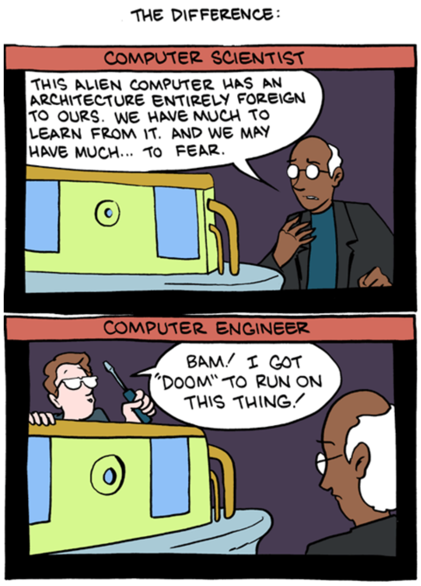
EE445M/EE380L.12 Embedded and Real-Time Systems/ Real-Time Operating Systems

Lecture 9:

Sensing & Acting,
Input Capture, PWM, Motors

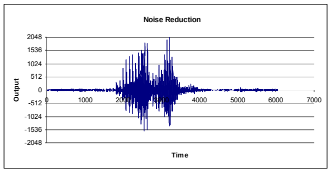
EE445M vs. EE461S/CS372


THE DIFFERENCE:
COMPUTER SCIENTIST
THIS ALIEN COMPUTER HAS AN ARCHITECTURE ENTIRELY FOREIGN TO OURS. WE HAVE MUCH TO LEARN FROM IT. AND WE MAY HAVE MUCH... TO FEAR.
COMPUTER ENGINEER
BAM! I GOT "DOOM" TO RUN ON THIS THING!

http://www.smbc-comics.com/index.php?db=comics&id=2158#comic

Class Agenda

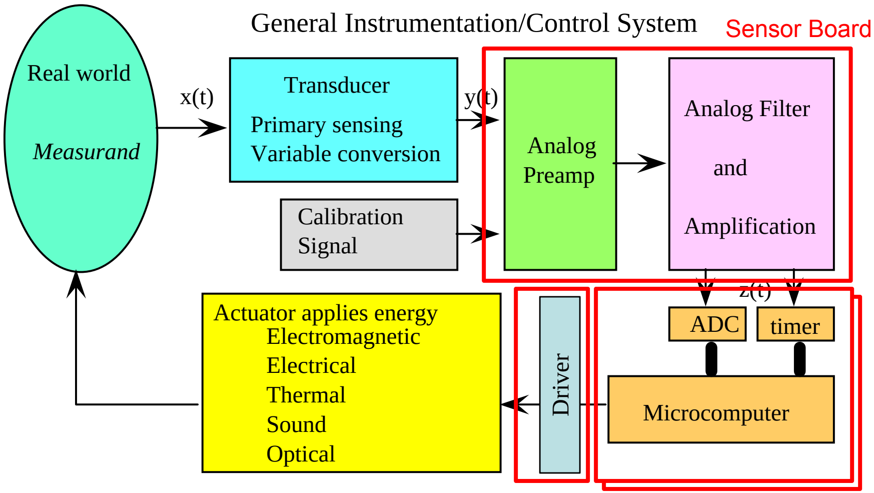
- Recap: RTOS Kernel
  - Multi-tasking, context switch, scheduling
  - Synchronization, communication, semaphores
  - File system, memory management
- Outlook: Applications of RTOS
  - Lab 6: Robot interfaces
    - Sensors, Motors
    - Networking
  - Lab 7: Robot control



Reference book, Chapter 10

Instrumentation & Control

General Instrumentation/Control System

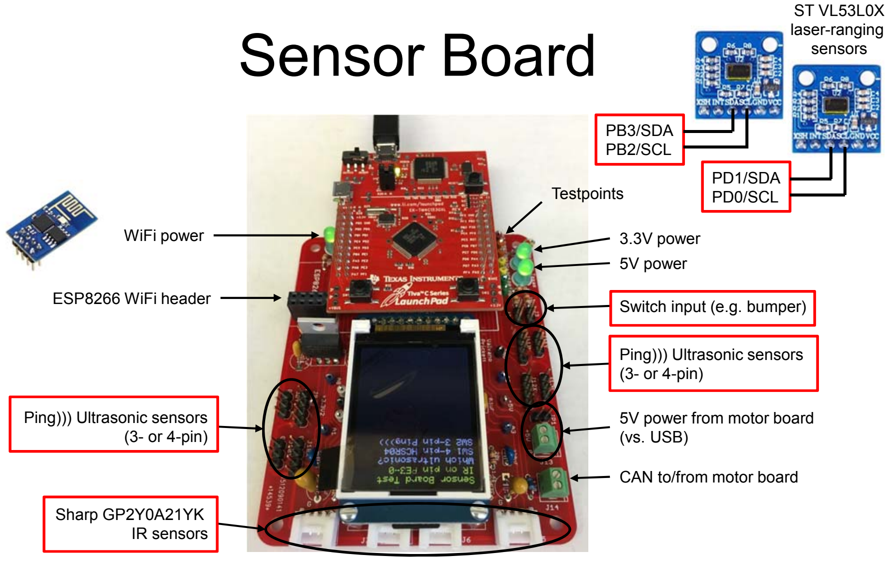


Sensor Board

Motor Board

2x LaunchPad connected via CAN

Sensor Board



ST VL53L0X laser-ranging sensors

PB3/SDA
PB2/SCL

PD1/SDA
PD0/SCL

Testpoints

WiFi power

ESP8266 WiFi header

3.3V power

5V power

Switch input (e.g. bumper)

Ping))) Ultrasonic sensors (3- or 4-pin)

Ping))) Ultrasonic sensors (3- or 4-pin)

5V power from motor board (vs. USB)

CAN to/from motor board

Sharp GP2Y0A21YK IR sensors

- **Reference material**
  - **Schematic:** http://www.ece.utexas.edu/~gerstl/ee445m_s16/resources/Robot_Sensor_v3.pdf
  - **PCB layout:** http://www.ece.utexas.edu/~gerstl/ee445m_s16/resources/sensor_top3.png

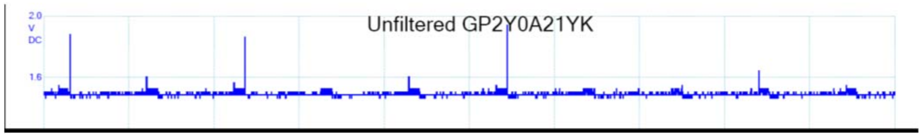
Sharp GP2Y0A21YK

- Infrared distance sensor
  - Distance to analog voltage
  - Powered by 5V
    - 10 uF or larger +5V to Gnd cap for each sensor
  - Needs analog LPF
    - Reduces noise
    - Analog input protection
  - Needs digital median filter
  - Needs calibration

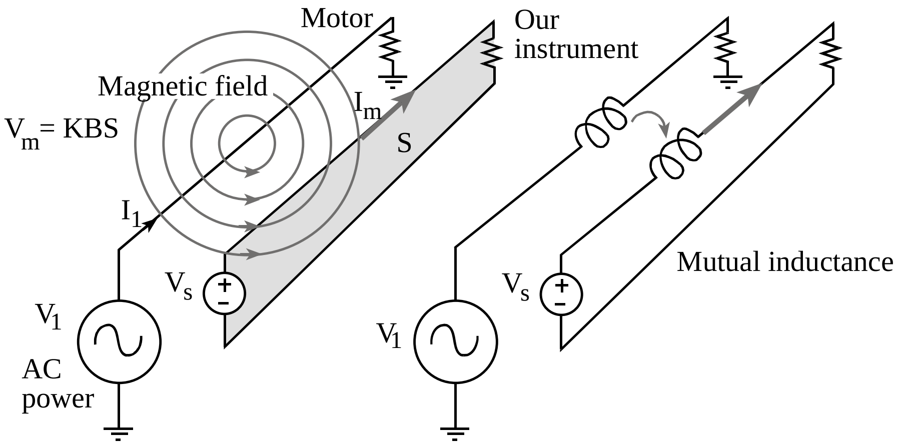
Sharp GP2Y0A21YK

- Noise => Filtering
- Accuracy => Calibration
- Precision => Resolution

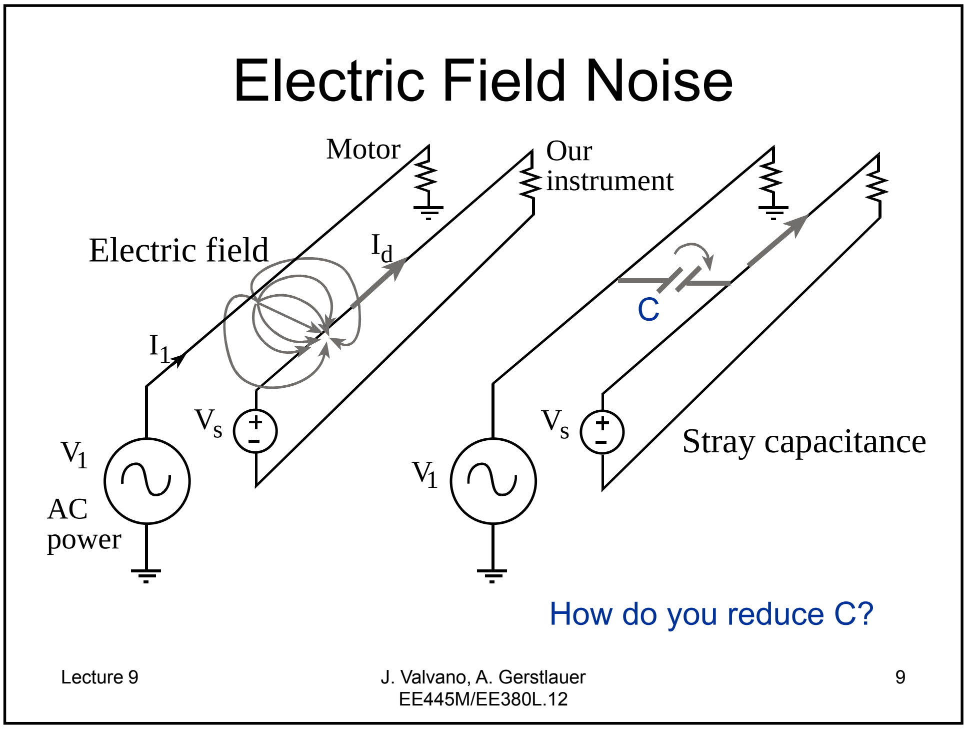
ADC = 6707/d+40
d = 6707/(ADC-40)
d (0.01cm) = 6706700/(ADC-40)



Magnetic Field Noise



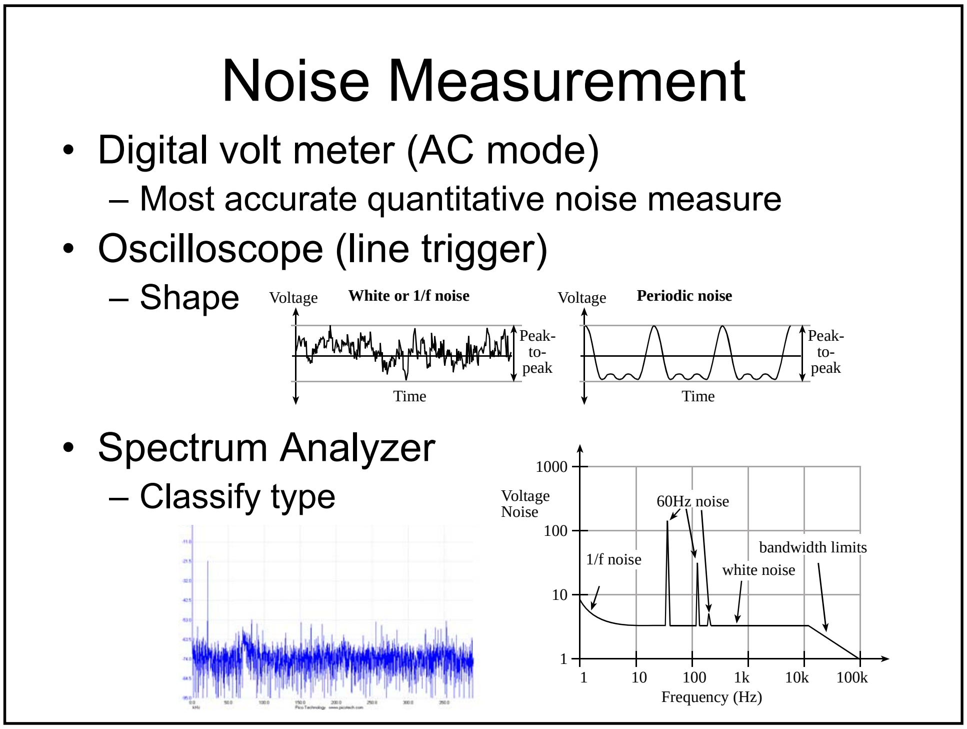
How do you reduce S?



