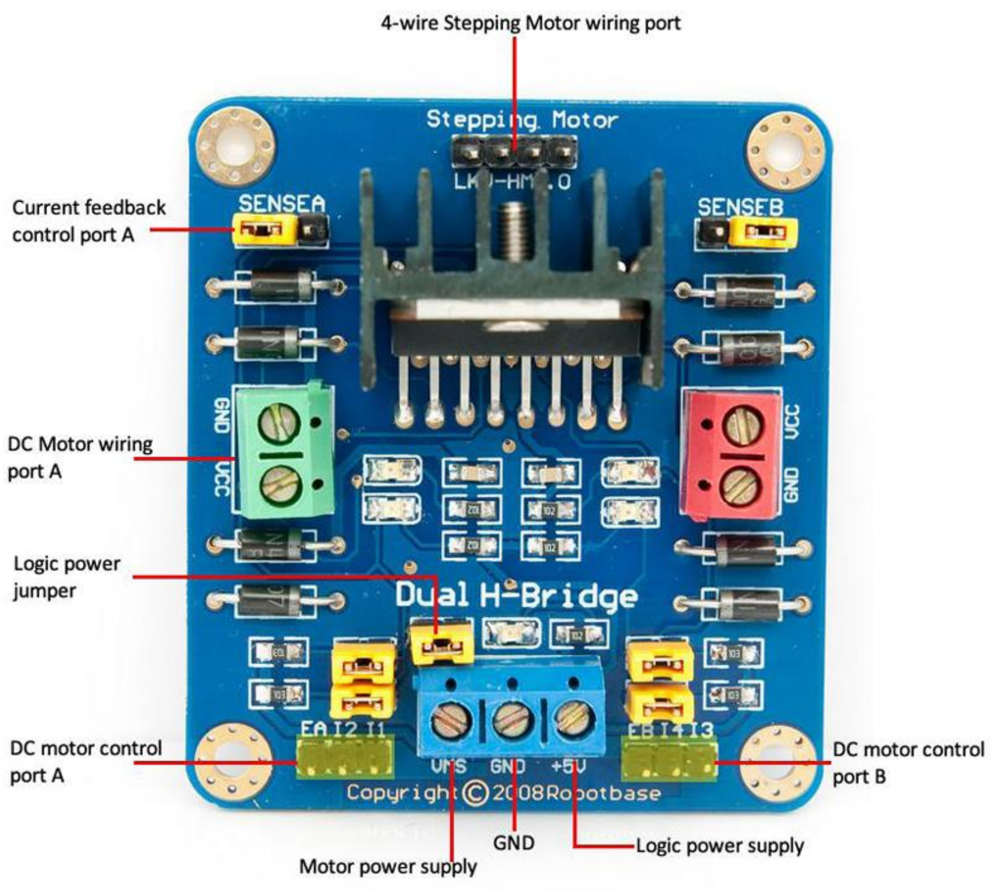


Figure 5: Dual H-Bridge

FEATURES

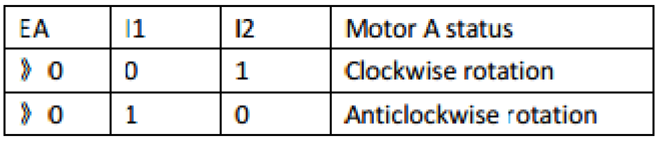
- Light weight, small dimension
- Super driver capacity
- FWD protection
- Heavy load heat sink
- Power selection switch
- 4 pull up resistor switch
- 2 DC motor / 4 coil dual phrase stepper motor output
- Motor direction indication LED
- 4 standard mouting holes

SPECIFICATIONS

- Driver:L298
- Driver power supply: +5v~+46v
- Driver peak current: 2A
- Logic current:0~36mA
- Controlling level
- Max drive power:25W
- Dimension: 60mm*54mm
- Driver Weight: ~48 gm

HARDWARE INSTALLATION

Double H driver module can drive two DC motors at the same time. Port A is completely symmetrical as port B on the board. DC motor input port A has three pins, I1, I2 and EA. I1 and I2 are digital ports which are used to control the direction of motor, EA is connecting with PWM port of control board to control the speed of motor. If I1=1and I2=0, the motor rotates clockwise. If I1=0 and I2=1, it rotates anticlockwise. If I1=I2， it stops rotating.



EA	I1	I2	Motor A status
》 0	0	1	Clockwise rotation
》 0	1	0	Anticlockwise rotation

Normally we do not need to connect wire to supply logic power. Only if the motor power supply is +5V～+7V or +18V～+46V we have to take off the logic power jumper and connect 5v to supply logic power. We can control 4-wire stepping Motor the same way as two DC motors with the signals from EA,I1,I2 and EB,I3,I4.

3.2.3 DC Geared Motor

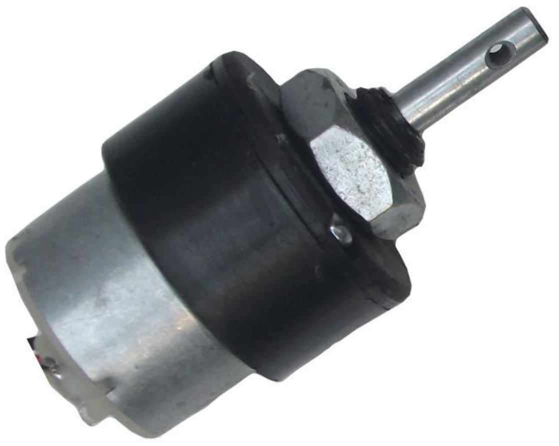


FIGURE 6: DC Motor

A **DC motor** is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line.

