Mobile Robots

EECN30169/535307

Kai-Tai Song

Institute of Electrical and Control Engineering

National Yang Ming Chiao Tung University

September 16, 2022

Contents

Chapter 1 Introduction

A. Brief description of EECN30169

1. This is a course concerning design and construction of autonomous mobile robotic systems.
2. Students in this course will build a mobile robot using components provided in the class in order to learn about how a mobile robot works.
3. Electrical drives, motion control, computer programming about building an autonomous mobile robot will be studied and learned through practical exercises.
4. ROS software and robot programming.

B. Feedback control and the use of feedback concept:

1. An autonomous robot is basically an intelligent machine that can understands its own state as well as the external environment such that it can take smart actions based on the acquired information.
2. Control is a type of smart interaction between robot computer and the external world
3. Feedback control is the basic and most important concept in intelligent robot.

C. Making a mobile robot

We will use ready-for-use components to build our mobile robots in this course. Through practical projects, students learn about real-life robotics. We emphasize the idea of "learning by doing" . Mission of EECN30169: A robot-hockey mobile robot will be accomplished via 5 check points step by step. All supplies are provided by the class. To learn about design and construction of an autonomous mobile robot.

D. **course materials**：

1. Course notes
2. Mobile Robots Inspiration to Implementation by Joseph L. Jones, Anita M. Flynn and Bruce A. Seiger, A.K. Peters, Ltd., 1999
3. Introduction to Autonomous Mobile Robots by Roland Siegwart, Illah Nourbakhsh and Davide Scaramuzza, 2nd Edition, The MIT Press, 2011
4. Mastering ROS for Robotics Programming, Lentin Joseph, PACKT Publishing 2015, eBook

Introduction to Intelligent Autonomous Robots

An intelligent robotic system is basically a machine that can understand its own states as well as external world such that it can react accordingly to complete a task. It counts on sensors and actuators, and on-board embedded computer. Feedback control is essential in robotics.

The goal of building an intelligent robot is to construct a intelligent machine that can demonstrate Human-level intelligence. It is required for the intelligent machine to possess the ability of planning its action and execution to complete these actions. Intelligence is difficult to define. Sometimes, it is a concept. When we observe something is intelligent, we can easily know it and we will also know what is not intelligent.Human-level intelligence is very complex.

Intelligent machines have been developed for decades (since the invention of computer), the research has been distributed in many small areas, such as speech recognition, image recognition, path planning, neural networks, and motion planning. In these individual areas, there have been many achievements. Can machine intelligence be built incrementally and grow gradually?

An intelligent behavior is a robot program that allows the robot to survive in an unstructured environment and complete a task. The robot cannot rely on a world model(map) to work in an unstructured and fast changing environment. It is important for an intelligent robot to complete a task in such real-world environments. We realize that an intelligent robot does not need human-level intelligence. Machine intelligence can be built and added incrementally from lower level to higher level. An intelligent robot needs to work in an unstructured and fast changing environment. This means the robot executes a task or running a test in the real world, not in a model in a computer simulator. Autonomous robots are free-ranging robot. That can execute a task without human intervention or supervision. Behavior-based robotics is a method to construct autonomous robots。There are already many autonomous robots exist in the market, such as vacuum cleaning robots. More are under developing.

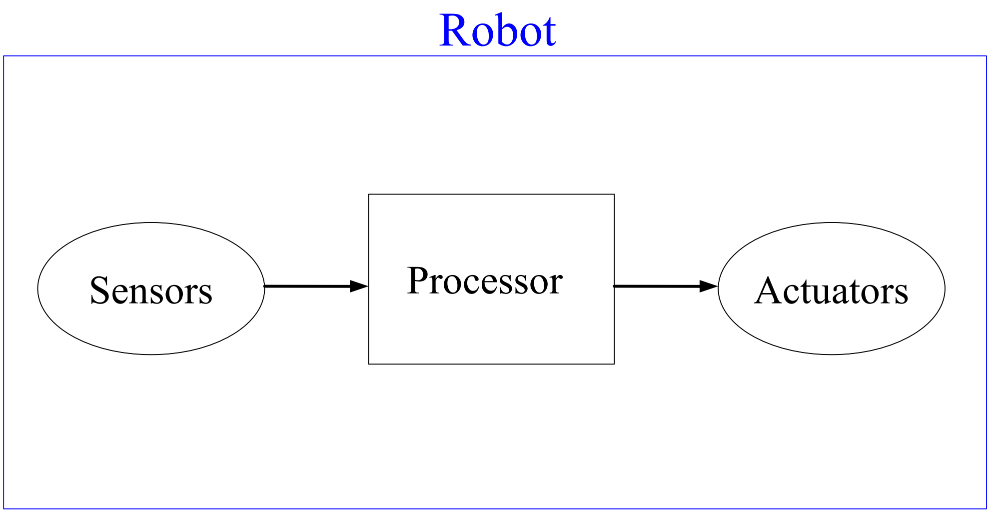


Fig. 1-1 Basic components of a robot

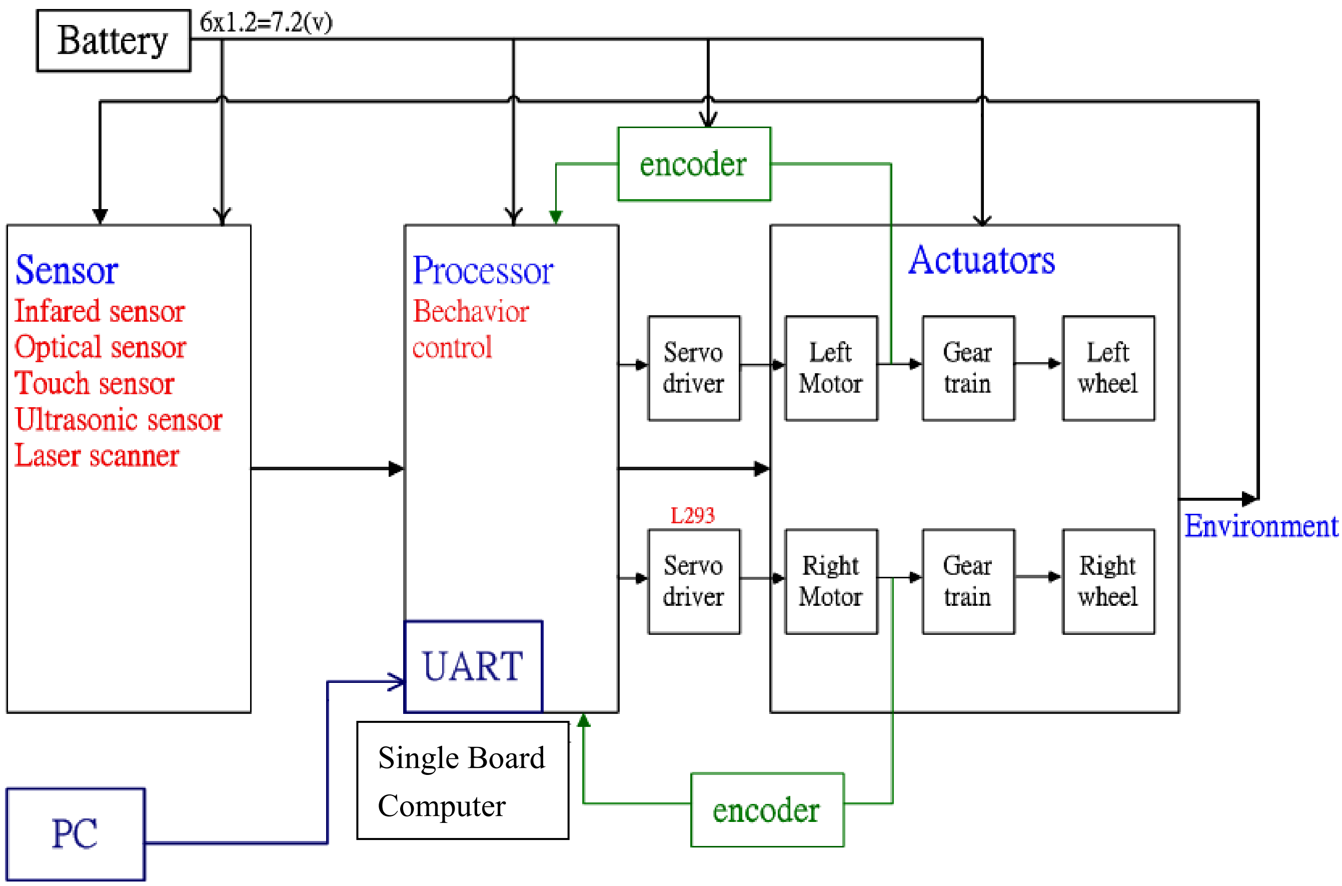


Fig. 1-2 System architecture of a mobile robot

Chapter 2 Embedded Computing System with ROS

In this tutorial we will go through the environment setting on Raspberry Pi, the architecture of ROS and how to connect Raspberry Pi and Arduino by using *rosserial* package.

A. Environment setting on Raspberry Pi
   1. Install Ubuntu mate 18.04
   2. Install ROS **Melodic**
   3. Setting SSH between Raspberry Pi and PC

B. ROS
   1. ROS file system level structure
   2. ROS computation graph level
   3. Create package file
   4. ROS nodes communication
   5. Publisher and Subscriber
   6. CMakeLists.txt
   7. roslaunch

C. *rosserial* package
   1. Arduino IDE setup
   2. Install the Software
   3. Create a publisher by using *rosserial*
   4. Create a subscriber by using *rosserial*

A. Environment Setting on Raspberry Pi

1. Install Ubuntu mate 18.04

(1-1) Download Ubuntu mate 18.04 (64-bit) from https://ubuntu-mate.org/download/

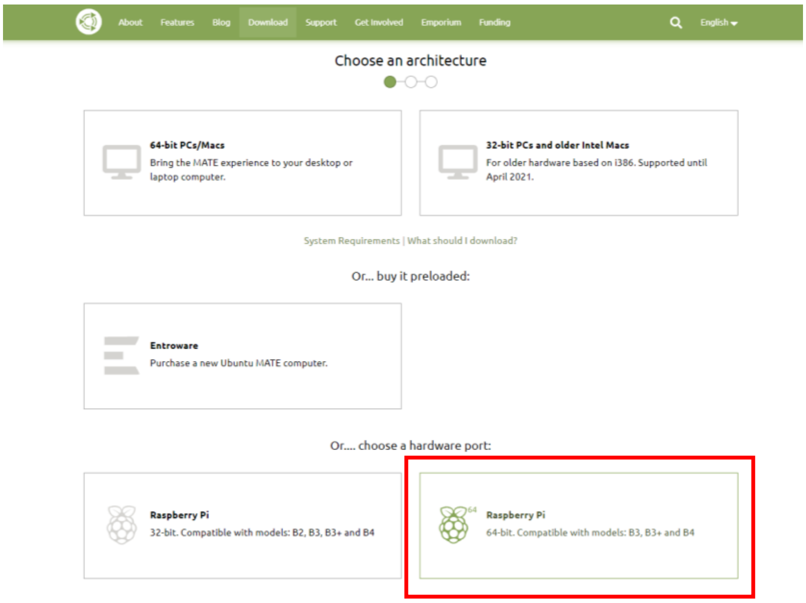


Fig.1 Ubuntu MATE 18.04 image file

(1-2) Then install Ubuntu mate image file in SD card. You can use any tool like GNOME Disk (like Fig.2 showed below) or other applications such as ddrescue on Linux or Win32 Disk Imager on Windows can be used.

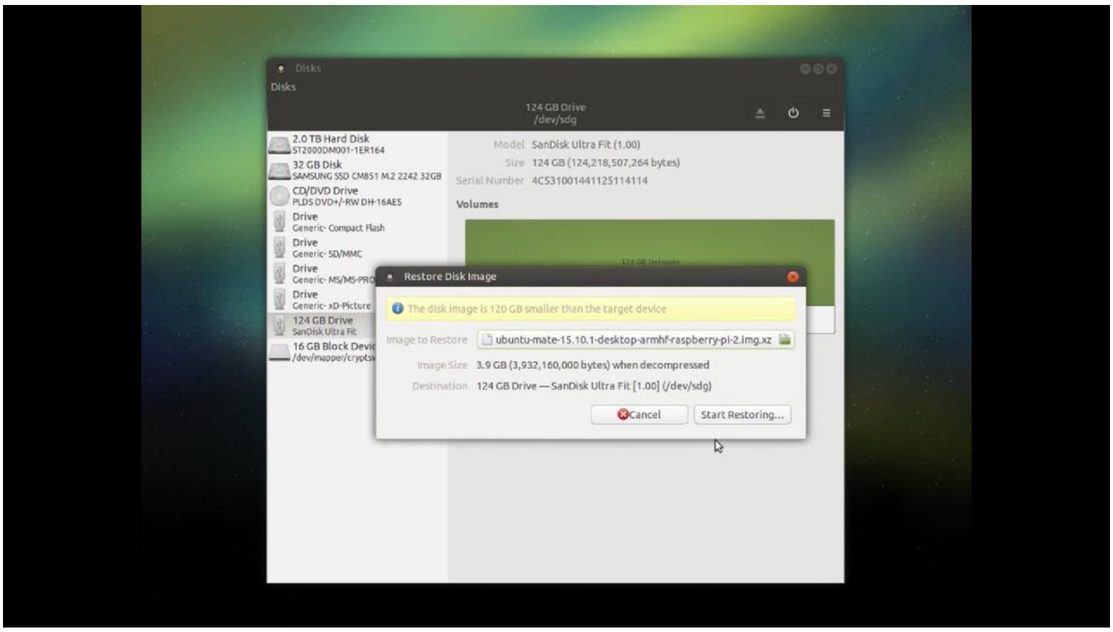
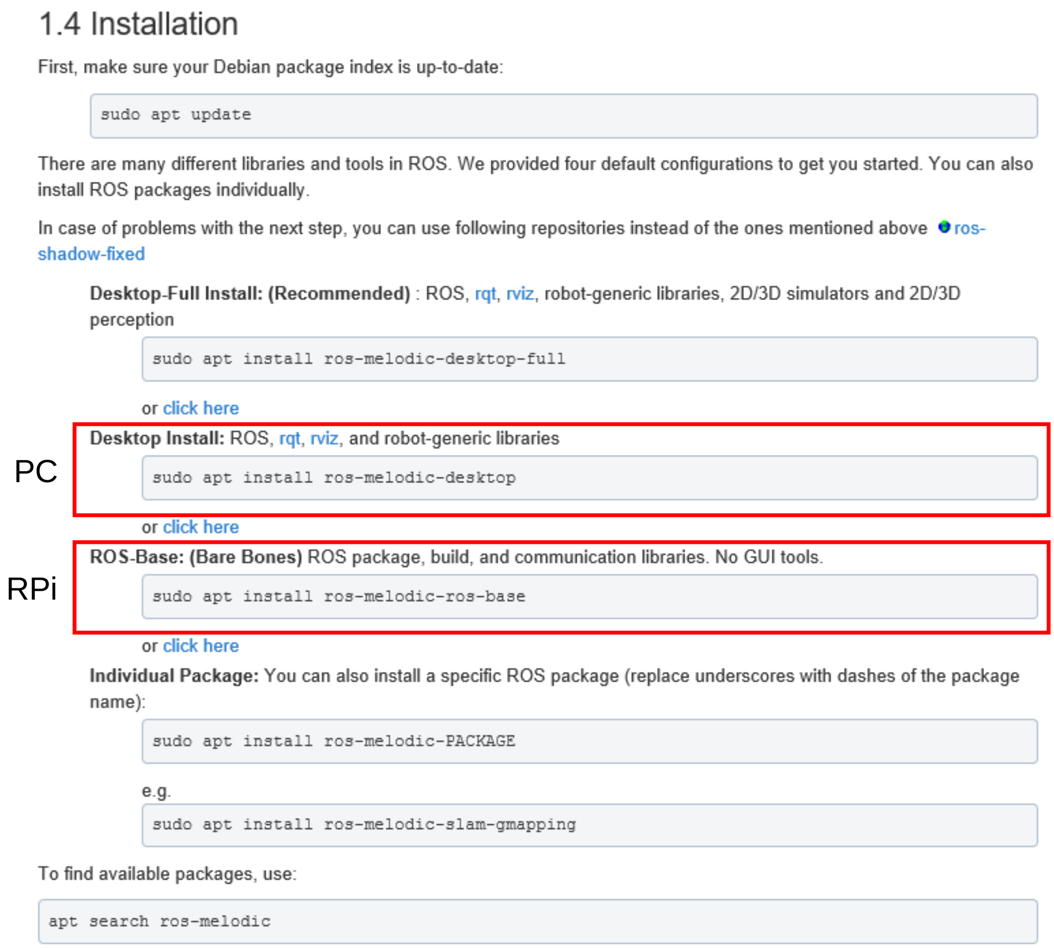


Fig.2 GNOME Disk restore image

(1-3) Then plug SD card in raspberry pi and connect to screen and finish all the initial setting.

2. Install ROS Melodic

(2-1) You can start to install ROS from http://wiki.ros.org/melodic/Installation/Ubuntu "1.2 Setup your sources.list" and keep following the instruction until "1.7 Getting rosinstall".
(2-2) In "1.4 Installation" we recommend you install **ROS-Base** in Raspberry Pi and **Desktop Install** in PC. Show in Fig.5 below.
This instruction takes about 2 hours to install ROS. Elapsed time may vary depending on network environment. Please start your environment setting earlier.



1.4 Installation

First, make sure your Debian package index is up-to-date:

```
sudo apt update
```

There are many different libraries and tools in ROS. We provided four default configurations to get you started. You can also install ROS packages individually.

In case of problems with the next step, you can use following repositories instead of the ones mentioned above ros-shadow-fixed

Desktop-Full Install: (Recommended) : ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators and 2D/3D perception

```
sudo apt install ros-melodic-desktop-full
```

or click here

PC

Desktop Install: ROS, rqt, rviz, and robot-generic libraries

```
sudo apt install ros-melodic-desktop
```

or click here

RPi

ROS-Base: (Bare Bones) ROS package, build, and communication libraries. No GUI tools.

```
sudo apt install ros-melodic-ros-base
```

or click here

Individual Package: You can also install a specific ROS package (replace underscores with dashes of the package name):

```
sudo apt install ros-melodic-PACKAGE
```

e.g.

```
sudo apt install ros-melodic-slam-gmapping
```

To find available packages, use:

```
apt search ros-melodic
```

Fig.3 ROS installation for different processor

(2-3) Then follow the "3. Create a ROS Workspace" from http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment to finish all the ROS setting.

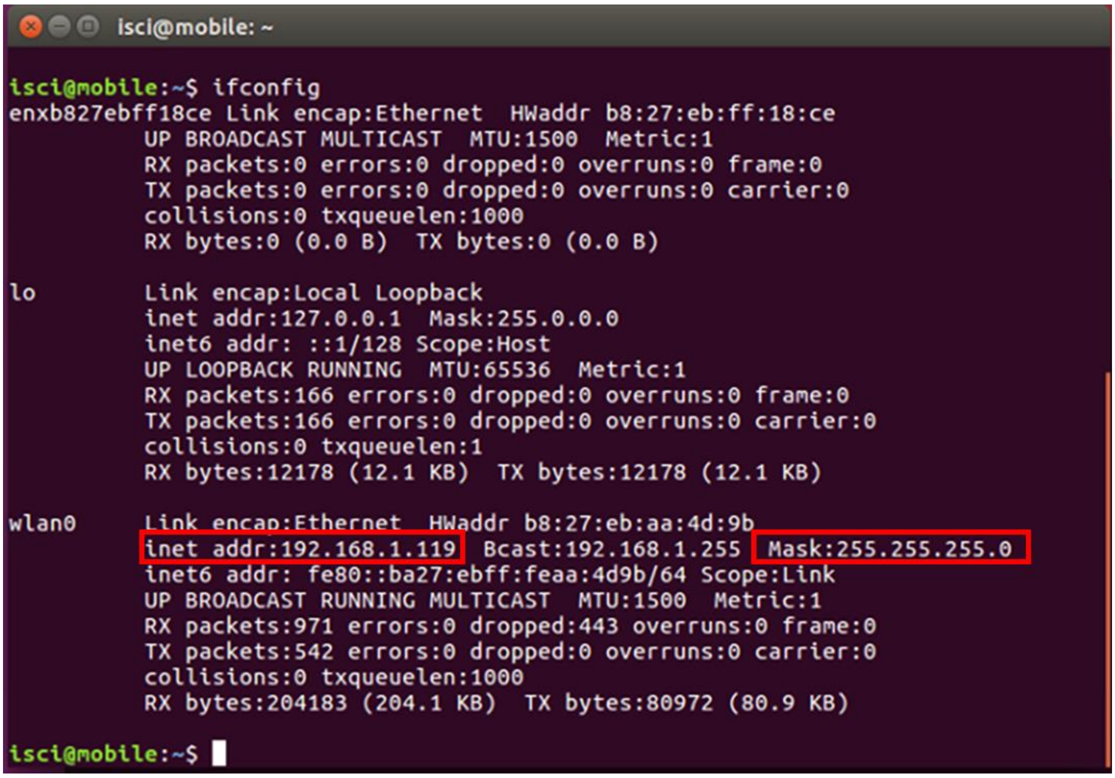
3. Setting *ssh* between Raspberry Pi and PC

(3-1) Follow the instruction down below to install *ssh* in both your PC and Raspberry Pi.

```
$ sudo apt-get update
$ sudo apt-get install openssh-server openssh-client
$ sudo service ssh start
$ systemctl enable ssh.socket
$ sudo dpkg-reconfigure openssh-server
$ sudo service ssh restart
```

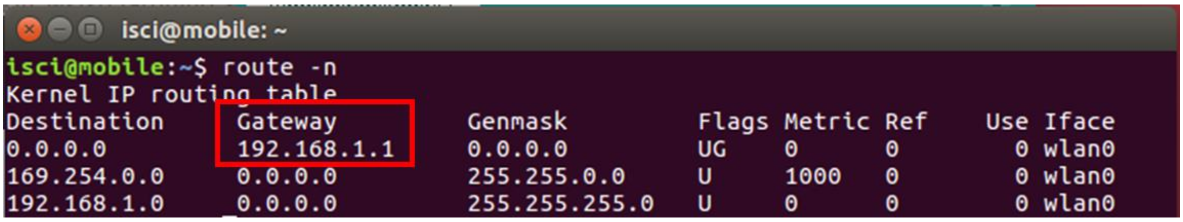
(3-2) Find Raspberry Pi's address, netmask and gateway by using

```
$ ifconfig
$ route -n
```



```
isci@mobile: ~
isci@mobile:~$ ifconfig
enxb827ebff18ce Link encap:Ethernet  HWaddr b8:27:eb:ff:18:ce
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:166 errors:0 dropped:0 overruns:0 frame:0
          TX packets:166 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:12178 (12.1 KB)  TX bytes:12178 (12.1 KB)

wlan0     Link encap:Ethernet  HWaddr b8:27:eb:aa:4d:9b
          inet addr:192.168.1.119  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ba27:ebff:feaa:4d9b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:971 errors:0 dropped:443 overruns:0 frame:0
          TX packets:542 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:204183 (204.1 KB)  TX bytes:80972 (80.9 KB)

isci@mobile:~$
```

Fig.4 Raspberry Pi's network information by using *ifconfig*



```
isci@mobile: ~
isci@mobile:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 wlan0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 wlan0
```

Fig.5 Raspberry Pi's network information by using *route –n*

(3-3) Setting static IP in Graphical User Interface.
You will need to set up address <rpi's inet addr>, netmask <rpi's Mask>, gateway <rpi's Gateway> and dns server <8.8.8.8> in GUI.

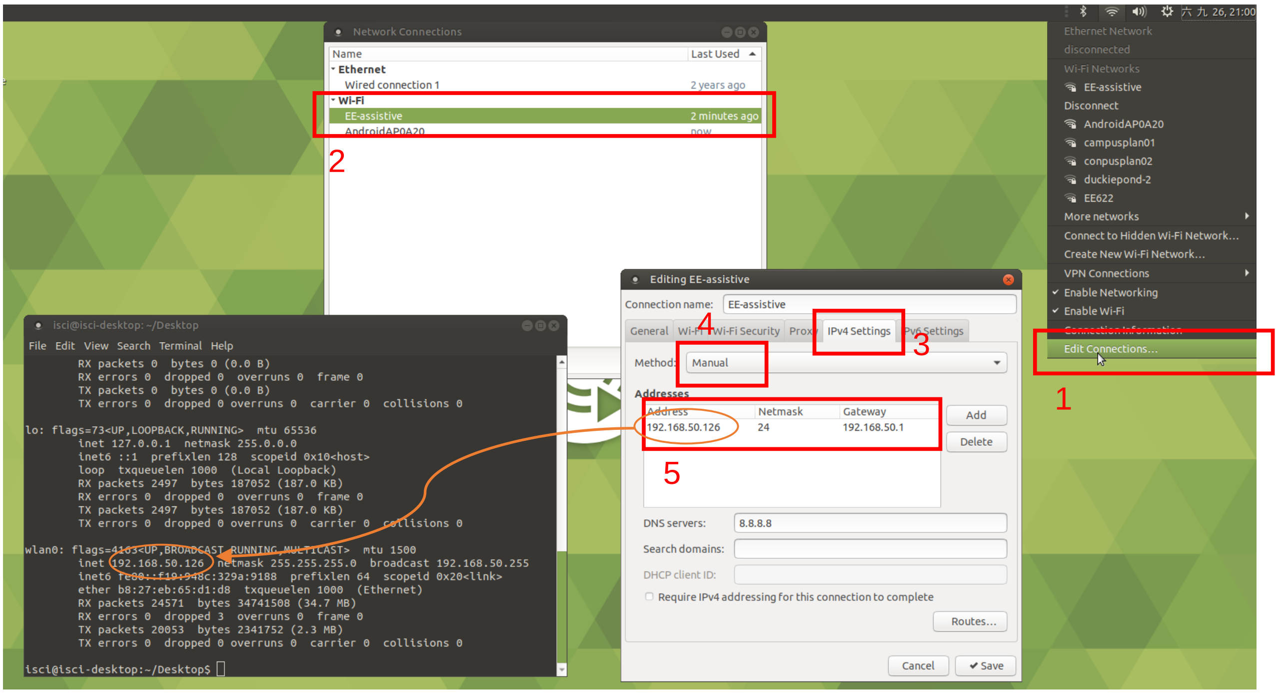


Fig.6 Steps for setting static IP

(3-4) Using *ssh* to remote connect with PC.

$ ssh <user_name>@<ip_addr>

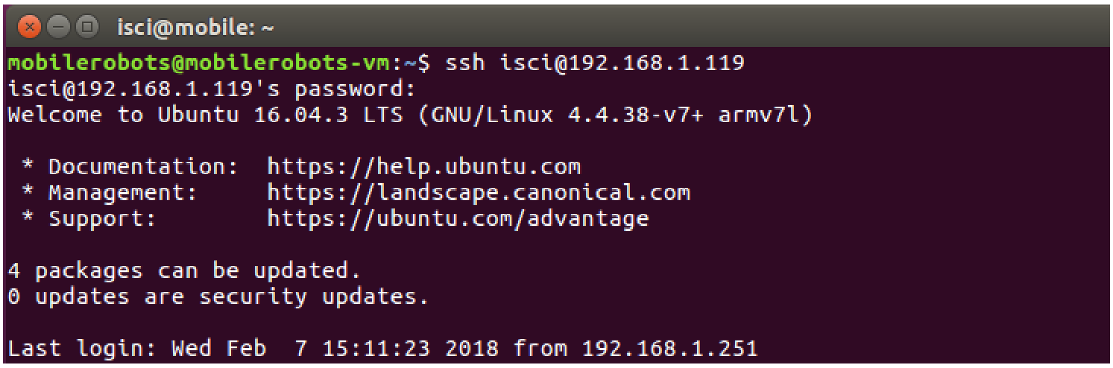


Fig.7 *ssh* command for remote connection

B. ROS

1. ROS file system level structure

Reference: http://wiki.ros.org/ROS/Concepts

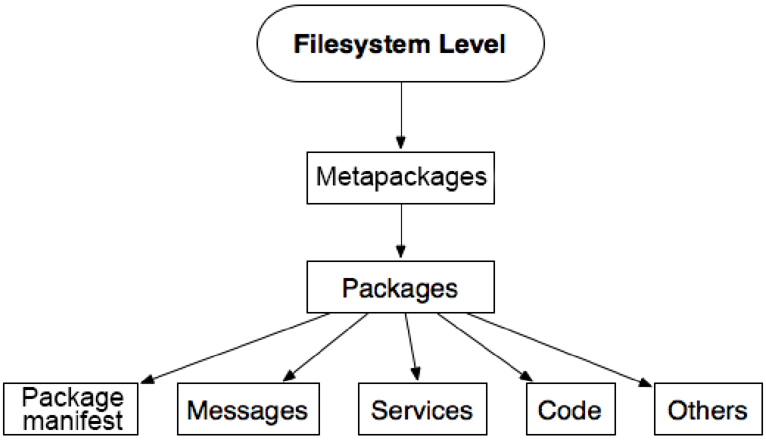


Fig.8 ROS file system level structure

(1-1) Packages:

Packages are the main unit for organizing software in ROS. A package may contain ROS runtime processes (nodes), a ROS-dependent library, datasets, configuration files, or anything else that is usefully organized together. Packages are the most atomic build item and release item in ROS. Meaning that the most granular thing you can build, and release is a package.