Electric Field Noise

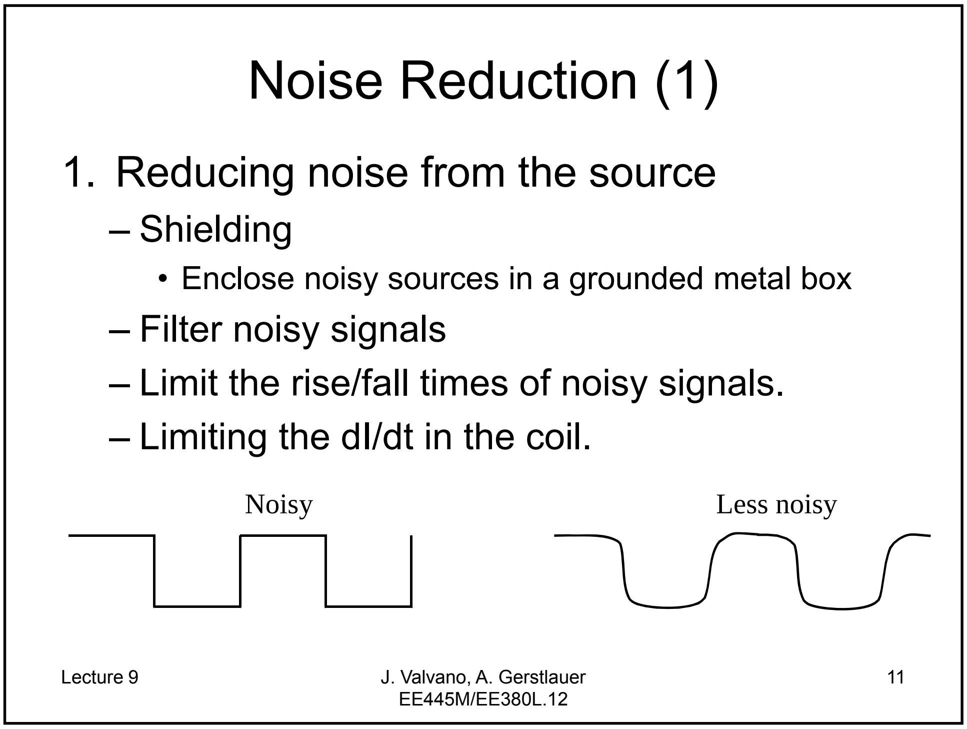
How do you reduce C?

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 9



Noise Measurement

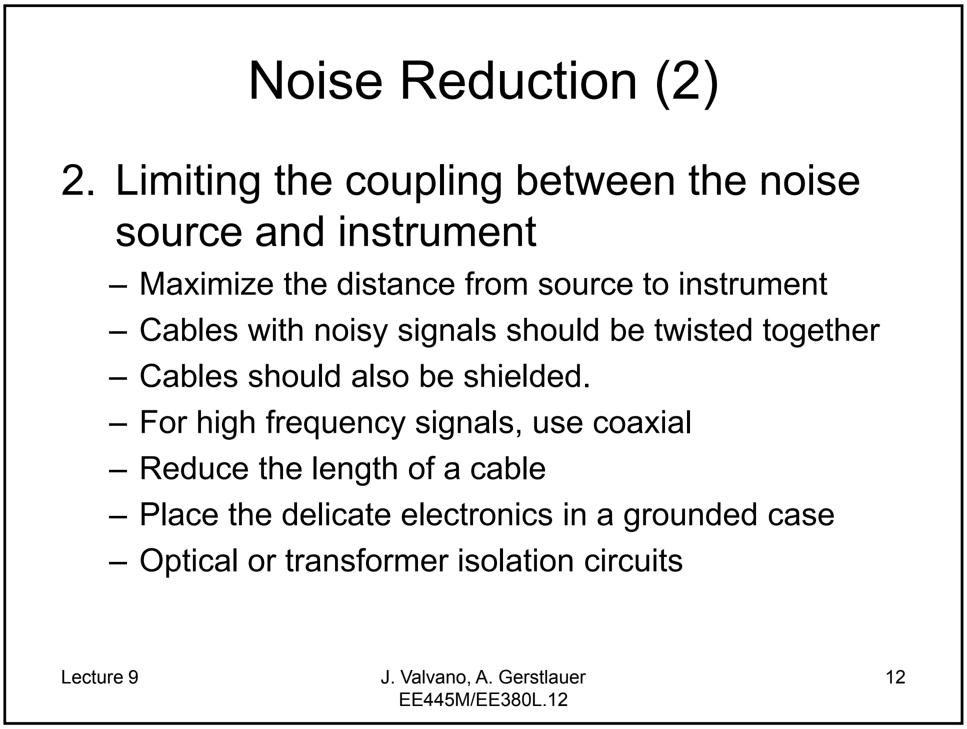
- Digital volt meter (AC mode)
  - Most accurate quantitative noise measure
- Oscilloscope (line trigger)
  - Shape
- Spectrum Analyzer
  - Classify type



Noise Reduction (1)

1. Reducing noise from the source
   - Shielding
     - Enclose noisy sources in a grounded metal box
   - Filter noisy signals
   - Limit the rise/fall times of noisy signals.
   - Limiting the dI/dt in the coil.

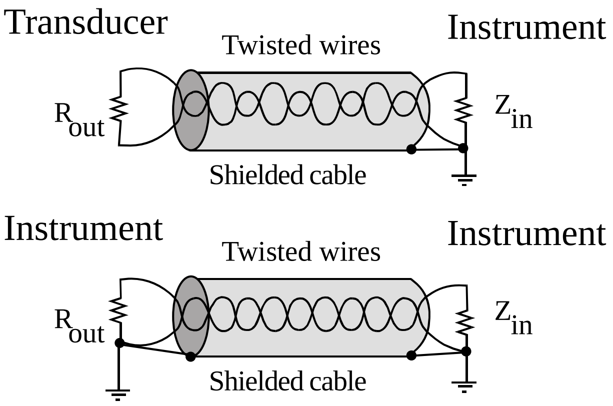
Noisy
Less noisy



Noise Reduction (2)

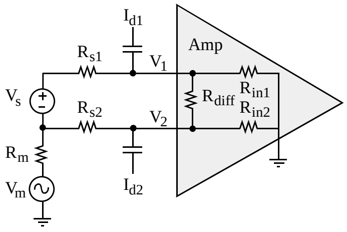
2. Limiting the coupling between the noise source and instrument
   - Maximize the distance from source to instrument
   - Cables with noisy signals should be twisted together
   - Cables should also be shielded.
   - For high frequency signals, use coaxial
   - Reduce the length of a cable
   - Place the delicate electronics in a grounded case
   - Optical or transformer isolation circuits

Limiting the Coupling



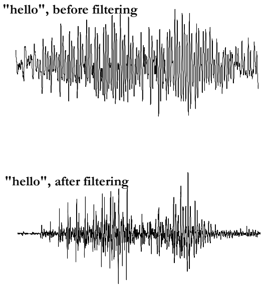
Noise Reduction (3)

3. Reduce the noise at the receiver
   - Bandwidth should be as small as possible
   - Add frequency-reject filters
   - Use power supply decoupling capacitors on each
   - Twisted wires then Id1 should equal Id2
     - V1-V2 = Rs1 Id1 - Rs2 Id2



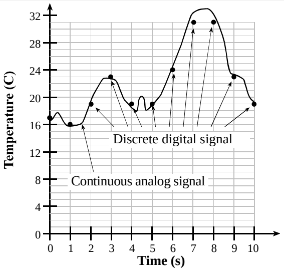
Henry Ott, Noise Reduction Techniques in Electronic Systems, Wiley, 1988.
Ralph Morrison, Grounding and Shielding Techniques, Wiley, 1998.

Filter Types

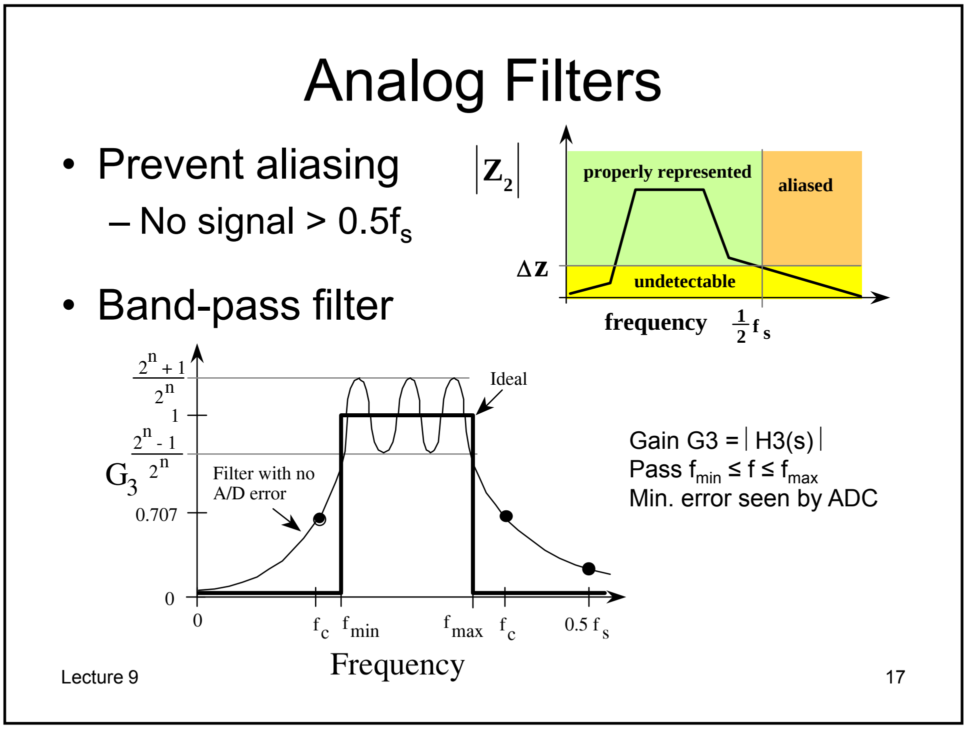


- Analog
  - Low pass filter (LPF)
  - High pass filter (HPF)
  - Band pass filter (BPF)
- Digital
  - Extremely flexible
  - But only available after sampling