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

Electromagnetic motors

A coil of wire with a current running through it generates an electromagnetic field aligned with the center of the coil. The direction and magnitude of the magnetic field produced by the coil can be changed with the direction and magnitude of the current flowing through it.

A simple DC motor has a stationary set of magnets in the stator and an armature with one or more windings of insulated wire wrapped around a soft iron core that concentrates the magnetic field. The windings usually have multiple turns around the core, and in large motors there can be several parallel current paths. The ends of the wire winding are connected to a commutator. The commutator allows each armature coil to be energized in turn and connects the rotating coils with the external power supply through brushes. (Brushless DC motors have electronics that switch the DC current to each coil on and off and have no brushes.)

The total amount of current sent to the coil, the coil's size and what it's wrapped around dictate the strength of the electromagnetic field created.

The sequence of turning a particular coil on or off dictates what direction the effective electromagnetic fields are pointed. By turning on and off coils in sequence a rotating magnetic field can be created. These rotating magnetic fields interact with the magnetic fields of the magnets (permanent or electromagnets) in the stationary part of the motor (stator) to create a force on the armature which causes it to rotate. In some DC motor designs the stator fields use electromagnets to create their magnetic fields which allow greater control over the motor.

At high power levels, DC motors are almost always cooled using forced air.

Different number of stator and armature fields as well as how they are connected provide different inherent speed/torque regulation characteristics. The speed of a DC motor can be controlled by changing the voltage applied to the armature. The introduction of variable resistance in the armature circuit or field circuit allowed speed control. Modern DC motors are often controlled by power electronics systems which adjust the voltage by "chopping" the DC current into on and off cycles which have an effective lower voltage.

Since the series-wound DC motor develops its highest torque at low speed, it is often used in traction applications such as electric locomotives, and trams. The DC motor was the mainstay of electric traction drives on both electric and diesel-electric locomotives, street-cars/trams and diesel electric drilling rigs for many years. The introduction of DC motors and an electrical grid system to run machinery starting in the 1870s started a new second Industrial Revolution. DC motors can operate directly from rechargeable batteries, providing the motive power for the first electric vehicles and today's hybrid cars and electric cars as well as driving a host of cordless tools. Today DC motors are still found in applications as small as toys and disk drives, or in large sizes to operate steel rolling mills and paper machines. Large DC motors with separately excited fields were generally used with winder drives for mine hoists, for high torque as well as smooth speed control using thyristor drives. These are now replaced with large AC motors with variable frequency drives.

If external power is applied to a DC motor it acts as a DC generator, a dynamo. This feature is used to slow down and recharge batteries on hybrid car and electric cars or to return electricity back to the electric grid used on a street car or electric powered train line when they slow down. This process is called regenerative braking on hybrid and electric cars. In diesel electric locomotives they also use their DC motors as generators to slow down but dissipate the energy in resistor stacks. Newer designs are adding large battery packs to recapture some of this energy

Wound stators

Series connection

A series DC motor connects the armature and field windings in series with a common D.C. power source. The motor speed varies as a non-linear function of load torque and armature current; current is common to both the stator and rotor yielding current squared (I^2) behavior. A series motor has very high starting torque and is commonly used for starting high inertia loads, such as trains, elevators or hoists. This speed/torque characteristic is useful in applications such as dragline excavators, where the digging tool moves rapidly when unloaded but slowly when carrying a heavy load.

A series motor should never be started at no load. With no mechanical load on the series motor, the current is low, the counter-EMF produced by the field winding is weak, and so the armature must turn faster to produce sufficient counter-EMF to balance the supply voltage. The motor can be damaged by overspeed. This is called a runaway condition.

Series motors called universal motors can be used on alternating current. Since the armature voltage and the field direction reverse at the same time, torque continues to be produced in the same direction. However they run at a lower speed with lower torque on AC supply when compared to DC due to reactance voltage drop in AC which is not present in DC.[3]Since the speed is not related to the line frequency, universal motors can develop higher-than-synchronous speeds, making them lighter than induction motors of the same rated mechanical output. This is a valuable characteristic for hand-held power tools. Universal motors for commercial utility are usually of small capacity, not more than about 1 kW output. However, much larger universal motors were used for electric locomotives, fed by special low-frequency traction power networks to avoid problems with commutation under heavy and varying loads.

Shunt connection

A shunt DC motor connects the armature and field windings in parallel or shunt with a common D.C. power source. This type of motor has good speed regulation even as the load varies, but does not have the starting torque of a series DC motor. It is typically used for industrial, adjustable speed applications, such as machine tools, winding/unwinding machines and tensioners.

Compound connection

A compound DC motor connects the armature and fields windings in a shunt and a series combination to give it characteristics of both a shunt and a series DC motor. This motor is used when both a high starting torque and good speed regulation is needed. The motor can be connected in two arrangements: cumulatively or differentially. Cumulative compound motors connect the series field to aid the shunt field, which provides higher starting torque but less speed regulation. Differential compound DC motors have good speed regulation and are typically operated at constant speed.

