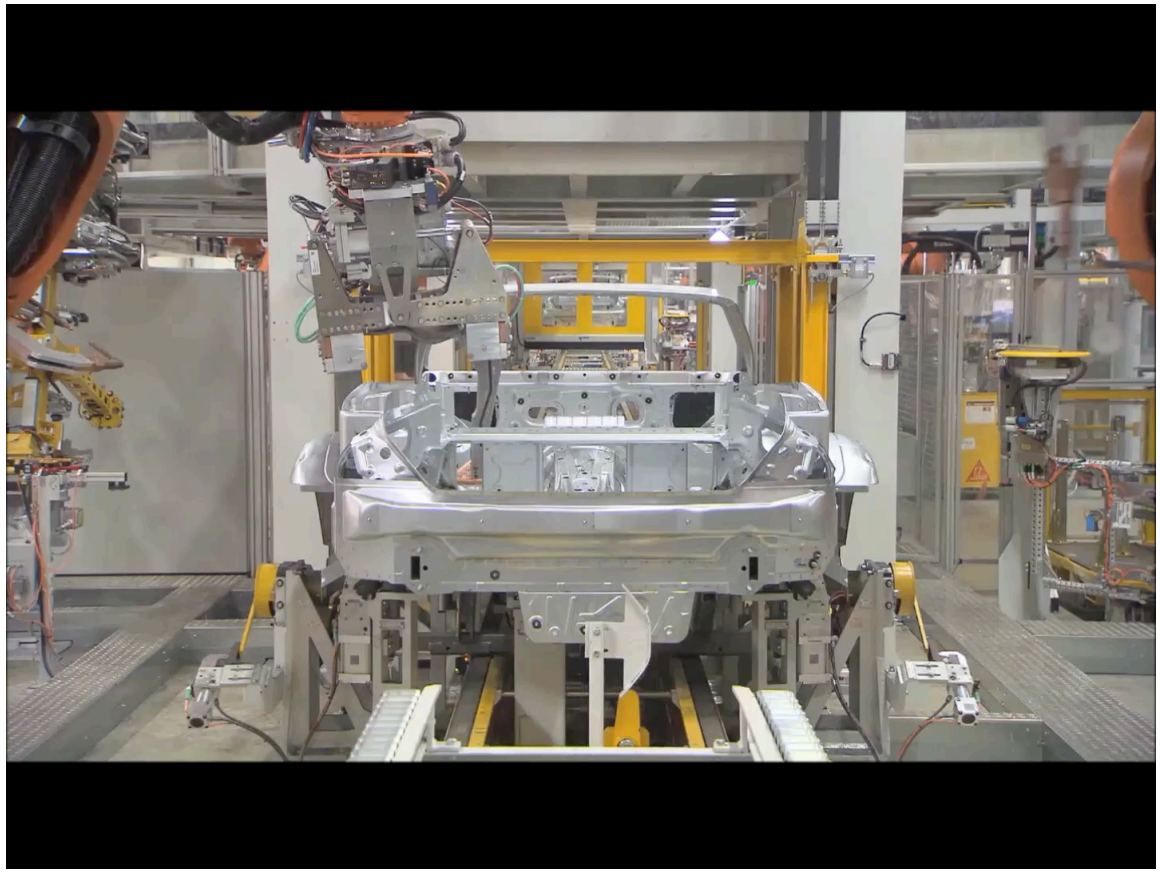
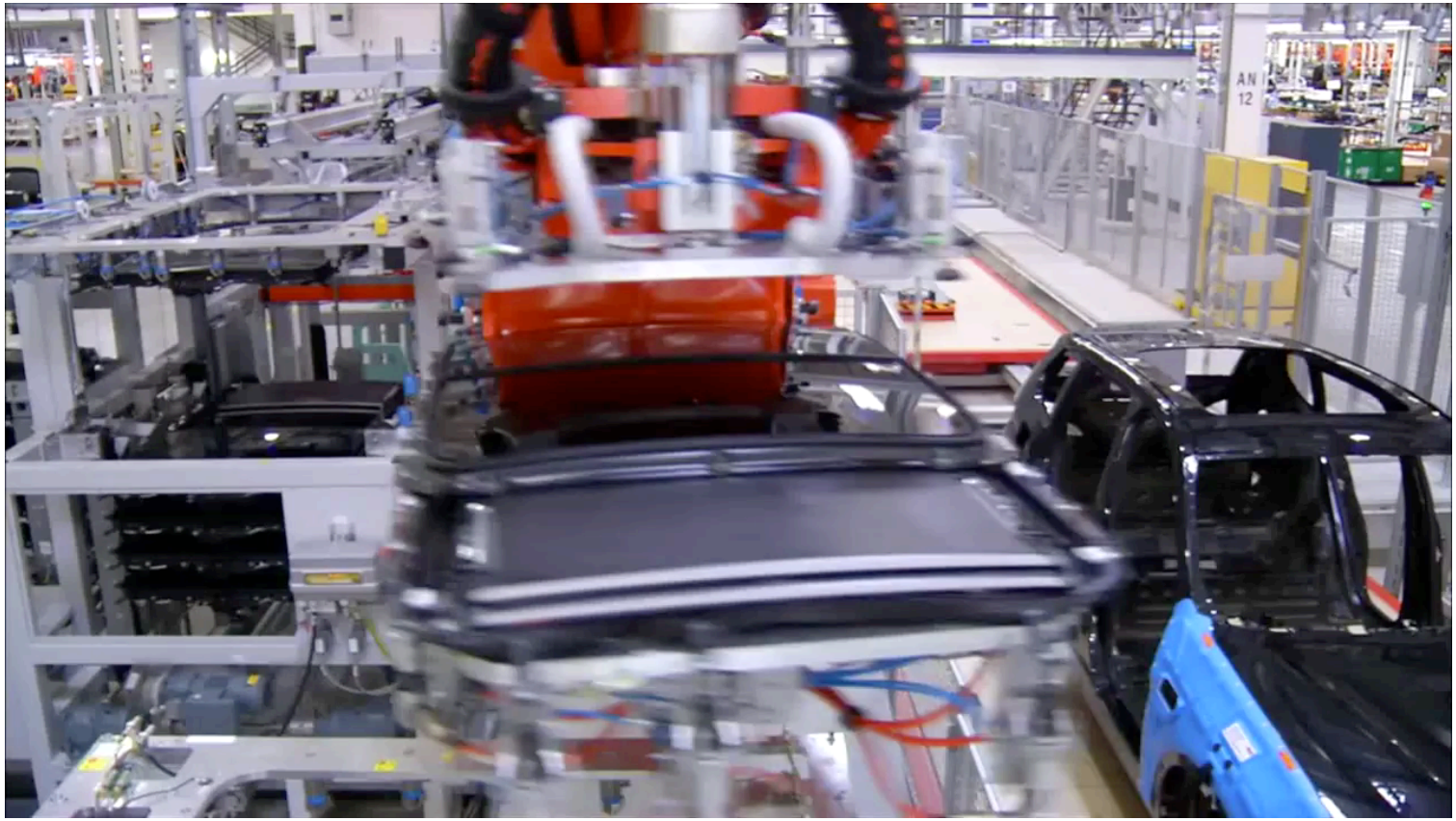
Introduction to Mobile Robot Platform Development

Introduction

- The use of robots started in the 60s to improve manufacturing.
- Robots are very popular since it can be programmed to do different tasks. (design one and use it many applications)