Sampling



- Time & value quantizing
  - Precision $n_z = 2^n$
- Nyquist theory
  - If sampled at $\mathbf{f_s}$, digital samples only contain frequency components from 0 to ½$\mathbf{f_s}$
  - If analog signal contains frequency components larger than ½$\mathbf{f_s}$, **aliasing** error
- System design
  - Choice of sampling rate: $\mathbf{f_s > 2\, f_{max}}$
  - Low pass analog filter to remove frequency components above 0.5$\mathbf{f_s}$
    - A digital filter can not be used to remove aliasing

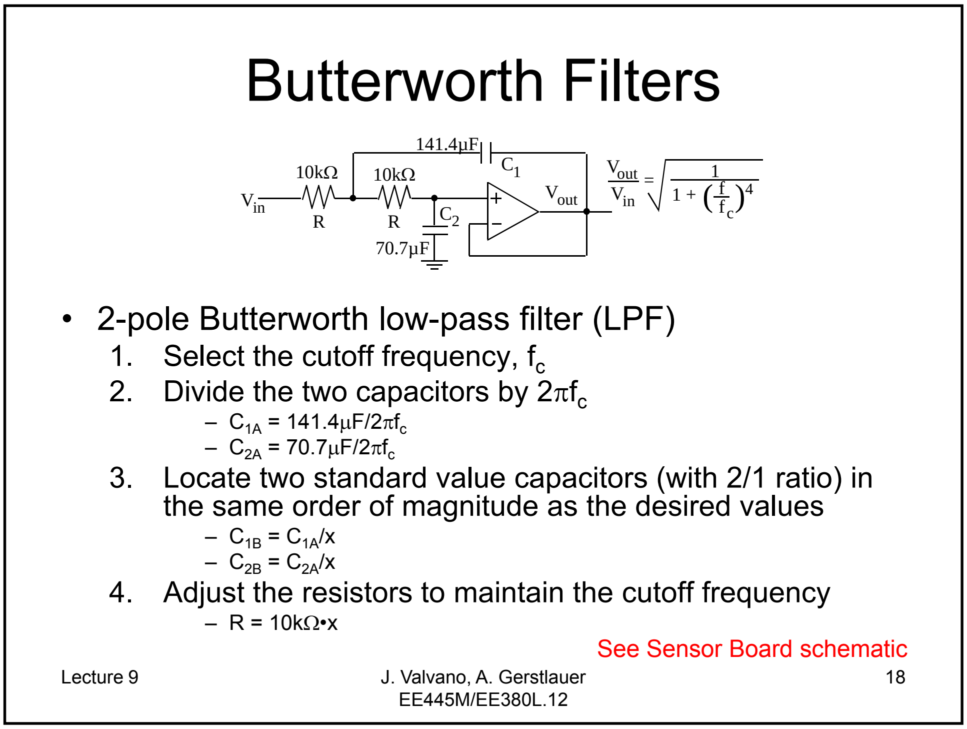


Analog Filters

- Prevent aliasing
  - No signal > $0.5f_s$
- Band-pass filter

Gain G3 = | H3(s) |
Pass $f_{min} \le f \le f_{max}$
Min. error seen by ADC

Lecture 9 17



Butterworth Filters

$$\frac{V_{out}}{V_{in}} = \sqrt{\frac{1}{1+\left(\frac{f}{f_c}\right)^4}}$$

- 2-pole Butterworth low-pass filter (LPF)
  1. Select the cutoff frequency, f_c
  2. Divide the two capacitors by $2\pi f_c$
     - $C_{1A} = 141.4\mu F/2\pi f_c$
     - $C_{2A} = 70.7\mu F/2\pi f_c$
  3. Locate two standard value capacitors (with 2/1 ratio) in the same order of magnitude as the desired values
     - $C_{1B} = C_{1A}/x$
     - $C_{2B} = C_{2A}/x$
  4. Adjust the resistors to maintain the cutoff frequency
     - R = 10kΩ•x

See Sensor Board schematic

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 18

Digital Filters

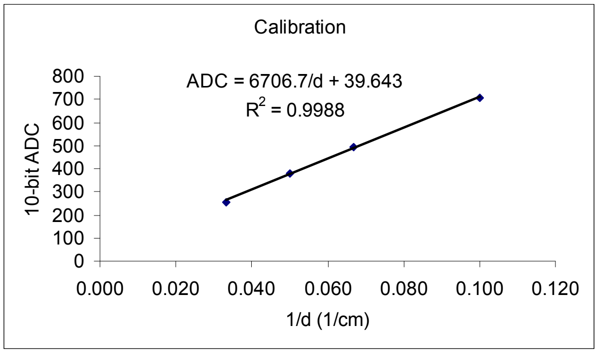
- Finite/Infinite Impulse Response (FIR/IIR)
  - Linear: HPF, LPF, BPF/notch
- Median filter
  - Non-linear: preserves edges, removes spikes

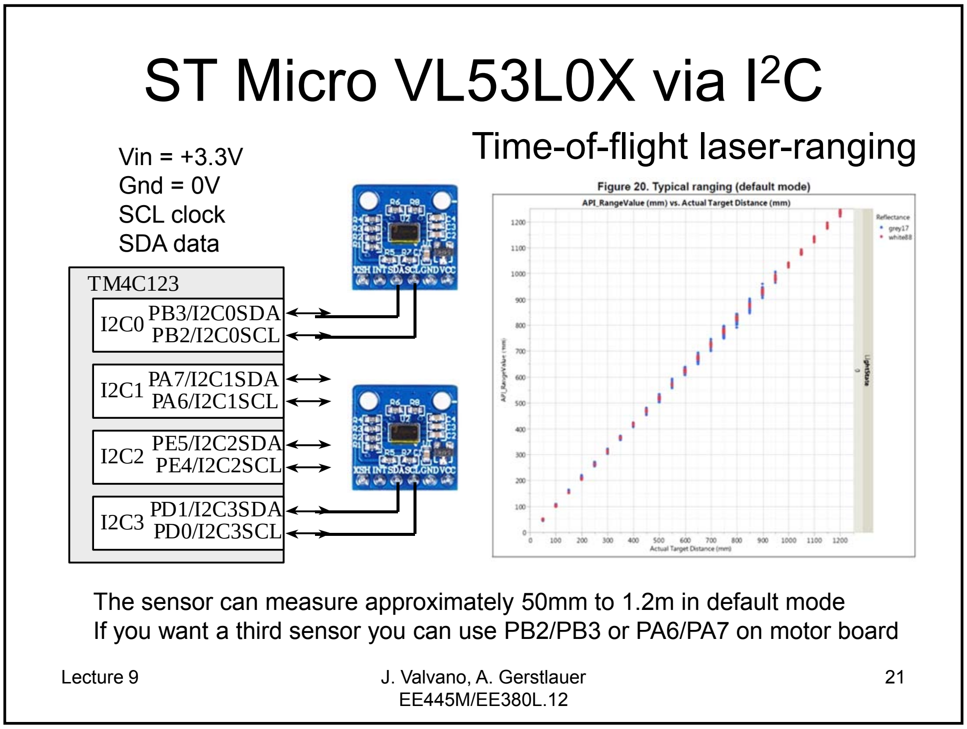
```
// Non-recursive, 3-point median filter
uint8_t Median(uint8_t u1,uint8_t u2,uint8_t u3) {
  if(u1>u2)
    if(u2>u3) return u2;   // u1>u2,u2>u3       u1>u2>u3
    if(u1>u3) return u3;   // u1>u2,u3>u2,u1>u3 u1>u3>u2
    return u1;             // u1>u2,u3>u2,u3>u1 u3>u1>u2
  else
    if(u3>u2) return u2;   // u2>u1,u3>u2       u3>u2>u1
    if(u1>u3) return u1;   // u2>u1,u2>u3,u1>u3 u2>u1>u3
    return u3;             // u2>u1,u2>u3,u3>u1 u2>u3>u1
}
```

Reference book, Chapter 6

IR Sensor Calibration

d (cm)	1/d	ADC
10	0.100	703
15	0.067	484
20	0.050	380
30	0.033	260





ST Micro VL53L0X via I^{2}C

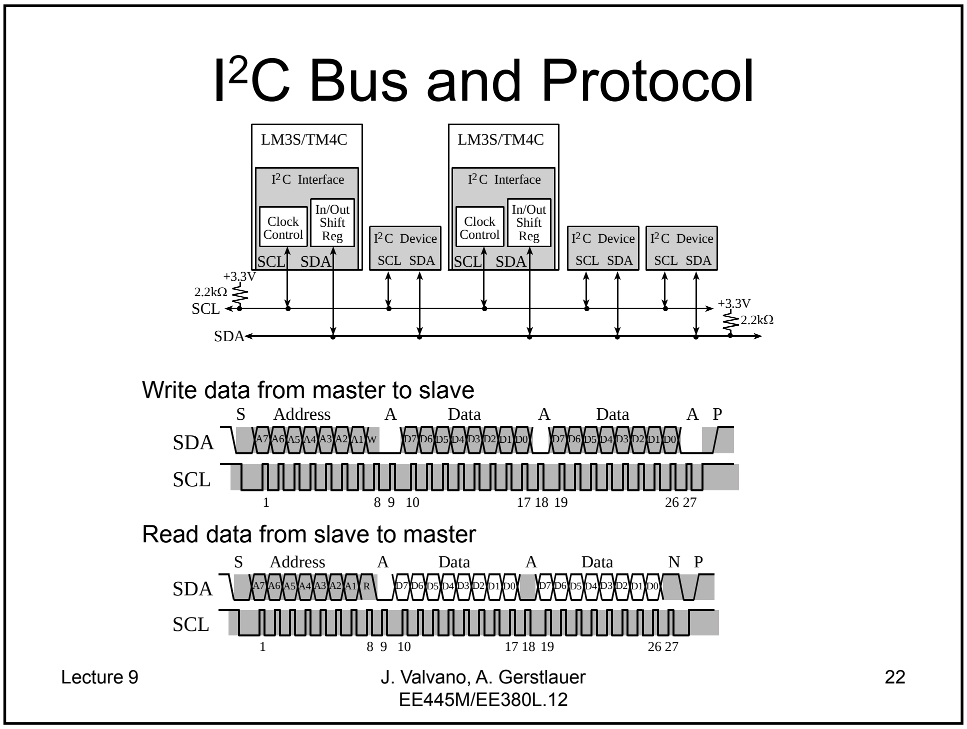
Vin = +3.3V
Gnd = 0V
SCL clock
SDA data

Time-of-flight laser-ranging

TM4C123
I2C0 PB3/I2C0SDA PB2/I2C0SCL
I2C1 PA7/I2C1SDA PA6/I2C1SCL
I2C2 PE5/I2C2SDA PE4/I2C2SCL
I2C3 PD1/I2C3SDA PD0/I2C3SCL
XSH INT SDA SCL GND VCC
Figure 20. Typical ranging (default mode)
API_RangeValue (mm) vs. Actual Target Distance (mm)
API_RangeValue (mm)
Actual Target Distance (mm)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
Reflectance
grey17
white88
LightState
0

