Design & Development of self-balancing single wheeled vehicle

A PROJECT REPORT

Submitted in partial fulfillment of the
requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in
MECHANICAL ENGINEERING

Submitted by:
ASHUTOSH GAUTAM (140003009)
AMAN DHANAWAT (140003004)
AYUSH BAGHMAR (140003011)

Guided by:
Dr. S.I. Kundalwal (Assistant Professor, IIT INDORE)
Dr. Indrasen Singh (Assistant Professor, IIT INDORE)

INDIAN INSTITUTE OF TECHNOLOGY INDORE
December 2017

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Design & Development of self-balancing single wheeled vehicle"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'MECHANICAL ENGINEERING' completed under the supervision of **Dr. S.I. Kundalwal (Assistant Professor) and Dr. Indrasen Singh (Assistant Professor),** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Signature and name of the students with date

CERTIFICATE by BTP Guides

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

Signature of BTP Guides with dates and their designation

Preface

This report on "Design & Development of self-balancing single wheeled vehicle" is prepared under the guidance of Dr. S.I. Kundalwal(Assistant Professor) and Dr. Indrasen Singh(Assistant Professor**)**.

Through this report we have tried to give a detailed description of the concept, design and making of a self balancing single wheeled vehicle. We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added 3-D designs and actual photographs of the structure for better understanding.

Ashutosh Gautam, Aman Dhanawat & Ayush Baghmar
B.Tech. IV Year
Discipline of MECHANICAL ENGINEERING
IIT Indore

Acknowledgements

We wish to thank Dr. S.I. Kundalwal and Dr. Indrasen Singh for their kind support and valuable guidance. We want to express our gratitude to Dr.M.Santhakumar for his help in "CONTROL SYSTEMS" which was a critical part of our project. Also we are grateful to the electrical lab and Central workshop for their technical guidance.

It is their help and support, due to which we became able to complete the design and technical report.

Ashutosh Gautam, Aman Dhanawat & Ayush Baghmar

B.Tech. IV Year

Discipline of Mechanical Engineering

IIT Indore

Abstract

When it comes to self-balancing personal transportation devices, it's look like the Solo wheel, Segway could all be in for a little competition. The rider controls the speed by leaning forwards or backwards, and steers by twisting the unit using their feet. The self-balancing mechanism uses gyroscopes, accelerometers for measuring the tilt and generates an output (motor speed and torque)accordingly. With our project we are trying to reduce the cost of these vehicles by using cheaper yet reliable materials.We have also tried to replace the in-hub transmission with a more common chain-sprocket system. This will make the vehicle not just cheap but also easy to repair and maintain for the user. Our aim is to first make this system balance using the most affordable electronic components available. This will give us an estimate of the minimum expenditure for a self balancing machine that can later be turned user friendly by adding more features including safety measures, seating etc.

Contents

List of Figures

Introduction

The rapid industrial growth in developing countries, like India, is giving birth to many new challenges in the form of air pollution, traffic jam and scarcity of parking places. In addition, saving energy to reduce the problem of fuel depletion is becoming increasingly important. To this end, many industries, manufacturing companies and educational institutions spread over huge areas are restricting the usage personal cars and bikes by their employees within campus. This has focused current research on developing eco-friendly transportation with minimal sizes.

Monobike, first time invented by by Chris Hoffman, the ceo of RYNO Motors, is a self-balancing electric vehicle with a size much smaller compared to other commercial vehicles for a single driver.This vehicle balances itself by moving forward or backward direction based on the angle of tilt. Monobike could be a very useful means of transport in large campuses like airports, universities, space centers and industries because of the small size and eco friendly nature.

Public transports are with no doubt a necessity to match today's mass transportation needs. But they can't satisfy the transportation needs of individuals completely. For example running errands or reaching a crowded location in the city where public transports can't reach. The concept of folding bicycles and other developments in the light weight vehicles that can be easily carried inside the public transports are very inspiring options for future. With our project we can try to achieve the same, by keeping the weight as low as possible.

Currently the self balancing(mostly segways) vehicles are being used in many foreign university and industrial campuses. Monobikes are still in their design and development phase, both inside and outside India. Although some companies (foreign companies)have started the production on a commercial scale, these vehicles are not popular with the common people. One of the major reasons is their high cost compared to the low applicability which makes them non feasible. Our attempt with this project is to make monobikes more affordable. Instead of fast cars our overcrowded cities need noiseless, light weight and pollution free vehicles. Development in monobikes will help making the Indian roads much safer and cleaner.

Design

Structure: The skeleton or framework of this bike has been designed to achieve required strength at minimum weight. As it can be seen in the 3D model below(Fig.2) ,the structure is just enough to keep the motor , batteries , electronic circuit and the rider keeping the vehicle light and compact. Inorder to choose the right material we did stress calculations for 120kg load. We found that 25mm×25mm cross section AISI 1018 square pipes were efficient, cheap and easily available in the market.

Transmission: Ryno bike as complicated it sounds and it doesn't get any cheaper as the estimated price of the mass manufactured will be 3500$ because of the fact that the main motor housing is assembled in the rim of the single-wheel which makes it complex and minute, which made it problematic for maintaining purposes and pricey contrary to the fact that it was meant to be used as personal transport. Our focus was to create a much simpler design that is cheap as well as reliable. So we replaced the design of assembling everything in the rim of wheel (motors, transmission) (Fig. 1) with a simple chain sprocket mechanism which is cuts our production cost and is more easy for maintenance. In which the wheel is driven by the motor with the help of chain and sprocket i.e. welded to the main shaft of the wheel (Fig. 2).