Introduction

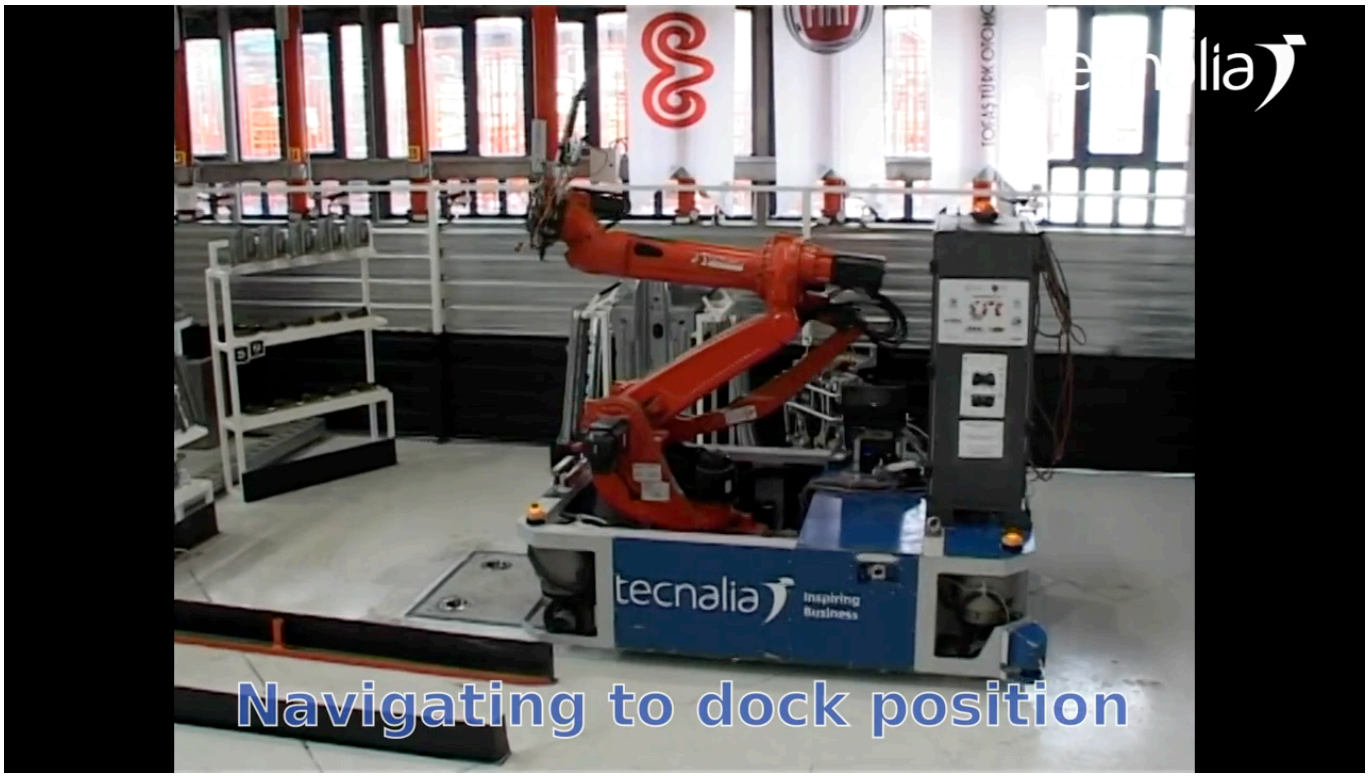


Introduction

- Robots are very successful and people want to use them in new applications.
- However, there is a drawback from the previously shown setups that leads to the development of mobile robots.

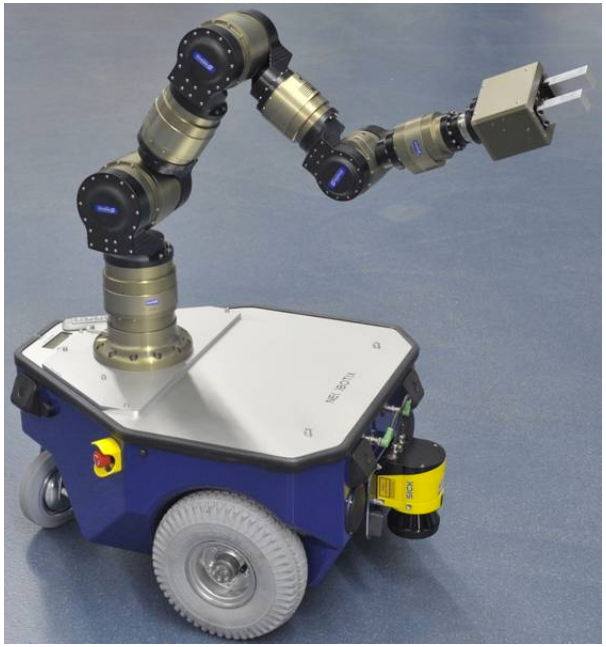
  - What do think is the drawback?

Introduction



Introduction

- Mobile manipulators – one of the original applications the mobile robots.

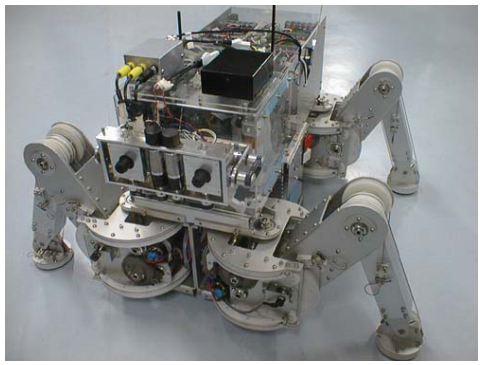


Manipulator Applications

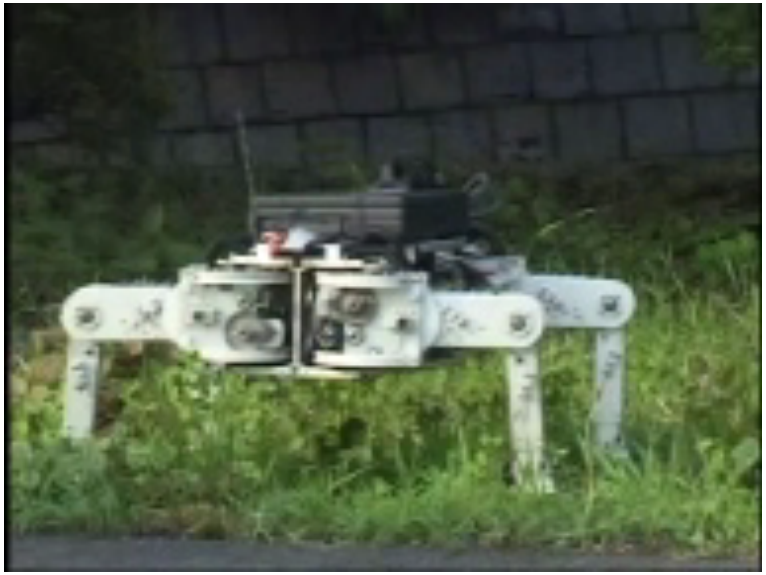
Mobile Robotics

- **Mobile Robots**
  - Robots that have the ability to move in a given environment.
    - They can be wheeled, tracked, or legged





Mobile Robotics

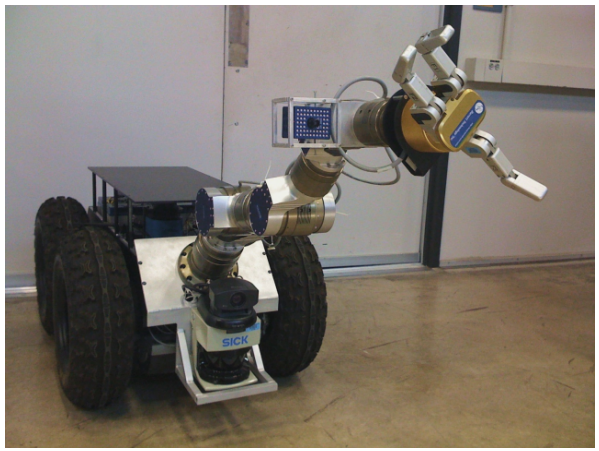


Introduction: Motivation

- **Motivation in developing mobile robots**
  - To increase/provide mobility
    - e.g., mobile manipulators
  - To reduce operation cost
    - e.g., application to mining
  - To improve quality of life
    - healthcare robots
  - To provide access where there was none before
    - space exploration

