Special Vehicle for Transporting Unstable Chemicals

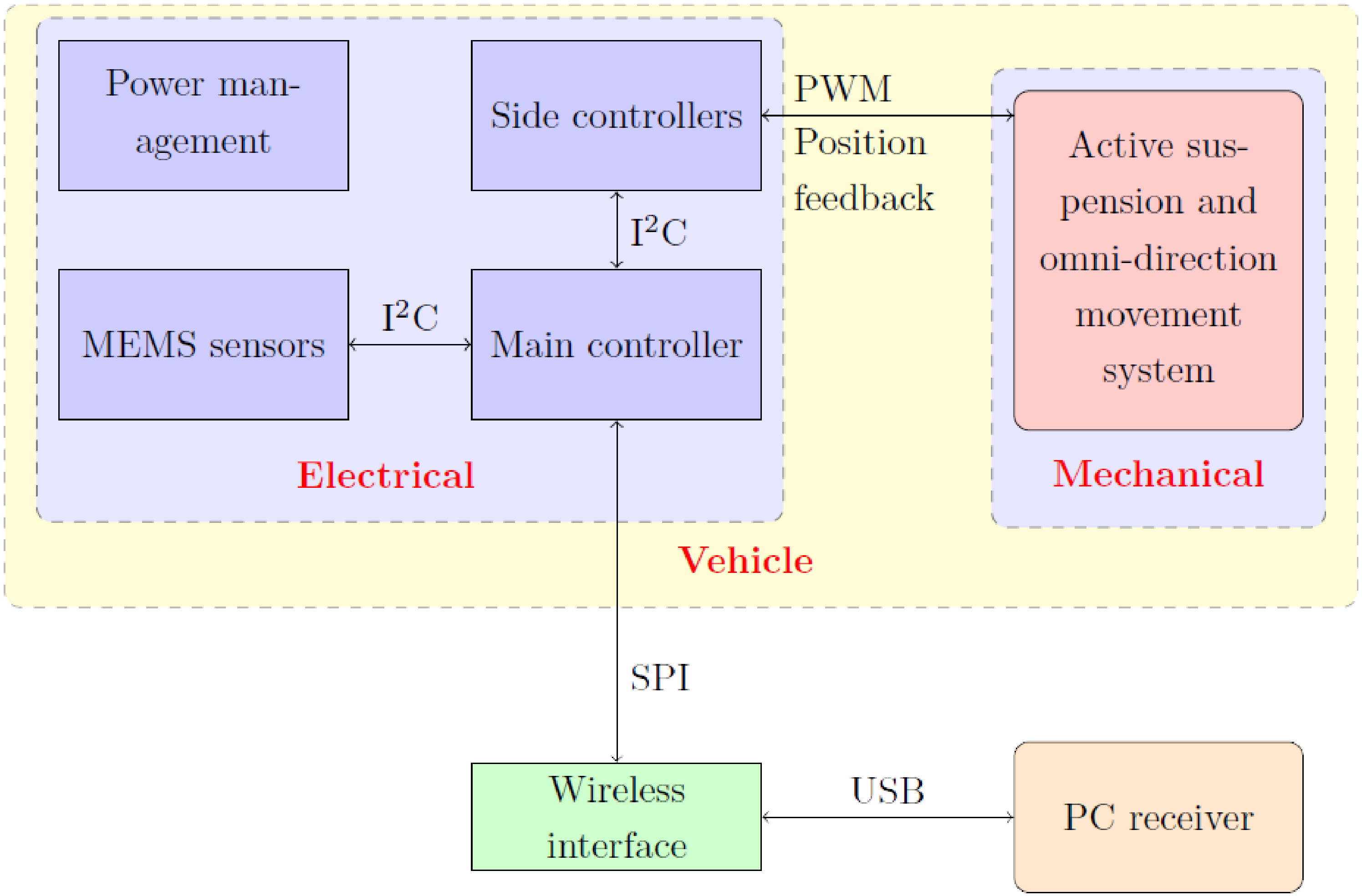
Team 15
Gong, Zhangxiaowen
Ma, Jun
Zhou, Wenjia
TA: Fortier, Justine

IEEE
Advancing Technology for Humanity

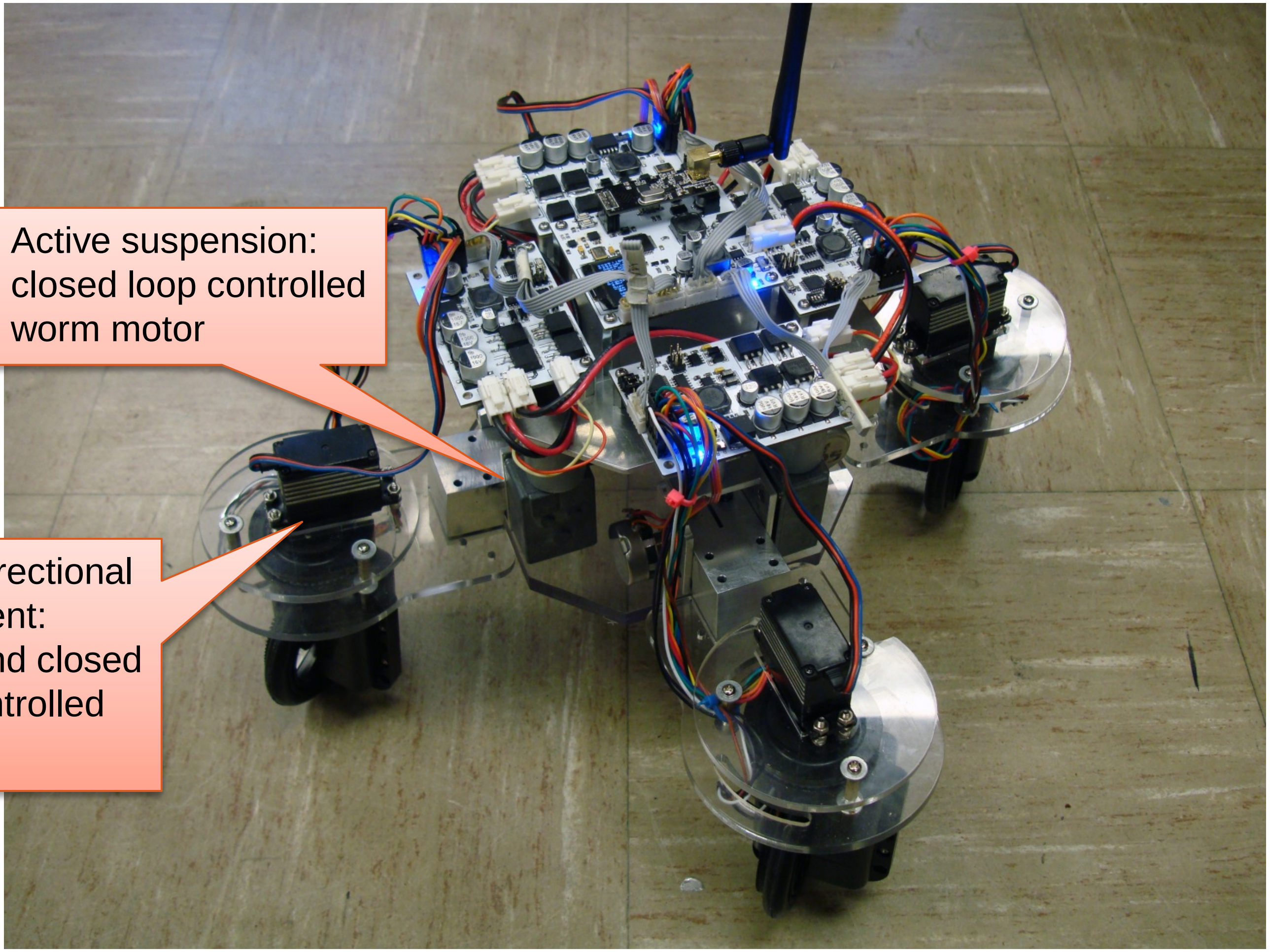
Introduction

- Motivation
  - Provide safe and high efficient transportation for dangerous or volatile chemicals
- Objective
  - Maintains a stable chassis
  - Fits both lab and road conditions
- Feature
  - Omni-directional movement
  - Active ground adaption
  - Real-time data monitoring

Modular Design



Mechanical System



Omni-directional Movement

- Design
  – Servo and closed loop controlled motor
- Why not use traditional omni-wheels?
  – Introducing vibration
  – Not suitable for real road condition

