TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

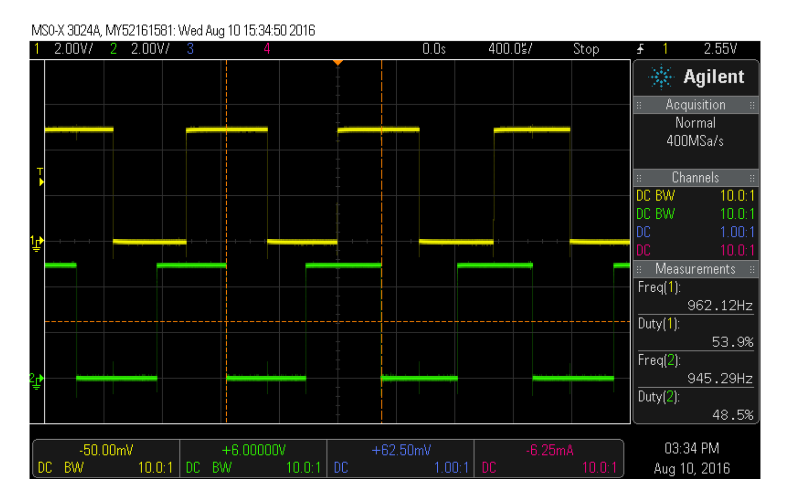
Module 16

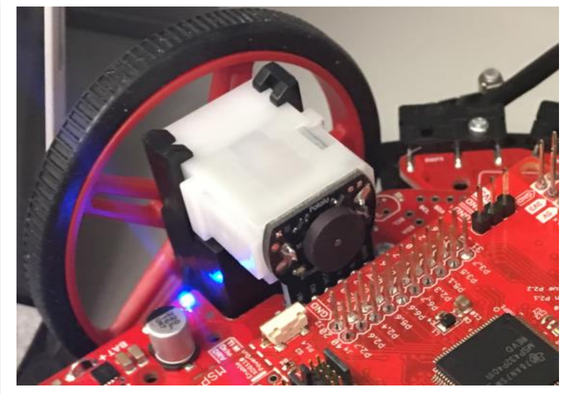
Lecture: Tachometer - Input Capture

Tachometer

You will learn in this module

- Timer A
  - Clock input, prescale
  - Input capture
- Period Measurement
  - Precision
  - Range
  - Resolution
- Motor Performance
  - Speed
  - Time constant

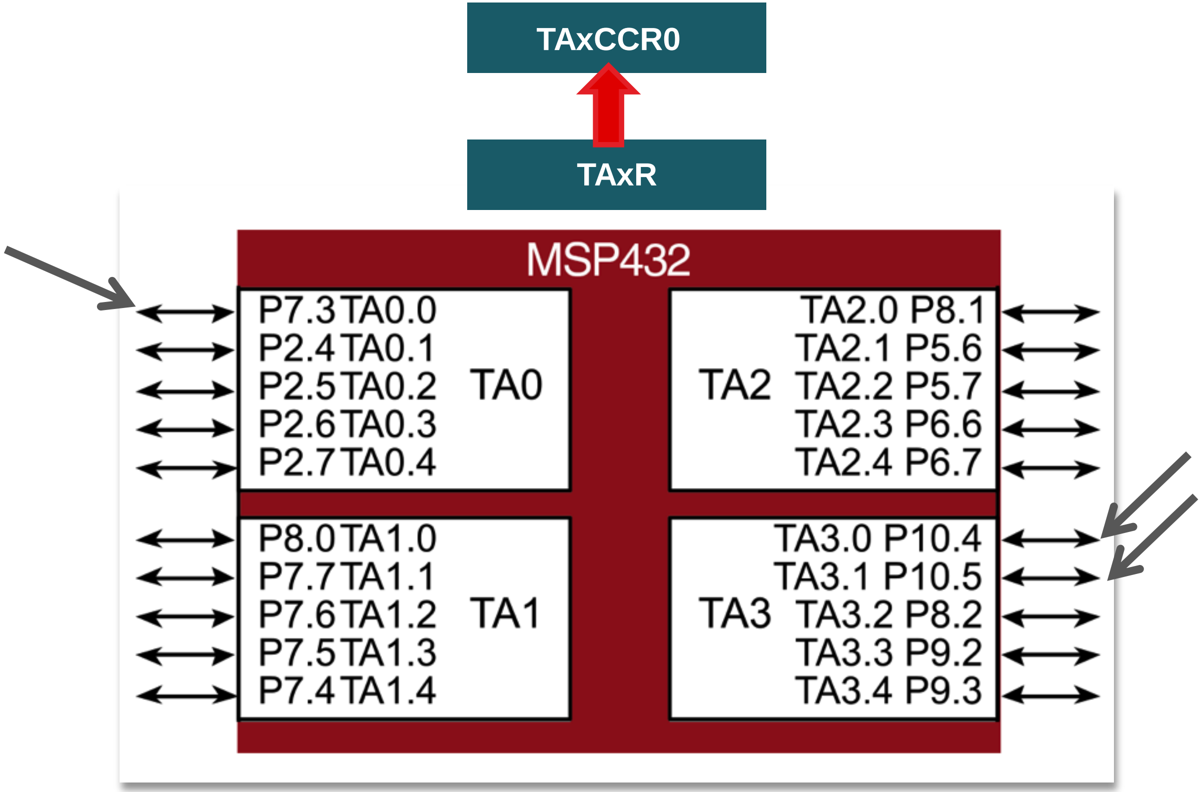




Timer_A for input and output

Components

- Pins
  - Input capture
  - Output compare
- Precision
  - 16-bits
- Resolution
  - Clock period
  - Prescale



SMCLK = 48MHz/4 = 12 MHz, 83.33ns

Timer_A registers

	15-10	9-8	7-6	5-4	3	2	1	0	Name
0.0000		TASSEL	ID	MC		TACLR	TAIE	TAIFG	TA0CTL

	15-14	13-12	11	10	9	8	7-5	4	3	2	1	0	
0.0002	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL0
0.0004	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL1
0.0006	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL2
0.0008	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL3
0.000A	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL4
0.000C	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL5
0.000E	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL6

	15–0	
0.0010	16-bit counter	TA0R
0.0012	16-bit Capture/Compare 0 Register	TA0CCR0
0.0014	16-bit Capture/Compare 1 Register	TA0CCR1
0.0016	16-bit Capture/Compare 2 Register	TA0CCR2
0.0018	16-bit Capture/Compare 3 Register	TA0CCR3
0.001A	16-bit Capture/Compare 4 Register	TA0CCR4
0.001C	16-bit Capture/Compare 5 Register	TA0CCR5
0.001E	16-bit Capture/Compare 6 Register	TA0CCR6

	15-3	2-0	
0.0020		TAIDEX	TA0EX0

	15-0	
0.002E	TAIV	TA0IV

Timer_A registers

	15-10	9-8	7-6	5-4	3	2	1	0	Name
0.0000		TASSEL	ID	MC		TACLR	TAIE	TAIFG	TA0CTL

	15-14	13-12	11	10	9	8	7-5	4	3	2	1	0	
0.0002	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL0
0.0004	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL1
0.0006	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL2
0.0008	CM	CCIS	SCS	SCCI		CAP	OUTMOD	CCIE	CCI	OUT	COV	CCIFG	TA0CCTL3
													0CCTL4
													0CCTL5
													0CCTL6

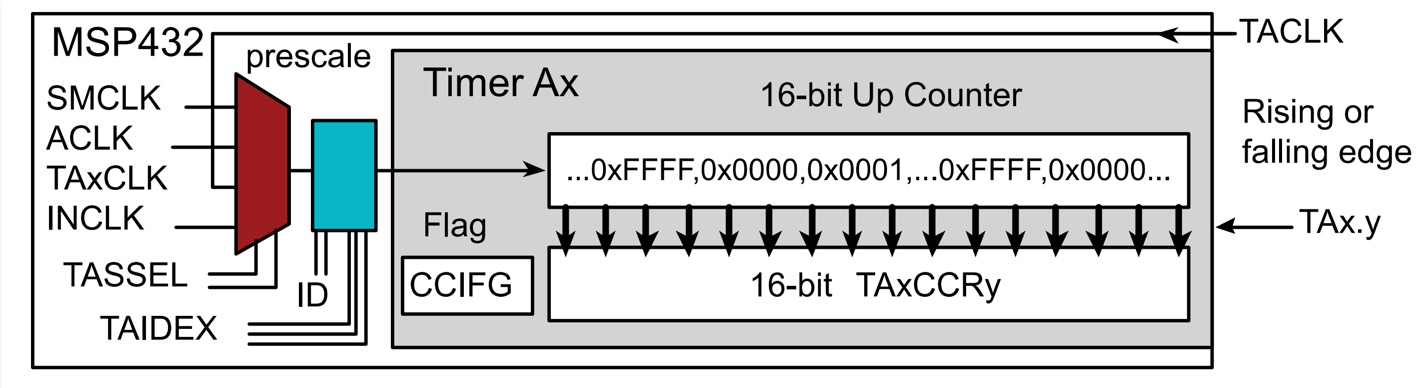
```
// 01,   00,   1,   0,   0,   1,   000,   1,   0,   0,   0,   0
TIMER_A0->CCTL[0] = 0x4910;
```