2. ROS computation graph level

Reference: http://wiki.ros.org/ROS/Concepts

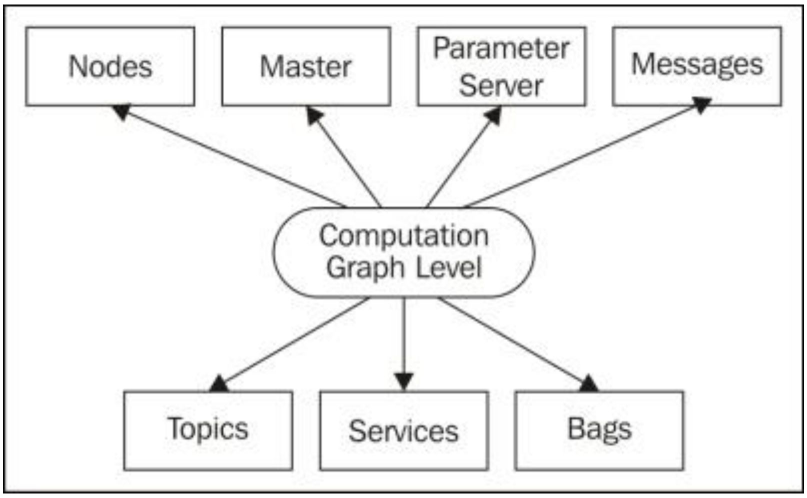


Fig.9 ROS computation graph level

(2-1) Nodes:

Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one Node provides a graphical view of the system, and so on. A ROS node is written with the use of a ROS client library, such as *roscpp* or *rospy*.

(2-2) Master:

The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

(2-3) Topics:

Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each other's existence. The idea is to decouple the production of information from its consumption. Logically, one can think of a topic as a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type.

(2-4) Messages:

Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).

(2-5) Parameter Server:

The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.

3. Create package file

Create a ROS package by using *catkin_create_pkg* command.

```
$ cd catkin_ws/src
$ catkin_create_pkg [package_name] [dependency1] [dependency2]
```

Common dependency such as *roscpp, std_msgs, actionlib, actionlib_msgs*…etc.

4. ROS nodes communication

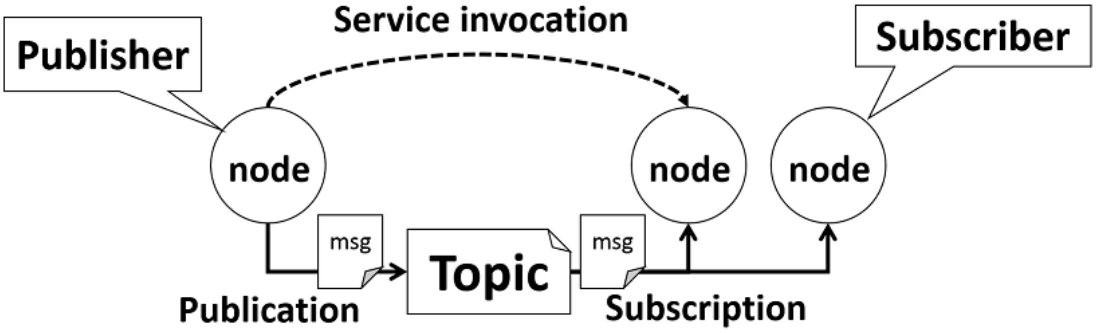


Fig.10 ROS node Communication

5. Publisher and Subscriber

Reference: *Mastering ROS for Robotics Programming, p.29-31*

(5-1) This example code will publish an integer value on a topic called */numbers*.

Demo_topic_publisher.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>
int main(int argc, char **argv)
{
    ros::init(argc, argv, "demo_topic_publisher");
    ros::NodeHandle node_obj;
    ros::Publisher number_publisher =
    node_obj.advertise<std_msgs::Int32>("/numbers", 10);
    ros::Rate loop_rate(10);
    int number_count = 0;
    while (ros::ok())
    {
        std_msgs::Int32 msg;
        msg.data = number_count;
        ROS_INFO("%d", msg.data);
        number_publisher.publish(msg);
        ros::spinOnce();
        loop_rate.sleep();
        ++number_count;
    }
    return 0;
```

```
}
```

Here is some detailed explanation of the example code:

```
6  ros::init(argc, argv, “demo_topic_publisher”);
```

This code will initialize a ROS node with a name. It should be noted that the ROS node should be unique. This line is mandatory for all ROS C++ nodes.

```
7  ros::NodeHandle node_obj;
```

This will create a *Nodehandle* object, which is used to communicate with the ROS system.

```
8  ros::Publisher number_publisher =
   node_obj.advertise<std_msgs::Int32>(“/numbers”, 10);
```

This will create a topic publisher and name the topic */number* with a message type *std_msgs::Int32*.

The second argument is the buffer size. It indicates that how many messages need to be put in a buffer before sending.

```
9  ros::Rate loop_rate(10);
```

This is used to set the frequency of sending data.

```
11  ros::ok()
```

This function returns zero when there is an interrupt like *Ctrl+C*.

```
16  number_publisher.publish(msg);
```

This will publish the message to the topic */numbers*.

(5-2) This example code is the definition of the subscriber node:

Demo_topic_subscriber.cpp

```
#include “ros/ros.h”
#include “std_msgs/Int32.h”
#include <iostream>
void number_callback(const std_msgs::Int32::Constptr& msg)
{
    ROS_INFO(“Received [%d]”, msg->data);
}
int main(int argc, char **argv)
{
    ros::init(argc, argv, “demo_topic_publisher”);
```

```
    ros::NodeHandle node_obj;
    ros::Subscriber number_subscriber =
    node_obj.subscribe("/numbers", 10, number_callback);
    ros::spin();
    return 0;
}
```

Here is some detailed explanation of the example code:

```
void number_callback(const std_msgs::Int32::Constptr& msg)
{
    ROS_INFO("Received [%d]", msg->data);
}
```

This is a callback function that will execute whenever a data comes to the */numbers* topic. Whenever a data reaches this topic, the function will call and extract the value and print it on the console.

```
ros::Subscriber number_subscriber =
node_obj.subscribe("/numbers", 10, number_callback);
```

This is the subscriber and here, we are giving the topic name needed to subscribe, buffer size, and the callback function. We are subscribing */number* topic and we have already seen the callback function in the preceding section.

6. CMakeLists.txt:

Reference: http://wiki.ros.org/catkin/CMakeLists.txt

(6-1) The file **CMakeLists.txt** is the input to the CMake build system for building software packages. Any CMake-compliant package contains one or more CMakeLists.txt file that describe how to build the code and where to install it to. The CMakeLists.txt file used for *catkin_make* project.

(6-2) Building the nodes for example. We have to edit the *CMakeLists.txt* file in the package to compile and build the source code. The following example code is responsible for building those two nodes above.

```
include_directories(
    include
    ${catkin_INCLUDE_DIRS}
    ${Boost_INCLUDE_DIRS}
)

# This will create executables of the nodes
add_executable(demo_topic_publisher src/demo_topic_publisher.cpp)
add_executable(demo_topic_subscriber src/demo_topic_subscriber.cpp)
```

```

# This will generate message header file before building the target
add_dependencies(demo_topic_publisher
mastering_ros_demo_pkg_generate_message_cpp)
add_dependencies(demo_topic_subscriber
mastering_ros_demo_pkg_generate_message_cpp)

# This will link executables to the appropriate libraries
target_link_libraries(demo_topic_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_topic_subscriber ${catkin_LIBRARIES})
```

Build mastering_ros_demo_package as follow:

```
$ cd ~/catkin_ws
$ catkin_make mastering_ros_demo_package
```

7. roslaunch:

Reference: *Mastering ROS for Robotics Programming, page 48*

(7-1) The *launch* files in ROS are a very useful feature for launching more than one node. It is difficult if we run each node in a terminal one by one. Instead of that, we can write all nodes inside a *XML* based file called *launch* files and using a command called *roslaunch*, we can parse this file and launch the nodes.

(7-2) Create a *launch* folder to keep the launch files

```
$ cd ~/catkin_ws/src/<ros_package>
$ mkdir launch
```

(7-3) The example launch file will launch two ROS nodes that are publishing and subscribing an integer value.

Demo_topic.launch

```
<launch>
    <node name="publisher_node" pkg="mastering_ros_demo_pkg"
type="demo_topic_publisher" output="screen" />

    <node name="subscriber_node" pkg="mastering_ros_demo_pkg"
type="demo_topic_subscriber" output="screen" />
</launch>
```

(7-4) After creating the launch file, you can launch it by using the following command:

```
$ roslaunch mastering_ros_demo_pkg demo_topic.launch
```

C. *rosserial* package

1. Arduino IDE setup

(1-1) Download Arduino IDE on your PC and Raspberry Pi by typing instructions below in Terminal. The first two instructions will take some time.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install arduino
```

(1-2) Set Serial Port Permission

Reference: https://www.arduino.cc/en/Guide/Linux

Connect your Arduino UNO board in the device you are setting in, if you're setting Arduino IDE in Raspberry pi then connect Arduino UNO with Raspberry Pi, so does your PC.

Open Terminal and type:

```
$ ls -l /dev/ttyACM*
```

You will get something like:

crw-rw---- 1 root dialout 188, 0 5 apr 23.01 ttyACM0

The "0" at the end of ACM might be a different number, or multiple entries might be returned. The data we need is "dialout" (is the group owner of the file).

Then add our user to the group by typing follow instruction in Terminal:

```
$ sudo usermod -a -G dialout <username>
```

Where <username> is your linux user name. You will need to log out and log in again for this change to take effect.

(1-3) You can test whether your Arduino IDE is work or not by typing *arduino* in Terminal. And after you can open Arduino sketch successfully, you can find a new folder called "sketchbook" in /home.

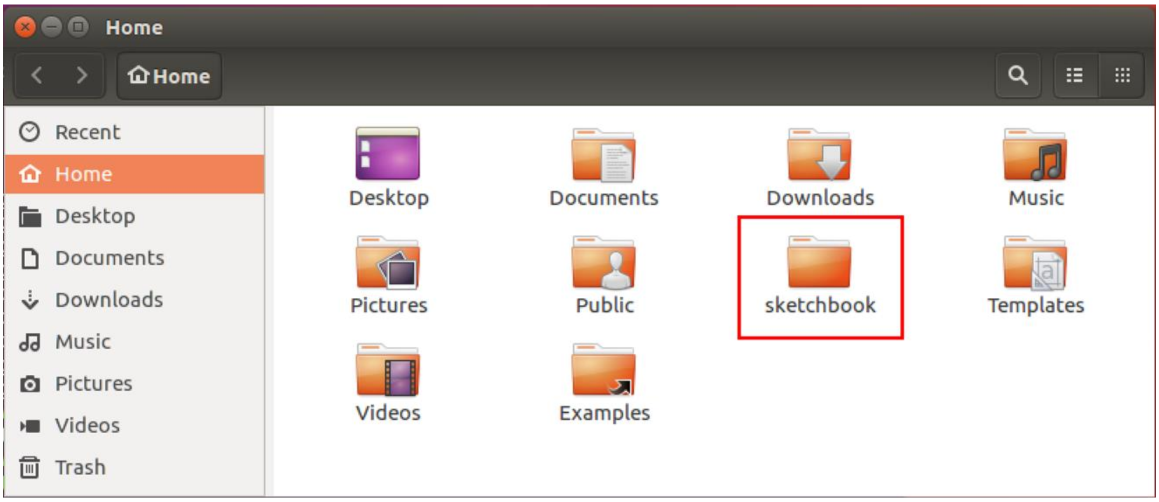


Fig.11 Find Arduino sketchbook folder

2. Installing the Software

Reference: http://wiki.ros.org/rosserial_arduino/Tutorials

(3-1) Installing on the ROS workstation

```
There are 2 options of how to install related libraries:
$ sudo apt-get install ros-melodic-rosserial-arduino
$ sudo apt-get install ros-melodic-rosserial
or
$ cd <ws>/src
$ git clone https://github.com/ros-drivers/rosserial.git
$ cd <ws>
$ catkin_make
$ catkin_make install
```

(3-2) Install *ros_lib* into the Arduino Environment

```
$ cd sketchbook/libraries
$ rm -rf ros_lib
$ rosrun rosserial_arduino make_libraries.py .
```

Notice that there is a dot (.) behind the python file. After restarting your IDE, you should see *ros_lib* listed under File>examples or File>sketchbook.

3. Create a publisher by using *rosserial*

Reference: http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World

(4-1) This example code is in the File>example>ros_lib>HelloWorld. This code will create a node publishes "Hello World" from Arduino.

Hello World.ino

```
#include <ros.h>
#include <std_msgs/String.h>

ros::NodeHandle nh;

std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);

char hello[13] = "hello world";

void setup()
{
    nh.initNode();
    nh.advertise(chatter);
}
```

```
void loop()
{
    str_msg.data = hello;
    chatter.publish( &str_msg);
    nh.spinOnce();
    delay(1000);
}
```

(4-2) To upload the code to your Arduino, use the upload function within the Arduino IDE. This is no different from uploading any other sketch.

(4-3) Running the code

Open a new Terminal and type:

```
$ roscore
```

Other new Terminal and type:

```
$ rosrun rosserial_python serial_node.py /dev/ttyACM0
```

Another new Terminal and type:

```
$ rostopic echo /chatter
```

```
isci@mobile:~$ rostopic echo chatter
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
data: "hello world!"
---
```

4. Create a subscriber by using *rosserial*

Reference: http://wiki.ros.org/rosserial_arduino/Tutorials/Blink

(5-1) This example code is in the File>example>ros_lib>Blink. This code will create a subscriber and the LED on the Arduino will toggle every time receive a empty message from Raspberry Pi.

Blink.ino

```
#include <ros.h>
#include <std_msgs/Empty.h>

ros::NodeHandle nh;

void messageCb( const std_msgs::Empty& toggle_msg){
   digitalWrite(13, HIGH-digitalRead(13));      // blink the led
}

ros::Subscriber<std_msgs::Empty> sub("toggle_led", &messageCb );

void setup()
{
   pinMode(13, OUTPUT);
   nh.initNode();
   nh.subscribe(sub);
}

void loop()
{
   nh.spinOnce();
   delay(1);
}
```

(5-2) To upload the code to your Arduino, use the upload function within the Arduino IDE.

(5-3) Running the code

Open a new Terminal and type:

```
$ roscore
```

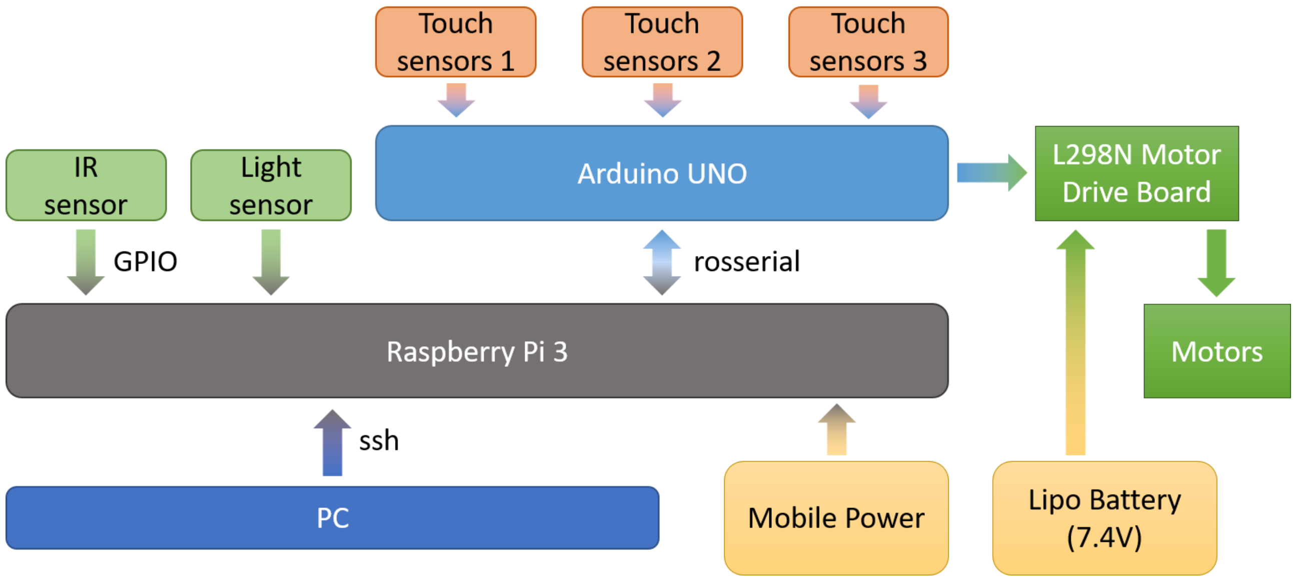
Other new Terminal and type:

```
$ rosrun rosserial_python serial_node.py /dev/ttyACM0
```

Another new Terminal and type:

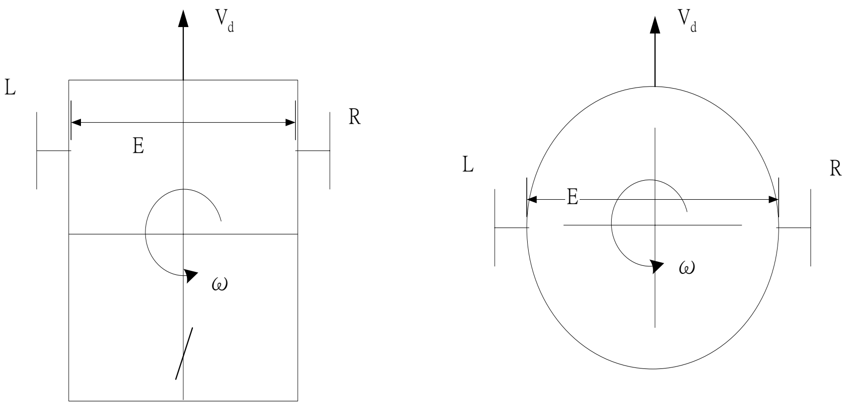
```
$ rostopic pub toggle_led std_msgs/Empty --once
```

- Hardware architecture



Chapter 3 Motion Control of Mobile Robots

Body(Motion platform) + Drive system(Battery + Motor + Wheels) + functionalities



$$\begin{cases} \omega = \dfrac{V_L - V_R}{E} \\ V_d = \dfrac{V_L + V_R}{2} \end{cases}$$

Fig. 3-1 Uni-cycle modeled mobile robots

DC motors:

- DC motor fundamentals
- A motor model + characteristic
- Gearing
- Selection of DC motors
- Servo drivers: DC servo(PWM+L293) + Amplifiers
- Position/velocity servo control
- Motion controller design: LM629, HCTL 1100, DSP, Embedded Microcontroller …

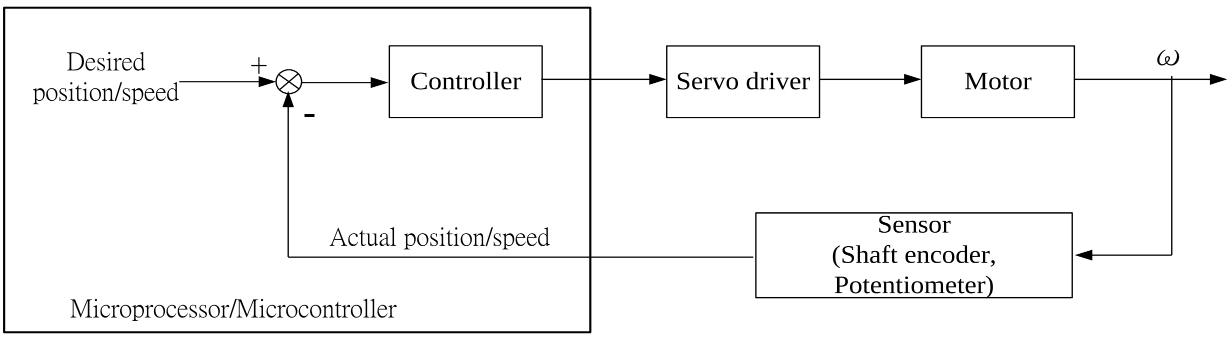


Fig. 3-2 Motor control system

AC motor (higher speed, AC supply)

DC motor (lower speed, DC supply): small, cheap, reasonable efficient, easy to use

Rotor: loops of wire mounted on a rotating shaft(armature)

Stator: permanent magnetic

When provided with a constant voltage, a motor draw current proportional to how much work it is doing. When there is no resistance to its rotation, the motor draws the least amount of current. When there is so much resistance as to cause the motor to stall, it draws the maximum amount of current (stall current).

Stall current:

The maximum amount of current that a motor can draw at its specified voltage.

The more current going through a motor, the more rotation force(torque, 扭矩, 轉矩) is produced at the motor’s shaft.

Stall torque:

The amount of rotation force produced when the motor is stalled at its recommended operation voltage, drawing the maximum stall current at this voltage.

Power:

Rotational velocity x torque

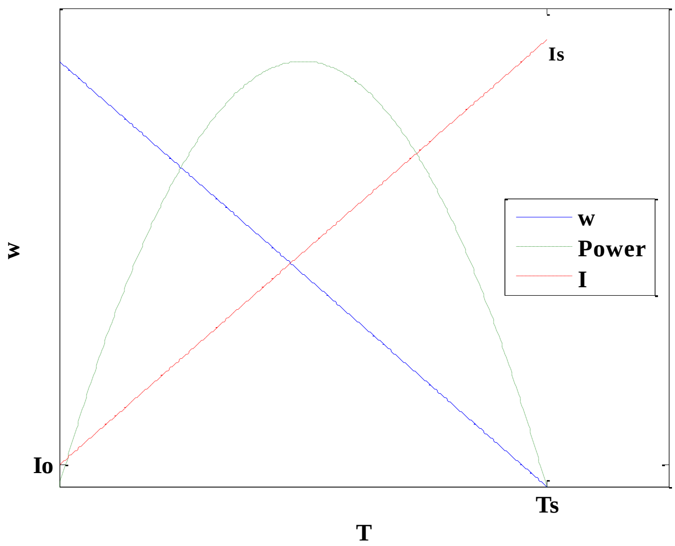


Fig.3-3 Characteristics of a motor

DC motors: normally run at high speed and low torque

→ Reduction gears are used to increase torque and reduce speed.

Fig. 3-4 Basic component of a gear-head motor

Most DC motors have two electronic terminals. Applying voltage across these terminals causes the motors to spin in one direction. While reversing polarity voltage will cause the motor to spin in other direction. Polarity of the voltage determines the rotation direction; amplitude of the voltage determines motor speed.

1)Servo motors :

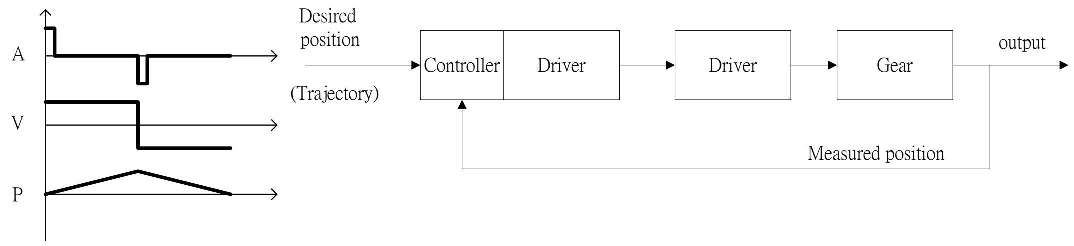


Fig. 3-5 Position, velocity and acceleration of a servo motor

The word servo refers to the system's capability to self-regulate its behavior, i.e. it measures its position and compensate for external loads when responding to a control signal (trajectory).

2) Three-wire DC servo motors (RC Servo)

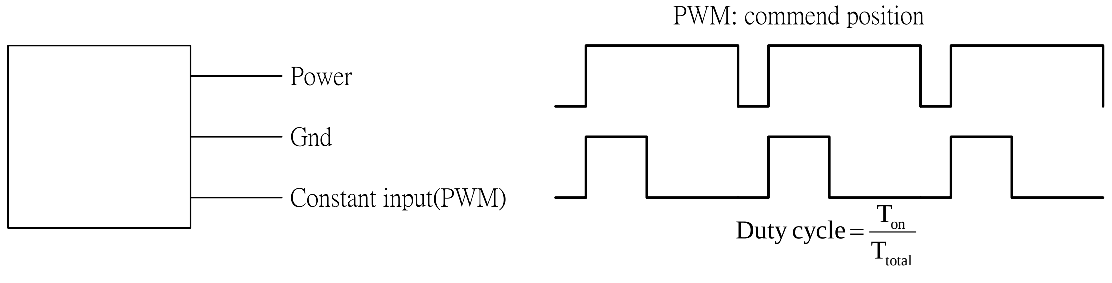


Fig. 3-6 RC Servo: three wire DC motor

The PWM is used to specify the command position. The module include a DC motor, a gear train, limit stops, potentiometer for position feedback, and a position control IC. Most used in toys and model airplanes for steering.

Components:

a. A gear head.

b. A position sensor on the shaft.

c. An integration circuit for control.

Characteristics:

a. Rotate < 360˚

b. If remove the position limit stops, which can be removed for mobile robot use.

3) Operation principle of a DC motor**:**

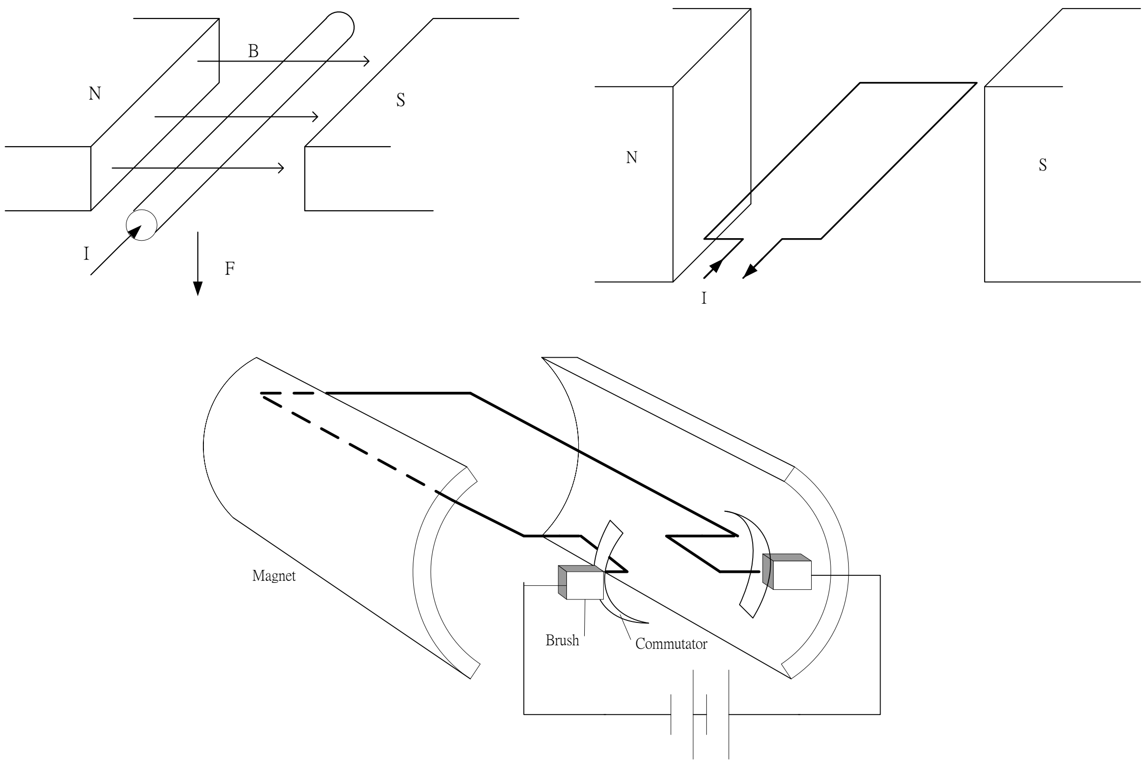


Fig. 3-7 Operation principle of a DC motor

Brush type:

If the DC current is commutated mechanically with brushes, the commutator segments at the ends of the rotating rotor coil physically slides against the stationary brushes that are connected to the motors terminals on the outside of the case.

Brushless type:

If the DC motor current is conversed into AC current in the rotor electronically, with position sensors and a microprocessor controller, then no brushes are needed.

- longer life.
- more expensive, RF interference.

4) A motor model & DC motor characteristics:

The armature (rotor) coil is essentially an inductor with a resistance R. As armature rotates, the brushes impart alternating current in coil.

$$v = L\frac{di}{dt},\ L: \text{inductance}$$

The induced voltage oppose applied voltage(Lentz law).

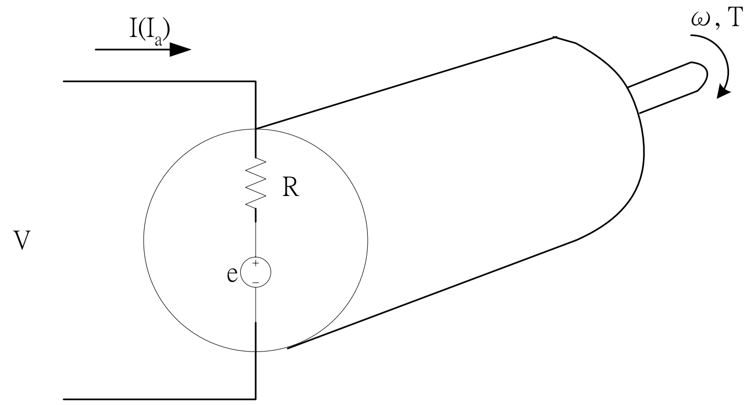


Fig. 3-8 A motor model

The faster the motor turns, the more of the current switches direction and so the larger the induced voltage.

