Laser Pointer for Star-Gazing

Kalpesh Patil (130040019)
Yash Bhalgat (13D070014)
Meet Shah (13D070003)

Guide: Prof. Dipankar,Prof. Joseph John
TA/RA : Aastha Tyagi, Deepak Malani

THIS PROJECT WAS DONE IN THE PARTIAL FULFILLMENT OF THE ELECCTRONIC DESIGN LABORATORY COURSE (EE 344) AT IIT- BOMBAY

INDEX

ABSTRACT

The laser pointer is used by thousands of skygazers to show beginners the way to stars and constellations. Our aim is to aid astronomy enthusiasts with a laser pointing tool that can be interfaced with computer applications for a pleasant and enjoyable star-gazing evening.

Keywords: Stellarium, star gazing, laser pointer, pan tilt mechanism

INTRODUCTION

Motivation

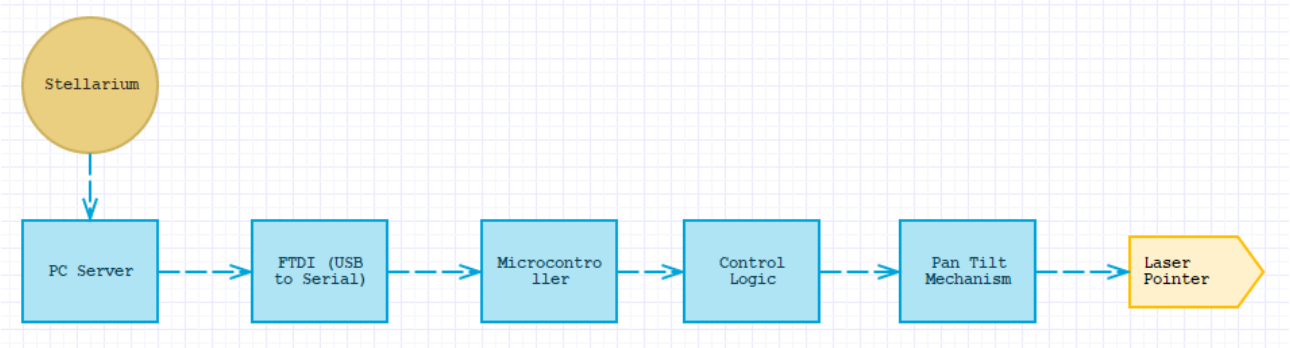
Throughout history humans have looked to the sky to navigate the vast oceans, to decide when to plant their crops and to answer questions of where we came from and how we got here. Being avid astronomers, and loving being able to look up into the night sky and know what star or planet it is, we thought of building a cost-*effective, accurate, portable* laser pointer by ourselves. Current state of the art techniques include manual laser pointing to satisfy the purpose. Also the autonomous laser pointing systems are expensive to deploy and infeasible in terms of mobility.

Goals

This project is aiming towards providing a low cost, easy to carry equipment which is compatible with the renowned star gazing application like Stellarium. The user will click on the star/constellation being viewed on the screen. The laser pointer is supposed to direct the viewer to the corresponding entity in real sky by shooting a laser beam at it.

BLOCK DIAGRAM AND DESCRIPTION OF BLOCKS

Block Diagram



Description of Blocks

1. ***Stellarium***

   Stellarium is a renowned application used to view the real time location of the celestial object in the sky. The application also provides a brief information of the celestial body user is viewing, from which we will extract the Az/Alt of the body to provide it to the laser-pointing subsystem.

2. ***PC server***

   PC Server will consist of a script running in parallel with Stellarium software. It will extract the required data about the location of the star where laser is supposed to be pointed from stellarium. The extracted data will be sent to the USB port of the PC, which is connected to the FTDI chip.

3. ***FTDI (USB to serial)***

   Data extracted from stellarium is sent to the USB port of the laptop which has to be sent to the μC IC. This is then converted to "serial" form by the FTDI chip on the PCB.

4. *Microcontroller IC and Controller Logic*

The data coming from PC will be received and appropriate calculation based on the calibrated parameters will be to initialise the laser pointer location and then move it to the required azimuth and altitude.

