INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

EKLAVYA SUMMER INTERNSHIP 2017

OPEN SKY PLANETARIUM

A project report

Submitted by

Jai Deep Mishra
NIT Durgapur

Ram Mohan Kumar
IIT-BHU

Project Mentors:

Mr. Rupak Rokade

Ms. Inderpreet Arora

Mr. Sudhakar Kumar

Supervisors:

Prof. D.B. Phatak

Prof. Kannan M. Moudgalya

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Prof. D.B Phatak, Prof. Avinash Awate** and **Prof. Kannan M. Moudgalya** for offering and allowing us to work on this novel project.

We further express our special thanks to our project guide, **Ms. Inderpreet Arora** and **Mr. Rupak Rokade** whose valuable contribution in stimulating suggestions and encouragement helped us in coming with the best solutions.

We would like to thank **Mr. Rajesh Kushalkar**, Senior Project Manager of IDL (Integrated Development Lab) for providing resources and constant support for the development of the project.

We would also like to express our sincere thanks to the entire IDL team, who helped us with their knowledge and support.

Last but not the least, we express our deepest gratitude to **Mr. Sudhakar Kumar**, Research Assistant who has invested his full efforts in guiding the team and achieving the goal.

CERTIFICATE

We hereby declare that this project report entitled, **"Development of Open Sky Planetarium"** is being submitted by us in fulfillment of the requirements for the completion of Summer Internship 2017 at Integrated Development Lab, FOSSEE Group, Indian Institute of Technology Bombay. This report is an authentic record of the aforesaid project carried out during the period of **8th May, 2017–3rd July, 2017**.

Jai Deep Mishra	**Ram Mohan Kumar**	**Saloni Mundra**
NIT Durgapur	**IIT BHU**	**DA-IICT**

This is to certify that Jai Deep Mishra, Ram Mohan Kumar, and Saloni Mundra have worked sincerely for the above mentioned project under my supervision and guidance during their summer internship. Their performance and conduct during the project was satisfactory.

Project Mentor:	**Project Guide:**
Ms. Inderpreet Arora	**Prof. D.B. Phatak**
Mr. Rupak Rokade	
Mr. Sudhakar Kumar	

CONTENTS

ABSTRACT

The project aims to provide low cost access to planetariums, especially to the schools of the country. The purpose is to educate and motivate the youth to explore and know about the night sky. Visit to a planetarium is expensive and seldom do we get a chance to see one. On the contrary, clear skies can still be observed away from the cities, but with the lack of proper knowledge base to rely on for stargazing, the experience becomes obsolete. Open Sky Planetarium is a project that seeks out schools to provide them with an educational tool to introduce kids to the night sky through a planetarium-like experience under an open sky. This is achieved by making the equipment low-cost, thus making it affordable and accessible to all schools to play a planetarium show in their own locality. It also aims to keep the operation of the equipment and the application as user-friendly as possible.

INTRODUCTION

Open Sky Planetarium makes use of Stellarium- an open source software that simulates stars in real-time and can be used as an on-screen planetarium. The tool has a LASER pointing device that guides the user through the stars and is controlled by an application supported by Stellarium. The idea is to calibrate the device by setting at least two stars as references and then any star can be guided to by the means of Stellarium and the OSP plugin built for it. The OSP plugin also provides the user with a script engine that can be used to write scripts with background audio to be played during the planetarium show.

STELLARIUM

Stellarium is open source software that simulates the night sky and displays a number of stars in their position at a particular time. The night sky simulated is specific to various locations; hence anybody can view the sky in their location in Stellarium to refer to actual objects in the sky while stargazing. The database in Stellarium is comprehensive: it contains the equatorial coordinates of an object in the sky in reference year J2000, from which the data is calculated to simulate the object in its position at any particular point of time. The objects in the Stellarium sky move just as the objects in the real sky do: the simulation mirrors the actual sky. There are also various sky cultures so that people from all parts of the world can use the software hassle-free.

Fig 1: Stellarium Window

MATHEMATICS INVOLVED IN PLUGIN FOR STAR TRACKING

We basically transform equatorial spherical coordinate system to horizontal spherical coordinate system. As sky can be considered as sphere with stars on it, we use spherical coordinate system.

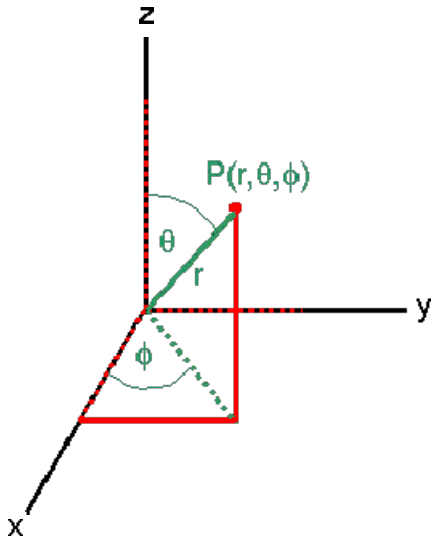


Fig 2: Spherical Coordinates

Spherical coordinate system deals with angles only hence it is useful for stepper motor driving and compatible with the data received from the Stellarium (which are also in terms of angle).

To define spherical coordinates, we take an axis (the polar axis) and a perpendicular plane (the equatorial plane), on which we choose a ray (the initial ray) originating at the intersection of the plane and the axis (the origin O). The coordinates of a point P are the distance r from P to the origin; the angle Θ (zenith) between the line OP and the positive polar axis; and the angle ϕ (azimuth) between the initial ray and the projection of OP onto the equatorial

plane. The range of ϕ is from 0 to 2π (360^o), and the range of Θ is from 0 to π (180^o).

To transform from Cartesian to spherical coordinates and vice versa, we use the transformation equations

x = r sinΘcosϕ

y = r sinΘ sinϕ,

z = r cosΘ,

r = $(x^2 + y^2 + z^2)^{0.5}$,

Θ = tan-1$(z/(x^2+y^2)^{0.5})$,

ϕ = tan-1(y/x).

Spherical coordinates with r = 1 (unit vector) are called as vector cosines.