→ limit the current through the resistance R (through the coil)

→ limit the torque of the motor.

→- w↑, T↓

$$T: \text{torque} \quad \text{unit}: \begin{cases} \text{N-m} \\ \text{gf -cm} \\ \text{lb-in} \\ \text{oz-in} \end{cases}$$

$mNm = 10^{-3}\,Nm$

$1lb - in = 0.113Nm$

$1oz - in = 7.06mNm$

$1lb = 445N = 16oz, 1in = 2.54cm$

$$V = IR + e \qquad \text{e: back emf (induced voltage)}$$

$$e = k_e\omega \qquad \text{k}_\text{e}: \text{back-emf constant}$$

Negative feedback of back-emf causes motor to reach steady-state operating point of speed and voltage as determined by applied voltage and load.

Note: $I_{\max} = \frac{V}{R}$, occurs when ω= 0, this is the starting current or stall current(I_s).

$$V = IR + k_e \omega$$

$$T = k_t I, \qquad k_t : \text{torque constant}$$

$$P_m = P_e - I^2 R$$

$$T\omega = VI - I^2 R$$

$$(k_t I)\omega = (IR + k_e \omega)I - I^2 R \Rightarrow k_t = k_e = k$$

$$\omega = \frac{V - IR}{k} = \frac{-(\frac{T}{k})R}{k} + \frac{V}{k} = -\frac{TR}{k^2} + \frac{V}{k}$$

$$P_m = T\omega = \frac{-RT^2}{k^2} + \frac{V}{k}T$$

Mechanical power has quadratic dependence on torque.

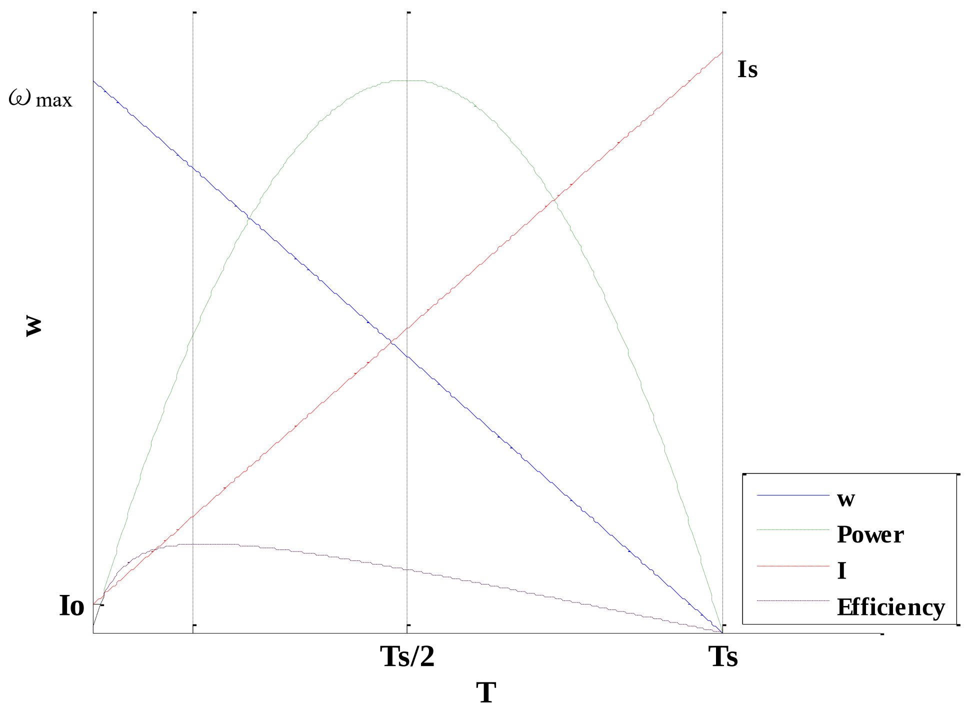


Fig. 3-9 Important parameters of a DC motor

- No load speed $\omega_0 = \omega_{max}$, the speed, at a given voltage, at which the torque is 0 (T = 0)

$$\omega_{max} = \frac{V}{k}$$

- No load current, at no load condition, I_0 is required to overcome motor friction and windage.

Maximum torque when motor stalled ($\omega = 0$, emf = 0)

$$I_s = I_{max} = \frac{V}{R}, T_s = kI = \frac{Vk}{R}$$, occurs at $\omega = 0, emf = 0$

$$\frac{dP_m}{dT} = 0 \Rightarrow \frac{-2R}{k^2}T + \frac{V}{k} = 0$$

Torque at max power: $$\Rightarrow T = \frac{kV}{2R} = \frac{1}{2}T_s = \frac{1}{2}T_{max}$$

ω at max power: substitute $T = \frac{kV}{2R}$ into $\omega = -\frac{R}{k^2}T + \frac{V}{R}$

$$\Rightarrow \omega = -\frac{R}{k^2}(\frac{kV}{2R}) + \frac{V}{k} = \frac{V}{2k} = \frac{1}{2}\omega_{max}$$, ω_{max}: ω at T=0

$$P_{max} = (\frac{1}{2}T_{max})(\frac{1}{2}\omega_{max}) = \frac{1}{4}T_{max}\omega_{max}$$

Efficiency: The ratio of mechanical power output to electrical power input.

$$P_m = \eta P_e, \eta_{max} = (1 - \sqrt{\frac{I_o}{I_s}})^2$$

- max efficiency ≠ max power

  (We would like to drive the motor at max efficiency.)

- Select an oversized motor so that it can run at an efficient operation point while supply enough torque.

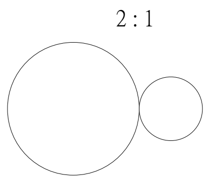
Ex.

Gear factor = 2

=> no load speed = half($\frac{1}{2}$), while doubling stall torque.

Power maintain constant => $P_e = T\omega$ (more loss due to gearing)

$$T_1 \times \frac{360°}{s} = T_2 \times \frac{720°}{s} \Rightarrow T_2 = \frac{1}{2}T_1$$



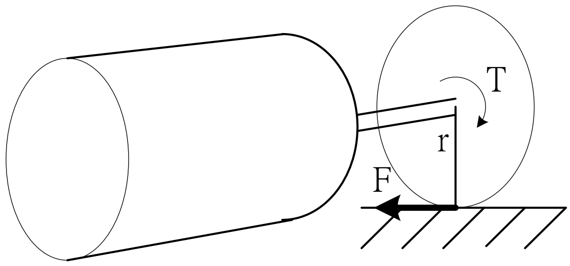


Fig. 3-10 Generation of torque using a DC motor

$$F = F_n \times \mu$$

$$F \times r = T \times r$$

Ex:

A DC motor : internal resistance R=2Ω, running at full load on a 7.2V battery, a current of 500mA is drawn.

(a) e, back emf?

$$V = e + iR$$

$$7.2 = e + 0.5 \times 2 \Rightarrow e = 6.2$$

(b) power delivered to the motor (Pe)

$$P_e = i \cdot V = 0.5 \cdot 7.2 = 3.6w$$

(c) power dissipation in motor:

$$P_d = i^2 R = 0.5^2 \times 2 = 0.5w$$

(d) what is the mechanical power developed?

$$3.6 - 0.5 = 3.1w$$

$$\eta = \frac{3.1}{3.6} \times 100\% = 86\%$$

$$(emf \cdot i = 6.2 \times 0.5 = 3.1w)$$

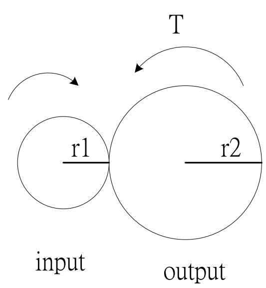
$$T = F \cdot r$$

From conservation of work:

$$W = T \cdot (\text{angular displacement})$$

$$T_{large} \times 360° = T_{small} \times 1080°$$

$$\Rightarrow \frac{T_{large}}{T_{small}} = \frac{1080°}{360°} = 3$$



$$\frac{r_1}{r_2} = \frac{1}{3}$$

Fig. 3-11 Gear ratio

Worm gear can attain large gear down in a small space.

➔ use screw mechanism to generate motion at right angle to shaft

5) Selection of DC motors:

Application data:

Parameter	Symbol	Unit	Value
Required torque	M	mNm	3
Required speed	N	Rpm	5500
Available supply voltage	U	V_{DC}	20
Available supply current	I	A	0.5
Available supply space	Φ	mm	(Φ)25*(L)50

Power required: the motor is expected to deliver

$$P_r = T\omega = M \cdot n \frac{2\pi}{60 \times 1000} = 3.5500 \frac{\pi}{30x1000} = 1.73w$$

A motor selected will deliver a least 1.5 to 2 times the power required.

$P_{2\cdot\max} \geq 2P_r, U_N \geq U$

Series 2233T024S: $U_N = 24V, P_{2\max} = 2.47w$

Should the available supply voltage be lower than the nominal voltage of the selected DC motor, the $P_{2,\max}$ from the motor catalogue should be corrected:

$$P_{2\max} = \frac{R}{4}(\frac{U}{R} - I_o)^2 \Rightarrow P_{2\max}(20V) = \frac{57}{4}(\frac{20}{57} - 0.005)^2 = 1.7w$$

R: terminal resistance

I_0: no-load current

Optimizing the pre-selection:

1). The required speed (n) has to be higher than half the no-load speed (ω_0) at nominal voltage.

2). The load torque (M) has to be less than half the stall torque (Ts)

$$(1)\ \ n \geq \frac{\omega_o}{2} \qquad (2)\ \ M \leq \frac{T_s}{2}$$

From data sheet, ω_0=8800 rpm, Ts=10.70mNm

$$\begin{cases} n(5500rmp) \geq \dfrac{\omega_o}{2}(\dfrac{8800}{2} = 4400rpm) \\ M(3) < \dfrac{T_s}{2}(\dfrac{10.7}{2} = 5.35mNm) \end{cases}$$

-Performance characteristics at normal voltage (24VDC)

Stall current $I_0 = \dfrac{U_n}{R} = \dfrac{24}{57} = 0.421A$

Torque at max efficiency:

$$\begin{aligned} T_{opt} &= \sqrt{T_s \cdot T_R} \\ &= \sqrt{10.7 * 0.13} = 1.18mNm \end{aligned}$$, T_R: friction torque

-Main parameters at 20VDC

1). No-load speed n_0 at 20V DC

$$n_0 = \frac{U - (I_0 xR)}{k_E} x1000$$

$$= \frac{20 - (0.05x57)}{2.690} x1000 = 7315rpm$$

2).Stall current I_H

$$I_H = \frac{U}{R} = \frac{20}{57} = 0.351A$$

3). Ttall torque: $T_H = K_m(I_H - I_o)$

$$T_H = (25.70\frac{mNm}{A})(0.351 - 0.005) = 8.91mNm$$

4) Output power: $P_{2\max}$

$$P_{2,\max} = \frac{R}{4}(\frac{U_N}{R} - I_0)^2$$

$$P_{2,\max}(20V) = \frac{57}{4}(\frac{20}{57} - 0.005)^2 = 1.7W$$

6) Servo drivers:

1) Linear servo amplifier

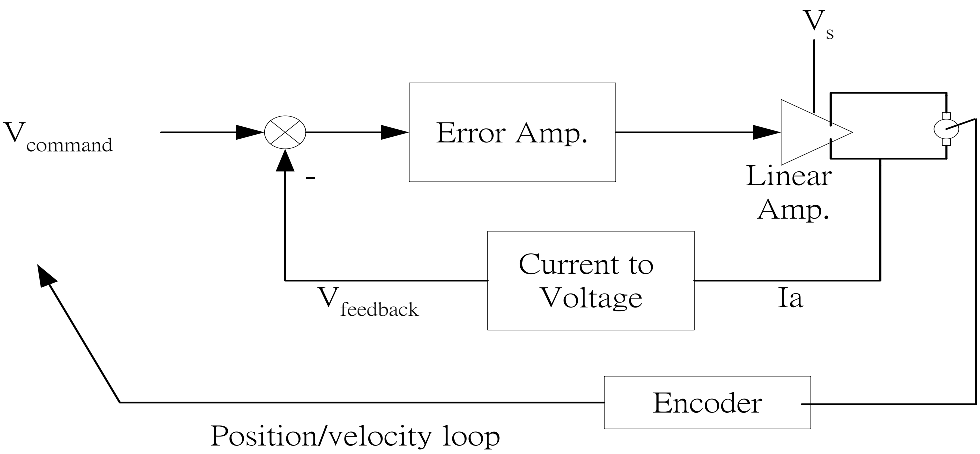


Fig. 3-12 Linear servo drivers

$V_{feedback}$: Sensing the armature current and converting to an analog voltage signal; a voltage representation of the actual current.

$V_{command}$:A voltage representation of a desired motor current.

Error amplifier: responsible for the stability and performance

2) PWM switching servo amplifier:

- Pulse-Width Modulation module

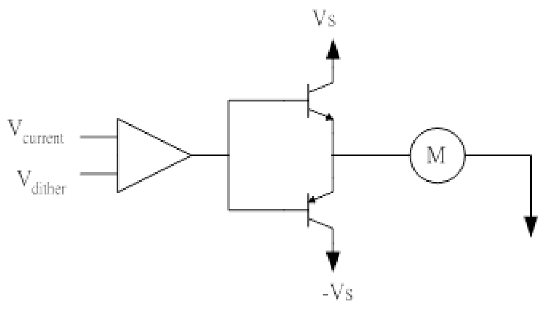


Fig. 3-13 PWM switching servo amplifier

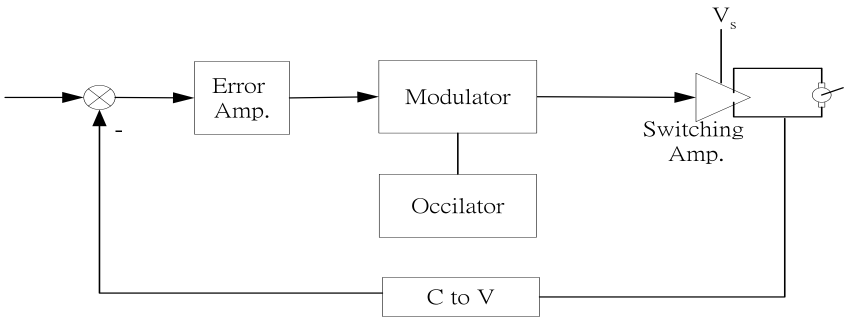


Fig. 3-14 Amplification of PWM signal

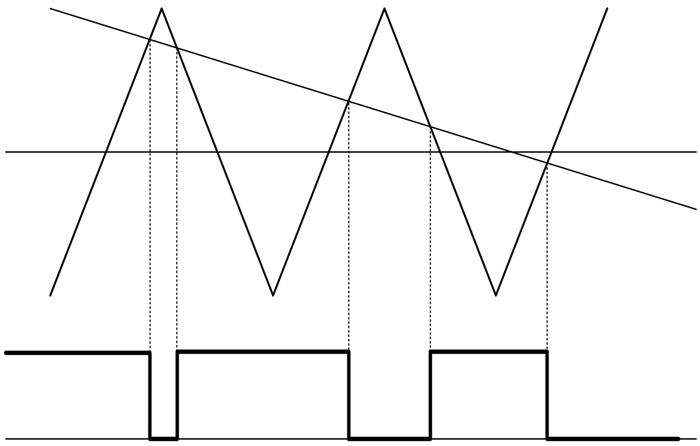


Fig. 3-15 Generation of PWM signal

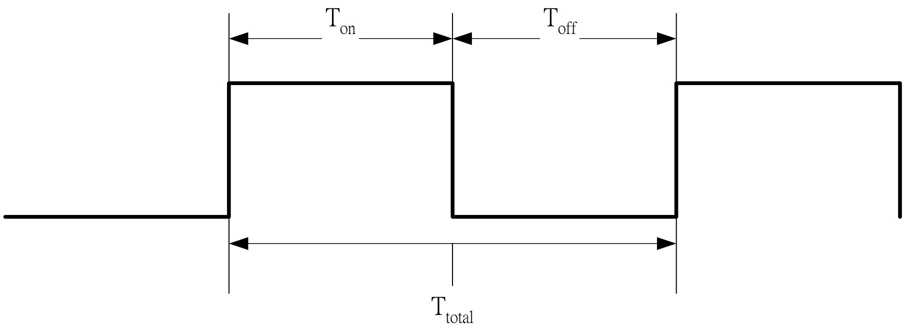


Fig. 3-16 The duty cycle

T_{total} = 1ms(BS2)

Duty cycle=T_{on}/T_{off}

2) H-bridge Power Stage:

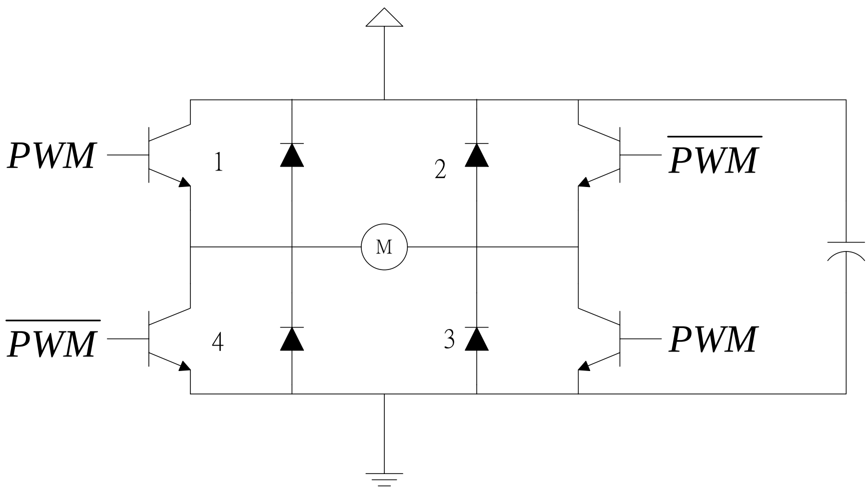


Fig. 3-17 H-bridge power stage

-A constant frequency variable duty cycle pulse train is generated at the comparator output. The duty cycle of the pulse train indicates the level of the correction (control) voltage. As the correction voltage becomes more positive, the duty cycle increases. Conversely, the duty cycle decreases. The transistors are always operated as switches, they are either fully on (saturated) or fully off(Cut-off). At any time, either Q1, Q3 or Q2, Q4 are on.

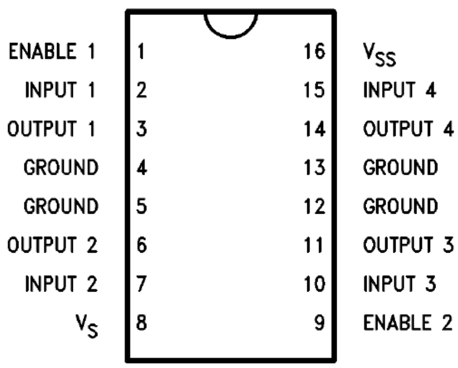
-The average value of the waveform is proportional to the correction voltage and is amplified to a level suitable for driving the motor.

-Ideally, the transistors are either in a state of current and no voltage or voltage and no current. This gives us zero-power operation of PWM, or a dissipationless power stage.

-Switching frequency of PWM ≒2KHz ~30KHz is well above motor's electrical bandwidth. So the motor's inductance can act as an effective filter to the supplied voltage pulse train.

3) Servo Driver components

L293 -> 1A , 2 full H-bridge

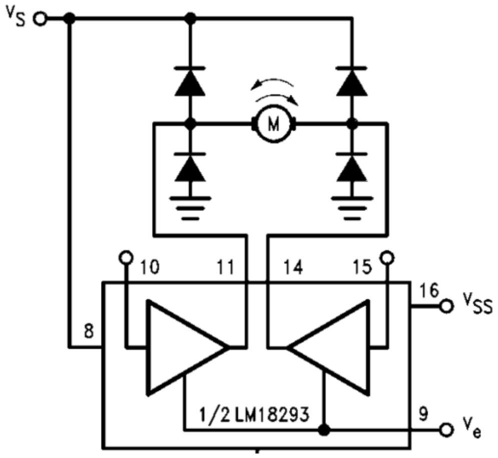


Enable 1 activates outputs 1 & 2

Enable 2 activates outputs 3 & 4

TL/H/8706-2

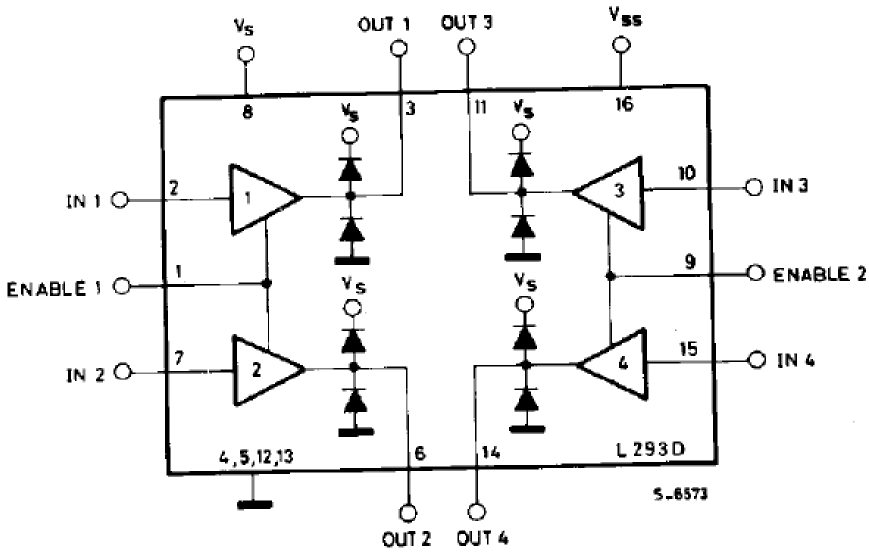
Bidirectional DC motor control



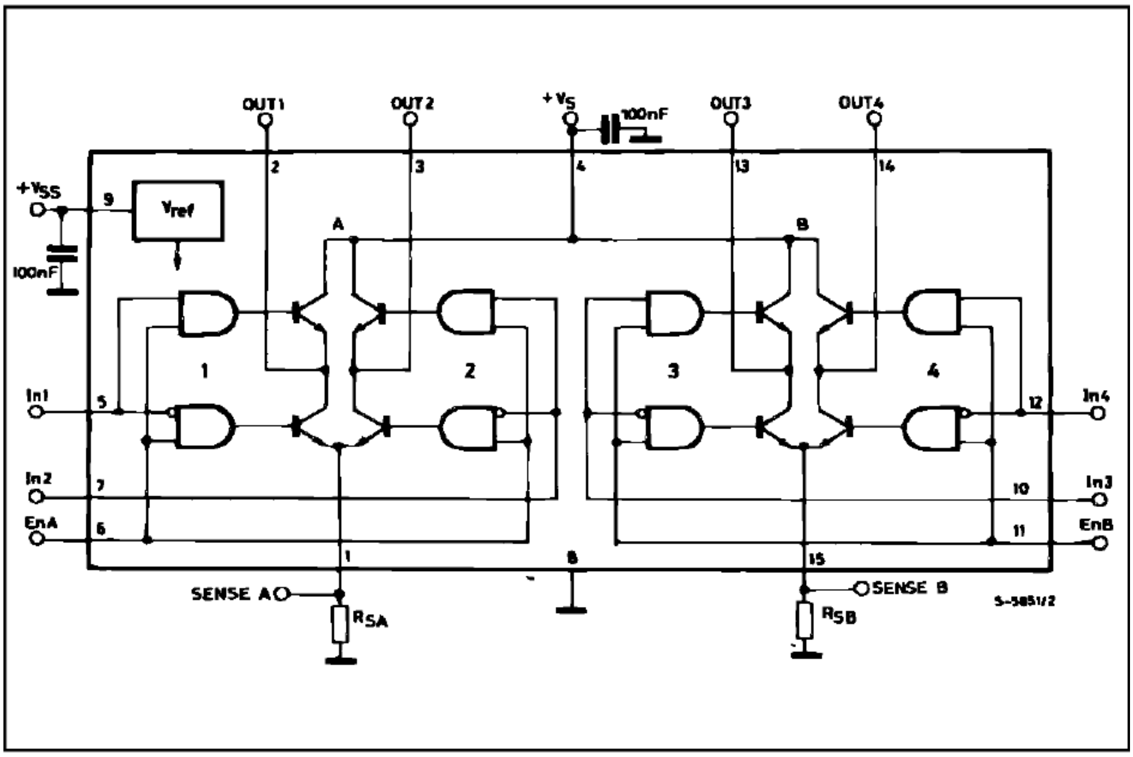
Inputs		Function
V_E = H	Pin 10 = H Pin 15 = L	Turn CW
	Pin 10 = L Pin 15 = H	Turn CCW
	Pin 10 = Pin 15	Fast Motor Stop
V_E = L	Pin 10 = X Pin 15 = X	Free Running Motor Stop

L = Low H = High X = Don't care

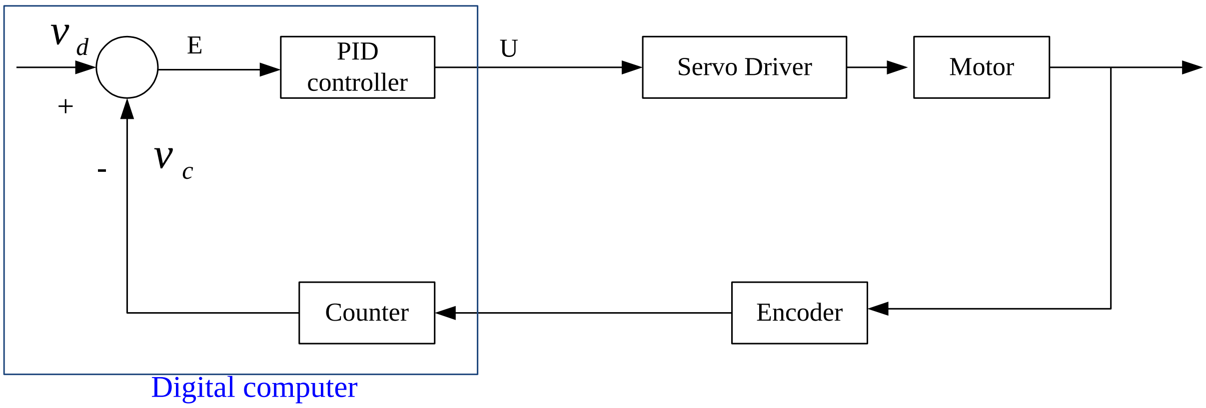
L293D Motor driver with Diodes



L298 -> 2A



7. **Digital PID Controller**



$$\frac{U(z)}{E(z)} = H(z) = k_p + k_i \cdot \frac{Tz}{z-1} + k_d \frac{z-1}{Tz}$$

$$z(z-1)\frac{U(z)}{E(z)} = k_p z(z-1) + k_i Tz^2 + \frac{k_d}{T}(z-1)^2$$

$$multiplyby\frac{1}{z^2}$$

$$(1-z^{-1})\frac{U(z)}{E(z)}=k_p(1-z^{-1})+k_i^{'}+k_d^{'}(1-2z^{-1}-z^{-2})$$

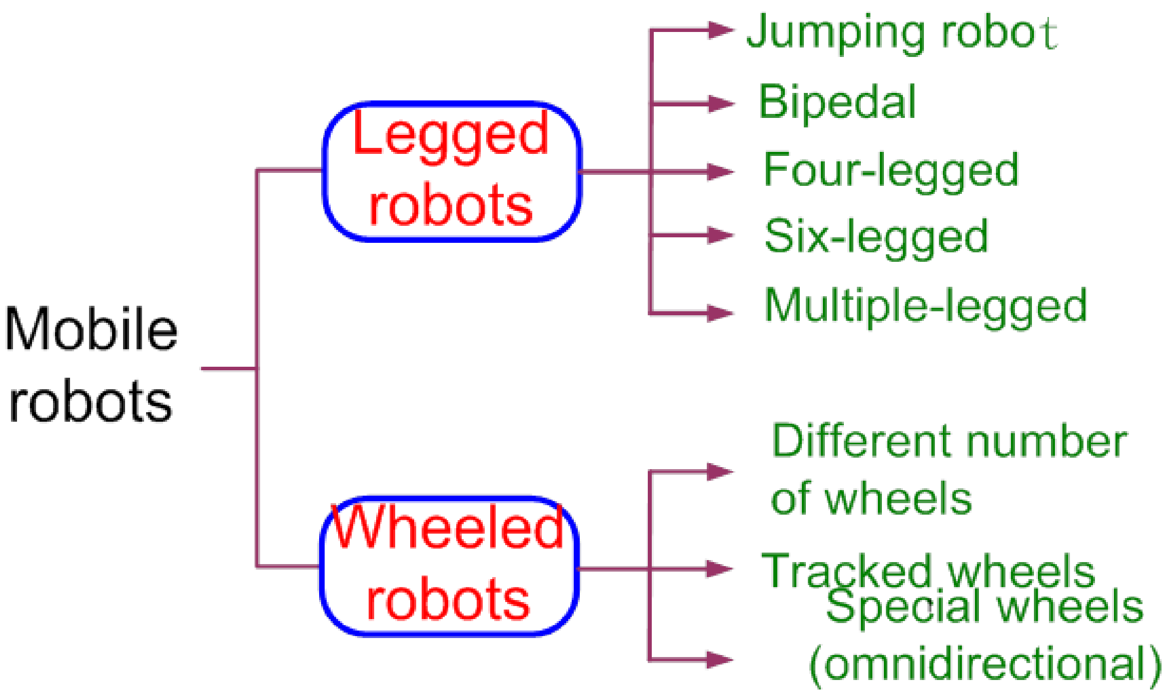
$$(1-z^{-1})U(z)=[k_p(1-z^{-1})+k_i^{'}+k_d^{'}(1-2z^{-1}-z^{-2})]E(z)$$

$$u(k)=u(k-1)+k_p[e(k)-e(k-1)]+k_i^{'}e(k)+k_d^{'}[e(k)-2e(k-1)-e(k-2)]$$

Chapter 4 Locomotion and Kinematics of Mobile Robots

Organization of the Chapter:

- Body + transmission system design
- Legged robot (1, 2, 4, 6 legs)
- Ttracked vehicle
- Wheeled robots
- Self–localization
-



4.1 Wheeled mobile robots, wheeled vehicles

A. Easy to construct

B. Relatively light

C. Easy to get parts (less expensive)

1). Disadvantages : may perform pooly on uneven terrain.

