
ISSN: 2321-2152
IJMECE
International Journal of modern
electronics and communication engineering
E-Mail
editor.ijmece@gmail.com
editor@ijmece.com
www.ijmece.com

SELF BALANCING ROBOT CONTROLLED BY BLUETOOTH MODULE

Mr. S. Samraj, Mr. K. Sedhuramalingam, Mrs. K. Devadharshini,Dr. D. Satyaraj

Assistant Professor [1,2,3,4]

ssamraj@actechnology.in, sedhuramalingam.k@actechnology.in,

devadharshini.k@actechnology.in, dsatyaraj@actechnology.in

Department of MCT, Arjun College of Technology, Thamaraikulam, Coimbatore-Pollachi Highway,

Coimbatore, Tamilnadu-642 120

ABSTRACT

Creating a bipedal robot is the goal. The two wheels will be connected by a single axle, which will then support a platform. On top of it, you'll see another platform. Without further support, the platform will collapse. This device's primary purpose is to keep the platform level and balanced by using distance sensors, such as a gyroscope sensor (Gyro MPU6050). For starters, the robot can stay upright on its two wheels by responding to platform inclines with signals sent by the microcontroller, an Arduino Nano in this example, which causes the motors to go ahead or backward, according to the direction and magnitude of the inclination. With the use of a Proportional-Integral Derivative (PID) controller and data from an accelerometer and gyroscope, the robot was able to maintain its balance. The robot's movement was controlled by stepper motors. Based on the inverted pendulum concept, a two-wheeled self-balancing robot may be constructed by determining the force applied, F, and the angle from the equilibrium, χ. The motors will provide torque that turns the wheels in the same direction as the tilt when an imbalance occurs. To keep everything in its proper place, the wheels will travel the same distance as the centre of gravity. Raising the angle set point shifts the equilibrium point, allowing for forward motion. Making a robot with a human-like skeleton allows it to balance itself. More conventional robots had four wheels, could be easily balanced, and were larger in stature. The difference between a self-balancing robot and a conventional robot is that the latter employs two wheels and motors for mobility, while the former uses four. The Segway is a well-known example of a self-balancing robot in action. A "human transporter" in its own right, Segway has been selling like hotcakes since 2011. Shorter distances are its primary application.

The system's inherent instability has also made it an attractive study subject. The inverted pendulum idea is the basis of the two-wheel self-balancing robot. There are a lot of real-world uses for inverted pendulums, including humanoid walking robots, missile launchers, and earthquake-proof construction, among many others. For the last few years, researchers have focused heavily on developing control systems for two-wheel self-balancing robots. The fundamental reason for this is because its dynamics are not linear. Vehicles, spacecraft, robots, missiles, and missile systems all used it as a proving ground.

Keywords: Robot, Gyroscopesensor, ArduinoNano, Blue Tooth Module.

INTRODUCTION

Students, robotics junkies, and enthusiasts all around the globe are fascinated with self-balancing robots. What makes it interesting is that it is inherently unstable. This project is an effort to create a robot that can balance itself without human intervention. An important part of keeping the robot upright is estimating the tilt angle. This is accomplished by fusing gyroscope and accelerometer data using the Kalman Filter, which has been developed and evaluated. Furthermore, the process used to choose and assemble the hardware has also been validated. Additionally, the software development process and the difficulties encountered during the Kalman Filter's implementation have been detailed. The creation of a Bluetooth-controlled self-balancing robot is detailed in the project report. Putting the robot's software and hardware into action, includes picking out the right parts and programming the microcontroller. A PID controller regulates the robot's motor speeds and ensures it stays balanced, while an accelerometer and gyroscope determine its direction. Orders are received by means of the Bluetooth module.

direct the robot's motion using a handheld device. In addition to testing findings and recommendations for future enhancements, the paper provides comprehensive descriptions of the design and implementation processes. Using Bluetooth technology to remotely operate self-balancing robots is a promising and feasible idea, as shown in this research.