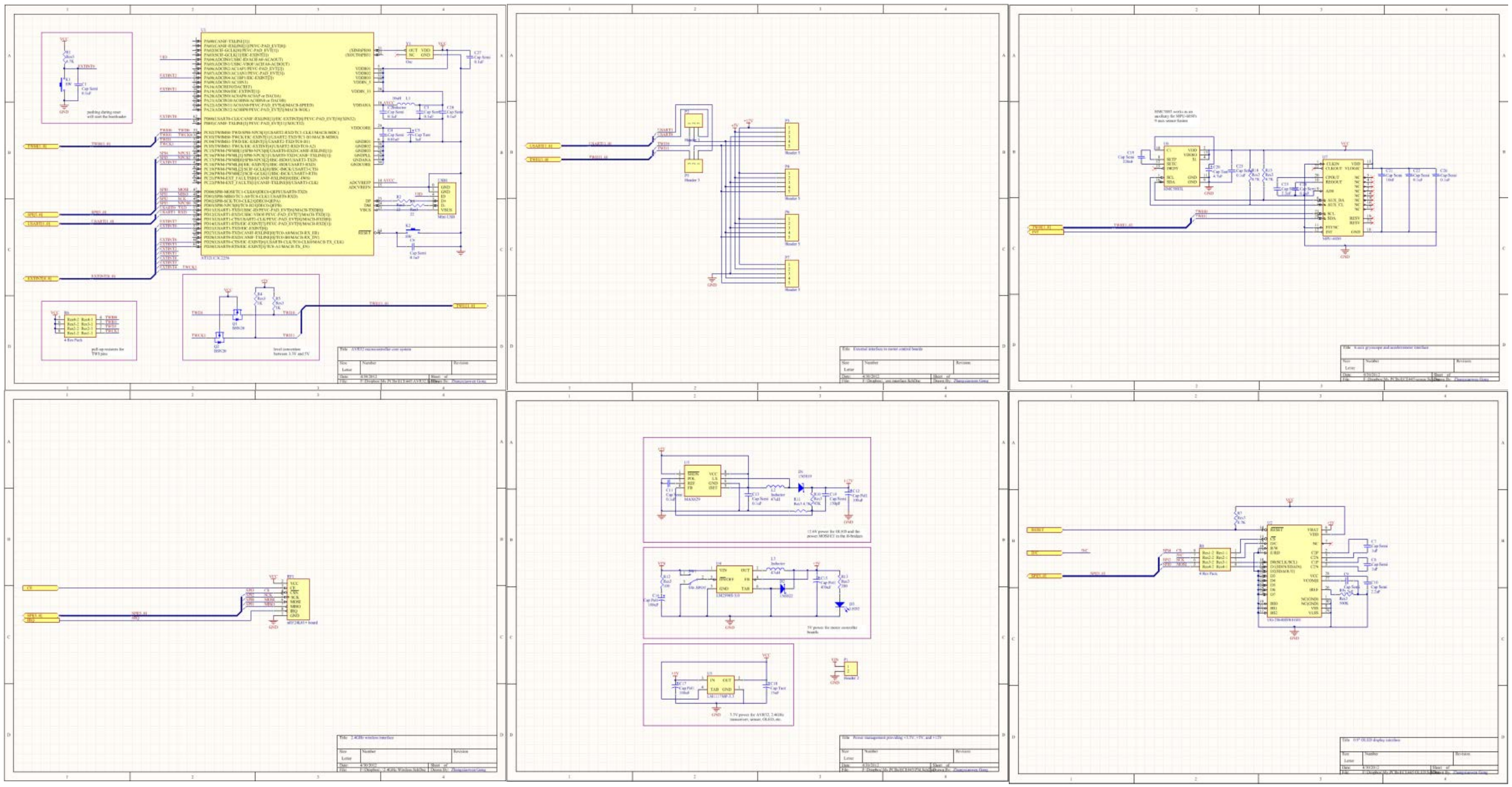
Embedded Solution

- Hardware
  - Master-slave multi-processor approach
- Software
  - High efficient driver design
  - Operating system based task management

Control System

- Simulation
  - Motor Transfer Function Testing
  - Filter for MEMS sensor
  - PID controller settling time estimation
- Calibration
  - Wireless data acquisition
  - PC software for data plotting and analysis

Main Controller



Main Controller

nRF24L01P 2.4GHz wireless point-to-point transceiver

LM2576S 5.0V 3A switching regulator

MAX629 step-up regulator (configured at 13.8V)

LM1117 3.3V 500mA linear regulator (beneath the module)

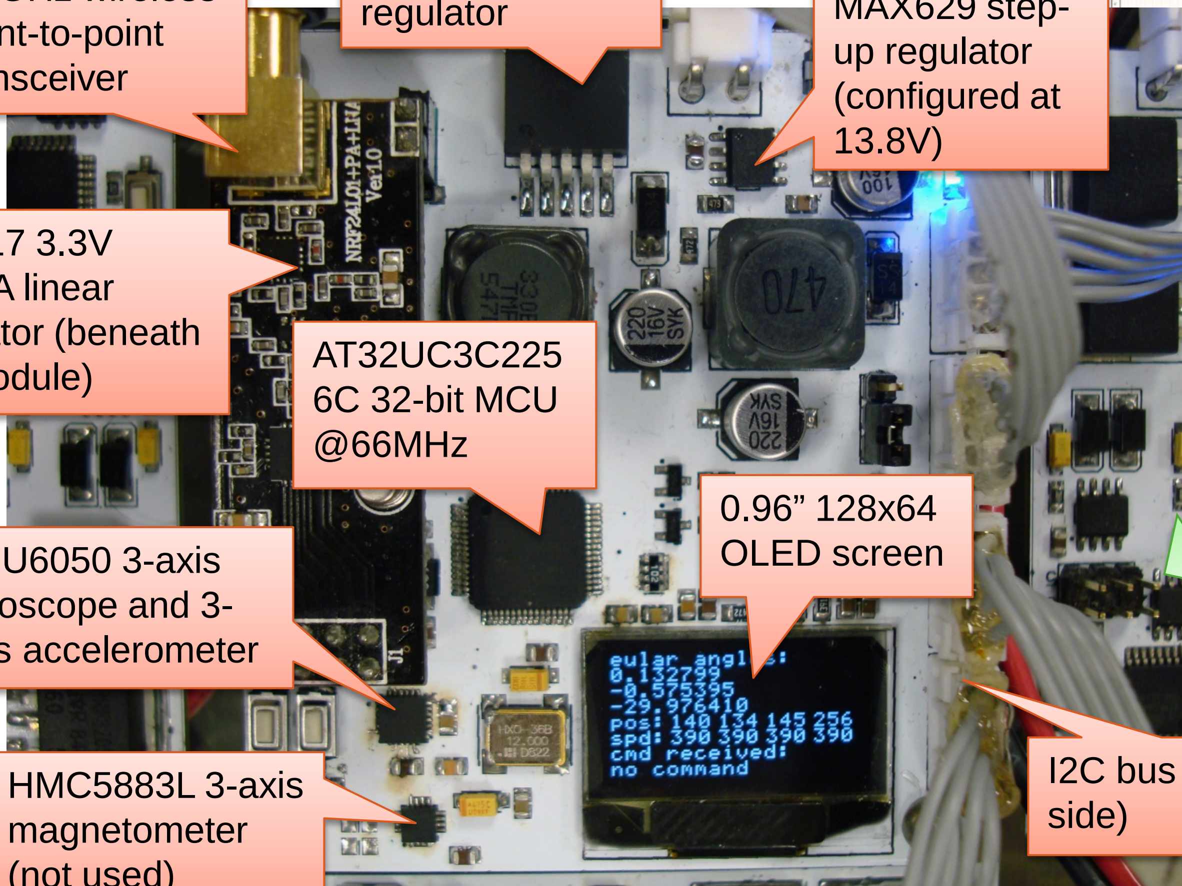
AT32UC3C2256C 32-bit MCU @66MHz

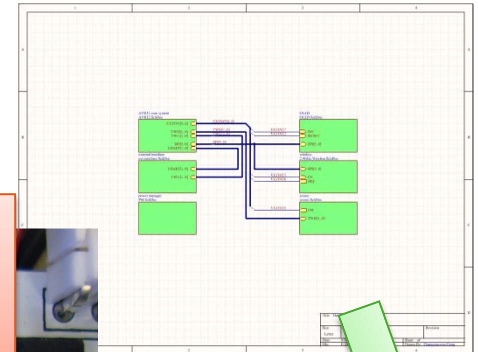
0.96” 128x64 OLED screen

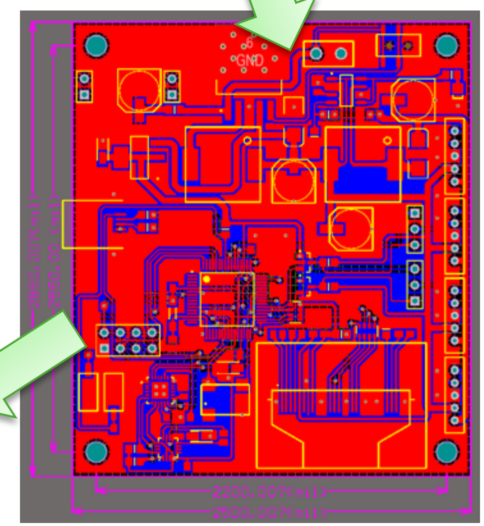
MPU6050 3-axis gyroscope and 3-axis accelerometer

HMC5883L 3-axis magnetometer (not used)

I2C bus (master side)







Main Controller

- Software highlights
  - Real-time operating system for preemptive task scheduling
  - OS optimized asynchronous I2C driver with DMA support for:
    - MEMS sensor (MPU6050)
    - Side controller
  - Asynchronous SPI driver with DMA support for:
    - OLED screen
    - Wireless transceiver
  - OS optimized dynamic memory management

Main Controller

- Real-time operating system
  - Preemptive scheduling ensures high priority tasks to be served in real-time
  - Task / priority / wake up period:
    - Active suspension / highest / 50ms
    - Update sensor reading / high / 10ms
    - Refresh OLED display / low / 100ms
    - Process received command / low / 100ms
    - Send sensor data over wireless / low / 50ms
    - Idle task that cleans up the dynamic memory / lowest / N/A

