:: PLATFORM:

BALANCED AND CONTROLLED

BY TEAM VIDYUT

Summer project completed under

Electronics club, IIT Kanpur

Table of contents

1. EXECUTIVE SUMMARY

Basically self-balancing platform consists of platform which is balanced by movement of three motors in opposite direction to the movement of the platform. Arduino Mega process the tilt angles obtained from IMU and give instruction to the respective servo motors to rotate by certain angle depending on its previous position to balance or control the platform. IMU consists of ADXL345 Accelerometer and ITG3200 Gyroscope whose outputs are calibrated properly by using FILTER to give the precise angle. This angle is sent to PID or DCM algorithm which measures the error i.e. how far the current position of platform is from the desired set point (balancing point). The algorithm attempts to minimize the error by adjusting the process control inputs.

2. PROJECT DESCRIPTION

2.1 MOTIVATION

That's fancy speak for the technology that enables balancing. With advent of self-balancing devices, be it Segway, DIY, or TIPI, we five were fascinated with the futuristic scope that self-balancing devices hold, be it flying cars or compact car modules on two wheels, be it self-stabilized and Bluetooth controlled cameras clicking in courteous moves of Hollywood stars or be it a simple self-stabilizing skateboard, controlled by your gestures, the idea of self-stabilizing skateboard controlled by our leg movements did take rounds in our fascinated team.

As of delving deep into vast knowledge pool of self-controlled and stabilized devices, the team felt to get firsthand knowledge of various control mechanisms, IMUs, filters, robust mechanical system, and henceforth, concluded to engineer a manually controlled-cum-self stabilizing platform with three axis of freedom.

2.2 OBJECTIVES AND GOALS

- To demonstrate the techniques involved in balancing a platform.
- To work on precise movements and accurate control of platform, with use of various algorithms and filtering process.
- To understand the working of IMU. IMU work involves understanding the pin configurations of the IMUIMU and configuring the correct libraries for the IMU.
- To identify the correct connections needed for all the peripheral hardware to communicate with the microcontroller.
- Establishing lines of communication with the correct hardware pin addresses will allow for easy identification on how each individual piece of hardware will transmit and process data to and from the IMU.
- To establish the power supply to each electronic components.

3. INITIAL RESEARCH

Without any prior knowledge of how a self-balancing cum camera controlled platform would work, further research was required to get an idea of what was possible for this project. Supplementary investigation was necessary such as to power the system, how to approach and which direction to take towards designing the software, how the platform would accomplish its main task of balancing.

3.1 INITIAL RESEARCH OF CODING IMPLEMENTATION

The software development really boils down to the programming techniques used on this project. Creating a balancing platform with the use of a gyroscope requires that the program keep track of the gyroscope's orientation and attempt to keep the gyroscope level. The conventional motor control method for keeping sensor readings within a certain threshold is the proportional integral derivative, or PID, control loop. PID control typically provides smooth control with minimal overshoot on corrective action. Although there are easier control methods like bang-bang, proportional (P), and Proportional-Derivative (PD), taking the extra time to factor in a smooth integral will be the best payoff for a smooth and efficient system. DCM is very effective algorithm which has authentic calibration of filters and PID whose constants can be manipulated according to the project.

Analog output accelerometers and gyroscopes communicate with a Pulse-Width Modulation (PWM) signal, which most microcontrollers support. Digital output accelerometers and gyroscopes, such as the ones found on sparkfun.com, communicate using standard I2C protocol. There are a few different ways to communicate with motor controllers, which can be categorized by either analog or digital input. Analog input is done via PWM. Digital input can be done a couple ways. The first is by simulating an R/C signal that sets the speed and direction of the motor until specified at another time. The second is to use serial data to communicate the speed and direction of the motors. The main advantage to using serial data is that the microcontroller can communicate with the motor controller with just one serial port. The shifting from control to stabilization mode is accomplished by using a switch between A0 pin of Arduino and a 9 V low current battery .Switching on implies change in Analog value at A0 pin which is used in our code to decide the respective mode.

