REACTION-WHEEL STABILIZED BICYCLE ROBOT

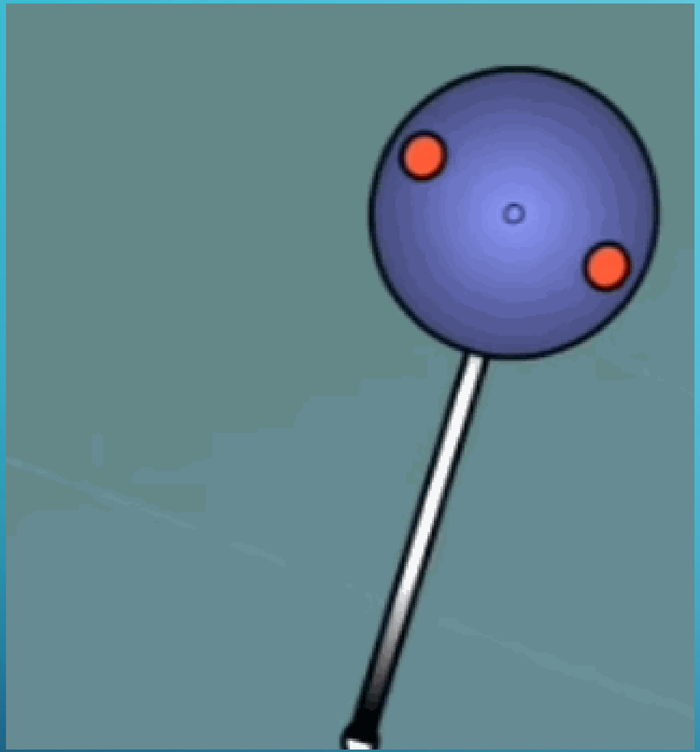
INTRODUCTION TO ROBOTICS - EGN4060C

ERIC BAKER

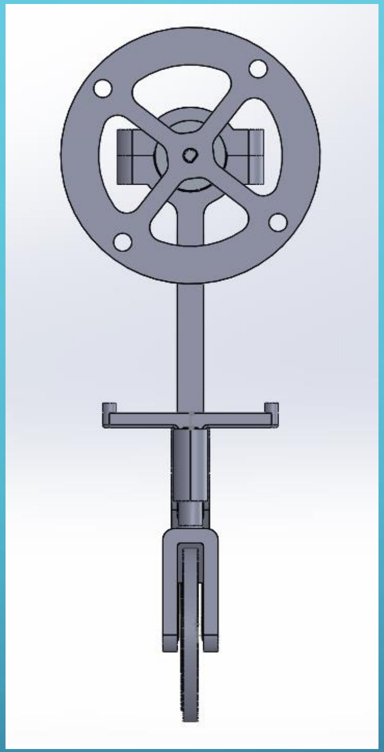
BRETT BURGESS

SALOMON HASSIDOFF

INTRODUCTION



Reaction-wheel pendulum animation



Frontal view of CAD assembly

- Our project consist of a bicycle robot that stays upright without toppling over by using a reaction wheel.
- The system is modeled as an inverted pendulum system. Like most inverted pendulum systems, this project heavily focused on dynamics and controls.
- Reaction-wheels work based on conservation of angular momentum. When the systems tips over, the reaction-wheel imparts a counter-torque to rotate the system back to equilibrium.