EQUATORIAL COORDINATE SYSTEM

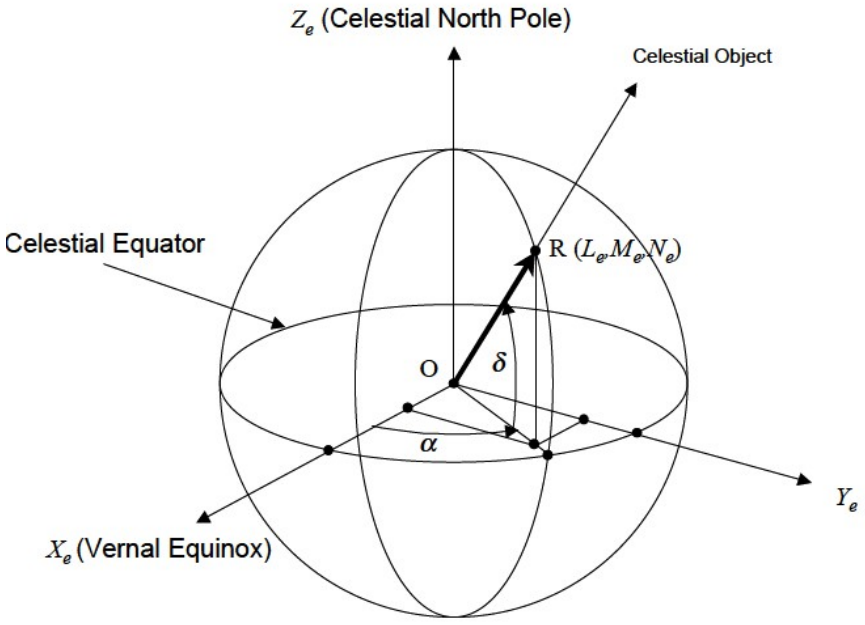


Fig 3: Equatorial Coordinates System

$$\begin{pmatrix} L_e \\ M_e \\ N_e \end{pmatrix} = \begin{pmatrix} \cos\delta\cos\alpha \\ \cos\delta\sin\alpha \\ \sin\delta \end{pmatrix}$$

In the equatorial coordinate system, Earth's equator is the plane of reference. Earth's axis of rotation points to the north and south celestial poles. The celestial sphere is as large as the known universe, and Earth is at the center of this sphere. The celestial poles do not move as Earth rotates. For an observer standing at Earth's equator, the celestial poles are on opposite horizons at exactly the north and south points, and the celestial equator passes overhead going exactly from the east to west horizon.

Declination -The declination symbol δ, (lowercase "delta", abbreviated dec) measures the angular distance of an object perpendicular to the celestial equator, positive to the north, negative to the south.

Right Ascension-The right ascension symbol α, (lowercase "alpha", abbreviated RA) measures the angular distance of an object eastward along the celestial equator from the vernal equinox to the hour circle passing through the object. The vernal equinox point is one of the two where the ecliptic intersects the celestial equator. Analogous to terrestrial longitude, right ascension is usually measured in sidereal hours, minutes and seconds instead of degrees, a result of the method of measuring right ascensions by timing the passage of objects across the meridian as the Earth rotates. There are (360° / 24h) = 15° in one hour of right ascension, 24h of right ascension around the entire celestial equator.

HORIZONTAL OR ALTAZIMUTH COORDINATE SYSTEM

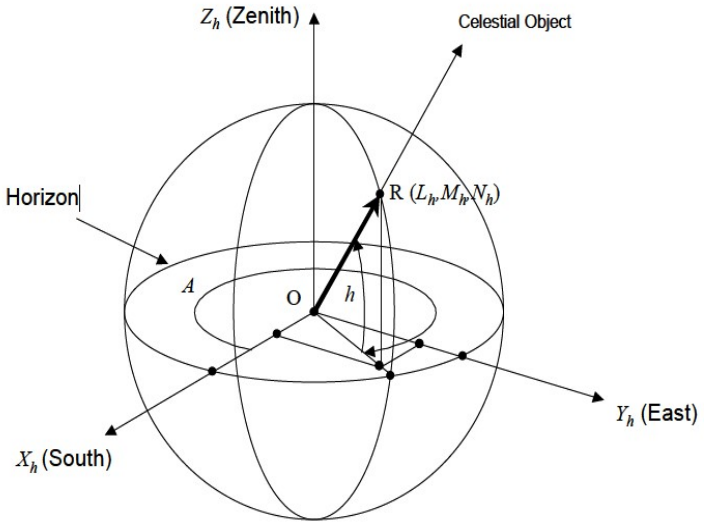


Fig 4: Horizontal Coordinates System

$$\begin{pmatrix} L_h \\ M_h \\ N_h \end{pmatrix} = \begin{pmatrix} \cos h \cos(-A) \\ \cos h \sin(-A) \\ \sin h \end{pmatrix}$$

The horizontal coordinate system is a celestial coordinate system that uses the observer's local horizon as the fundamental plane. It is expressed in terms of altitude (or elevation) angle and azimuth.

In practice, the horizon can be defined as the plane tangent to a still liquid surface such as a pool of mercury. The pole of the upper hemisphere is called the zenith. The pole of the lower hemisphere is called the nadir.

There are two independent horizontal angular coordinates:

- **Altitude (h)**, sometimes referred to as elevation, is the angle between the object and the observer's local horizon. For visible objects it is an angle between 0 degrees and 90 degrees.

- **Azimuth** (**A)**, that is the angle of the object around the horizon, usually measured from the north increasing towards the east

MATRIX TRANSFORMATION

Relationship between horizontal coordinates and equatorial coordinates can be derived by the following way. Transformation from equatorial coordinates to horizontal coordinates is expressed in matrix form as follows

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{23} & T_{33} \end{bmatrix} \begin{pmatrix} L \\ M \\ N \end{pmatrix} = [T] \begin{pmatrix} L \\ M \\ N \end{pmatrix}$$

Transformation from horizontal coordinates to equatorial coordinates is expressed as follows. This is inverse form of above equation.