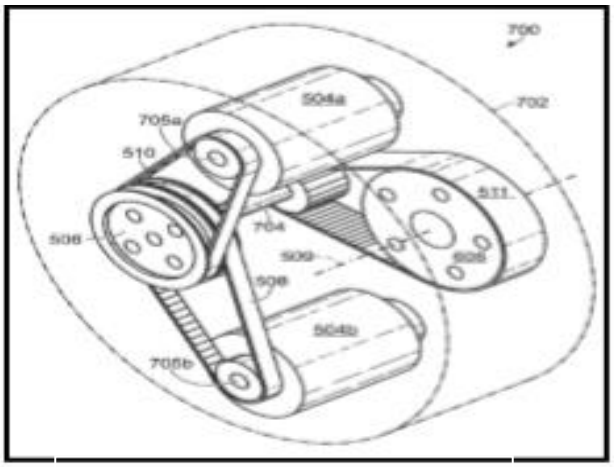

Fig. 1. Ryno Bike Driver assembly

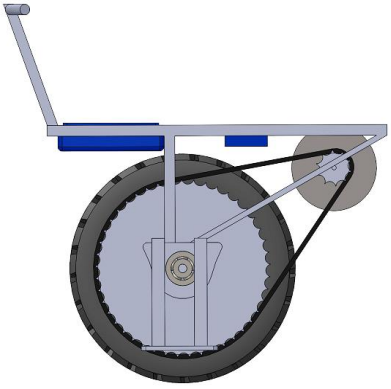

Fig. 2. Final project Design

Motor & Batteries: Also for powering this vehicle we used a 24V DC motor that gives a maximum output power of 350 Watts. This motor gives a maximum torque of about 55 N.m, which will be needed for balancing the mass of rider on the platform. Two ups batteries - 24V, 12Ah have been used to power the motor. Although these batteries are heavier in comparison to other available options like lipo(lithium polymer batteries) ,but are much more affordable.

Concept

Self-balancing vehicles of all kinds (single wheel or two lateral wheels) based on the concept of inverted pendulum. It is often implemented with the pivot point mounted on a cart that can move horizontally and may be called a cart and pole (Fig. 3). Most applications limit the pendulum to 1 degree of freedom by affixing the pole to an axis of rotation. Whereas a normal pendulum is stable when hanging downwards, an inverted pendulum is inherently unstable, and must be actively balanced in order to remain upright; this can be done either by applying a torque at the pivot point, by moving the pivot point horizontally as part of a feedback system, changing the rate of rotation of a mass mounted on the pendulum on an axis parallel to the pivot axis and thereby generating a net torque on the pendulum, or by oscillating the pivot point vertically. A simple demonstration of moving the pivot point in a feedback system is achieved by balancing an upturned broomstick on the end of one's finger.

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, state space representation, 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 understanding of a similar problem can be shown by simple robotics in the form of a balancing cart.

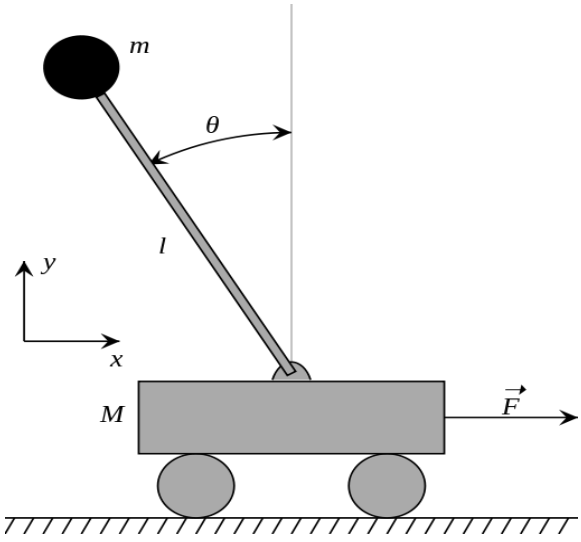

Fig .3. Inverted Pendulum

Control System

Control systems will be responsible for balancing the vehicle .It will be a closed loop feedback system. The reading from the IMU(inertial measurement system) will be processed in the microprocessor by an algorithm to calculate the acceleration needed for the given angle of tilt.This value will further be used to estimate the amount of current that needs to be supplied to the motor for obtaining this acceleration.The output from processor will be input to the electronic speed controller or ESC which will change the direction and speed of the motor by changing the direction and amount of current respectively.

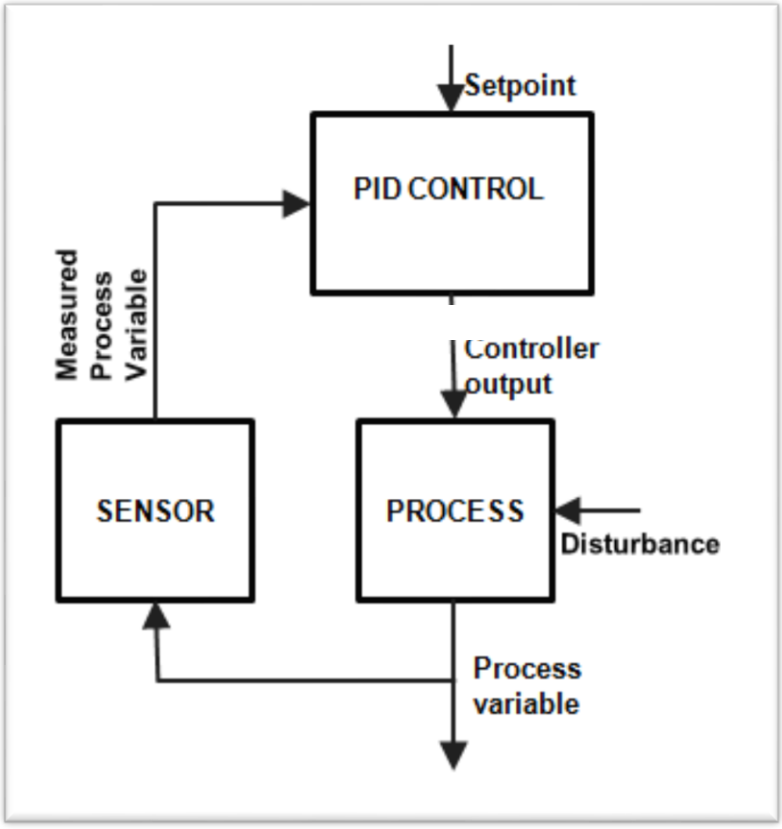

Fig..4. Algorithm Flow Chart

MPU6050 sensor: 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. Therefore 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.

Reading the raw values for the accelerometer and gyro is easy. The sleep mode has to be disabled, and then the registers for the accelerometer and gyro can be read. But the sensor also contains a 1024 byte FIFO buffer. The sensor values can be programmed to be placed in the FIFO buffer. And the buffer can be read by the Arduino. The FIFO buffer is used together with the interrupt signal. If the MPU-6050 places data in the FIFO buffer, it signals the Arduino with the interrupt signal so the Arduino knows that there is data in the FIFO buffer waiting to be read.