5. *Pan Tilt Mechanism*

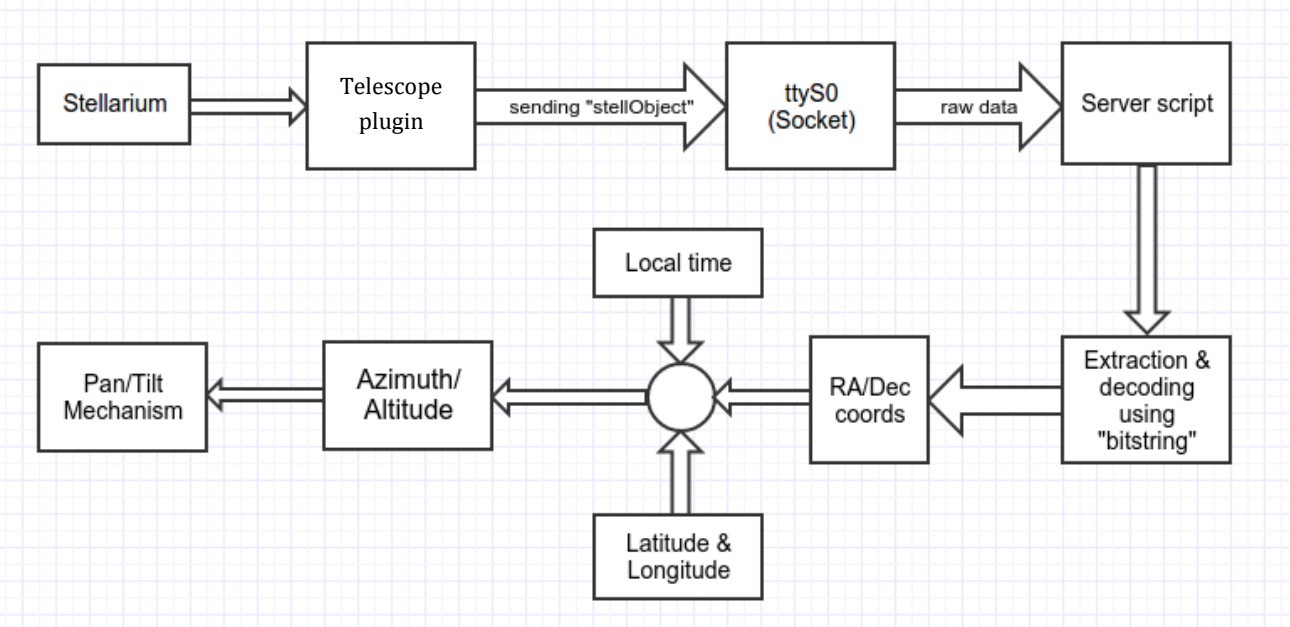
Two motors will be used for two degrees of freedom (azimuth and altitude angle control). The motors will be situated on a mount and Laser pointer will be rotated by the movement of the shaft of the motors and L-clamp. This will provide 180° of rotation in both the direction, which is enough to span entire hemisphere.

6. *Laser Pointer*

A simple laser pointer with on/off control will be situated on L-clamp whose orientation will be controlled using two motors.

PROJECT IMPLEMENTATION

Data Extraction



Following steps are involved in extraction of Alt/Az values from Stellarium software for any celestial body

1. **Configuring Telescope server plugin**
   First of all a localserver is created using python script. Stellarium is configured to connect to the local server using Telescope server plugin.

2. **Writing onto socket ttyS0**
   Due to mishandling of concurrent read and write requests in case of files, we decided to use software port for writing data. When appropriate command is given, Stellarium sends data packets wrapped in a class called 'stellObject' to the socket ttyS0 which was created locally to receive data from stellarium.