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix} = [T]^{-1} \begin{pmatrix} l \\ m \\ n \end{pmatrix}$$

Where l, m, n vector cosine of an object in horizontal coordinate system

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \cos\theta \cos\varphi \\ \cos\theta \sin\varphi \\ \sin\theta \end{pmatrix}$$

L, M, N vector cosine of an object in equatorial coordinate system

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix} = \begin{pmatrix} \cos\delta\cos(\alpha - k(t-t_0)) \\ \cos\delta\sin(\alpha - k(t-t_0)) \\ \sin\delta \end{pmatrix}$$

STEPS INVOLVED IN MATRIX TRANSFORMATION

Reference Star	Observation Time	Equatorial Coordinates		Telescope Coordinates	
		Right Ascension	Declination	Horizontal Angle	Elevation Angle
Star 1	t_1	α_1	δ_1	φ_1	θ_1
Star 2	t_2	α_2	δ_2	φ_2	θ_2
Star 3	t_3	α_3	δ_3	φ_3	θ_3

Right Ascension and Declination – these values are from **Stellarium** of the star pointed.

Horizontal Angle and Elevation Angle- these values are given by arduino according to the steps moved for calibrating.

Time - these are the system time.

As we know,

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \cos\theta\cos\varphi \\ \cos\theta\sin\varphi \\ \sin\theta \end{pmatrix} \quad \begin{pmatrix} L \\ M \\ N \end{pmatrix} = \begin{pmatrix} \cos\delta\cos(\alpha - k(t-t_0)) \\ \cos\delta\sin(\alpha - k(t-t_0)) \\ \sin\delta \end{pmatrix}$$

Hence,

$$\begin{pmatrix} l_1 \\ m_1 \\ n_1 \end{pmatrix} = \begin{pmatrix} \cos\theta_1\cos\varphi_1 \\ \cos\theta_1\sin\varphi_1 \\ \sin\theta_1 \end{pmatrix} \quad \begin{pmatrix} L_1 \\ M_1 \\ N_1 \end{pmatrix} = \begin{pmatrix} \cos\delta_1\cos(\alpha_1 - k(t_1-t_0)) \\ \cos\delta_1\sin(\alpha_1 - k(t_1-t_0)) \\ \sin\delta_1 \end{pmatrix}$$

$$\begin{pmatrix} l_2 \\ m_2 \\ n_2 \end{pmatrix} = \begin{pmatrix} \cos\theta_2 \cos\varphi_2 \\ \cos\theta_2 \sin\varphi_2 \\ \sin\theta_2 \end{pmatrix} \quad \begin{pmatrix} L_2 \\ M_2 \\ N_2 \end{pmatrix} = \begin{pmatrix} \cos\delta_2 \cos(\alpha_2 - k(t_2 - t_0)) \\ \cos\delta_2 \sin(\alpha_2 - k(t_2 - t_0)) \\ \sin\delta_2 \end{pmatrix}$$

$$\begin{pmatrix} l_3 \\ m_3 \\ n_3 \end{pmatrix} = \begin{pmatrix} \cos\theta_3 \cos\varphi_3 \\ \cos\theta_3 \sin\varphi_3 \\ \sin\theta_3 \end{pmatrix} \quad \begin{pmatrix} L_3 \\ M_3 \\ N_3 \end{pmatrix} = \begin{pmatrix} \cos\delta_3 \cos(\alpha_3 - k(t_3 - t_0)) \\ \cos\delta_3 \sin(\alpha_3 - k(t_3 - t_0)) \\ \sin\delta_3 \end{pmatrix}$$

Let T be the transformation matrix to convert l, m n horizontal vector cosines to L, M, N equatorial vector cosines. Then,

$$\begin{pmatrix} l_1 \\ m_1 \\ n_1 \end{pmatrix} = [T] \begin{pmatrix} L_1 \\ M_1 \\ N_1 \end{pmatrix}$$

$$\begin{pmatrix} l_2 \\ m_2 \\ n_2 \end{pmatrix} = [T] \begin{pmatrix} L_2 \\ M_2 \\ N_2 \end{pmatrix}$$

$$\begin{pmatrix} l_3 \\ m_3 \\ n_3 \end{pmatrix} = [T] \begin{pmatrix} L_3 \\ M_3 \\ N_3 \end{pmatrix}$$

From above three equations we get the following equation

$$\begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} = [T] \begin{bmatrix} L_1 & L_2 & L_3 \\ M_1 & M_2 & M_3 \\ N_1 & N_2 & N_3 \end{bmatrix}$$

Hence the transformation matrix calculated is

$$[T] = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \begin{bmatrix} L_1 & L_2 & L_3 \\ M_1 & M_2 & M_3 \\ N_1 & N_2 & N_3 \end{bmatrix}^{-1}$$

We can also calculate transformation matrix from two sets of horizontal and vertical spherical coordinates the third point is calculated as below.

Cross product of two vectors are done to get the third vector

$$\overrightarrow{OP_3} = \overrightarrow{OP_1} \times \overrightarrow{OP_2} = \begin{pmatrix} m_1 n_2 - n_1 m_2 \\ n_1 l_2 - l_1 n_2 \\ l_1 m_2 - m_1 l_2 \end{pmatrix} \quad \overrightarrow{OP_1} = \begin{pmatrix} l_1 \\ m_1 \\ n_1 \end{pmatrix}, \quad \overrightarrow{OP_2} = \begin{pmatrix} l_2 \\ m_2 \\ n_2 \end{pmatrix}$$