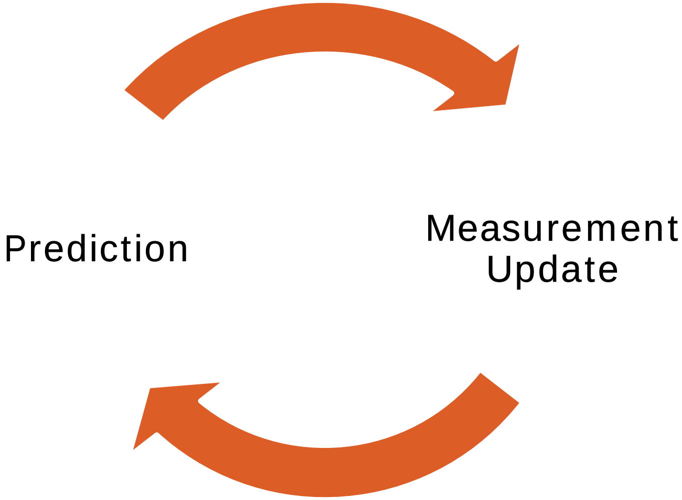
Main Controller

- I2C driver
  - OS optimized: suspend the calling task when a block read/write is required and resume it after the transaction finishes
  - DMA support that reduce the CPU load significantly
  - Full automatic streaming algorithm
- SPI driver
  - DMA support and streaming algorithm
  - Automatically switch modes for different slave devices
  - Not OS optimized because context switching may cause unnecessary overhead at high data rate

Main Controller

- Problems of MEMS sensor readings
  – Accelerometer data is noisy at high acceleration
  – Gyroscope data has drift at rest
- Choose of filter for sensor data noise reduction
  – Traditional software Kalman filter
  – MPU6050 sensor built-in complementary filter (ASIC)

Kalman Filter



gyro output

$$\dot{X} = FX + U$$

$$\dot{P} = FPF^{T}$$

$$Y = Z - HX$$

$$S = HPH^{T} + R$$

Kalman Gain $$K = PH^{T}S^{-1}$$

$$\dot{X} = X + KY$$

$$\dot{P} = (I - KH)P$$

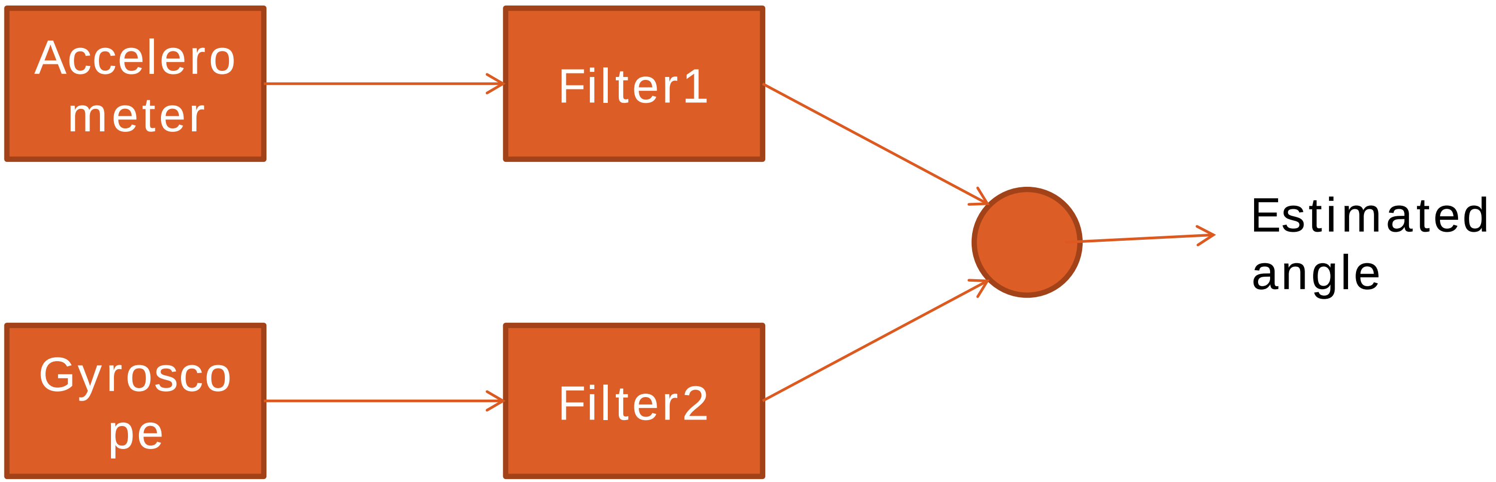
Kalman Filter

- $\theta_{k+1} = \theta_k + (\omega - \beta)dt$
- the new angle θ is equal to the previous angle plus the angular velocity ω (from Gyroscope) with drift correction multiply by time
- State-space representation

$$\begin{bmatrix} \theta_{k+1} \\ \dot{\beta}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_k \\ \beta_k \end{bmatrix} + \begin{bmatrix} dt \\ 0 \end{bmatrix} \omega$$

- X’ = F X + U
- H = $\begin{bmatrix} 1 & 0 \end{bmatrix}$

Complementary Filter

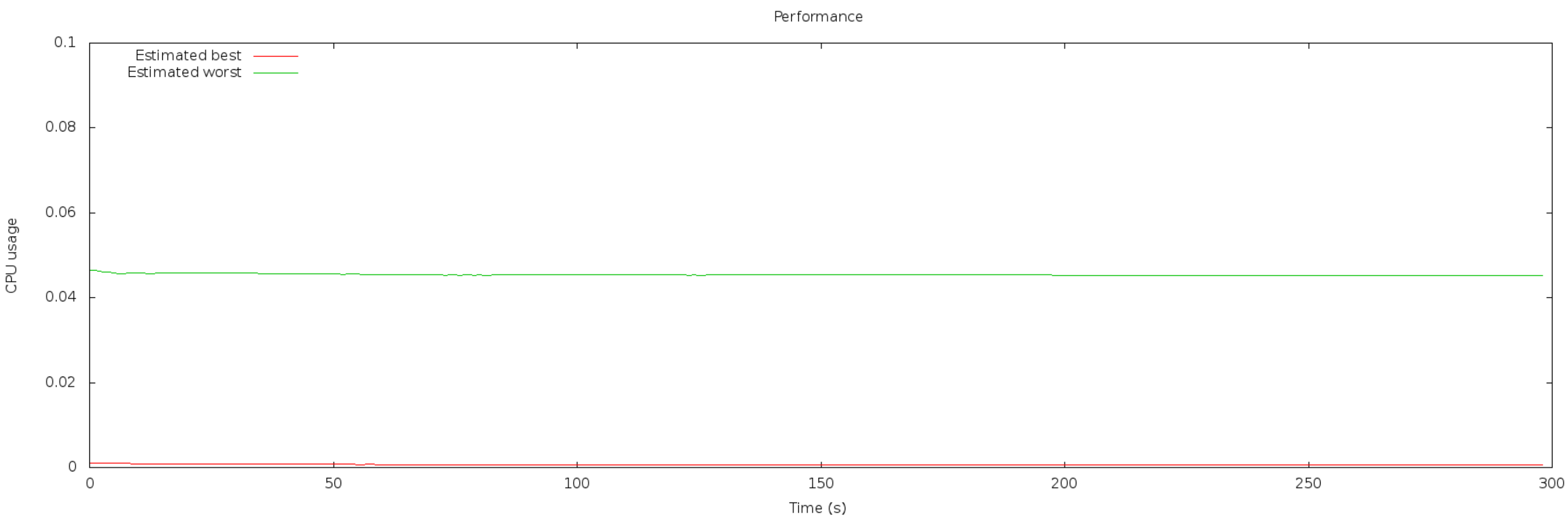


Complimentary Filter

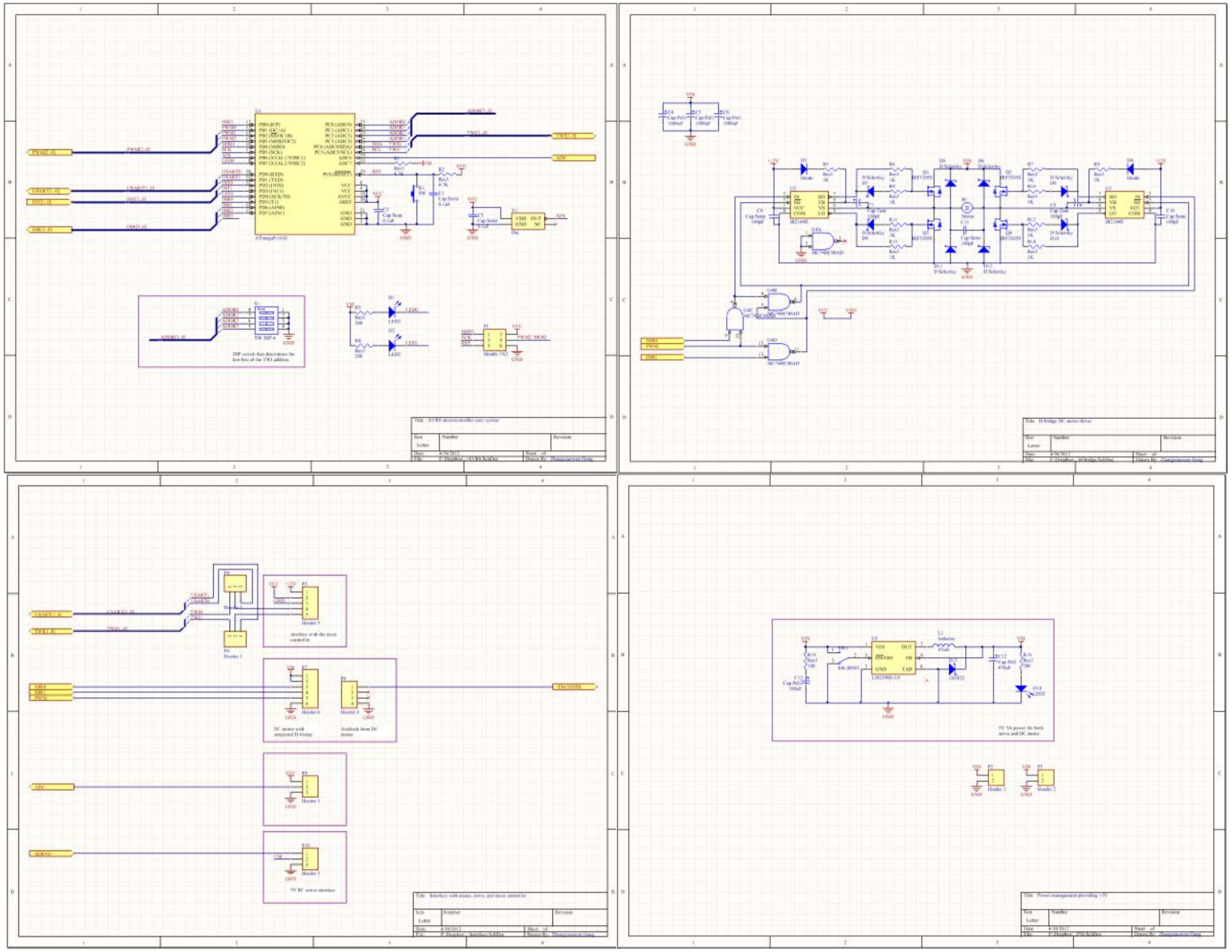
- Ha(s)Ga(s)+sHg(s)Gg(s)=1
- Assume ideal sensors
- Ha(s)=Hg(s)=1
- Ga(s)= $\frac{2\tau s + 1}{(\tau s + 1)^2}$
- Gg(s)= $\frac{\tau^2 s}{(\tau s + 1)^2}$
- Tune the constant tau