	15–0	
0.0010	16-bit counter	TA0R
0.0012	16-bit Capture/Compare 0 Register	TA0CCR0
0.0014	16-bit Capture/Compare 1 Register	TA0CCR1
0.0016	16-bit Capture/Compare 2 Register	TA0CCR2
0.0018	16-bit Capture/Compare 3 Register	TA0CCR3
0.001A	16-bit Capture/Compare 4 Register	TA0CCR4
0.001C	16-bit Capture/Compare 5 Register	TA0CCR5
0.001E	16-bit Capture/Compare 6 Register	TA0CCR6

	15-3	2-0	
0.0020		TAIDEX	TA0EX0

	15-0	
0.002E	TAIV	TA0IV

Clock and prescale



TASSEL	Selected Clock
00	TAxCLK
01	ACLK
10	SMCLK
11	INCLK

ID	Prescale
00	/1
01	/2
10	/4
11	/8

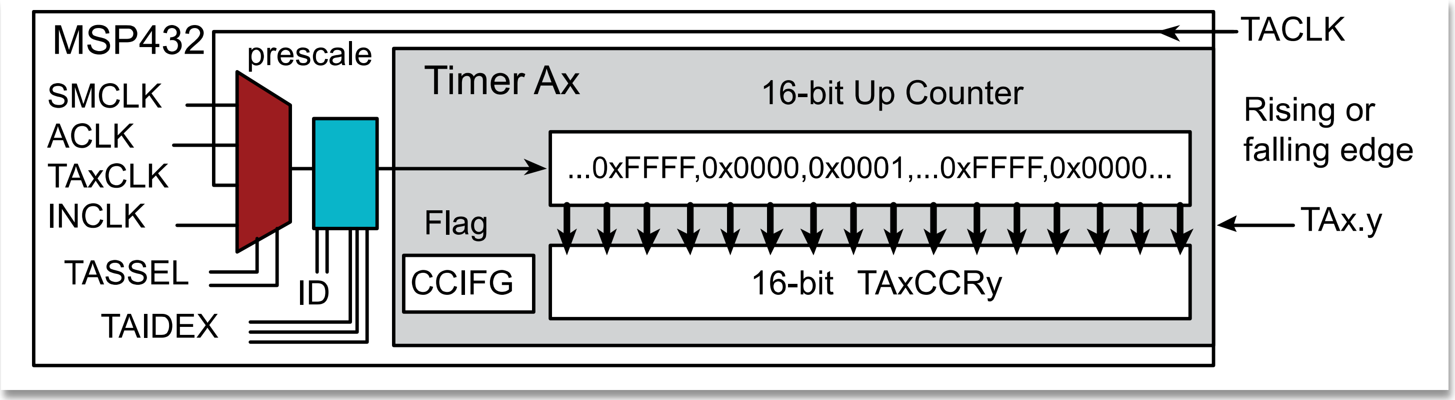
Resolution = $T * 2^{ID} *$ (**TAIDEX**+1)
Range = Precision * Resolution

Timer_A Modes

CM	Capture mode
00	No capture
01	Capture on rising edge
10	Capture on falling edge
11	Capture on both rising and falling edges

CM	bits15-14=01	capture on rising edge
CCIS	bits13-12=00	capture input on CCIxA
SCS	bit 11=1	synchronous capture source
CAP	bit 8=1	capture mode
CCIE	bit 4=1	arm for interrupt
CCIFG	bit 0=0	trigger, set by hardware, cleared by software

Clock and prescale



0) Halt the timer (MC=00),

1) Set the timer clock and prescale,

2) Set submodule 0 to capture rising, arm interrupt

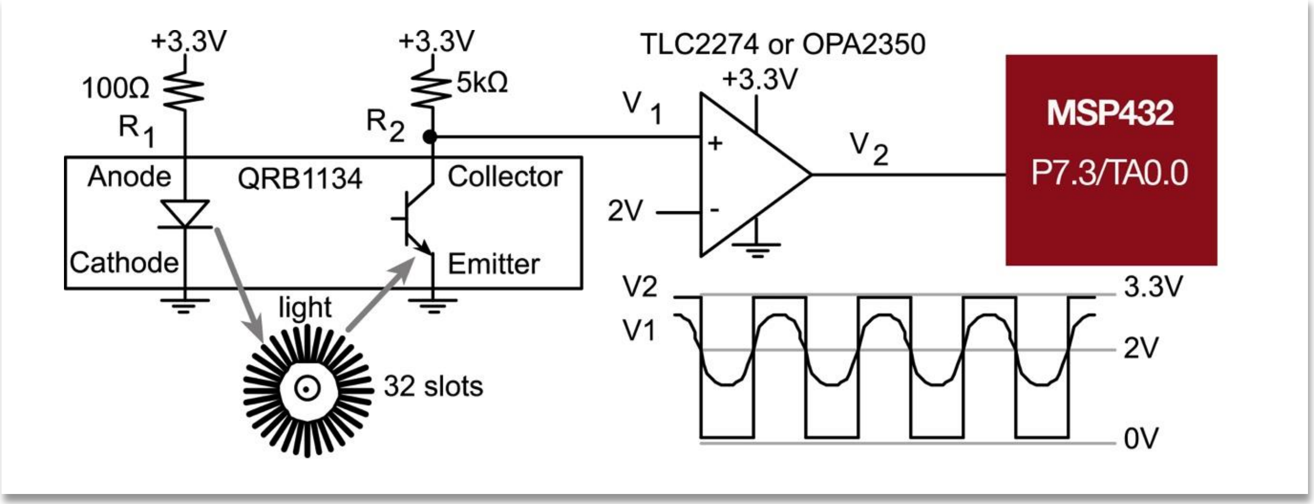
3) Set the priority in the correct NVIC Priority register

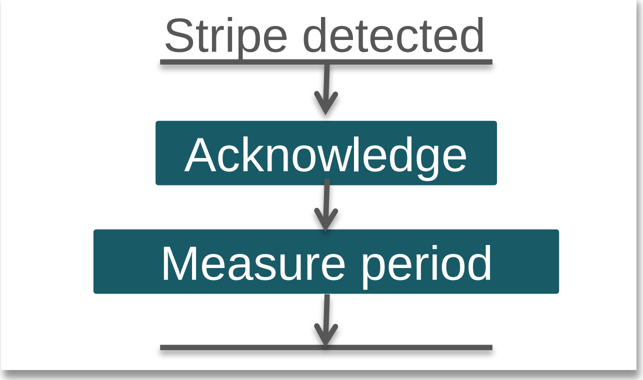
4) Enable the interrupt in the NVIC Interrupt Enable register

5) Reset the timer and start it in up mode

6) Enable interrupts (in the main program after all devices initialized)

Period Measurement Example





Period Measurement Example

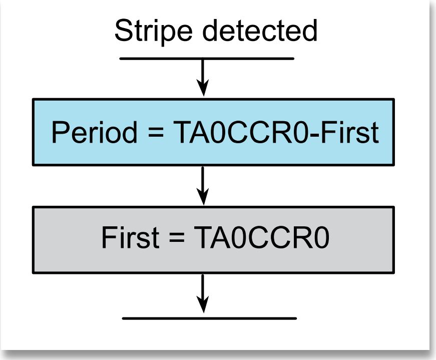
```
void TimerA0Capture_Init(void){
  P7->SEL0 |= 0x08;
  P7->SEL1 &= ~0x08;    // configure P7.3 as TA0CCP0
  P7->DIR &= ~0x08;       // make P7.3 in
  TIMER_A0->CTL &= ~0x0030;      // halt Timer A0
  // bits9-8=10,        clock source to SMCLK
  // bits7-6=00,        input clock divider /1
  // bits5-4=00,        stop mode
  TIMER_A0->CTL = 0x0200;
  // bits15-14=01,      capture on rising edge
  // bits13-12=00,      capture/compare input on CCI0A
  // bit11=1,           synchronous capture source
  // bit8=1,            capture mode
  // bit4=1,            enable capture/compare interrupt
  // bit0=0,            clear capture/compare interrupt pending
  TIMER_A0->CCTL[0] = 0x4910;
  TIMER_A0->EX0 &= ~0x0007; // configure for input clock divider /1
  NVIC->IP[2] = (NVIC->IP[2]&0xFFFFFF00)|0x00000040; // priority 2
  NVIC->ISER[0] = 0x00000100; // enable interrupt 8 in NVIC
  TIMER_A0->CTL |= 0x0024;   // reset and start Timer A0 in continuous up mode
}
```