LITERATURE REVIEW

Students, robotics junkies, and enthusiasts all around the globe are fascinated with self-balancing robots. What makes it interesting is that it is inherently unstable. This project is an effort to create a robot that can balance itself without human intervention. An important part of keeping the robot upright is estimating the tilt angle. This is accomplished by fusing gyroscope and accelerometer data using the Kalman Filter, which has been developed and evaluated. Furthermore, the process used to choose and assemble the hardware has also been validated. Additionally, the software development process and the difficulties encountered during the Kalman Filter's implementation have been detailed. The creation of a Bluetooth-controlled self-balancing robot is detailed in the project report. Putting the robot's software and hardware into action, includes picking out the right parts and programming the microcontroller. A PID controller regulates the robot's motor speeds and ensures it stays balanced, while an accelerometer and gyroscope determine its direction. Orders are received by means of the Bluetooth module.

direct the robot's motion using a handheld device. In addition to testing findings and recommendations for future enhancements, the paper provides comprehensive descriptions of the design and implementation processes. Using Bluetooth technology to remotely operate self-balancing robots is a promising and feasible idea, as shown in this research.

A two-wheeled, self-balancing robot for control teaching driven by stepper motors was created at the University of Seville and is presented in this work by J.A. Borja et al. [2020]. Students have been building and controlling this robot for the last five years as a method of learning electronics, computer programming, modelling, control, and signal processing; this design improves upon an earlier model based on DC motors. The new design lowers the device's overall price while also improving its performance. CORTEZ et al., 2017 A cheap, self-balancing robot built for control teaching at the University of Seville using the Arduino platform. Students may get a control education in electronics at the University of Seville, which is the core aim. Building and operating this robot is based on the premise that students may gain knowledge in electronics, computer programming, modelling, control, and signal processing. A multivariable unstable nonlinear system with a non-minimal phase zero is the resultant model. Experimental

findings that were acquired by students at the University of Seville are used to illustrate potential prototypes. In this study, we provide an instructional control system for a low-cost, experimental self-balancing robot that is built on the Arduino platform. Both Croche (2014) and Gonzalez (2016) completed their diploma theses on this system, but students in future control and robotics classes will be able to construct and operate the suggested robot.

OBJECTIVES

Stability: When in motion, a self-balancing robot's principal goal is to keep its balance. This is achieved by constantly modifying the robot's centre of gravity based on data received from sensors like gyroscopes and accelerometers.
The robot may be remotely controlled using a smartphone or any other device that supports Bluetooth technology thanks to its wireless control feature. This has the potential to be more convenient and flexible than a wired control system.
Accuracy and Stability: A robot that can balance itself should be able to move with great accuracy and stability. For the robot to go in the intended direction and at the specified speed, it needs precise control algorithms and reliable data from its sensors.

capacity to Manoeuvre: A self-balancing robot's manoeuvrability is crucial for its capacity to negotiate complicated settings and confined places, as determined by its unique application. Swiftness, dexterity, and quickness are all need for this.

An essential goal for any robot should be to provide a good user experience. This encompasses aspects like user-friendliness, dependability, and the capacity to react to human input. Bluetooth control can provide a user-friendly interface for controlling the robot and receivingfeedback on its status.

Hardware Components

GYROSCOPE SENSOR

To find out how fast something is spinning around, you may use a gyroscope sensor. In all three dimensions relative to gravity, the 3-axis gyroscope sensor can determine the person's orientation and rotation. The location of the individual during the fall might be indicated by the angle value that is provided. The human body's acceleration and motion may be detected using an accelerometer sensor. A triaxial accelerometer determines the acceleration along the x, y, and z axes, in that order. In order to measure the user's motion, the accelerometer sensor gives a parameter value. The purpose of both of these sensors is to detect falls.

With a precision of 13 bits and a measuring range of up to ±16 g, the ADXL345 is a tiny, thin, ultralow power, 3-axis accelerometer. Through an SPI (3-or 4-wire) or I2C digital interface, the digital output data may be accessed in a 16-bit twos complement format. When it comes to mobile devices, the ADXL345 shines. For use in tilt-sensing applications, it monitors both the static acceleration of gravity and the dynamic acceleration caused by shock or motion. With its 3.9 mg/LSB resolution, it can monitor inclination changes less than 1.0°. A number of specialised sensing capabilities are offered. In order to determine whether motion is there or not, activity and inactivity sensors compare the acceleration along any axis with user-defined limits. One or two taps in either direction may be detected via tap sensing. If the gadget is tumbling, the freefall sensor will pick it up. You may assign each of these functions to one of the two interrupt output pins separately. To reduce overall system power consumption and host processor activity, data may be stored using an integrated memory management system with a 32-level first in, first out (FIFO) buffer. Intelligent motion-based power management is made possible by active acceleration measurement and threshold detection in low power modes, which dissipate very little power.