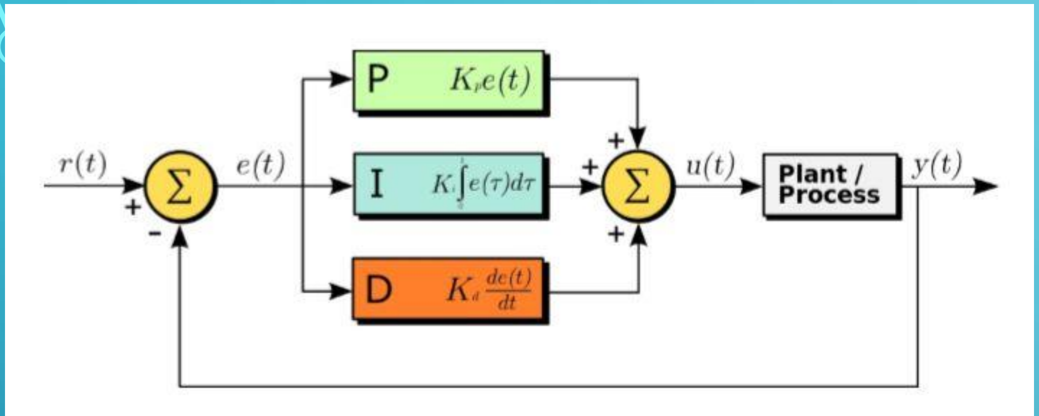
ROBOTS THAT MAKE USE OF REACTION-WHEELS

Murata Boy:
https://www.youtube.com/watch?v=G3_0OzaoQ00

Murata Girl:
https://www.youtube.com/watch?v=IAWYgZbUvHs&t=96s

Cubli:
https://www.youtube.com/watch?v=n_6p-1J551Y

SOFTWARE DESIGN



Control loop

```
// PID and controls values
double Kp = 3;         // Proportional gain
double Ki = 0.01;      // Integral gain
double Kd = 0.5;       // Derivative gain
double setpoint;       // Desired value in degrees
double input;          // Inclination angle from IMU
double output;         // Reaction wheel spin for counter torque
PID myPID(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);  // PID instance
```

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	No Change

PID Tuning

Our algorithm utilizes a PID controller inside a feedback loop to maintain a desired tilt angle.

- r(t): desired output
- y(t): actual output
- e(t): error
- u(t): controller output / system input

The PID parameters are adjusted until optimal system behavior is achieved.

Proportional: looks at present error to provide quick response
Integral: looks at past error to eliminate steady-state error
Derivative: predicts future error to prevent overshoot

IMPLEMENTATION

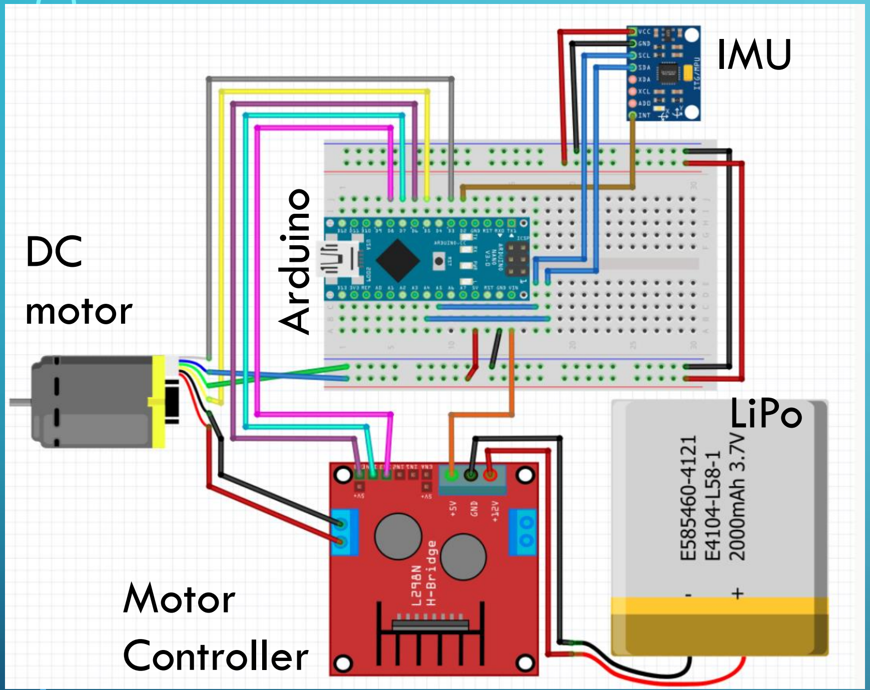
- Gyro is prone to drift.
- Accelerometer is sensitive to vibration.
- A Kalman filter is an optimal estimation algorithm that fuses the measurements for a more accurate estimate of the position.

```
set PID parameters (kp, ki, kd)
set desired roll angle
set max motor voltage

loop
{
    function getIMUangle()
    {
        read gyro data
        apply Kalman filter

        return roll angle
    }

    error equals desired angle minus measured angle

    P equals kp times error
    I equals ki times integral of error
    D equals kd times derivative of error

    controller output equals sum of P, I, D terms

    function WriteDriverVoltage(output, max motor voltage)
    {
        convert output to PWM signal
        set direction of motor
    }
}
```