A little more complicated is the ability to control a second I2C-device.
The MPU-6050 always acts as a slave to the Arduino with the SDA and SCL pins connected to the I2C-bus. But beside the normal I2C-bus, it has it's own I2C controller to be a master on a second (sub)-I2C-bus. It uses the pins AUX_DA and AUX_CL for that second (sub)-I2C-bus.
It can control, for example, a magnetometer. The values of the magnetometer can be passed on to the Arduino. Things get really complex with the "DMP" .The sensor has a "Digital Motion Processor" (DMP), also called a "Digital Motion Processing Unit". This DMP can be programmed with firmware and is able to do complex calculations with the sensor values. For this DMP, InvenSense has a discouragement policy, by not supplying enough information how to program the DMP. However, some have used reverse engineering to capture firmware.

The DMP ("Digital Motion Processor") can do fast calculations directly on the chip. This reduces the load for the microcontroller (like the Arduino). The DMP is even able to do calculations with the sensor values of another chip, for example a magnetometer connected to the second (sub)-I2C-bus.

Accelerometer part: Accelerometer of MPU6050 gives acceleration along the 3 axes. It actually gives a 16 bit integer that has to be divided by the respective sensitivity factor(based on our requirement of range of acceleration) as given in the datasheet to obtain the acceleration value in multiples of □(acceleration due to gravity). In our case the required range was -2 to 2 and the sensitivity factor was 16384.0 .

Once the acceleration in all 3 axes is known, tilt from the horizontal(or vertical) can be found out by estimating the angle between any two acceleration components in one plane.In our case y axis was chosen as rotational axis(parallel to wheel's axel).

Although accelerometer gives accurate readings , its response time is is poor.

Gyroscope part: Gyro gives angular velocity values. Integrating these values over a period of time gives total angular shift. In this case also we don't directly get the desired values, instead an integer value is obtained which needs to be divided by the required sensitivity factor. In this case the maximum of angular rate was chosen to be and according to datasheet the sensitivity factor is 131.

Gyro sensor has a good response time ,but due to inaccuracy in every reading values drift over time.

Electronic components: We have already discussed the sensor used. Now we'll briefly describe the circuit and the other components involved:-

Arduino: We have used an arduino UNO as processor. This is where the error is estimated from the readings sent by MPU6050. Also the error correction term and value of PWM duty cycle, that is needed to change the speed of the motor are obtained here. Typically Arduino can operate at about 400hz . Below is a circuit diagram depicting arduino and MPU6050 connections.

Motor Controller: We used a RMCS-2302 with a maximum input voltage of 40V and a maximum output current of 20A . Also this driver is compatible with (can take commands from) Arduino Uno and can give pulse width modulated output i.e. change the duty cycle of the output to the motor based on the commands from arduino. All these specifications and the low cost made it suitable for our purpose.

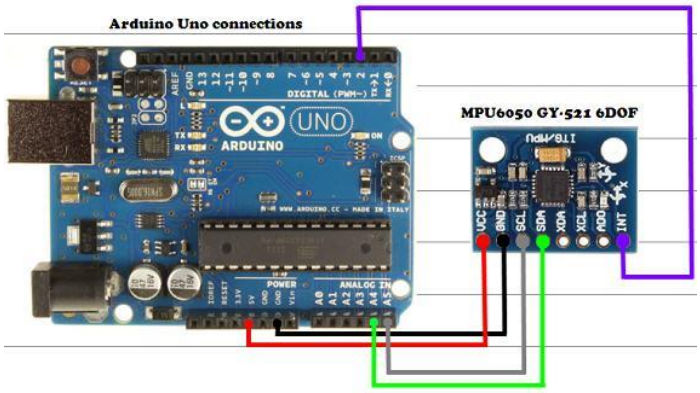

Fig.5. Sensor Arduino Circuit

Programming Logic

A proportional–integral–derivative controller (PID controller or three term controller) is a control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value **e(t)** as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively) which give the controller its name.

In practical terms it automatically applies accurate and responsive correction to a control function. An everyday example is the cruise control on a road vehicle; where external influences such as gradients would cause speed changes, and the driver has the ability to alter the desired set speed. The PID algorithm restores the actual speed to the desired speed in the optimum way, without delay or overshoot, by controlling the power output of the vehicle's engine.

The first theoretical analysis and practical application was in the field of automatic steering systems for ships, developed from the early 1920s onwards. It was then used for automatic process control in manufacturing industry, where it was widely implemented in pneumatic, and then electronic, controllers. Today there is universal use of the PID concept in applications requiring accurate and optimized automatic control.

The distinguishing feature of the PID controller is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. A PID controller which continuously calculates an error value **e(t)** as the difference between a desired setpoint SP= **r(t)** and a measured process variable PV= **y(t)** and applies a correction based on proportional, integral, and derivative terms. The controller attempts to minimize the error over time by adjustment of a control variable **u(t),** such as the opening of a control valve, to a new value determined by a weighted sum of the control terms.

Term **P** is proportional to the current value of the SP-PV error **e(t).** For example, if the error is large and positive, the control output will be proportionately large and positive, taking into account the gain factor "K". Using proportional control alone will always result in an error between the setpoint and the actual process value, because it requires an error to generate the proportional response. If there is no error, there is no corrective response.

Term **I** accounts for past values of the SP-PV error and integrates them over time to produce the I term. For example, if there is a residual SP-PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.

Term **D** is a best estimate of the future trend of the SP-PV error, based on its current rate of change. It is sometimes called "anticipatory control" as it is effectively seeking to reduce the effect of the SP-PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or dampening effect.

PID Tuning: The balance of these effects is achieved by "loop tuning" to produce the optimal control function. The tuning constants are shown below as "K" and must be derived for each control application, as they depend on the response characteristics of the complete loop external to the controller. These are dependent on the behaviour of the measuring sensor, the final control element (such as a control valve), any control signal delays and the process itself. Approximate values of constants can usually be initially entered knowing the type of application, but they are normally refined, or tuned, by "bumping" the process in practice by such as introducing a setpoint change and observing the system response.

We used manual tuning for PID controller in this project:

- Set I and D term to 0, and adjust P so that the robot starts to oscillate (move back and forth) about the balance position. P should be large enough for the robot to move but not too large otherwise the movement would not be smooth.
- With P set, increase I so that the bike accelerates faster when off balance. With P and I properly tuned, the robot should be able to self-balance for at least a few seconds.
- Finally, increase D so that the bike would move about its balanced position more gentle, and there shouldn’t be any significant overshoots.
- If first attempt doesn’t give the satisfying results, reset PID values and start over again with different value of P.
- Repeat the steps until you find a certain PID value which gives the satisfactory results.
- A fine tuning can be done to further increase the performance of PID system.·
- In fine tuning, PID values are restricted to neighboring values and effects are observed in practical situations.