Motivation

Increase/Provide Mobility

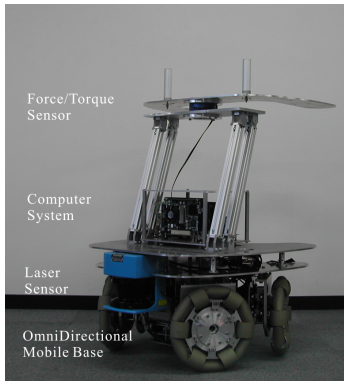


Motivation

Reduce Operating Cost

Motivation

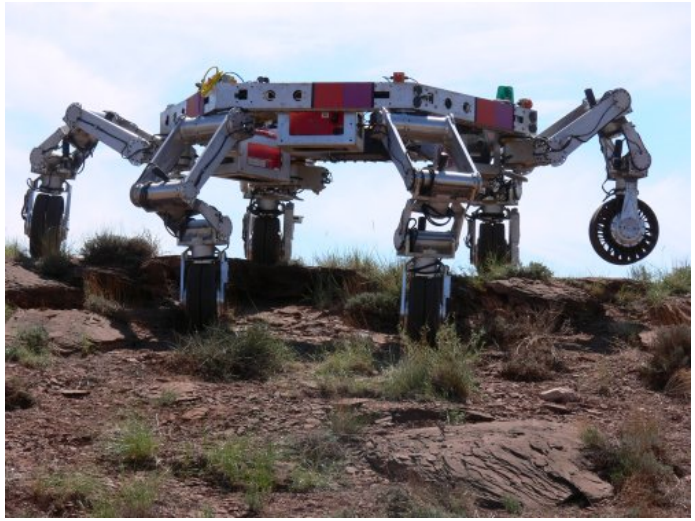
Improve quality of life

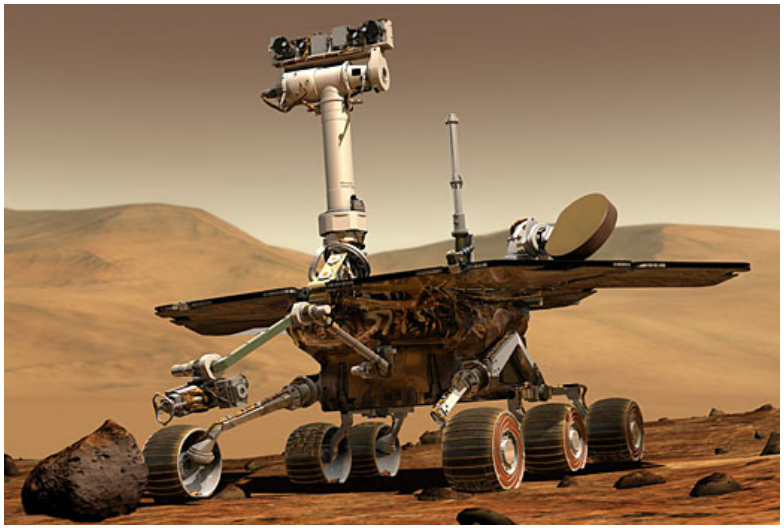


Healthcare Robots

Motivation

Provide Access



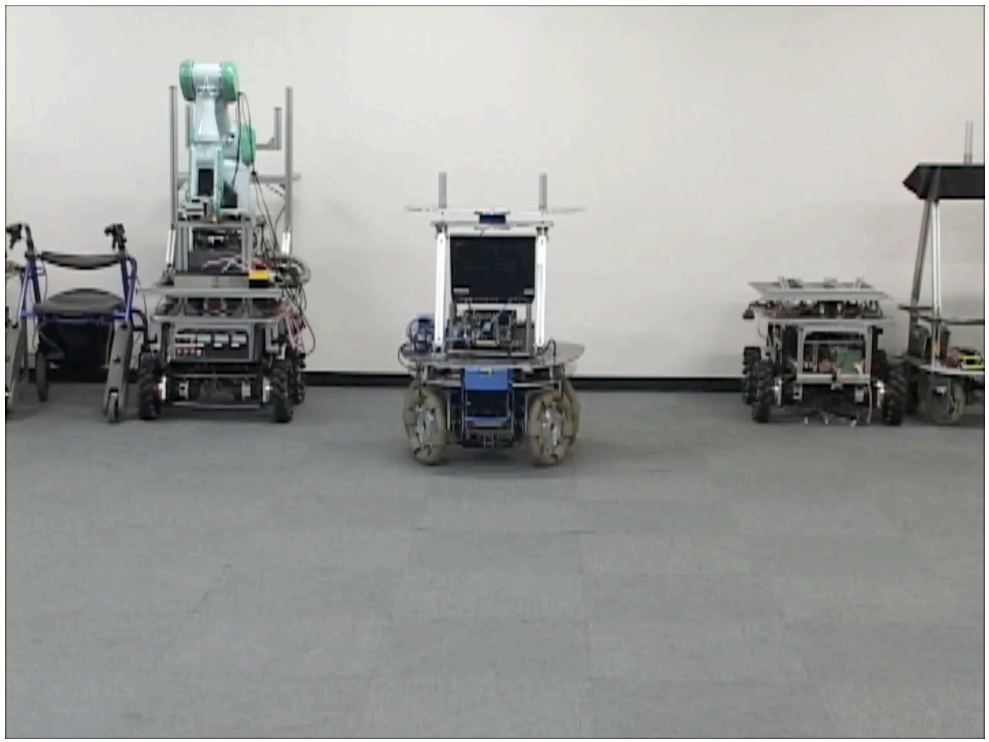


Challenges

Terrain Challenges

Questions

Roadmap



Objective: Understand different hardware and software layers of mobile robots

Roadmap

- **Locomotion Mechanism**(Wheeled and legged)
- **Kinematics** (Inverse and forward)
- **Mobile Robot Platform Hardware**
  - Requirements
    - (Please review your C programming)
    - Review your mathematical tool (Vectors and Matrices Operation)
  - DC Motors
  - Encoders
  - Pulse Width Modulation (Generating Variable Voltage)
  - Feedback Control
  - Realtime: Concept and Implementation
  - Integration

Future Directions

- Future Directions
  - Path Tracking Control
  - Perception (Sensors : Lidar, IMU, and Vision system)
  - Concept of Localization
  - Path Planning

Locomotion Mechanism

Locomotion

- **Locomotion mechanism**
  - Allows mobile robots to move. Transfers actuator's motion to robot motion. For example, from rotational motion of the motors to robot motion (linear and angular velocities).

Locomotion

- Common type of locomotion mechanisms
  - Wheel, Leg, and Track

Locomotion

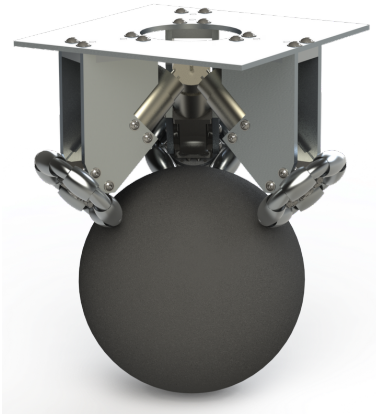
- Wheels:

Standard wheel

Universal omniwheel

Mecanuum wheel

Caster



Spherical wheel

Locomotion

- Robot using spherical wheels

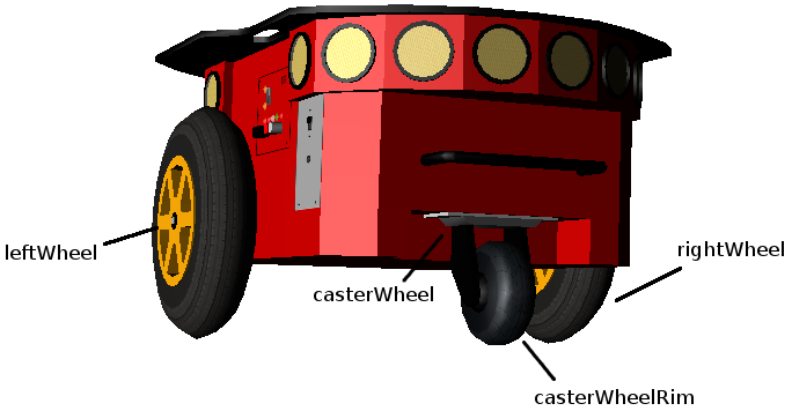
Locomotion: Wheel Configuration

- Wheel configuration
  - Placement of wheels and related to the analysis on stability

Bicycle/motorcycle

Two powered wheels without a caster

Locomotion: Wheel Configuration

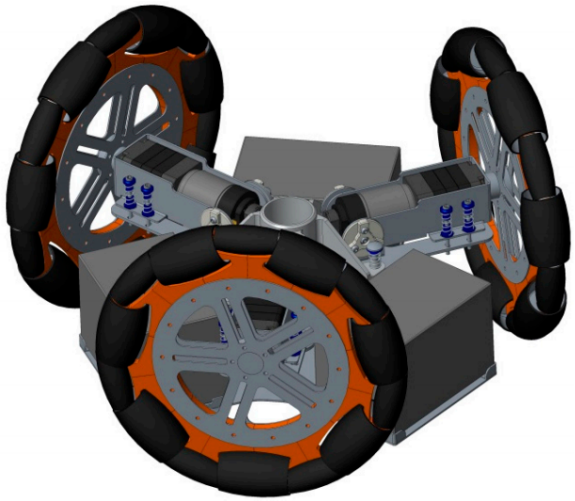


Two powered wheels
with a caster

Four Mecanuum wheels

Locomotion: Wheel Configuration

- Four/three universal wheels



Mobile Robot Hardware

Introduction

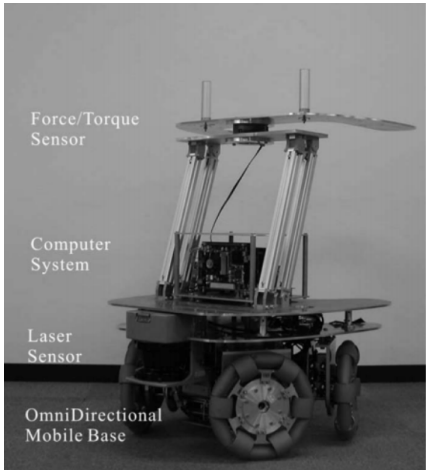
- **Mobile Robots**
  - Have mobile base with control system such that they will be able to track commanded velocities to accomplish given tasks.

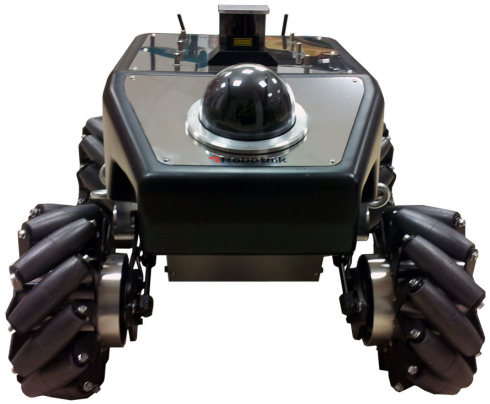
- **Mobile Base**
  - Generally contains DC motors as actuators that convert electrical energy to motion.
  - Use **wheels** or legs

Introduction

Differentially Steered Mobile Platform

Skid Steered Mobile Platform





Omnidirectional Mobile Platform

Task:

Identify the components of a mobile base

Mobile Base

- **Mobile Base Components**

  1- Computer (microcontroller)