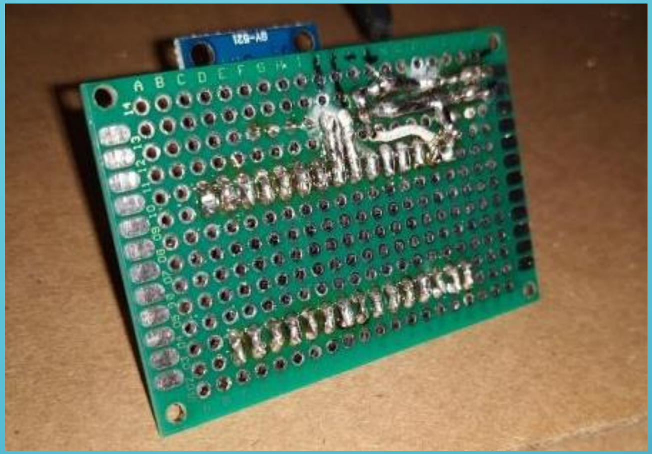
Pseudocode

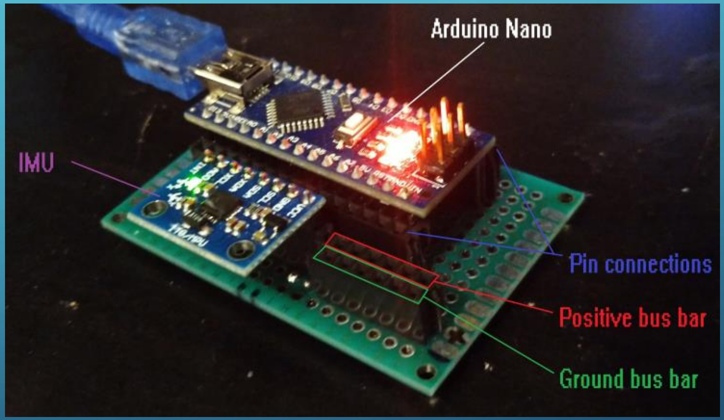
Demo of IMU output

ELECTRONICS HARDWARE DESIGN



Circuit Schematic





The Arduino and IMU were soldered onto a circuit board to save on space and weight.

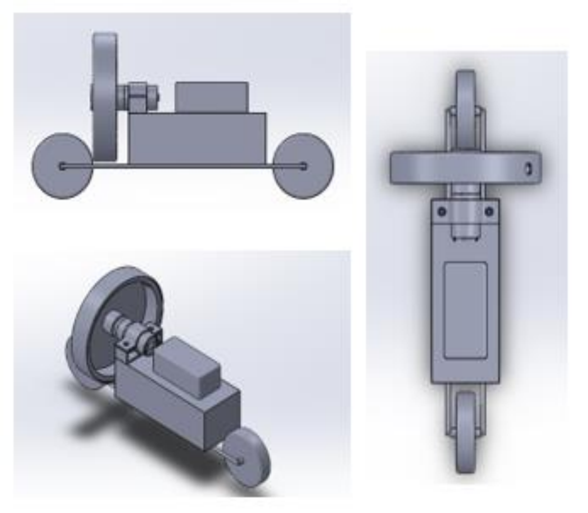
Primary Components

- **Microcontroller** (Arduino Nano)
- **IMU** (MPU6050)
- **Motor** (12V Brushed DC Motor)
- **Motor Controller** (L298N)
- **Battery** (14.8V 4 Cell LiPo)