WHERE TO USE HC-05 BLUETOOTH MODULE

One well-liked module that may provide your projects full-duplex wireless capabilities is the HC-05. This module may connect to any device that has Bluetooth capabilities, such as a phone or laptop, or it can communicate between two microcontrollers, such as an Arduino. A lot of useful apps are already available on Android, which streamlines the procedure. Assisting the module in communicating are

Connecting to any microcontroller that supports USART is a breeze because of the 9600 baud rate USART. In addition, the command mode allows us to change the module's default settings. This wireless module might be a good fit if you need to send data from a computer or mobile device to a microcontroller or vice versa. Nevertheless, this module is not designed to transport multimedia files. Mode that allows you to adjust the default settings of your device. According to the pin description, we may use the key pin to switch between these two modes of operation. The HC-05 module's operation via the Serial Port Protocol (SPP) makes pairing it with microcontrollers a breeze. Connect the module's Rx pin to the Tx of the MCU and the Tx pin to the Rx of the MCU, as indicated in the picture below, and then power the module using +5V.

HOW TO US THE HC-05 BLUETOOTH MODULE

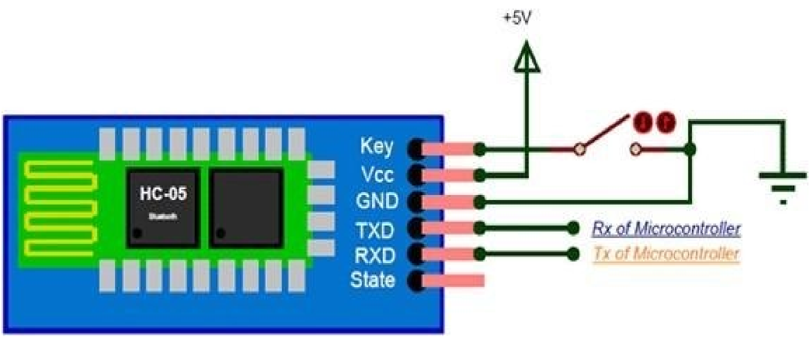

Fig. 2 HC5 Bluetooth Module.

MOTOR DRIVER

Motor driver ICs are inexpensive, easy-to-implement design elements that speed up the whole circuit design process. It is possible to choose the drivers according to the motor ratings, which include things like voltage and current. The most widely used motor drivers, such as ULN2003, are not based on H-bridges. The stepper motor may be driven using it. A Darlington pair, integrated within this driver, is capable of withstanding voltages and currents up to 50VDC and 500mA, respectively. The circuit for driving the stepper motor is shown below.

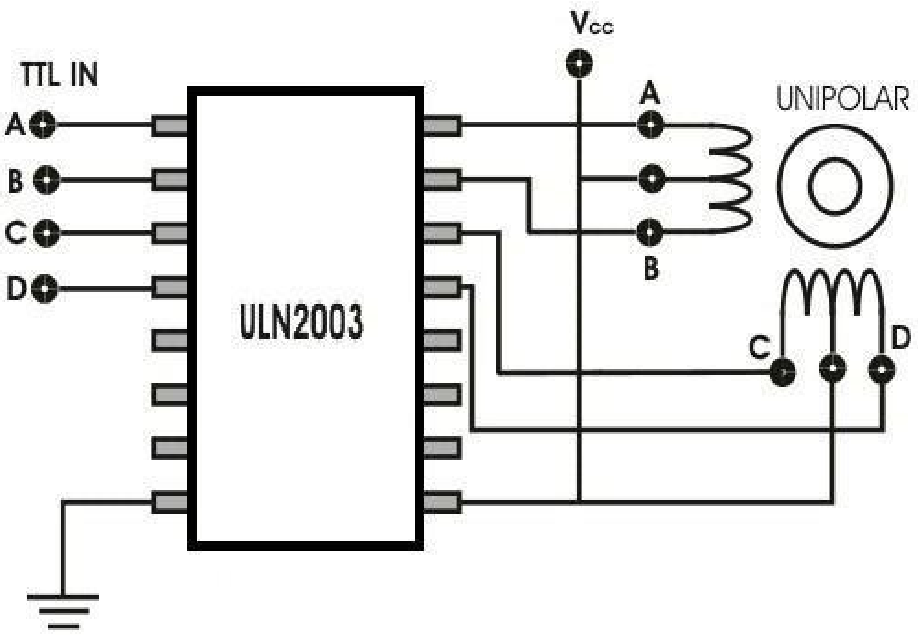

Fig. 3 Motor-driver-circuit-diagram.