3. **Intercepting raw data**
   This data written on the port is intercepted and read by a python script. This raw string of received data is passed onto next level of extraction.
4. **Extraction of Ra/Dec cords**
   Stellarium encodes data into a different format. Therefore we can't directly extract information from it. We had to use 'BitString' library in python to convert this encoded data into ASCII. We extract Ra/Dec coordinates from this string whose locations are specified in the documentation of stellarium.
5. **Generating Alt/Az values**
   Motor coordinates should be given in the form of altitude and azimuthal angle. Therefore we convert the extracted Ra/Dec coordinates to Alt/Az coordinates. This conversion requires local time (taken from system) and latitude and longitude values of the point of operation. Once we get Alt/Az coordinates we are ready to send them to the microcontroller through serial communication
   Formulae for Ra, Dec to Alt, Az

$$LST = 100.46 + 0.985647d + long + 15UT$$
$$HA = LST - Ra$$
$$\sin(\emptyset) = \sin(Dec)\sin(Lat) + \cos(Dec)\cos(latt)\cos(HA)$$
$$\cos(\theta) = \frac{\sin(Dec) - \sin(\emptyset)\sin(latt)}{\cos(\emptyset)\cos(latt)}$$

where,

LST = *local standard time* $\quad$ $\emptyset$ = *altitude angle*
UT = *Universal time* $\quad$ θ = *azimuthal angle*
HA = *hour angle*
$long$ = *longitude of point of operation* $(72.917542\,°\,for\ Mumbai)$
$latt$ = *lattitude of point of operation* $(19.130701\,°\,for\ Mumbai)$

Software Architecture

Stellarium being open source, it greatly helped ease the implementation of our software architecture. One of the greatest challenges in developing the software architecture was developing an architecture which worked with the default build of Stellarium so as to maintain the portability and scalability of our architecture. We followed a socket based pipe interfacing with Stellarium with multiple error checks in order to make the fragile socket pipeline as robust as possible. A few critical points of our Software architecture which contribute to its robustness are :

- **Two-way message checks** : Every message that goes to-fro from arduino to PC is checked twice using a two-way handshake protocol. This greatly helps in detecting any unsend packets and consequently resending them.
- **OS Agnostic Design :** The software architecture was developed keeping in mind the wide range of users it will cater to , hence no OS specific library or feature is used to keep it as OS agnostic as possible.

A few major commands that our architecture utilizes as a base for the complexx motions of the motors are :

- ***movx :*** This command takes the signed x_offset as an input moves the lower motor by the x_offset one least count at a time.
- ***movy :*** This command takes the signed y_offset as an input moves the upper motor by the y_offset one least count at a time.
- ***goto :*** This command takes the signed x_offset and y_offset as an input moves the entire mechanism to the given offset.

Device controller GUI

The above figure shows the GUI developed by us. To cater to rookie users as well, we kept even low level features like connecting to an Arduino on the GUI. The GUI has the following functionality:

- Laser Control
- Pan Tilt Control (WASD mode)
- Custom defined keyboard shortcuts for ease of use.
- Easy switch between Pointer and Reference Set mode.
- Reference set status.
- Motor position with real time update.
- Arduino Connect and show available ports

Serial Interfacing

Two way confirmation protocol is implemented. Command and value associated with it (if required) is sent through the ACM0 port to the microcontroller. On reception of a particular command, microcontroller sends back a confirmation message ensuring the successful transmission of data.

Pan Tilt Mechanism

Two servo motors having 180° of range of rotation each are used for pan tilt mechanism. Both of the motors can be controlled independently. The motors are powered using 5V supply down converted from 12V battery using a buck convertor. The PWM signal for the motors is coming from microcontroller. The PWM to be given to each of the motor is determined by the control logic scripted in microcontroller. For ***movx, movy*** commands, motors are rotated through 1° each. For ***goto*** command, motors are rotated through required angles which is received in the microcontroller from the system.