Main Controller

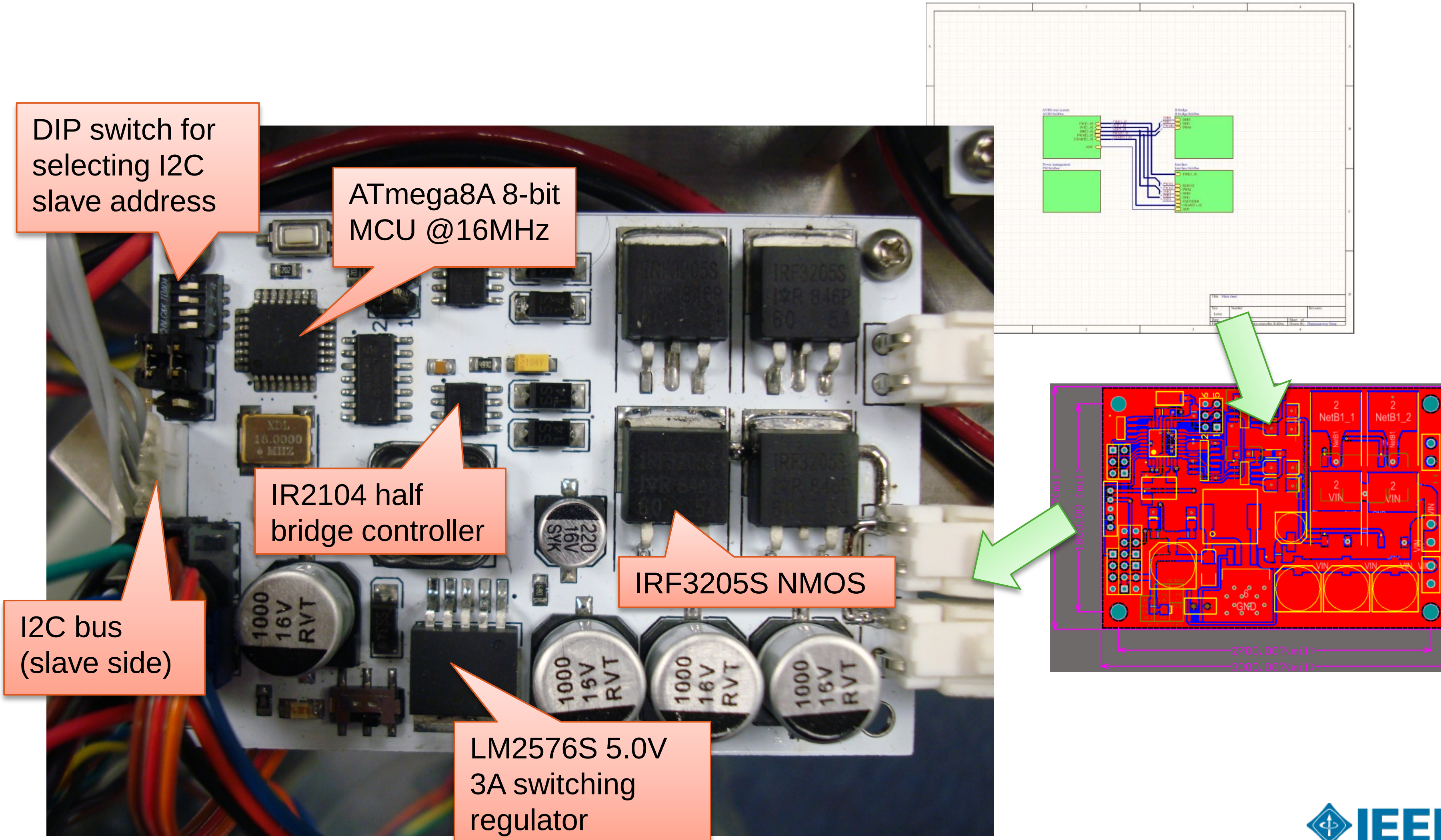
- CPU usage analysis
  - Worst case estimation: consider idle only when the idle task occupies a whole OS tick (1ms)
  - Best case estimation: consider idle whenever the idle task is executed in an OS tick



Side Controller



Side Controller



IEEE Advancing Technology for Humanity

Side Controller

- Software highlights
  - Foreground-background structure
  - Asynchronous I2C slave driver
  - Wheel motor driver with PID speed control
  - Worm motor driver with position monitor
  - Software generated PPM signal for servo control

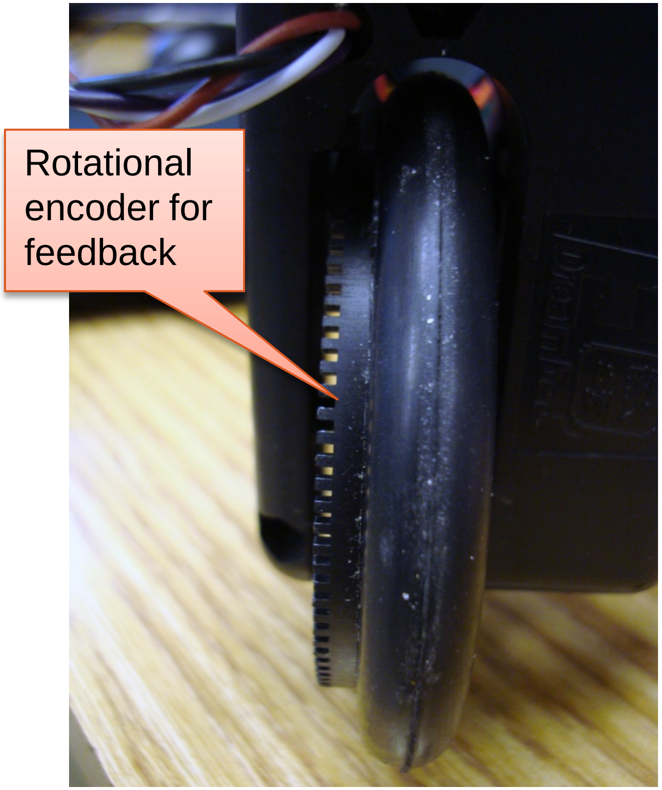
Side Controller

- Asynchronous I2C slave driver
  - Interrupt based auto package handling
  - Fully compatible with traditional I2C slave register convention
  - Commands:
    - Set wheel motor PWM output
    - Set worm motor PWM output
    - Set servo position
    - Set servo calibration setting
    - Set worm motor position threshold
    - Worm motor break
    - Wheel motor break
    - Set wheel motor PID set point
    - Get wheel motor speed
    - Get worm motor position

IEEE
Advancing Technology for Humanity

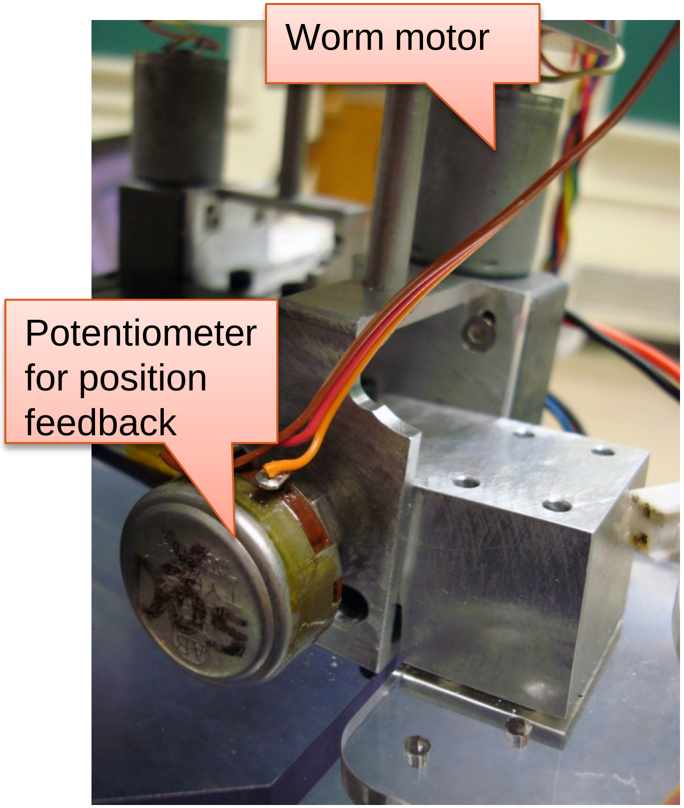
Side Controller

- Wheel motor driver with PID speed control
  - External interrupt for capturing encoder events
  - Timer interrupt for measuring wheel speed
  - Speed sampling period: 50ms
  - PID settling time: 300ms