Arduino obtains tilt angle from IMU by using I2C Interface and then control servo motors using its PWM pins and the frequency of operation of servo motors can be made synonymous with that of IMU.

3.2 INERTIAL MEASUREMENT UNIT:

TILT SENSOR MODULE

Tilt sensing is the crux of this project and the most difficult part as Well.

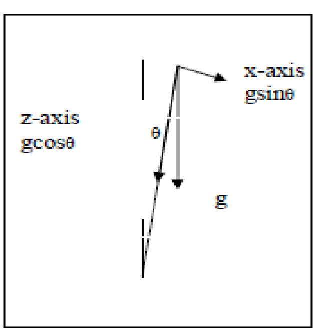
The inertial sensors used are:

- ADXL345 Accelerometer
- ITG3200 Gyroscope

An accelerometer measures the acceleration in specific directions. We can measure the direction of gravity w.r.t the accelerometer and get the tilt angle w.r.t the vertical. It is free from drifts and other errors. The angle of tilt can be measured from the acceleration along x-direction as follows:

$A_x = g \sin \theta$ (θ is the angle w.r.t vertical)

$A_z = g \cos \theta$ (A_x, A_z are accelerations in x and z directions)



For small angles, $A_x = g\,\theta$

So, $\theta = K * A_x$ (K is a constant)

This problem is solved by using a **Gyroscope.** It would measure angular velocity and angle can be found by integrating.

The rate gyro measures angular velocity and outputs a voltage V_g:

$V_g = \omega + f(T) + e_g$

Where f(T) represents the effect of temperature and e_g represents error, which is not known. So the correct formula is:

Angular Velocity $\omega = (V_g - V_{at\ zero\ tilt})$/Sensitivity [in V/(deg/s)]

But this approach fails at slow angular velocities due to gyroscopic drift (small errors in slow velocities integrate and accumulate into a big error resulting in drift).

In reality, we do not have static conditions, but dynamic conditions. So in case of dynamic accelerations, the accelerometer and gyroscope outputs are combined together using the **Kalman Filtering Method**

Or Complimentary Filtering Method.

3.3 KALMAN AND COMPLIMENTARY FILTERS

The Kalman filter operates recursively on streams of noisy input data to producea statistically optimal estimate of the underlying system state.

The algorithm works in a two-step process :

In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. Because of the algorithm's recursive nature, it can run in real time using only the present input measurements and the previously calculated state; no additional past information is required.

The Kalman filter uses a system's dynamics model (e.g., physical laws of motion), known control inputs to that system, and multiple sequential measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using any one measurement alone. As such, it is a common sensor fusion and data fusion algorithm.

3.4 FREEIMU LIBRARIES

IMU can be easily used on Arduino compatible boards using the Arduino FreeIMU library which implements sensor fusion MARG orientation filter (a very efficient filter) enabling you to do easy and straightforward orientation sensing using tri axis accelerometer and gyroscope. It is very easy to implement and the functions used in this library is easily comprehended. We found it easily compatible with our 6DOF IMU.

Yaw drift was also major issue which posed serious challenge as our yaw motor vibrated after almost every 0.5 seconds .The drift was reduced by using an algorithm described in next section.

In the library we find code for reading the raw data from the ITG3200 and ADXL345 but not for the sensor fusion. For fusing their data, we use a very powerful DCM algorithm described below.