SOFTWARE IMPLEMENTATION

SOFTWARE REQUIREMENTS

Arduino 1.0.6 software tools used to program microcontroller. The working ofsoftware tool is explained below in detail.

PROGRAMMING MICROCONTROLLER

The generation time is reduced by a compiler that is designed for an abnormal state dialect. A microcontroller called an Arduino UNO may be programmed using an Arduino. The computer applications are written only in the C dialect that is installed. For microcontrollers supported by Windows, there is a set of tools called Arduino that can be run as an executable and is open-source. If you want to control more of the physical world than what your desktop PC can handle, you need an Arduino. Built on top of a simple microcontroller board, it's an open-source physical registration stage with a dedicated enhancement domain for writing board-specific code. The limited amount of resources that a developer must oversee is one of the difficulties of working with microcontrollers in programming. When compared to microcontrollers, PC assets, such as RAM and processing speed, have almost unlimited potential. In contrast, microcontroller code should take up as few resources as is practically possible.

ABOUT ARDUINO COMPILER

Arduino compiler is an open-source software development environment used for writing, compiling, and uploading code to Arduino microcontrollers. The Arduino IDE (Integrated Development Environment) provides a simple platform for developing and testing code, making it easier for users to create and implement projects.The Arduino compiler is basedon the C++ programming language and comes with a set of libraries and functions that allow users to interact with the hardware.

GET AN ARDUINO BOARD AND USB CABLE

You additionally require a standard USB link (An attachment to B plug): the kind you would associate with a USB printer, for instance. (For the Arduino Nano, you'll require an A to Mini-B link.)

Fig. 4 Arduino board and USB cable

CONNECT THE BOARD

As a result, the Arduino Uno, Mega, Duemilanove, and Nano all get their power from an external power source or the USB connection to the computer. If you're using an Arduino Diecimila, make sure the board can receive control over the USB connection.
A jumper, a little plastic piece that snaps into two of the three sticks between the power and USB connectors, allows the user to choose the power source. Keep an eye out for the two sticks that are closest to the USB port; they contain it. Connect the Arduino board to your own computer by means of the USB cable. The verdant power LED (named PWR) ought to goon.

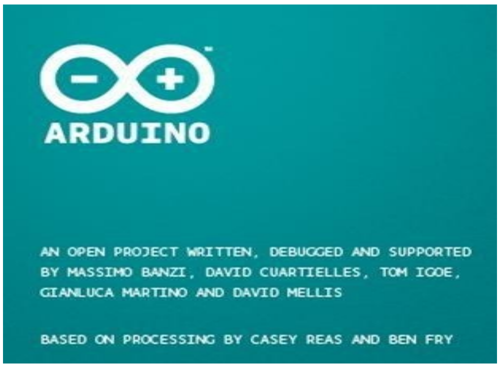

Fig. 5 Connect the board.

SOURCE CODE

```
#include <Wire.h>

#include <Adafruit_MotorShield.h>

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX, TX pins for Bluetooth module

Adafruit_MotorShield AFMS = Adafruit_MotorShield();

Adafruit_DCMotor *leftMotor = AFMS.getMotor(2);

Adafruit_DCMotor *rightMotor = AFMS.getMotor(1);

int16_t accel_x, accel_y, accel_z, gyro_x, gyro_y, gyro_z;

float angle = 0.0, error = 0.0, previous_error = 0.0, Kp = 15.0, Kd = 0.0, Ki = 0.0;

float pid_value = 0.0, integral = 0.0, derivative = 0.0;

long last_time = 0;

void setup() {

AFMS.begin();

leftMotor->setSpeed(0);

rightMotor->setSpeed(0);

leftMotor->run(RELEASE);

rightMotor->run(RELEASE);

Serial.begin(115200);

BTSerial.begin(9600); // Set baud rate for Bluetooth communication

Wire.begin();

Wire.beginTransmission(0x68);
```

```
Wire.write(0x6B);

Wire.endTransmission(true);

}

void loop() {

// Read accelerometer and gyro values

Wire.beginTransmission(0x68);

Wire.write(0x3B);

Wire.endTransmission(false);

Wire.requestFrom(0x68, 14, true);

accel_x = (Wire.read() << 8 | Wire.read());

accel_y = (Wire.read() << 8 | Wire.read());

accel_z = (Wire.read() << 8 | Wire.read());

gyro_x = (Wire.read() << 8 | Wire.read());

gyro_y = (Wire.read() << 8 | Wire.read());

gyro_z = (Wire.read() << 8 | Wire.read());

// Calculate angle using complementary filter angle = 0.96 * (angle + gyro_x * 0.0000611) + 0.04 * (atan2(-accel_y, -accel_z) *57.2958); // Calculate PID value

error = angle;

integral = integral + error;

derivative = (error – previous_error) / (millis() – last_time);

pid_value = Kp * error + Ki * integral + Kd * derivative;
```

```
// Update motor speeds leftMotor-

>run(FORWARD); rightMotor-

>run(FORWARD);if (pid_value >

0) {

leftMotor->setSpeed(abs(pid_value));

rightMotor->setSpeed(abs(pid_value) – 30);

} else {

leftMotor->setSpeed(abs(pid_value) – 30);


rightMotor->setSpeed(abs(pid_value));

}
// Send sensor and motor data over Bluetooth

BTSerial.print(angle);

BTSerial.print(“,”); BTSerial.print(pid_value); BTSerial.print(“,”); BTSerial.print(leftMotor-
>getSpeed()); BTSerial.print(“,”); BTSerial.print(rightMotor->getSpeed());


BTSerial.println();

// Update previous error and last time for derivative term

previous_error = error;

last_time = millis();
}
```