The sensor can measure approximately 50mm to 1.2m in default mode
If you want a third sensor you can use PB2/PB3 or PA6/PA7 on motor board

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 21



I^{2}C Bus and Protocol

LM3S/TM4C
I^{2}C Interface
Clock Control
In/Out Shift Reg
SCL SDA
I^{2}C Device
SCL SDA
+3.3V
2.2kΩ
SCL
SDA

Write data from master to slave

S Address A Data A Data A P
SDA A7 A6 A5 A4 A3 A2 A1 W D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
SCL
1 8 9 10 17 18 19 26 27

Read data from slave to master

S Address A Data A Data N P
SDA A7 A6 A5 A4 A3 A2 A1 R D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
SCL
1 8 9 10 17 18 19 26 27

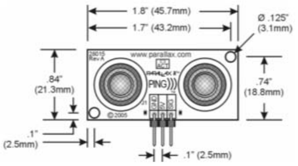
Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 22

Software Example

```
int main(void) {
  PLL_Init(Bus80MHz);      // bus clock at 80 MHz
  Serial_Init();
  VL53L0X_Init(VL53L0X_I2C_ADDR); // initialize sensor
  VL53L0X_RangingMeasurementData_t measurement;
  while(1){
    Serial_println("Measuring... ");
    VL53L0X_getSingleRangingMeasurement(&measurement);
    if(measurement.RangeStatus != 4) {
      Serial_println("D= %u mm", measurement.RangeMilliMeter);
    } else {
      Serial_println("Out of range :(");
    }
    delay(1000);
  }
}
```

Ping Distance Sensor

- Ultrasound transducers to measure distance
  - Ping)))
    - One **SIG** pin for both input & output

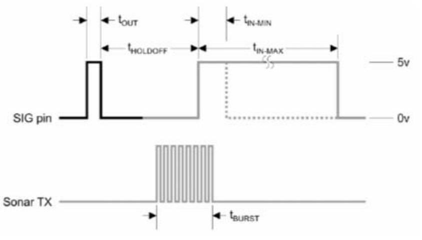


  - HCSR04
    - Two signals: **Trig** output and **Echo** input

  - Need 5V to power
    - Use 5V tolerant input (PD4, PD5, PB0 and PB1 must be <3.6V)

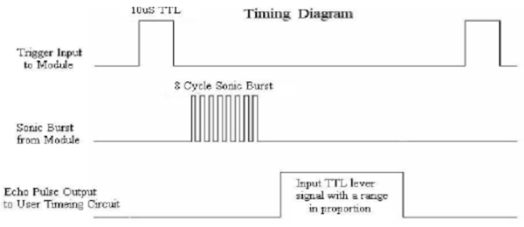
Ping))) Sensor

- Sample 10 times a second
  1) Disable interrupts
  2) Make the **SIG** pin an output
  3) Issue a 5μs output pulse (causing a sound pulse)
  4) Switch the **SIG** pin back to an input
  5) Enable interrupts
  6) Measure time until the echo is received
     - Busy-wait if foreground, interrupt if background



HCSR04 Sensor

- Sample 10 times a second
  1) Disable interrupts
  2) Issue a 10μs output pulse (causing a sound pulse)
  3) Enable interrupts
  4) Measure time until the echo is received
     - Busy-wait if foreground, interrupt if background


10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

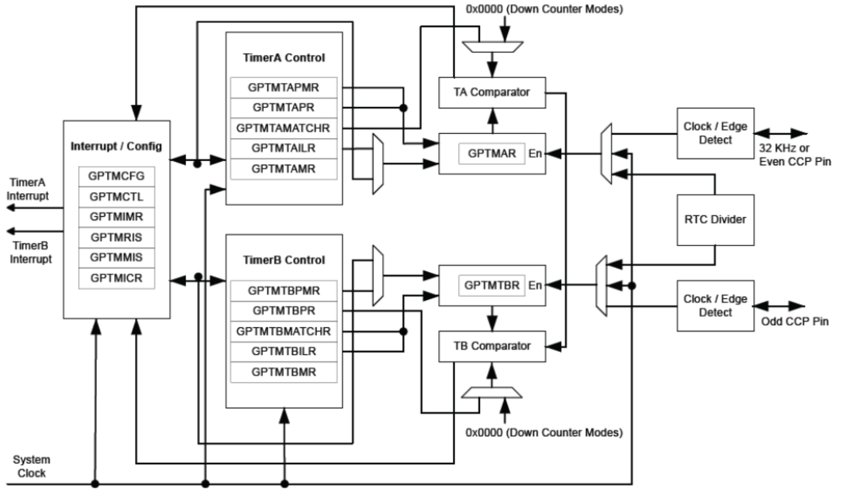
Input Capture

- General purpose timers
  - – TM4C123: 6 GP timers (Timer 0…Timer 5)
  - – CCPx pins used for input capture
- Input edge time (input capture) mode
  - – Detect rising/falling input edges
  - – Make time measurements on input signals

See book Section 8.1

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 27

General Purpose Timer



Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 28

Input Capture Mode

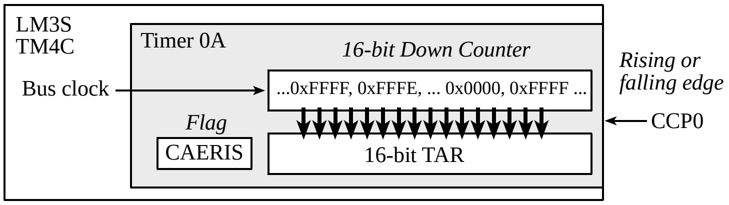


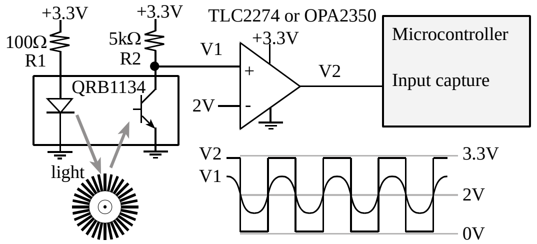
Figure 8.2. Rising or falling edge of CCP0 causes the counter to be latched into TAR, setting CAERIS.

- Generate edge based interrupts
- Count events
- Measure period
- Measure pulse width

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 29

Event Counting

- Count wheel turns (tachometer)



InputCapture_4C123.zip

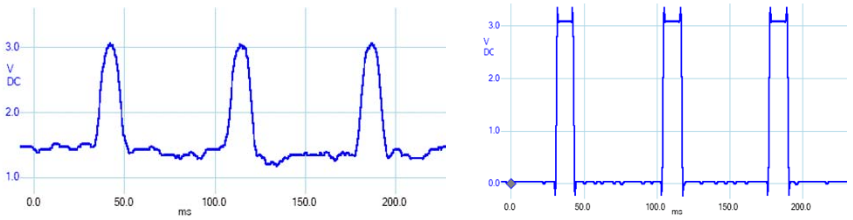


Figure 8.4. Measured V1 and V2

Period Measurement

- Init
  - Select clock period, Δt (measurement resolution)
  - TIMER0_TAILR_R = 0xFFFF (reload=wraparound)
  - Choose edge (rise or fall)
  - Arm interrupt on capture
- ISR
  - Poll to see which channel (if needed)
  - Now = captured time (TIMER0_TAR_R)
  - Period = Last – Now
  - Last = Now
  - Acknowledge interrupt
  - Save/process period

PeriodMeasure_4C123.zip

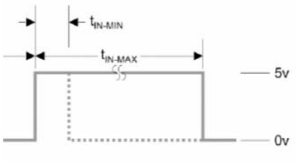
Resolution, Precision, Range

- How to choose the resolution?
  - Determine minimum & maximum robot speed
  - Convert speed to tachometer period

Period	7100	
	4	holes/rotation
Resolution	10	µsec
Speed	3.521127	rps
Speed	211.2676	RPM

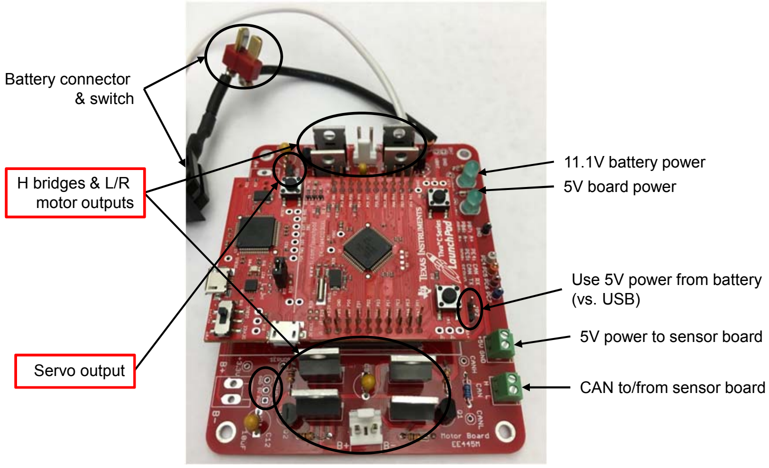
- How to detect speed too slow (period too large)?
  - Clear a counter on each tachometer edge
  - AddPeriodicThread
    - Increment the counter on each rollover 0000 to FFFF
    - If counter >= 2, then wheel is stopped

Ping Distance Measurement

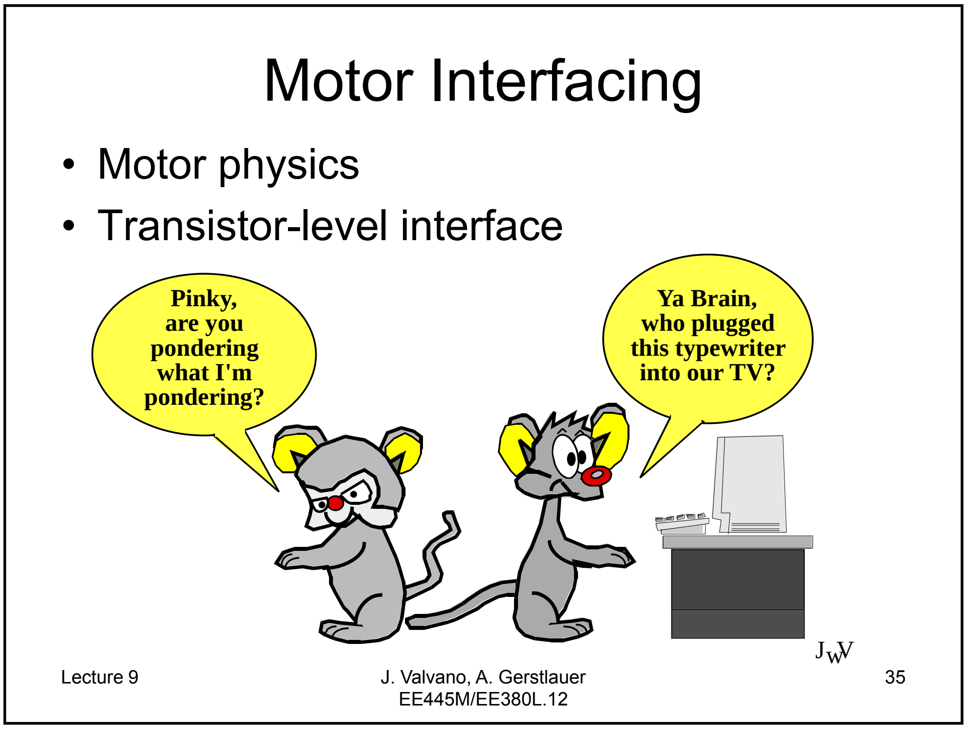


- Input pulse width
  - Time $\mathbf{t_{IN}}$ for sound to travel back and forth
  - $\mathbf{t_{IN}} = 2\ \mathbf{d/c}$ (**c**: speed of sound)
- Measure using input capture
  - Rising edge: record TAR
  - Falling edge: calculate distance $\mathbf{d} = \mathbf{c} * \mathbf{t_{IN}}/2$