  2- Actuators - DC motors

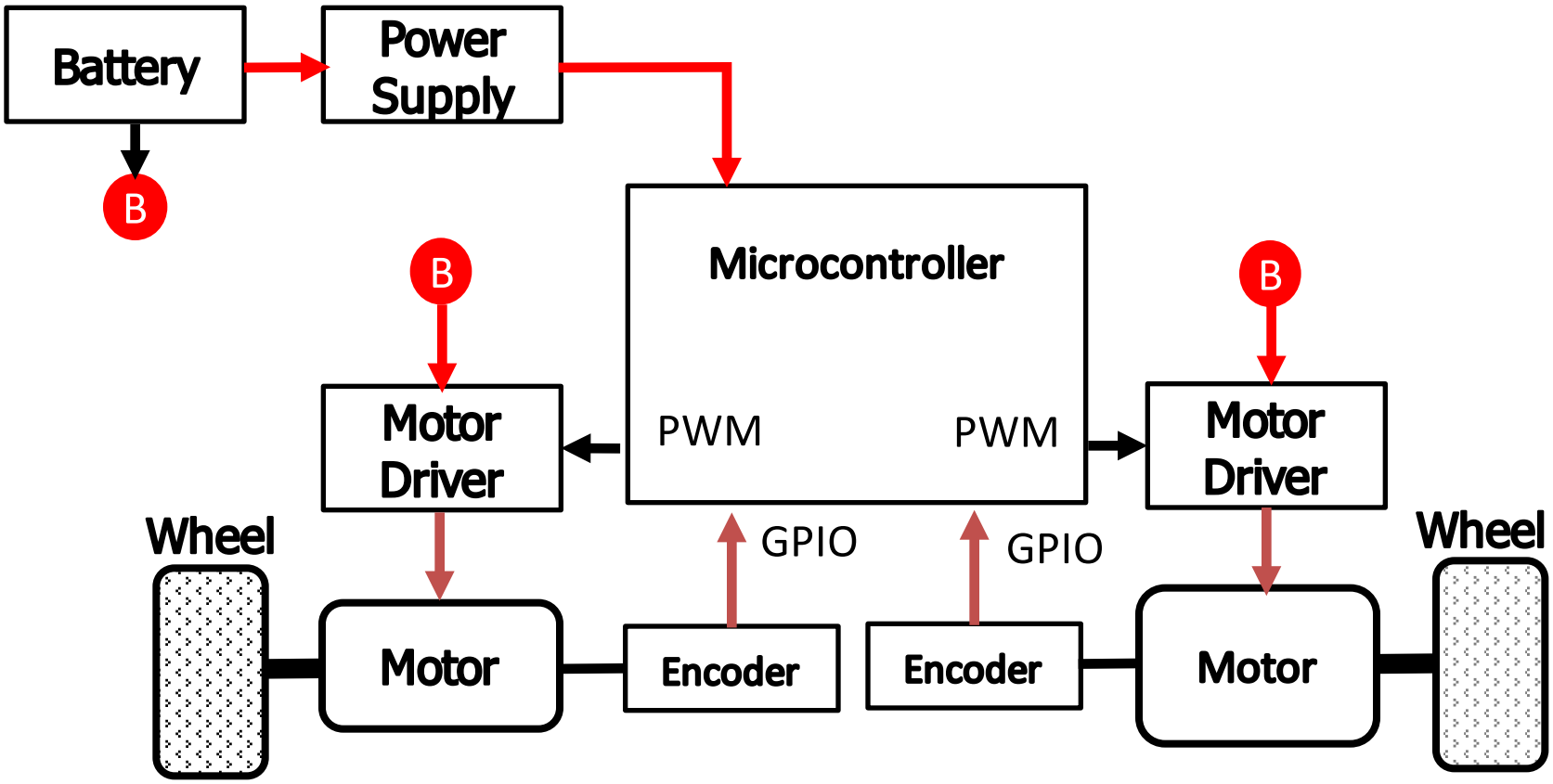
  3- Power Amplifiers – Motor driver (current amplifier)

  4- Sensor – read state of the mobile base, e.g., encoders is used to track wheel position for localization and feedback

    - Localization – to know where you are in the environment

  5- Locomotion mechanism (wheels, legs or track)

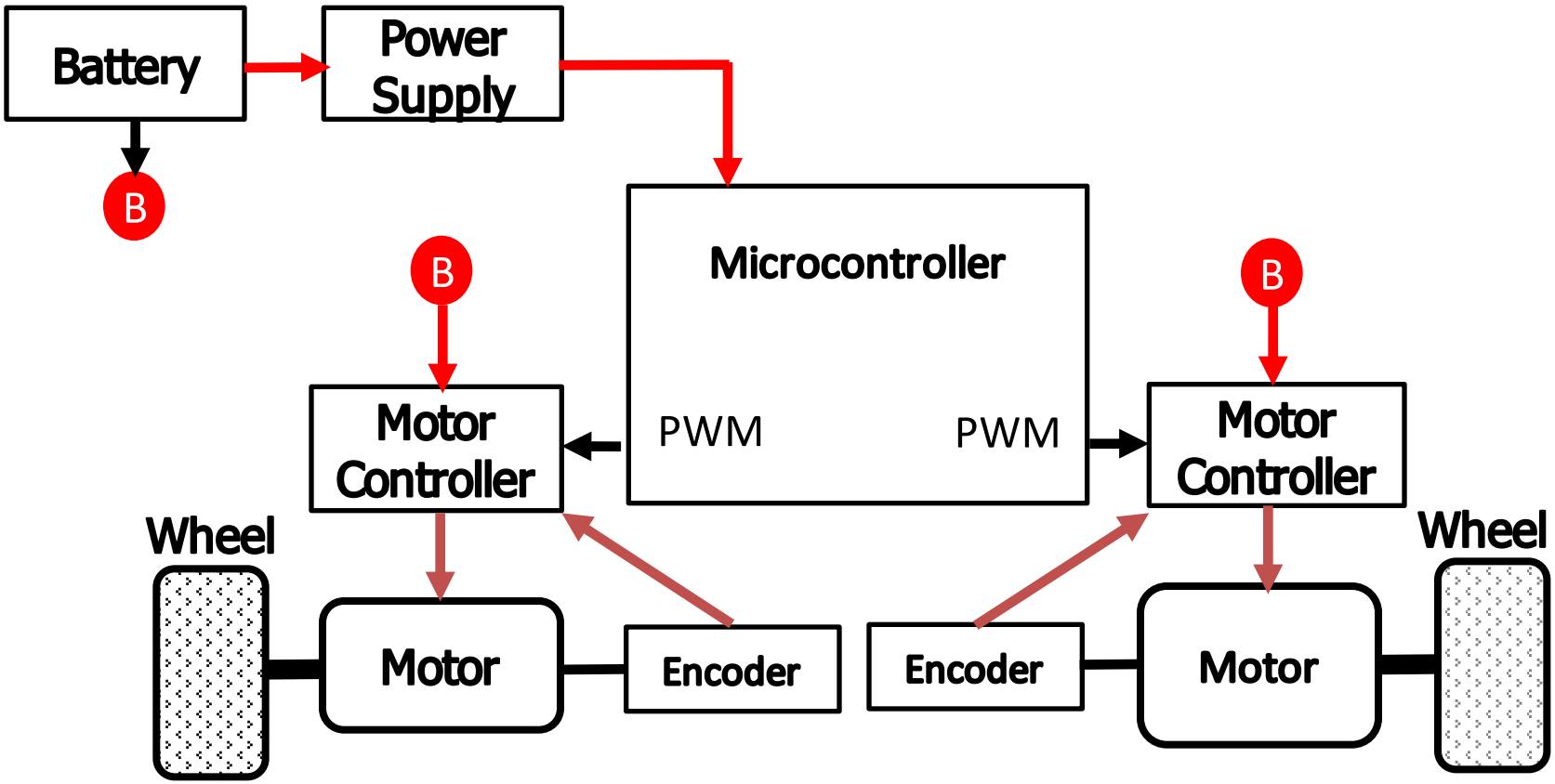
Hardware Diagram of a Mobile Base



PWM - Pulse Width Modulation

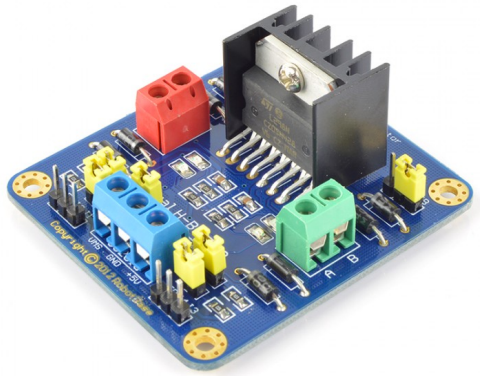
GPIO – General Purpose Input/Output

Hardware Diagram of a Mobile Base



Hardware

- Motor driver
  - simply a power amplifier
- Motion/motor controller
  - Power amplifier plus feedback controller to track desired velocity



Motor Driver

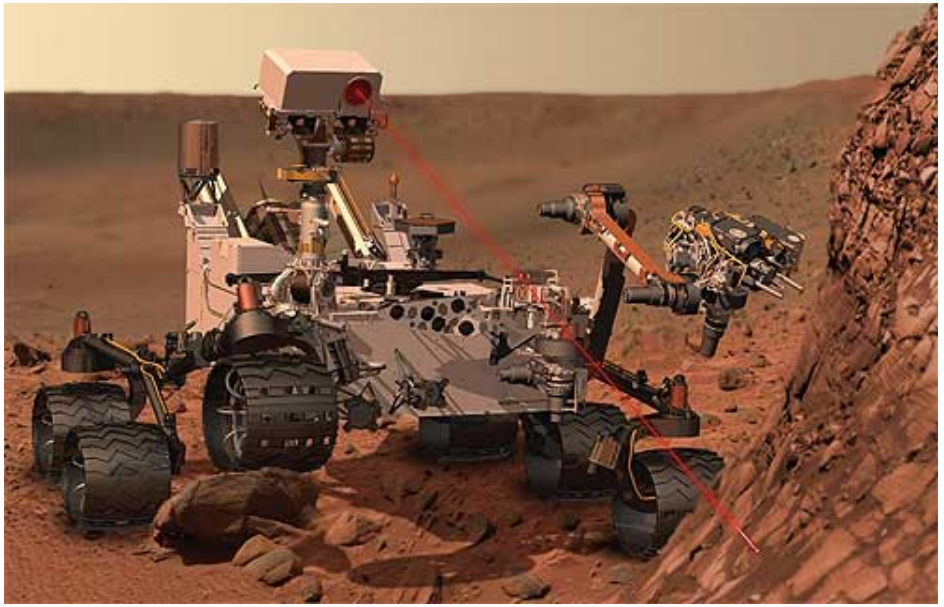
Motor Controller

DC Motor and Pulse Width Modulation

Introduction to Mobile Robotics

DC Motor

- **Direct Current (DC) motor**
  - Converts electrical energy into rotational mechanical energy
  - Earliest form of motor and it is easy to control.
  - High torque and good speed controllability
  - Typically used in robotic manipulators and mobile robots.
  - Considered as torque generator