A. Generally cannot go over objects higher (larger) than radius of wheel.

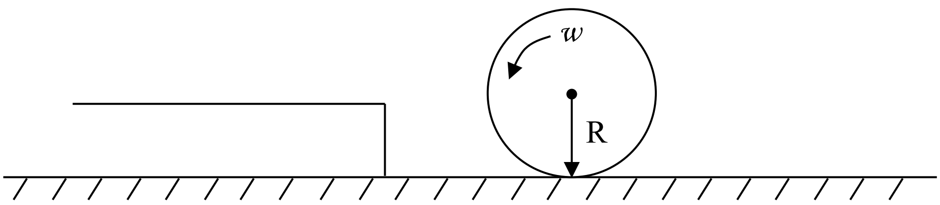


Fig. 4-1 Wheeled robot can not pass a step higher than its wheel radius

B. Wheel slip on soil, wet surfaces especially when accelerating hard.
C. Accumulated error in position estimation, making dead-reckoning difficult.

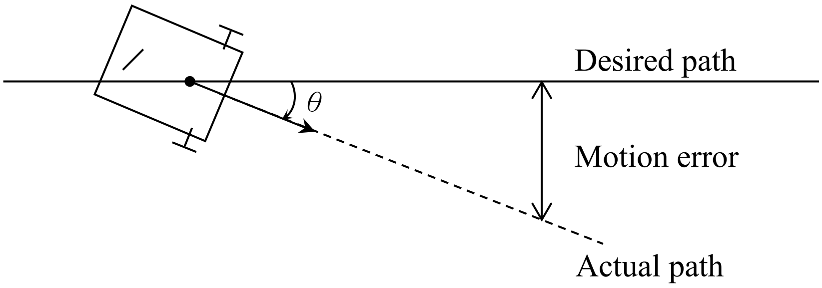


Fig. 4-2 Accumulated position error due to angular error

2). Differential Drive (Two-independent drive wheels)

A. Two independent drive wheels
One of the least complicated locomotion systems

B. Wheels on an aligned axis, each controlled by a separate motor
go straight
turn in plase
move in an arc
(2.5 DOF motion)

C. Balance
a.) Adds a caster (a three-wheel robot)
b.) Add two casters (becomes a four-wheel robot)
can get tight turn
can get trapped on uneven terrain

D. Major difficulty : making robot go straight
a). Two independent servo loops
solution : cross – coupled dynamic control
b). Motors with the same signal have different speeds
c). Different resistance in gear trains
d). Different surface conditions for each wheel

3). Front wheel drive and steer system

A. Dead reackoning on the passive wheels because these two wheels are not powered, slippage is loss likely to occur.
B. Drive and steer on th efront wheel

C.One motor drives and another motor steesr

D. Ssimilar to cars

E. 2-DOF in a plane

4) Non-holonomic constraint

In a world coordinate system, a robot location is specified by (x,y, θ), robot pose has three degrees of freedom.

- Can we position and orient our robot any where on the plane?
- If we give it any (x,y, θ), can the robot be able to move to that location?
- The robot's orientation and position are coupled. In order to turn, it must move forward or backward.

  $\Rightarrow$ It has only have two DOFs $\Rightarrow$ nonholonomic constraint
- Non-holonomic constraint:

The mobile robot can only move in the direction perpendicular to the drive wheels' axis

$$\frac{Vy}{Vx} = \tan\theta \qquad \frac{dy}{dx} = \tan\theta$$

The traveling direction of the vehicle is always tangent to the trajectary of its center position.

4.2. Kinematics

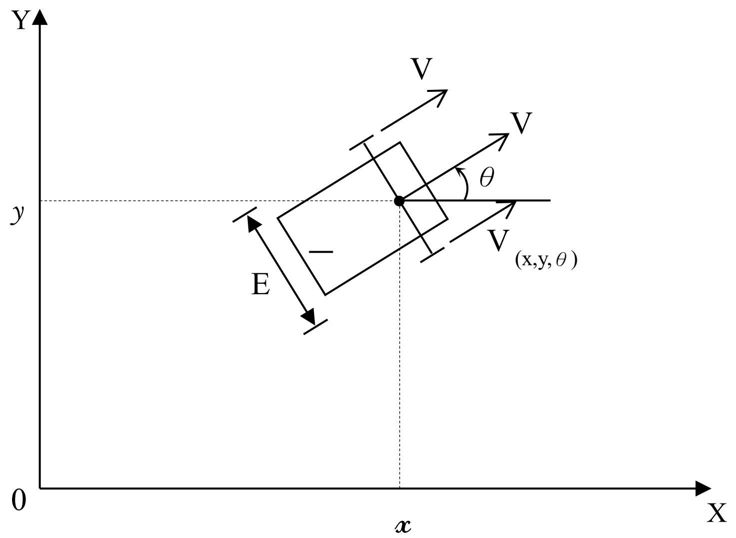


Fig. 4-3 Kinematics of a mobile robot

Linear velocity Vo

(Vl + Vr) / 2 = Vo

w = (Vr – Vl) / E

E : distance between two drive wheels

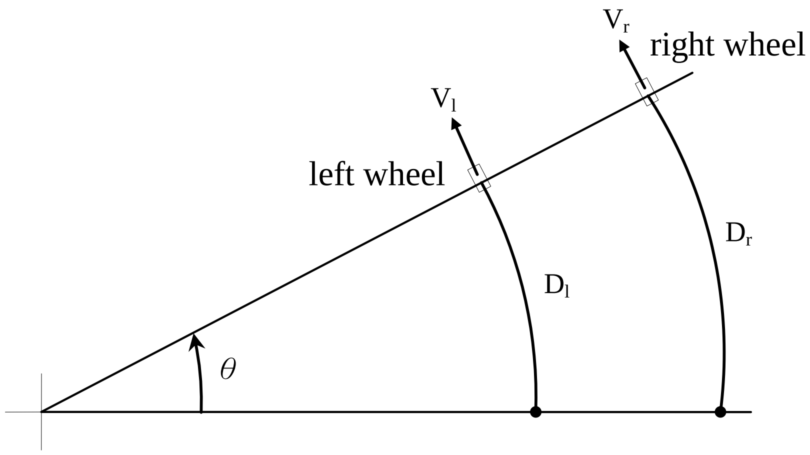


Fig. 4-4 Maneoubility of two independent drive mobile ronbot

Dr – Dl = R1 ϕ – R2 ϕ = ϕ (R1 – R2) = E ϕ

ϕ = (Dr – Dl) / E

Minimize ϕ by maximize E

Maximize ϕ by minimize E

1). Robot Shape

- Circular
    1) can rotate while in contail with object
    2) can be represented as a point (shink to a point)
    3) can mimic the shape of a human being
- Square
    1) must back-up then rotate
    not clear how for and what to do if has 2^{nd} collision while backing-up

4.3. Self-localization using an odometer (odometry)

Dead-reckoning (use internal sensors)

Want two wheels to spin at same rate based on shaft encoders pulse counts to represent angular displacement of motor shaft

1). we sample the pulse counts of left and right motor periodically to calculate the pose of robot (xk ,yk, θ k) at time instant k.

2). the estimated pose is relative to the motor shaft, but not absolute relative to the world coordinate system. Because the pulse counts from the motor shaft do not represent the actual speed of the left and right wheels.

3). while slippage occurs or error due to sampling, it will cause odometry error which will accumulate.

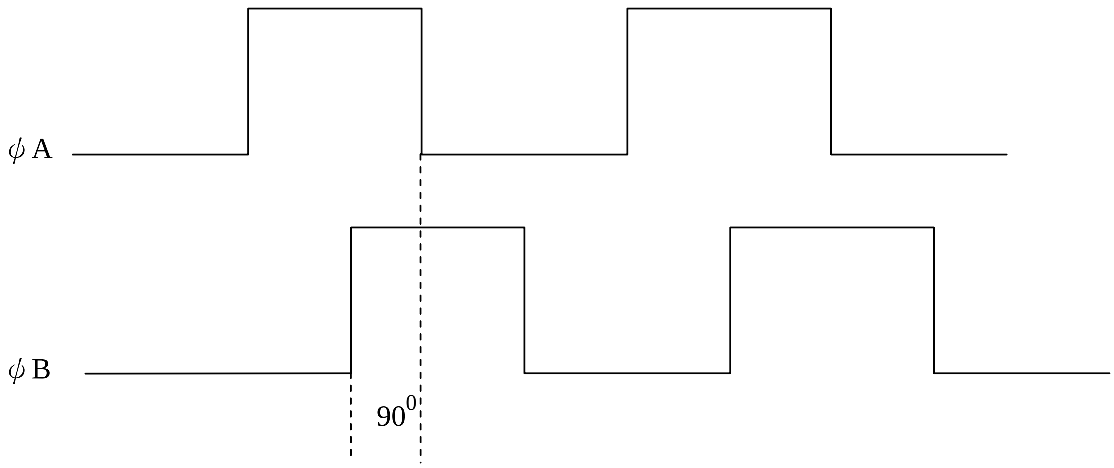


Fig. 4-5 Incremental encoder signals

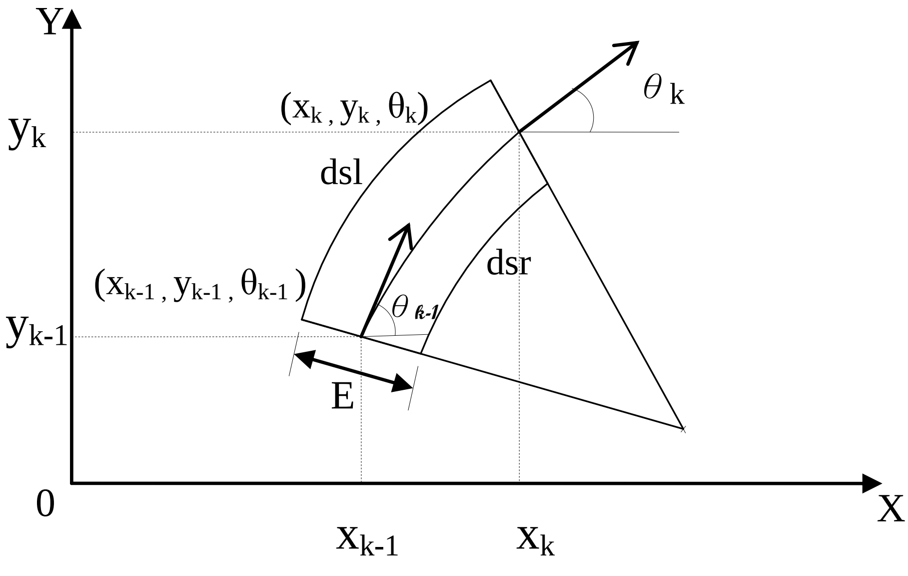


Fig. 4-6 Odometer calculation

$S = 2\pi R / \theta$

$d\theta_k = (dS_r - dS_l) / E$

$dS_k = (dS_l + dS_r) / 2$

$\theta_k = \theta_{k-1} + d\theta_k$

$$x_k = x_{k-1} + dS_k \cos(\frac{\theta_k + \theta_{k-1}}{2})$$

$$y_k = y_{k-1} + dS_k \sin(\frac{\theta_k + \theta_{k-1}}{2})$$

4.4. Omni-directional wheel system

Independent control three degree-of-freedom(x,y, θ). Three/or four drive wheels that can demonstrated real three degrees of freedom motion in a plane. See below for more detailed explanation. To have complete 3 Degree-of-freedom on a plane. Special wheels are used for omni-directional motion. An omni-directional wheel consists of multiple passive rollers on a circular shape. The roller provides free side motion as the wheel rotates. With three or four omni-directional wheels on the mobile platform, the robot can have three-directional motion on a plane.

Fig. 4.7. 全向輪

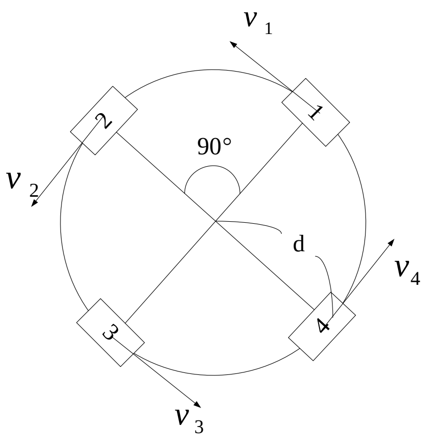
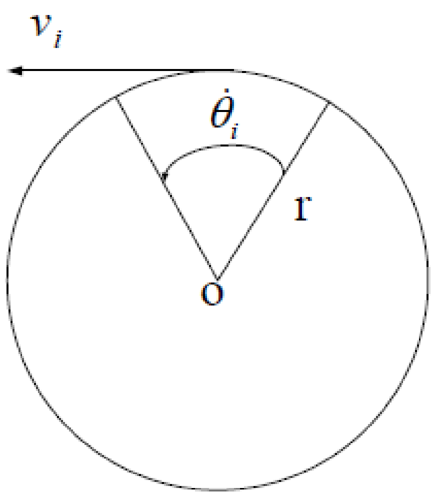


Fig. 4.8. 四輪全向式移動平台配置圖



$$v_i = r\dot{\theta}_i \quad , \quad i = 1, \sim 4$$

Fig. 4.9.全向輪轉動示意圖

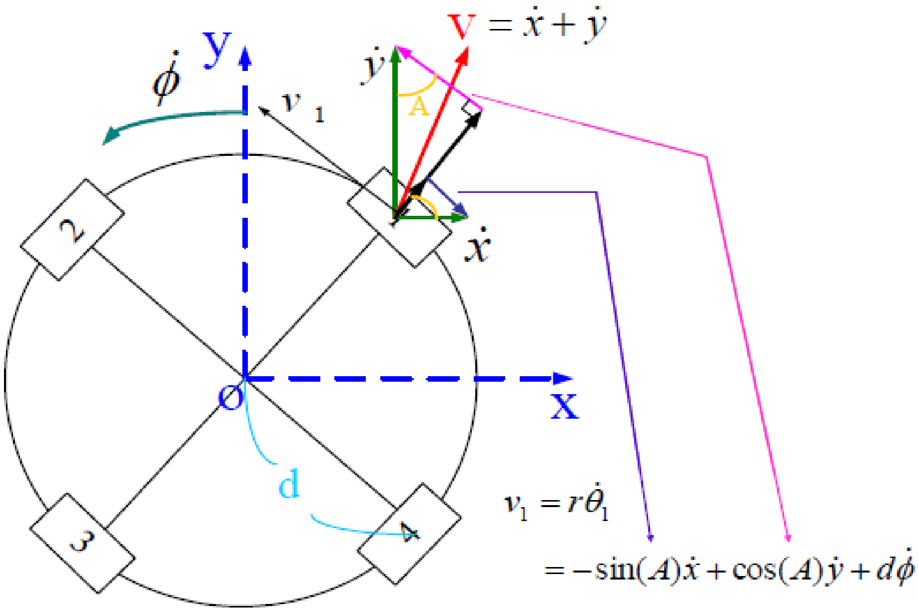


Fig. 4.10.全向輪運動幾何關係

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -\sin(A) & \cos(A) & d \\ -\sin(A) & -\cos(A) & d \\ \sin(A) & -\cos(A) & d \\ \sin(A) & \cos(A) & d \end{bmatrix} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}$$

The mobile platform can be controlled to move in any direction by giving four wheel velocities As the linear and angular velocities of the platform is given, the velocities of four wheels can be calculated and executed for provide the motion.

Chapter 5 Sensors for Mobile Robots

Organization of the chapter:
- Importance of sensors
- Most used sensors for mobile robots
- Sensor interfacing electronics

An autonomous mobile robot moving around in an unstructured and dynamically changing environment must use various sensors to obtain information about its surroundings to determine its action and accomplish its assigned task. From the viewpoint of control, sensor data are required to feedback the actual state of the plant or process, such that the desired state can be achieved. Depending on how the feedback loop is closed, the behavior of robot needs different type of sensor information.

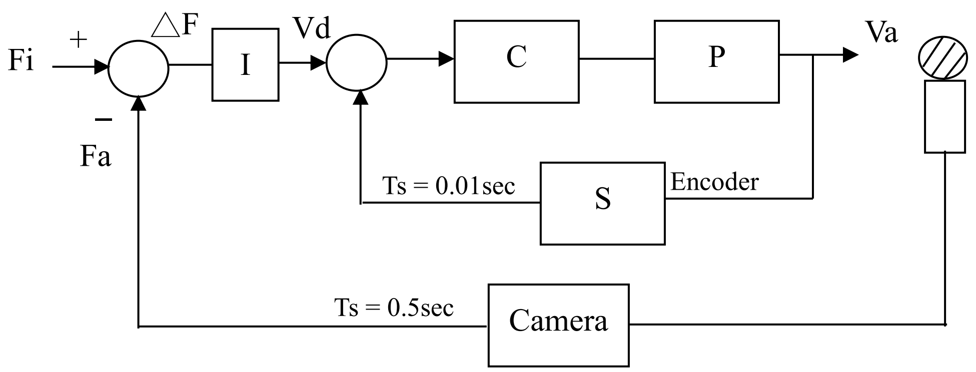


Fig. 5-1 Two control loops of an autonomous robot

5.1 Categorization of Robotics Sensors

1) International Sensors and External Sensors

Internal sensors: sensors used for robots to understand the state of itself.
e.g. sensors for self localization (pose estimation),
sensors for motor servo control, shaft encoders, gyro, limit switches, etc

External sensors: sensors used for robots to understand the external world.
e.g. ultrasonic range sensors, infrared sensors, laser scanner (range finder), image sensor, microphone, touch sensor, gas sensor, etc

2) Ob-board Sensors and Sensors in Intelligent Environments

On-board sensors: sensors that are installed on board the robot.

Intelligent environment: intelligent devices that are installed (deployed) in the environment. These devices can measure the changes of the environment as well as robot pose and provide the information to the robot.

5.2 A General Interfacing Circuit for Robotics Sensors

Sensor Interfacing: Digital

Analog

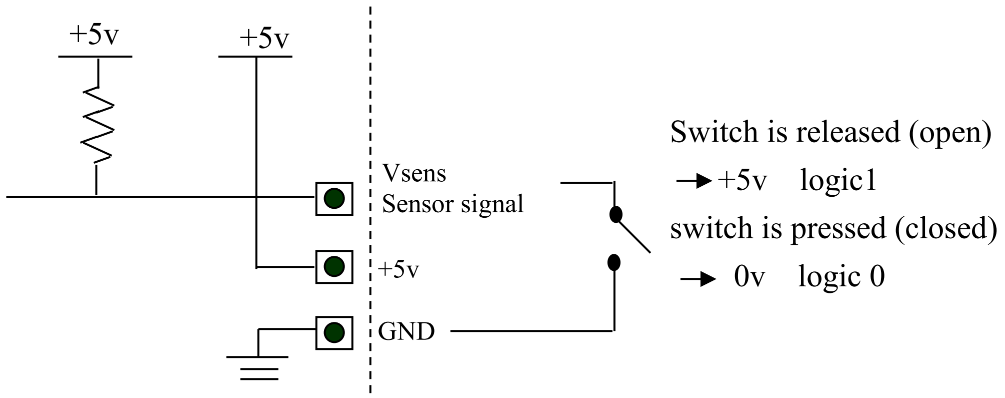
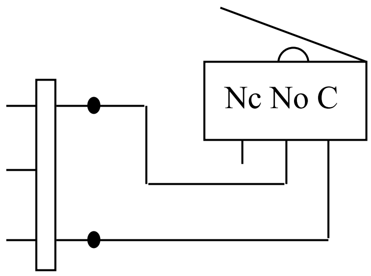


Fig. 5-2 A general connection for sensor inputs

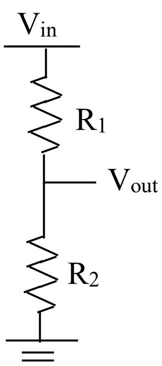
Vsens converts to either digital input circuitry or analog input circuitry.



Logic high = not pressed

Logic low = pressed

Fig. 5-3 Connecting a touch sensor



$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

light seeking experiment

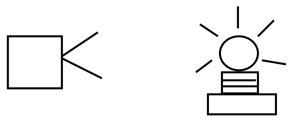
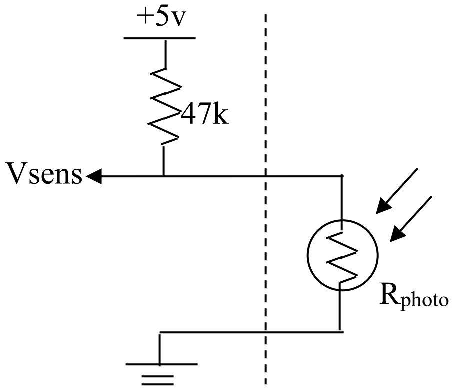


Fig. 5-3 Connecting a resistive sensor

Light sensor, photocell, photo resistor, CdS



Signal photocell circuit

1. Brightly illuminated

   R_{photo} : small

   Vsens : close to 0

2. Dark

   R_{photo} : small

   Vsens : close to 0

Fig. 5-4 Connecting a photocell

Analog inputs: need an A/D converter

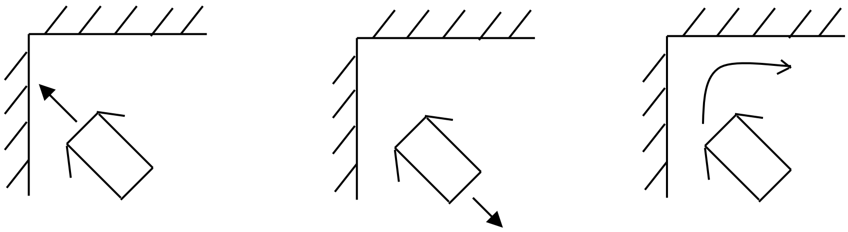
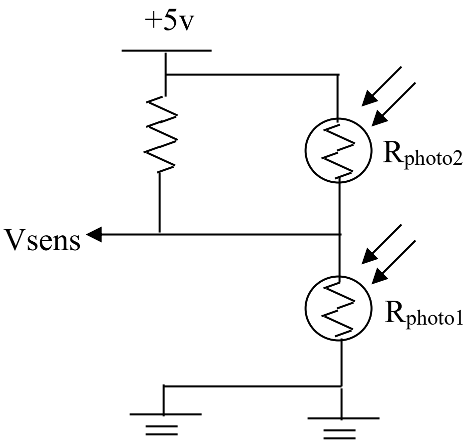


Fig. 5-5 Using touch sensor to turn around a corner

Differential photocell sensor



$$Vsens = 5\frac{R_{photo1}}{R_{photo2} + R_{photo1}}$$

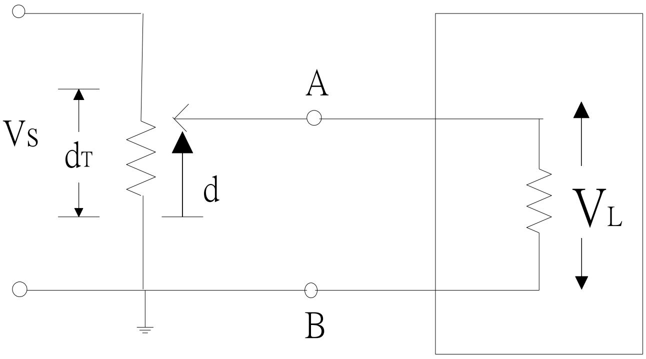
choose photocell that haves a relatively small dark resistance (i.e. about 10k)

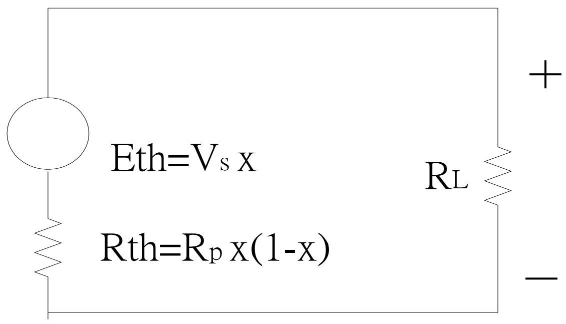
$$\frac{1}{10k} + \frac{1}{47k} \approx 8.25k\Omega$$

Fig. 5-6 Connecting two light sensors

Resistive position sensors

potentiometers

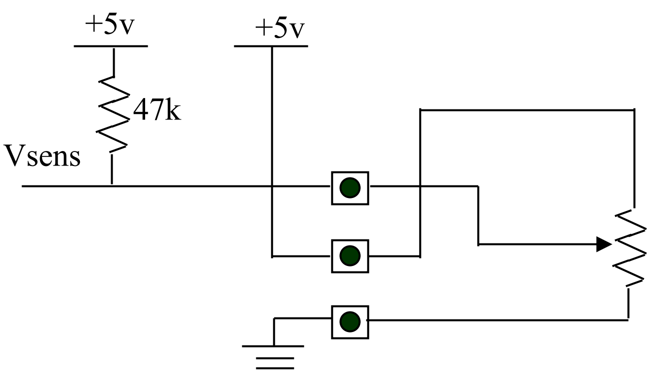




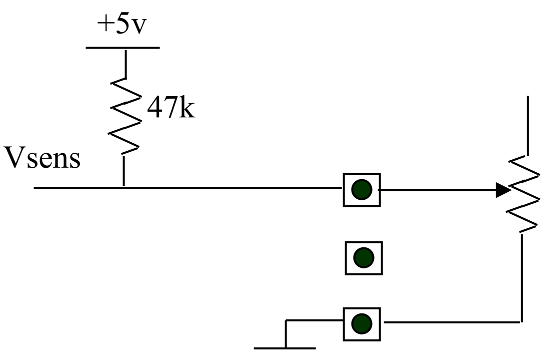
$$\frac{d}{d_T} = x$$

$$V_L = V_s x \frac{R_L}{R_{th} + R_L}$$

Fig. 5-7 Using potentiometer position sensor



Three-wired connection
For end-to-end resistance
less than 10k



Two-terminal wiring
Otherwise

Fig. 5-8 Three terminal and three terminal connection

5.3 Near-infrared proximity sensor (~880nm from LED element(Emitter))

Photo diode/ photo transistors vs. photocells(CdS) photo resistors

- rapid response time
- more sensitive to small level of light

1) These sensors typically do not return actual distance to an object, they signify whether or not something is present within the cone of detection.
2) These types of sensors usually have much narrower beam width than sonar range.

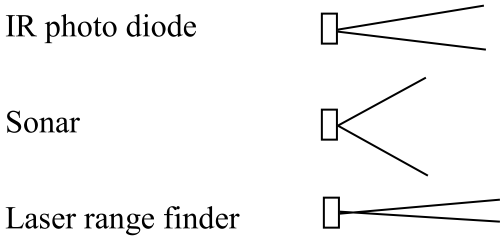
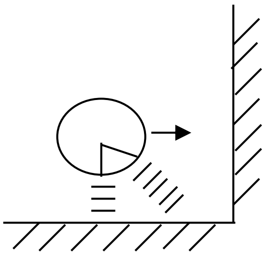


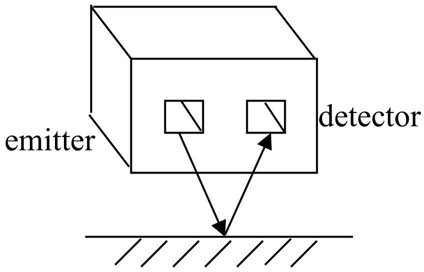
Fig. 5-9 Different types of range sensor



Reflective optosensors (active sensors)

Light is reflective off a surface into a detector element, consisting of an emitter LED and a detector photo diode phototransistor.