3.2.4 MPU – 6050

The InvenSense MPU-6050 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefor it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino.

The MPU-6050 is not expensive, especially given the fact that it combines both an accelerometer and a gyro.

Also note that Invensense has combined the MPU-6050 with a magnetometer (compass) in a single chip called MPU-9150.

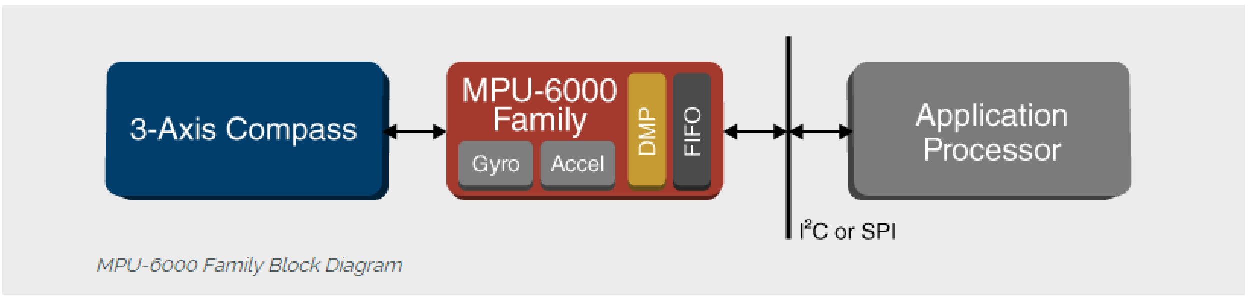
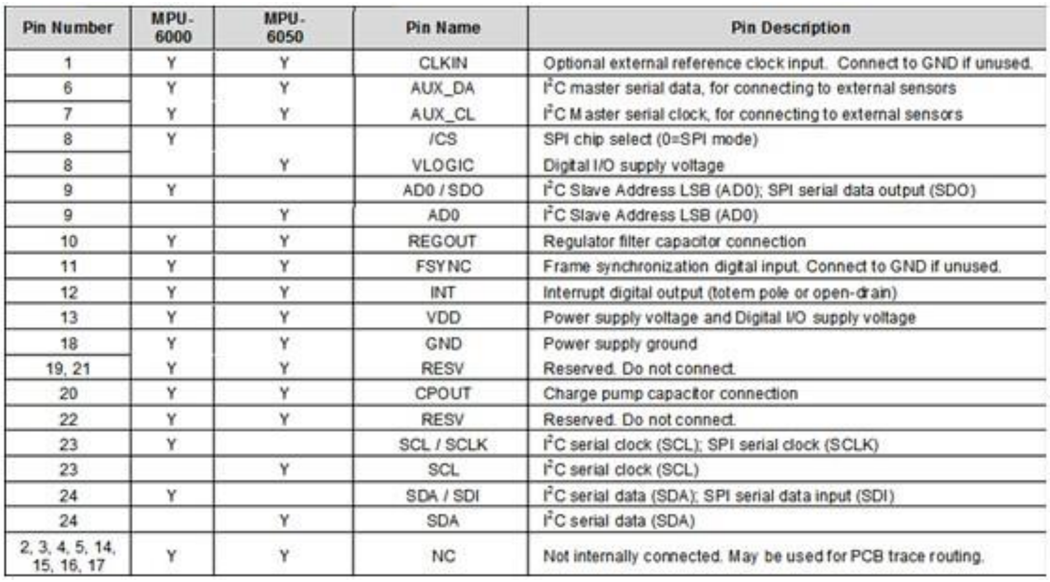
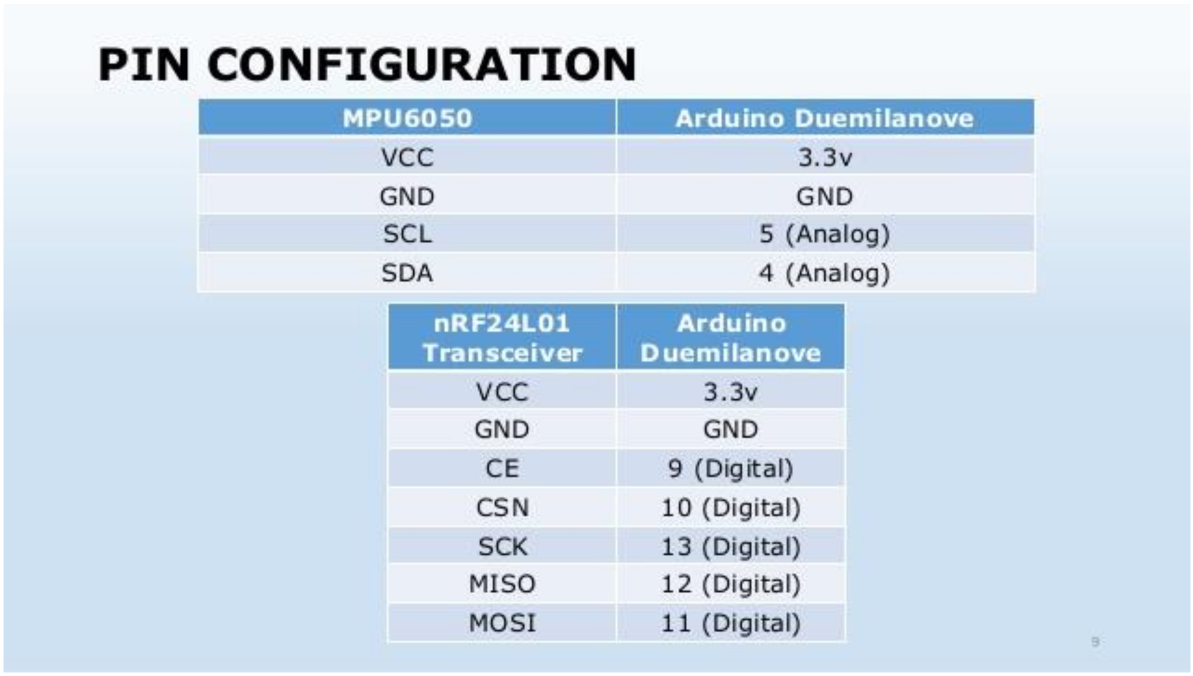