Laser Controller

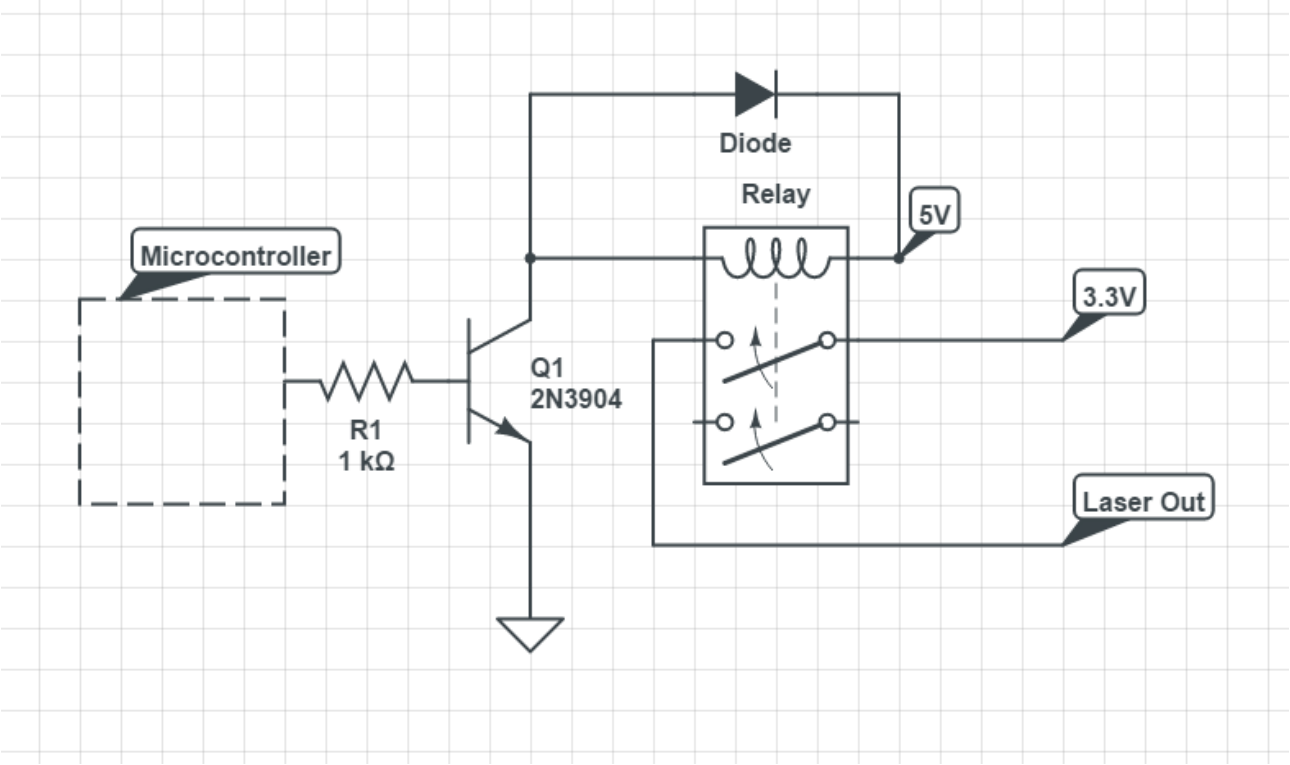
A standard astronomical laser pointer is situated at the shaft of the motor. The laser is powered through a 3.3V voltage regulator (LM 1117). Relay is used control on and off states of the laser pointer. Input to the relay is given through a BJT switch, whose base is controlled using microcontroller output pin. The microcontroller switches on that pin when ***laon*** command is received and switched off when ***laoff*** command is received.