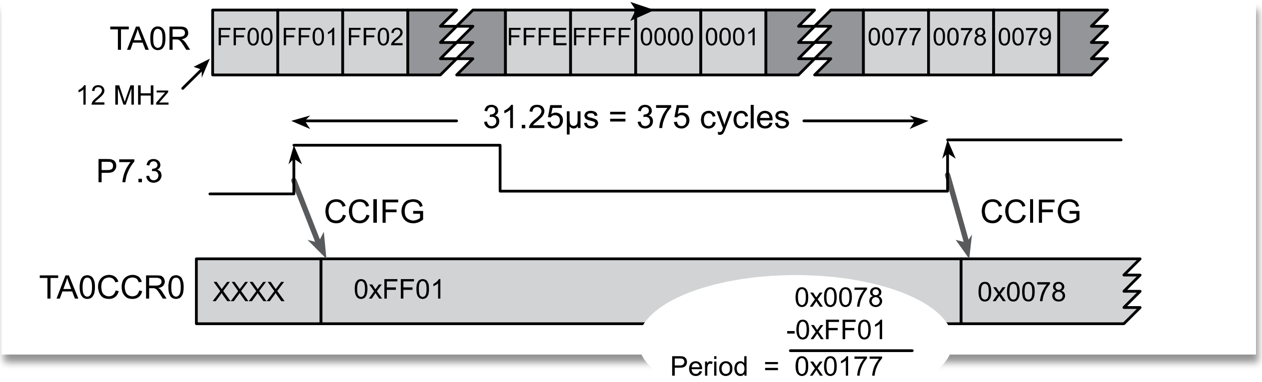
Interrupts enabled in the main program after all devices initialized

Period Measurement

Interrupt on rising edge

```
uint16_t First,Period;
void TA0_0_IRQHandler(void){
  TIMER_A0->CCTL[0] &= ~0x0001;
  // acknowledge capture/compare interrupt 0
  Period = TIMER_A0->CCR[0]-First;
  First = TIMER_A0->CCR[0];
}
```



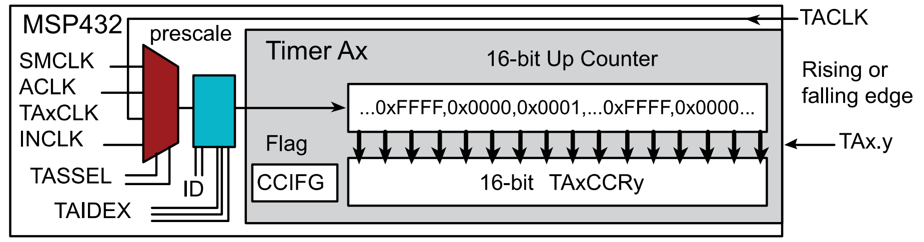


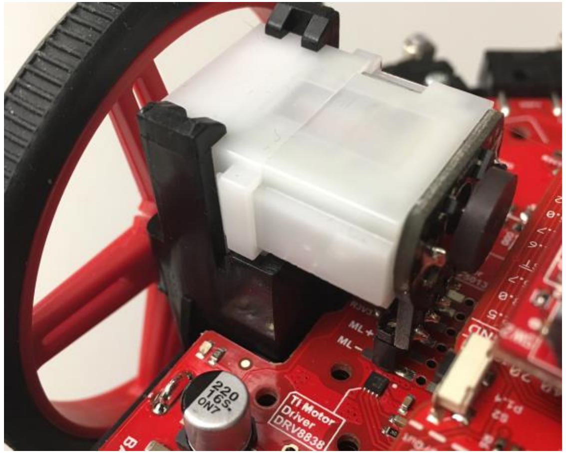
Precision	16 bits
Resolution	83.33ns
Range	10us to 5.4ms

Summary

Measuring speed with input capture

- Timer_A
  - Clock input,
  - Prescale
  - Input capture
- Period Measurement
  - Precision
  - Range
  - Resolution
- Motor Performance
  - Speed
  - Time constant





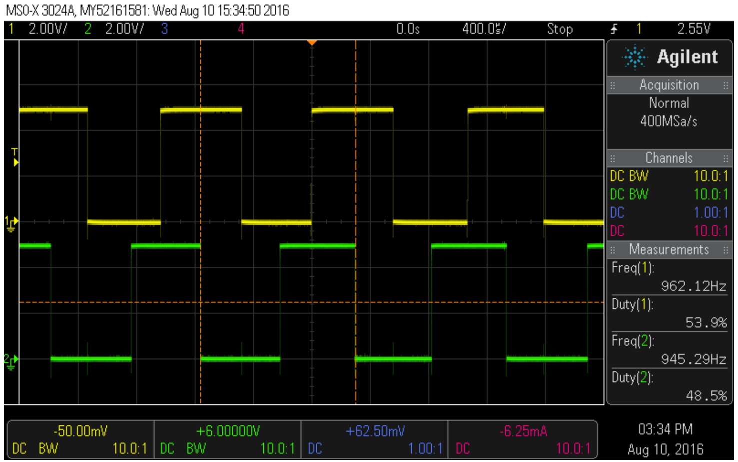
Module 16

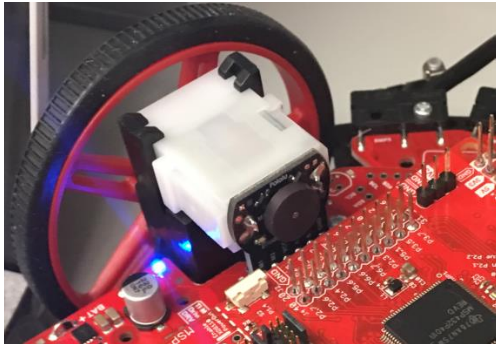
Lecture: Tachometer - Interface

Tachometer

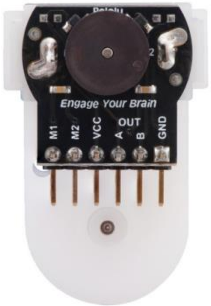
You will learn in this module

- Encoder
  - Motor speed
  - Motor direction
- Motor Performance
  - Speed
  - Time constant

