TWO WHEELED SELF BALANCING VEHICLE

Shubham
CSE-3rd SEM
Dr SPM IIIT NAYA RAIPUR
Raipur, India
shubham16100@iiitnr.edu.in

Nirbhay Kumar Giri
CSE-3rd SEM
Dr SPM IIIT NAYA RAIPUR
Raipur, India
nirbhay16100@iiitnr.edu.in

Saurabh Kumar Singh
CSE-3rd SEM
Dr SPM IIIT NAYA RAIPUR
Raipur, India
saurabh16100@iiitnr.edu.in

Abstract— This paper reports the design, construction and control of a two-wheel self-balancing vehicle. The system architecture comprises a pair of DC motor and an Arduino micro-controller board; a triple-axis gyroscope and a 3-axis accelerometer are employed for attitude determination. In addition, a complementary filter is implemented to compensate for gyro drifts. Electrical and kinematic parameters are determined experimentally; PID and LQR-based PI-PD control designs, respectively, are performed on the linearised equations of motion. Experimental results show that self-balancing can be achieved with PI-PD control in the vicinity of the upright position.

I. INTRODUCTION

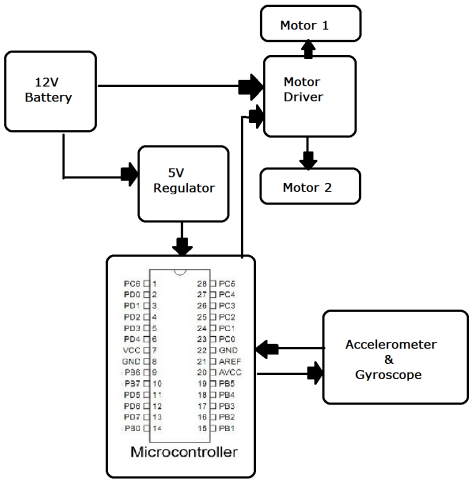
In the past decade, mobile robots have stepped out of the military and industrial settings, and entered civilian and personal spaces such as hospitals, schools and ordinary homes. While many of these robots for civil applications are mechanically stable, such as "Aibo" the Sony robotic dog, or four-wheel vacuum cleaners, one that ordinary on-lookers would find awe-inspiring is the Segway personal transport, a mechanically unstable, two-wheel self-balancing vehicle that has seen deployment for law-enforcement, tourism, etc. This vehicle can be rightfully called a robot because, without the sensory capability and intelligent control that accompany every robot, the Segway can never stay upright.

While Segway may have been a well-known commercial product, research into the control of such a mechanical system has been diverse. A two-wheel self-balancing robot is very similar to the inverted pendulum, which is an important testbed in control education and research. In this paper, we report a student project on the design, construction and control of a two-wheel self-balancing robot. The robot is driven by two DC motors, and is equipped with an Arduino UNO board which is based on the ATmega328 processor, an MPU6050(a triple axis accelerometer and triple axis gyroscope) sensor . To compensate for gyro drifts common in COTS sensors, a complementary filter is implemented; for a single-axis problem such as this balancing robot, the complementary filter approach is significantly simpler than the Kalman filter. Two control designs based on the linearised equations of motion is adopted for this project: a proportional-integral-differential (PID) control, and a proportional-integral proportional-differential control based on linear-quadratic regulator (LQR) design. The approach is found to be robust to modelling errors which can be incurred during experimental determination of such electrical and kinematic parameters as moments of inertia and motor gains. Simulation and experimental results are presented, which show that stability of the upright position is achieved with PI-PD control within small tilt angles.

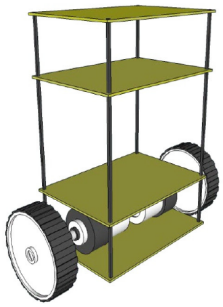
II. SYSTEM DESIGN

The structure of a self-balancing robot can be classified into three parts: sensors, motor and motor control, and develop board.This section introduces the application and advantage of the sensors on the proposed balancing robot, and how these sensors are employed to obtain measurements of acceleration, distance traveled, and robot tilt angle.

