# Vertigo - wall climbing robot

*Team: Tarantula     Members: Athul. A , Anuj Verma, Chaitanya Jain  Team ID: 107*
*Video:* https://drive.google.com/open?id=0BxFCZy2DRzJHelZmU1RRVC1hbUk

**Objective:** To design a wall climbing robot which is agile on wall and capable of performing transition from ground to wall. The project idea was largely inspired by project VERTIGO by ETH Zurich and Disney.

**Learning Objectives:**

- Learning about various fabrication processes like CNC/manual milling and implementing them.
- To understand and implement designs and choosing materials that provide good strength is to weight ratio, costs less and are easier to manufacture.
- Develop strong cadding/visualisation skills and bringing out effective alternatives from materials easily available in local market.
- Understanding basics of microcontrollers, Brushless-DC motors,servos and bluetooth module like HC-06.
- Usage of IMU and acquiring basic knowledge about complementary/kalman filters.

**Project implementation:**

Project was divided into 4 stages:

1. CADDing and manufacturing parts
2. Assembling mechanical structure
3. Individually testing electronic components and assembling them
4. Running the vehicle with the combined code.

**CADDing and manufacturing:**

Some precalculation(basic) :
Estimated static thrust from 2 BLDC motors using 8*4 props with 1450 kv rating : 2.4 kg
( static thrust can be fairly used for all cases as vehicle speed is less)
Estimated weight : 1.45 kg
Frictional coefficient between wall and rubber grip = 0.6 approx ( range 0.6- 0.8)
Angle of BLDC motors with plane of vehicle during translation = 10 deg

So safety factor :  2.4*0.7* cos(10 deg) /(1.45) =1.14 ( relatively less)

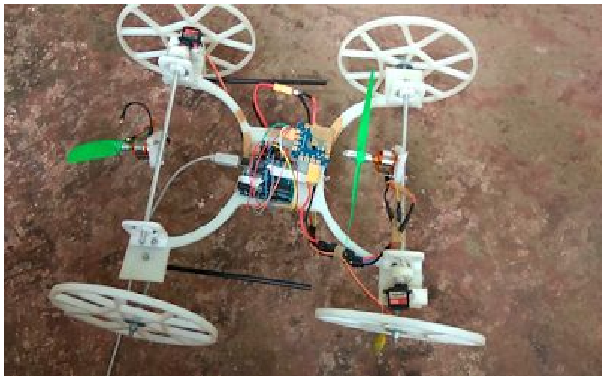
The whole CAD of the vehicle was developed on solidworks. The major challenge was to design the rotating part on which BLDC motors could be fixed and the steering part.

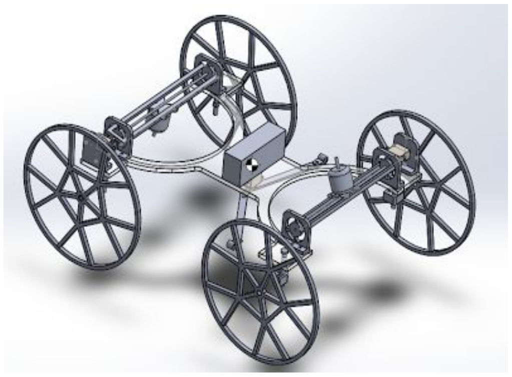
The rotating part was designed as shown with a rod in the centre bearing weight and a freely rotating part to which motors are attached. Steering part was largely inspired from *Ackermann mechanism.*

Choosing the appropriate material was another task. Based on our experience with handling materials like delrin, alloys of aluminium, nylon-6,acrylic and polycarbonate it was decided that chassis should be made of material that has greater *fracture toughness and less weight.* Accordingly Delrin and nylon were chosen due to easier availability and lesser weight.

All parts used in model were manufactured using CNC/Manual milling and lathe. G-Code was learnt and implemented in easier parts, relatively tough parts were done using 3-axis CNC milling machine that bypassed G-Code.

Final CAD of vehicle:
https://drive.google.com/file/d/0BxFCZy2DRzJHUDdXVVN4dElscnM/view?usp=sharing


Actual Model:steering part was completed later


CAD: solidworks model

**Assembling Mechanical structure:**

Assembling mechanical structure presented a lot of problems.

1) The propeller size chosen and space given were not matching due to a visualisation error as props were not included in CAD. This led to a significant reduction in thrust and bringing the safety factor near to one.
2) Since no shafts were used to connect wheels there was significant bending under load.Also wheels were passive so they need to rotate freely making it difficult to fix *Solution*: adding two bolts to each wheel at a distance instead of one and replaced simple rods with threaded rods with large washers and nuts on both ends of wheel.
3) The rotating frame was not getting attached to servos and ultimately we had fuse two delrin pieces together under controlled flame to complete it.