3.5 DCM ALGORITHM / PI CONTROL UNITS

The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output, the final form of the PID algorithm is:

$$\mathrm{u(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

Where:

Kp: Proportional gain, a tuning parameter

Ki: Integral gain, a tuning parameter

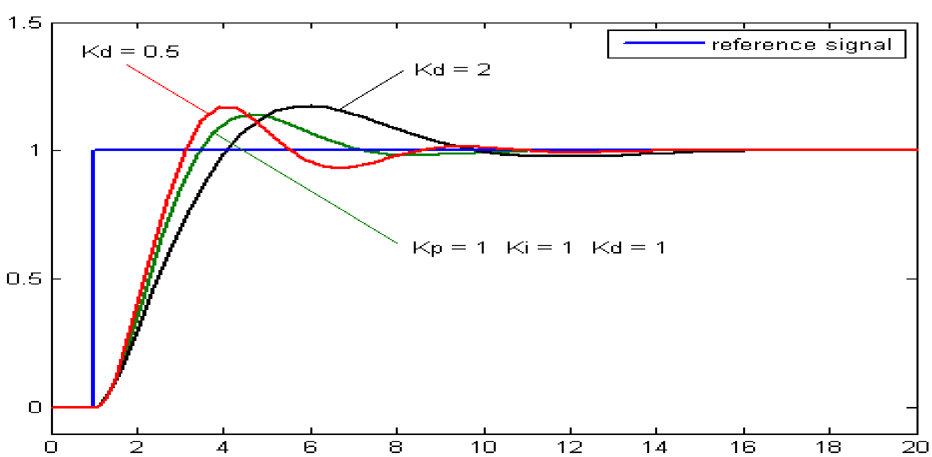
Kd: Derivative gain, a tuning parameter

e: Error

t: Time or instantaneous time (the present)

We tune the PID controller by varying the constants K_p, K_d and K_i and optimizing them.

Process Variable (PV) vs time:

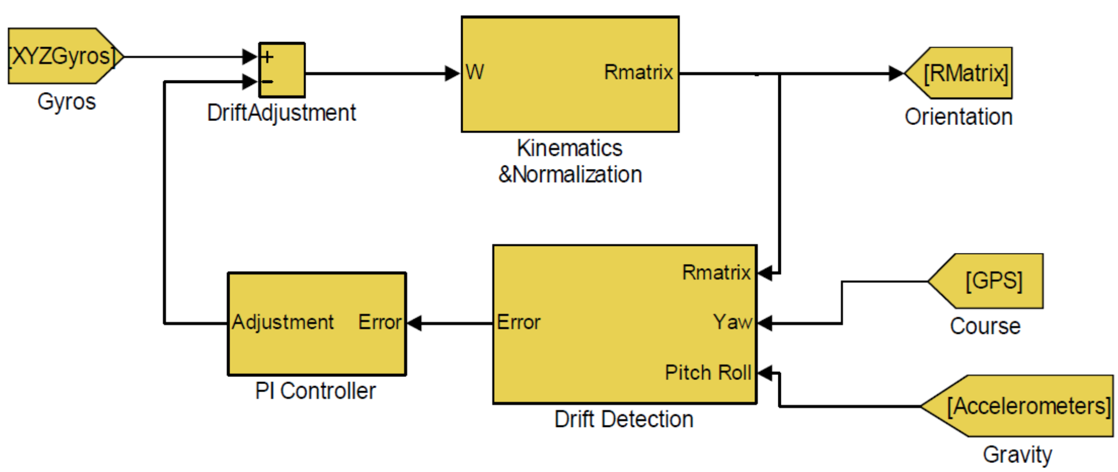


DCM can be thought of a strong and robust algorithm for precise control of servo motors .It uses Euler angles , Direction Cosine Matrix(DCM) and Quaternion approaches. It has inbuilt Filters and Proportional – Integral Control Units which in turn is very effective in giving calibrated outputs.

The DCM algorithm uses Proportional – Integral Control units which are almost similar to PID except for the absence of Integral term.

The tuning values , proportional constant (K_p) and derivative constant (K_d) changes in the code depending on how far the position of platform is from setpoint (balanced point).

Recognizing that numerical errors, gyro drift, and gyro offset will gradually accumulate errors in the DCM elements, we use reference vectors to detect the errors, and a proportional plus integral (PI) negative feedback controller between the detected errors and the gyro inputs, to dissipate the errors faster than they can build up. GPS is used to detect yaw error, accelerometers are used to detect pitch and roll.