Fig. 5-10 Reflective optosensor



Ambient light is a problem

Fig. 5-11 Transmitter LED and Receiver photo diode

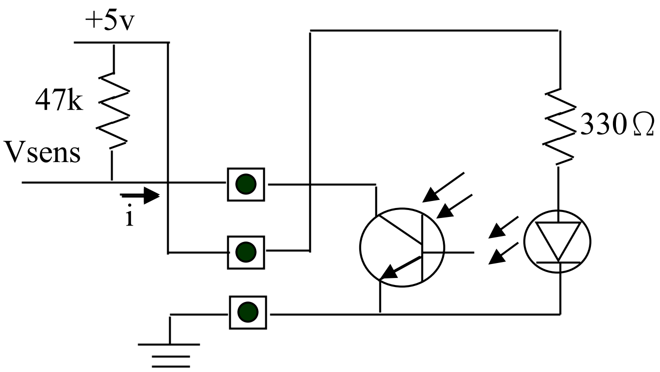


Fig. 5-12 Interface circuit of light sensor

The more light received by the phototransistor, the more current flows. This creats a voltage drop in the 47K pull-up resistor. This voltage drop is reflected in a smaller voltage on the Vsens signal line.

i =0.01mA

$V= i R = 0.01*10^{-3}*47*10^{3}= 0.47V$

Vsens= 5 - 0.47 = 4.53V

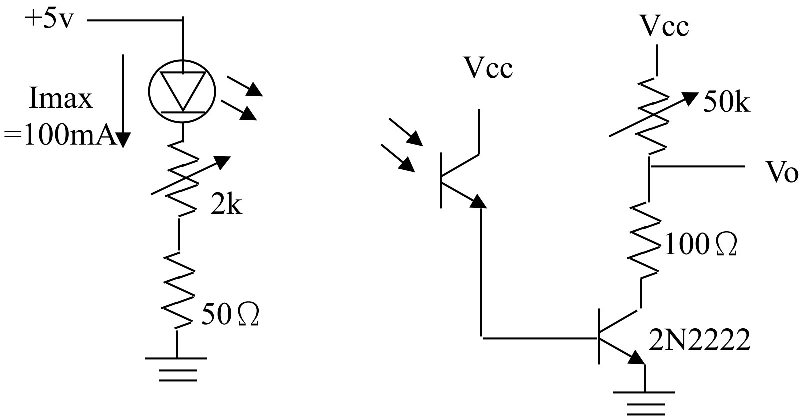


Fig. 5-13 Light emitting LED and receiving photo transistor interface circuit

Break-beam sensors

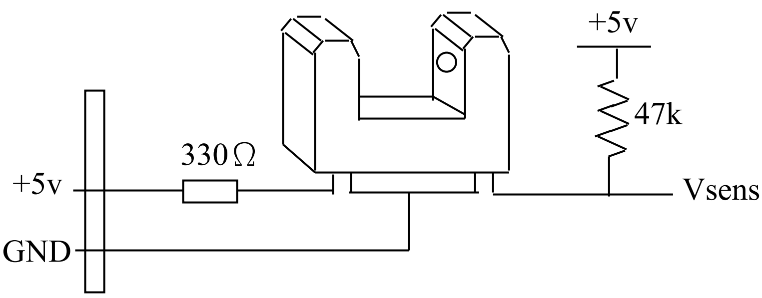


Fig. 5-14 Break-beam sensors

The break-beam device consists of a light-emitting component aimed at a light-detecting component. When an opaque object comes between the emitter and detector, the beam of light is occluded, and the output of the detector changes

5.4 Modulated IR signal

IR detectors respond to a modulated carrier sent by the near-infrared LED. The programmer is responsible for brinking the LED in certain pattern such that the detect will respond.

Emitter signal :

Fig. 5-15 Modulated signal

- Sharp GP1U 52X near-infrared proximity detector

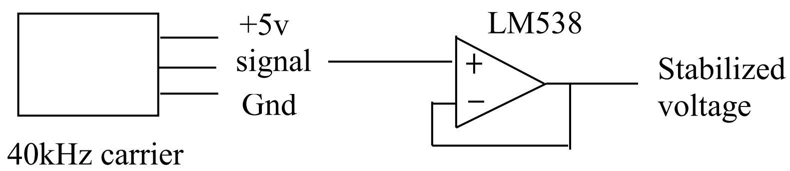
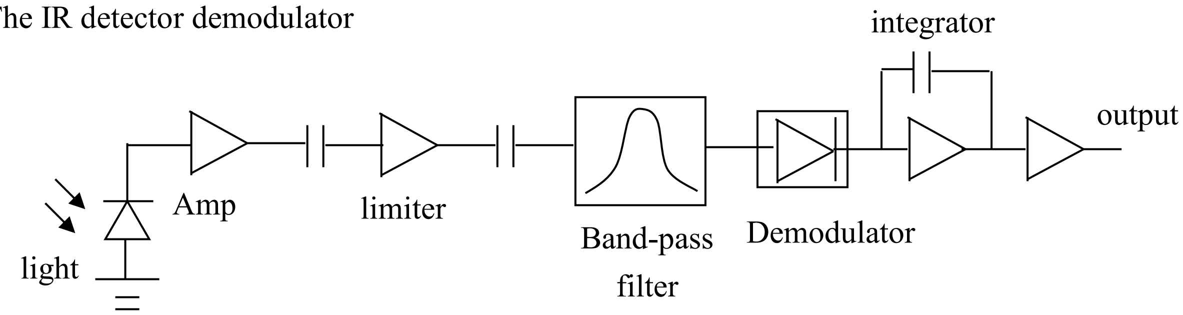


Fig. 5-16 Near-infrared proximity detector

The IR detector demodulator



BP filter center freq: 38kHz or 40kHz

Fig. 5-17 Signal block diagram of IR receiver module

Liteon IR330, 38KHz 940nm

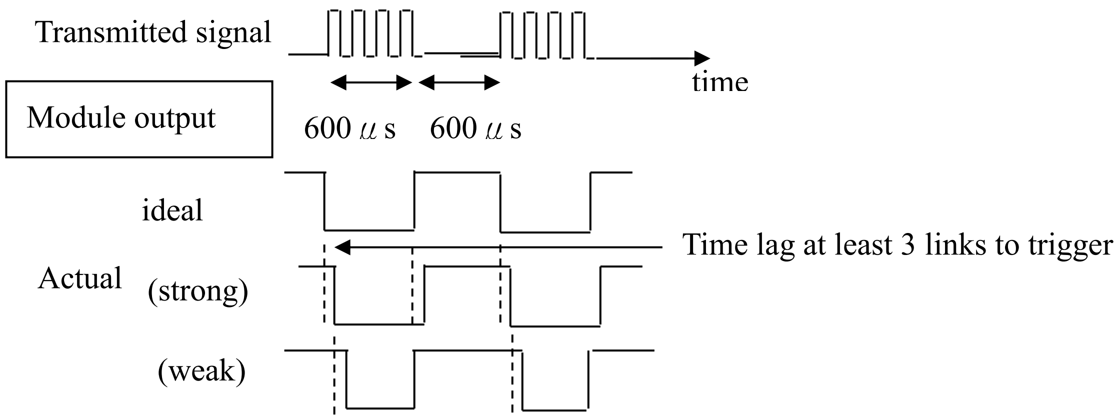
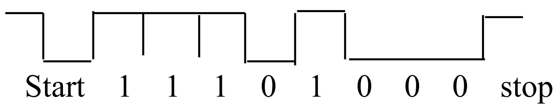


Fig. 5-18 Actual received signal

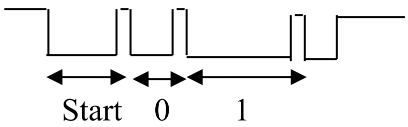
Homing: need to recognize a digital code for distinguish robot's home.

How to read a digital code?

Bit form



Bit interval



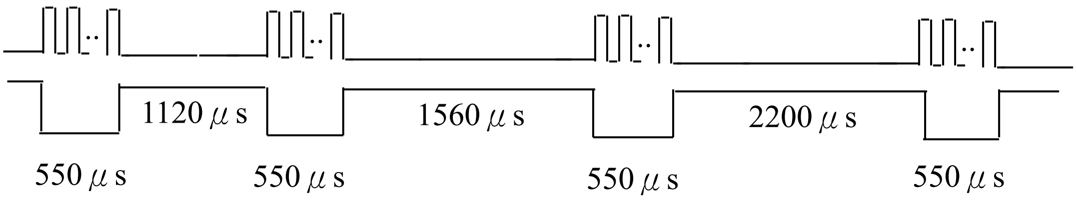


Fig. 5-19 representation of a digital code

Look at lapsed time between falling edges to determine if a bit is "1" or "0".

Ex: 1.67ms --> start

2.11ms --> 1

2.75ms --> 0

IR modules receive light(signals) from wide angles need to narrow angle of received signal to leave it on it.

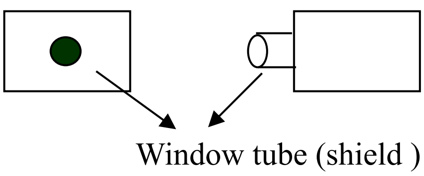


Fig. 5-20 Shield of light

5.5 Ultrasonic ranging system(SONAR) (Polaroid 6500)

- Measures the time of flight for a sonar "chirp" to bounce off a target and return to the sonar
- More accurate than IR, giving a distance to a direction, possible to make a scanning of the environment.

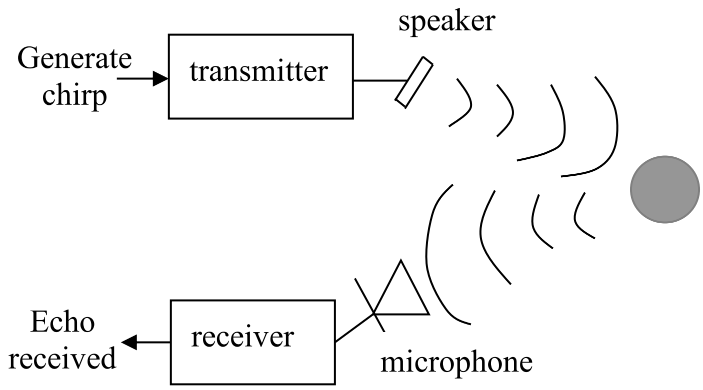


Fig. 5-21 Ultrasonic range sensor

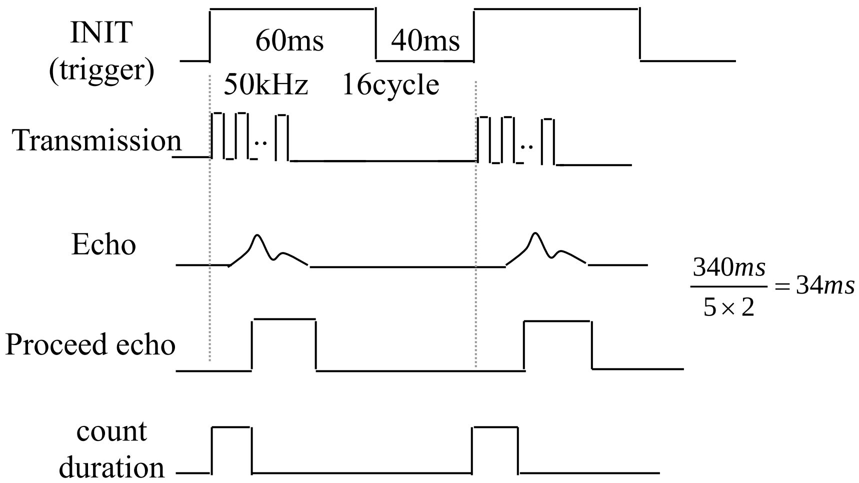


Fig 5-22 Signal used in an ultrasonic sensor

Polaroid 6500 sonar ranging system

- A transducer which acts as both the speaker and microphone
- A circuit board
- chirp frequency 50kHz (49.4kHz), 16 cycles
- Blanking of signal in the beginning for 2.38ms (40cm)

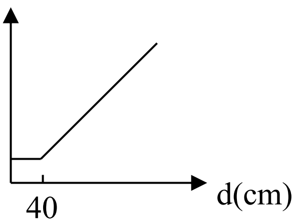


Fig. 5-23 Characteristics of ultrasonic ranging system

Limitations

1) Specular reflection

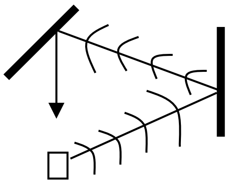
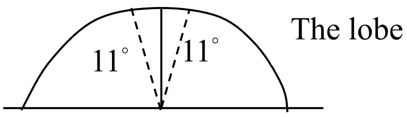


Fig. 5-24 Multiple reflection causes measurement error

2)Beam (Opening) Angle = 22.5°



Influence of beam angle

(a) (b)

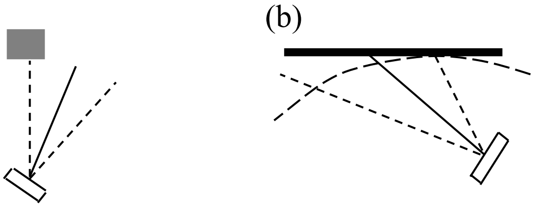


Fig. 5-254 Bean angle of the ultrasonic transducer causes measurement error

Optical distance sensing detector with Sharp GP2D02

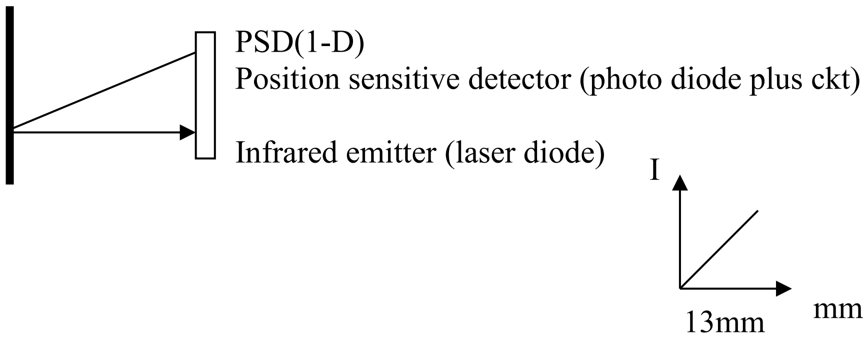


Fig. 5-26 Use of PSD in distance measurement

5.6 Other types of sensors for environment detection

1) Pyroelectric sensor can detect the existence of heat generated by human body (1~2 μm infrared)

2) Cameras and vision systems

a. CCD

b. CMOS

c. PTZ

Image frames contain much information of the immediate environment. It's therefore required to segment useful features and extract information need for task execution.

→image acquisition + image processing + recognition algorithm (visual servoing
visual tracking)

3) Laser scanners(laser range finders)

Can get accurate environment distance, fast scanning, powerful, but expensive, 75Hz, 81m.

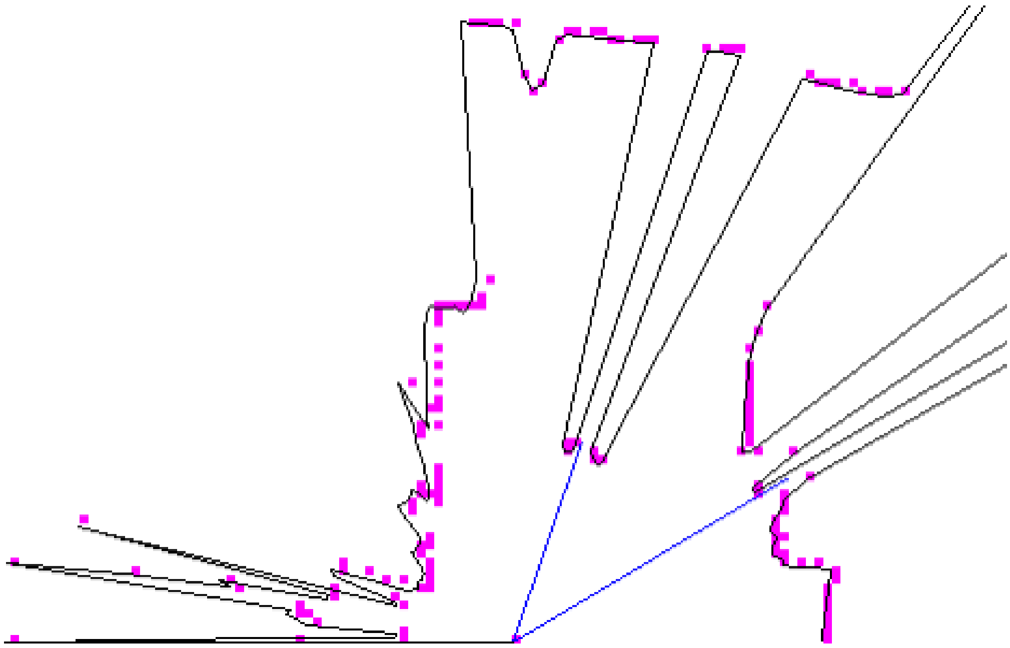


Fig. 5-27 Experiment of SICK laser scanner in distance measurement

Chapter 6 Data Acquisition Systems

6.1 Sensor signal conditioning

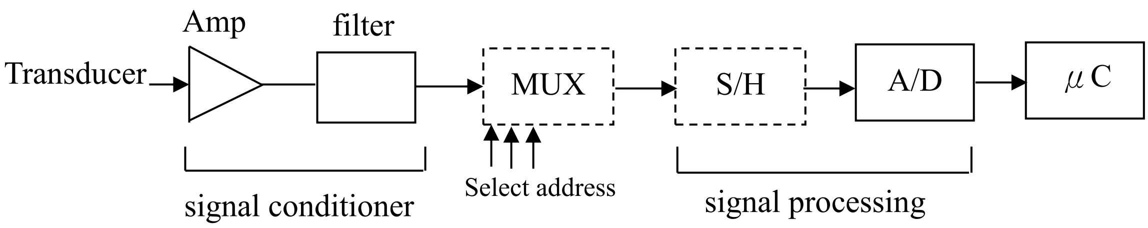


Fig. 6-1 Signal conditioning of sensors

Transducer: convert physical phenomenon to electrical signal.

Amplifier: amplify transducer output signal into the range of the A/D converter

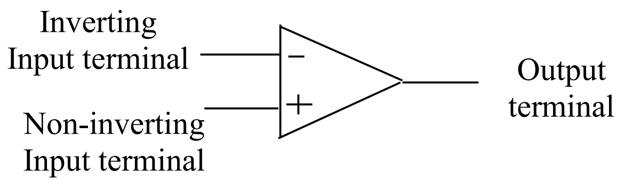
Filter: remove unwanted high frequency signals to prevent aliasing errors

Multiplexer (MUX) : for using one A/D with multiple sensors

Sample and hold (S/H) : to hold an analog signal steady which it is being digitized

Analog to digital converter (A/D) : digitize analog signal so they can be used by a μC

6.2 Operation Amplifiers (OP Amp)



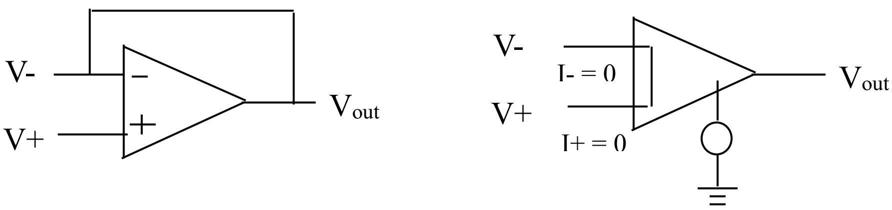


Fig. 6-2 General models of an OP Amp

Golden rule for ideal OP Amp

1. I+ = I- = 0 no currents enter input terminals,
2. infinite input impedance
3. V+ =V- infinite gain

1) Inverting amplifier

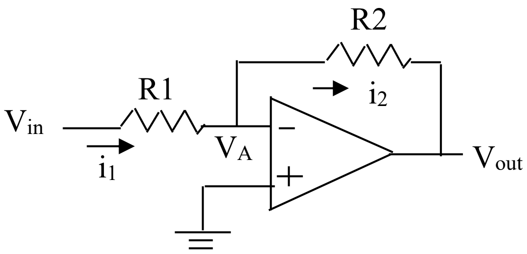


Fig. 6-3 Inverting amplifier

$$i_1 = \frac{V_{in} - V_A}{R_1} = \frac{V_{in}}{R_1}$$

$$i_2 = \frac{V_A - V_{out}}{R_2} = -\frac{V_{out}}{R_2}$$

$$i_1 = i_2 \quad => \quad \frac{V_{in}}{R_1} = -\frac{V_{out}}{R_2}$$

$$V_{out} = -\frac{R_2}{R_1} V_{in} \quad , \quad gain = -\frac{R_2}{R_1}$$

2) Non-inverting amplifier

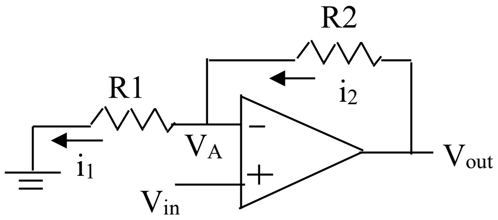


Fig. 6-4 Non-inverting amplifier

$$i_1 = i_2 = \frac{V_{out}}{R_1 + R_2}$$

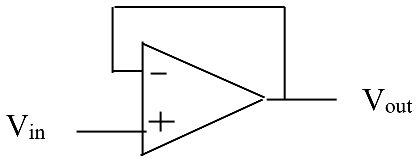
$$V_A = V_{in} = i_1 R_1 = \frac{R_1}{R_1 + R_2} V_{out}$$

$$V_{out} = \frac{R_1 + R_2}{R_1} V_{in} = (1 + \frac{R_2}{R_1}) V_{in}$$

$$gain = 1 + \frac{R_2}{R_1} \geq 1$$

3) Voltage follower

If $R_2 = 0$, $R_1 = \infty$



Voltage follower, buffer Amp

Fig. 6-5 Voltage follower

- $V_{out} = V_{in}$
- couple to voltage signal with loading the source of the voltage (high input impedance)

Example:

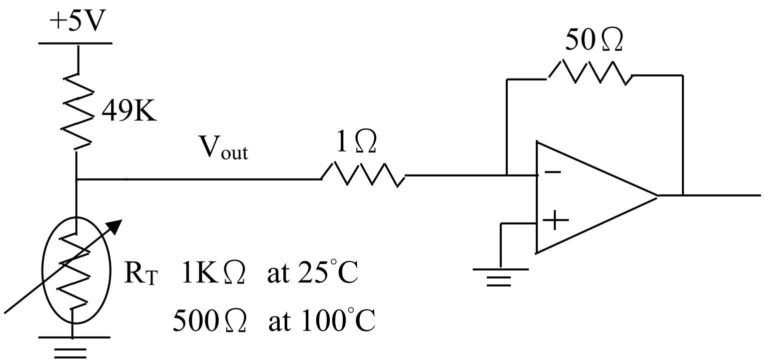
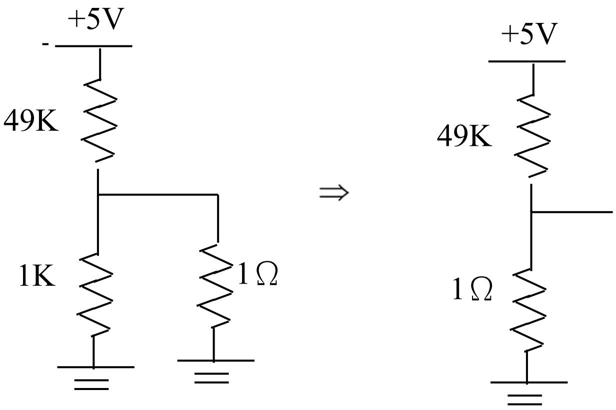


Fig. 6-6 An example of design for a temperature sensor

25°C $V_{out} = \dfrac{1k}{49k+1k} \times 5 = 0.1V$

100°C $V_{out} = \dfrac{0.5k}{49k+0.5k} \times 5 = 0.05V$

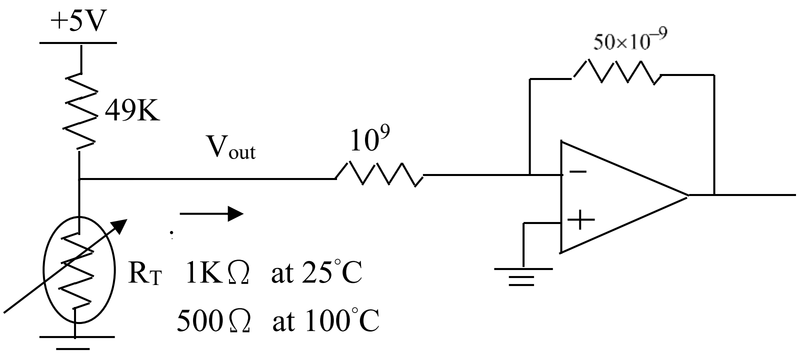
Want Vout=-5V at 25°C → $gain = \dfrac{5}{0.1} = 50$倍

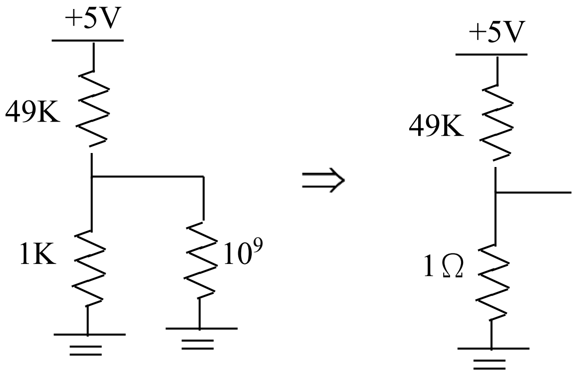


$V_{in} = \dfrac{1}{49k+1} \times 5 = 1 \times 10^{-4} V$

$10^{-4} \times 50 = 50mv$ (not 5V!)

Have impedance loading problem





$$V_{in} = \frac{1k}{49k + 1k} \times 5 = 0.1V$$

$$i = \frac{0.1v}{10^9} = 10^{-10} A$$

Amplifier current too low

Susceptible to noise

→ Voltage follower

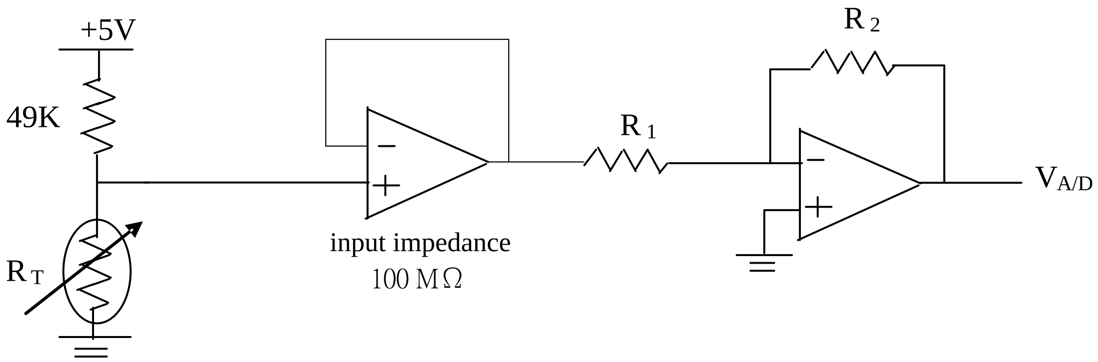


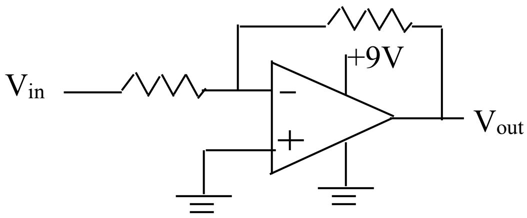
Fig. 6-7 Use of voltage follower to solve the problem

A typical OP amp parameters

Parameter	ideal OP Amp	Typical OP Amp
A	∞	100dB
Zin	∞	2MΩ
Zout	0	75Ω
Temp. coef. of		
Input offset voltage	0	5mV/°C
Vos	0	1mV
Input bias current	0	80mA
0~f_B	0~∞	0~10Hz
CMRR	∞	90dB

4) Rail-to-rail output

Means that the OP Amp can output a signal swing very close to both the +Ve supply rail and –Ve supply rail



Vout(max) = 7~8V
Vout(min) = 1~2v
741series cannot reach perfect 9V to 0V
Want Bipolar OP Amp with JFET input

5) Frequency response of OP Amp

A real OP has a finite bandwidth, which is a function of the gain established by external components.

- The gain bandwidth product (GBP) of an OP Amp is the open loop gain and the bandwidth at that gain.
- The GBP is a constant over a wide range of frequencies.
- Open loop gain decreases with input signal frequency.
- High quality OP Amp has large GBPs.
- The closed-loop gain is always limited by the open loop gain of the OP Amp.

Example:
A noninverting OP amp with a closed loop gain of 100 would have a bandwidth of 0Hz-10,000Hz
The frequency where the open loop gain curve starts to limit the closed loop gain is called fall-off frequency.