BLOCK DIAGRAM



3d- frame of Chassis



A. Selection and Application of Sensors

To balance the robot, data of the robot's tilt angle and angular velocity are needed. Using MEMS sensors including a gyroscope, an accelerometer and wheel-angle encoders, we can measure all the data needed for balancing control.

1) Gyroscope: The gyroscope is the sensor which can measure the angular velocity of the balancing robot, and send the data to the development board. The present robot uses the MPU6050 three axis Gyroscope, which sends data via serial communication to the develop board, and has the advantage of three-axis angular measurement (although only one axis is used), low-power consumption, and low- cost. Theoretically, integrating the angular velocity will yield the angle θ directly; however, this process will also integrate noise in the gyroscope measurement. As a result, the value of θ will diverge. Section III-A will explain compensation of angular drift using the complementary filter approach.

2) Accelerometer: The accelerometer measures the total external acceleration of the balancing robot, which includes the gravitational and motion accelerations. In this project, we choose the MPU6050 which features properties such as three-axis sensing, output-voltage signal conditioning, and low cost. Following the direction cosine method, one can use the X - and Z -axis gravitational acceleration measurements to calculate the tilt angle θ. Although this method gives θ quickly, it can be easily influenced by external forces and noise.

3) *Motor Driver(L293d):* A motor driver is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver act as an interface between Arduino and the motors . The most commonly used motor driver IC's are from the L293 series such as L293D, L293NE, etc. These ICs are designed to control 2 DC motors simultaneously. L293D consist of two H-bridge. H-bridge is the simplest circuit for controlling a low current rated motor.

4) *Bluetooth Module(HC-05):* HC−05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication.This 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 AFH (Adaptive Frequency Hopping Feature).

5) *Arduino UNO Board:* The Arduino Uno is a micro-controller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the micro-controller; simply connect it to a computer (or appropriate wall power adapter) with a USB cable or power it with a AC-to-DC adapter or battery to get started.The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features an ATmega16U2 programmed as a USB-to-serial converter. This auxiliary micro-controller has its own USB boot-loader, which allows advanced users to reprogram it.

6) ARDUINO SOFTWARE 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 International Journal of Engineering Science and Computing, April 2017 6462 http://ijesc.org/ 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. 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.

1) Performance: The self-balancing robot needs almost real-time response to estimate and correct its tilt angle. Hence, the development board must provide a processing speed that is sufficiently fast to perform the processing tasks, including data acquisition, control computation and signal output, within the sampling time. Based on preliminary calculations, a sampling time smaller than 0.05 seconds is required. The Arduino Mega development board is equipped with the ATmega2560 processor, which features a maximum clock rate of 16 MHz. In our implementation, a sampling time of 0.01 seconds can be achieved with a 30% to 40% buffer.

2) I/O Pins: Another issue is the number of I/O pins available. On the robot, sensors are deployed to obtain measurements of its motion: a gyroscope and an accelerometer are used to estimate the tilt angle, encoders are used to obtained odometric measurements. In addition, a motor control board is interfaced with the development board for delivery of PWM signals. Base on the pin-outs of these sensors and the motor control board, it has been determined that at least thirty I/O

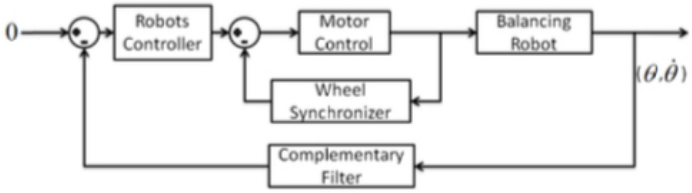
pins are needed. The Arduino Mega features 54 general-purpose digital I/O pins of which 15 provide PWM output. Some of the digital I/O pins also support serial communication such as I2C and SPI, as well as interrupt handles. The board also features sixteen 10-bit analog inputs for 0-5V input, giving a quantisation limit of 4.9mV.