Motor Board



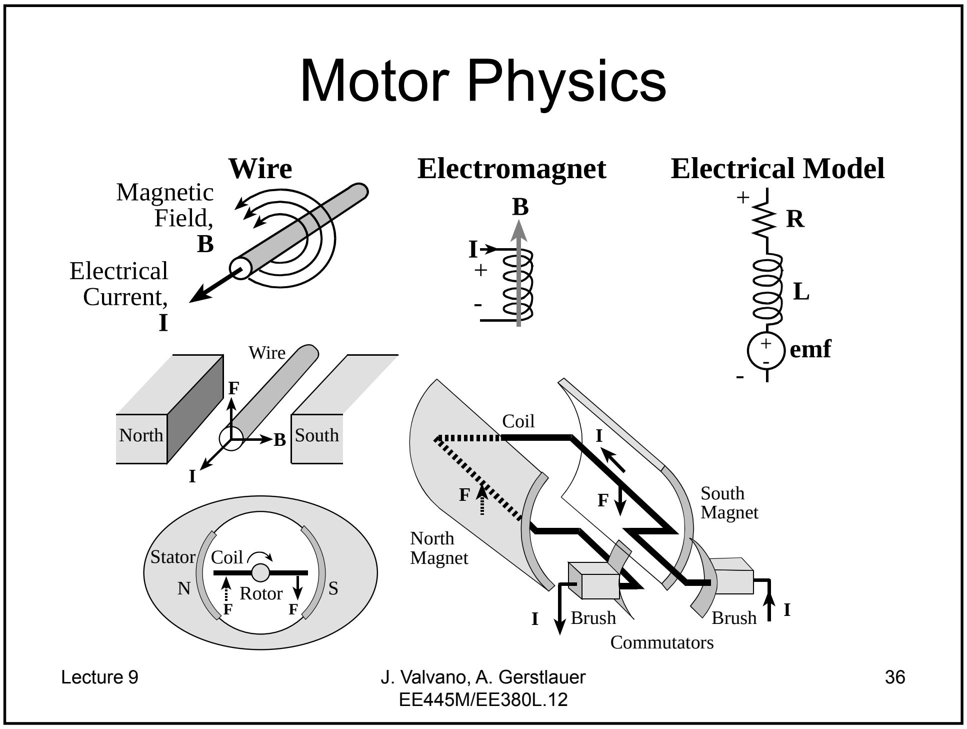
- **Reference material**
  - **Schematic:** http://www.ece.utexas.edu/~gerstl/ee445m_s18/resources/Robot_Motor_v6.pdf
  - **PCB layout:** http://www.ece.utexas.edu/~gerstl/ee445m_s18/resources/Robot_Motor_v6_top.png



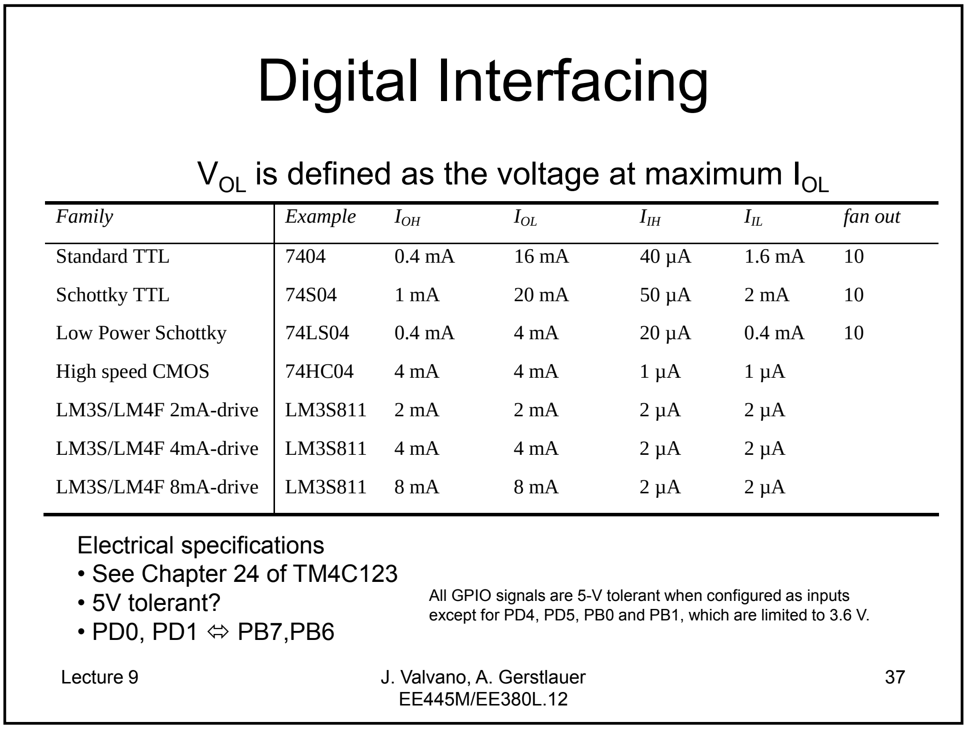
Motor Interfacing

- Motor physics
- Transistor-level interface

Pinky, are you pondering what I'm pondering?
Ya Brain, who plugged this typewriter into our TV?
JWV



Motor Physics



Digital Interfacing

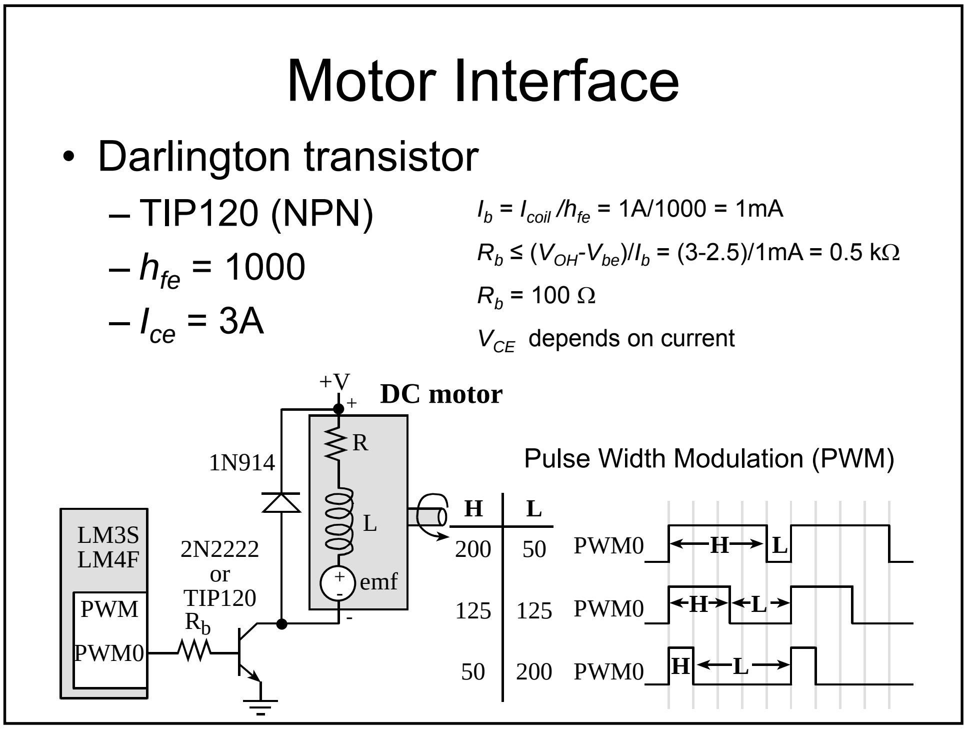
V_{OL} is defined as the voltage at maximum I_{OL}

Family	*Example*	I_{OH}	I_{OL}	I_{IH}	I_{IL}	*fan out*
Standard TTL	7404	0.4 mA	16 mA	40 µA	1.6 mA	10
Schottky TTL	74S04	1 mA	20 mA	50 µA	2 mA	10
Low Power Schottky	74LS04	0.4 mA	4 mA	20 µA	0.4 mA	10
High speed CMOS	74HC04	4 mA	4 mA	1 µA	1 µA	
LM3S/LM4F 2mA-drive	LM3S811	2 mA	2 mA	2 µA	2 µA	
LM3S/LM4F 4mA-drive	LM3S811	4 mA	4 mA	2 µA	2 µA	
LM3S/LM4F 8mA-drive	LM3S811	8 mA	8 mA	2 µA	2 µA	

Electrical specifications
- See Chapter 24 of TM4C123
- 5V tolerant?
- PD0, PD1 ⇔ PB7,PB6

All GPIO signals are 5-V tolerant when configured as inputs except for PD4, PD5, PB0 and PB1, which are limited to 3.6 V.

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 37



Motor Interface

- Darlington transistor
  - TIP120 (NPN)
  - h_{fe} = 1000
  - I_{ce} = 3A

$I_b = I_{coil} / h_{fe}$ = 1A/1000 = 1mA

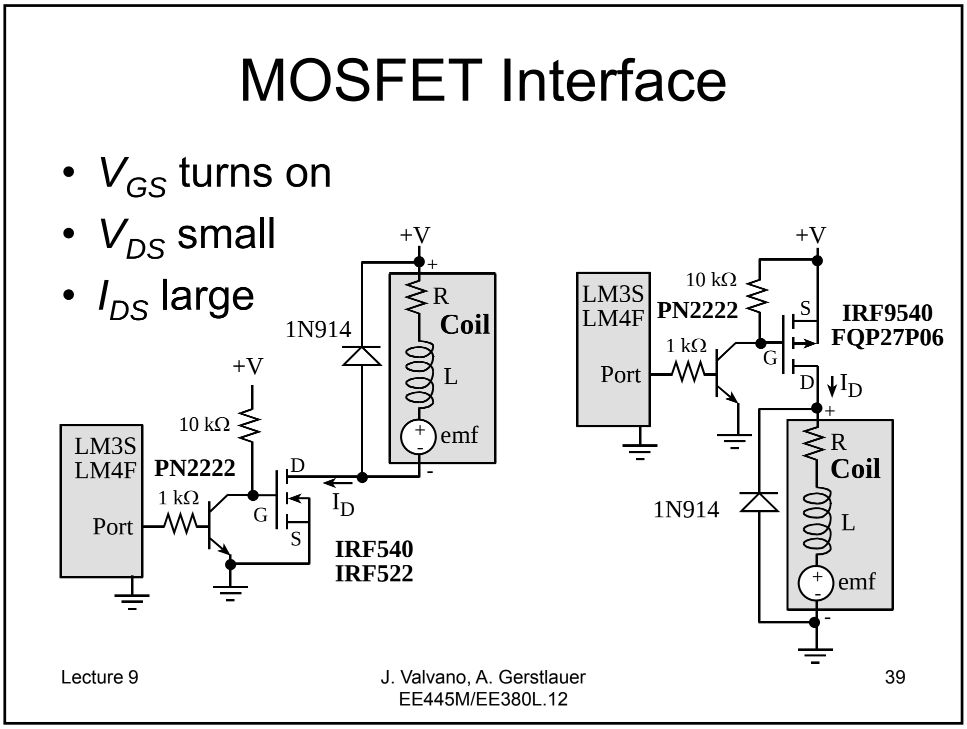
$R_b \le (V_{OH}-V_{be})/I_b$ = (3-2.5)/1mA = 0.5 kΩ