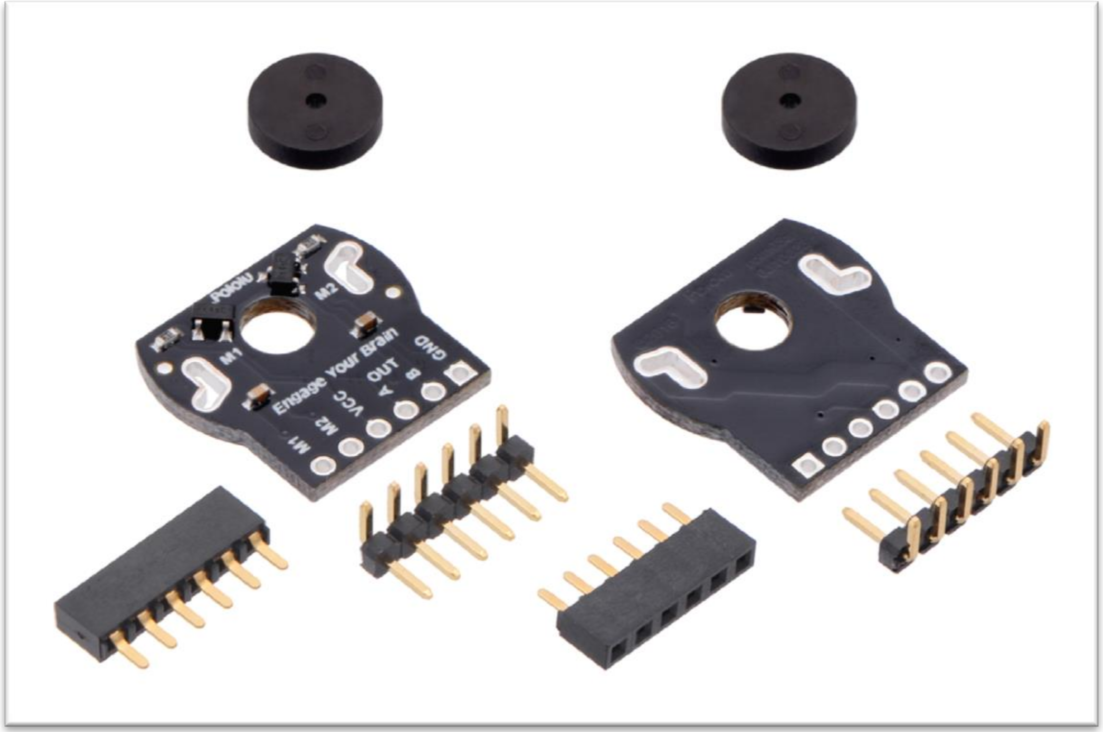




Encoders



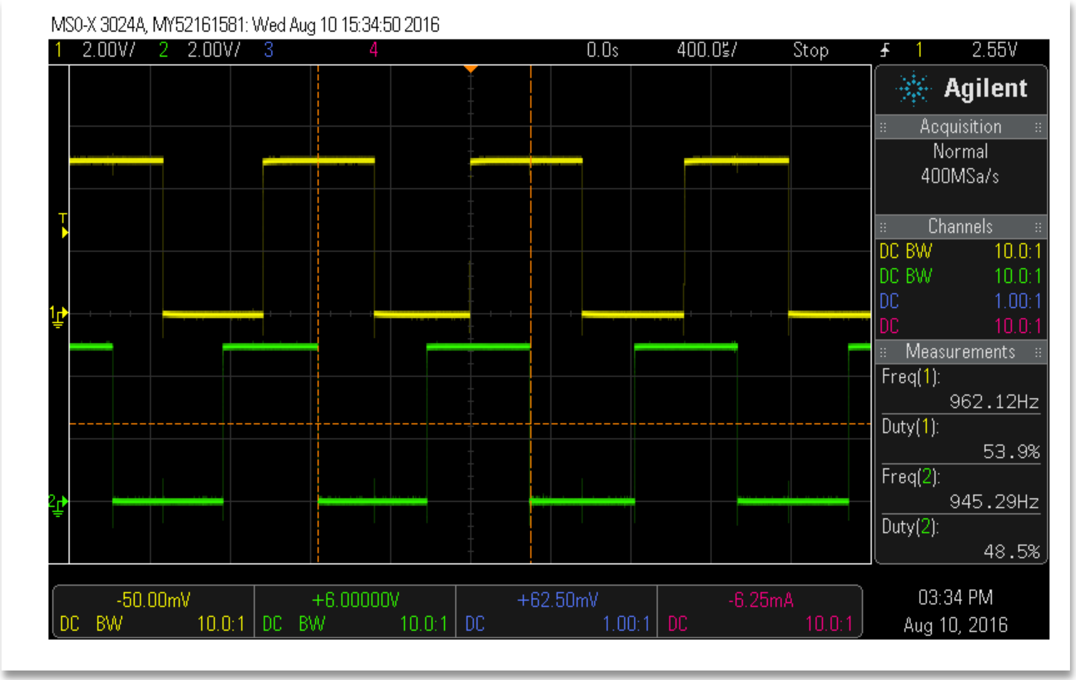
Gearmotor for the Romi/TI-RSLK has an encoder already installed



360 pulses/rotation

Speed (rpm) =
(1 rotation/360pulses)
*(1,000,000,000ns/sec)
*(60sec/min)
/(Period*83.33ns/pulse)

Speed = 2,000,000/Period

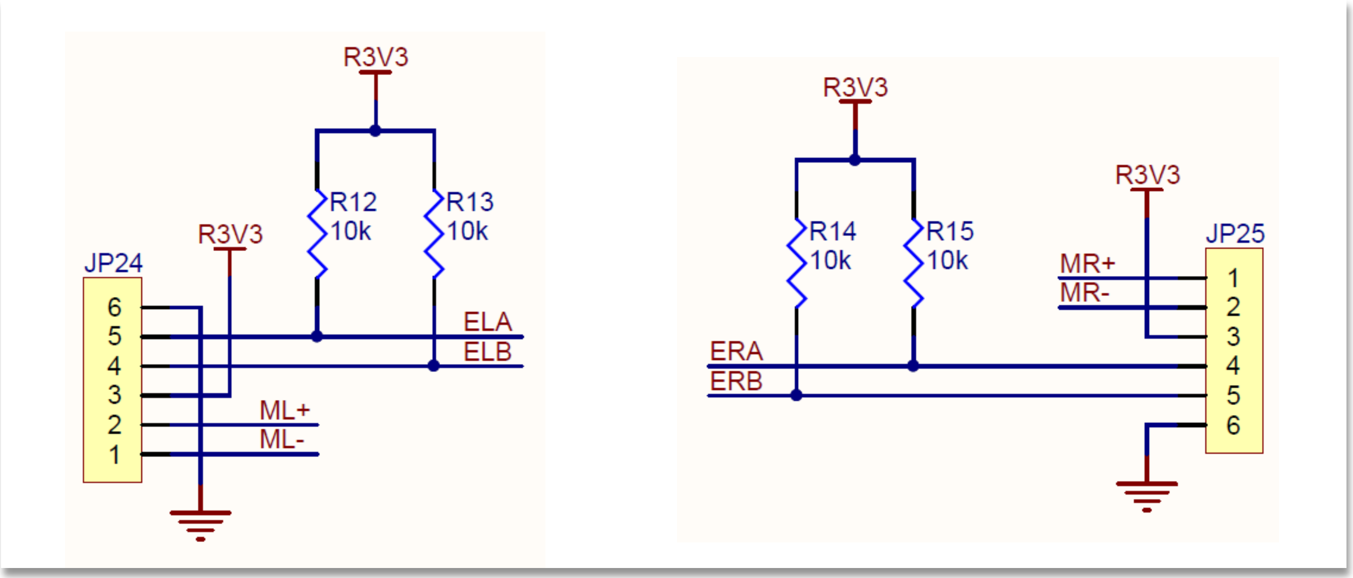


TI-RSLK Chassis board with of two encoders

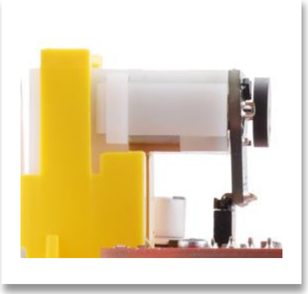
Lab 5:

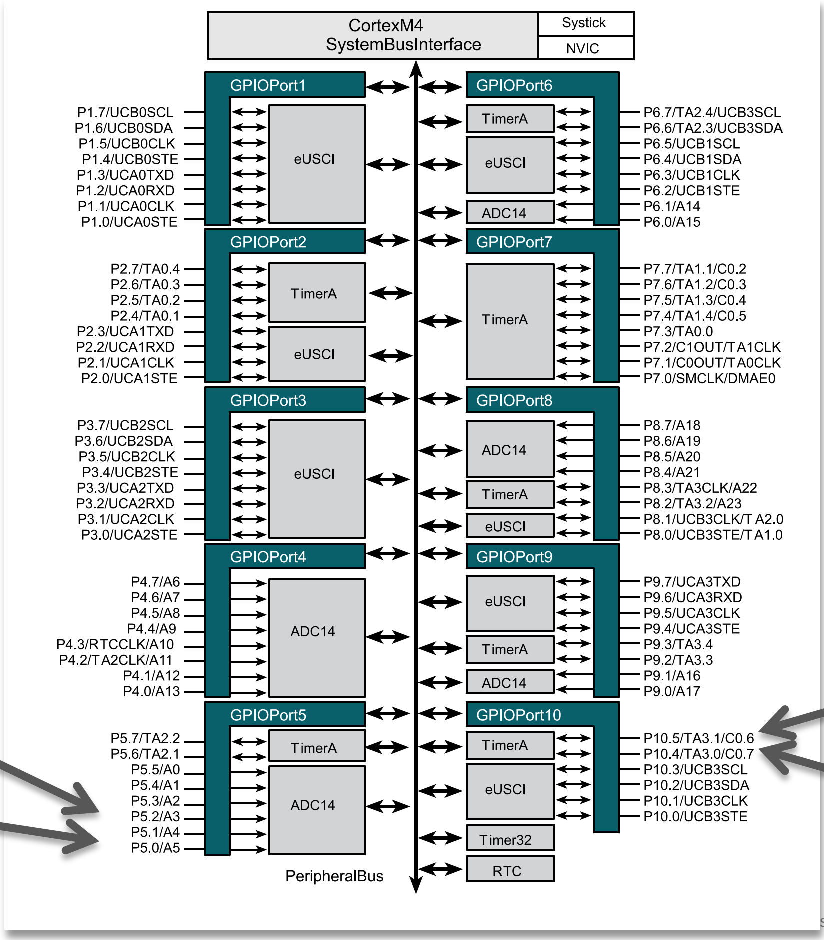
- Plug gear motor assembly with encoder as you build the TI-RSLK Chassis board and the robot