Secondary Components

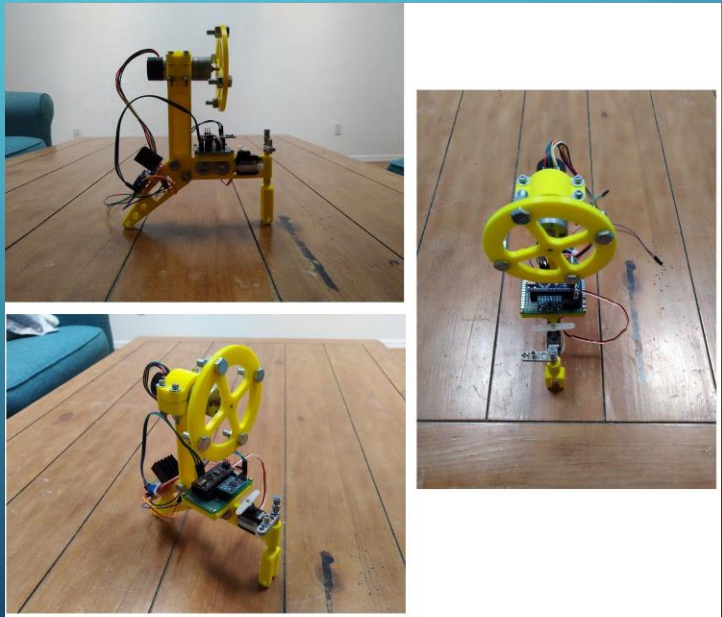
- **Servo** (5V 180°)
- **Motor Encoder** (included)

MECHANICAL HARDWARE DESIGN



1st Iteration

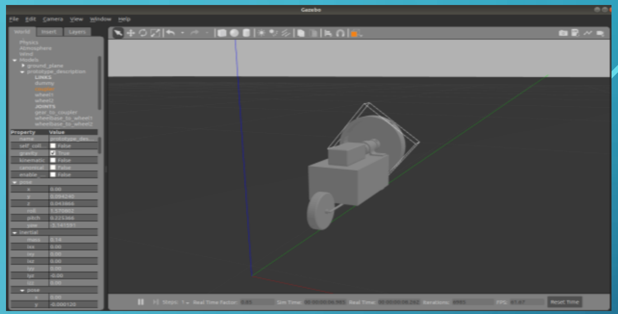
2nd Iteration

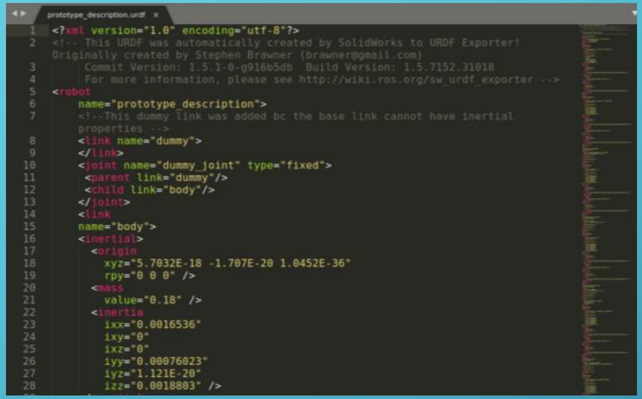


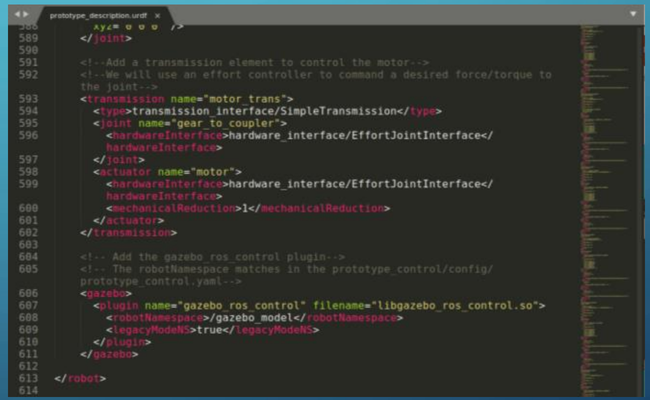
We modeled our prototypes in SolidWorks for 3D printing and importing to Gazebo. A lot of the mounting materials like nuts and bolts were scavenged from around the house.