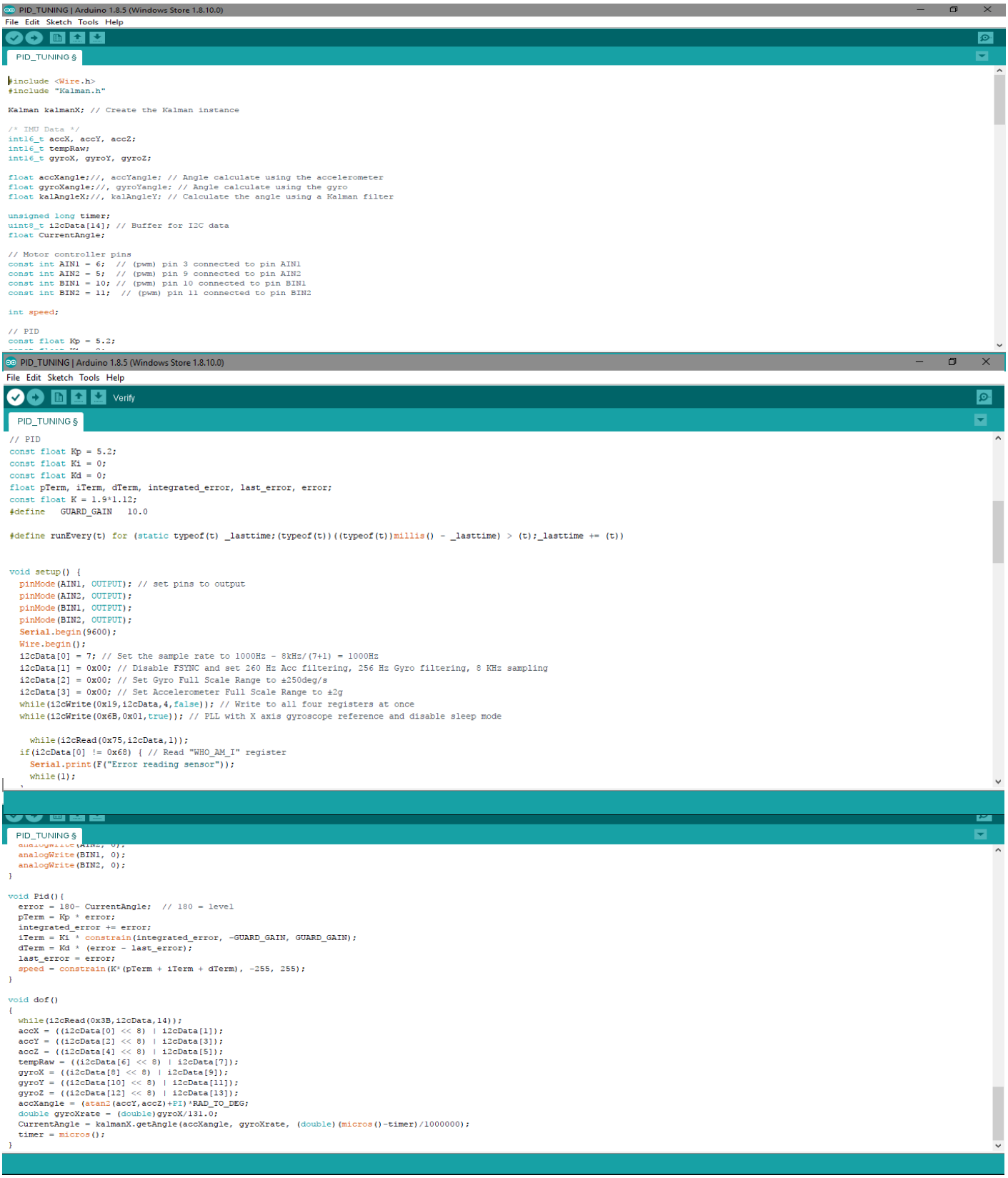

Fig.6(a). PID Tuning Arduino Program

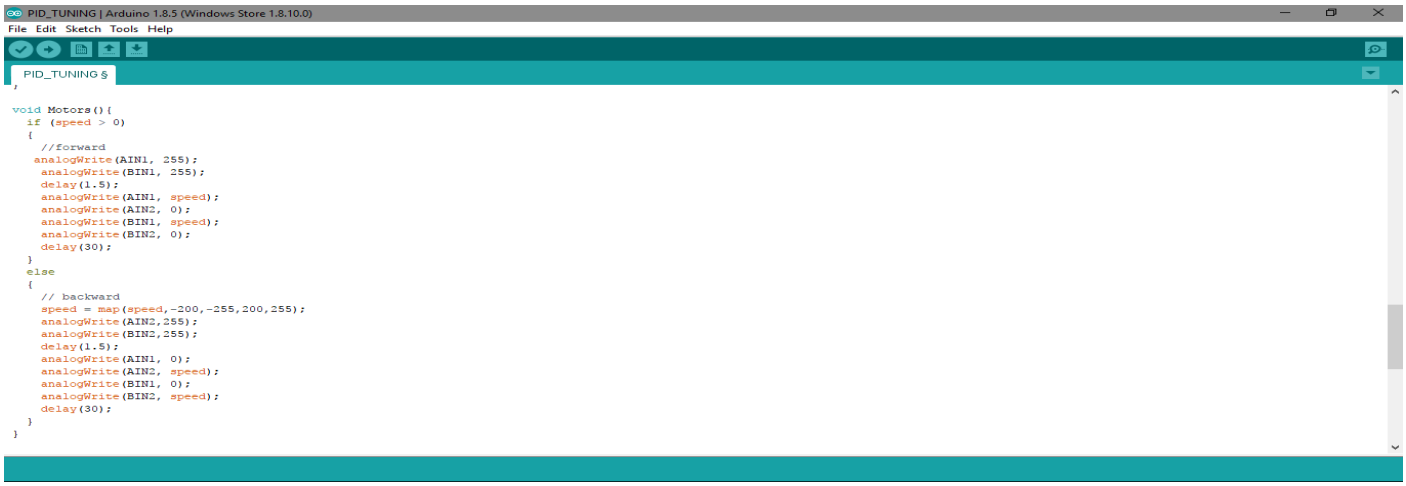

Fig.6(b). PID Tuning Arduino Program

Arduino Program

We have used Arduino-Uno as our microcontroller which use a modified version C as its programming language. Also we need a filter because of the problem that both accelerometer and gyroscope consists. As an accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely. If we are working on an actuated system , then the forces that drive the system will be visible on the sensor as well. The accelerometer data is reliable only on the long term, so a "low pass" filter has to be used.