Battery Charging Circuitary

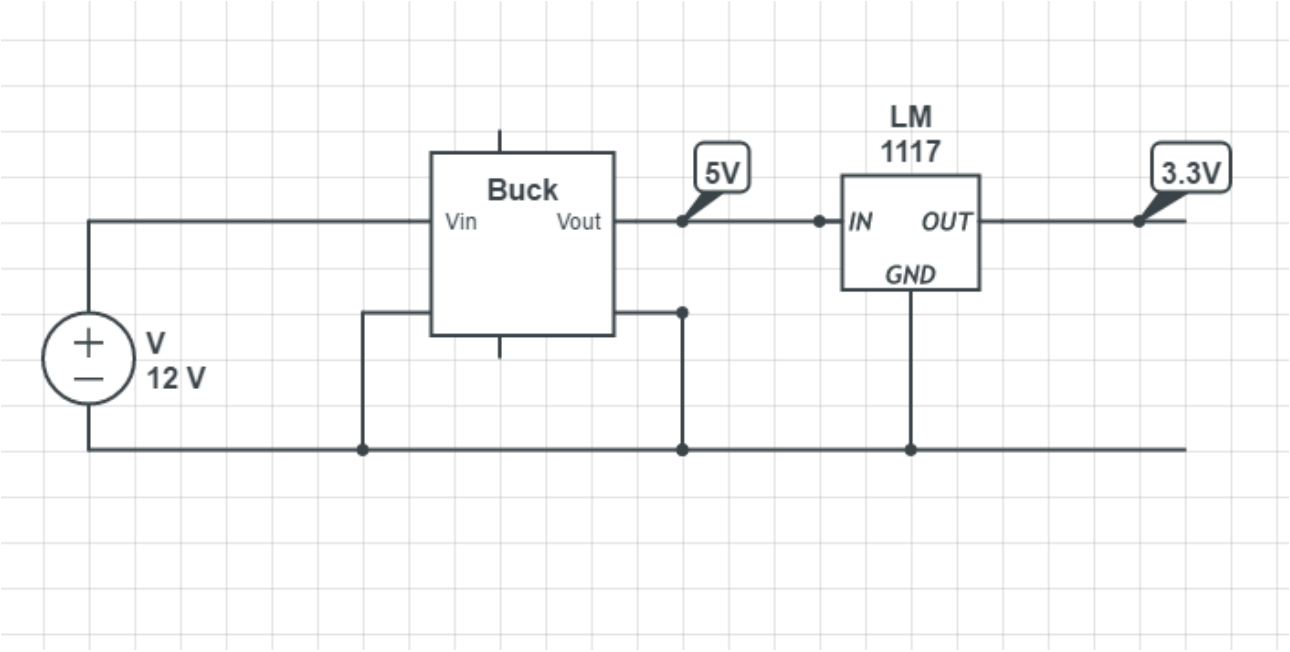
A 12V battery charging adopter is designed in order to make the module independent of power supply. The adopter consists of a transistor which converts 230V mains AC to 12 V AC. Later AC output is rectified using Diode Bridge and capacitor. Later this output is passed through 12V voltage regulator (7812) to produce regularized 12V voltage for charging battery.

