Components:

Microcontroller: ATmega32

Gyroscope: mpu-6050 (in the video I used a LISY300)

Accelerometer mpu-6050 (in the video I used a BMA180)

Encoder: Ball mouse

Motor Controller: Pololu TB6612FNG

Motors: Old DC motors madi in Japan :) SM - E048  
- 12500 RPM without load.  
- Current: 0.1 A Without load 1.5 A When the wheels are changing the direction of rotation, or when going uphill.  
  
Battery: Electric drill 9.6V NiMH

Wheels: Toy tractor

Geer ratio: 1:8

Bluetooth-SerialPort converter: hc - 05  
  
Complementary filter (For the angular velocity I used only the Gyro)  
With the use of euler angles balancing end turning in inclined surface is more easier

Remote Controller:

ATmega328P

PS2 Analoge Stick

Bluetooth-SerialPort converter: hc - 05

//--------------------------------------­---------------------------//

The requested velocity when the robot is standing still is:  
ReqVelocity = RobotPosition(mm) \* positionKP

poitionKP = 0.008  
  
The requested angle is calculated by a PI controller. The setpoint for this controller is ReqVelocity. The measured velocity is filtered by a Low pass filter. the output of the controller is the ReqAngle

velocityKP = 2

velocityKI = 1.64

PIDoutput is constrained between = -20,20 this is the maximum angle that the robot would try to reach when accelerating.

The maximum value whereby the ‘I’ can change is 6.6. So when calculating the integrator value then error can constrained between +-maxIchange/velocityKI  
  
The requested PWM for the motors is calculated by a PD controller. The setpoint for this controller is ReqAngle. To eliminate the derivative spikes D is calculated from the measured angle change, not from the error.  
angleKP = 40  
angleKD = -4.5 (this is minus because it isn’t calculated from the error)  
PIDoutput is constrained between -255, 255  
The output of this controller is also filtered with a Low pass filter  
  
To keep the robot’s orientation another PWM value is calculated and then added to one of the motors and subtracted from the other.  
orientationKP = 5  
orientationKD = 0.1  
  
the whole program runs in a loop 100 times in a second.  
If the value of the filters would changed, than the PID’s value would also need to be changed.   
The low pass filter’s code with fixed timing is:  
filteredARRAY[0] = rawDATA\*LPFgain + filteredARRAY[1]\*(1-LPFgain);  
filteredARRAY[1] = filteredARRAY[0];   
velocity LPFgain = 0.35 //0.45  
PWM LPFgain = 0.2 //0.45  
reqeustedVelocity LPFgain = 0.05 (this is to smooth the setpoint changes)