R_b = 100 Ω

V_{CE} depends on current

Pulse Width Modulation (PWM)

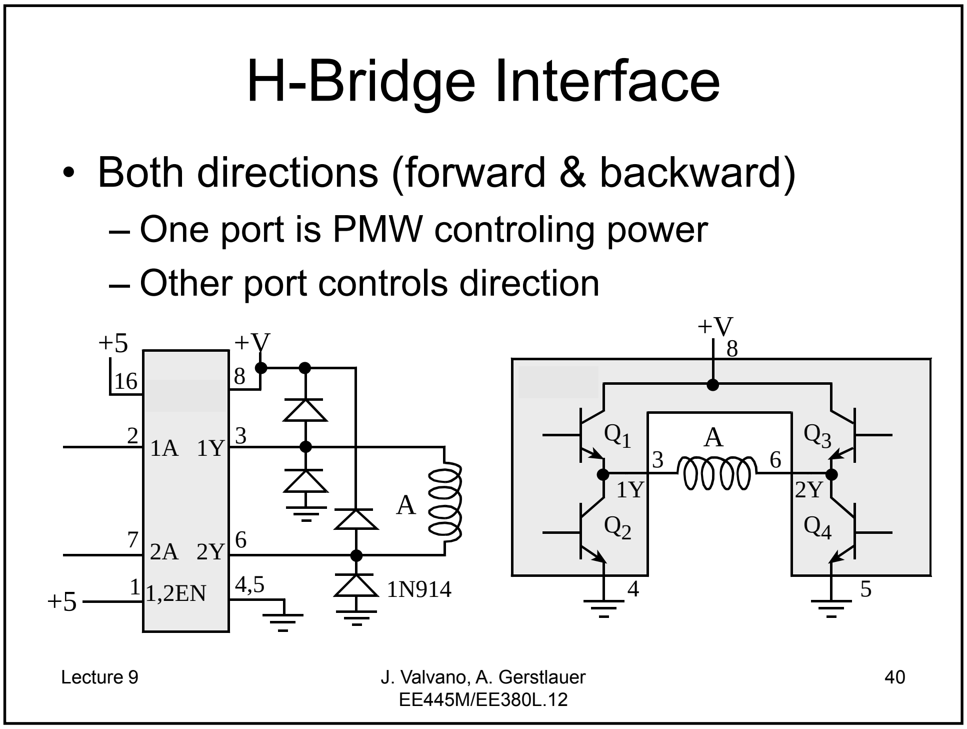
H	L	
200	50	PWM0
125	125	PWM0
50	200	PWM0



MOSFET Interface

- V_{GS} turns on
- V_{DS} small
- I_{DS} large

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 39



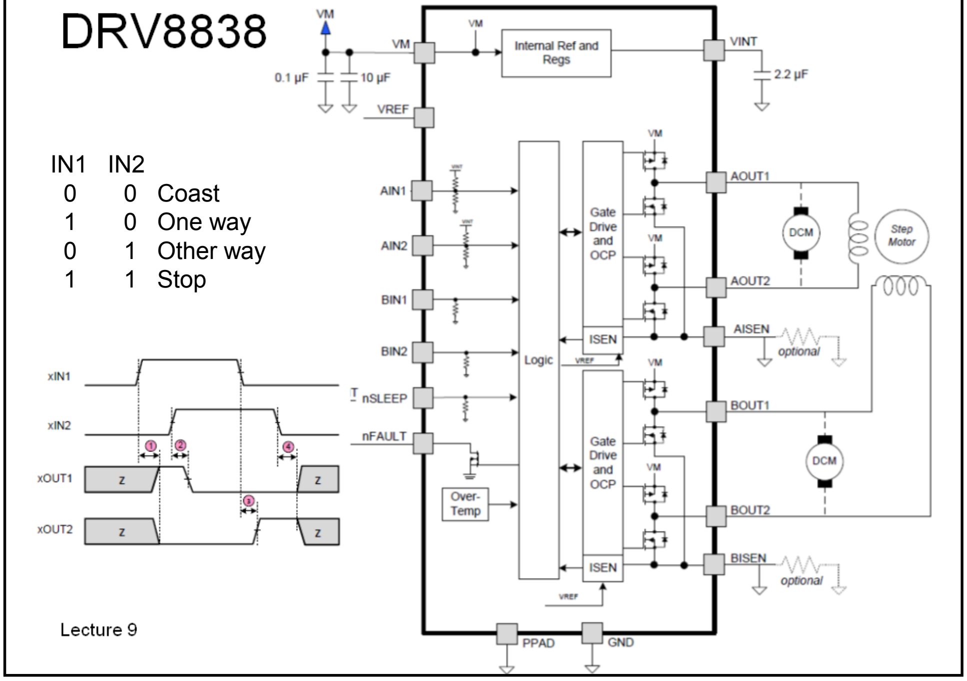
H-Bridge Interface

- Both directions (forward & backward)
  - One port is PMW controling power
  - Other port controls direction

Lecture 9 J. Valvano, A. Gerstlauer EE445M/EE380L.12 40

DRV8838

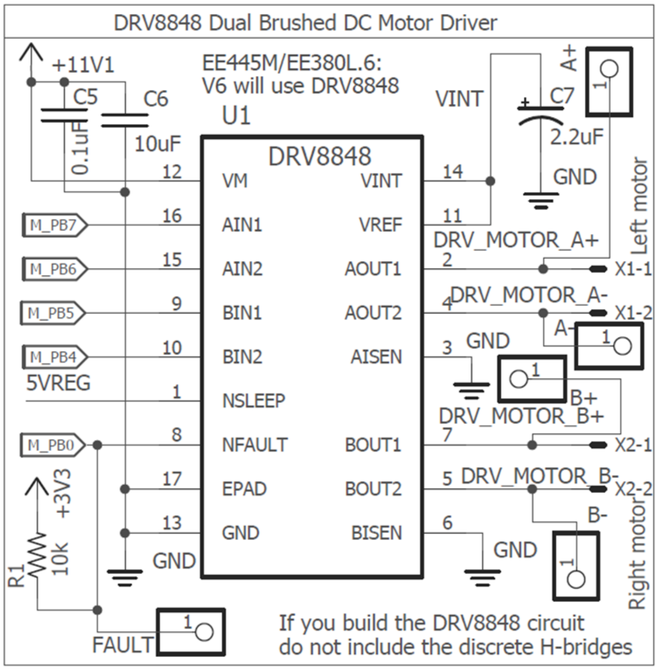
IN1	IN2	
0	0	Coast
1	0	One way
0	1	Other way
1	1	Stop



Lecture 9

H Bridge

- PWM
- Direction

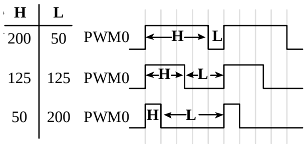


If you build the DRV8848 circuit do not include the discrete H-bridges

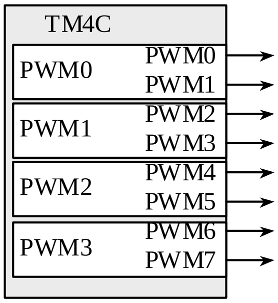
Lecture 9

Pulse Width Modulation (PWM)

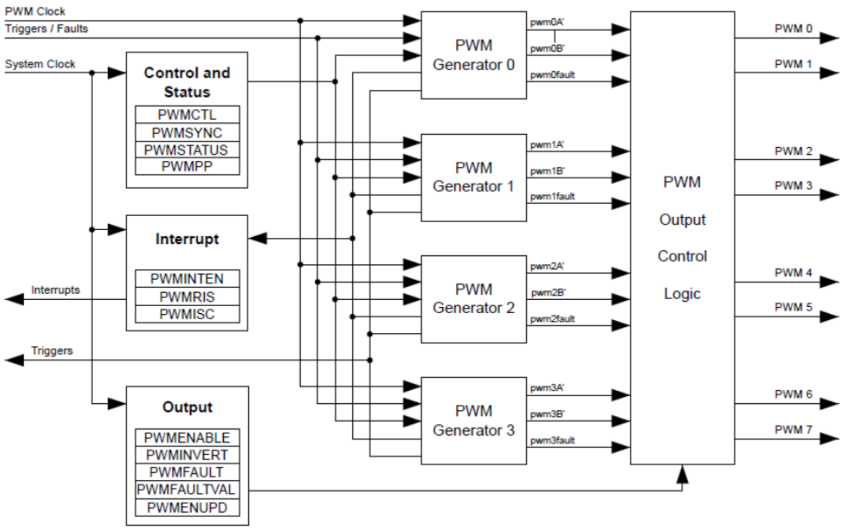
- Generate output waveform
  - Period = High + Low
  - Duty cycle = High / Period

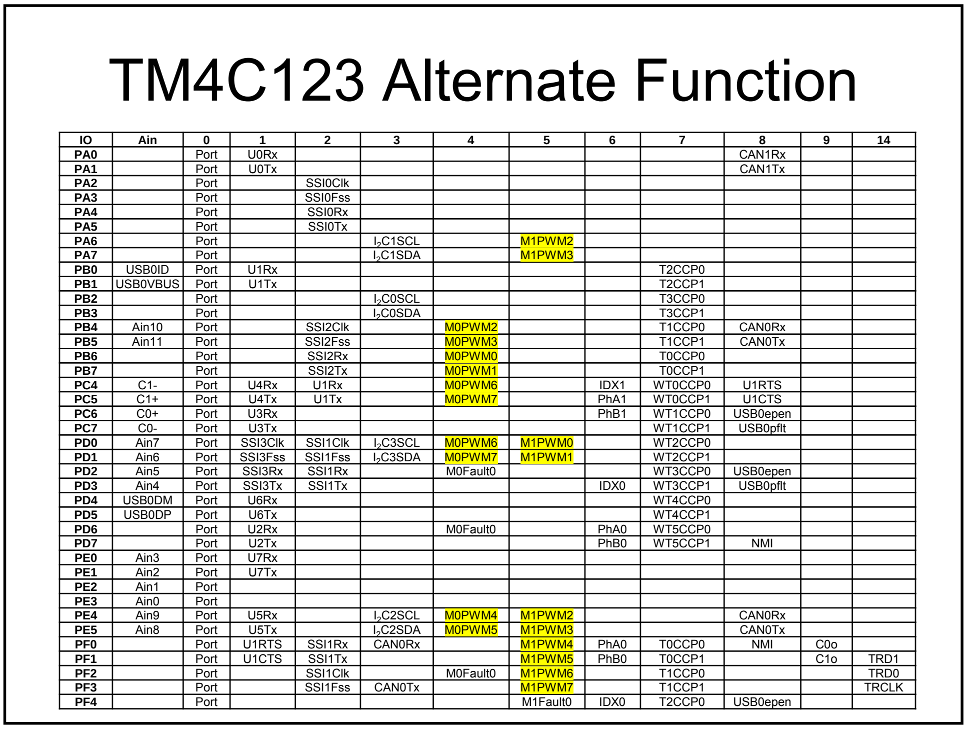


- PWM generators
  - TM4C123: 2 modules
    - 4 generators per module
    - 2 PWM signals per generator