FIGURE 7: MPU -6000 Family Block Diagram

PIN DESCRIPTION



Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I^2C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I^2C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (0=SPI mode)
8		Y	VLOGIC	Digital I/O supply voltage
9	Y		AD0 / SDO	I^2C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I^2C Slave Address LSB (AD0)
10	Y	Y	REGOUT	Regulator filter capacitor connection
11	Y	Y	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	Y	Y	INT	Interrupt digital output (totem pole or open-drain)
13	Y	Y	VDD	Power supply voltage and Digital I/O supply voltage
18	Y	Y	GND	Power supply ground
19, 21	Y	Y	RESV	Reserved. Do not connect.
20	Y	Y	CPOUT	Charge pump capacitor connection
22	Y	Y	RESV	Reserved. Do not connect.
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock (SCL)
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data (SDA)
2, 3, 4, 5, 14, 15, 16, 17	Y	Y	NC	Not internally connected. May be used for PCB trace routing.



PIN CONFIGURATION

MPU6050	Arduino Duemilanove
VCC	3.3v
GND	GND
SCL	5 (Analog)
SDA	4 (Analog)

nRF24L01 Transceiver	Arduino Duemilanove
VCC	3.3v
GND	GND
CE	9 (Digital)
CSN	10 (Digital)
SCK	13 (Digital)
MISO	12 (Digital)
MOSI	11 (Digital)