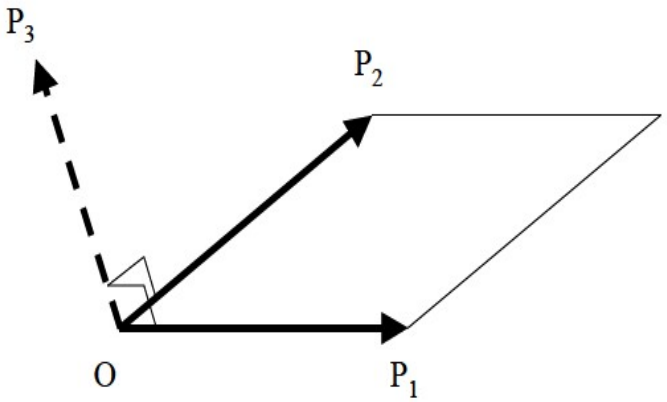


Fig 5: Vector cross-product

Making OP_3 a unit vector

$$\begin{pmatrix} l_3 \\ m_3 \\ n_3 \end{pmatrix} = \frac{1}{\sqrt{(m_1 n_2 - n_1 m_2)^2 + (n_1 l_2 - l_1 n_2)^2 + (l_1 m_2 - m_1 l_2)^2}} \times \begin{pmatrix} m_1 n_2 - n_1 m_2 \\ n_1 l_2 - l_1 n_2 \\ l_1 m_2 - m_1 l_2 \end{pmatrix}$$

$$\begin{pmatrix} L_3 \\ M_3 \\ N_3 \end{pmatrix} = \frac{1}{\sqrt{(M_1 N_2 - N_1 M_2)^2 + (N_1 L_2 - L_1 N_2)^2 + (L_1 M_2 - M_1 L_2)^2}} \times \begin{pmatrix} M_1 N_2 - N_1 M_2 \\ N_1 L_2 - L_1 N_2 \\ L_1 M_2 - M_1 L_2 \end{pmatrix}$$

Once transformation matrix is calculated then we can calculate further horizontal vector cosines of any star by multiplying its equatorial vector cosines (which is provided by Stellarium open source software) to the calculated transformation matrix T.

The horizontal vector cosines can be converted to horizontal coordinate system (azimuth and altitude) by following way

$\Theta = \tan^{-1}(m/l)$ Azimuth

$\Phi = \sin^{-1}(n)$ Altitude

NEMA17

This is a bipolar stepper motor. This 4-wire bipolar stepper has 1.8° per step for smooth motion and a nice holding torque. The motor was specified to have a maximum current of 350mA so that it could be driven easily with an Adafruit motor shield for Arduino (or other motor driver) and a wall adapter or lead-acid battery.

Technical details:

- 200 steps per revolution, 1.8 degrees
- Coil #1: Red & Yellow wire pair. Coil #2 Green & Brown/Gray wire pair.
- Bipolar stepper requires 2 full H-bridges!
- 4-wire, 8 inch leads
- 42mm/1.65" square body
- 31mm/1.22" square mounting holes, 3mm metric screws (M3)
- 5mm diameter drive shaft, 24mm long, with a machined flat
- 12V rated voltage (you can drive it at a lower voltage, but the torque will drop) at 350mA max current
- 28 oz*in, 20 N*cm, 2 Kg*cm holding torque per phase
- 35 ohms per winding

Reasons for selecting NEMA17 for our system

- More reliable than servo motors.
- Low cost with respect to maxima motors
- Sufficient to meet the torque requirement

MICROCONTROLLER SETUP

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328; offers the same connectivity and specs of the UNO board in a smaller form factor. 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.

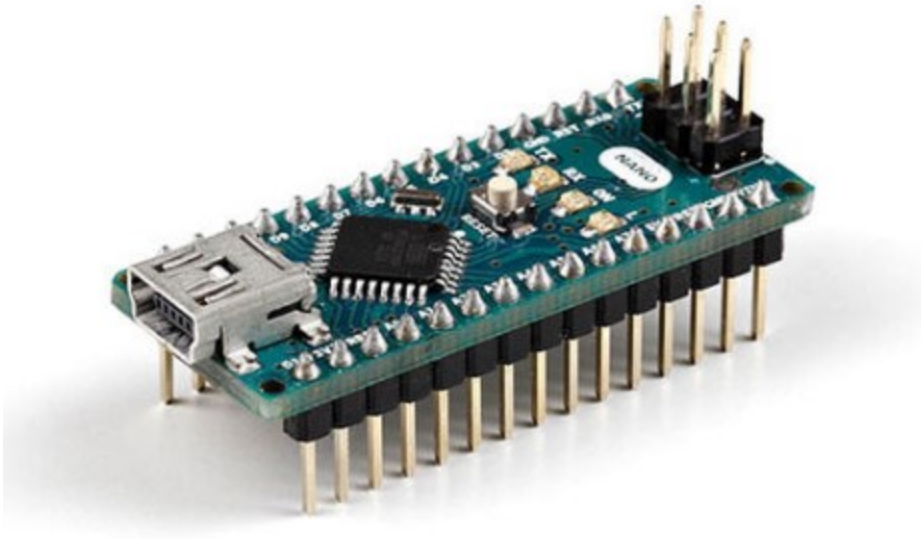


Fig. 6 Arduino Nano

We have used following pins of the Arduino Nano:

- Digital pin 7 ,8,9 are used as pulse pin ,direction pin and enable pin respectively for first motor

- Digital pin 10,11,12 are used as pulse pin ,direction pin and enable pin respectively for second motor

- Digital pin 3 (PWM) to turn on/off and control the intensity of laser
- Digital pins 2,4,5,6 are used to turn on/off relays.
- Supply ground, pulse –ve, direction –ve and enable –ve is connected to ground.