On using this algorithm, we came to realise that Yaw angle accumulate leading to a significant value after some time inspite of no change in orientation. We analysed the Yaw output and accordingly made a simple algorithm that is very effective in reducing drift. What we basically did was to ignore extremely small change of orientation angles in every cycle. These angles being in the order of .01 degrees are almost impossible to replicate by the physical motion of hands, leading to quite stable data without loss of much accuracy.

The code for which is below:

```
#include <ADXL345.h>
#include <bma180.h>
#include <HMC58X3.h>
#include <ITG3200.h>
#include <MS561101BA.h>
#include <I2Cdev.h>
#include <MPU60X0.h>
#include <EEPROM.h>


//#define DEBUG
#include "DebugUtils.h"
#include "CommunicationUtils.h"
#include "FreeIMU.h"
#include <Wire.h>
#include <SPI.h>
#include <Servo.h>

int raw_values[9];
//char str[512];
float ypr[3]; // yaw pitch roll
float val[9];
```

```
Servo servo1;
Servo servo2;
Servo servo3;
float old[3];
float apple[3];
int flag1;
int flag2;

// Set the FreeIMU object
FreeIMU my3IMU = FreeIMU();

void setup() {
  Serial.begin(9600);
  Wire.begin();
  servo1.attach(8);
  servo2.attach(9);
  servo3.attach(10);
  flag1=1;
  flag2=1;
  apple[0]=apple[1]=apple[2]=0;
  old[0]=old[1]=old[2]=0;

  delay(5);
  my3IMU.init(); // the parameter enable or disable fast mode
  delay(5);
}

void loop() {

  my3IMU.getYawPitchRoll(ypr);
  flag1 = analogRead(A0);
  if (flag1<500)

 {  flag2 = 1;
 }

 else
 {
   flag2 =0;
 }

  if(ypr[0]-old[0]<0.16 && ypr[0]-old[0]>-0.16)
  {
    Serial.print("Yaw: ");
```

```
  Serial.print(apple[0]);
}
if((ypr[0]-old[0]<90 && ypr[0]-old[0]>=0.16) || (ypr[0]-old[0]>-90 && ypr[0]-old[0]<=-0.16))
{
  Serial.print("Yaw: ");
  apple[0]=apple[0]+ypr[0]-old[0];
  Serial.print(apple[0]);
  if(apple[0]>-90 && apple[0]<90)
  {
    if (flag2==1)
   {  servo1.write(90+apple[0]);
   }
   else
   {
     servo1.write(90-apple[0]);
   }
    Serial.print("***");
  }
}
if(ypr[0]-old[0]>=90 || ypr[0]-old[0]<=-90)
{

  Serial.print("Yaw: ");
  Serial.print(apple[0]);

}

if((ypr[1]-old[1])>0.12 ||   (-ypr[1]+old[1])>0.12)
{
  Serial.print("  Pitch: ");
  apple[1]=ypr[1];
  Serial.print(apple[1]);
        if (flag2==1)
   {  servo2.write(92.5-apple[1]);
   }
   else
   {
     servo2.write(90+apple[1]);
   }
}
else
{
  Serial.print("  Pitch: ");
  Serial.print(apple[1]);
```

```
  }
  if((ypr[2]-old[2])>0.12 ||   (-ypr[2]+old[2])>0.12)
  {
    Serial.print("  Roll: ");
    apple[2]=ypr[2];
    Serial.print(apple[2]);
          if (flag2==1)
     {  servo3.write(87+apple[2]);
     }
     else
     {
       servo3.write(87-apple[2]);
     }
  }
  else
  {
    Serial.print("  Roll: ");
    Serial.print(apple[2]);
  }
  old[0]=ypr[0];
  old[1]=ypr[1];
  old[2]=ypr[2];
  Serial.print("       OYaw: ");
  Serial.print(ypr[0]);
  Serial.print("  OPitch: ");
  Serial.print(ypr[1]);
  Serial.print("  ORoll: ");
  Serial.print(ypr[2]);
  Serial.println("");
  delay(10);


}
```

3.6 POWER SUPPLY

Designing the power supply for this project took serious considerations on what exactly to look for, how to layout the power supply to affect the rest of the system and what exact voltage and milliamp hours needed.