DC Motor

- **DC Motor Fundamentals**
  - current flowing through the motor is proportional to the generated torque

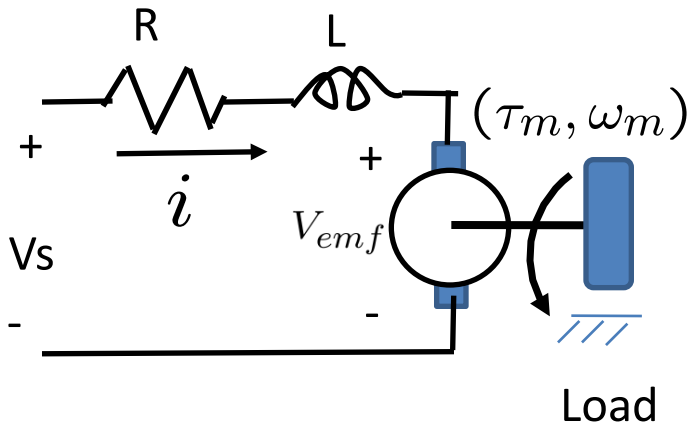
$$\tau_m = K_m i$$

  - angular rotation generates back-electromotive force

$$V_{emf} = K_v \omega_m$$

  - operation:
    - Vs is applied and current is generated which translates to torque generation

Motor Model



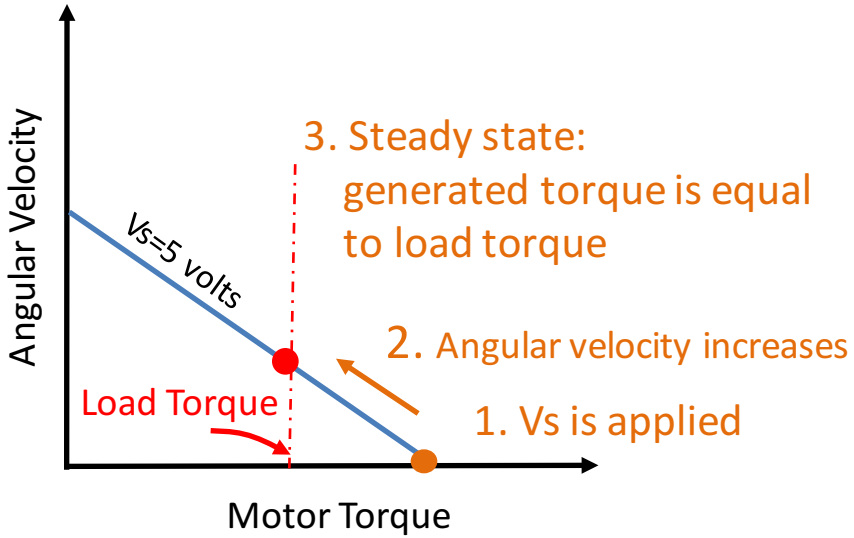
$$V_s = iR + L\frac{di}{dt} + V_{emf}$$

$$V_{emf} = K_v \omega_m$$

$$\tau_m = K_m i$$

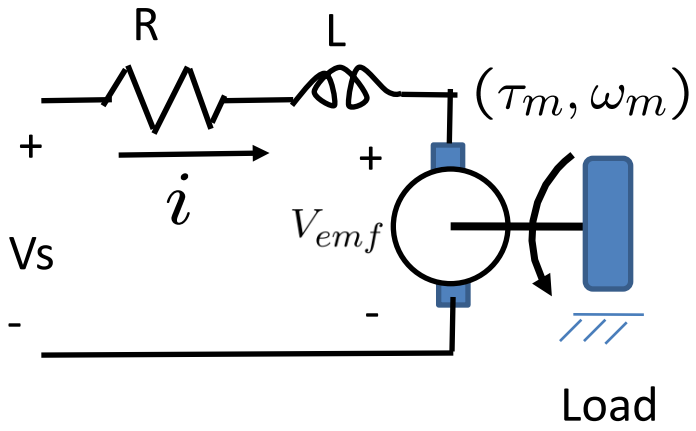
R armature resistance
L armature inductance
Kv voltage constant V/rpm
Km torque constant

DC Motor



For constant load torque, as the input voltage Vs is increased, the angular velocity _____.

Motor Model



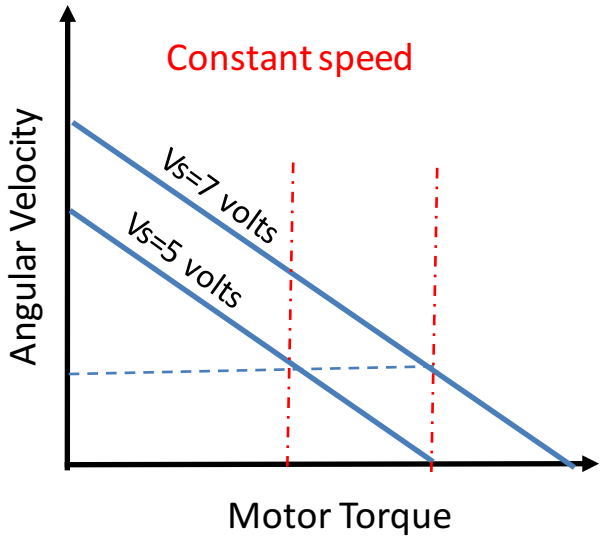
$$V_s = iR + L\frac{di}{dt} + V_{emf}$$

$$V_{emf} = K_v \omega_m$$

$$\tau_m = K_m i$$

R armature resistance
L armature inductance
Kv voltage constant V/rpm
Km torque constant

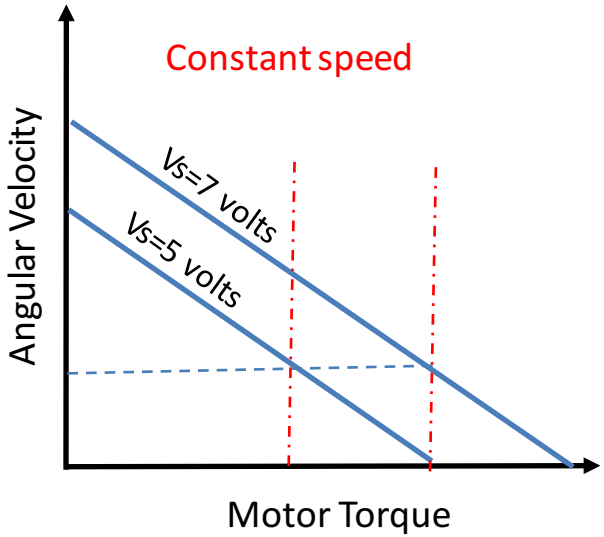
DC Motor

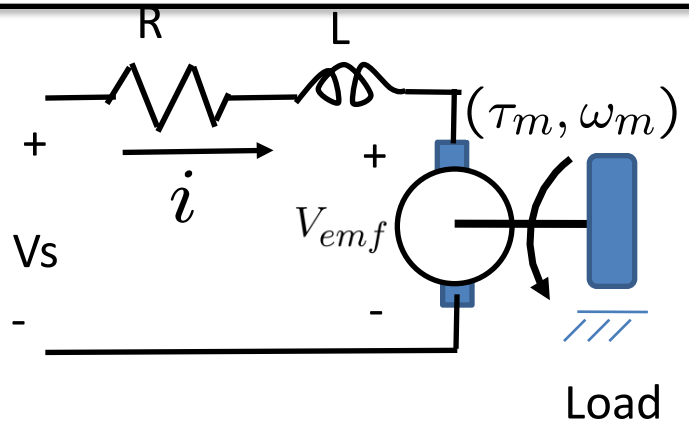


Constant angular velocity with varying load

What variable needs to be varied to achieve constant velocity?