While gyroscope also some problems as it was very easy to obtain an accurate measurement that was not susceptible to external forces. The less good news was that, because of the integration over time, the measurement has the tendency to drift, not returning to zero when the system went back to its original position. The gyroscope data is reliable only on the short term, as it starts to drift on the long term. When looking for the best way to make use of IMU-sensor, thus combine the accelerometer and gyroscope data, a lot of people get fooled into using the very powerful but complex Kalman filter. However the Kalman filter is great, there are 2 big problems with it that make it hard to use:

- Very complex to understand.
- Very hard, if not impossible, to implement on certain hardware (8-bit microcontroller etc.)

That's the reason why we are using Complimentary filter as it gives us a "best of both worlds" kind of deal. On the short term, we use the data from the gyroscope, because it is very precise and not susceptible to external forces. On the long term, we use the data from the accelerometer, as it does not drift. In it's most simple form, the filter looks as follows:

```
/*---X axis angle---*/

  Total_angle[0]=.98 × (Total_angle[0]+Gyro_angle[0]×elapsedtime)+.02 ×Acceleration_angle[0];


 /*---Y axis angle---*/

  Total_angle[1]=.98 × (Total_angle[1]+Gyro_angle[1]×elapsedtime)+.02 ×Acceleration_angle[1];
```

The gyroscope data is integrated every time-step with the current angle value. After this it is combined with the low-pass data from the accelerometer (already processed with atan2). The constants (0.98 and 0.02) have to add up to 1 but can of course be changed to tune the filter properly.

The function "Complementary Filter" has to be used in a infinite loop. Every iteration the pitch and roll angle values are updated with the new gyroscope values by means of integration over time. The filter then checks if the magnitude of the force seen by the accelerometer has a reasonable value that could be the real g-force vector. If the value is too small or too big, we know for sure that it is a disturbance we don't need to take into account. Afterwards, it will update the pitch and roll angles with the accelerometer data by taking 98% of the current value, and adding 2% of the angle calculated by the accelerometer. This will ensure that the measurement won't drift, but that it will be very accurate on the short term.

Below are screenshots of our final code that we use:

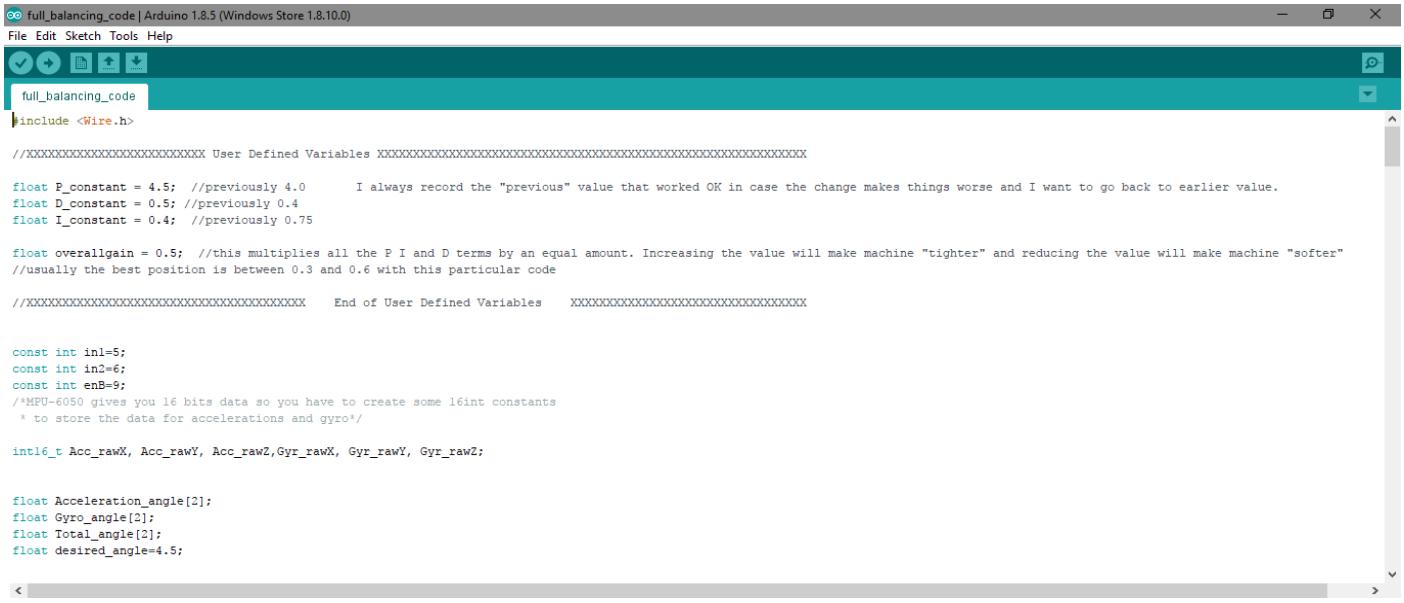

Fig.7(a). Defining variables

In Fig.7(a) above we can see that all the variables like Acc_rawX, Acc_rawY, Acc_rawZ,Gyr_rawX, Gyr_rawY, Gyr_rawZ which define the yaw, pitch and roll about X, Y and Z axis and final PID values are sent into the system with their predefined libraries also various arduino pins are also connected and numbered.