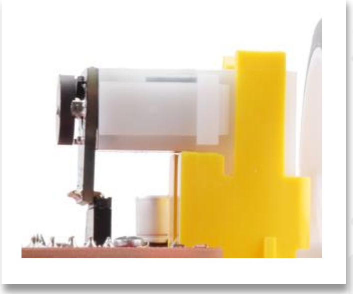
ELA to P10.5 Input Capture
ELB to P5.2 GPIO
ERA to P10.4 Input Capture
ERB to P5.0 GPIO



MSP432 Planning







P5.2 GPIO

P5.0 GPIO

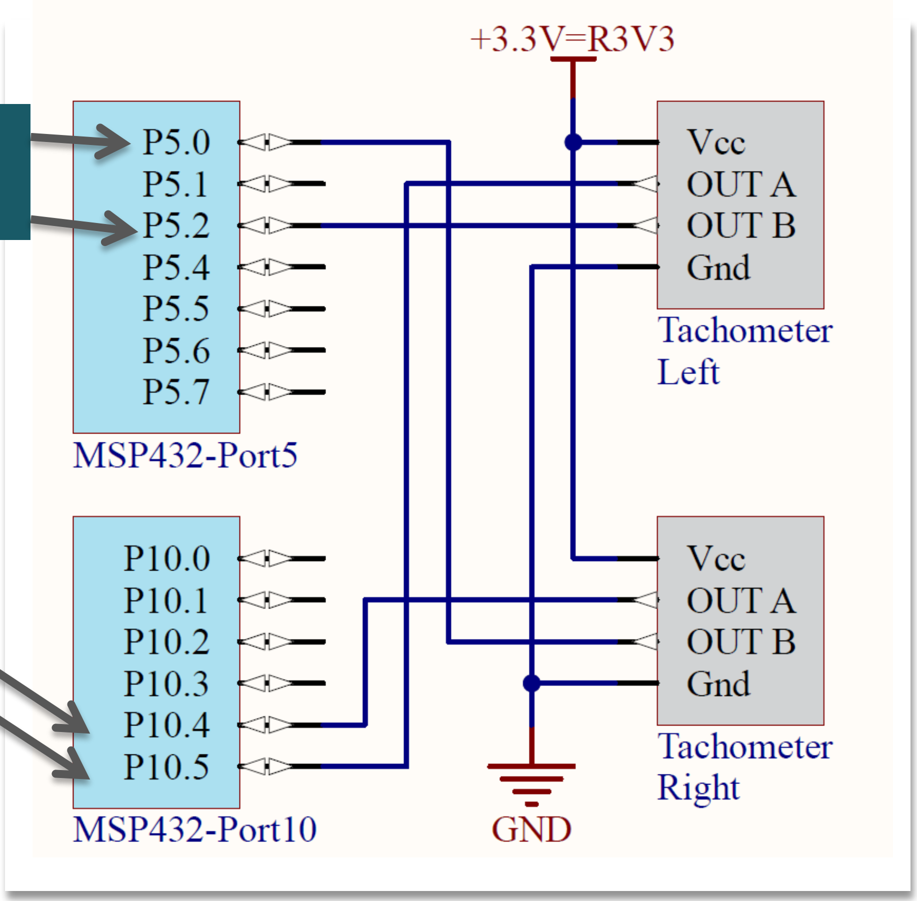
P10.5 Capture

P10.4 Capture

TI-RSLK Chassis board – Interface circuit MSP432-Encoder

Direction on P5.0, P5.2
- 1 = forward
- 0 = backward

Input capture on P10.4,10.5



Interrupt Vectors, numbers, names, and priority

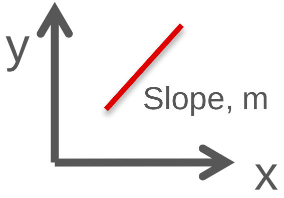
Vector	Number	IRQ	ISR name	NVIC priority	Priority
0x0000002C	11	-5	SVC_Handler	SCB_SHPR2	31 – 29
0x00000038	14	-2	PendSV_Handler	SCB_SHPR3	23 – 21
0x0000003C	15	-1	SysTick_Handler	SCB_SHPR3	31 – 29
0x00000060	24	8	TA0_0_IRQHandler	NVIC_IPR2	7 – 5
0x00000064	25	9	TA0_N_IRQHandler	NVIC_IPR2	15 – 13
0x00000068	26	10	TA1_0_IRQHandler	NVIC_IPR2	23 – 21
0x0000006C	27	11	TA1_N_IRQHandler	NVIC_IPR2	31 – 29
0x00000070	28	12	TA2_0_IRQHandler	NVIC_IPR3	7 – 5
0x00000074	29	13	TA2_N_IRQHandler	NVIC_IPR3	15 – 13
0x00000078	30	14	TA3_0_IRQHandler	NVIC_IPR3	23 – 21
0x0000007C	31	15	TA3_N_IRQHandler	NVIC_IPR3	31 – 29
0x00000080	32	16	EUSCIA0_IRQHandler	NVIC_IPR4	7 – 5
0x00000084	33	17	EUSCIA1_IRQHandler	NVIC_IPR4	15 – 13
0x00000088	34	18	EUSCIA2_IRQHandler	NVIC_IPR4	23 – 21
0x0000008C	35	19	EUSCIA3_IRQHandler	NVIC_IPR4	31 – 29
0x00000090	36	20	EUSCIB0_IRQHandler	NVIC_IPR5	7 – 5
0x				R5	15 – 13
0x				R5	23 – 21
0x				R5	31 – 29
0x				R8	31 – 29
0x				R9	7 – 5
0x				R9	15 – 13
0x				R9	23 – 21
0x				R9	31 – 29
0x				R10	7 – 5

```
void TA3_0_IRQHandler(void){
  TIMER_A3->CCTL[0] &= ~0x0001; // ack
// body
}
void TA3_N_IRQHandler(void){
  TIMER_A3->CCTL[1] &= ~0x0001; // ack
// body
}
```

Motor Model

Duty cycle, $X(t)$ → Motor → Speed $Y(t)$

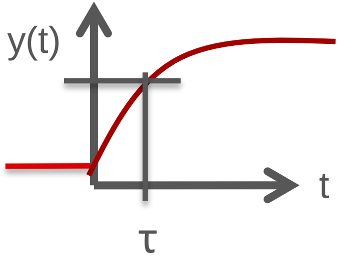
Linear model
- Gain, m
- Time constant, τ



Duty cycle, $X(s)$ → Motor → Speed $Y(s)$

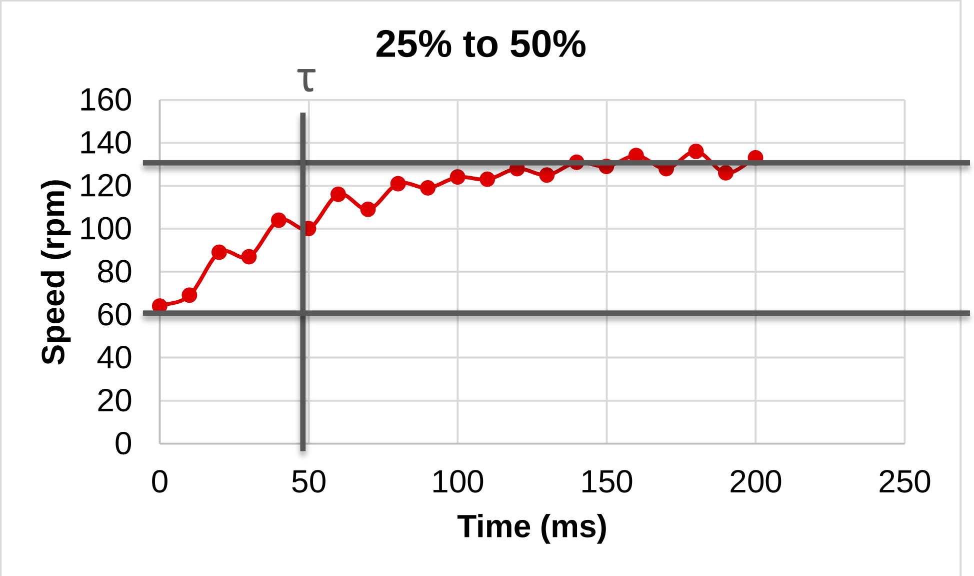
$$H(s) = \frac{Y(s)}{X(s)} = \frac{m}{1+s\tau}$$