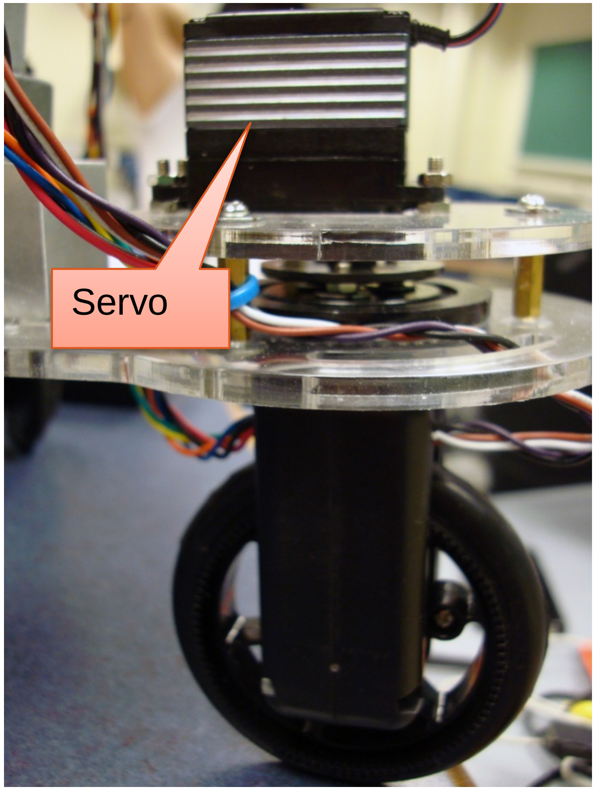
Side Controller

- Worm motor driver with position monitor
  - ADC works at continuous sampling mode to monitor the worm motor's position
  - ADC interrupt checks whether the worm motor is at threshold position and disables its further movement accordingly



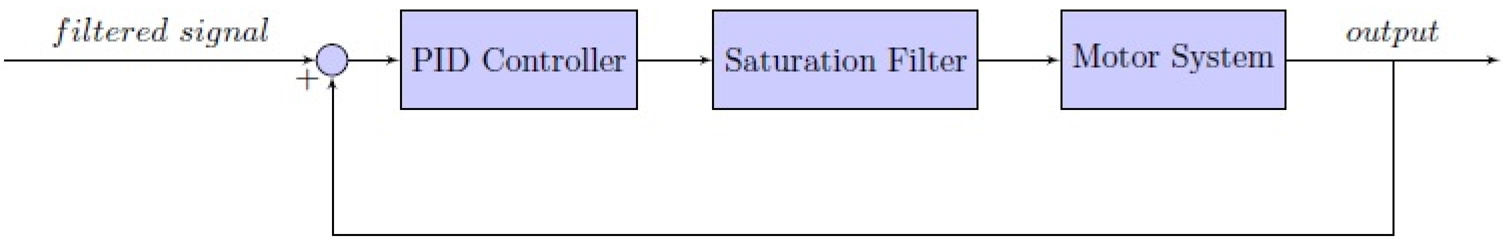
Side Controller

- Software generated PPM signal for servo control
  - Accumulated timer interrupts for generating 20ms period PPM signals
  - Position calibration setting can be received from the main controller over I2C bus

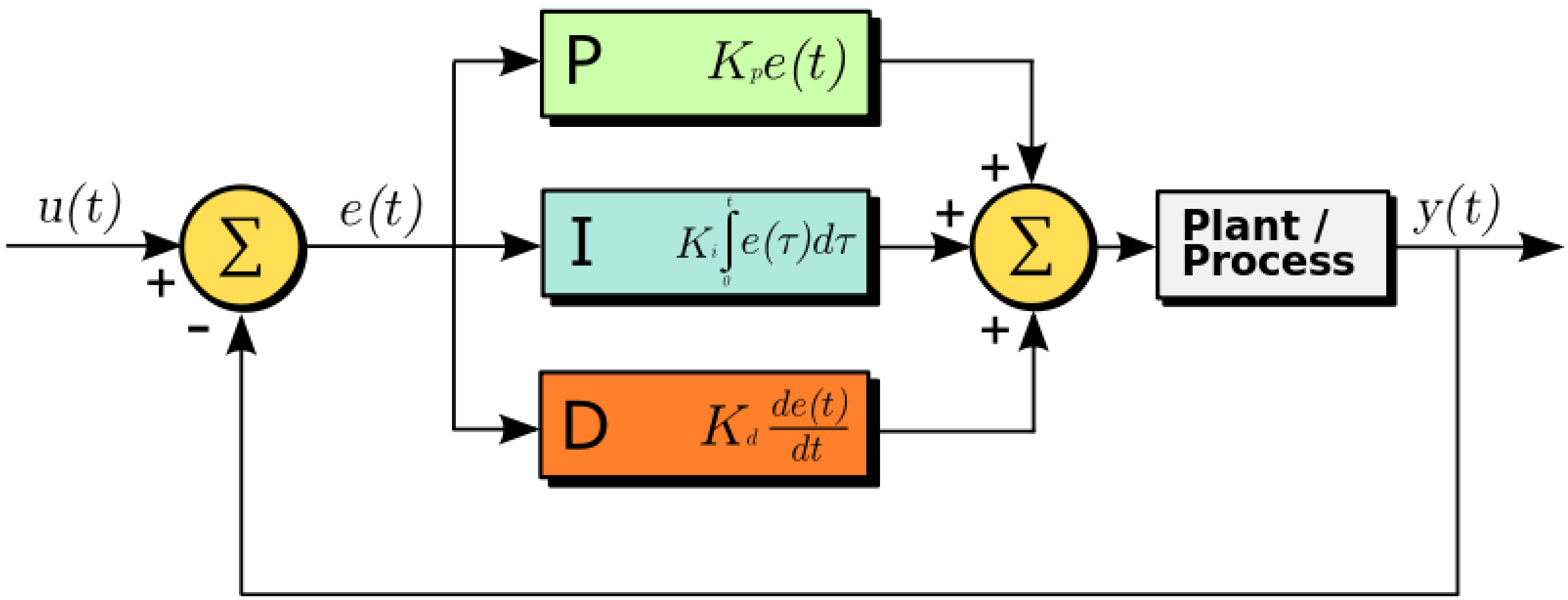


Feedback Control System

- PID Controller
- Saturation Filter
- Worm Motor System



PID Controller



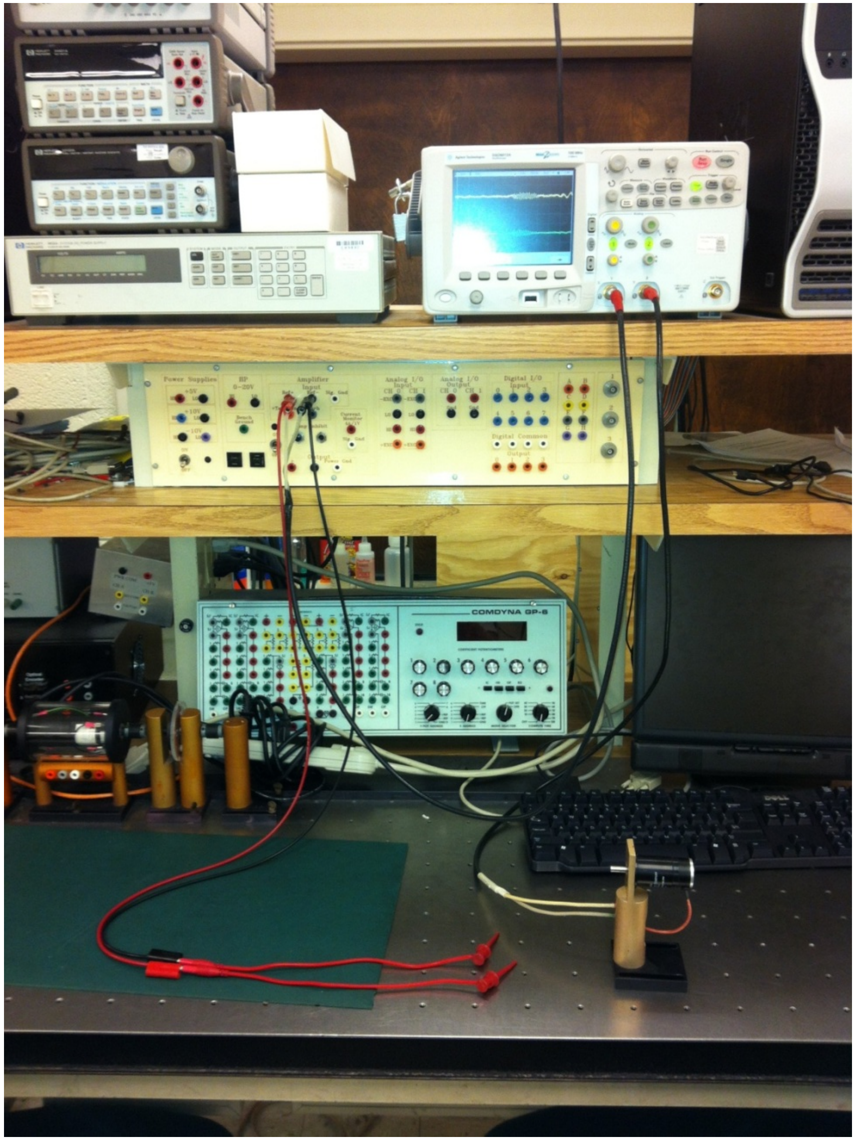
IEEE Advancing Technology for Humanity

Saturation Filter

- A filter that will cut off the signal if it detects the signal has gone beyond the limit