CIRCUIT DIAGRAMS

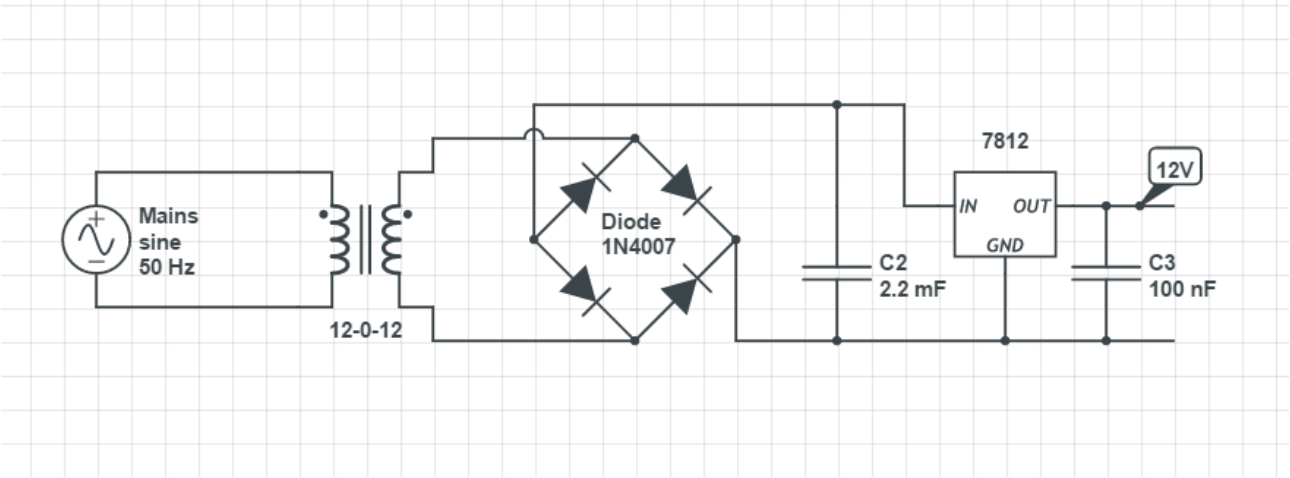
Relay IC for Laser Control



Voltage Regulators



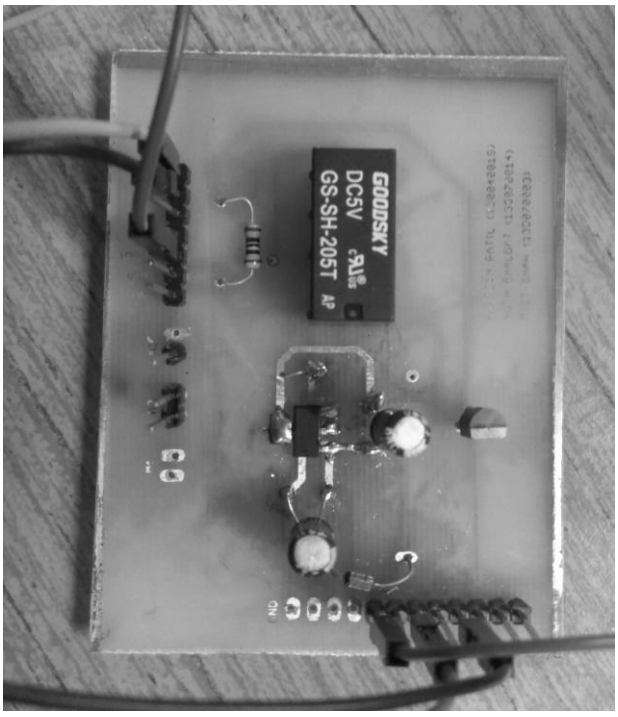
Battery Charging Circuitry

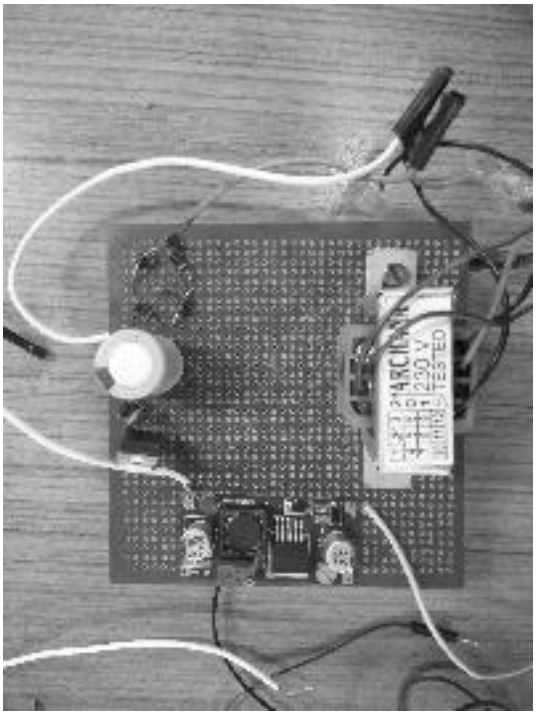


PHOTOGRAPHS

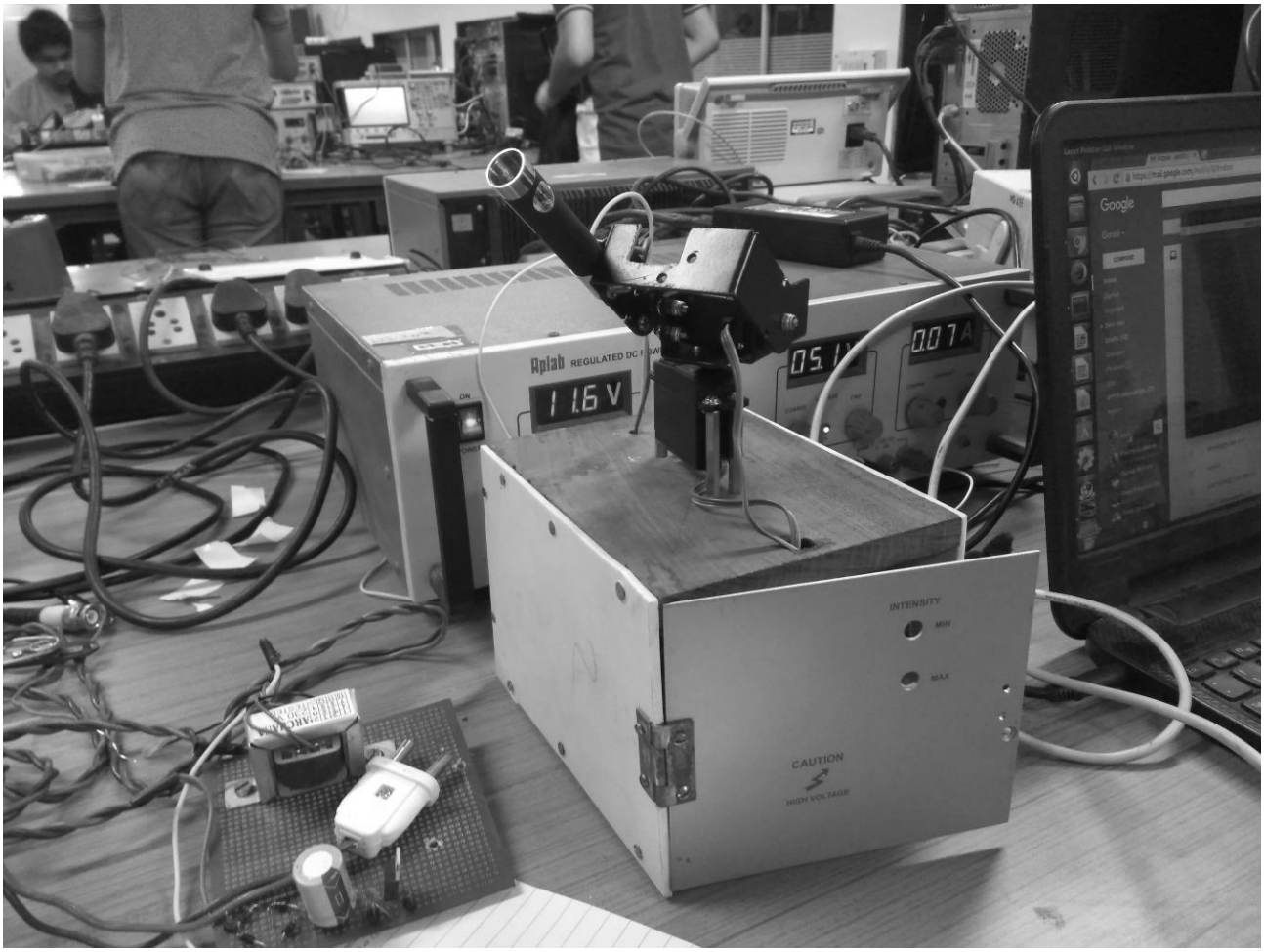
Stellarium software

Assembled PCBs





Pan-tilt mechanism



RESULTS

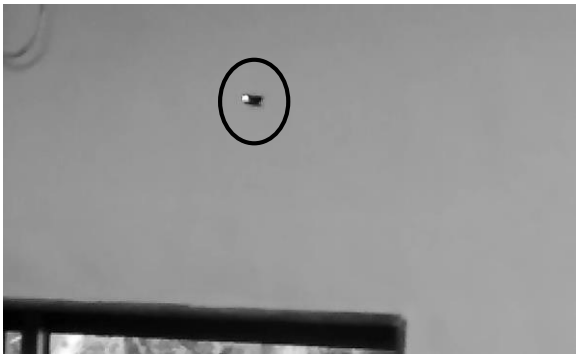
We mapped 11 stars from real time coordinates at 18:34 IST on 11th April, 2016. Locations were marked using stickers. We started our laser pointer device to test the accuracy of our system.

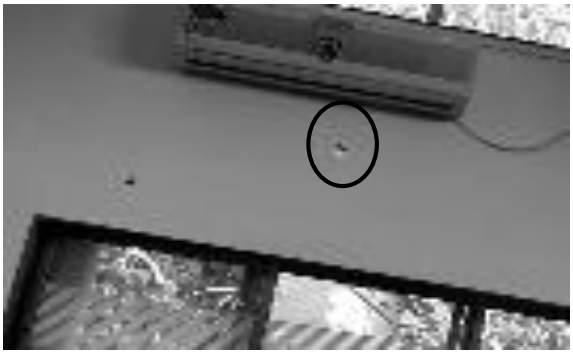
The markers were pointed by the laser sequentially by selecting the respective stars from Stellarium. This showed the successful interfacing as well as accuracy of the mechanism. The error in angle was found to be around 0.68°.