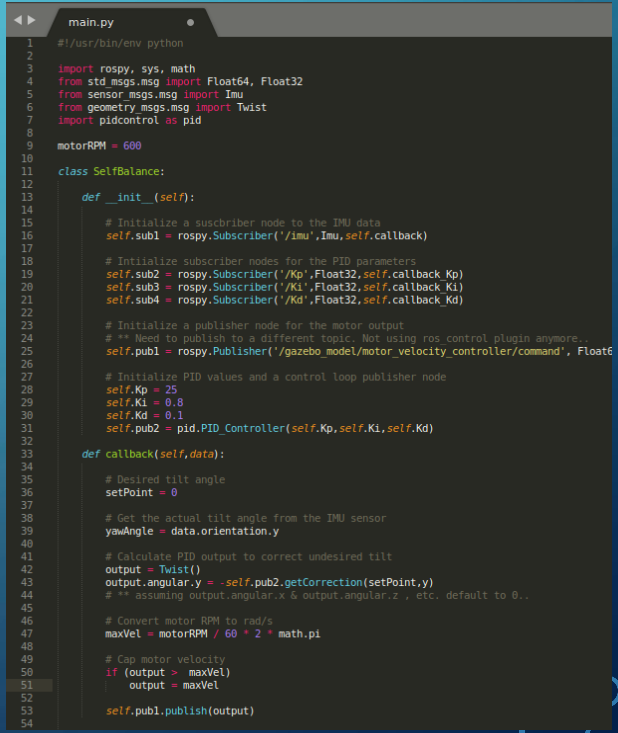
SIMULATIONS

- Simulation seemed to be the best option when trying to integrate obstacle avoidance and path planning
- Used ROS and Gazebo to implement our code and test new things on an imported 3d model of our prototype.
- The RQT GUI and a little bit of RVIZ proved useful as well