For powered the whole mechanism, we have used a lead-acid rechargeable battery having following specifications:-

Output voltage-6v

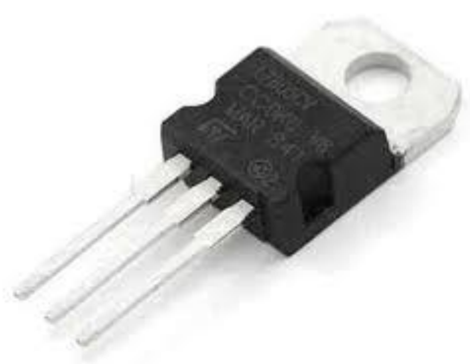
Ampere hour-4.5

Net weight-0.73±0.015 kg

Dimension-70*47*101

We have used this battery only because of its property of recharge and easily available as it is so cheap.

We power the servo motors directly by using lead acid battery and arduino by using a voltage converter LM7805.



A LM7805 Voltage Regulator is a voltage regulator that outputs +5 volts.

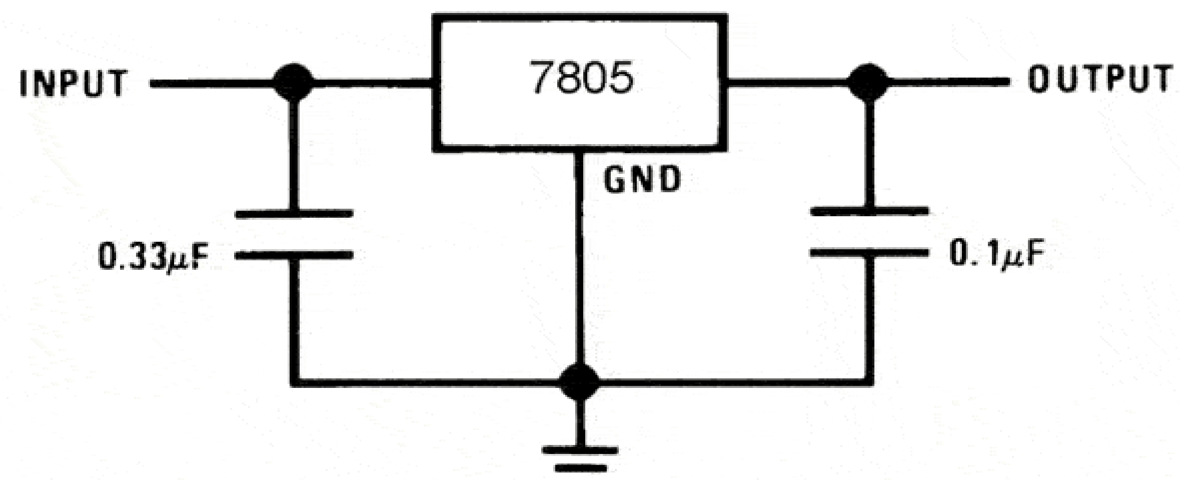
An easy way to remember the voltage output by a LM78XX series of voltage regulators is the last two digits of the number. A LM7805 ends with 05, thus it outputs 5 volts. The 78 part is just the convention that the chip makers use to denote the series of regulators that output voltage is positive. The other series of regulators, the LM79XX, is the series that output voltage is negative.

The LM7805, like most other regulators, is a three-pin IC.

Pin 1(input pin): The input pin is the pin that accepts the incoming DC voltage, which the voltage regulator will eventually regulate down to 5volts.