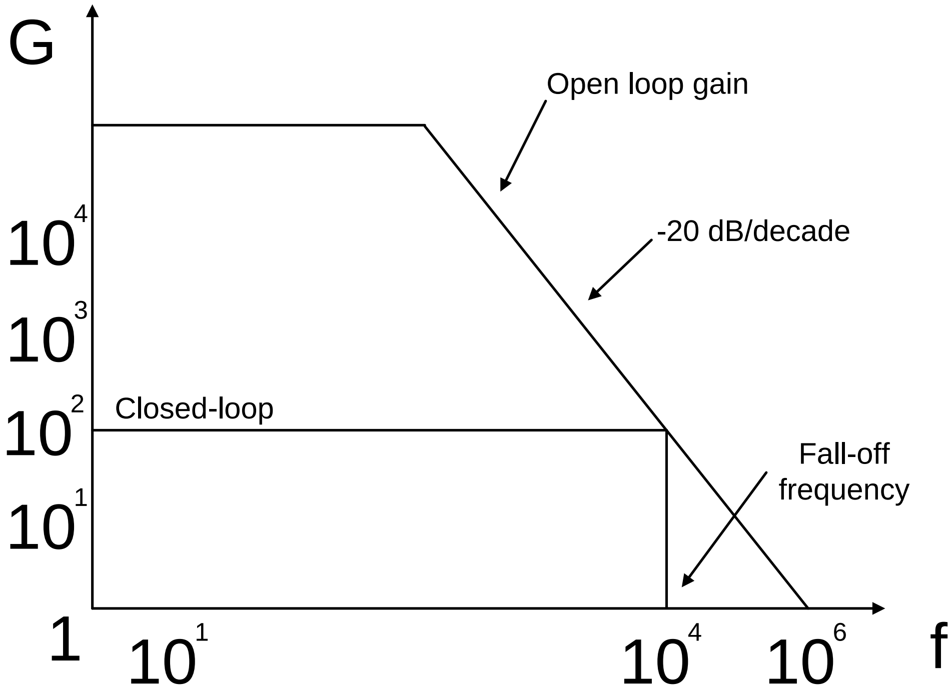
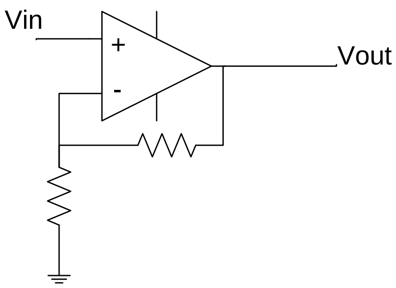


Fig. 6-8 Frequency characteristics of an OP Amp



$$Gain = \left(\frac{R_2}{R_1} + 1\right)\left(\frac{1}{1 + \frac{1}{A_{OL}\beta}}\right)$$

A_{OL} = open-loop gain of amplifier

$$\beta = \frac{R_1}{R_1 + R_2}$$

Fig. 6-9 A non-inverting amplifier

If $V_{in} = 10Hz$

$$Gain = \left(\frac{990K}{10K} + 1\right)\left(\frac{1}{1 + \frac{1}{10^5 \cdot 10^{-2}}}\right) = 99.9$$

If $V_{in} = 100\,KHz$

$$Gain = \left(\frac{990K}{10K} + 1\right)\left(\frac{1}{1 + \frac{1}{100 \cdot 10^{-2}}}\right) = 50$$

If V_{in} contains frequencies up to 100KHz need to use 2 Amplifiers, need to use:

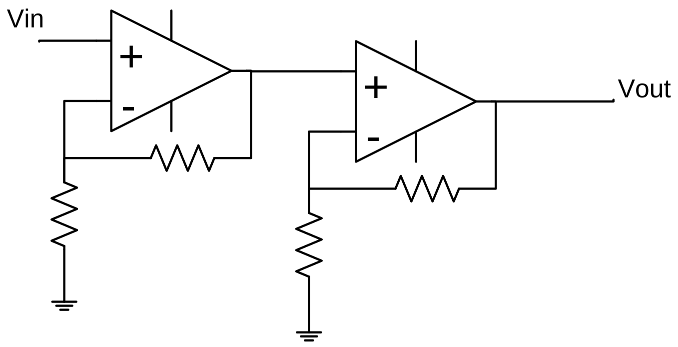
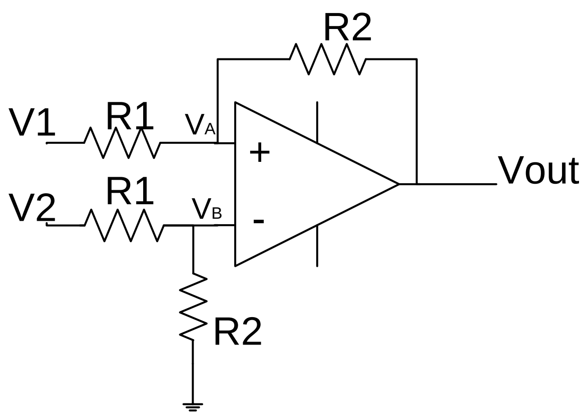


Fig. 6-10 Two amplifier cascade to solve the problem

6) Differential Amplifiers



$$V_A = V_B = V_2 \frac{R_2}{R_1 + R_2}$$

$$i_1 = \frac{V_1 - V_A}{R_1} = i_2 = \frac{V_A - V_{out}}{R_2}$$

$$\frac{\left(V_1 - \frac{V_2 R_2}{R_1 + R_2} \right)}{R_1} = \frac{\left(\frac{V_2 R_2}{R_1 + R_2} - V_{out} \right)}{R_2}$$

(Inadequate : high gain, R1 decreases → low input impedance)

$$\frac{R_2}{R_1}\left(V_1 - \frac{V_2 R_2}{R_1 + R_2} \right) = \frac{V_2 R_2}{R_1 + R_2} - V_{out}$$

$$V_{out} = \frac{V_2 R_2}{R_1 + R_2} - \frac{R_2}{R_1}\left(V_1 - \frac{V_2 R_2}{R_1 + R_2} \right) = \frac{V_2 R_2}{R_1 + R_2} - \frac{R_2}{R_1} V_1 + \frac{V_2 R_2}{R_1 + R_2} \frac{R_2}{R_1}$$

$$= \frac{R_2 V_2}{R_1 + R_2}\left(\frac{R_1 + R_2}{R_1} \right) - \frac{R_2}{R_1} V_1 = \left(\frac{R_2}{R_1} \right) V_2 - \frac{R_2}{R_1} V_1 = \frac{R_2}{R_1}\left[V_2 - V_1 \right]$$

7) Instrumentation Amplifier (IA)

An instrumentation amplifier is a high performance differential amplifier system of several closed-loop OP Amp

$V_{out} = K(V_2 - V_1)$

$K =$ precisely known, can be over a wide range can be set by a AD524 single external resistor

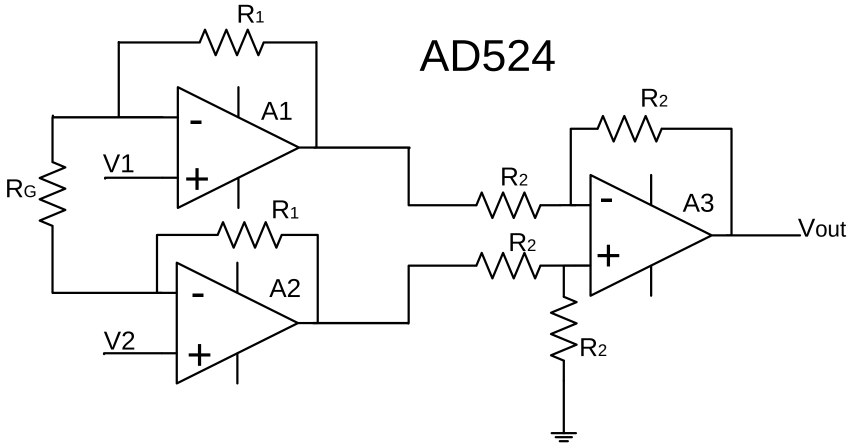


Fig.6-12 Instrumentation amplifier

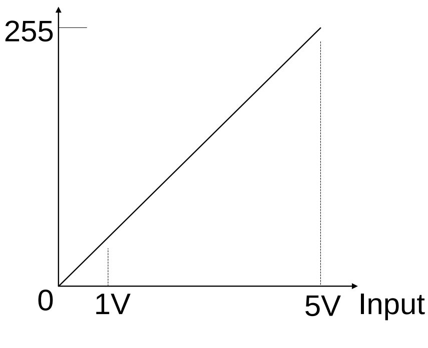
$$V_{RG} = V_1 - V_2$$

$$I_{RG} = \frac{V_1 - V_2}{R_G}$$

$$V_O^{'} = I_{RG}(2R_1 + R_G)$$

$$= \frac{V_1 - V_2}{R_G}(2R_1 - R_G) = (V_1 - V_2)(1 + 2\frac{R_1}{R_G})$$

6-3 Data acquisition system



1) A/D : input range(0~5V)

\# of bits = (n), resolution

Conversion time: $t_a = \frac{1}{f_{\max}}$

f_s : sampling frequency

$f_s \le f_{\max}$

Resolution: depends on range and # of bits

1 LSB = Range(Full scale)/ 2^n

0-5V (Full scale = 5V)

n = 8 bit

resolution = $\frac{5V}{2^8}$ =0.0195V

2) S/H amplifier

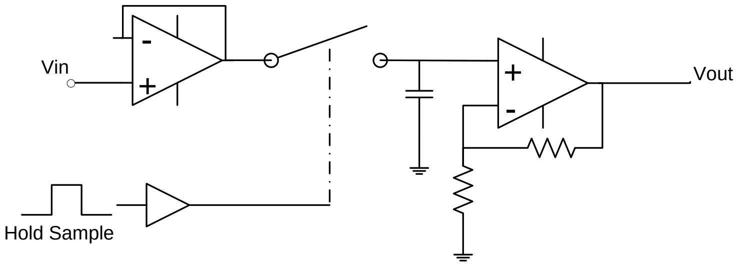


Fig. 6-13 Sample hold amplifier

$V_{in} = \hat{V} \sin 2\pi f t$

Sampling time t_a, A/D range = 0-5V

We want $\left(\dfrac{dV_{in}}{dt} \times t_a \right) \le 1LSB$ for no S/H

$$\frac{dV_{in}}{dt} = \hat{V} \cdot 2\pi f \cos 2\pi f t$$

$$\left(\frac{dV_{in}}{dt} \right)_{\max} = \hat{V}\, 2\pi f$$

$$\boxed{\hat{V}\, 2\pi f \cdot t_a \le \frac{V_{A/D}}{2^n}}$$

where $V_{A/D}$ =input range of A/D

$$f_{\max} \le \frac{V_{A/D}}{\hat{V}\, 2\pi t_a \cdot 2^n} \qquad \hat{V}_{\max} \le \frac{V_{A/D}}{2\pi f t_a \cdot 2^n}$$

3) Specification

A. What do we want to measure? Eg. Temperature 0°C-100°C

B. How accuracy what resolution?
   0°C-100°C with resolution of 0.5°C
   Frequency of interest: 0~0.2Hz
   A/D input voltage: 0-5V
   A/D conversion time: 0.1ms

Select transducer RTD

$R_T = 100 + 0.4T$ (°C)

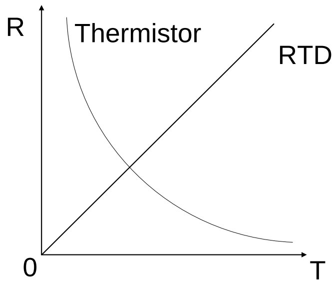


Fig. 6-14 Thermister temperature sensor

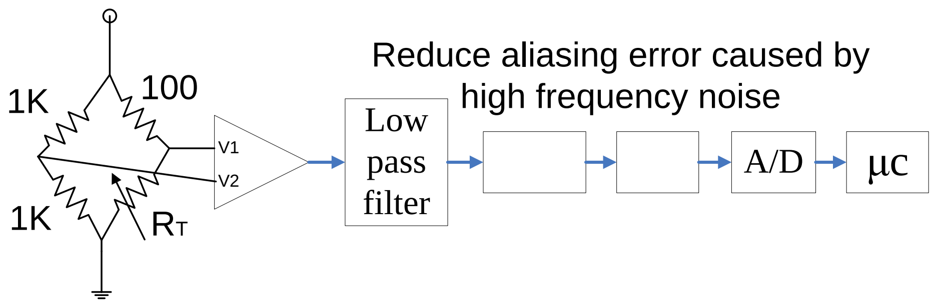


Fig. 6-15 measurement of RTD resistive temperature sensor

(a) What gain should the Amp have?

At 0°C $V_2 - V_1 = 0$

At 100°C $V_0 - V_1 = \dfrac{140}{100+140} \times 5V - \dfrac{1K}{1K+1K} \times 5V = \dfrac{140}{240} \times 5V - \dfrac{1}{2} \times 5V = 0.159V$

gain$=\dfrac{5V}{0.159V} = 31.35 \cong 31$

(b)How many bits should the A/D have?

$\dfrac{100^\circ C}{0.5^\circ C} = 200$ scale variations

8 bit → 256 → 8 bit n=8

7 bit → 128

(c) Is a S/H Amplifier required?

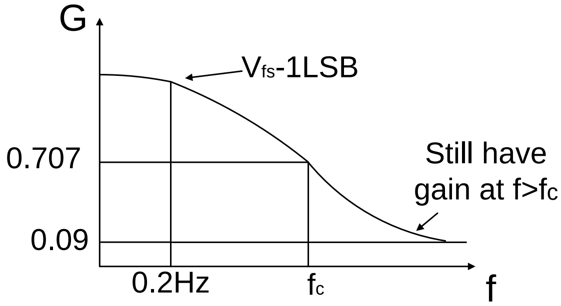
$$\left(\frac{dV_{in}}{dt}\right)_{\max} = \hat{V} \times 2\pi f = 5(2\pi)(0.2) = 2\pi$$

$$\left(\frac{dV_{in}}{dt}\right) t_a = 2\pi \times 10^{-4} = 6.28 \times 10^{-4}$$

A/D resolution $\dfrac{5V}{2^n} = 0.0195V \quad >> \quad 6.28 \times 10^{-4}$

No need S/H

(d) What cutoff frequency should a one-pole or low-pass filter have so that the A/D error is 1 LSB or less at 0.2Hz?



Note $G = \dfrac{1}{\sqrt{1+(f/f_c)^{2k}}}$ k: order of the filter

Gain= $\dfrac{V_{filterout}}{V_{filterin}} = \dfrac{V_{fs} - 1LSB}{V_{fs}} = \dfrac{V_{fs} - \dfrac{V_{fs}}{2^n}}{V_{fs}} = \dfrac{V_{fs}(1-\dfrac{1}{2^n})}{V_{fs}} = \dfrac{2^n - 1}{2^n} = \dfrac{255}{256}$

$$f_c = \frac{f}{\sqrt{\dfrac{1}{G^2} - 1}} = 2.26Hz$$

(e) Suppose it is known that the combined temperature and voltage noise at the Amplifier input is equivalent to 0.007V, what sampling frequency is required to ensure the aliasing error is no more than 1LSB?

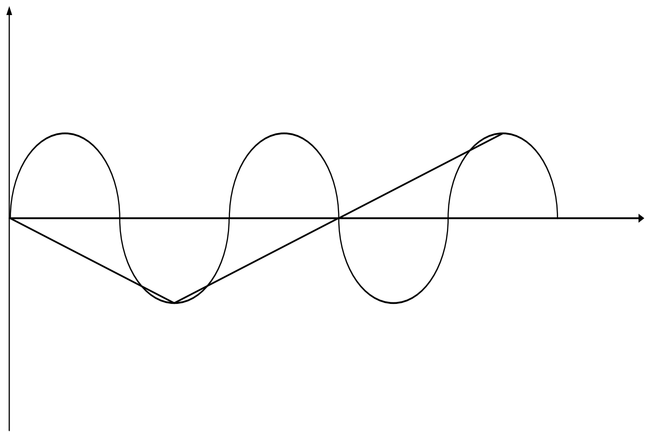


Fig. 6-16 Aliasing error

We want: $noise \times Amp.Gain \times Filter.gain \le 1LSB$

Filter gain $\le \dfrac{1LSB}{Noise \times Amp.gain} = \dfrac{5/2^8}{0.007 \times 31} = 0.09$

Filter gain < 0.09

$$f = f_c \cdot \sqrt{\frac{1}{G^2} - 1} = 2.26\sqrt{\frac{1}{0.09^2} - 1} = 25Hz$$

$$f_s \ge 2(25) = 50Hz$$

When f>25Hz, aliasing error coursed by higher frequencies is no more than 1LSB

4) Some tips

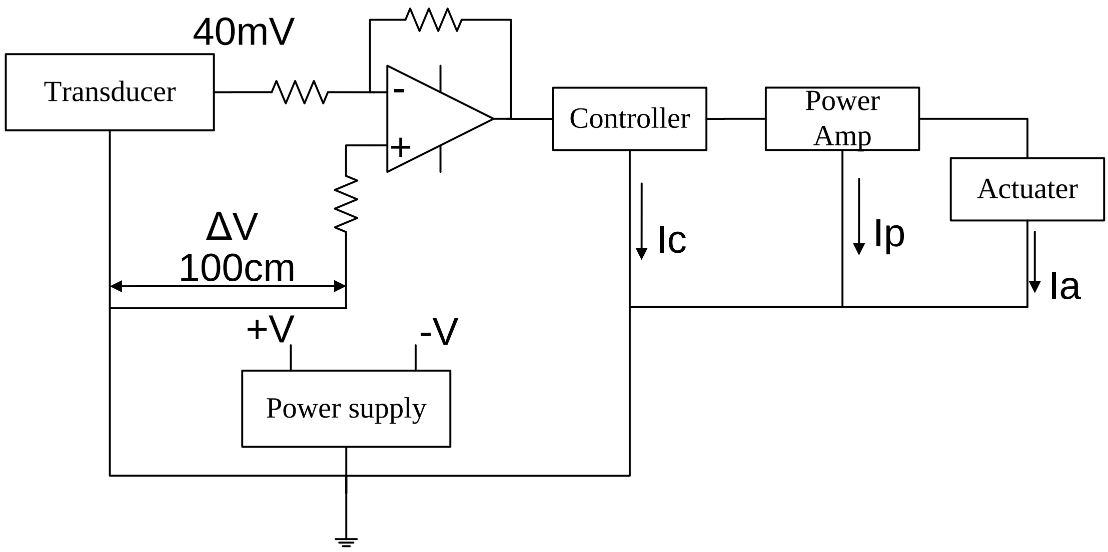


Fig. 6-17 consideration of grounding of power stage and low level sensor stage

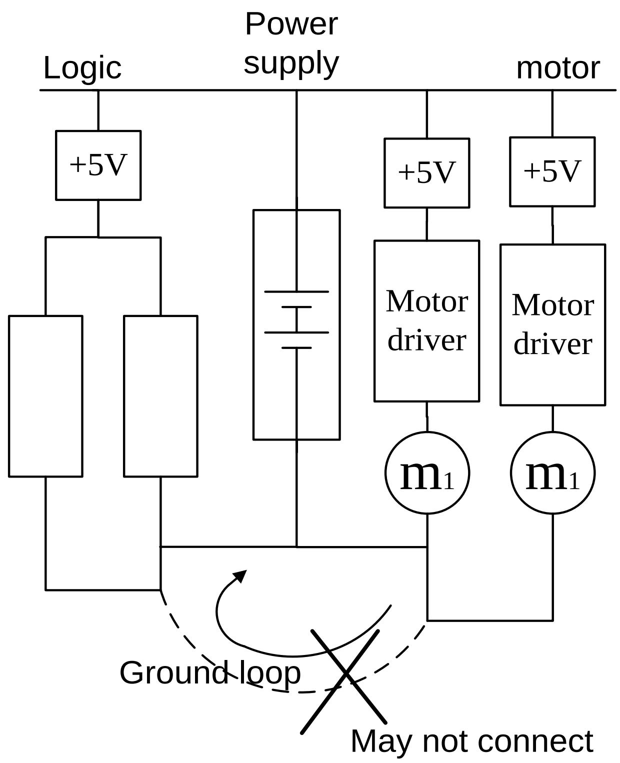


Fig. 6-18 Prevent from a ground loop

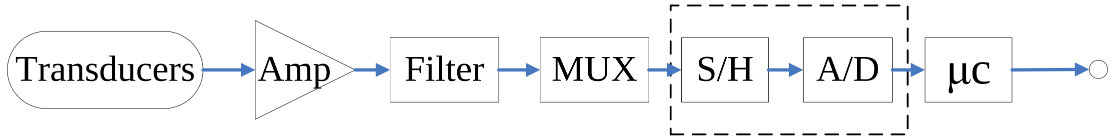
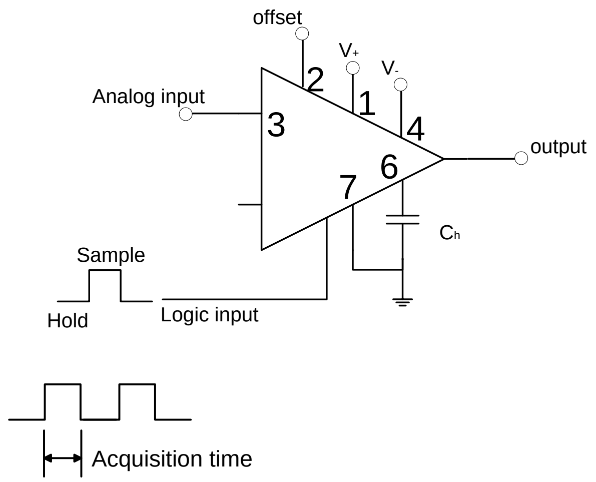


Fig. 6-19 Design of sample hold and A/D converters

5) LF398 S/H Amplifier 10μs Acquisition time



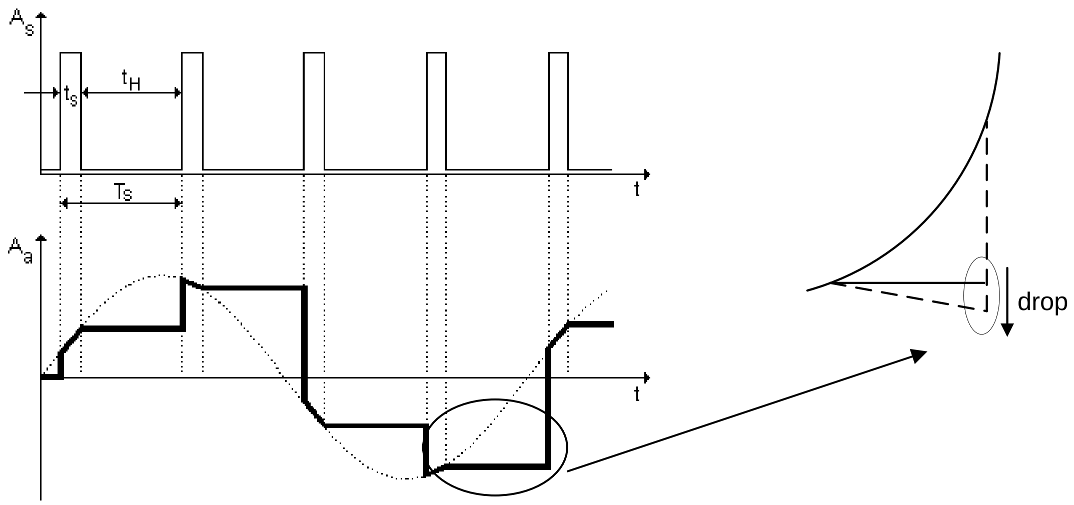


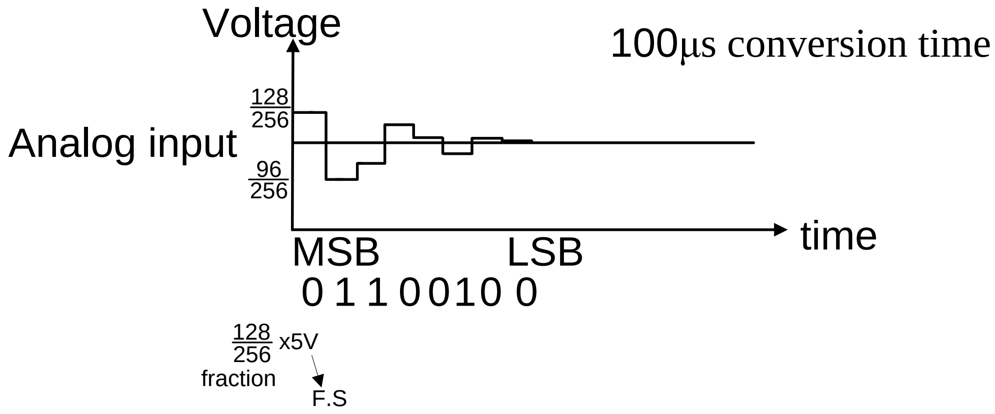
Fig. 6-20 Operation principle of sample hold

BI-FET Technology

Bipolar input stage for low offset voltage and wide bandwidth JFET in the output amplifier for low drop rate (5mV/min)

6) ADC 0804 8 bit μP compatible A/D converter

Successive Approximation



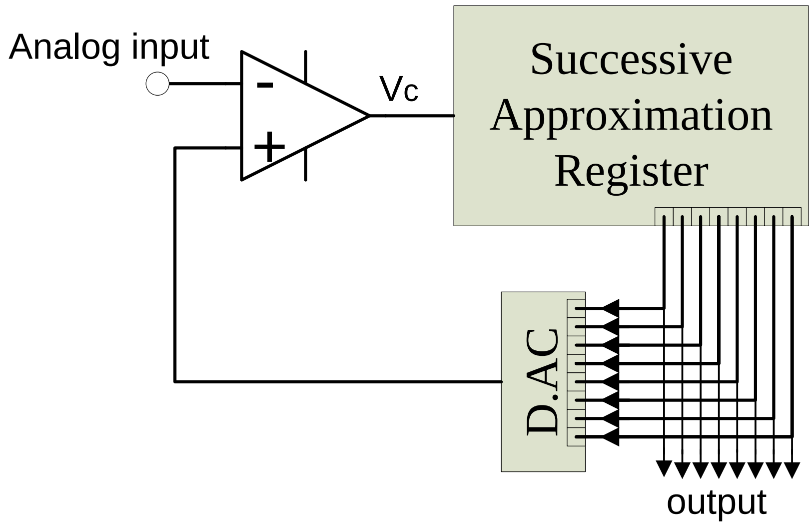


Fig. 6-21 Operation principle of successive approximation A/D converter

1) conversion time is fixed (100μs)

2) need 8 successive guesses for 8-bit A/D converter

3) The analog input needs to hold stable for accurate conversion

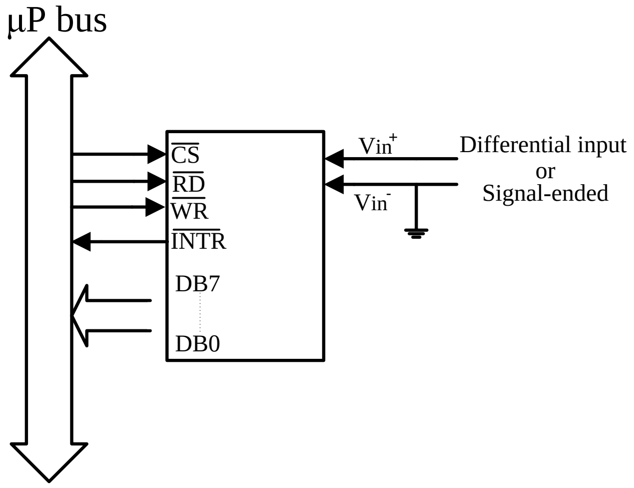


Fig. 6-22 Interfacing the A/D converter

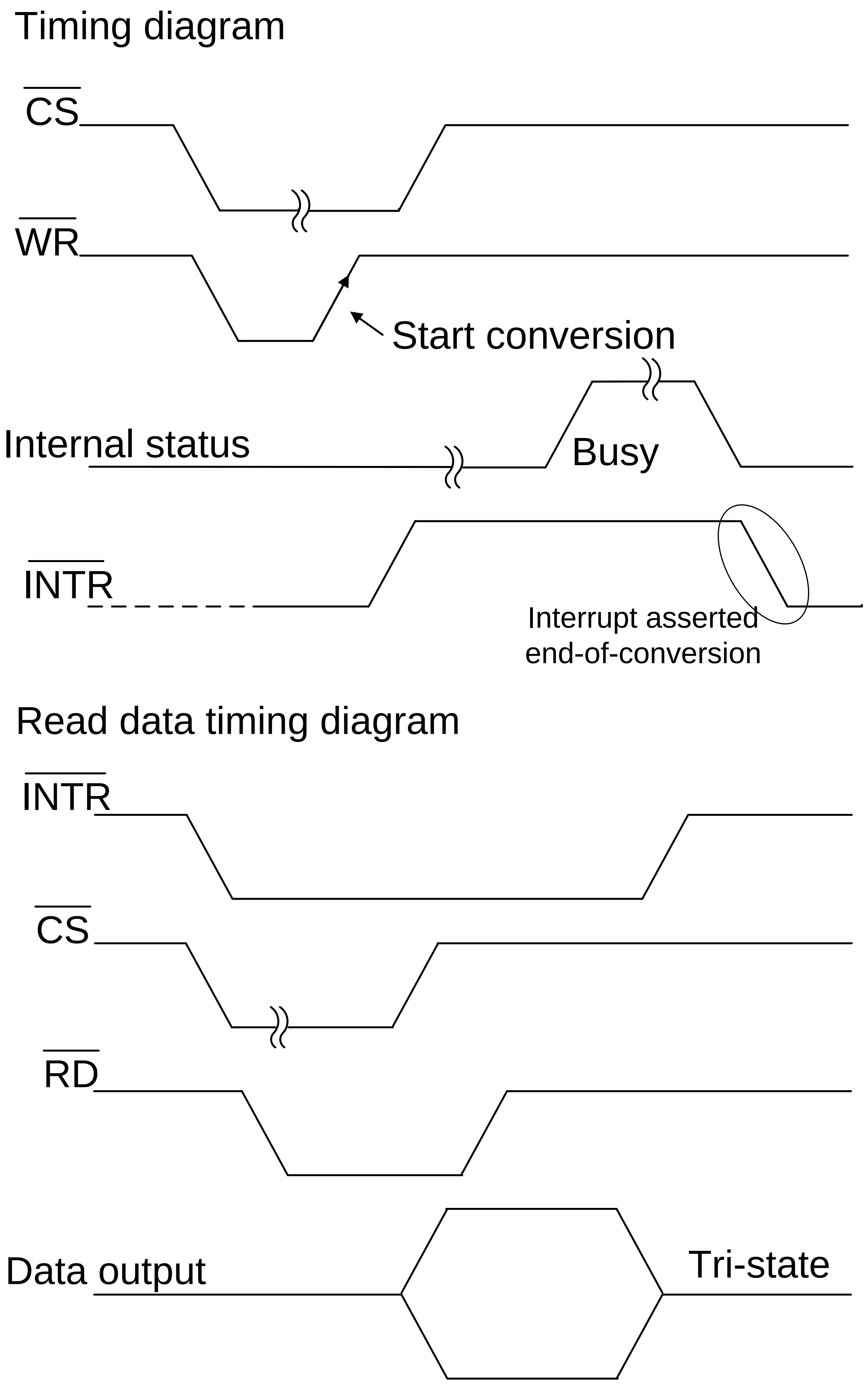
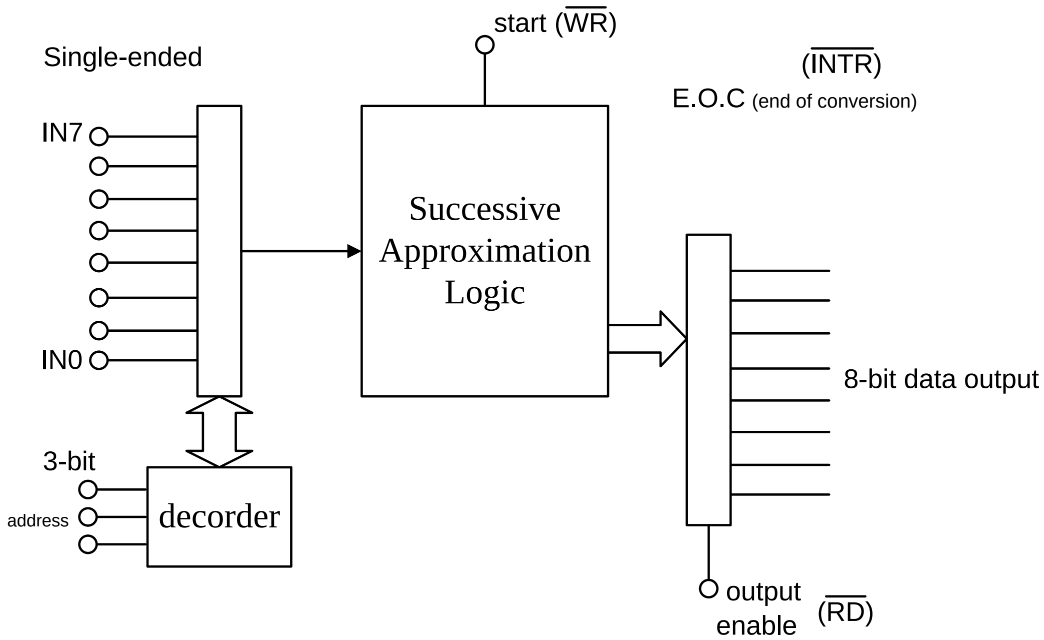