DC Motor Control





$$V_s = iR + L\frac{di}{dt} + V_{emf}$$

$$V_{emf} = K_v \omega_m$$

$$\tau_m = K_m i$$

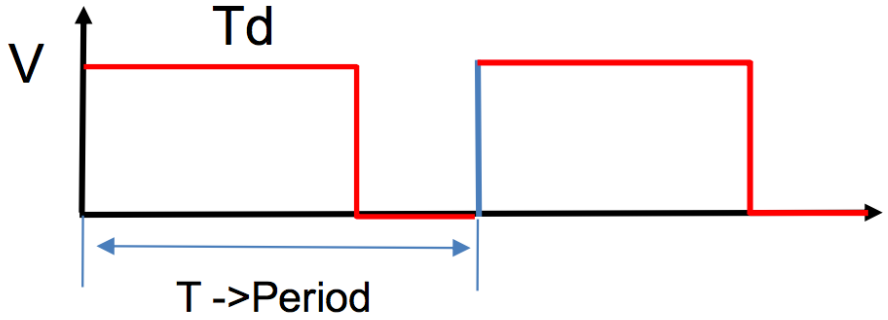
How can we generate a varying Vs from a constant voltage source (e.g., battery)?

Introduction

- **Pulse Width Modulation**
  - an efficient way of controlling motors
  - DC or average value is changed by varying the duty cycle

Introduction

- **Pulse Width Modulation**
  - Typical PWM freq. = 1/T is 100 Hz to 10 kHz.



$$V_{dc} = \frac{1}{T}\int_0^T v(t)\,dt$$

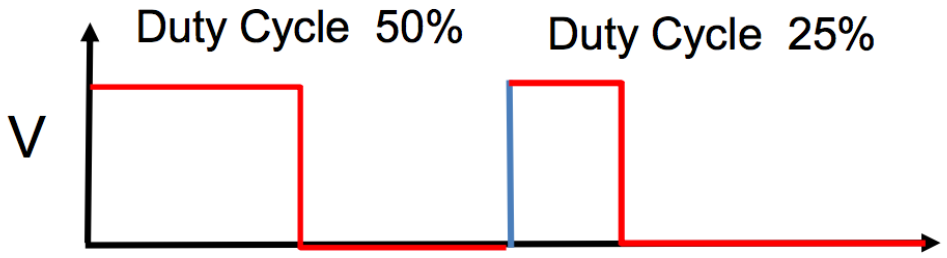
$$V_{dc} = \frac{1}{T}\,Area$$

PWM

- **Duty Cycle**

$$\text{Duty Cycle} = \frac{\text{Pulse is high (duty)}}{\text{Period}} \times 100\%$$

$$\text{Duty Cycle} = \frac{Td}{T} \times 100\%$$

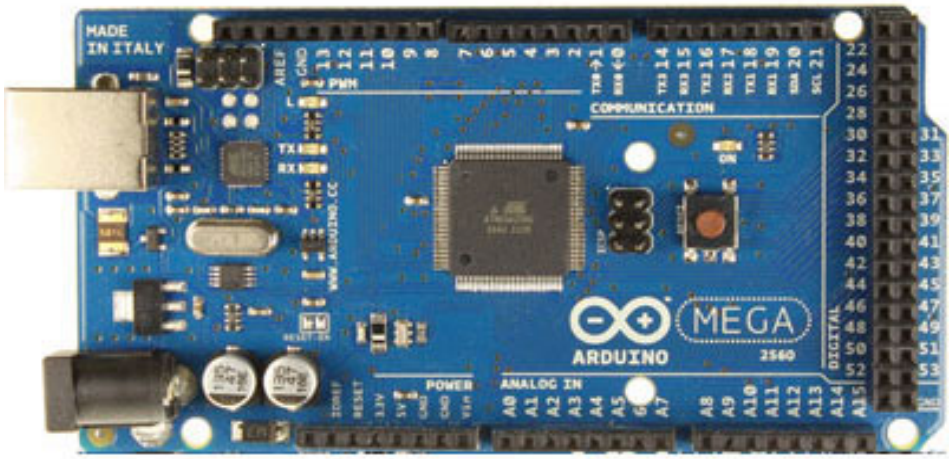


Pulse Width Modulation

- Draw a PWM signal with 75% duty cycle

PWM and Microcontroller Bit Representation

- **Computer (Microcontroller)**
  - Arduino Mega 8 bit microcontroller will be used to control the mobile base



PWM and Microcontroller Bit Representation

- **BIT or bit**
  - smallest memory unit
- **N bit system (microcontroller)**
  - Represents N bit data size, register, data bus, and address bus.
  - For example, an 8 bit system implies that the registers are 8 bit.
- **N bit PWM**
  - Implies that 0-100% duty cycle is represented by 0 – $(2^N - 1)$.
  - For example an 8 bit PWM has the mapping below

duty cycle (%)	
0	0
50	127
100	255

PWM and Bit Representation

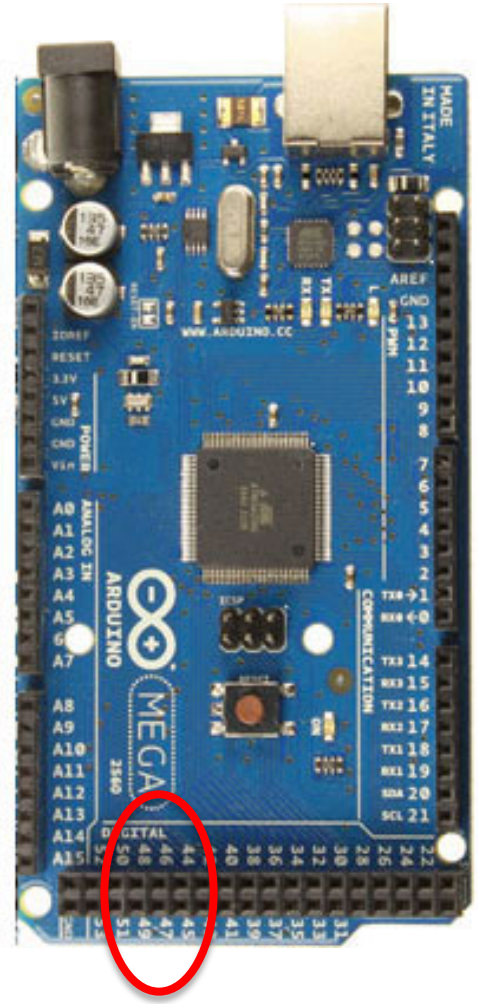
- In terms of minimum change of output voltage (output voltage resolution), what does a high number of bits imply?

PWM on ATmega2560

- **Atmega2560 (Microcontroller)**
  - Used by Arduino Mega
  - Note: Built in Arduino function for PWM has a frequency of 500 Hz, which is intended RC servo motors and other devices. Some applications require different freq. or higher frequency. Hence, we will not use Arduino built in function.

- **In the given sample code, PWM generation will be based on TIMER 5**
  - Pin 45 for motor 0
  - Pin 46 for motor 1

PWM Implementation Using Arduino Mega



Code Initialization

pwm_init();

```
void pwm_init(void){
    pinMode(45, OUTPUT);
    pinMode(46, OUTPUT);
    TCCR5A = _BV(COM5A1) | _BV(COM5B1) | _BV(WGM52) |
    _BV(WGM50);

    TCCR5B = _BV(CS51) | _BV(CS50);   //set prescaler to 64
    OCR5A  =  0;     OCR5B  =  0;
}
```

The above code (pwm_init) initializes timer 5 as PWM generators(2 PWMs). The PWMs are 8 bits at 500 Hz.

TCCR5A, TCCR5B, OCR5A, and OCR5B are microcontroller registers and already defined in the ARDUINO environment.

PWM Duty Cycle

- **OCR5A (Output Compare of TIMER 5)**
  - Will set the duty of channel/motor 0. The maximum value of duty is 255.

255 -> 100% duty cycle

127 -> 50% duty cycle

0 -> 0% duty cycle

PWM Duty Cycle

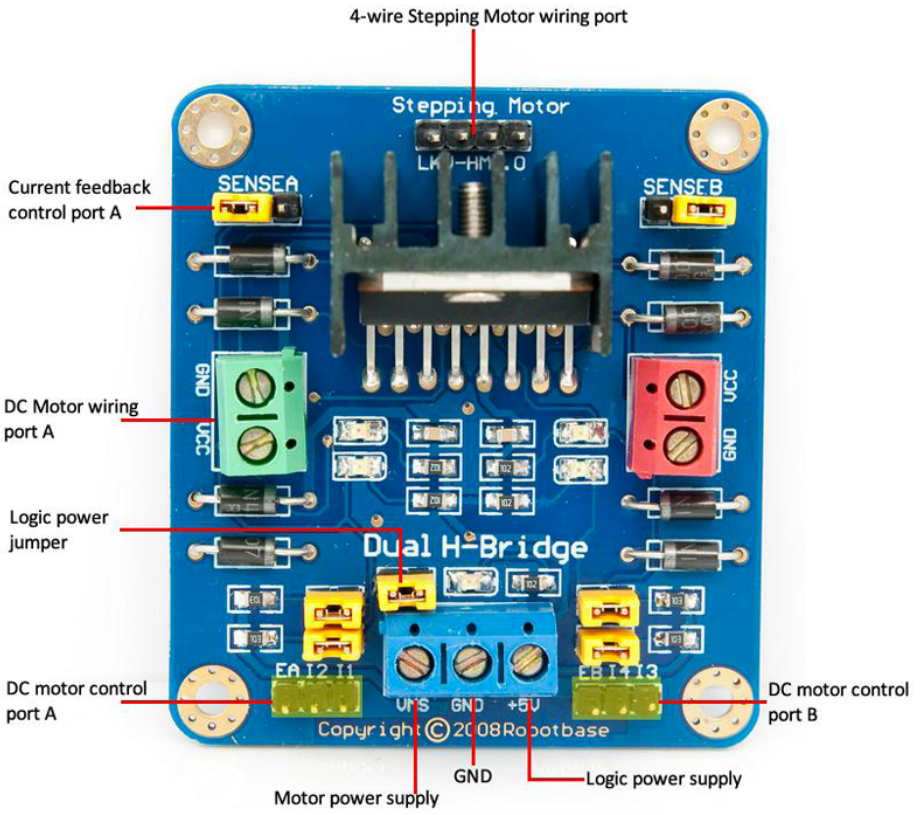
- **OCR5B**
  - Will set the duty of channel/motor 1. The maximum value of duty is 255.