**Testing electronic components and bringing everything together:**

Specs of components used : D282 turnigy 1450 kv BLDC motors with 20A UBEC turnigy ESC, Avionic 2kg stall torque servos , ESC power distribution board, LiPO 1.3 Ah 20C. Arduino UNO HC-06, MPU-6050 ( IMU module)
All components were individually tested and then tested as a combination. All three servos were connected to Arduino UNO itself and ESC's powered uno , bluetooth module and MPU6050(IMU).

ESC's were connected to power distribution board which is connected to lipo. All instructions to bluetooth was given using *PuTTy* terminal.

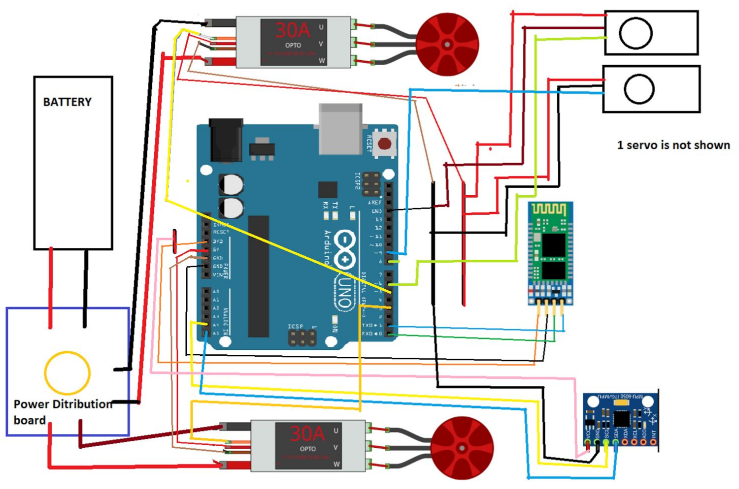
The main bug was observed in the final assembly of one of the motors and esc's which eventually led to battery failure which couldn't be replaced due to time constraint.

Using MPU6050 was another tougher challenge. The method of using complementary filter were a low pass filter is used with accelerometer and high pass with gyroscope gave error of nearly 10 deg ( horizontal was set using spirit level and compared ) while Kalman filter was used with help of library. The idea of predicting present state from a priori state on a bayesian approach and Kalman gain calculation using matrices were vaguely understood.

The idea of using MPU was during ground to wall transition where one needs to be known the exact angle of vehicle with horizontal ground as rear motors are oriented towards wall and front motors upwards which later changes when vehicle approx makes 60 deg( 60 deg a very biased approx and not based on calculation )

**Weekly progress of team:**
https://drive.google.com/open?id=10_Z2PYqpLc_p7Xd2UmbB5xN9-5Xzj5Zdf7Ccbg0mT_I

Circuit:



Complete Code:-

https://drive.google.com/file/d/0BxFCZy2DRzJHbWZKeGN4ZzZZQ1k/view?usp=sharing

Libraries used: Wire.h, Kalman.h, Servo.h

Functions for initialising IMU ( MPU6050) :

```
void Setup_imu()
{
  #if ARDUINO >= 157
  Wire.setClock(400000UL); // Set I2C frequency to 400kHz
#else
  TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C frequency to 400kHz
#endif

  i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
  i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz Gyro filtering, 8 KHz sampling
  i2cData[2] = 0x00; // Set Gyro Full Scale Range to ±250deg/s
  i2cData[3] = 0x00; // Set Accelerometer Full Scale Range to ±2g
  while (i2cWrite(0x19, i2cData, 4, false)); // Write to all four registers at once
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis gyroscope reference and disable sleep mode

  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
    Serial.print(F("Error reading sensor"));
    while (1);
  }

  delay(100); // Wait for sensor to stabilize


  /* Set kalman and gyro starting angle */
  while (i2cRead(0x3B, i2cData, 6));
  accX = (i2cData[0] << 8) | i2cData[1];
  accY = (i2cData[2] << 8) | i2cData[3];
  accZ = (i2cData[4] << 8) | i2cData[5];

#ifdef RESTRICT_PITCH // Eq. 25 and 26
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else // Eq. 28 and 29
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  kalmanX.setAngle(roll); // Set starting angle
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
 // compAngleX = roll;
//  compAngleY = pitch;

  timer = micros();
  Serial.print("done initialising");
}
```