Fig. 6-23 Timing diagram of ADC0804

7) ADC 0809 8-bit μP compatible A/D converter



ALE
(address latch enable)

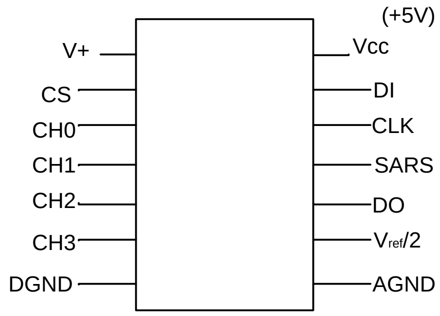
	A	B	C
IN7	1	1	1
IN0	0	0	0

Fig. 6-24 Operation principle of ADC 0809

8) ADC 0838 8-bit serial I/O A/D converter with 4 channels multiplexer

Conversion time 32μs

Mux addressing



Address				Channels			
SGL/DIFF	ODD/SIGN	SELECT 1	SELECT 0	0	1	2	3
1	0	0	1	+			
1	0	1	1			+	
1	1	0	1		+		
1	1	1	1				+
Differential Mode							
Address				Channels			
SGL/DIFF	ODD/SIGN	SELECT 1	SELECT 0	0	1	2	3
0	0	0	1	+	-		
0	0	1	1			+	-
0	1	0	1	-	+		
0	1	1	1			-	+

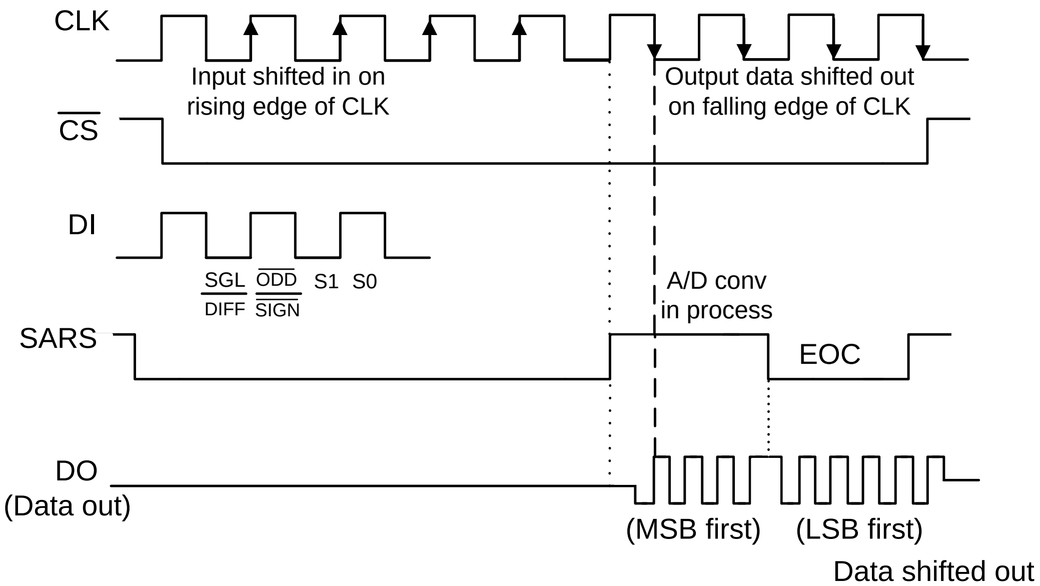


Fig. 6-25 Operation principle of ADC 0838 Serial S/D converter

Chapter 7 Cognition and Behavior-based Control

"The concept cognition is a term that can be contributed to the effect of the perception-action connection, but can not be located, spatially or functionally."

The world

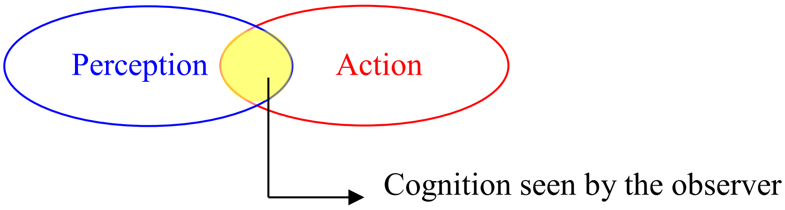


Fig. 7-1 The concept of cognition or robotic intelligence

"Intelligence" is in the eye of the observer. It arises from the interaction between perception and action.

Two important concepts in behavior-based robotics(BBR).

1) Situatedness

"The robot are situated in the world, it needs to sense its current surroundings and avoids to use abstract representation. It deals with the "here" and "now" of the environment which directly influences the behavior of the robot."

2) Embodiment

The robots are physical creatures and thus experience the world directly, their actions are determined through interaction with the real world and their actions have feedback on the robot's own perception.

Behavior-Based Robotics

Prof. Rodney Brooks proposed the subsumption architecture in mid-1980's. He proposed the use of a layered control system, embodied by the subsumption architecture.

Manipulate the world
Build maps
Sensors → Explore → Actuators
Avoid obstacles
Locomotion

This approach is a purely reactive behavior-based method. Task-achieving behaviors in the subsumption architecture are represented as separate layers. Individual layers work on individual goals concurrently and asynchronously Stimulus (inputs) or Responses (outputs) of a behavior module can be suppressed or inhibited by other active behaviors. High-level behaviors have the power to temporarily suppress lower-level behaviors.

Subsume: "to think about an object as taking part of a group"

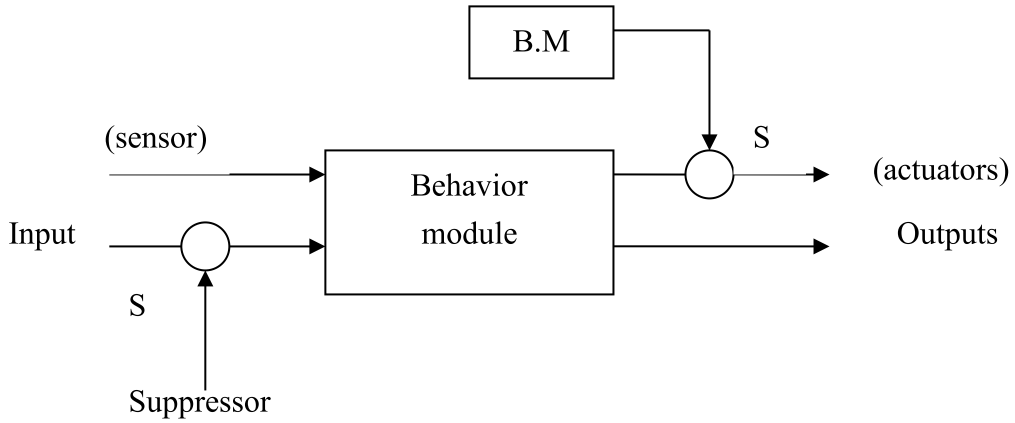


Fig. 7-2 The concept of subsunption system

Remark: For the problem of conflicting behaviors, fusion is performed at the output of behaviors (behavior fusion), rather than the output of sensors.

Compare with traditional robotic systems.

1. Traditional robot control programs are based on the ideas of world modeling and planning. This approach decomposes a robot program into a sequence of functions.

   Sensors → sensing | modeling | planning | execution | motor control | → actuators

   SMPA: sensing, modeling, planning, action; various sensor data are collected to obtain an internal representation, a plan is made based on the representation and then executed using an actuators.

- SMPA is a sensor fusion approach (at the input). But in behavior-based programming we use behavior fusion (at the output).

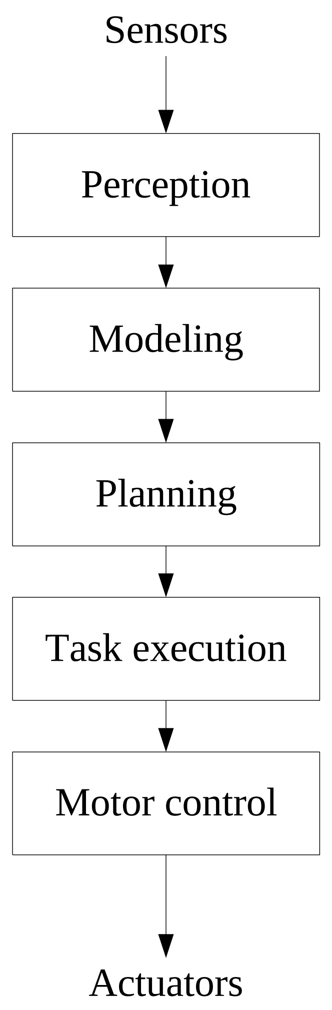


Fig. 7-4 Traditional robot programming approach: SMPA

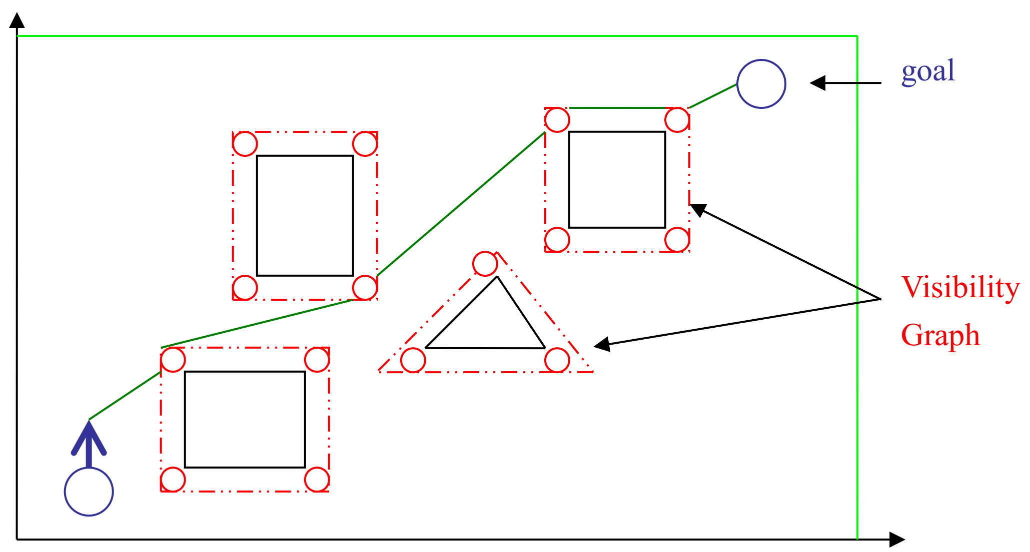


Fig. 7-4 Path planning using visibility graph

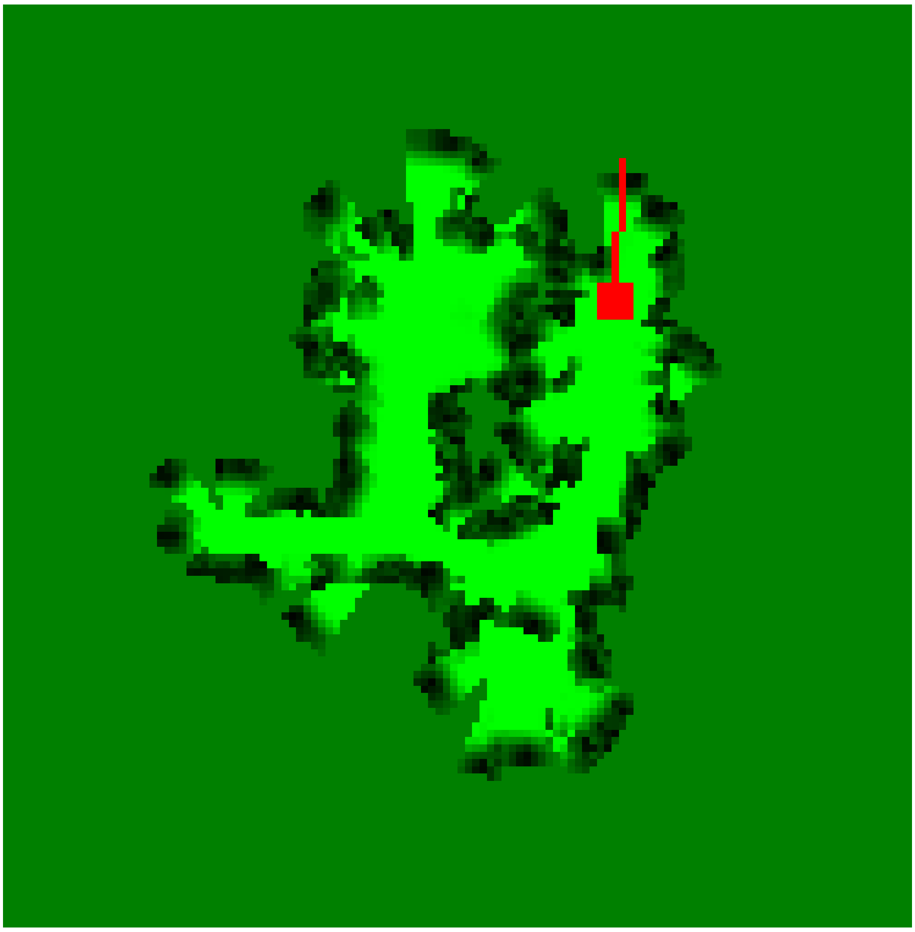


Fig. 7-5 A map generated by ultrasonic sensors

Brooks argued that the sense-plan-act paradigm was in fact detrimental to the contribution of real working robots.

Because:

1) computation

World modeling require large data storage and intense computation.

2) modeling

Reliable planning, an accurate global model is required actually. The world model is never complete and accurate.

3) time

Sensing/modeling/planning paradigm is by natural sequential, more time is required to complete the process longer time delay between sensing and action.

An example of behavior networks.

A mobile robot equipped with a ring of sonar sensors, an infrared detection system and a small μ-controller is programmed to avoid bumping into objects.

1) A subsumption program consists of three parts.

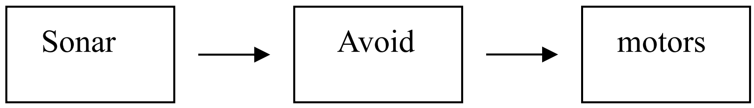


Fig. 7-6 The “Avoid” behavior

2) A second behavior: Dock

3) Battery level sensing.

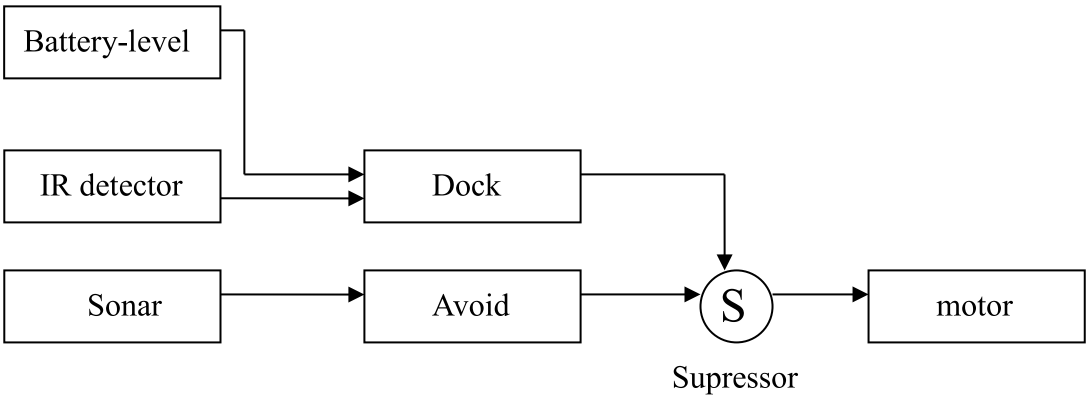


Fig. 7-7 An example of subsumption system

4) Dock subsumes the function of Avoid in order to produce a higher-level of competence. This style of robot programming, where the robot’s control system is decomposed into a network of task-achieving behaviors is the essence of subsumption architecture.

A more complex example.

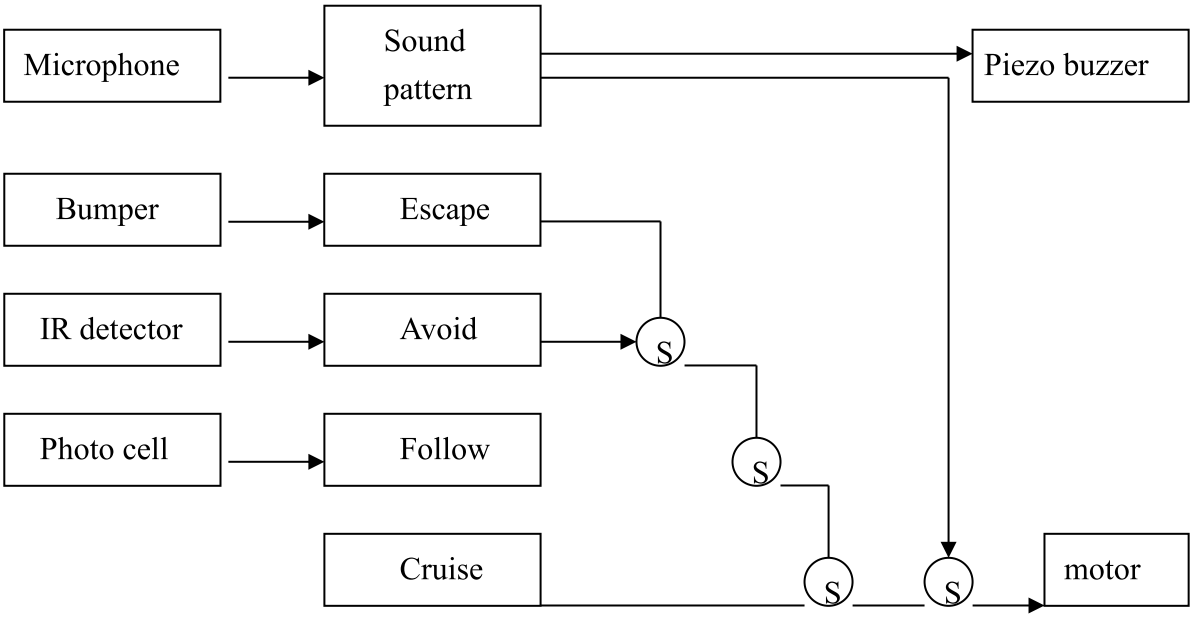


Fig. 7-8 A more complete example of behavior-based design

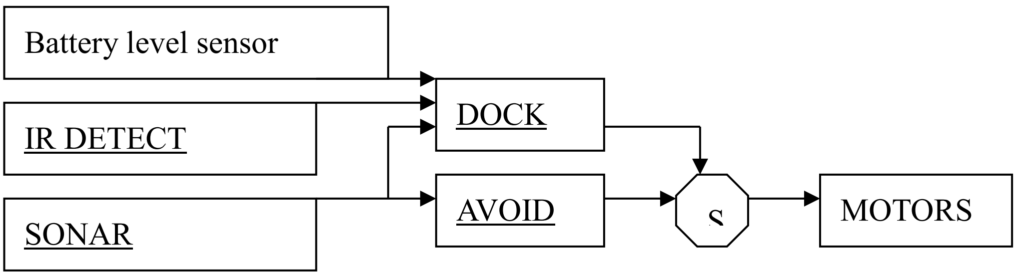


Fig. 7-9 A mobile robot with survival behaviors

Implementation:

How do we implement a network of many behaviors, all running in parallel on a small micro-processor that is inherently a sequential machine? (Multitasking, each process is allowed to compute for a short time at a regular interval.)

```
Void multi-flash()
{while(1){       /*loop forever
flash-led();
sleep(1,0)    /*do nothing for 1 sec
}}
```

- Process and schedulers

We need many behaviors to run in parallel, multi-flash should be activated in such a way that it does not consume all the procedures of the micro-process. (Make multi-flash into a "process".)

A process or a task is a piece of code that can be though of as running simultaneously with other process or program. A supervisory program called a scheduler is responsible for giving the appearance that different pieces of code are running in parallel.

Finite-state machine (FSM) (AFSM)

An effective approach for passing control between the process and the scheduler (and the approach employed in subsumption implementation) is to implement each process as a finite-state machine.

A finite-state machine (FSM) is a abstract computation element which is composed of a collection of states. In the absence of a sophisticated scheduler, it is possible to build a subsumption system by implementation the behaviors as FSMs.

Ex:

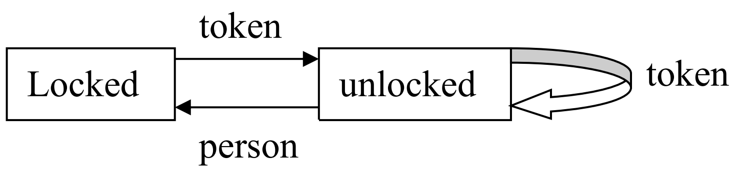


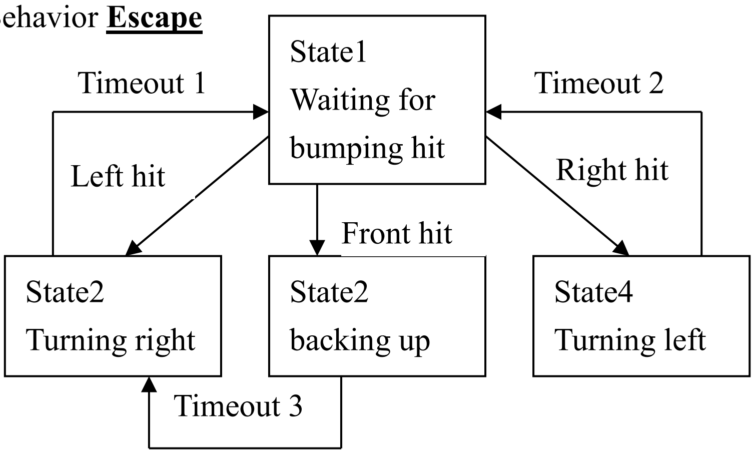
Fig. 7-10 An example of finite state machine

Giving a particular input, a FSM may change to a different state or stay in the same state. The specification of an FSM includes rules that determine the relationship between inputs and state change.

Design of behavior controller

1) Start with small set of non-overlapping survival behaviors. E.g. obstacle avoidance.
2) Specify the behavior in terms of actions. E.g. obstacle avoidance has a sub-goal of moving away from obstacles.
3) Specify the actions in terms of the robots actuators.

Behavior **Escape**



4) Complex behaviors are implemented as combination of simpler behaviors.

Autonomous robot control strategy(approach)

Basic strategies:

1) planners
2) behavioral(reactive)
3) hybrid controls

Eg.

Complex behaviors: Foraging

Simpler behaviors: safe wondering
find punk
capture punk
home in angle

- Planner

Typically use a world model plus sensor inputs to generate a sequence of actions. i.e. a plan.

a. AI techniques are often used. e.g. A*-algorithm.

b. works well in a well known and stable environment. E.g. visibility graph.

c. often fails, in practical applications--cannot handle unexpected events

d. excessive computation time.

e. sequential programming: a simple type of planner used in robotic contest.

A sequence of actions are coded into the robot's memory.

A world model implicitly programmed into the robot.

Drawback: the robot can become disoriented robot needs to be able to re-orient itself.

Reactive controller

Action depends on the state of sensors, no need of a world model.

*very fast reaction, little computation required.

(Eg. fuzzy-neuro mapping)

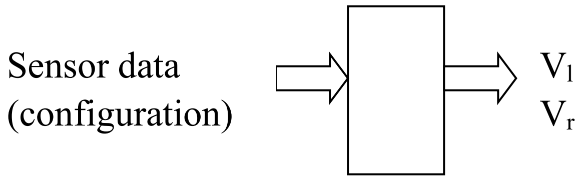


Fig. 7-11 Concept of reactive behaviors

Prof. Mataric of USC.

Differences between reactive control and behavior-based control:

1) While behavior-based systems embody some of the properties of reactive systems, and they contain the reactive components, their computation is not limited to loop-up.
2) Distributed nature of behavior based control they consists of a collection of parallel, concurrent executing behaviors, devoid of a centralized arbiter or reasoner.

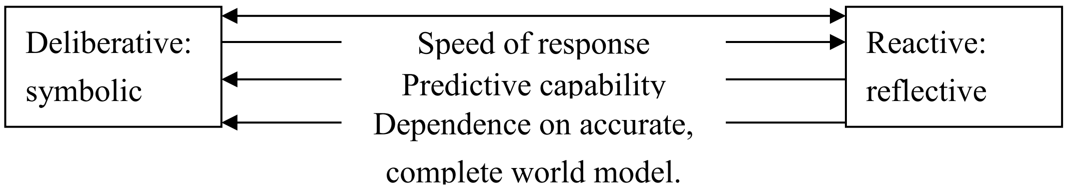


Fig. 7-12 Comparison of deliberative and reactive robotics

- Behaviors are defined as independent, actuator command producing modules, each with their own access to the sensors.
- A fitness measure is determined for each behavior.
- Some coordination scheme using the fitness measures combines the output to the different behaviors.

Behavior-based strategy

1) subsumption
2) scheme-based control

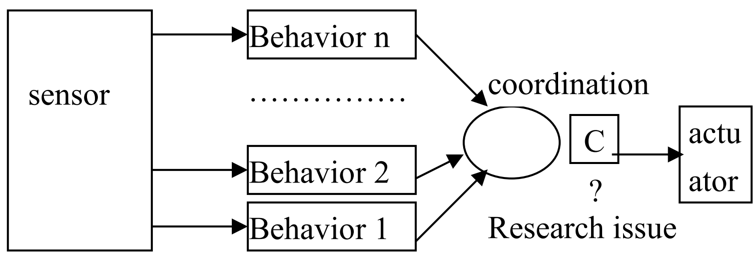


Fig. 7-13 the structure of behavior coordination

Hybrid controller

- Combination of planner(for high-level planning) and reactive controller(for survival tasks).
- Usually use a third system to arbitrate between the planner and the reactive controller.
- Planner still requires long computation time.

For the problem of conflicting behaviors, fusion is performed at the output of behaviors (behavior fusion), rather than the output of sensors.