Worm Motor System

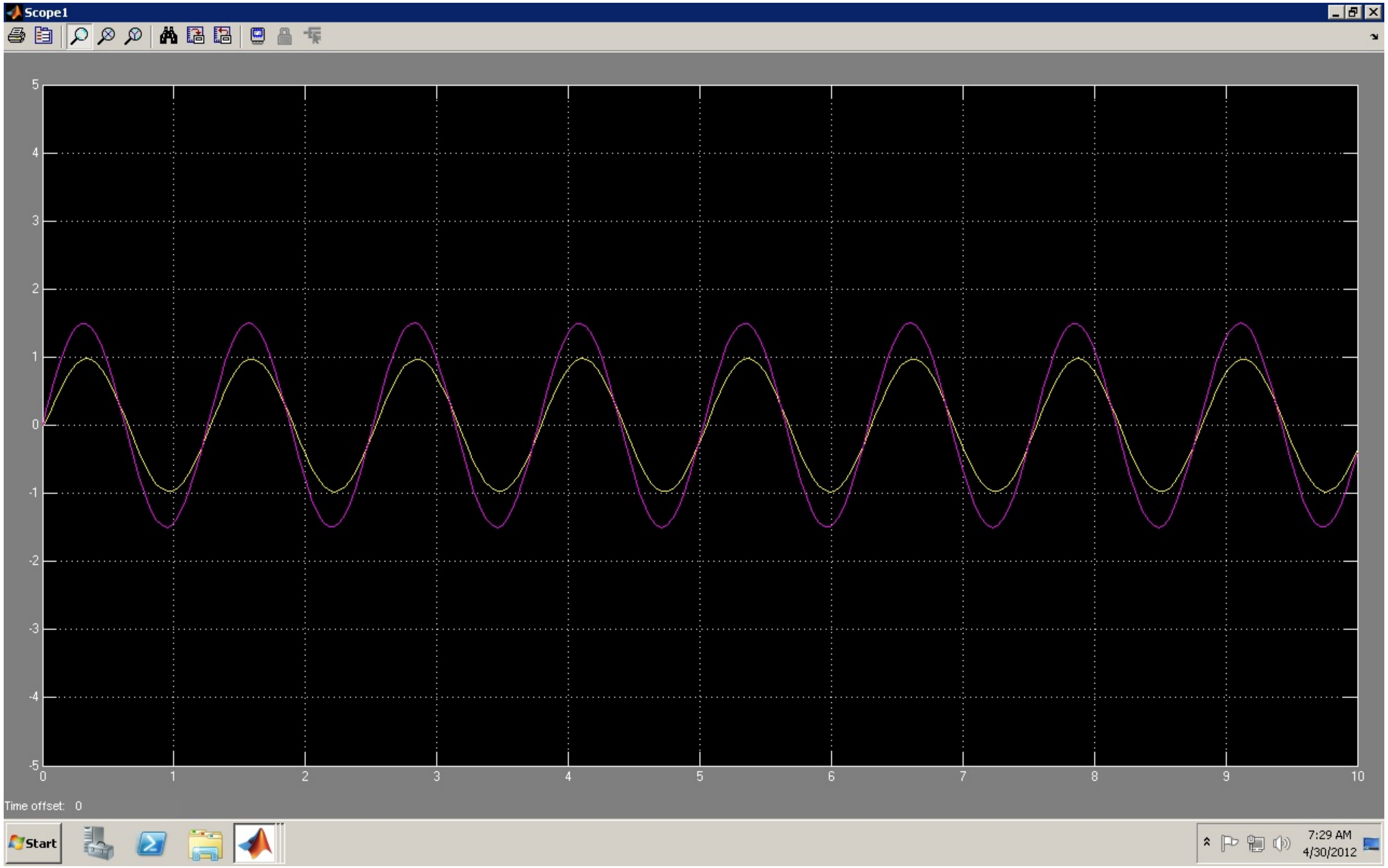
- Determine the transfer function of the motor system
- Frequency Response
- Step Response