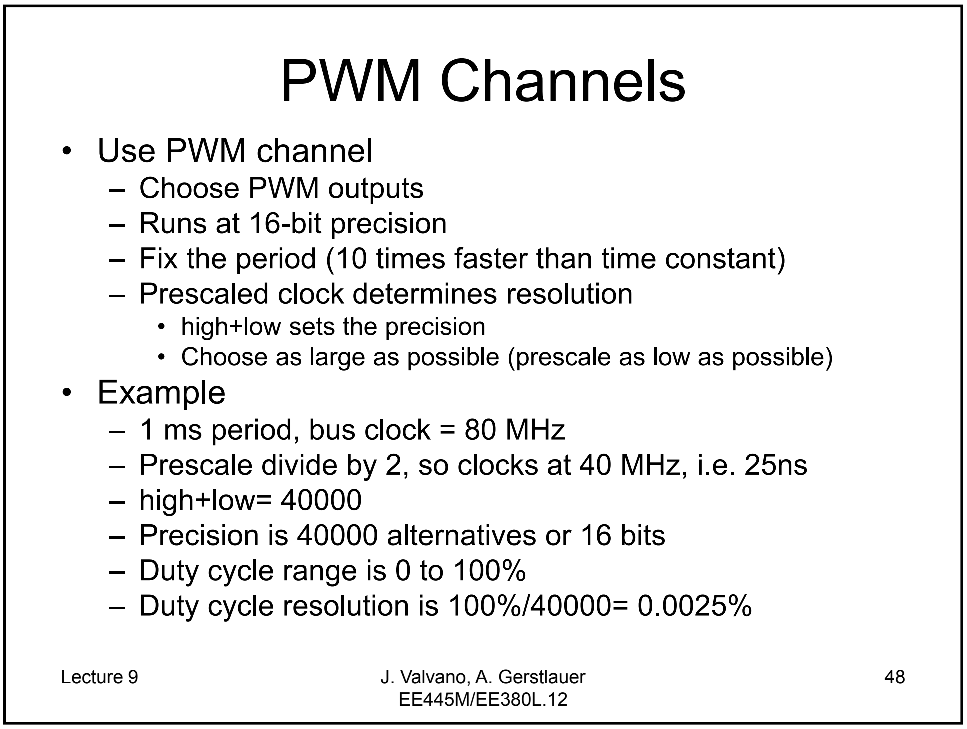
PWM Module





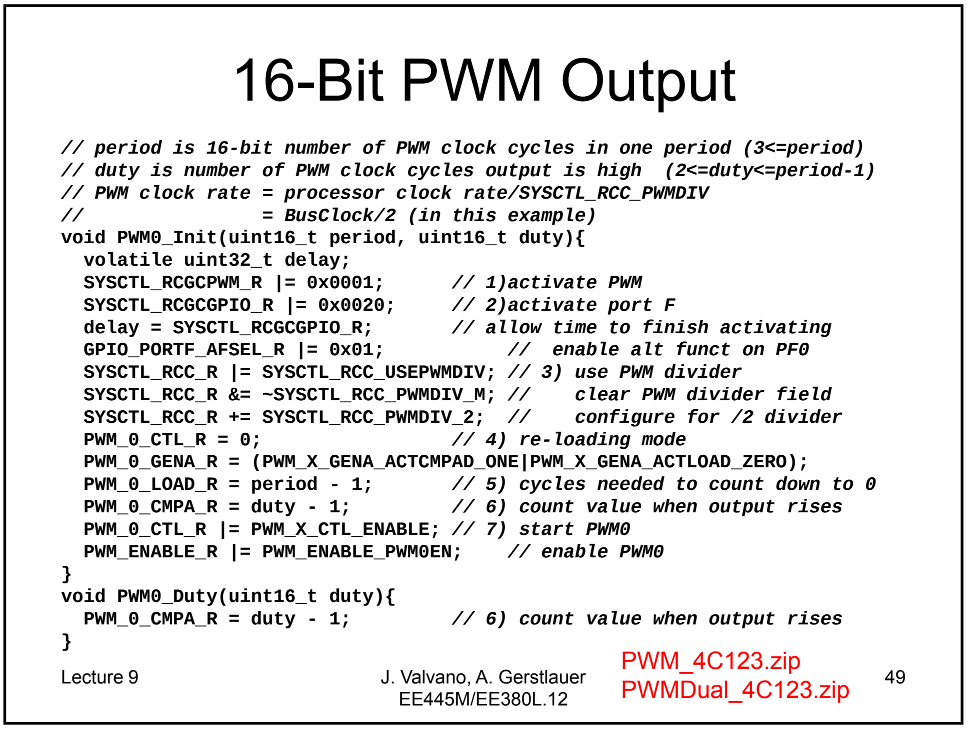
TM4C123 Alternate Function

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PA0		Port	U0Rx							CAN1Rx		
PA1		Port	U0Tx							CAN1Tx		
PA2		Port		SSI0Clk								
PA3		Port		SSI0Fss								
PA4		Port		SSI0Rx								
PA5		Port		SSI0Tx								
PA6		Port			I_2C1SCL		M1PWM2					
PA7		Port			I_2C1SDA		M1PWM3					
PB0	USB0ID	Port	U1Rx						T2CCP0			
PB1	USB0VBUS	Port	U1Tx						T2CCP1			
PB2		Port			I_2C0SCL				T3CCP0			
PB3		Port			I_2C0SDA				T3CCP1			
PB4	Ain10	Port		SSI2Clk		M0PWM2			T1CCP0	CAN0Rx		
PB5	Ain11	Port		SSI2Fss		M0PWM3			T1CCP1	CAN0Tx		
PB6		Port		SSI2Rx		M0PWM0			T0CCP0			
PB7		Port		SSI2Tx		M0PWM1			T0CCP1			
PC4	C1-	Port	U4Rx	U1Rx		M0PWM6		IDX1	WT0CCP0	U1RTS		
PC5	C1+	Port	U4Tx	U1Tx		M0PWM7		PhA1	WT0CCP1	U1CTS		
PC6	C0+	Port	U3Rx					PhB1	WT1CCP0	USB0epen		
PC7	C0-	Port	U3Tx						WT1CCP1	USB0pflt		
PD0	Ain7	Port	SSI3Clk	SSI1Clk	I_2C3SCL	M0PWM6	M1PWM0		WT2CCP0			
PD1	Ain6	Port	SSI3Fss	SSI1Fss	I_2C3SDA	M0PWM7	M1PWM1		WT2CCP1			
PD2	Ain5	Port	SSI3Rx	SSI1Rx		M0Fault0			WT3CCP0	USB0epen		
PD3	Ain4	Port	SSI3Tx	SSI1Tx				IDX0	WT3CCP1	USB0pflt		
PD4	USB0DM	Port	U6Rx						WT4CCP0			
PD5	USB0DP	Port	U6Tx						WT4CCP1			
PD6		Port	U2Rx			M0Fault0		PhA0	WT5CCP0			
PD7		Port	U2Tx					PhB0	WT5CCP1	NMI		
PE0	Ain3	Port	U7Rx									
PE1	Ain2	Port	U7Tx									
PE2	Ain1	Port										
PE3	Ain0	Port										
PE4	Ain9	Port	U5Rx		I_2C2SCL	M0PWM4	M1PWM2			CAN0Rx		
PE5	Ain8	Port	U5Tx		I_2C2SDA	M0PWM5	M1PWM3			CAN0Tx		
PF0		Port	U1RTS	SSI1Rx	CAN0Rx		M1PWM4	PhA0	T0CCP0	NMI	C0o	
PF1		Port	U1CTS	SSI1Tx			M1PWM5	PhB0	T0CCP1		C1o	TRD1
PF2		Port		SSI1Clk		M0Fault0	M1PWM6		T1CCP0			TRD0
PF3		Port		SSI1Fss	CAN0Tx		M1PWM7		T1CCP1			TRCLK
PF4		Port					M1Fault0	IDX0	T2CCP0	USB0epen		



PWM Channels

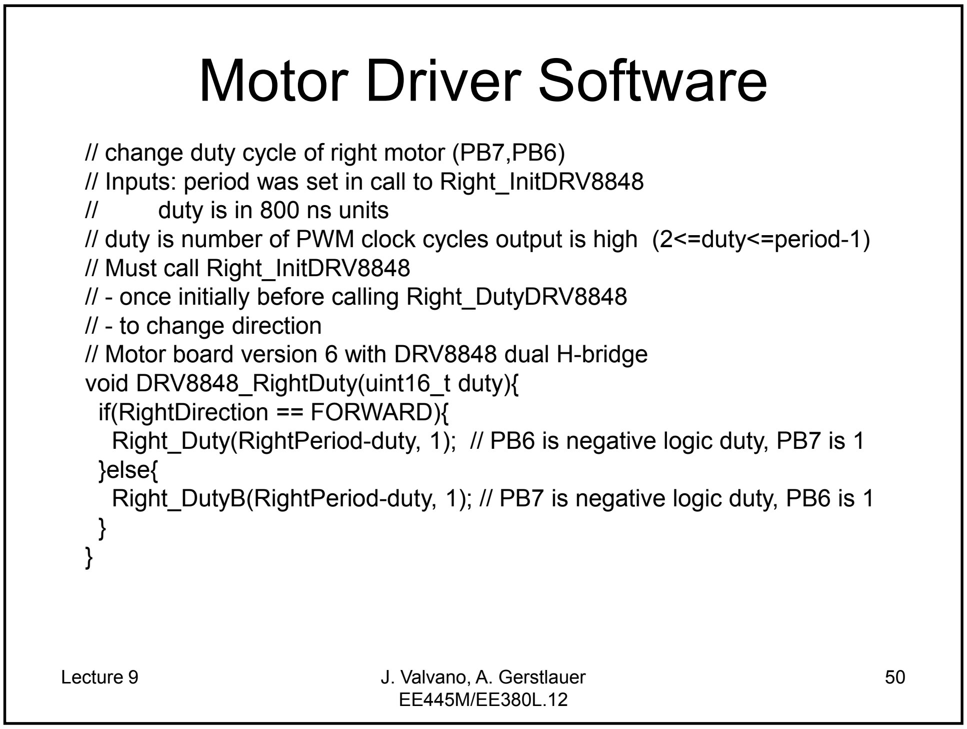
- Use PWM channel
  - Choose PWM outputs
  - Runs at 16-bit precision
  - Fix the period (10 times faster than time constant)
  - Prescaled clock determines resolution
    - high+low sets the precision
    - Choose as large as possible (prescale as low as possible)
- Example
  - 1 ms period, bus clock = 80 MHz
  - Prescale divide by 2, so clocks at 40 MHz, i.e. 25ns
  - high+low= 40000
  - Precision is 40000 alternatives or 16 bits
  - Duty cycle range is 0 to 100%
  - Duty cycle resolution is 100%/40000= 0.0025%