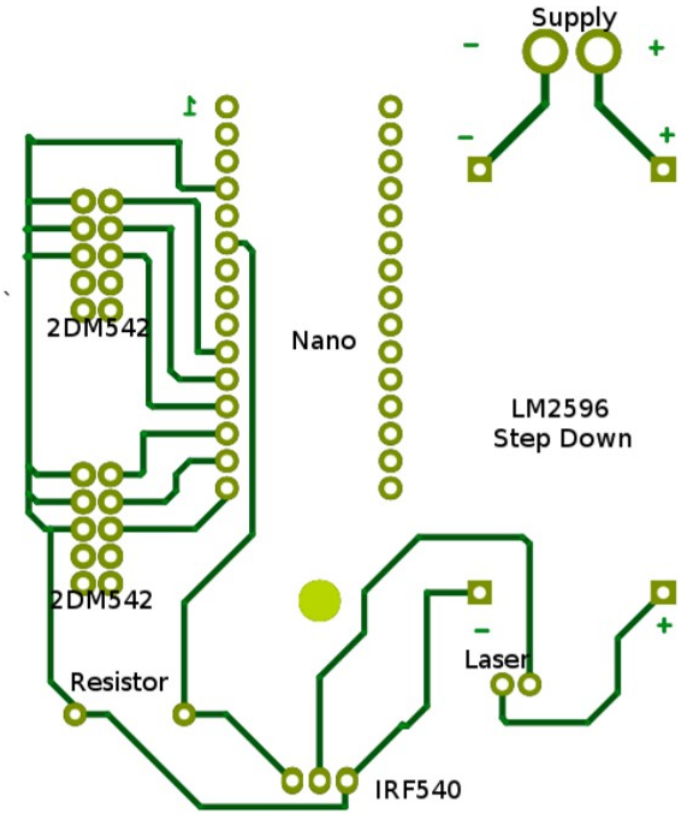
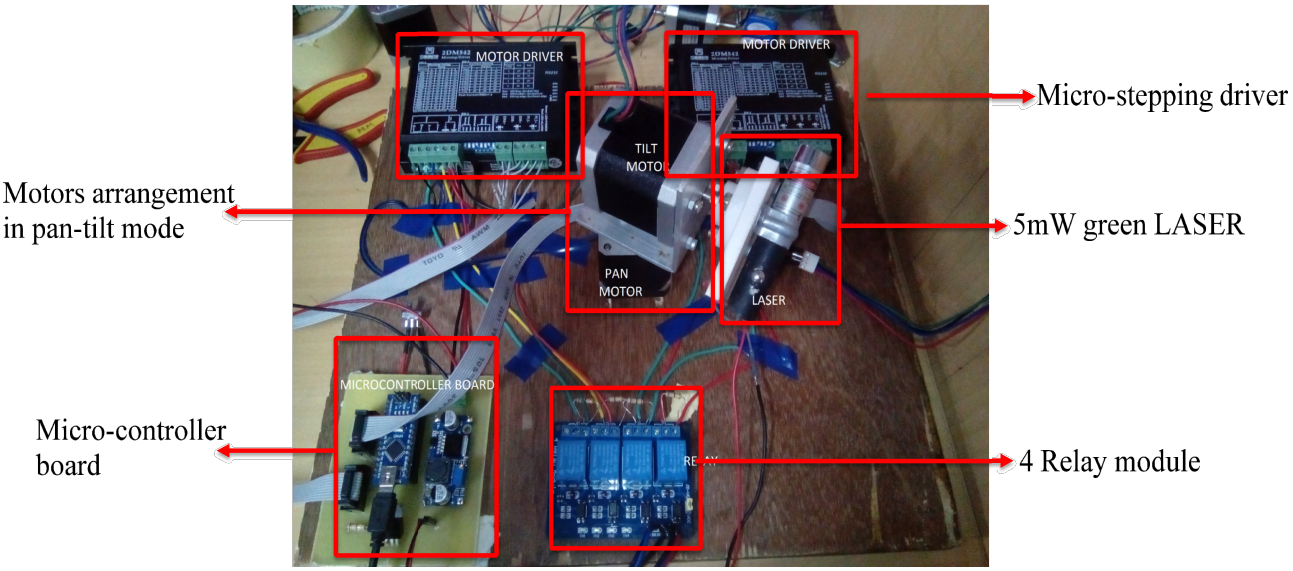


Fig. 7 Control Circuitry

HARDWARE SETUP



15

Fig 8 Hardware Setup

NEMA17 WITH A4988

For initial testing purposes, we connect NEMA17 with one motor driver called A4988 [1]. Following are some important features of this driver:

- Simple step and direction control interface
- Five different step resolutions: full-step, half-step, quarter-step, eighth-step, and sixteenth-step Adjustable current control to set the maximum current output with a potentiometer
- Intelligent chopping control that automatically selects the correct current decay mode (fast decay or slow decay)
- Over-temperature thermal shutdown, under-voltage lockout, and crossover-current protection

● Short-to-ground and shorted-load protection

An A4988 motor driver has 16 pins including the motor driver input ranging from 8-35V. Following diagram presents the connection of one NEMA17 with A4988