Motor Transfer Function

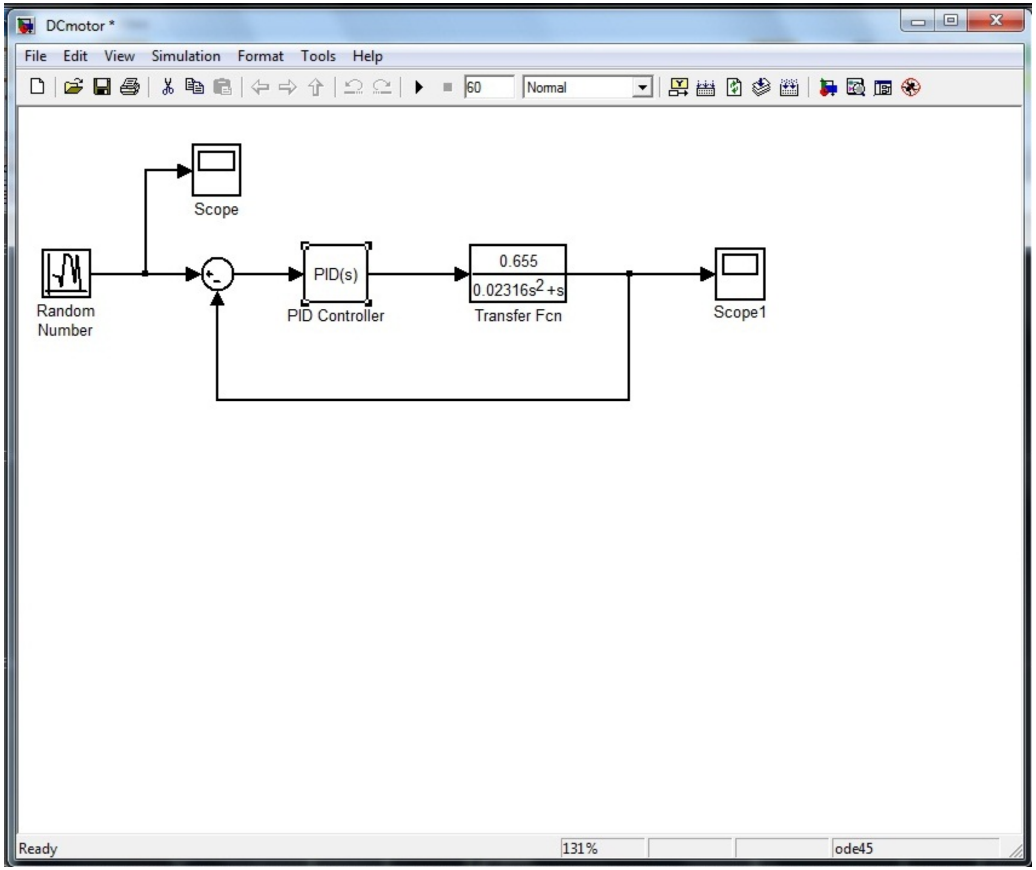
$$G_m = \frac{K_m}{s\tau_m + 1}$$

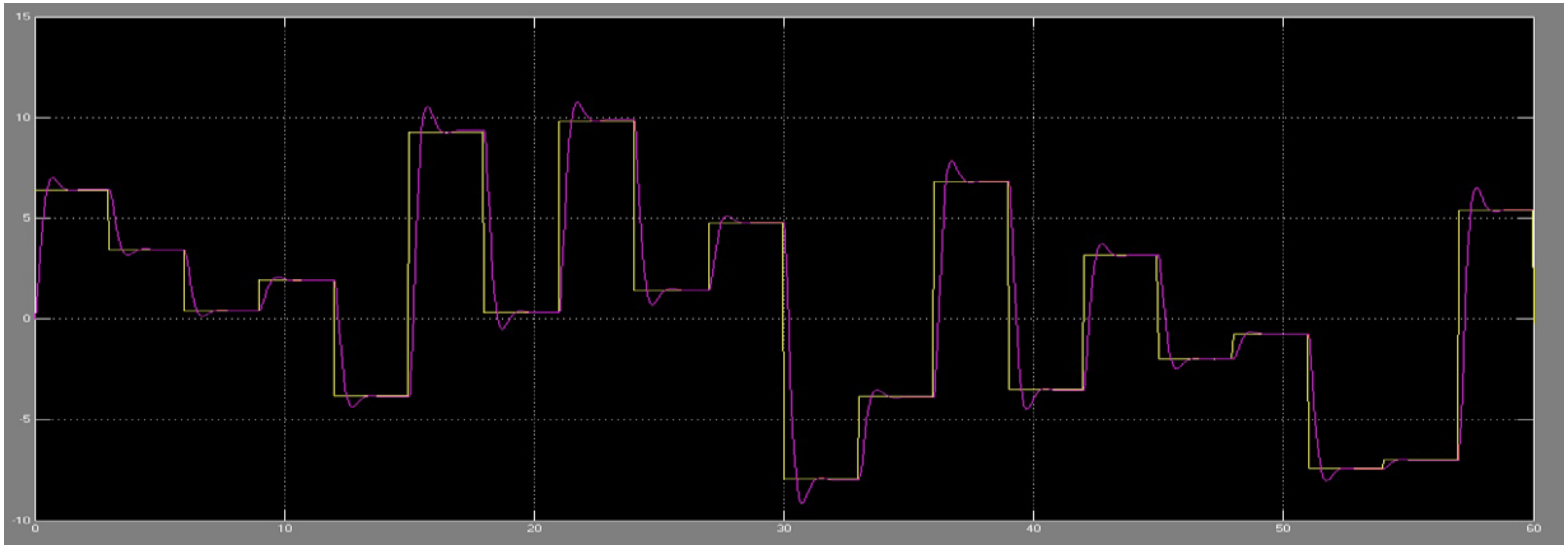
Frequency Response

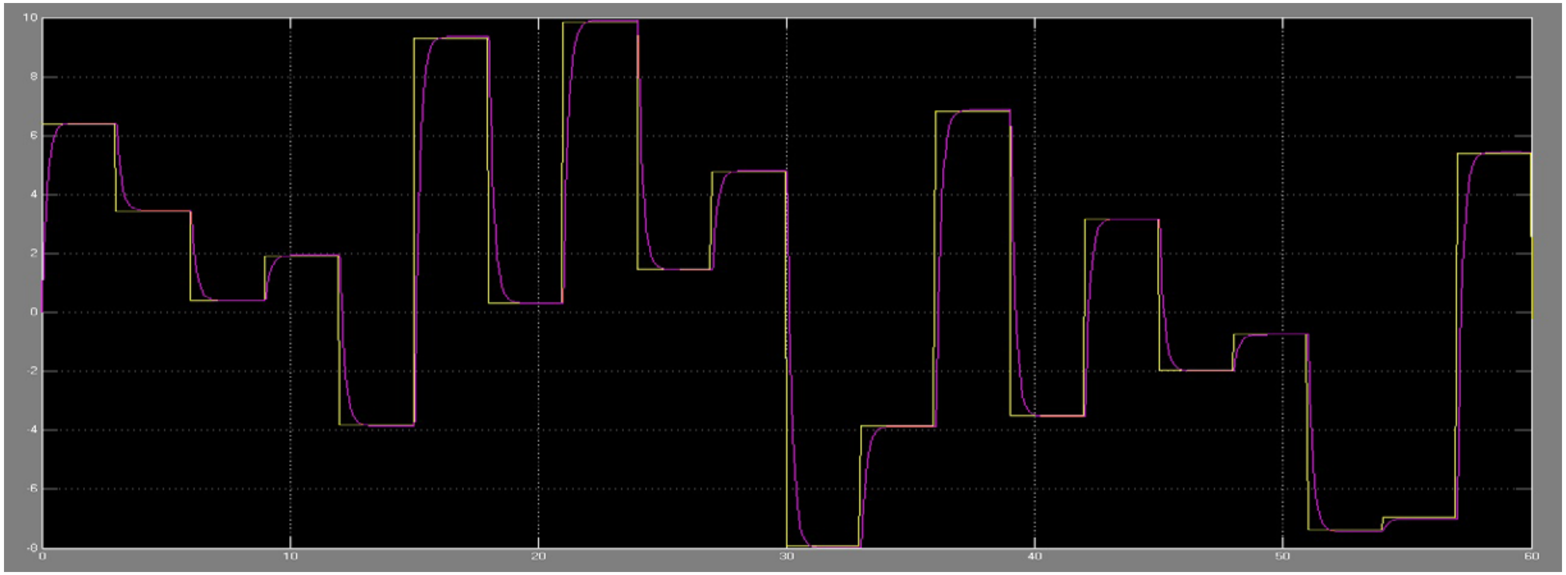


Step Response

Overall Suspension System Simulation




15
10
5
0
-5
-10
0
10
20
30
40
50
60

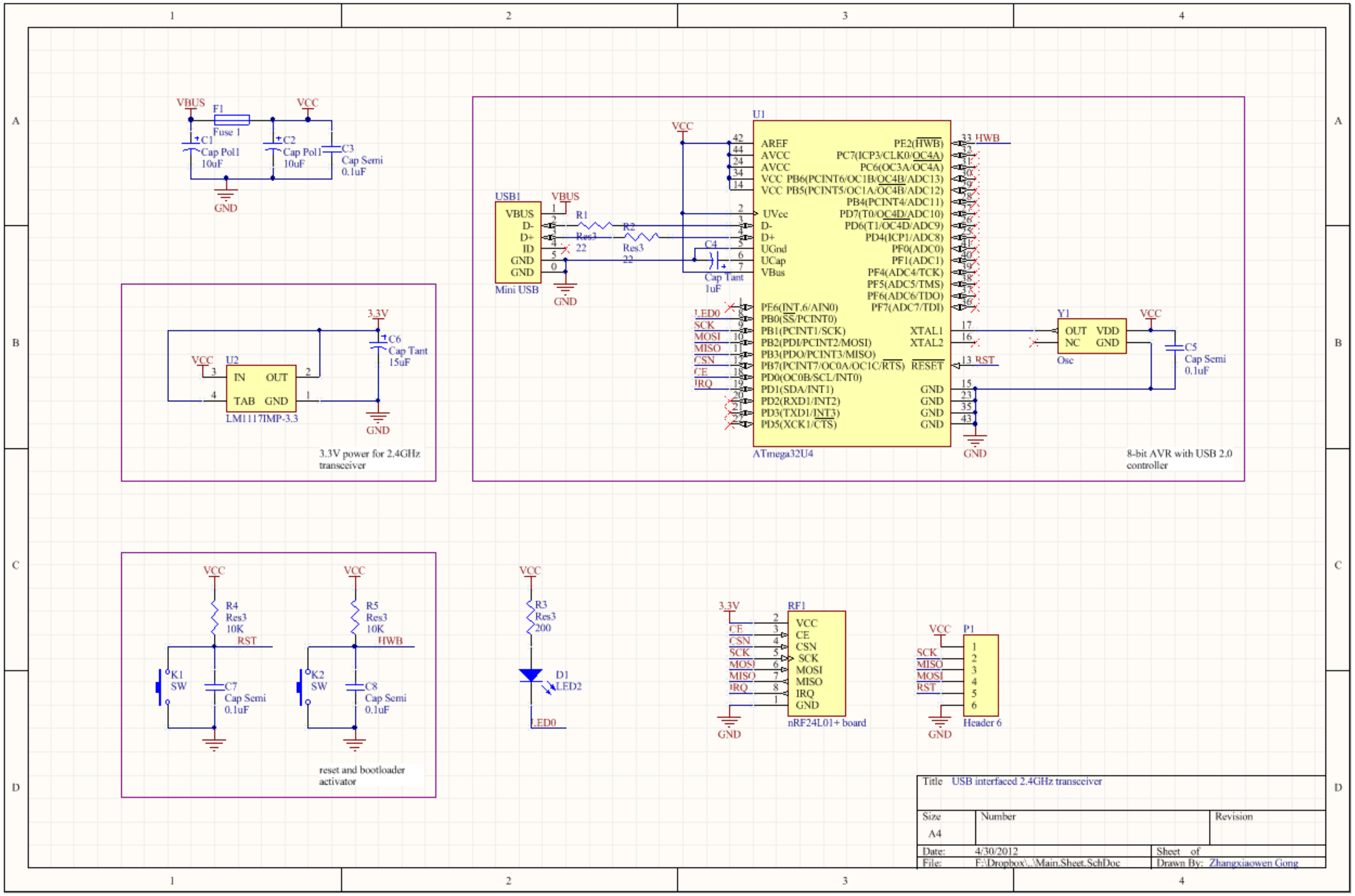

10
8
6
4
2
0
-2
-4
-6
-8
0
10
20
30
40
50
60

Pseudo Code for Control System

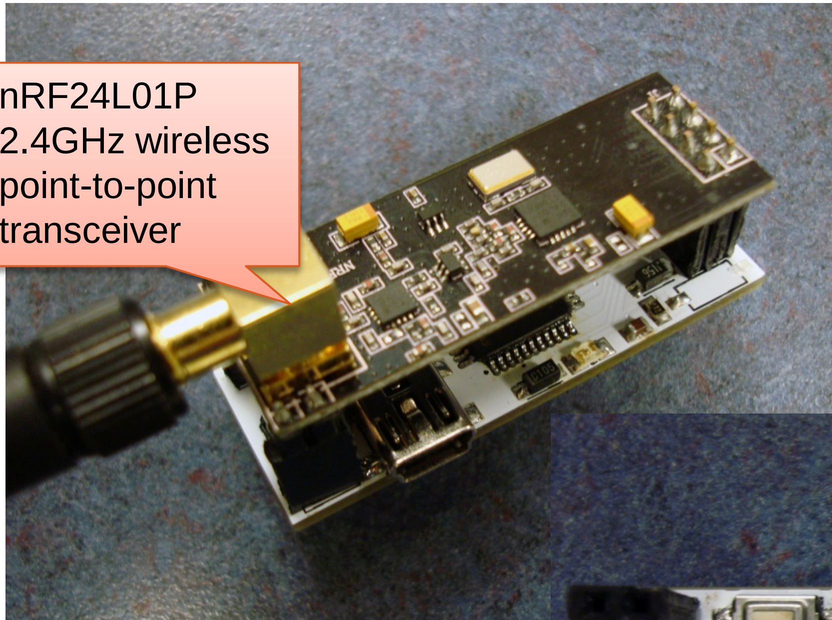
```
integral = 0
preverror = 0

error= sp – pv
output = Kp*error + Ki*(integral + error*dt) + Kd*(error -
   preverror)/ dt

preverror = error

if (output > max)
{ output = max; }
else if (output < min)
{ output = min; }
```