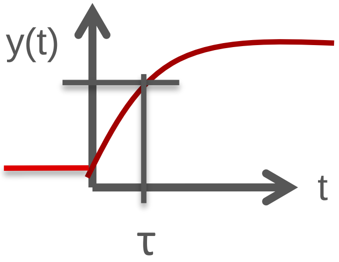
Motors are not linear
Friction affects everything



$$y(t) = S_0 + \Delta S\, e^{-t/\tau}$$

Motor Time Constant



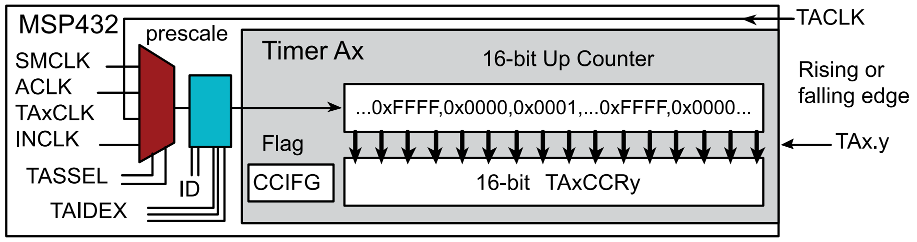


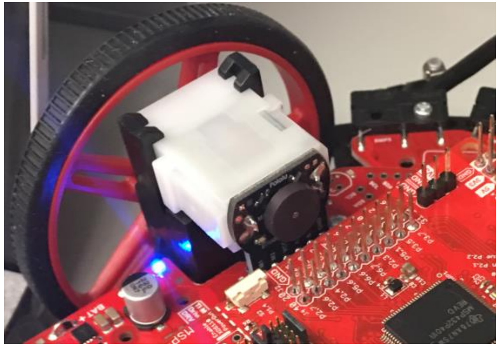
$y(t) = S_0 + \Delta S\ e^{-t/\tau}$

Summary

Measuring speed with input capture

- Timer_A
  - Clock input,
  - Prescale
  - Input capture
- Period Measurement
  - Precision
  - Range
  - Resolution
- Motor Performance
  - Speed
  - Time constant





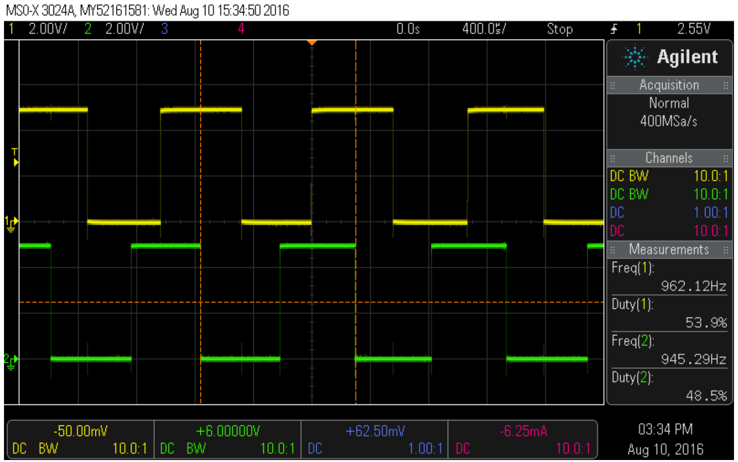
Module 16

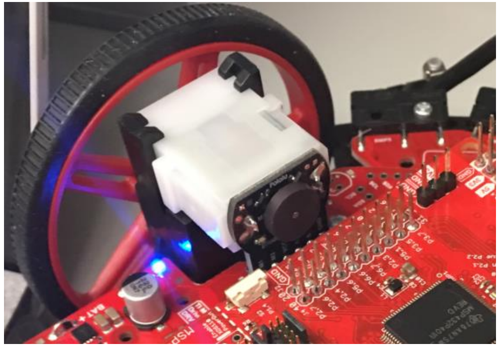
Lecture: Tachometer - Odometry

Odometry

You will learn in this module

- How the wheels affect position and angle
  - Robot wheelbase
  - Wheel circumference
  - Wheel rotations
- Geometry used in odometry
  - Wheel counts as input
  - Old position and angle as input
  - New position and angle as output
- Limitations of odometry
  - Calibration
  - Accumulation of error





Fixed point

What

Value = integer*constant

Why

Integer math is faster than floating point

32-bit floating point only has 23 bits of precision

When

Need non-integer values

Range of values is known and small

How

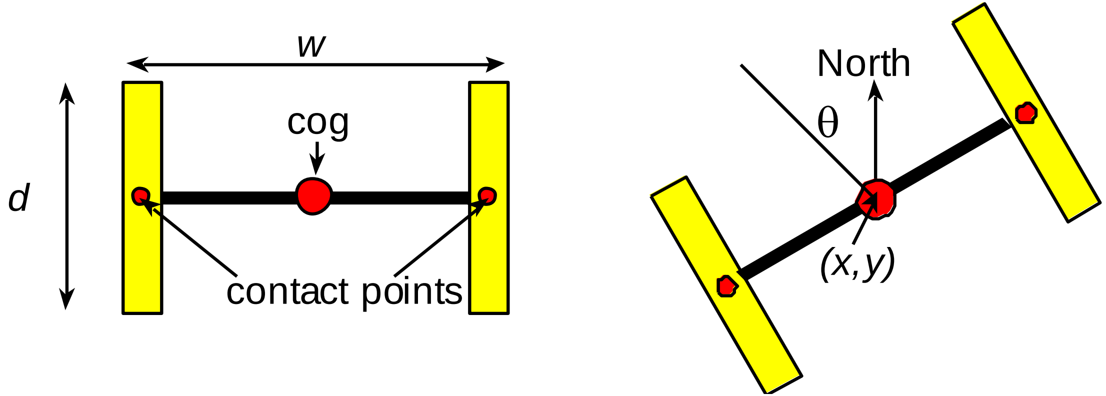
Choose a constant, e.g., 0.0001cm

Choose an integer type, e.g., int32_t

e.g., range will be ±200000 cm

e.g., resolution will be 0.0001cm

Physical dimensions of the robot



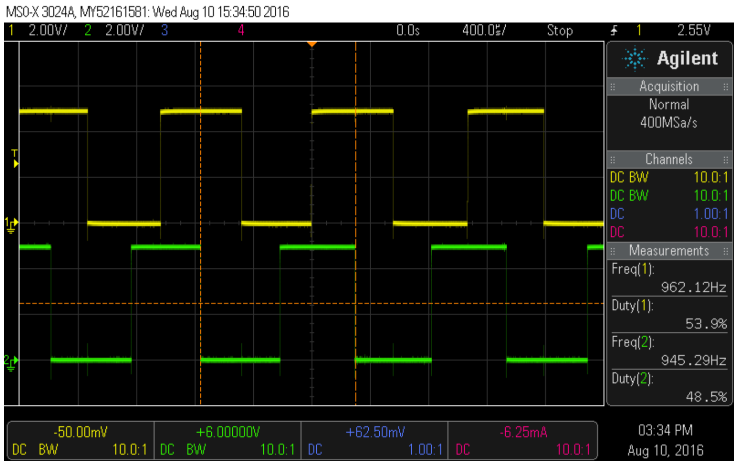
Number of slots/rotation, $n = 360$
Wheel diameter, $d = 70000$ in units of 0.0001cm
Wheelbase (distance between wheels), $w = 140000$ in units of 0.0001cm
Wheel circumference, $c = \pi d = 219911$ in units of 0.0001cm
Robot position (x, y) in units of 0.0001cm
Robot heading *theta* in units of $2\pi/16384$ radians
We define the center of gravity (cog) as a point equidistant from the pivot points

When to run odometry?

Tachometers interrupt 360 times/rotation
ISRs maintain *Lcount*, *Rcount*

Odometry is math that

- Takes *Lcount*, *Rcount* as inputs
- Updates (*x*, *y*, *theta*)



You should not run odometry on each count.

Errors if counts are too big
Errors if counts are too small

Option 1: Periodic

- Run the odometry calculations at fixed rate.
- Choose a rate so the counts are about 10 to 40.
- If either count goes above 40, you could increase rate.
- If both counts drop below 10, you could decrease rate.