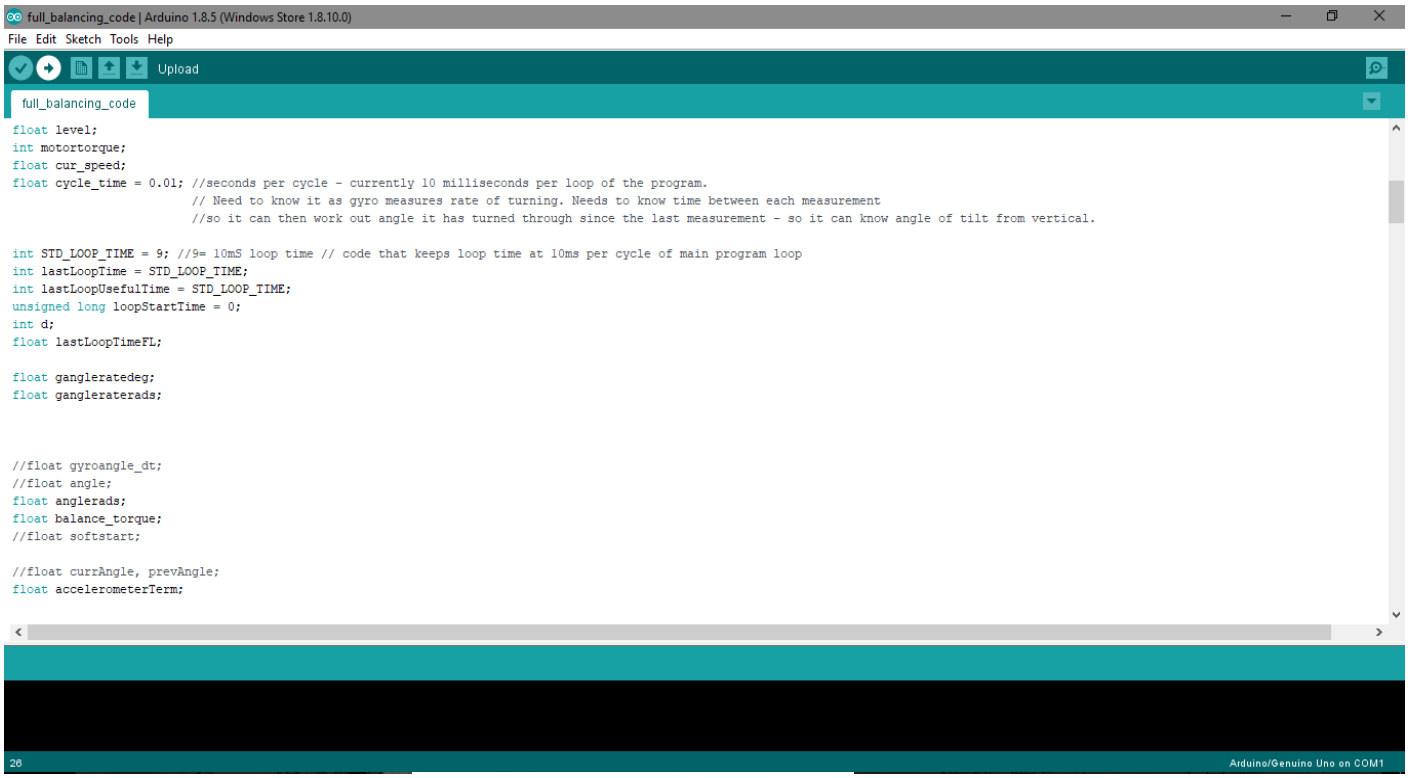

Fig.7(b). User defined variables

In Fig.7(b) cycle is time defined and set two a value of 0.01 which means it will take 0.01secs for every cycle to complete.

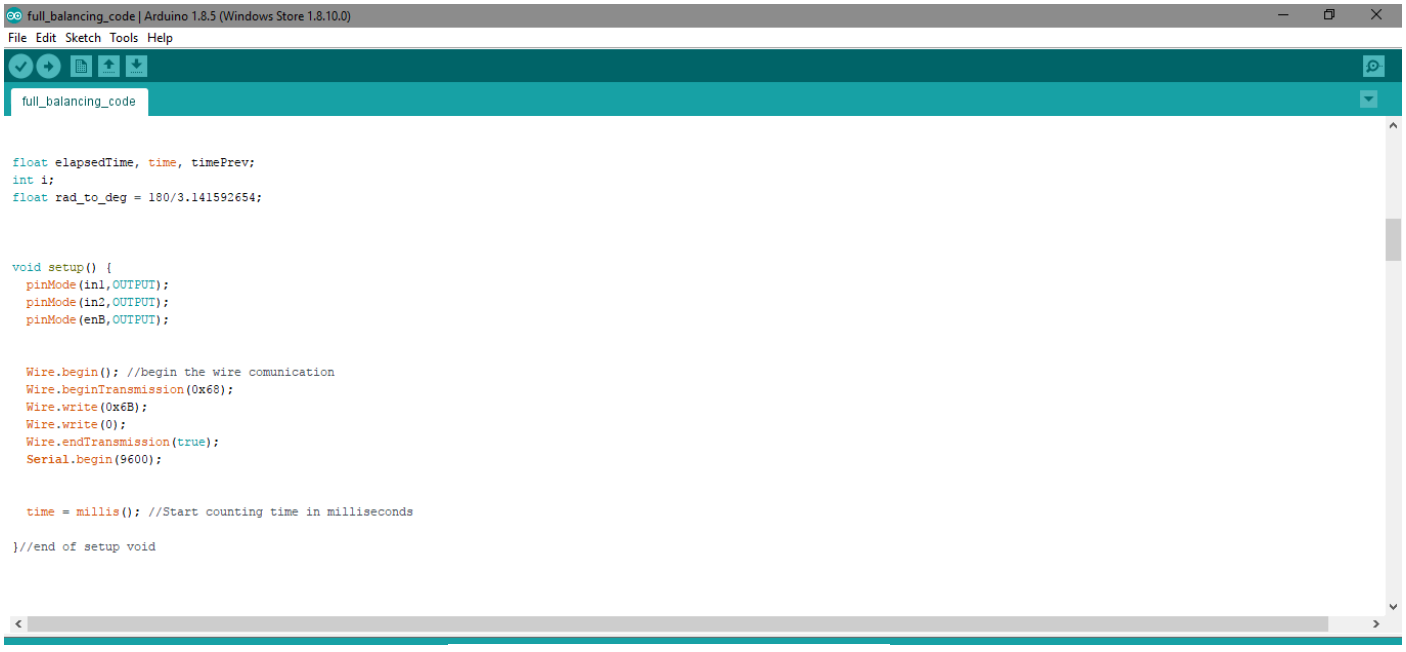

Fig.7(c). setup() function

In Fig.7(c) different pin-mode of the connected pins are defined to which they will operate i.e., Output or Input and time counting is converted to milliseconds. The data transfer has also started in the program. With this the setup part of the program has completed.