3) Open Source: To alleviate difficulties in programming, a user-friendly development environment, useful function libraries and references are preferred. These requirements are well met by the Arduino development environment which is based on the C language. User-contributed function libraries like PWM control, I2C and SPI communication reduce difficulties in learning to program the Arduino boards. Most importantly, Arduino is open-source with a large user community and up-to-date discussion forums. This allows students to study other users' codes, compare results, and make modifications according to the project's needs.

4) Price and Expansions: Arduino boards are low-cost and expandable, where optional peripherals called shields can be purchased as and when needed. The accessibility of these products in terms of price versus functionality makes them ideal solutions for academic and student projects.

B. Power Supply : For power supply, the motors need a voltage between 12V to 16V, and the development board needs between 5V to 15V. Hence, two sets of batteries are incorporated: for the motors we select a 14.8 volt lithium battery, and for the development board, we choose a four-cell (4×1.5 volts) Ni-MH battery pack.

Architecture of Control System



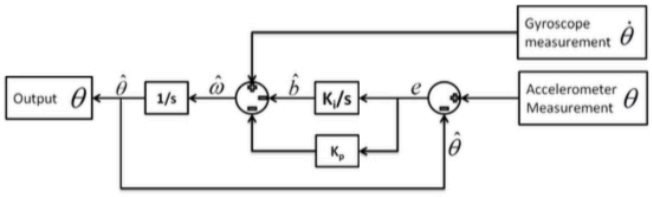
C. ANGLE ESTIMATION AND BALANCING CONTROL

The system architecture of the self-balancing robot is as shown in Fig. 2, where the forward loop comprises the robot, a balancing controller which delivers a motor-control signal, and a inner-loop controller for wheel synchronisation. Feedback is provided through a complementary filter whose function is to provide an estimate of the robot tilt angle from gyroscope and accelerometer measurements. In the following sections, we shall first describe the complementary filter, followed by the wheel synchronization controller, and finally the balancing controller.

A. Angle Estimation via Complementary Filter

In Section II, it has been seen that a gyroscope is used to measure the angular velocity of the robot, whereas an accelerometer measures the Y - and Z-components of gravitational acceleration, and encoders measure the distance travelled by the wheels. In the balancing control, the tilt angle is corrected to the upright position by using wheel motion. This requires a measurement of the tilt angle θ which must be accurate. The easiest way to obtain the angle is to use the gyroscope; since the gyroscope provides the angular velocity, integrating the latter will produce the angle. However, the gyroscope measurement contains noise. Hence the previous method will not only integrate the angular velocity, but also the noise. This will make the integrated data diverge from the correct angle. Another method is to apply the direction- cosine method to the gravity components measured by the accelerometer to obtain θ immediately; however, this is also affected by noise and the robot's accelerating motion.

To solve the problem of angle measurement, we design a complementary filter [20], a block diagram of which is shown in Fig. 3. The complementary filter uses the measurements of the gyroscope and accelerometer, $\dot{\theta}$ and θ respectively, as input to estimate the bias in the gyroscope-measured angular velocity. Subtracting the estimated bias $\hat{b}$ from $\dot{\theta}$ yields the estimated angular velocity $\hat{\omega}$; integrating $\hat{\omega}$ provides the estimated angle $\hat{\theta}$.



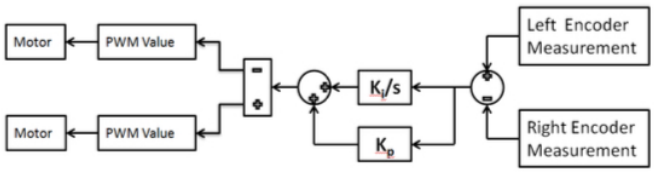
θ = measured angle of accelerometer
$\dot{\theta}$ = measured angular velocity of gyroscope
$\hat{\theta}$ = estimated angle of balancing robot
e = estimation error between $\hat{\theta}$ and θ measurement
$\hat{b}$ = estimated gyro bias
$\hat{\omega}$ = estimated angular velocity
k_p = proportional gain
k_i = integral gain