Option 2: As needed

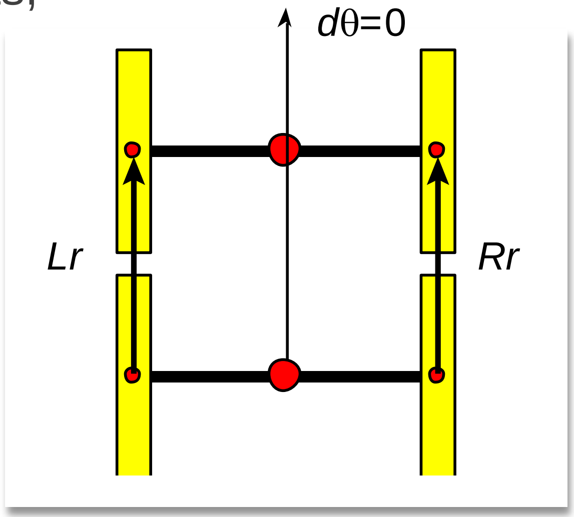
- Run the odometry when either count gets to a fixed limit, e.g., 20.

Odometry

- Tachometer used to count rotations
- Predict position from wheel rotations
- Assume the wheels do not slip

Examples

If both wheels move the same number of counts, it goes in a straight line.



Arc distance each wheel moves

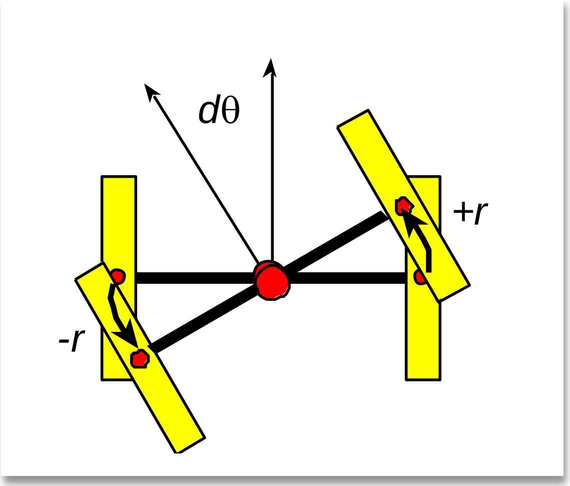
- *Lr = Lcount *c/n*
- *Rr = Rcount *c/n*

Odometry

- Tachometer used to count rotations
- Predict position from wheel rotations
- Assume the wheels do not slip

Examples

If one wheel goes forward the same count as the other goes back, it pivots about center of the robot.



Arc distance each wheel moves

- *Lr = Lcount *c/n*
- *Rr = Rcount *c/n*

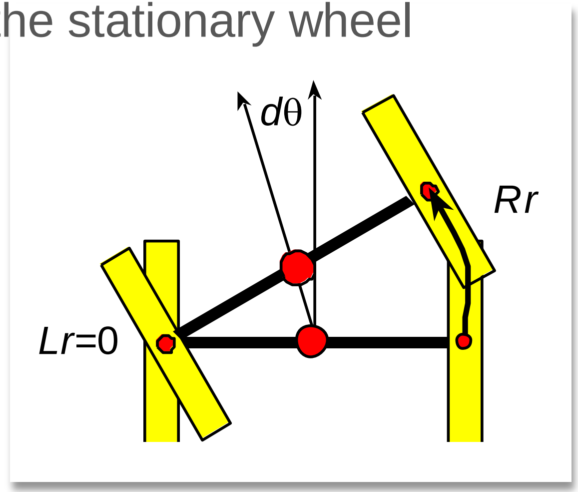
Odometry

- Tachometer used to count rotations
- Predict position from wheel rotations
- Assume the wheels do not slip

Examples

If one wheel moves but the other does not,

it pivots about single contact point of the stationary wheel



Arc distance each wheel moves

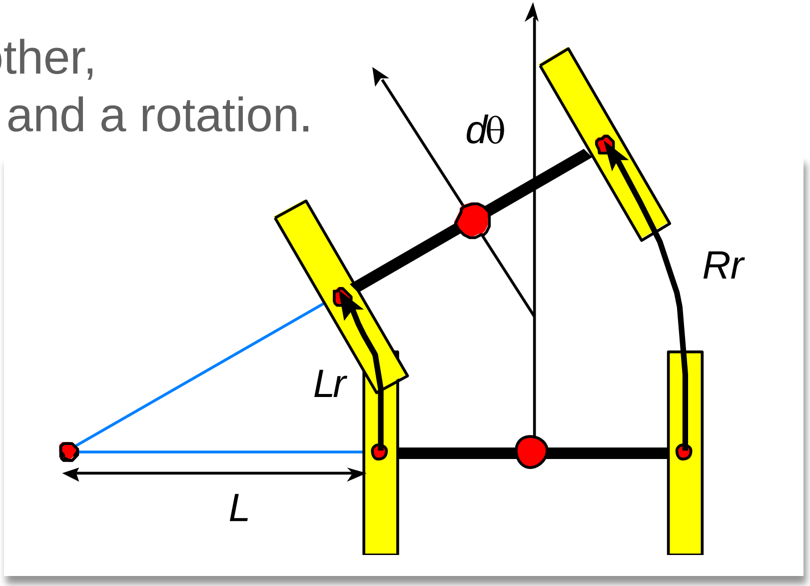
- *Lr = Lcount *c/n*
- *Rr = Rcount *c/n*

Odometry

- Tachometer used to count rotations
- Predict position from wheel rotations
- Assume the wheels do not slip

Examples

If one wheel moves more than the other,
there will be both a motion and a rotation.



Arc distance each wheel moves

- *Lr = Lcount *c/n*
- *Rr = Rcount *c/n*

Odometry: find the pivot point

Inputs: *Lcount*, *Rcount*

Arc distance each wheel moves

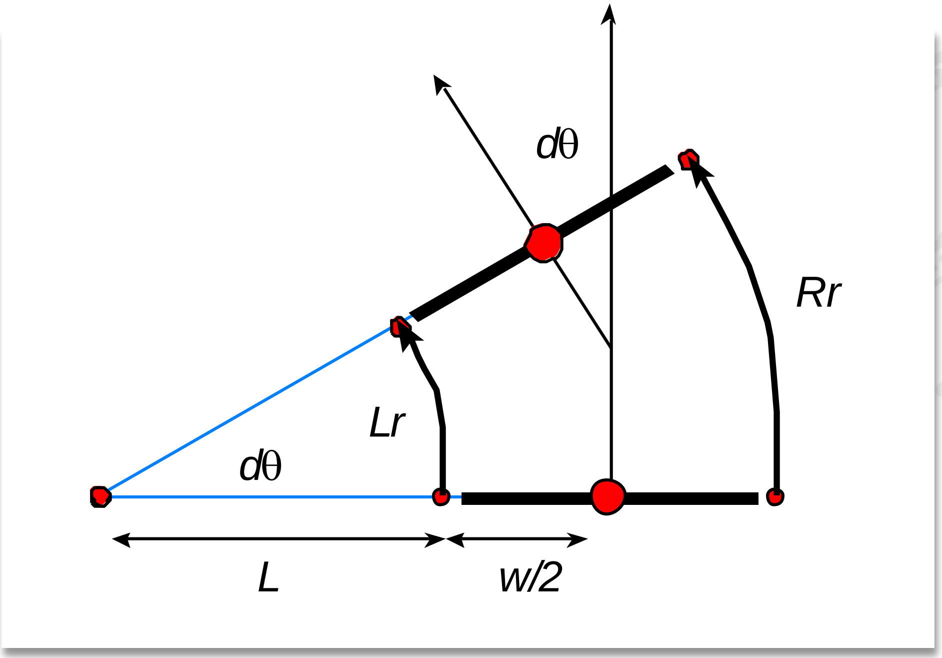
- *Lr = Lcount *c/n*
- *Rr = Rcount *c/n*

Assume

- *Rr* and *Lr* are both positive
- *Rr > Lr*

Using similar triangles

- *L/Lr = (L+w)/Rr*
- *L/Lr - L/Rr = w/Rr*
- *L Rr - L Lr = w Lr*
- *L = w Lr/(Rr - Lr)*