9

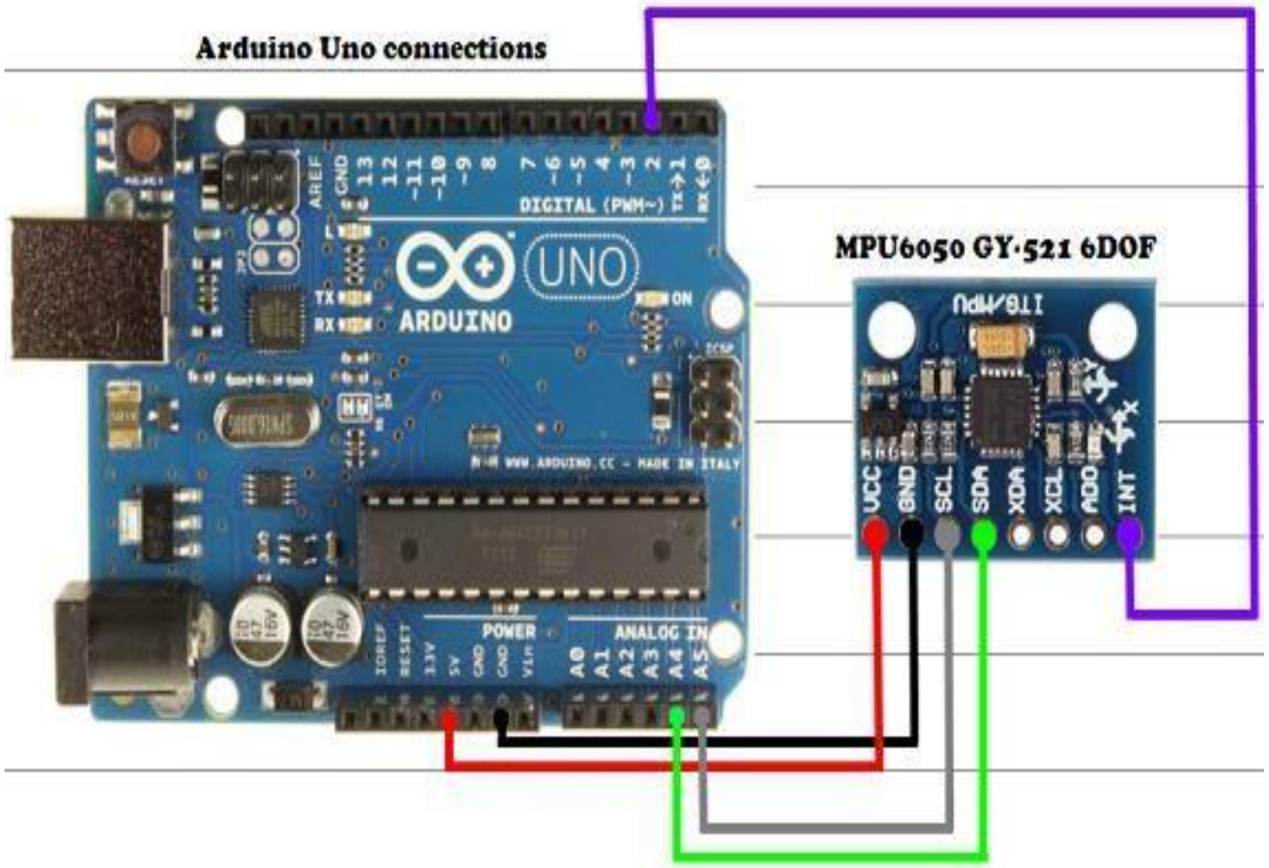


FIGURE 9: Arduino & MPU6050

Bluetooth Module V2.0

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

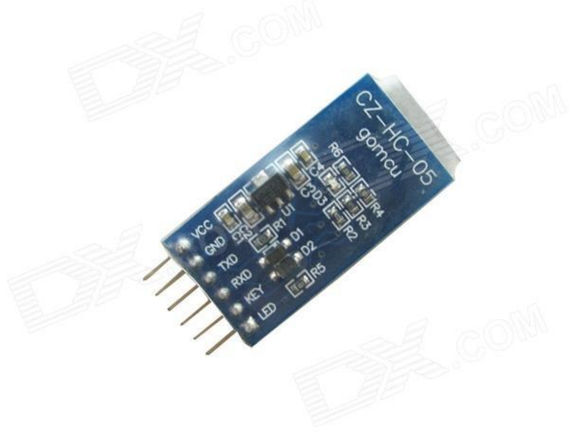


FIGURE 10: Bluetooth Module

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband .It uses CSR Blue core 04-External single chip Bluetooth system with CMOS technology and with Adaptive Frequency Hopping Feature. It has the footprint as small as 12.7mmx27mm.

Hardware Features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control

• UART interface with programmable baud rate

• With integrated antenna

• With edge connector Software Features

• Default Baud rate: 38400, Data bits: 8, Stop bit:1Parity:No parity,

• Data control: has supported baud rate: 9600, 19200, 38400, 57600, 115200, etc.

• Given a rising pulse in PIO0, device will be disconnected.

• Status instruction port PIO1: low-disconnected, high-connected;

• PIO10 and PIO11 can be connected to red and blue led separately.

• When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2 times/s.

• Auto-connect to the last device on power as default.

• Permit pairing device to connect as default.

• Auto-pairing PINCODE:"0000" as default

• Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Temperature Sensor (LM35)

LM35 is a basic temperature sensor that can be used for experimental purpose. It gives the readings in centigrade (degree Celsius) since its output voltage is linearly proportional to temperature.

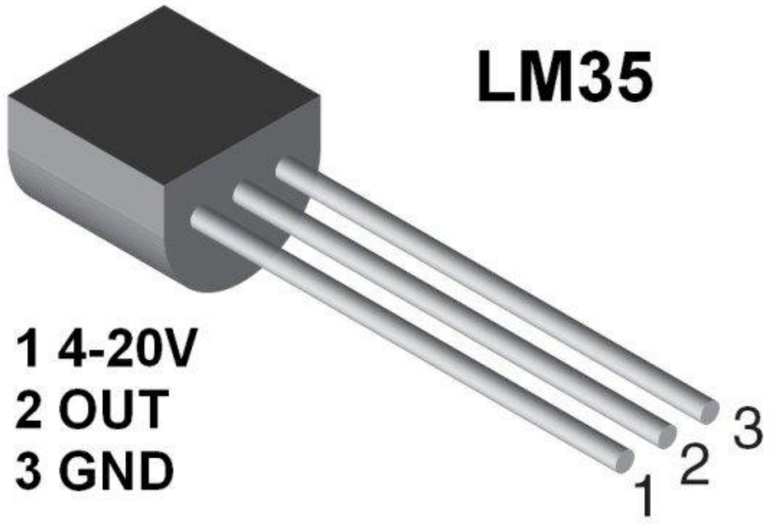


Figure 11: Temperature Sensor

It uses the fact that as temperature increases, the voltage across diode increases at known rate (actually the drop across base-emitter junction of transistor). Its disadvantage is its sluggish response. It has an output voltage that is proportional to the Celsius temperature. The scale factor is .01V/C or 1mV/C. Another important characteristic of the LM35 is that it draws only 60 micro amps from its supply and possesses a low self-heating capability. The sensor self-heating causes less than 0.1 C temperature rises in still air. The operating temperature range is from -55°C to 150°C.

3.3) Software Used

Arduino Integrated Development Environment (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino.