Function for getting angle based on char input given as  x & y for x-y axis.

```
double Angle( char co_ordinate) // returns angle along x and y axis of vehicle i.e imaging vehicle in 2-D ...transition takes palce in
x-y plane
{
    /* Update all the values */
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]);
  accY = ((i2cData[2] << 8) | i2cData[3]);
  accZ = ((i2cData[4] << 8) | i2cData[5]);
  gyroX = (i2cData[8] << 8) | i2cData[9];
  gyroY = (i2cData[10] << 8) | i2cData[11];
  gyroZ = (i2cData[12] << 8) | i2cData[13];

  double dt = (double)(micros() - timer) / 1000000; // Calculate delta time
  timer = micros();

#ifdef RESTRICT_PITCH
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else // Eq. 28 and 29
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  double gyroXrate = gyroX / 131.0; // Convert to deg/s
  double gyroYrate = gyroY / 131.0; // Convert to deg/s

#ifdef RESTRICT_PITCH
  // This fixes the transition problem when the accelerometer angle jumps between -180 and 180 degrees
  if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
  //  compAngleX = roll;
    kalAngleX = roll;
    gyroXangle = roll;
  } else
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // Calculate the angle using a Kalman filter

  if (abs(kalAngleX) > 90)
    gyroYrate = -gyroYrate; // Invert rate, so it fits the restriced accelerometer reading
  kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
#else
  // This fixes the transition problem when the accelerometer angle jumps between -180 and 180 degrees
  if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 && kalAngleY < -90)) {
    kalmanY.setAngle(pitch);
    //compAngleY = pitch;
    kalAngleY = pitch;
    gyroYangle = pitch;
  } else
    kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt); // Calculate the angle using a Kalman filter

  if (abs(kalAngleY) > 90)
    gyroXrate = -gyroXrate; // Invert rate, so it fits the restriced accelerometer reading
  kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // Calculate the angle using a Kalman filter
#endif

//  gyroXangle += gyroXrate * dt; // Calculate gyro angle without any filter
//  gyroYangle += gyroYrate * dt;
  gyroXangle += kalmanX.getRate() * dt; // Calculate gyro angle using the unbiased rate
  gyroYangle += kalmanY.getRate() * dt;

 // Reset the gyro angle when it has drifted too much
if (gyroXangle < -180 || gyroXangle > 180)
  gyroXangle = kalAngleX;
if (gyroYangle < -180 || gyroYangle > 180)
  gyroYangle = kalAngleY;

if ( co_ordinate=='x')
  return gyroXangle;
else
  return gyroYangle;

}
```

For initialising and starting BLDC motors ...a 0.8 ms signal needs to be sent to start in routine mode and 2ms in program mode:

```
void Initialise_esc()
{
  esc1.attach(esc_pin1);
  esc2.attach(esc_pin2);
  esc1.writeMicroseconds(800);// 800 microseconds =0 throttle sets to routine start mode
  esc2.writeMicroseconds(800); // for turnigy ESC used ideally a 1ms to 2ms signal should work ...but we found giving 0.8 ms was
equivallent to zero throttle
  Serial.print("esc initialised");
}
void Initialise_servo()
{
  servo[0].attach(servo1_pin);
  servo[1].attach(servo2_pin);
  servo[2].attach(servo3_pin);
  for(int i=0;i<3;i++)
  {
    servo[i].write(0);
    delay(10);
    servo[i].write(0);
    delay(10);
    servo[i].write(0);
  }
  Serial.println("servo-initialised");
}


void bldc_start(char choice)
{
  int value;
    if(choice=='o')
    {
      value=1500;
      esc2.writeMicroseconds(value);
      esc1.writeMicroseconds(value);
```

```
   }
   else if (choice=='s')
    {
     esc1.writeMicroseconds(800);
     esc2.writeMicroseconds(800);
     return;
    }
}
```

Function for transition from ground to wall :

```
void Transition()
{
  Serial.println("Transition begins");
  double gx,gy,gx1,gy1;
   Forward(-10);
  delay(1500); // guess work here for delay...for bot to come back a sufficient distance

  for(int angle=0; angle<=upright;angle=+2)
  {
    servo[0].write(angle);
    delay(10);
  }


  for(int angle=0; angle<=wall_push;angle=+2)
  {
    servo[1].write(angle);
    delay(10);
  }

  double  Rate_angle=((gx-gx1)+(gy-gy1))/2;
  start:
  gx=Angle('x');
  gy=Angle('y');
  delay(50); // Another experimental fine tuning required here ..90/3000~0.03 degree per ms so 1.5 deg per 50 ms...3000 is assumed
from video.
  gx1=Angle('x');
  gy1=Angle('y');
  if(gx<60 and gy<60)
  {

   servo[0].write(servo[0].read()+Rate_angle);
   delay(5);
   servo[0].write(servo[0].read()+Rate_angle);

   servo[1].write(servo[0].read()+Rate_angle);
   delay(5);
   servo[1].write(servo[0].read()+Rate_angle);
   goto start;
  }

  else if (gx>60 and gy>60 and gy<90 and gx<90)
  {
      for(int angle=0; angle<=-90
      ;angle=+2)
    {
```

```
    servo[1].write(angle);
    delay(10);

  }

  servo[1].write(servo[0].read()+Rate_angle);
  delay(5);
  servo[1].write(servo[0].read()+Rate_angle);
  goto start;
 }

}
```

## References:

https://drive.google.com/open?id=1jSflpDVbvZIIn7BvaEqZZko-0CfosDXJomIAqFng4Gk

Team : Tarantula   ID: 107