IEEE
Advancing Technology for Humanity

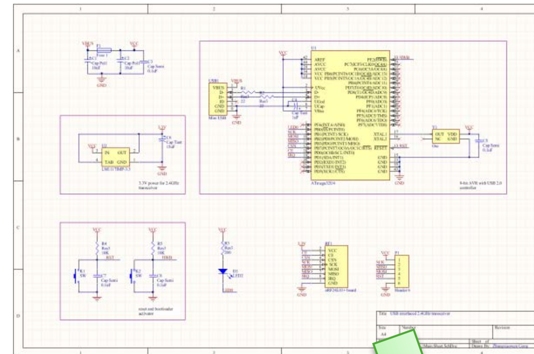
Wireless Receiver

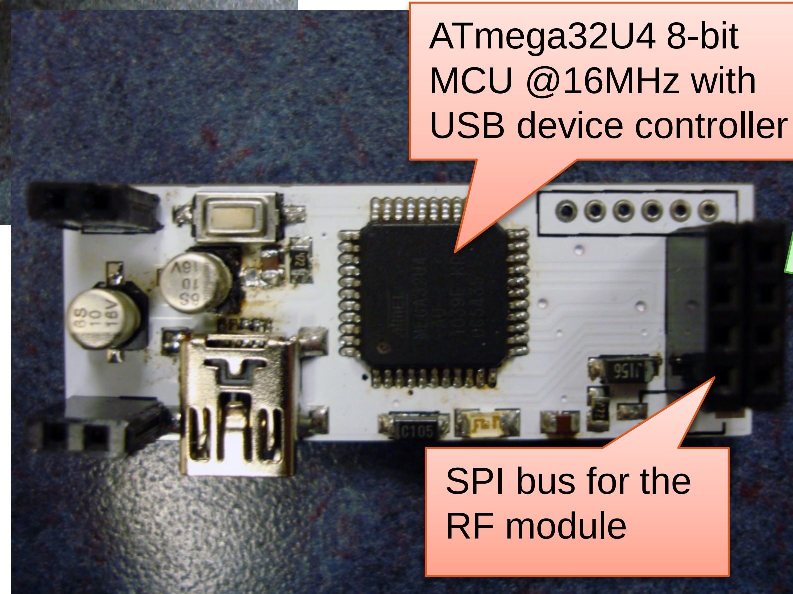


Wireless Receiver



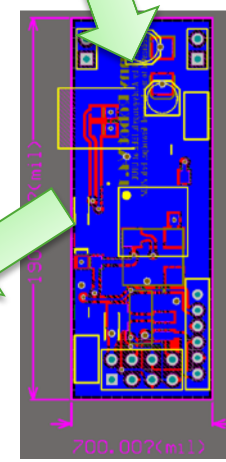
nRF24L01P 2.4GHz wireless point-to-point transceiver





ATmega32U4 8-bit MCU @16MHz with USB device controller

SPI bus for the RF module

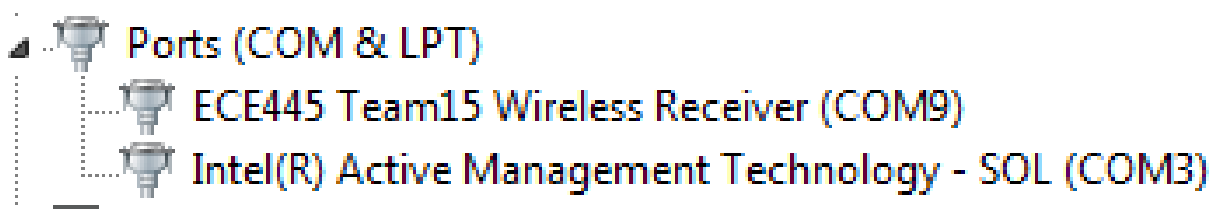


Wireless Receiver

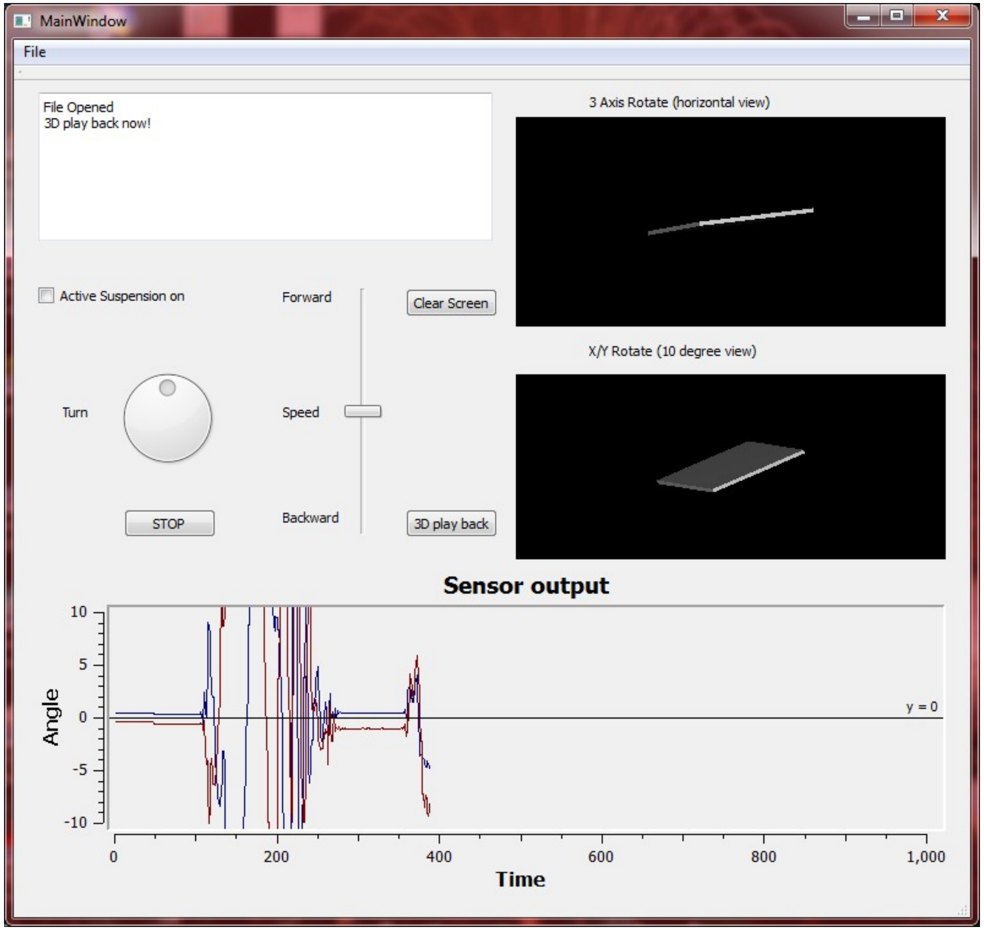
- Software highlights
  - Asynchronous USB device driver
  - CDC (Communication Device Class) driver
  - Support for connecting up to 6 other transceivers

Wireless Receiver

- CDC driver
  - Emulate a virtual serial port
  - Data rate is not limited by the serial port BAUD rate setting
  - Reduce the complexity of the PC side USB driver



PC Software



- Windows Serial Port driver
- Data Storage
- 3D View
- Real time Data Plotting
- Motion replay

Communication Protocol

Incoming data

- Find “0x00 0x00 0x00 0x00” header
- Start reading the next 12 bytes

Binary to Float

- Read 4 bytes a time ,convert string to hex
- IEEE 754 Binary to Float conversion

Axis Adjust

- Map Y axis in OpenGL to Z axis
- Map Z axis in OpenGL to –Y axis

IEEE Advancing Technology for Humanity

Communication Protocol

Get Command

- Read status from Dial and Slider bar
- Get the corresponding command code

Assemble the Command

- Check if active suspension is turned on
- Format : (0xFF) * 4 + Command + Value

Send Command

- Check if serial port is open
- Write to serial port

Data Storage

- Serial port receiver buffer size increasing with time and Qt Text browser insertion lead to accumulated delay
- Need another way of saving data
- Dynamically allocate memory
- Use linked list to avoid copy data
- Create a new file format

Sample File Format (demo.ECE445)

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000	00	12	00	24	a8	ce	f6	3e	25	5d	09	c2	dd	c4	b7	3e
00000010	a8	ce	f6	3e	25	5d	09	c2	dd	c4	b7	3e	1f	3e	02	3f
00000020	66	96	09	c2	6a	5c	bf	3e	8a	fd	f3	3e	e2	d2	09	c2

00004810	62	f7	81	40	c7	18	61	c2	70	b9	dd	c0	62	f7	81	40
00004820	c7	18	61	c2	70	b9	dd	c0								

0x12=18 0x24=36
16*1024+36 = 18468 bytes

0x4824 = 18468 bytes

File is stored correctly

Data Demonstration

- 3D view
  - OpenGL Basics
  - initializeGL()
  - paintGL()
  - UpdateGL()
- Data line charts
  - Qwt open source library
  - A Qt widget for scientific data plotting
  - Reload stored data

Challenges and Improvements

- Stronger motor with greater torque should be adopt for the active suspension
  - May use linear actuators instead of worm motors
- Less powerful MCU could be used in order to reduce both cost and power consumption
  - Low end ARM Cortex-M3 MCU (around $3)

Special Thanks

- Prof. Carney
- TA Justine
- Machine Shop
- Service Shop
- Dan Block
- Everyone who supported us during the project