Experiment results

We created and tested the self-balancing robot, and it passed. By taking precise readings from the robot's gyroscopes, the Arduino was able to maintain the robot's equilibrium by adjusting the motor speed. The user may command the robot from their mobile device thanks to the Bluetooth module's ability to send control signals wirelessly.

The experiment involving the self-balancing robot that used a Bluetooth module was finished with promising outcomes. A combination of the robot's gyroscope sensors and stepper motors allowed it to maintain its equilibrium. The robot may be controlled wirelessly from a mobile device thanks to the Bluetooth module. On the basis of these findings, we may have the following conversations:

the report on the project:

Placing the self-balancing robot on a level surface did not affect its ability to balance. The robot's gyroscopes provided precise angle readings, which the Arduino microcontroller used to fine-tune the motor speed and maintain stability.

Control using wireless means: the Bluetooth module was effectively coupled with the Arduino microcontroller, allowing for the transmission of control signals wirelessly from a mobile device. Because of this, the user may operate the robot remotely, cutting eliminating the requirement for cords and connections.

It didn't matter whether the robot was pushed or bumped; it could keep its equilibrium anyway. In response to unexpected obstacles, the Arduino's control system swiftly adjusted the motor speed, allowing the robot to maintain its equilibrium.

The project showcased the effective use of a Bluetooth module to create a self-balancing robot. The autonomous robot could maintain its equilibrium and react wirelessly to commands from the operator. There are still opportunities for further study and enhancements, and the project's findings were encouraging despite its limitations.

CONCLUSION

There are a lot of possible uses for a self-balancing robot that is controlled via Bluetooth. In settings where steadiness is paramount, the robot may be useful since it can keep its equilibrium even while moving. Wireless control, made possible by Bluetooth technology, offers more freedom and convenience than a wired control system. The robot can't maintain its stability and accuracy without precise sensor data and algorithms for control. As a result, it can be directed and moved at the specified speed, making it an invaluable tool for jobs requiring pinpoint accuracy. The robot's manoeuvrability is also critical for its ability to handle confined places or complicated surroundings. One of the main goals of a self-balancing robot should be to improve the user experience. A simple way to interact with the robot and see its current state is via the Bluetooth control. This will make sure that users can get easy control of the robot and get all the information they need to get their duties done quickly and effectively. In sum, a self-balancing robot that can be operated over Bluetooth might prove to be an invaluable asset in a wide range of contexts, from personal to industrial. An successful solution for numerous jobs may be achieved by satisfying the goals of balance, wireless control, stability and accuracy, manoeuvrability, and a user-friendly experience.

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’,
7. Steve Hassenplug, 2002, ‘Steve’s Legway’,
8. Dean Kamen ,2001,
9. John Green, David Krakauer, March 2003, New Imems Angular Rate Sensing Gyroscope
10. Peter Hemsley, 32-bit signed integer maths for PICS
11. Mosfets and Mosfet’s drivers, power management with threshold sensing andactive acceleration measurement at extremely low power dissipation.