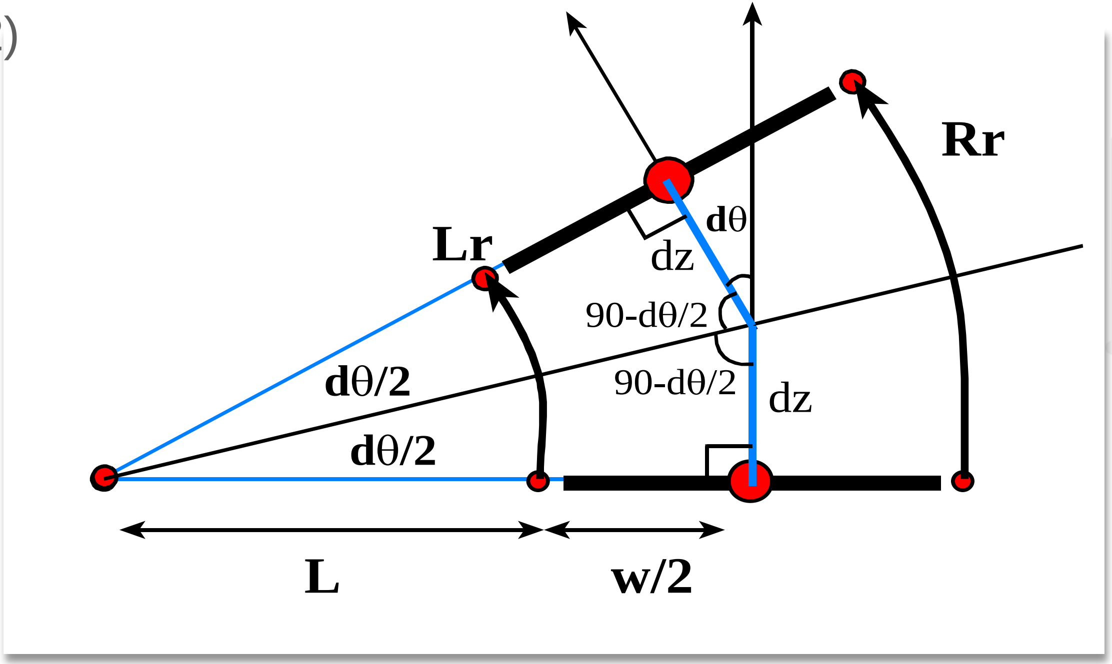
Odometry: find the change in distance

The exact calculation for position change is

$$dz = (L+w/2)*\tan(d\theta/2)$$

If dθ is small, $\tan(d\theta/2) \approx d\theta/2$

$$dz = (L+w/2) *(d\theta/2)$$



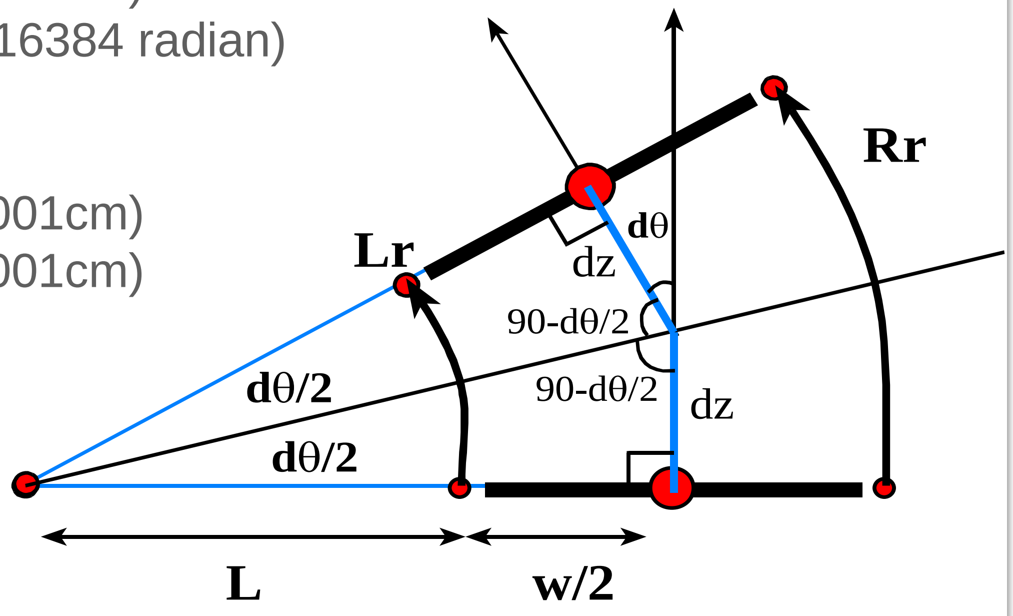
Odometry: break move into two parts

First part of the move

- $x = x + dz*\cos(\theta)$ (0.0001cm)
- $y = y + dz*\sin(\theta)$ (0.0001cm)
- $\theta = \theta + d\theta$ (2π/16384 radian)

Second part of the move

- $x = x + dz*\cos(\theta)$ (0.0001cm)
- $y = y + dz*\sin(\theta)$ (0.0001cm)



Odometry: errors accumulate

Error in tachometer

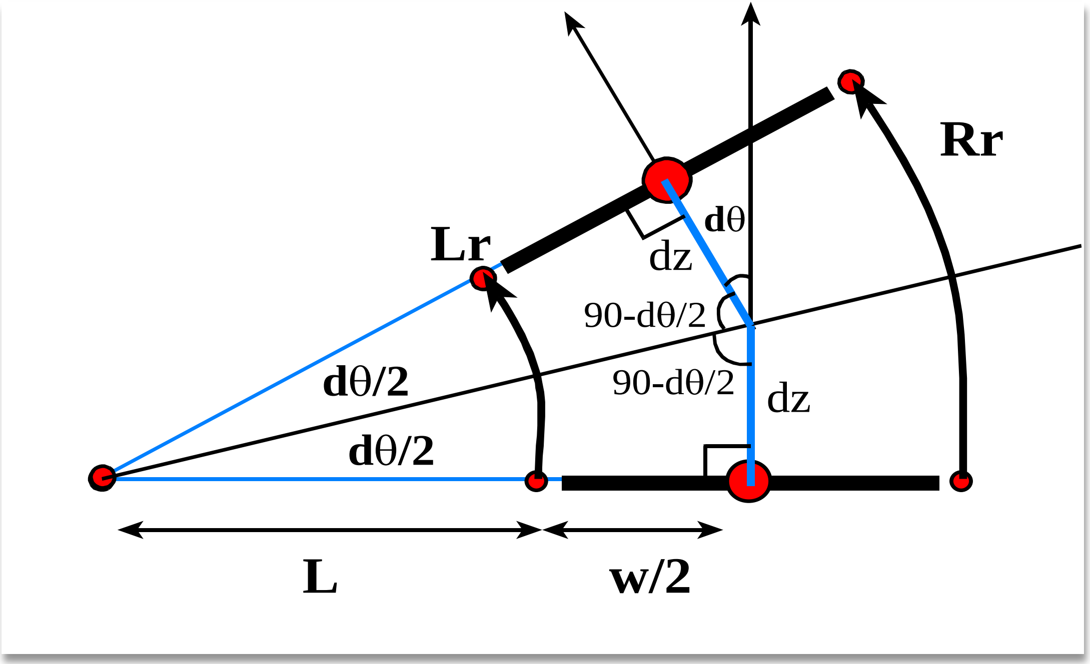
- Extra counts
- Missing counts

Error in constants

- *w* wheelbase
- *c* wheel circumference

Errors in calculations

- tan($d\theta/2$) ≈ $d\theta/2$
- sin(θ)
- cos(θ)

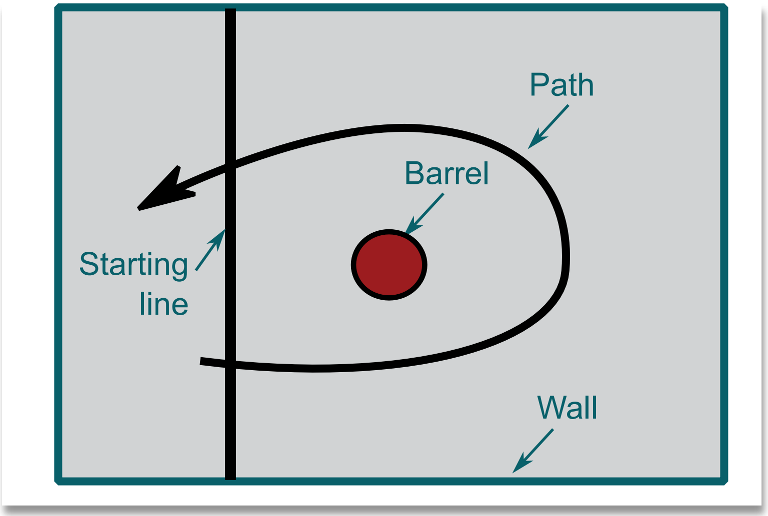


Summary

Odometry

- Tachometer input
  - Robot wheelbase
  - Wheel circumference
- Where am I?
  - Position
  - Angle
- Limitations
  - Calibration
  - Accumulation of errors

Barrel Racing



ti.com/rslk

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265