Pin 2(ground): Ground pin establishes the ground for the regulator.

Pin3 (output Pin): The output pin is the regulated 5 volts DC.



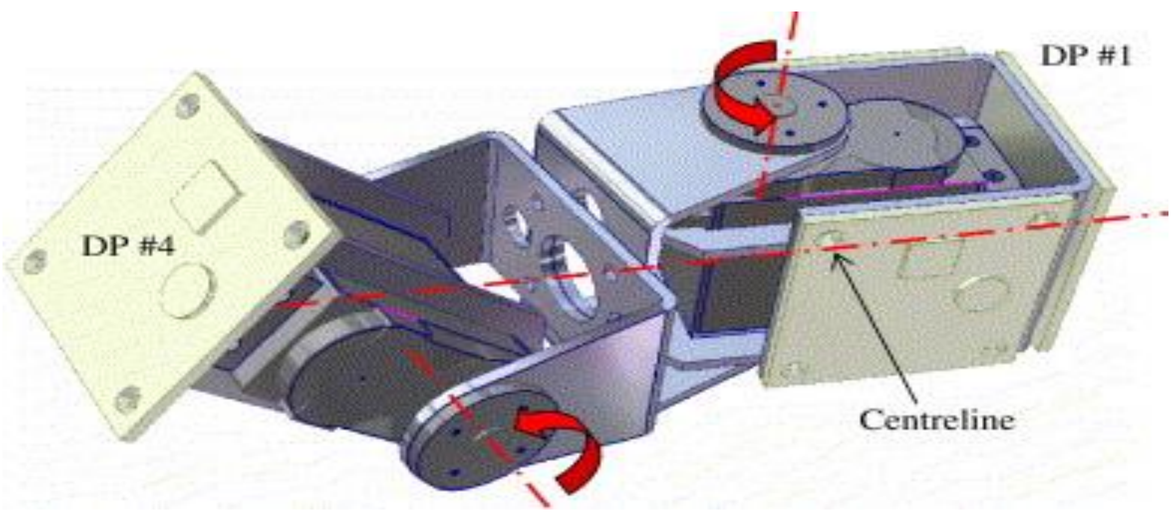
We have also used 9v hi-watt battery to give external interrupt to arguing for control and stabilizing mode.

COMPONENTS

1. IMU Digital Combo Board – 6DOF
2. Arduino Mega
3. Three DC Servo motors (High Torque Metal Geared)
4. Motor Connectors
5. Perspex and acrylic material for mechanical setup
6. 6 Volt Rechargeable Battery

4. PROTOTYPE

In order to answer some questions about designing the self-balancing platform, it was decided to first design a small scale prototype. This section details on findings about hardware selection, and software approaches to solving the problem. In addition to the control scheme, the hardware selection was re-evaluated and re-imagined at almost every aspect of the hardware selection. From hardware design to software design, the group has undergone several important revisions that have helped obtain a better understanding for designing the final product:



4.1 Hardware Selection

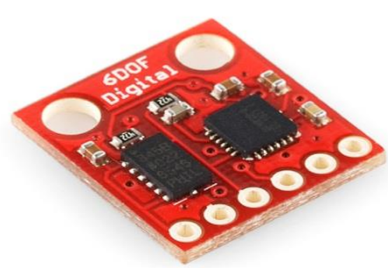
4.1.1 6DOF IMU

IMU is an Inertial Measure Unit (sensors and hardware filter circuitry) this hardware consists of a 3 axis accelerometer (AdXL345) and three gyro sensors (ITG3200),

Accelerometer gives the component of acceleration (g) along the three axes. Gyroscope gives the component of angular velocity along the three axes which is then integrated to find angle.

Features:

- Tiny!
- Two mounting holes
- ADXL345 Accelerometer
- ITG-3200 gyro
- 3.3V input
- I2C interface



4.1.2 MOTORS

These servo motors would provide the proper amount of power required to maintain its own weight as well as the weight of all the electrical and physical hardware while at the same time still maintaining balanced equilibrium.