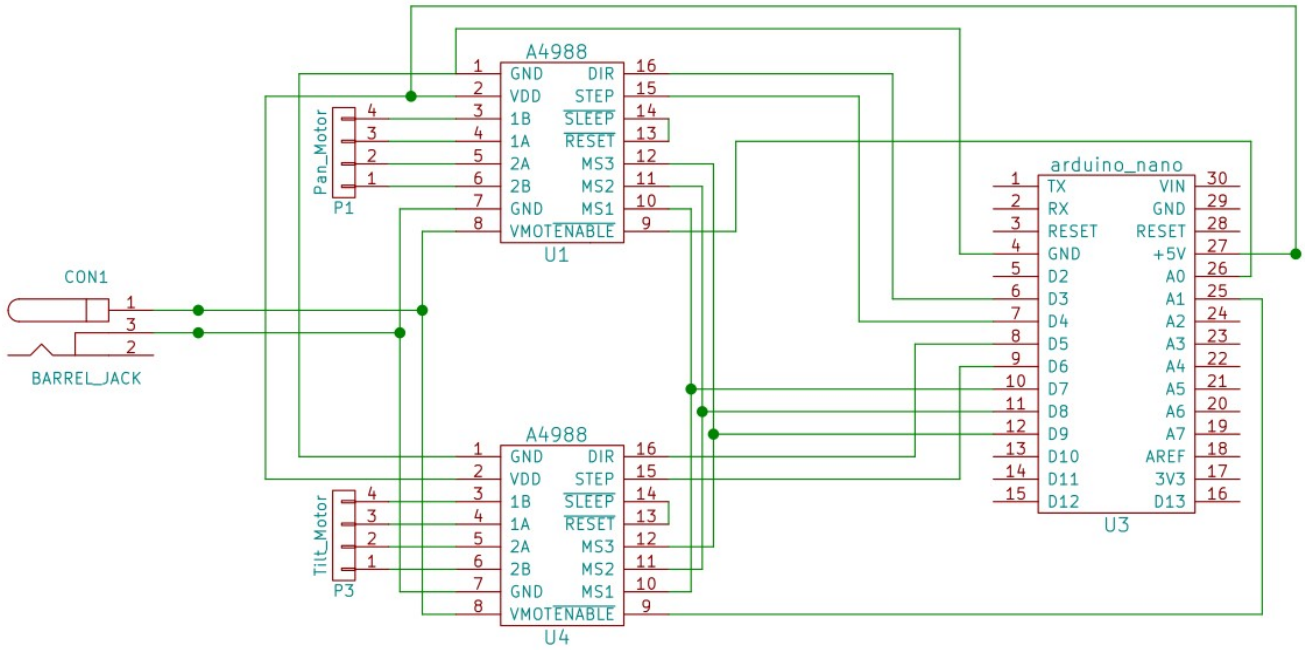


Fig 9: Circuit Diagram for connecting A4988 with Arduino Nano

NEMA17 WITH 2DM542

The current set-up involves NEMA17 stepper being driven by 2DM542 [2] micro-stepping drivers. These drivers have advanced control algorithms, which can bring a unique level of system smoothness, provide optimum torque and mid-range instability. Following are the significant features of 2DM542:

● Multi-Stepping inside, small noise, low heating and smooth movement

● Torque compensation in high speed

● Variable current control technology and high current efficiency

● Accelerate and decelerate control inside,
● Great improvement in smoothness of starting or stopping the motor
● Optically isolated input and compatible with 5V or 24V
● User-defined micro steps
● Micro-step resolutions and Output current programmable
● Over current and overvoltage protection
● Automatic detection, flexible selection of pulse edge count mode.

A 2DM542 has six control signals namely ENA-, ENA+, DIR-, DIR+, PUL-, and UL+. In addition to this, there are four pins namely B-, B+, A-, and A+ which are dedicated for motor pins. For the driving input, we have two pins namely V and GND. Following diagram presents the connections of micro stepping driver with NEMA17:

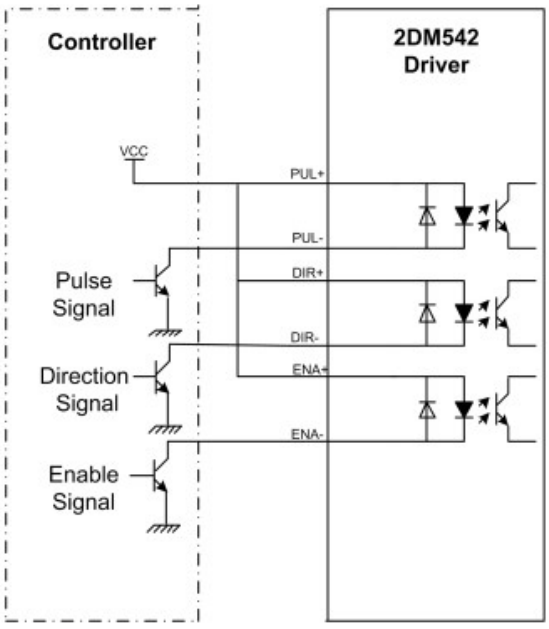


Fig 10: 2DM542 Micro-stepping driver

BUCK CONVERTER (LM2596) FOR 24V TO 3.8V

The green laser used in this project operates on 3.3V, 300mA [3]. Thus, we use a buck converter module based on LM2596 [4] to extract 3.8V from the main supply of 24V. Next, this output of 3.8V from LM2596 module is fed to LASER via a MOSFET IRF540 [5]. In order to toggle the state of LASER, we connect one digital pin of Arduino to the gate (G). A higher value of 3.8V (than 3.3V) is selected so as to counter any drop across the terminals of IRF540. The connections can be visualized from the diagram given below:

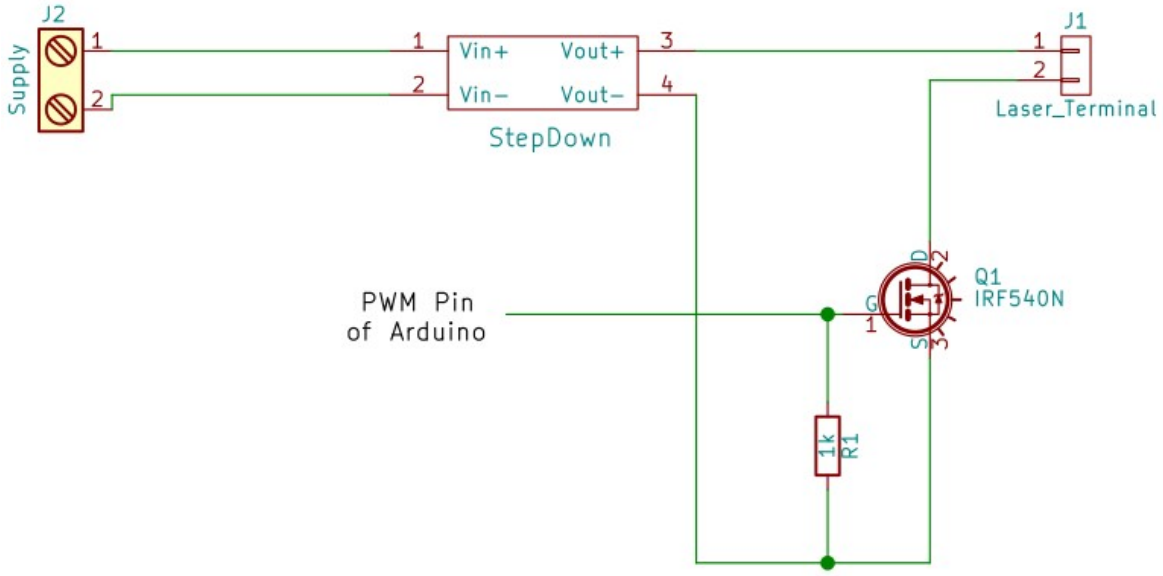


Fig 11: Laser Intensity Control

CONNECTIONS FOR DIGITAL VOLT AMP METER

In order to calculate the current drawn by the proposed set-up of Open Sky Planetarium, we use a digital volt amp meter [6]. This meter is able to read voltage between 0-100V and current ranging from 0 to 10A. The connections of this meter with our set-up can be visualized from the diagram given below:

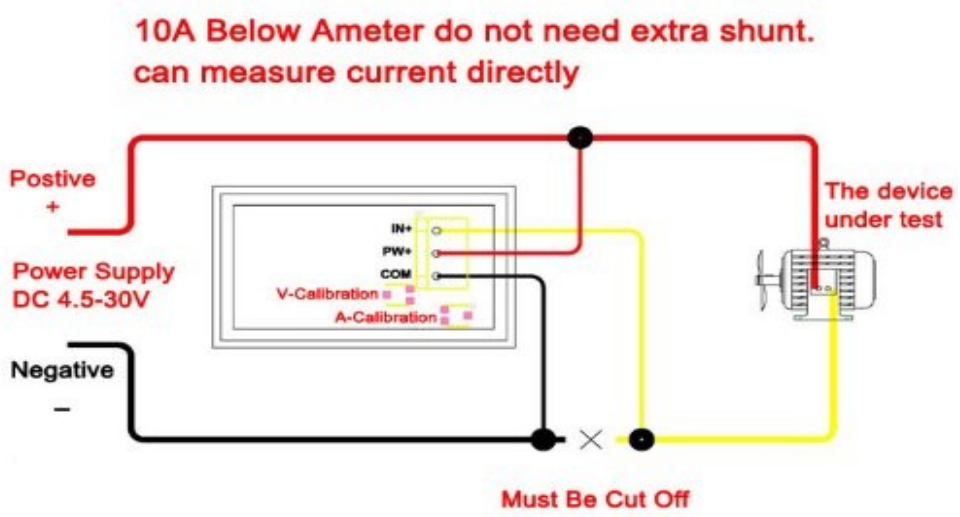


Fig 12: Volt-amp circuit connection

POWER CALCULATION FROM VOLT AMP METER

The volt amp meter gives the voltage and current simultaneously. Thus, we perform the following test cases to evaluate the power requirement.

Test case I:
Input: 24V from SMPS
Load: LASER only
Readings from the volt amp meter:
Voltage: 24.5V
Current: 0.06A (at the maximum intensity of LASER)

Test case II:
Input: 24V from SMPS
Load: One NEMA17 (with 2DM542) and LASER
Readings from the volt amp meter:

Current: 0.21A (during motor's movement and LASER's maximum intensity)
Voltage: 24.5V

From the 2nd test case, it can be inferred that the current will go upto 0.3A with both of the motors connected. It may be pertinent to note here that motors run one by one in the current set-up. So, at any given time, one motor will be running along with the LASER. Hence, with both of the motors connected and LASER in ON state, the power can be calculated as
Power = 24.5*0.3W = 7.35W < 10W

LI-ION BATTERIES REQUIREMENT

The current set-up needs 24V and 1A at full load. In order to derive this power, we connect an RPS i.e. Regulated Power Supply to the loads. Since RPS might not be easily available, we propose to run the set-up by rechargeable Li-ion batteries [7]. Following are some of the advantages associated with Li-ion batteries:
1. Relatively shorter charging time
2. Light weight
3. Ability to run for 400-500 cycles

CURRENT SPIKE

Inrush current, input surge current or switch-on surge is the maximum, instantaneous input current drawn by an electrical device when first turned on. The selection of over current protection devices such as fuses and circuit breakers is made more complicated when high inrush currents must

be tolerated. The over current protection must react quickly to overload or short circuit faults but must not interrupt the circuit when the (usually harmless) inrush current flows.
When an electric motor, AC or DC, is first energized, the rotor is not moving, and a current equivalent to the stalled current will flow, reducing as the motor picks up speed and develops a back EMF to oppose the supply. While brushed motors present essentially the winding resistance. The duration of the starting transient is less if the mechanical load on the motor is relieved until it has picked up speed.
An inrush current limiter is a component used to limit inrush current to avoid gradual damage to components and avoid blowing fuses or tripping circuit breakers. Negative temperature coefficient (NTC) thermistors and fixed resistors are often used to limit inrush current. NTC thermistors can be used as inrush-current limiting devices in power supply circuits when added in series with the circuit being protected. They present a higher resistance initially, which prevents large currents from flowing at turn-on. As current continues to flow, NTC thermistors heat up, allowing higher current flow during normal operation. NTC thermistors are usually much larger than measurement type thermistors, and are purposely designed for power applications.

Fixed resistors are widely used to limit inrush current. These are inherently less efficient, since the resistance never falls from the value required to limit the inrush current. Consequently they are generally chosen for lower power circuitry, where the additional ongoing power waste is minor. Inrush limiting resistors are much cheaper than thermistors. They are found in most compact fluorescent lamps (light bulbs).

They can be switched out of the circuit using a relay or MOSFET though, after inrush current is complete. Hence relay was used with 69 ohm resistance. Current spike decreased from around 3.7ampere to 0.74 ampere. To further decrease the spike current resistance can be increased. Now SMPS can be used because it has 24V/2A rating.

FEEDBACK MECHANISM FOR OSP

1. Using MPU6050 which consists of a 3-axis gyroscope and a 3 axis accelerometer:

The gyro measures degrees per second (^{0}/s) while the accelerometer measures acceleration (g's) in three dimensions. Both output the measurements as an analog signal. Both of these are converted to degrees using suitable coding.

Angle measurement through gyro:

First we have to translate quids (a number from 0-1023) into something useful (this is for an ADC with a 10 bit resolution, for example this should be 4095 (2^12-1=4095) for 12 bit ADC). To do this, just use this simple equation:

gyroRate = (gyroAdc-gyroZero)/sensitivity - where gyroAdc are the readed value from our sensor, gyroZero is the value when it is stationary (this is done in the code - look in the "Setup" section) while sensitivity is the sensitivity found in the datasheet, but translated into quids.

If we look in the two gyros datasheets [9] and [10] we will see that the sensitivity is 3.33mV/0/s for the 4xOUT. To translate these into quids is pretty easy: sensitivity/3.3*1023.
So in this example we get:
0.00333/3.3*1023=1.0323.

NB: to translate mV to V simple just divide by one thousand.