PROBLEMS & FUTURE WORK

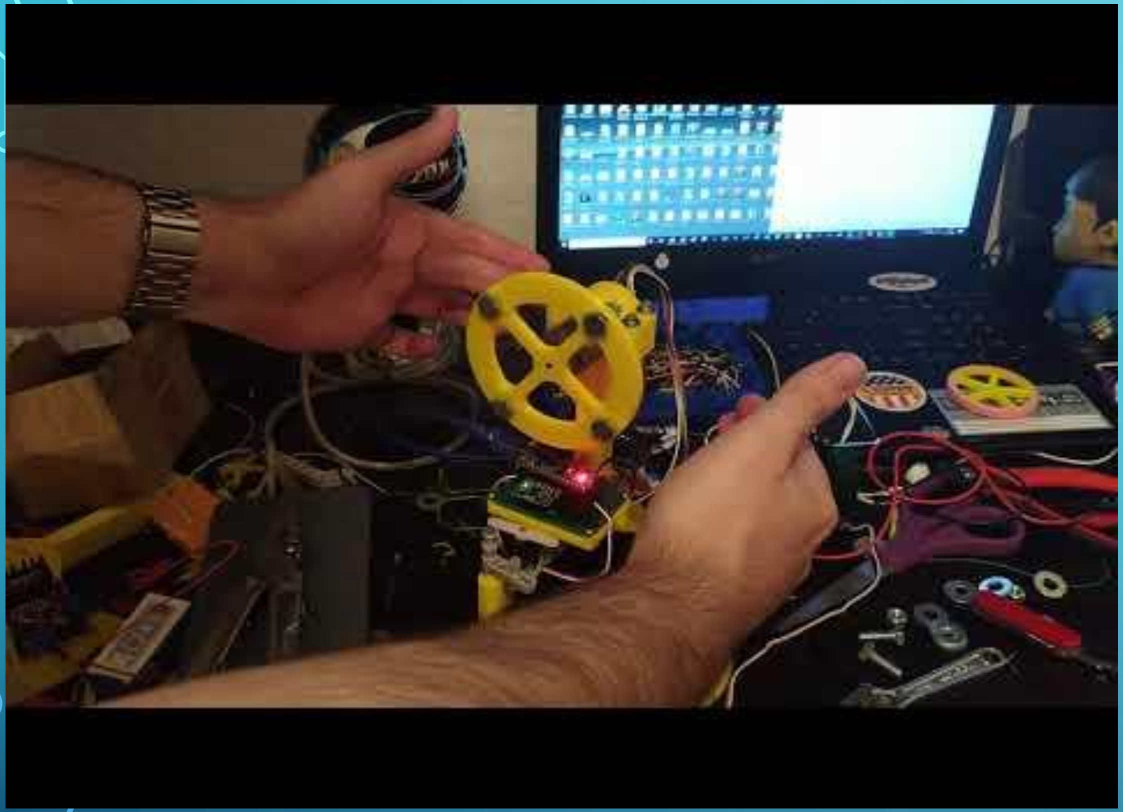
Problems

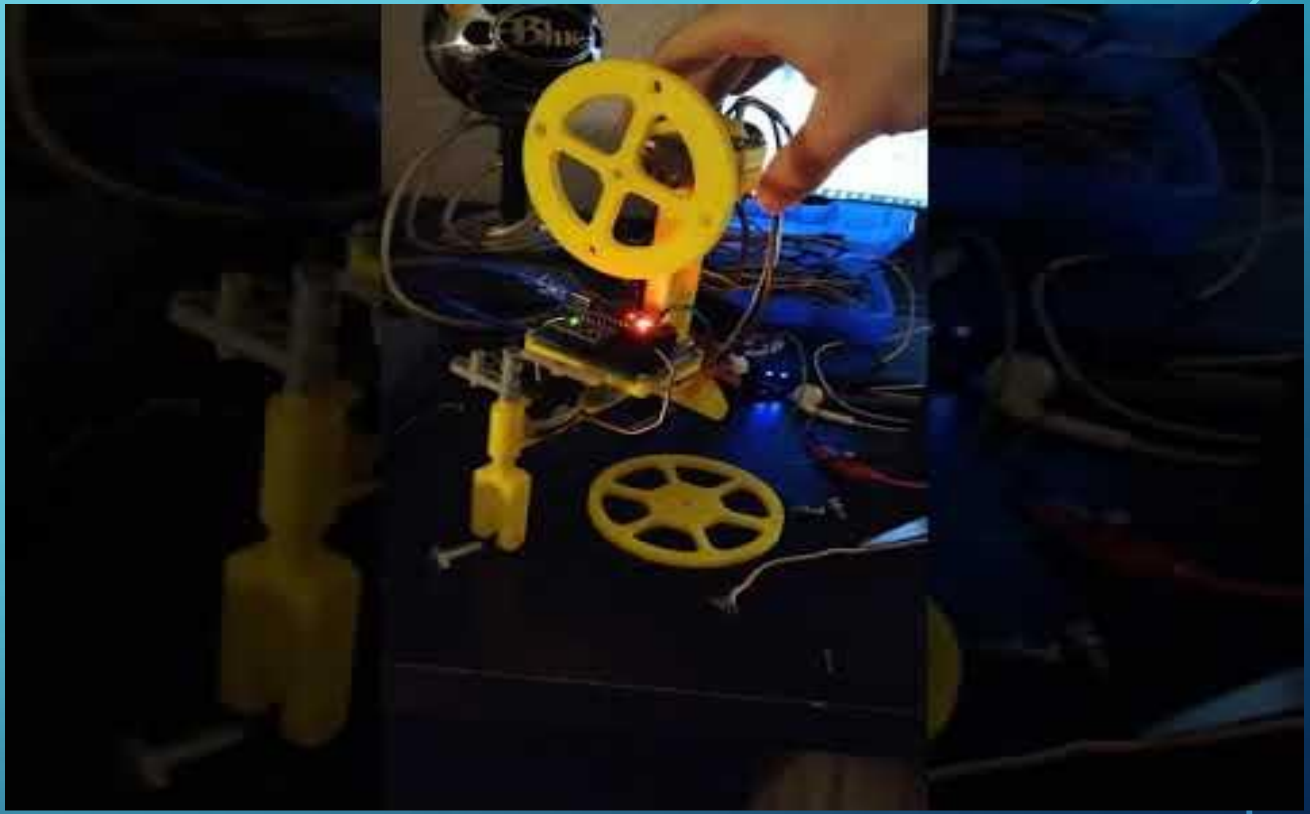
- Manual tuning of the PID parameters
- Drift in IMU measurements & inaccurate readings
- Integrating plugins for Gazebo simulation

Possible improvements

- Use encoder to improve stability
- Pole placement method aka full state feedback
- Fully implement steering capabilities
- Add obstacle avoidance and path-planning
- Add googly eyes

DEMONSTRATION





First attempt:
https://www.youtube.com/watch?v=dG3BSEhRTi8&feature=youtu.beA&feature=youtu.be

Second attempt:
https://www.youtube.com/watch?v=HYV_trc0QiA&feature=youtu.be

FIN