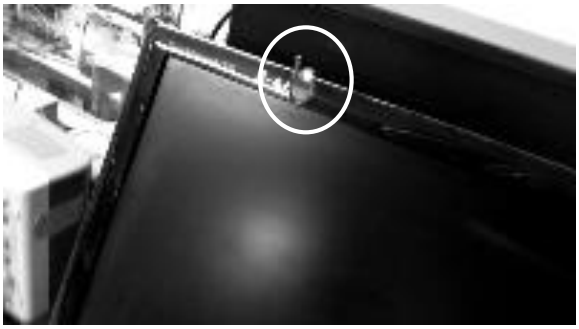
Entire demo video can be found at
https://www.youtube.com/watch?v=KEhirwSOlVI

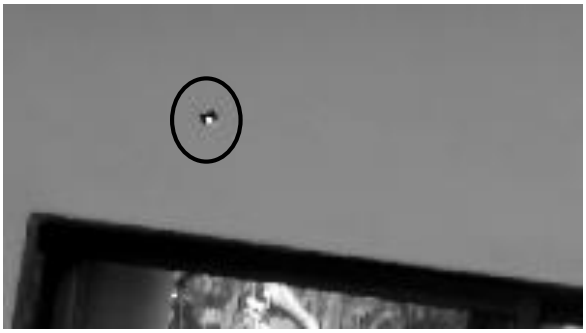
Excerpts from the video are displayed below

Sticker Pointing Demo Images









PROBLEMS FACED

Interfacing (microcontroller)

The integral part of the project was to extract the coordinates from the application engine. For that, we had to first send the coordinates to the socket and then to the USB port. After that, the problem was to comminucate the specific commands. To be able to do that, we defined a set of commands of a specific length of 4. After the commands, specific arguments for that commands were sent, which were to be used to move the pointer to the desired position.

Power management

We had to get an idea of how long the circuit was going to work. So, we first measured the current consumed by the Laser pointer and the servos in the stationary and moving states. From those values, the power consumed was calculated, which helped us to decide the Battery specs to be used (in terms of current and power ratings).

Voltage regulator:

First we were using a 7805 converter. But the current consumed by the laser pointer led to a lot of heating of the regulator, which even a heat sink could not solve. It was also affecting the efficiency of our power system. We used a buck converter in its place, which gave an efficiency of about 80%, also solved our current problem.

CONCLUSION

The complete system is now working perfectly with an angular accuracy of around 0.7 degrees. The interfacing is facilitated with our easy-to-use GUI and the Stellarium application. We have also checked the system with an indoor system of stickers on the wall and also on the outside (though there is a lot of light-pollution here).

So, we hope our application reaches long distances to the people!

Thanks to Prof. Dipankar, Prof. Joseph and our TAs for their guidance, without which this would not have been possible. :)

FUTURE WORK

Automation/ Scripting

Right now, coordinates are sent from the application to the pan-tilt mechanism by pressing CTRL+1. In future, we intend to make an automated script along with an audio to show a "journey" of stars in a planetarium kind of environment. It would serve as an amazing guide to kids and enthusiasts in the village who are deprived of such facilities.

GPS

Right now, the values of Longitude and latitude for Mumbai have been used. But in the future, when the system moves from place to place, a GPS device can be installed on the system, which will give the longitude and latitude automatically.

Telescope Mounting:

Our application framework is pretty generalised. It can be easily ported to telescope systems. So that, the telescope can be pointed to specifics celestial bodies very conveniently. It can be a very powerful aid to the professional astronomers.

REFERENCES

[1] Stellarium developers documentation:
http://www.stellarium.org/doc/head/

[2] Bitstring Library in Python
https://pypi.python.org/pypi/bitstring/3.1.3

[3] Buck Convertor
http://www.learnabout-electronics.org/PSU/psu31.php

[4] 12V battery charger
http://www.instructables.com/id/Make-a-simple-12-volt-power-supply/

[5] https://www.google.co.in/ (of course!)