255	->	100% duty cycle
127	->	50% duty cycle
0	->	0% duty cycle

Motor Driver

Motor Driver

- **Current amplifier**
  - Allows low power signal from microcontroller to drive DC motors
  - EA will be connected to PWM signal for Motor 0
  - I1 and I2 are used for direction control for Motor 0
  - EB will be connected to PWM signal for Motor 1
  - I3 and I4 are used for direction control for Motor 1.



EA	I1	I2	Motor A status
》 0	0	1	Clockwise rotation
》 0	1	0	Anticlockwise rotation

Motor Driver

- **Motor Control Summary**
  - Aside from PWM signal, a motor needs 2 digital pins for direction.
  - For a differentially steered (2 wheels) mobile robot
    - 2 motors – 2 PWM signals and 4 direction pins

Motor Functions

```
void motor_init(void);

void set_motor_duty(int channel, int duty);
```

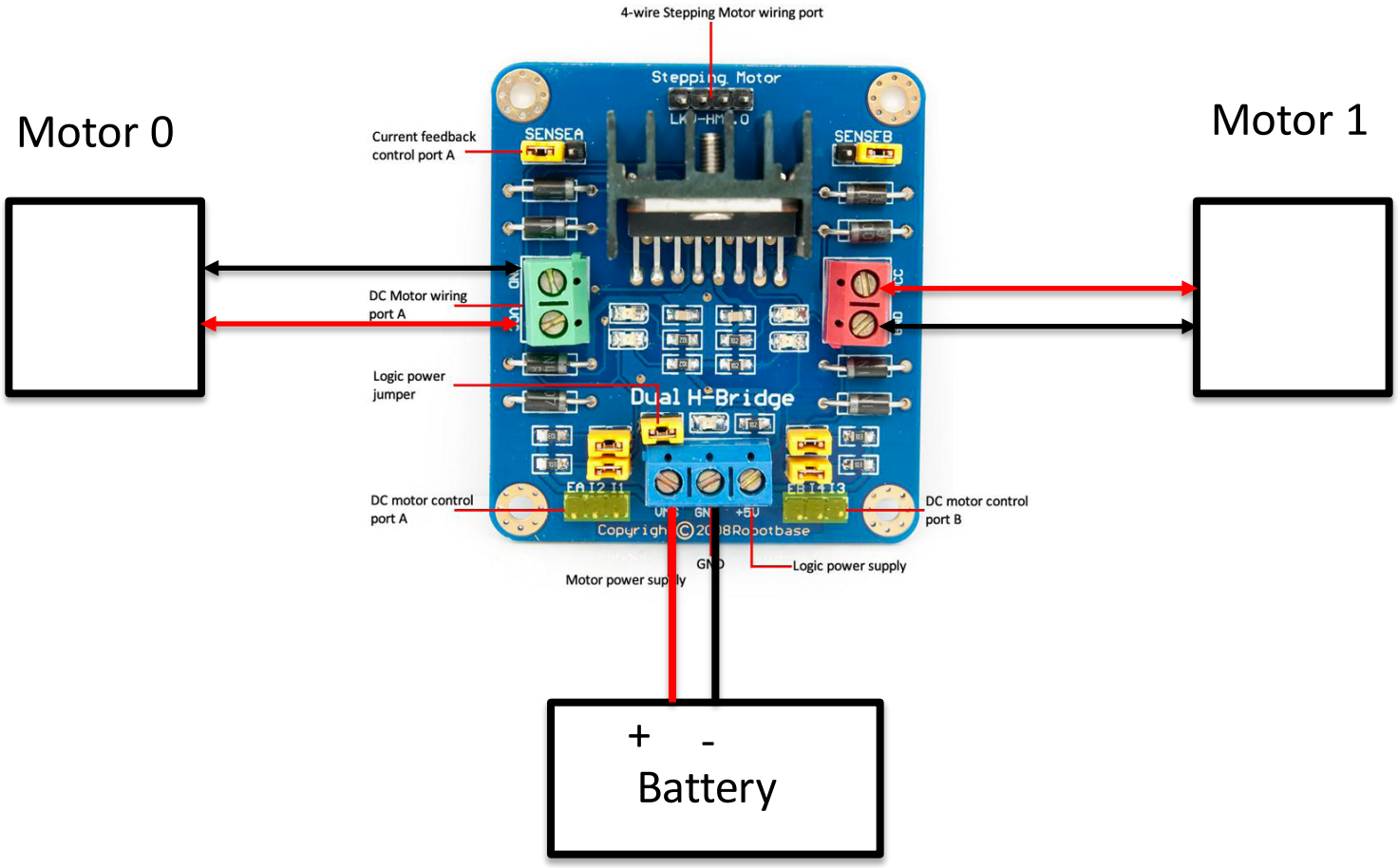
motor_init() initializes the PWM and direction pins of the motors

set_motor_speed(channel, speed)
channel = 0 implies motor 0 and channel = 1 -> motor 1
speed = 127 implies 50% duty cycle moving forward
speed = -127 implies 50% duty cycle moving backward

Trouble Shooting

`set_motor_speed(0,100)` should yield a forward motion. If it is backward, you can reverse the motor wiring (+ -> -) or change the direction code.

Diagram


4-wire Stepping Motor wiring port
Motor 0
Motor 1
Current feedback control port A
DC Motor wiring port A
Logic power jumper
DC motor control port A
DC motor control port B
Motor power supply
Logic power supply
Battery
+
-
SENSEA
SENSEB
Dual H-Bridge

Pins

Arduino		Motor Driver
45	PWM	EA
50	Direction	I1
51	Direction	I2
46	PWM	EB
52	Direction	I3
53	Direction	I4
GND		GND
+5V		+5V

Sample Code

```
#include "mrobot.h”
void setup()
{
    motor_init();
}
void loop() {
    set_motor_speed(0,100);
    set_motor_speed(1,100);
}
```

Summary

- The provided motor functions can change the voltage and thereby change the speed of the motor.

- The main drawback is that the implementation is open loop, meaning for a constant duty or voltage, the speed will change if the load varies.

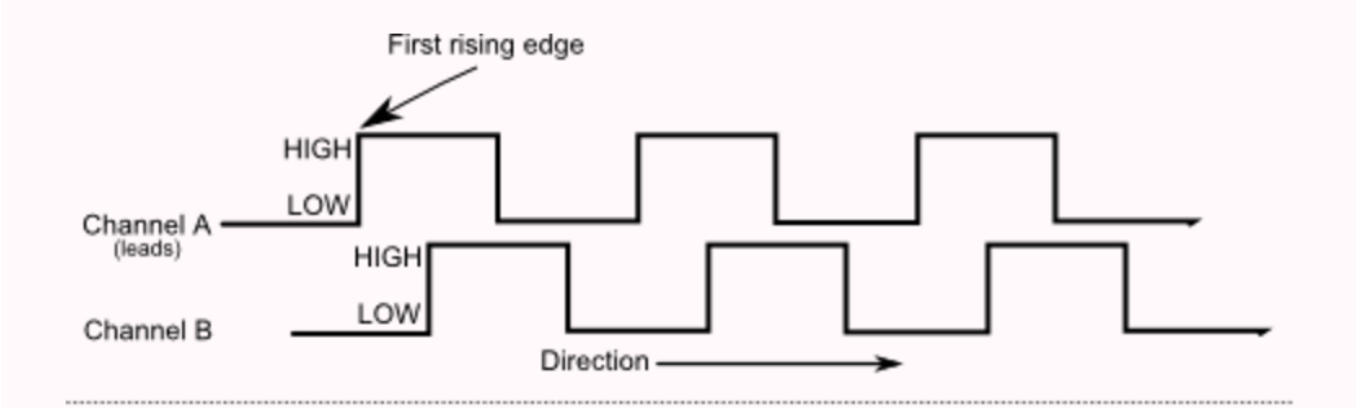
Mobile Robot: Measuring Wheel Position Using Incremental Encoder

Introduction

- **Incremental Encoder**
  - Use to measure angular position and generally attached to motor shaft.