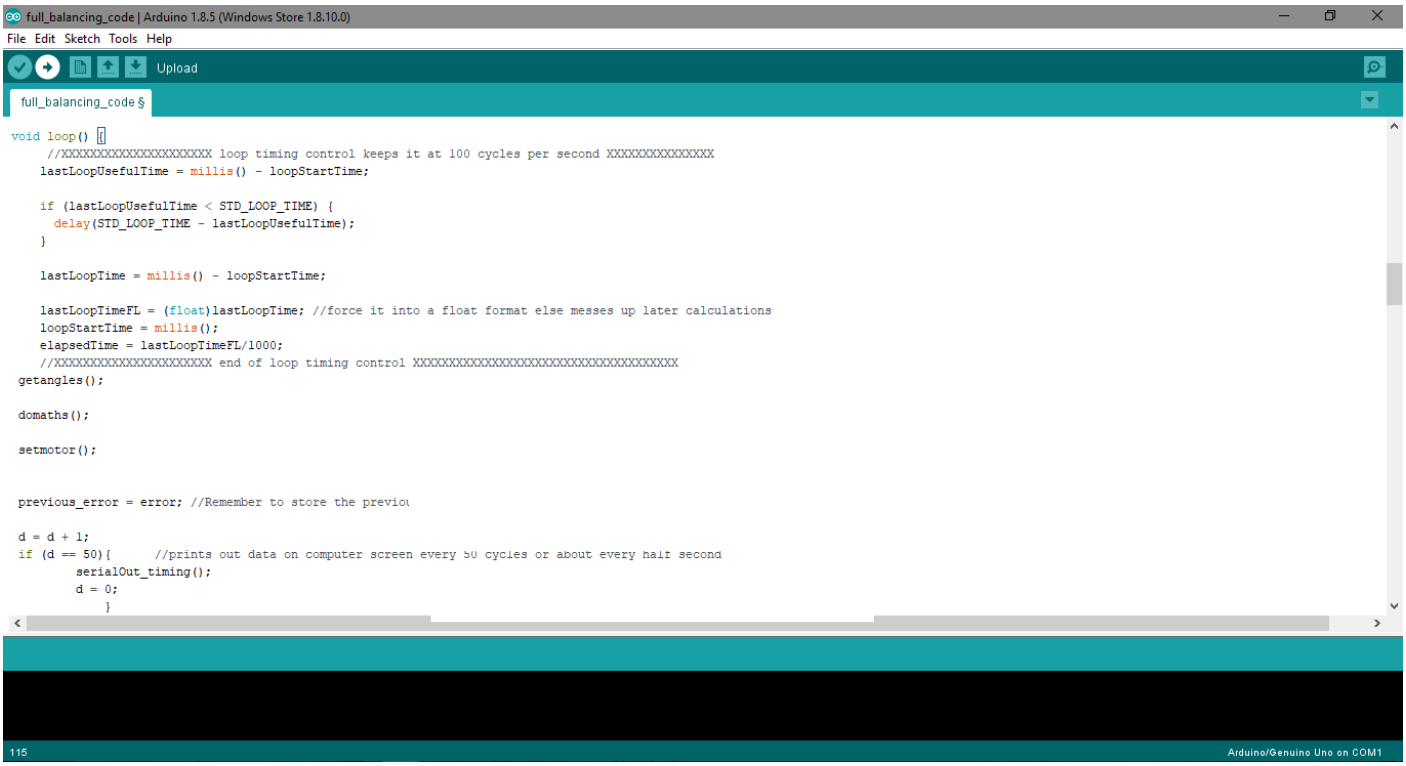

Fig.7(d). Main loop() function

In Fig.7(d) This part is main which control the loops and the data that has been coming to the IMU it prints and it calls the certain function that controls the bike like getangles(), domaths(), setmotor().

In Fig.7(e) This part is where the angle is calculated and converted to degrees as input is coming in degrees per second with the help of gyroscope and complementary filter is implemented.

Fig.7(e). getangle() function

Fig.7(f). domaths() function

Fig.7(f) Consists of domaths() function of program which calculate anglerads, gangleraterads, balance_torque, cur_speed and level.