The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

ARDUINO
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

FIGURE 12: IDE

3.4) Assembling The Robot

Block Diagram For Balancing Bot

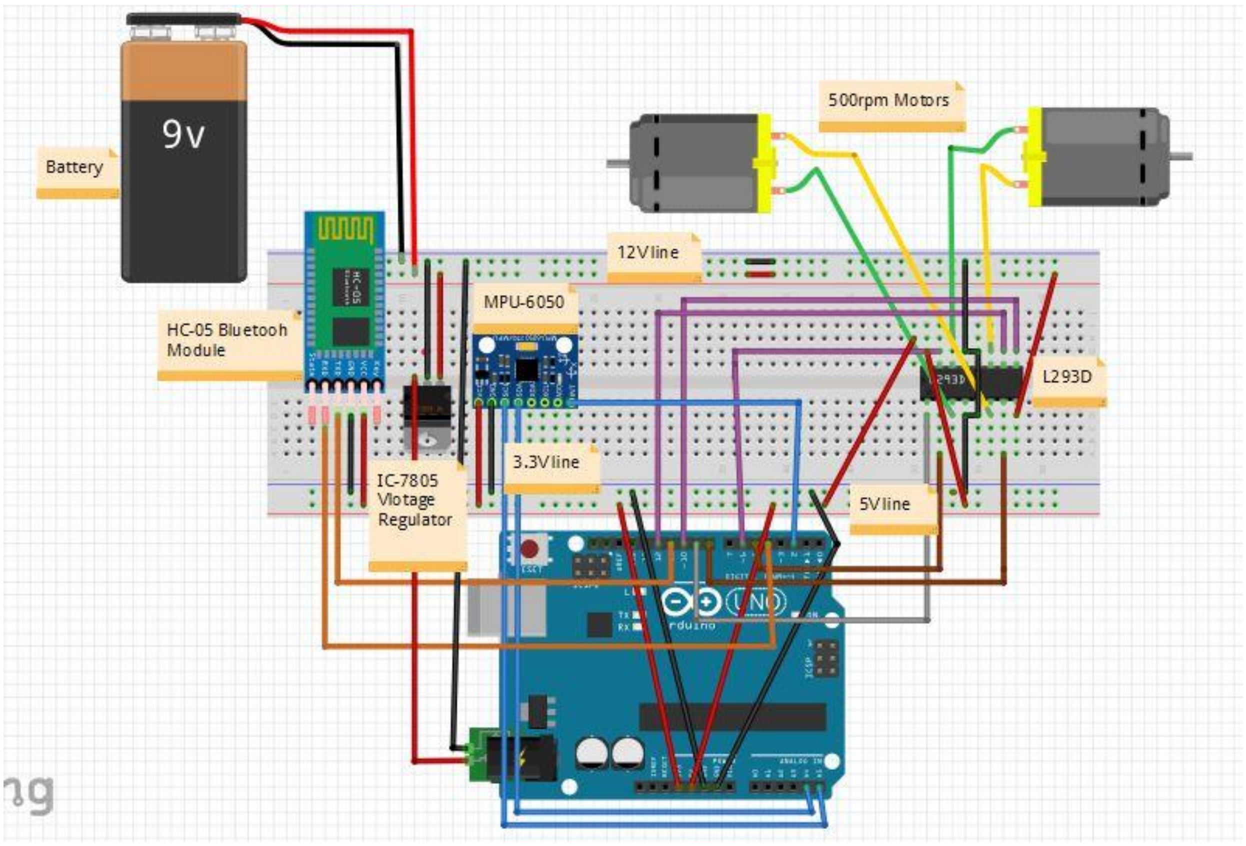


FIGURE 13: Block Diagram

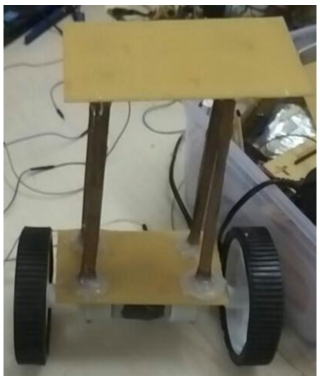


FIGURE 14: Chassis

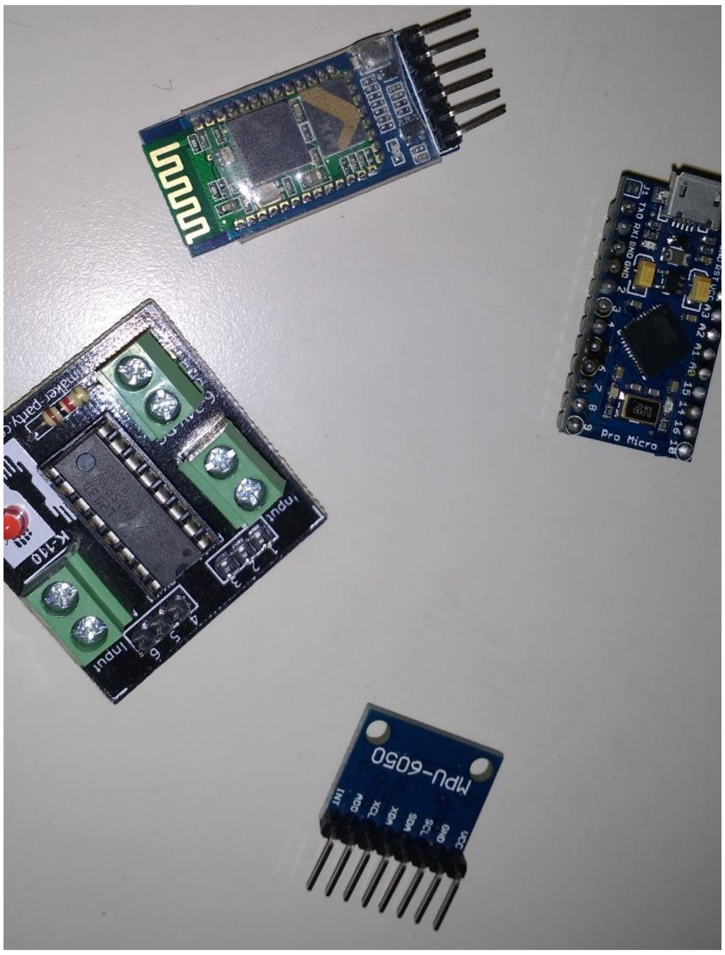


FIGURE 15: Components

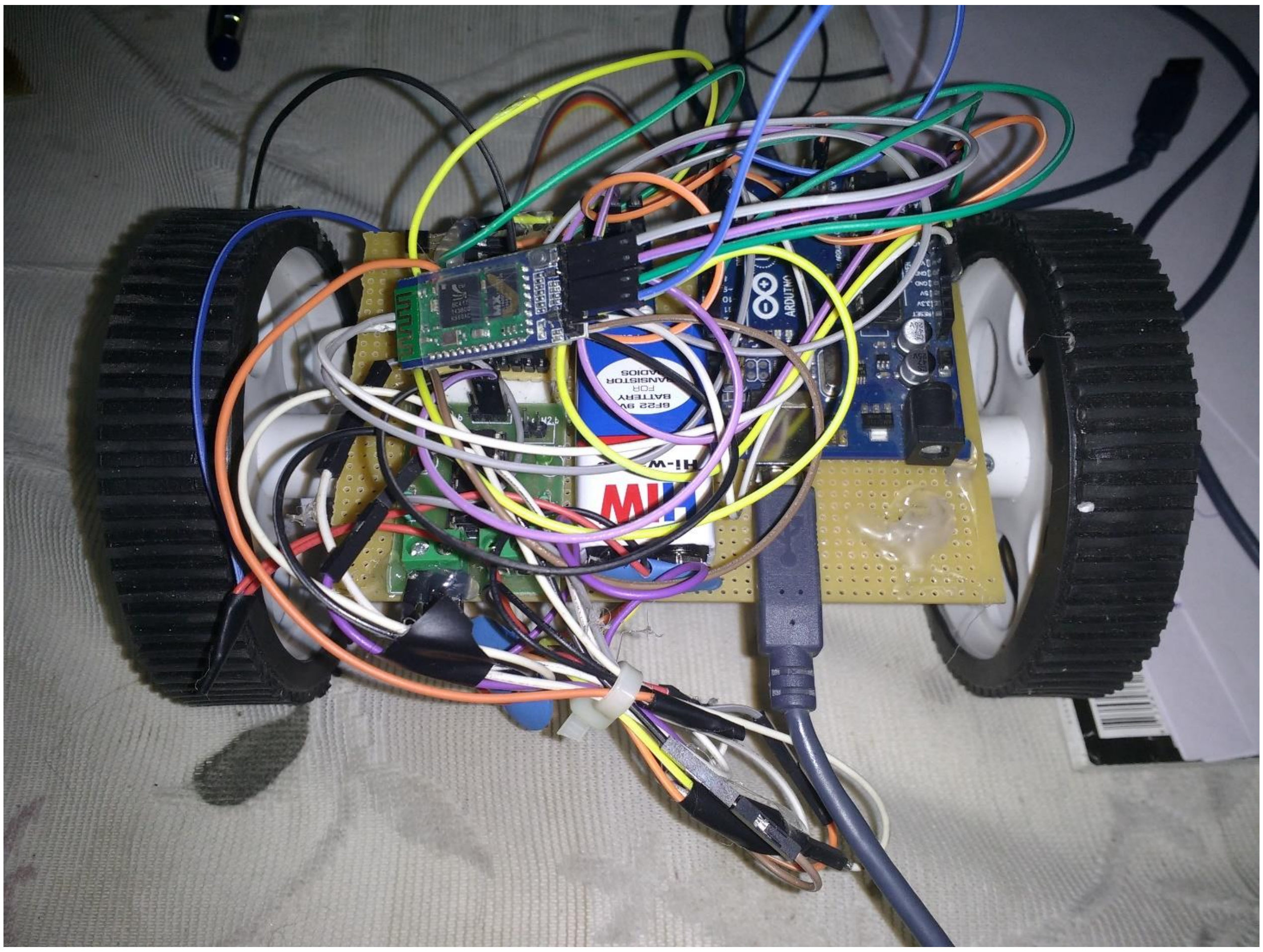


FIGURE 16: Final Structure

CHAPTER 4

PERFORMANCE ANALYSIS

4.1 Algorithm to control Self Balancing Bot vertically upward

- Take the necessary header files and global variables to support the main software.
- Configure PID angle and speed of the motor as well as PID constants.
- Take the read values of the IMU sensor and store it in a matrix up to 100 samples Configure the start button, stop button and device configure button.
- Configure PID module to update it.

- Set digital pins, serial pins and PWM pins on the arduino board.
- Initialize PID speed, PID angle and Time action module.
- Update the IMU sensor and update the IMU matrix.
- Then we will face three conditions

  1) When debug-> a) calibrate the previous values and update the previous declared variables.