The shaft can be easily angled between 0 to 180 degrees. This wire is given a pulse application for a specified duration, which in turn controls the angle of the shaft in a particular position for a certain point of time. This modulation is famously referred to as the PWM (Pulse Width Modulation). The servomotor expects a coded signal every few seconds. The duration of the pulse determines the angular degree of the shaft.

Servo motor that we have selected has torque of 15kg/cm, operating voltage of 4.8 V to 6 V, speed of 60degree/0.20sec, dimensions: length-49.3mm, width-25.4mm, height-42.9mm and weight about 80 g.

4.2. Mechanical Model

4.2.1 Horizontal or Vertical

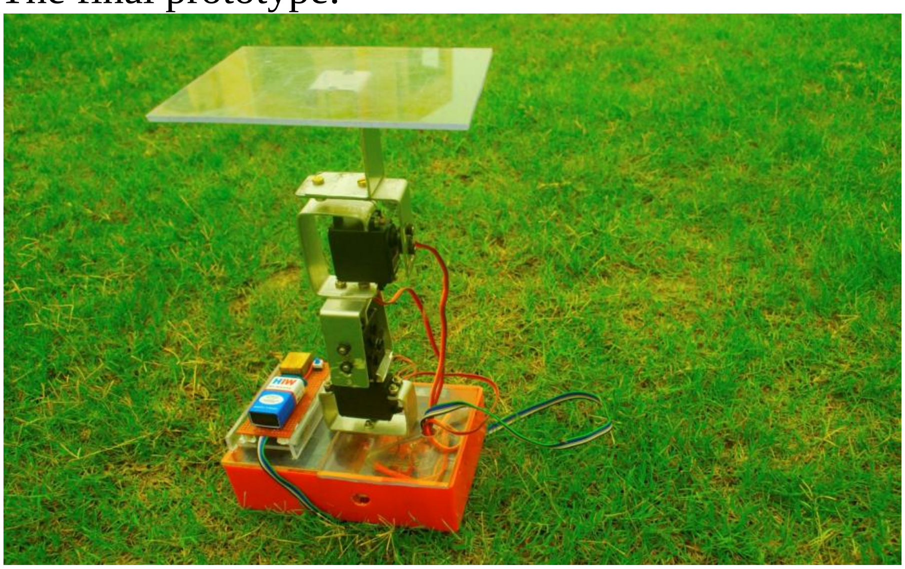
Based upon discussion over several prototypes we finally agreed upon two of them which seemed most suitable for our project considering mainly the form factor and feasibility. One of them is having the base and the platform on the same level and the other having a more conventional type of design having platform vertically above the main base. Considering aesthetics and being popular among most of the team members we finally agreed on to the second design. Placing motors perpendicular to each other gives us the advantage of having three degrees of rotation. We will be using self-made Aluminums brackets to hold the motors in their place.

4.2.2Detailed Overview Prototype

The first motor is attached to the base and capable to rotate other two motors and the upper platform with camera about z direction. The other two motors gives the freedom to rotate the platform about x and y axis. Main base will consists of Arduino Mega, a battery and IMU circuit board that can be detached from the base to establish both control and stabilizing. While the IMU board is attached to the base, we can stabilize the upper platform with a camera or any other object placed on the platform. When we separate the IMU board we are able to control the camera holding platform about of the 3 axis, just by pressing a button.

We used a 9v small battery to generate analog voltage at analog pin of arguing to trigger the control and stabilizing process.

The final prototype:



TEAM MEMBERS:

AARSHEE MISHRA
ANKIT RAJ
ISH KUMAR JAIN
PIYUSH JAIN
SHUBHAM KUMAR

MENTORED BY:

RANVIJAY SINGH
SONU AGARWAL
VIPUL GUPTA