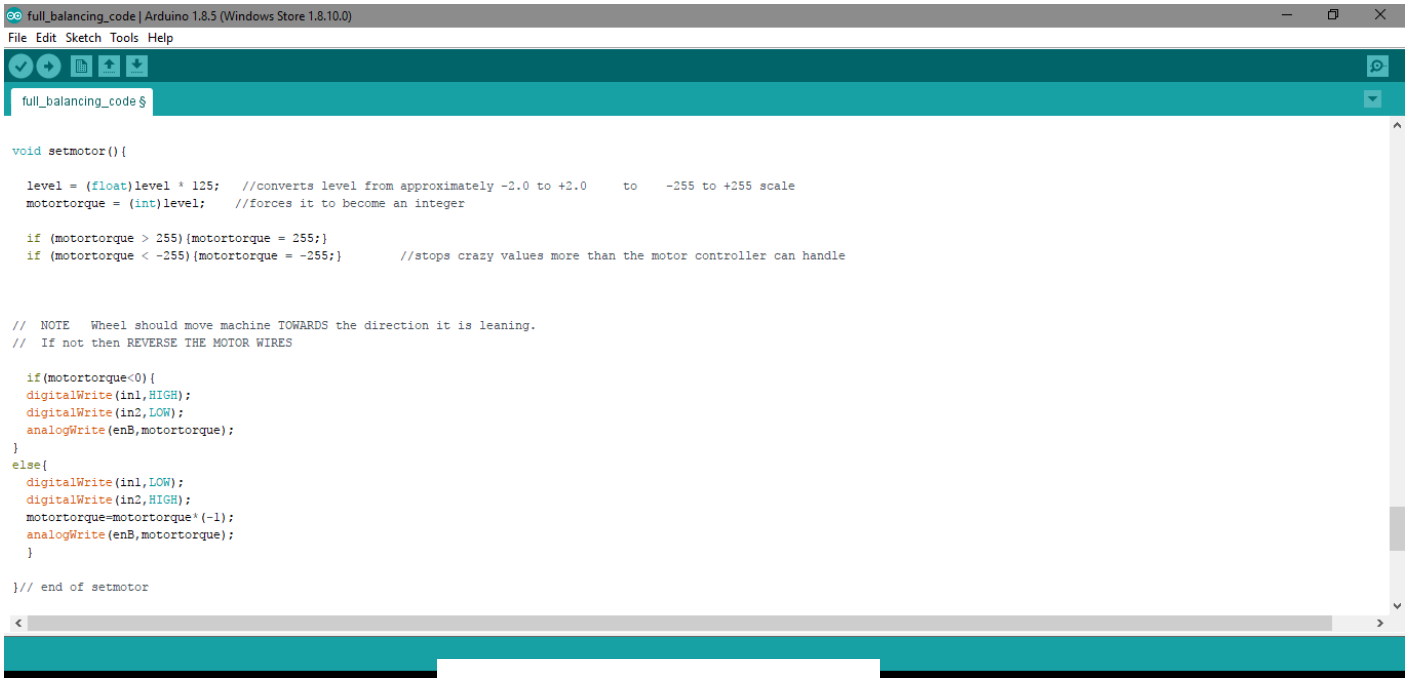

Fig.7(g) contains setmotors() function which calculates the torque and direction that should be given by the motor for perfect balancing of bike.

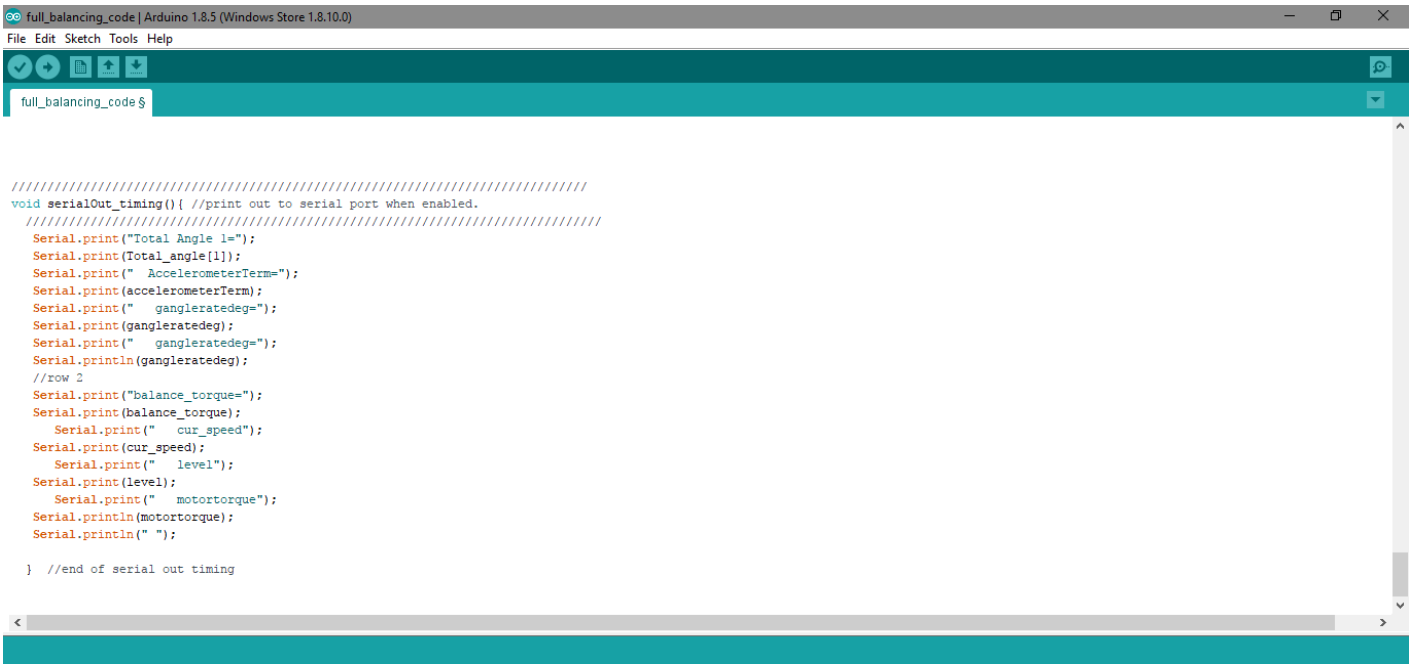

Fig.7(h) contains serialOut_timing he serial monitor so that if we wants to check whether the sensors and electrical components are working properly or not.

Progress and Conclusion

The structural and electronic aspects of this project are completed, as it can be seen in the figures below.

Fig.8(a). Progress Photos

Also we are ready with a basic static balancing code. Proper calibration of the P,I and D gain constants can give us better results in terms of static balance of the bike.

However to make the vehicle ready for riding, modifications will be needed.Most importantly, at the moment we need to calibrate gain constants for every rider(different weight).This way every time a new rider comes we have go through the same process again and again. Using a weighing sensor and a code to estimate the gain constants within the processor will make it convenient to the user.

Apart from the technicalities ,our main aim was to make this vehicle cheaper.Till now our total expenditure has been around Rs24000(which includes spare parts, and loses and therefore more than the actual cost of product).We do agree that more work and probably more funding is still needed to achieve accuracy, but with our progress so far we are confident that the finished product will be affordable.

Future Scope

We have already discussed the need and benefits of self balancing small sized electric vehicles in our cities. In this section we will focus on the future scope and usability of our project in our own campus.

Once finished this bike will serve as the most appropriate short range personal transport. Apart from being eco-friendly ,its low speed makes it suitable for a campus like ours. More focus on using light weight ,durable material will make it easier to be carried. With a folding handle ,it may even be carried as wheeled luggage.

Linking a fleet of these vehicles with the smart cards in future, will make their use by our students, faculty and staff members more efficient.

Not only the finished product will be highly useful to the institute, but by continuing further development of this bike, our students will experience working in one of the most interesting and challenging interdisciplinary projects.

References

- http://www.segway.com/products/consumer-lifestyle/ninebot-ones1
- http://rynomotors.com
- US20120217072,Christopher J. Hoffmann, Anthony J. Ozrelic, Published in Aug 30, 2012.
- Sreevaram Rufus Nireektion Kumar, Bangaru Akash and T. Thaj Mary Delsy "Designing the Monowheel by Using Self Balancing technique", 2016.
- Manpreet Singh, Ankit Sharma, Anshul Agnihotri, Pranabesh Dey, Diganta Kalita, Sushobhan Shekhar Dutta "An Investigation Study Based on Emerging Demand of Electric Unicycle Vehicles", May 2015.