  2) When started->. calculate the PID angle and motor speed. B. According to the angel check motor speed regularly and update the IMU sensor. C. If steering is not applied then set the external offset to zero D. Else refer to SBB console program.

  3) When stopped-> do the power off. Fall automatically.

To simplify things a little bit, I take a simple assumption; the robot's movement should be confined on one axis (e.g. only move forward and backward) and thus both wheels will move at the same speed in the same direction. Under this assumption the mathematics become much simpler as we only need to worry about sensor readings on a single plane. If we want to allow the robot to move sidewise, then you will have to control each wheel independently. The general idea remains the same with a less complexity since the falling direction of the robot is still restricted to a single axis.

4.2) Blutooth Control Algorithm

- Control Analog Pin

```
 GET analogWrite DATA From ARDUDROID
If (ard_command==CMD_ANALOGWRITE)
analogWrite (pin_num,pin_value);
```

- Control Analog Pin

```
 GET DigitalWrite DATA From ARDUDROID
If (ard_command==CMD_DIGITALWRITE)
 If (pin_value==PIN HIGH)
pin_value=HIGH
Else return;
Set_digitalwrite(pin_num,pin_value);
```

4.3) Temperature sensing Module

```
Analog Read the temp pin
Convert the temperature to Celsius
If (temp==predefined value)
Display temperature
Else no display
```

CHAPTER 5

CONCLUSIONS

5.1 Conclusions

To build a self-balancing robot we first derived the system equation then check its real time response (both time and frequency). Then we designed a PID controller to control the close loop function. We checked the controllability and set the pole location.

Then by choosing the appropriate components we analyze their simulation successfully. The above test steps are successful, then we are near to build a self- balancing bot. I've spent quite some time trying to tune the system but with no success.

The stability of the self- balancing bot may be improved if you use a properly designed gearbox that is having negligible gear backlash. So by implementation all of these concepts and avoid the errors that we came across the self-balancing bot is completely build.

5.2 FUTURE SCOPE

1. Position recovery. For that we need encoders for measure the movement of the wheel and use it for bring the robot back to the initial position.

2. Control the movement of the robot, forward, backward and turning , that is easy, the only thing we have to do is change the angle we want the robots stay then the gravity will do his work and the robot will move in the direction of the angle, then we put the angle to zero again and the robot stops. For turning we have to put some offset in the motor velocity, for turning right we subtract the offset to the velocity in right wheel and sum it to the velocity of the left wheel.

3. Adding a WiFi shield to control it via internet.

4. Implementation with Raspberry Pi to allow the robot to use a camera.

5. Implementation of a camera and artificial vision for the robot.

6. Use a ball instead of wheels, so hard, but we'll try.

Two wheel self-balancing robot is also a development in the field of robotics. This two wheel self-balancing robot is actually based on the concept of Inverted pendulum theory. This type of robot has gained fame and interest among researchers and engineers because it utilizes such a control system that is used to stabilize an unstable system using efficient microcontrollers and sensors. Two-wheeled balancing robots can be used in several applications with different perspectives such as an intelligent gardener in agricultural fields, an autonomous trolley in hospitals, shopping malls, offices, airports, healthcare applications or an intelligent robot to guide blind or disable people. These types of robots can effectively work in non-uniform surfaces due to their balanced control system.

REFERENCES

- http://www.instructables.com/id/Chappie-Self-Balancing-Robot/

- http://www.allaboutcircuits.com/technical-articles/an-introduction-to-control-systems-designing-a-pid-controller-using-matlabs/
- http://www.codeproject.com/Articles/882552/CloudCooker-IoT-Temperature-Controller
- http://www.csimn.com/CSI_pages/PIDforDummies.html
- https://en.wikipedia.org/wiki/PID_controller
- http://www.ijarcsse.com/docs/papers/Volume_3/7_July2013/V3I7-0257.pdf
- http://www.ijaiem.org/Volume2Issue2/IJAIEM-2013-02-16-015.pdf
- http://www.academia.edu/11153860/Simulation_of_boiler_using_PID_controller_and_Automation_PLC_and_SCADA_A_Comprehensive_Study
- https://www.arduino.cc/en/Main/ArduinoBoardUno
- http://www.uniassignment.com/essay-samples/information-technology/two-wheel-self-balancing-robot-information-technology-essay.php
- Lei Guo , Hong Wang , “PID controller design for output PDFs of stochastic systems using linear matrix inequalities”, IEEE Transactions on Systems, Man, and Cybernetics, Part B 01/2005; 35:65-71. pp.65-71
- R.S.Meena,Vikas Kumawat(2011) .Controller Design For Servo Motor Using MATLAB .In Proceeding of National Conferences on Advances & Research in Electrical System Technology
- http://www.kerrywong.com/2012/03/08/a-self-balancing-robot-i/
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6565146&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6565146
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6021051&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6021051
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5420511&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5420511
- http://home.iitk.ac.in/~vipgupta/self-balancing-bot
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6885816&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6885816
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5668001&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5668001
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6568193&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6568193
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5968611&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5968611

- A. Warnasch, and A. Killen, Low cost, high G, micro electro-mechanical systems (MEMS), inertial measurements unit (IMU) program, IEEE Position Location and Navigation Symposium, 2002, pp. 299-305