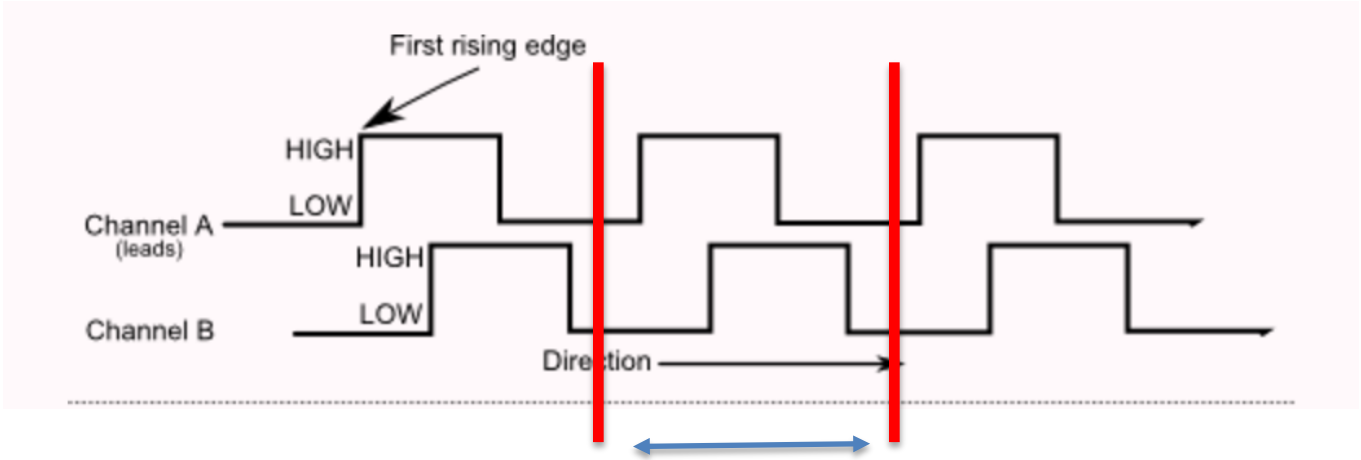
Introduction

- **Incremental Encoder**
  - Generates two signals that are 90 deg. out of phase



Introduction

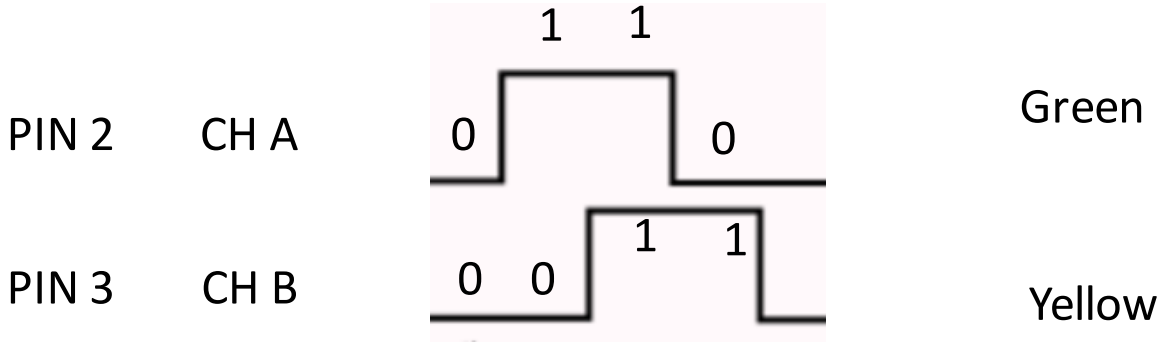
- **Decoding Incremental Encoder (Quadrature Decoding)**
  - Channel A and Channel B will be wired to interrupt pins. If there is a change in state for example high to low or low to high an interrupt is triggered.
  - Quadrature decoding – all rising and falling edges create interrupts. Quadrature since there are four sign changes in Ch A and Ch B for one encoder count.



1 encoder count

Introduction

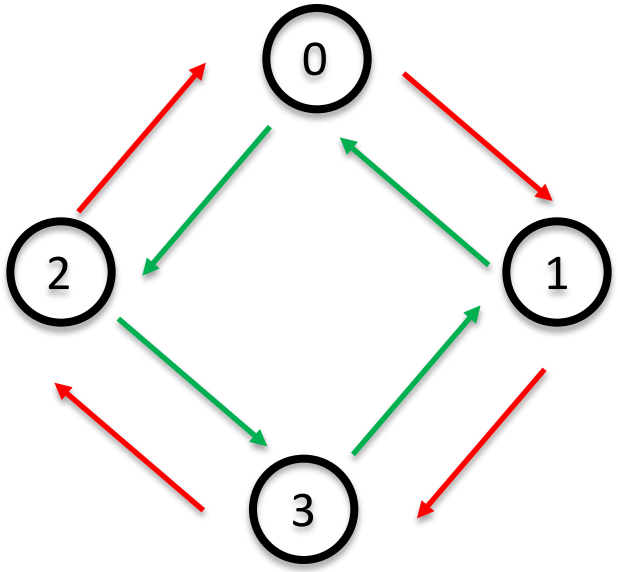
- **Decoding Incremental Encoder (Quadrature Decoding)**
  - State assignment – assign states to one count as basis for determining direction.



	CHB CH A			
	0 0	0 1	1 1	1 0
State	0	1	3	2

Decoding

- Conditions:
  - If current state is 0 and next state is 1 then in clockwise direction (increment the value of the encoder)
  - If current state is 0 and next state is 2 then in counter clockwise direction (decrement the value of the encoder)



Encoder

Red = +5vdc
Black = Ground
Green = Output A
Yellow = Output B

Encoder functions

```
encoder_init();
```

Initialize the encoder. Global variable `encoder0_val` will have the position of motor 0 and `encoder1_val` will be the position of motor 1.

Note the values of `encoder0_val` and `encoder1_val` are in terms of counts. For example, 20000 counts per wheel revolution.

Counts per wheel revolution = 4* encoder counts per rev * motor gear ratio

Wheel Position (radians) = encoder_val / (counts per wheel revolution)

Test Code

```
#include "mrobot.h”
void setup()
{  encoder_init();
   serial.begin(9600);
 }

void loop()
{
  Serial.println(encoder0_val);
//Serial.println(encoder1_val);
}
```

Encoder

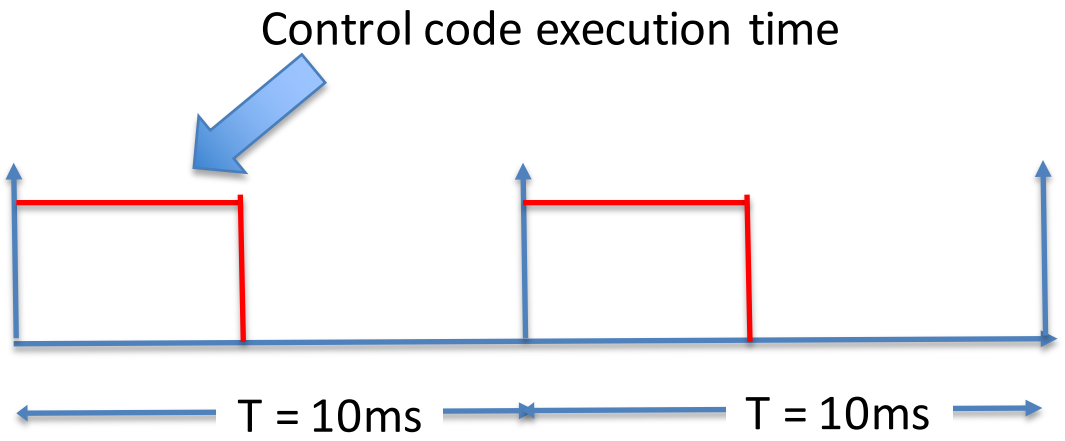
Arduino Pins	Encoder	
2	Encoder 0 CHA	Left Wheel
3	Encoder 0 CHB	
18	Encoder 1 CHA	Right Wheel
19	Encoder 1 CHB	

Note encoders have +5 and GND pins.

Real-time Interrupt (Timer)

Timer

- **Timer (Realtime Interrupt)**
  - Creates a deterministic execution of a function
  - For example, use to numerically determine velocity from position information (encoder)



Timer

- Deterministic function execution
  - Allows to accurately determine time derivative of a signal for example velocity

$$velocity\ [n] = (position[n] - position\ [n-1])/T$$

Current velocity = (current position – previous position)/T

Timer Implementation (Timer1)

```
void setup_timer(void);
```

The function setup_timer() creates an interrupt every 10ms and execute ISR(TIMER1_COMPA_vect)

```
/*The function below is coupled to timer1
interrupt*/


ISR(TIMER1_COMPA_vect)
// timer compare interrupt service routine
{
 control();
}
```

Additional Functions for Mobile Robot Control (mrobot.h)

Additional Functions that are executed every 10ms

```
void get_current_status(void);
```

The above function determines the current velocity of the wheels.

```
void low_level_control(void)
```

Computes the desired voltage and duty to track the desired velocity.

$$voltage = K_p(desired\ position - current\ position) + K_d(desired\ velocity - current\ velocity)$$

Low Level Control

- Based on PD (Proportional + Derivative) Control

$$voltage = K_p(desired\ position - current\ position) + K_d(desired\ velocity - current\ velocity)$$

- Given Vcc and for a 8 bit PWM, what is the expression for duty?

duty =

Sample Code

```
#include "mrobot.h”

void setup() {
setup_timer();

    /*This will setup an Interrupt Service Routine
      to be executed at 10ms*/

encoder_init();
motor_init();
Serial.begin(9600);
 }
void loop() {

}
```

Sample Code

```
void control(void){
get_current_status();
   /*****************************/
des_wvel[0] = 0.25;  // set motor 0 to 0.25 rad/sec
des_wvel[1] = 0.25;  // set motor 1 to 0.25 rad/sec
   /*****************************/
low_level_control();
}
/**This function will be executed every 10ms **/
ISR(TIMER1_COMPA_vect) /* timer compare interrupt
service routine*/
{
  control();
}
```