16-Bit PWM Output

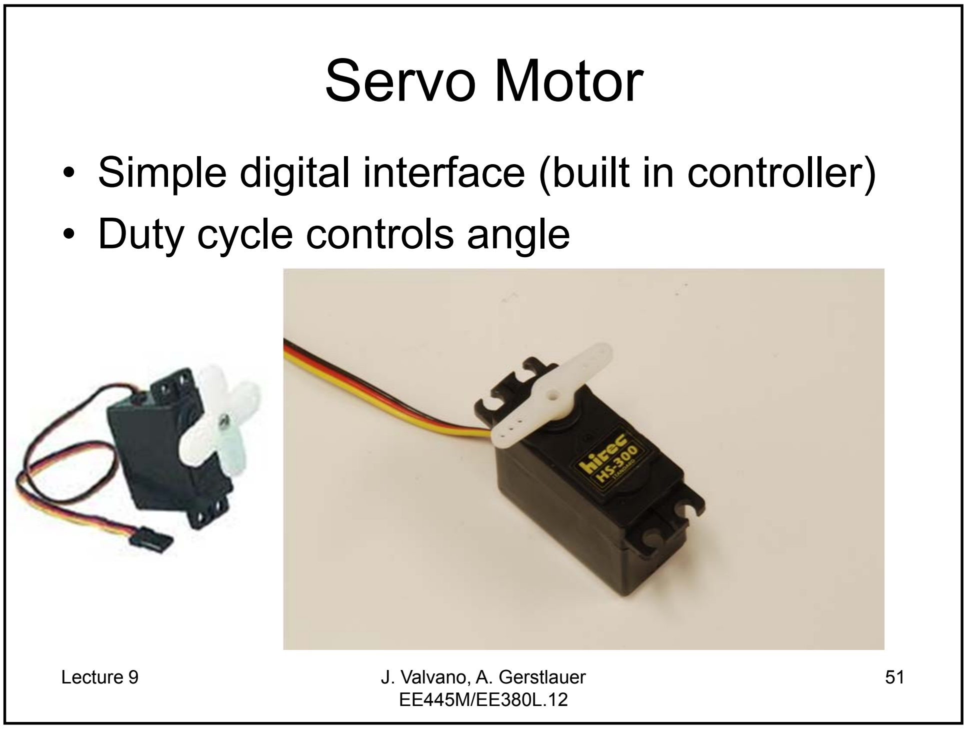
```
// period is 16-bit number of PWM clock cycles in one period (3<=period)
// duty is number of PWM clock cycles output is high  (2<=duty<=period-1)
// PWM clock rate = processor clock rate/SYSCTL_RCC_PWMDIV
//                = BusClock/2 (in this example)
void PWM0_Init(uint16_t period, uint16_t duty){
  volatile uint32_t delay;
  SYSCTL_RCGCPWM_R |= 0x0001;      // 1)activate PWM
  SYSCTL_RCGCGPIO_R |= 0x0020;     // 2)activate port F
  delay = SYSCTL_RCGCGPIO_R;       // allow time to finish activating
  GPIO_PORTF_AFSEL_R |= 0x01;           //  enable alt funct on PF0
  SYSCTL_RCC_R |= SYSCTL_RCC_USEPWMDIV; // 3) use PWM divider
  SYSCTL_RCC_R &= ~SYSCTL_RCC_PWMDIV_M; //    clear PWM divider field
  SYSCTL_RCC_R += SYSCTL_RCC_PWMDIV_2;  //    configure for /2 divider
  PWM_0_CTL_R = 0;                 // 4) re-loading mode
  PWM_0_GENA_R = (PWM_X_GENA_ACTCMPAD_ONE|PWM_X_GENA_ACTLOAD_ZERO);
  PWM_0_LOAD_R = period - 1;       // 5) cycles needed to count down to 0
  PWM_0_CMPA_R = duty - 1;         // 6) count value when output rises
  PWM_0_CTL_R |= PWM_X_CTL_ENABLE; // 7) start PWM0
  PWM_ENABLE_R |= PWM_ENABLE_PWM0EN;    // enable PWM0
}
void PWM0_Duty(uint16_t duty){
  PWM_0_CMPA_R = duty - 1;         // 6) count value when output rises
}
```

PWM_4C123.zip
PWMDual_4C123.zip



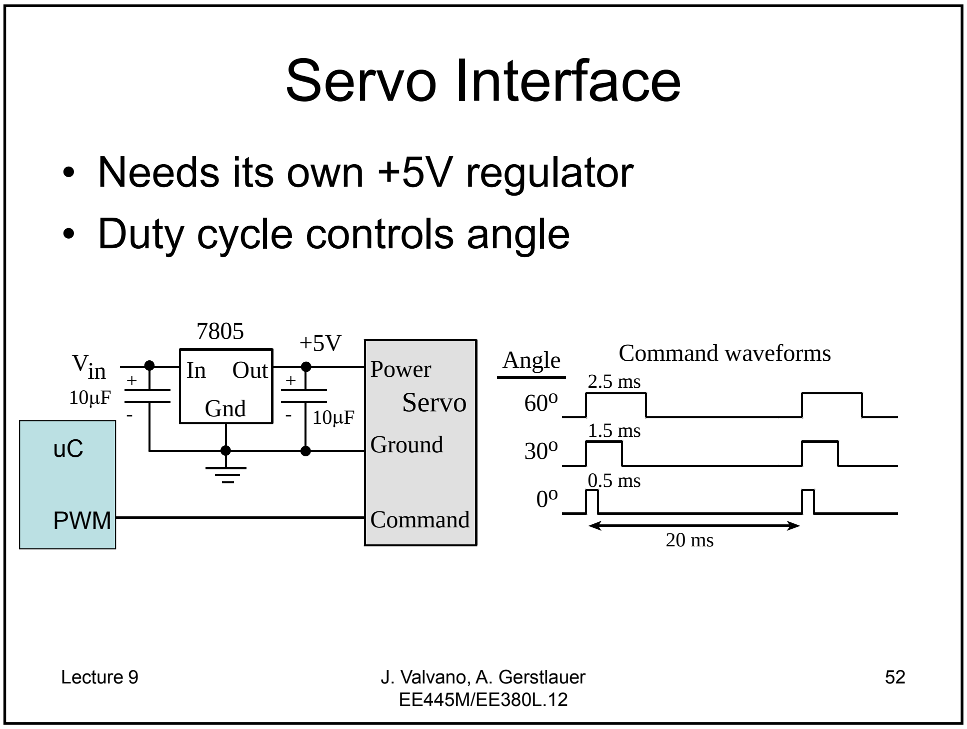
Motor Driver Software

```
// change duty cycle of right motor (PB7,PB6)
// Inputs: period was set in call to Right_InitDRV8848
//         duty is in 800 ns units
// duty is number of PWM clock cycles output is high  (2<=duty<=period-1)
// Must call Right_InitDRV8848
// - once initially before calling Right_DutyDRV8848
// - to change direction
// Motor board version 6 with DRV8848 dual H-bridge
void DRV8848_RightDuty(uint16_t duty){
  if(RightDirection == FORWARD){
    Right_Duty(RightPeriod-duty, 1);  // PB6 is negative logic duty, PB7 is 1
  }else{
    Right_DutyB(RightPeriod-duty, 1); // PB7 is negative logic duty, PB6 is 1
  }
}
```



Servo Motor

- Simple digital interface (built in controller)
- Duty cycle controls angle



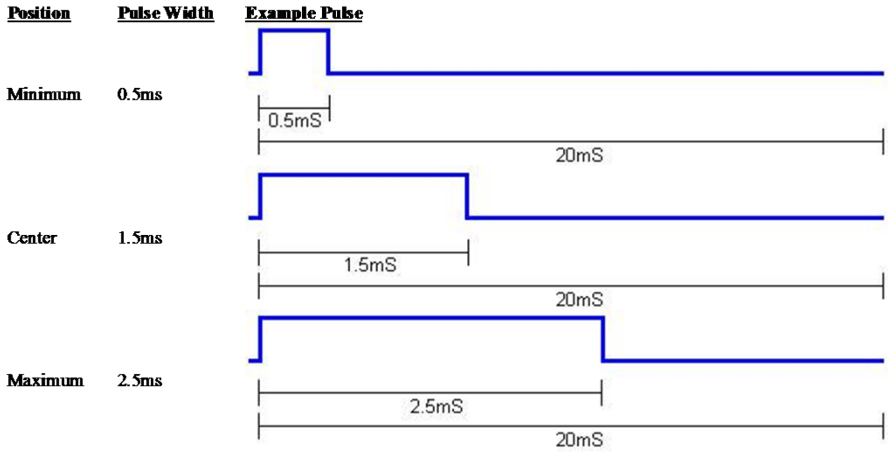
Servo Interface

- Needs its own +5V regulator
- Duty cycle controls angle

Servo Software

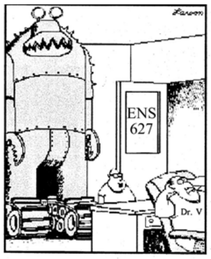
- Duty cycle controls angle

Position	Pulse Width
Minimum	0.5ms
Center	1.5ms
Maximum	2.5ms



Robot Interfacing (Lab 6)

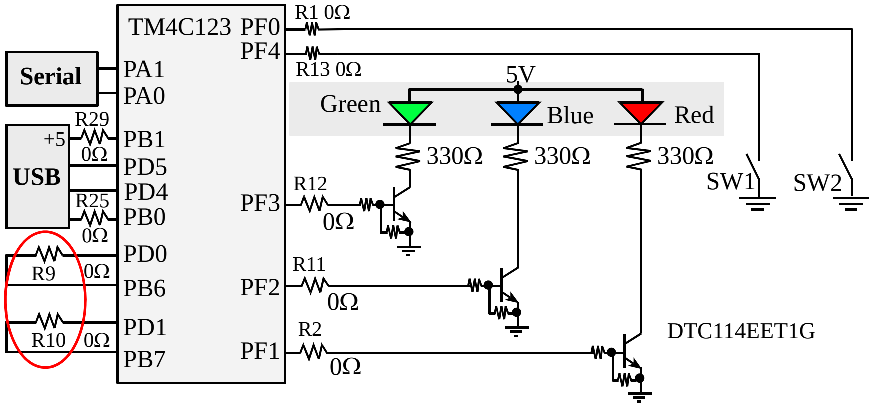
- Sensor board
  - 4x IR sensors
    - ADC input w/ analog filter
  - 4x Distance sensors
    - Timer input, 3- or 4-pin headers
  - 2x Switches (bumper)
    - Digital input, pull-down
- Motor board
  - 2x Discrete H bridge or integrated driver chip
    - PWM output, 2 motor connectors (0.156in header)
  - 2x Servo
    - PWM output



"My project's ready for grading, Dr. Big Nose... Hey! ... I'm talking to you, squid brain!"

Robot_Sensor_v3.pdf
Robot_Motor_v6.pdf

LaunchPad Board



Remove R9 and R10

Summary

- Be careful of the currents
- Sensors are noisy, need calibration
- Time lag makes it unstable
- Component testing
- Visualization and control