The final equation will look like this:
gyroRate = (gyroAdc-gryoZero)/1.0323

The result will come out as degrees per second (0/s). To translate this into degrees you have to know the exact time since the last loop. Fortunately, the Arduino got a simple command to do so: millis(). By using that, one can calculate the time difference (delta time) and thereby calculate the angle of the gyro. The final equation will look like this:
gyroAngle += gyroRate*dtime/1000

Angle measurement through accelerometer:
The accelerometer measures the acceleration (g's) in three dimensions. To translate the analog readings into degrees we simply need to read the axis and to subtract the zero offset like so:
accVal = accAdc-accZero

Where accAdc is the analog reading and accZero is the value when it reads 0g - this is calculated in the start of the code, look in the "Setup" section. The zero value can also be found in the datasheet [8]. We will see that the zero voltage at 0g is

approximately 1.5V, to translate this into quids, we again have to use this equation: zeroVoltage/3.3*1023.
So in this example I get:
1.5/3.3*1023=465.

We can then calculate the pitch and roll using the following equations:
pitch = atan2(accYval, accZval)+PI
roll = atan2(accXval, accZval)+PI

To convert it from radians to degrees we simply multiply the result by 57.2957795130823208767981548141 05 - this is predefined in the Arduino IDE as RAD_TO_DEG.

Unfortunately, the gyro drifts over time. That means it cannot be trusted for a longer time span, but it is very precise for a short time. This is when the accelerometer comes in handy. It does not have any drift, but it is too unstable for shorter time span.
Now for precise measurement of angle both these values are considered.

There are two methods for finding out the precise angle:
a. Using a Complementary Filter: It's a very simple method but it's not as precise as the other methods (i.e. Kalman filter). Our aim is to find whether Complementary Filter will be able to meet our desired accuracy of <0.1 degree.
We use a digital low-pass filter on the accelerometer and digital high-pass filter on the gyroscope readings

angle = 0.98 *(angle+gyro*dt) + 0.02*acc – We can choose any number but the sum must be 1.

b. Kalman filter: Kalman filter [11] [12] is an algorithm which uses a series of measurements observed over time, in this context an accelerometer and a gyroscope. These measurements will contain noise that will contribute to the error of the measurement. The Kalman filter will then try to estimate the state of the system, based on the current and previous states, that tend to be more precise that than the measurements alone. [13]

2. Using Encoders:

For our required precision of <.1 degree Angle encoders of 3600 PPR can also be used. PPR is the number of pulses generated for a complete revolution of 360 degree. (i.e. 1 pulse for .1 degree). It can measure angles upto .1 degree accurately but the price of each encoder is around ₹3584 [14] and we need to use two encoders; one for the pan motor and the other for the tilt motor.
Arduino code [15].

3. Using Resolver:

A resolver with a 14 bit RDC(resolver to digital converter) can be used to measure angle of rotation with a max error of .01 degree. But it is facing the same problem as the encoders i.e. It is expensive (₹9000 per piece) [16] and

we need to use two of them; one each for pan and tilt motors.

USING THE PLUGIN

Fig 13: OSP Plugin

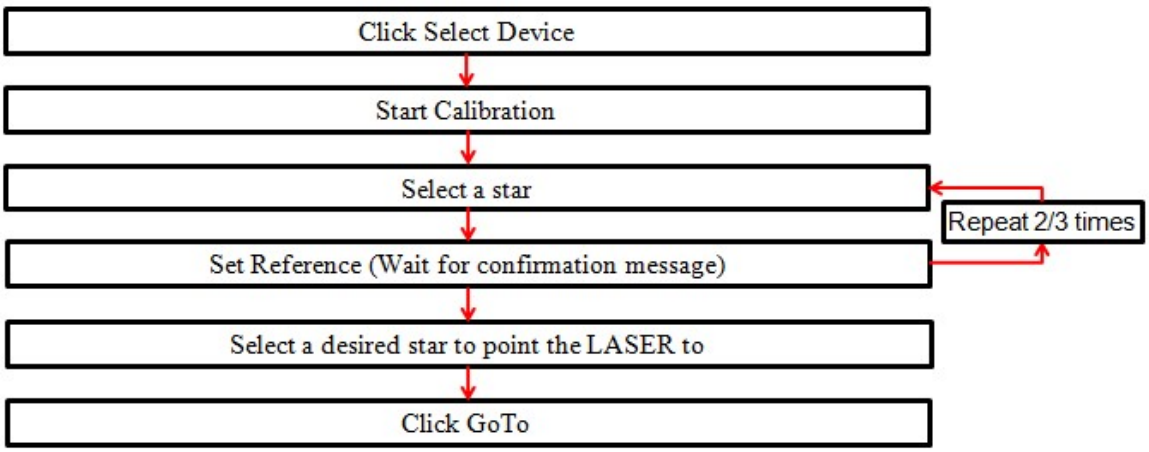


Fig 14: Flowchart

REFERENCES:

[1] http://www.robotshop.com/media/files/PDF/datasheet-1182.pdf
[2] http://www.jmc-motor.com/upload/file/20160423/20160423105943_35678.pdf
[3]https://www.sparkfun.com/datasheets/Components/5mw%20532nm%20Laser%20Module.pdf
[4]http://www.amazon.in/LM2596-Converter-Module-Supply-1-23V-30V/dp/B009P04YTO
[5] https://www.vishay.com/docs/91021/91021.pdf
[6] http://www.amazon.in/0-100V-Voltmeter-Ammeter-Digital-display/dp/B01M0QTAXI/
[7] http://www.ebay.in/itm/282413800750?aff_source=Sok-Goog
[8]http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf
[9]http://www.sparkfun.com/datasheets/Sensors/IMU/lpr530al.pdf
[10]http://www.sparkfun.com/datasheets/Sensors/IMU/LY530ALH.pdf
[11]http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/
[12]http://www.youtube.com/watch?v=FkCT_LV9Syk
Implementing MPU6050 using Kalman Filter for precise measurement:
[13]https://github.com/TKJElectronics/Example-Sketch-for-IMU-including-Kalman-filter/blob/master/IMU/MPU6050/MPU6050.ino

[14]http://www.mouser.in/Sensors/Encoders/_/N-ae944?P=1yyawsa
[15]http://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/
[16]www.indiantradebird.com/encoders-india-resolver?spid=MTYwOTUsMg==