B. Inner-loop Control for Wheel Synchronization

Ideally, if the same motor control signal is sent to the motors control board, both motors should rotate at the same speed. But in the real environment there are many reasons for which the motors rotate at different velocities under the same input signal, such as defects of the motors, terrain and hindrance on the ground. Thus, a method to synchronise both motors is needed. To solve this problem we design a wheel synchronisation controller consisting of a simple PI controller with the block diagram given in Fig. 5. This controller adjusts the PWM inputs to the motors so that the difference

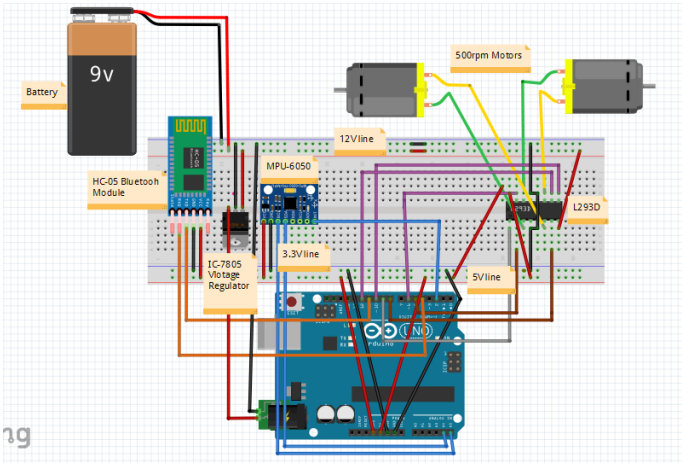
between the left and right encoders tracks zero. Fig. 6 shows the experimental results of wheel travel distances with and without wheel synchronization, and Fig. 7 shows the motor control signals under wheel synchronization. It is clear that wheel synchronization is effective.

Block Diagram of Wheel Synchroniser

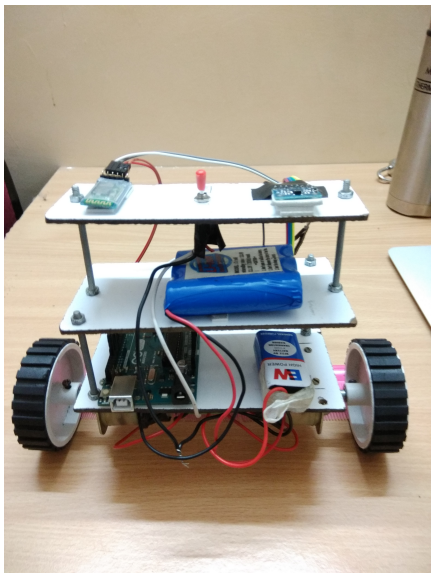


III. EXPERIMENTAL PROTOTYPE

PIN DIAGRAM



FINAL PROJECT MODEL



IV. CONCLUSION

We have constructed a two-wheel self-balancing robot using low-cost components, and implemented a tilt-angle estimator and a stabilising PI-PD controller for the balancing motion. The following are some future goals for improvement.

1) Design improvement: Design improvement for better balancing motion will require optimisation of the mechanical design such as relocating the centre of mass, better sensor placement, modification of the complementary filter to reject disturbances due to translational motion, and a robust control design.

2) Remote control: Low-cost communicate hardware such Xbee may be added to the robot, so that the user can control its motion remotely and read the data via telemetry. Also, user-controlled motion such as forward, backward, turn right and left, may be implemented.

3) Obstacle avoidance and perimeter following: A possible extension is to combine the Arduino system and other sensors such ultrasonic and IR senses, GPS, digital compass and Camera to address other advanced applications, e.g. obstacle avoidance and perimeter following.

ACKNOWLEDGMENT

All the work mentioned above was possible under the guidance of our Professor Dr. Debanjan Das sir and other lab assistants who provided necessary resources and facilities along with helping us to achieve our goal. All the group members would like to sincerely acknowledge their help for the same.

REFERENCES

[1] https://www.electronicshub.org/dc-motor-control-arduino/

[2] http://wired.chillibasket.com/2015/01/calibrating-mpu6050/

[3] https://playground.arduino.cc/Main/MPU-6050

[4] https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/

[5] http://www.instructables.com/id/GY-521-MPU6050-3-Axis-Gyroscope-and-Accelerometer-/