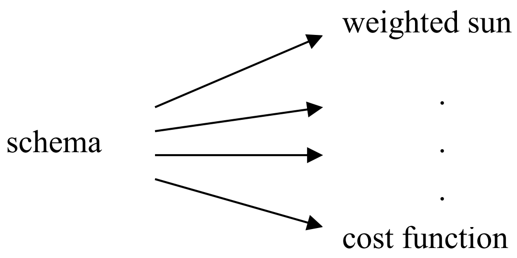


Fig. 7-14 The concept of behavior coordination

Ex. Force-field (like potential field) robot navigation

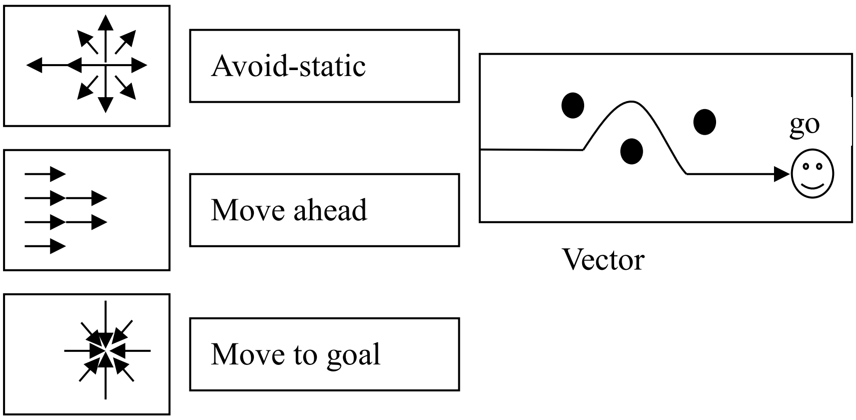


Fig. 7-15 An example of Schema-based design

Chapter 8 Fuzzy Control and Behavior Fusion

1) Human reasoning : normally not in a precise manner. For instance:
   "It's dangerous to drive on a very icy road."
   But computer programs make decisions in a clear-cut manner. For instance:
   "Does the temperature exceed 75.6℃?"

How to program a computer to reason in a way similar to human?

⇒Fuzzy sets.

Fuzzy sets are sets that do not have a crisply defined membership, but rather allow objects to have grades of membership from 0 to 1.

2) Fuzzy logic :

Created by Prof. Zadeh in 1965.

A method to draw definite conclusions from ambiguous or imprecise data (FUZZY).

- Is a man old (what age is old)?
- Is a woman beautiful?
- Go faster or slower (at what speed?).
- An object is at right side, turn left (how many degrees?)

Precision is not necessary for many problems.

Fuzzy logic is based on human though process.

3) Fuzzy Control :

Advantages : reduced development time.
Simple approach for complex problems.

Disadvantages : can not guarantee stability.
need to develop a rule set.

4) Fuzziness and Imprecision

A. Fuzzy sets allow computer to use the type of human knowledge called "common sense".

B. With fuzzy set, the programmer can translate the linguistic values into a computer program by providing the membership function that defines them.

C. Fuzzy set programming avoids the rigidity of conventional mathematical reasoning and computer programming.

5) Fuzzy and Probability (describe and deal with uncertainty)

A. Probability means a kind of uncertainty of some event which will happen or will not happen.
B. Fuzziness deals with an existent event for which there is certain uncertainty of description.

6) Fuzzy Control Method :

Continuous				Inference		
or	$\rightarrow$	Fuzzification	$\rightarrow$	with	$\rightarrow$	Deduzzification
Discrete data (IN)				Fuzzy logic		Continuous data (OUT)

6.1 Fuzzification : Continuous variable $\rightarrow$ Fuzzy variable.

Age in year $\rightarrow$ young, middle-aged, old.

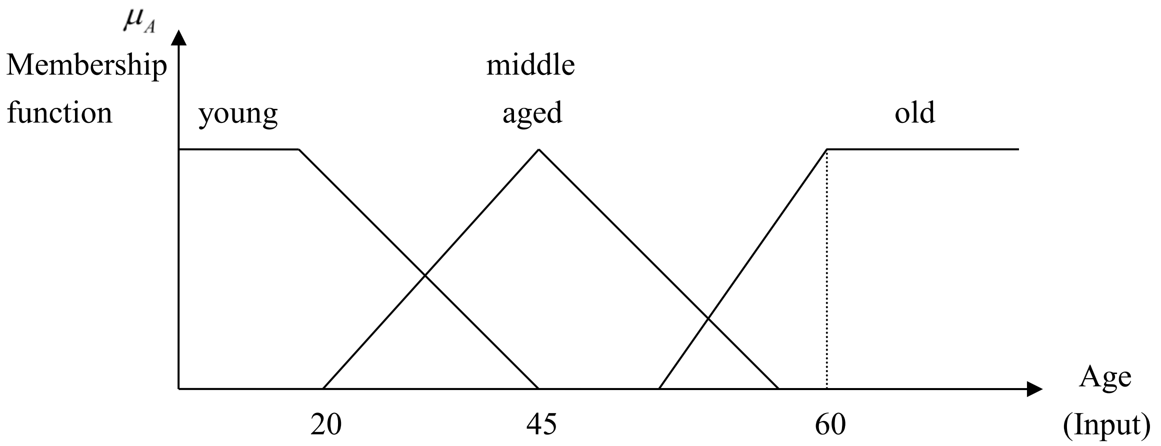


Fig. 8-1 The membership and membership function

Ex:

$$\mu_{young} = 0.25$$

$$\mu_{middle-aged} = 0.8$$

$$\mu_{old} = 0$$

Get partial membership in fuzzy sets.

Membership functions

- Designer need to find suitable membership functions for fuzzy variables.

$\rightarrow$ Can be difficult to determine.

Common membership functions

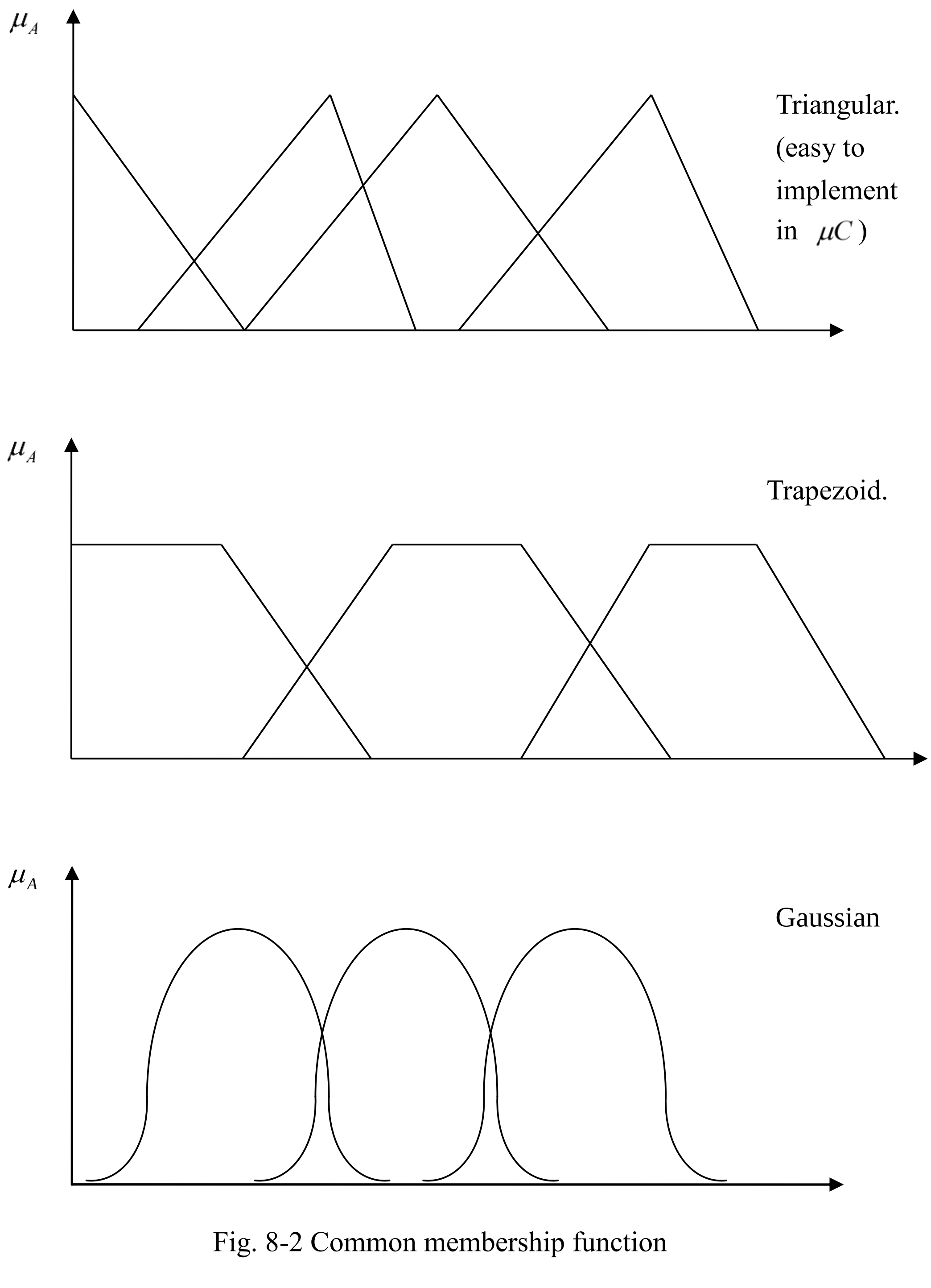


Fig. 8-2 Common membership function

Can use a template of predefined membership functions for each problem and adjust output by changing rules.

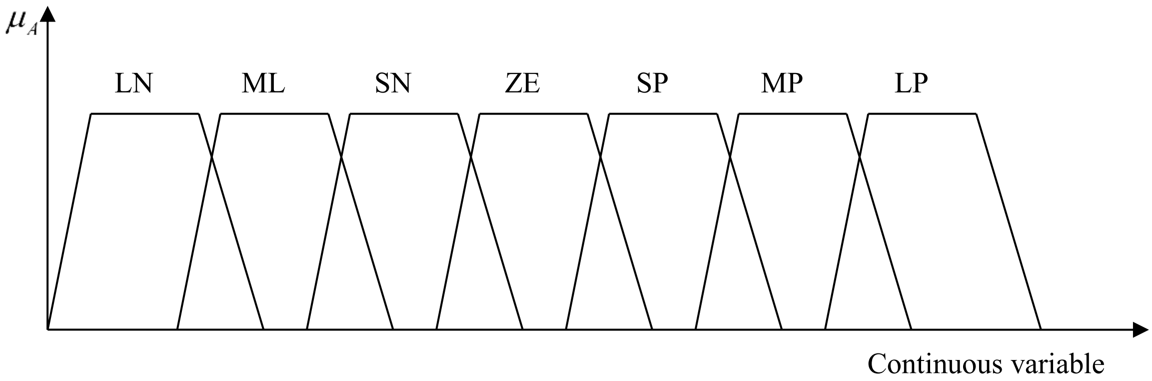


Fig. 8-3 A template for membership function

6.2 Inference $\rightarrow$ If … Then … (evaluating rules)

Rules are from expert, common sense, trial and error or from data.

Basic operations

NOT : $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$, negation

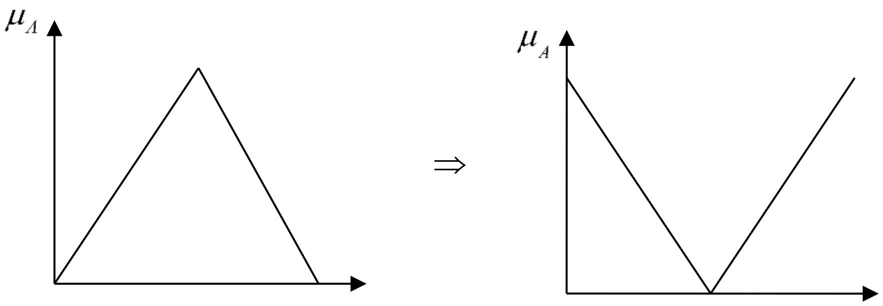
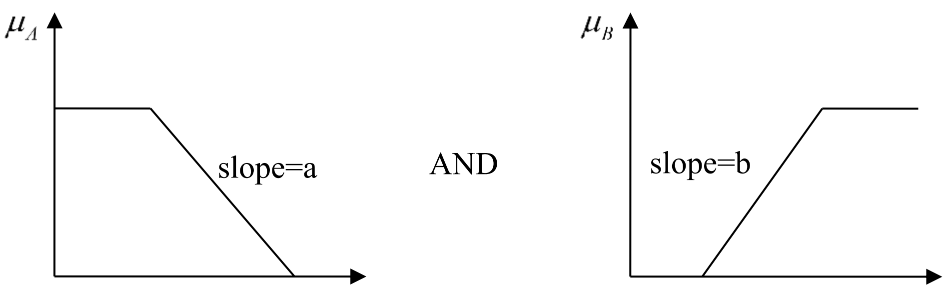


Fig. 8-4 Negation operation

AND : $\mu_{A \cap B}(x) = \min[\mu_A(x), \mu_B(x)]$, conjunction



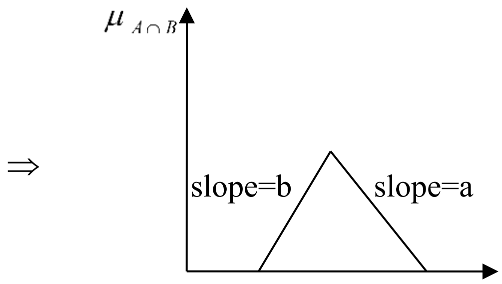


Fig. 8-5 AND operation

OR : $\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)]$, disjunction, intersection

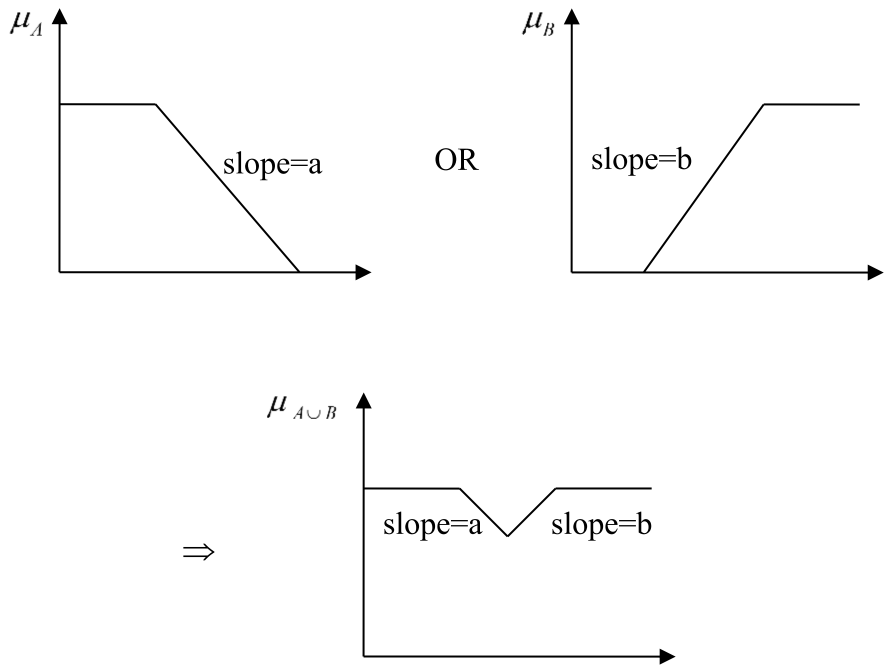


Fig. 8-6 OR operation

6.3 Rules : methods of inferencing

1) rule : If animal is a dog them it has four legs.

2) premise (前提) : The animal is a dog.

3) conclusion : The animal has 4 legs.

* If premise (antecedent) is true, then the conclusion (consequent) is true.

6.4 Fuzzy Logic : Premise has a certain strength (μ_A), conclusion is true with the same strength as the premise.

Modify conclusions membership function to reflect the strength of the premise.

A. Method 1, minimum inference (truncation)

Let μ_A = premise membership function

μ_B = conclusion membership function

$$\mu_{B,A}(x) = \min[\mu_A, \mu_B(x)]$$

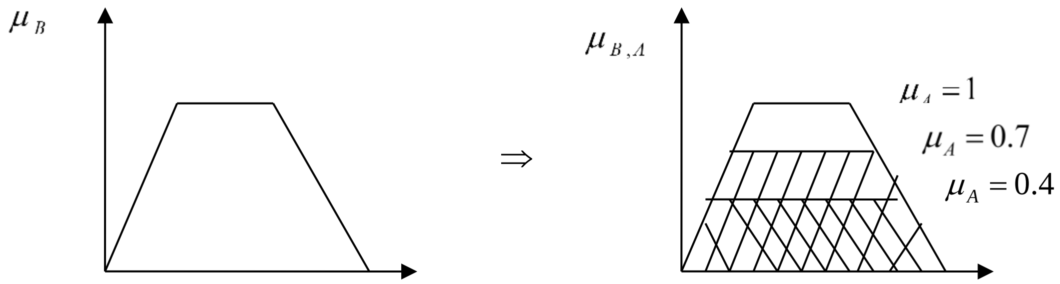


Fig. 8-7 Minimum inference

B Method 2, product inference

$$\mu_{B,A}(x) = \mu_A - \mu_B(x)$$

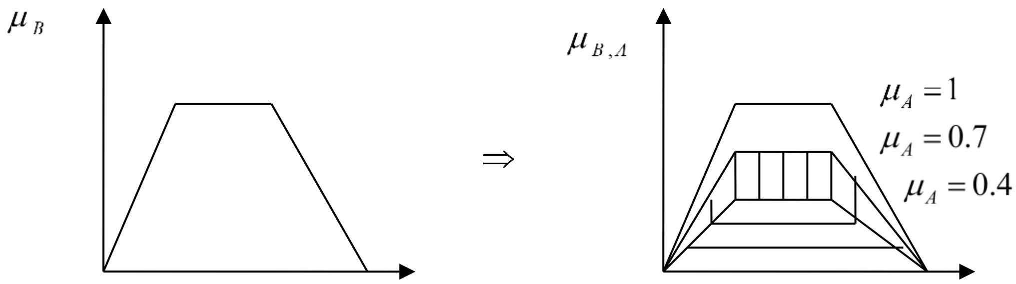


Fig. 8-8 Product inference

C. Fuzzy reasoning

For simplicity, assume that we have two fuzzy control rules :

R_1: If x is A_1 and y is B_1, then z is C_1.

R_2: If x is A_2 and y is B_2, then z is C_2.

Mamdanc's Minimum Operation Rule

For 1st rule,

$$\alpha_1 = \mu_{A_1}(x_0) \wedge \mu_{B_1}(y_0)$$

$$\alpha_2 = \mu_{A_2}(x_0) \wedge \mu_{B_2}(y_0)$$

The i^{th} rule leads to the control decision :

$$\mu_{C_1}(w) = \alpha_1 \wedge \mu_{C_1}(w)$$

which implies that the membership function μ_C of the inference consequence C is point wise given by

$$\mu_C(w) = \mu_{C_1} \vee \mu_{C_2}$$

$$= [\alpha_1 \wedge \mu_{C_1}(w)] \vee [\alpha_2 \wedge \mu_{C_2}(w)]$$

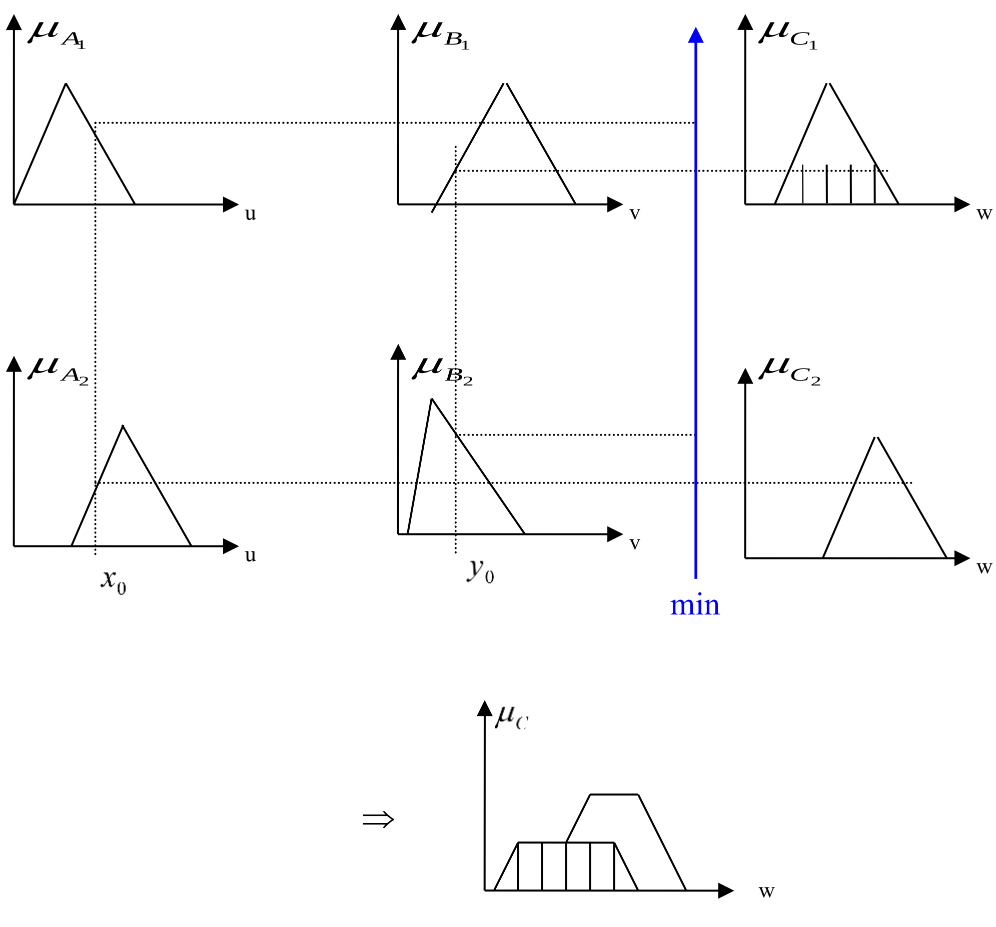


Fig. 8-9 Mamdani's minimum operation rule

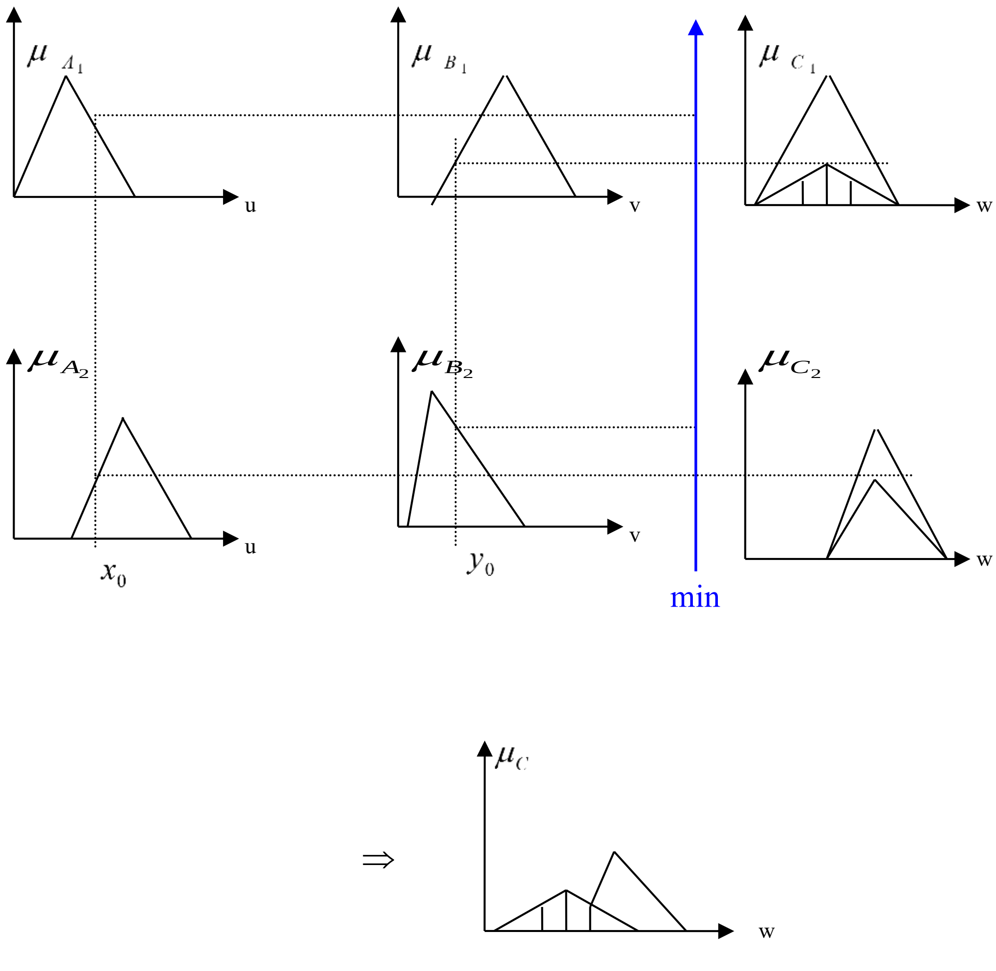
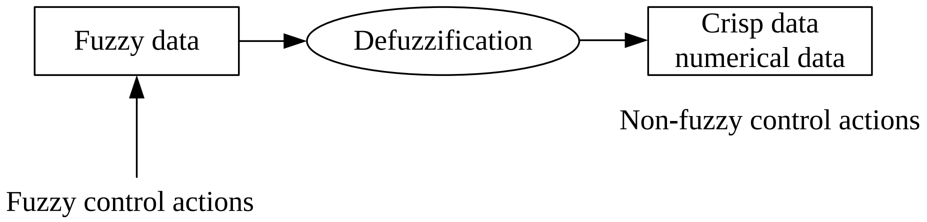


Fig. 8-10 Larsen's product operation rule

6.4 Defuzzification



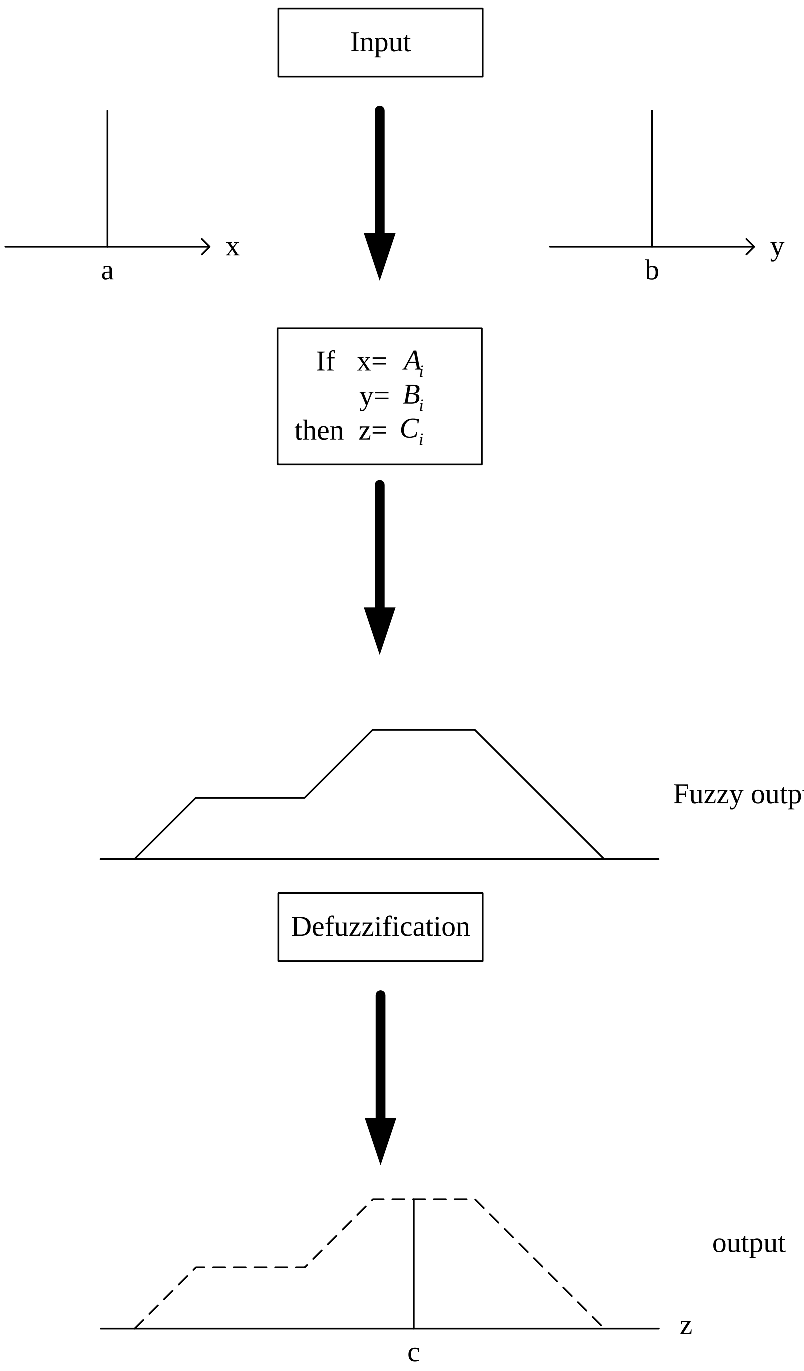


Fig. 8-11 A method of defuzzification

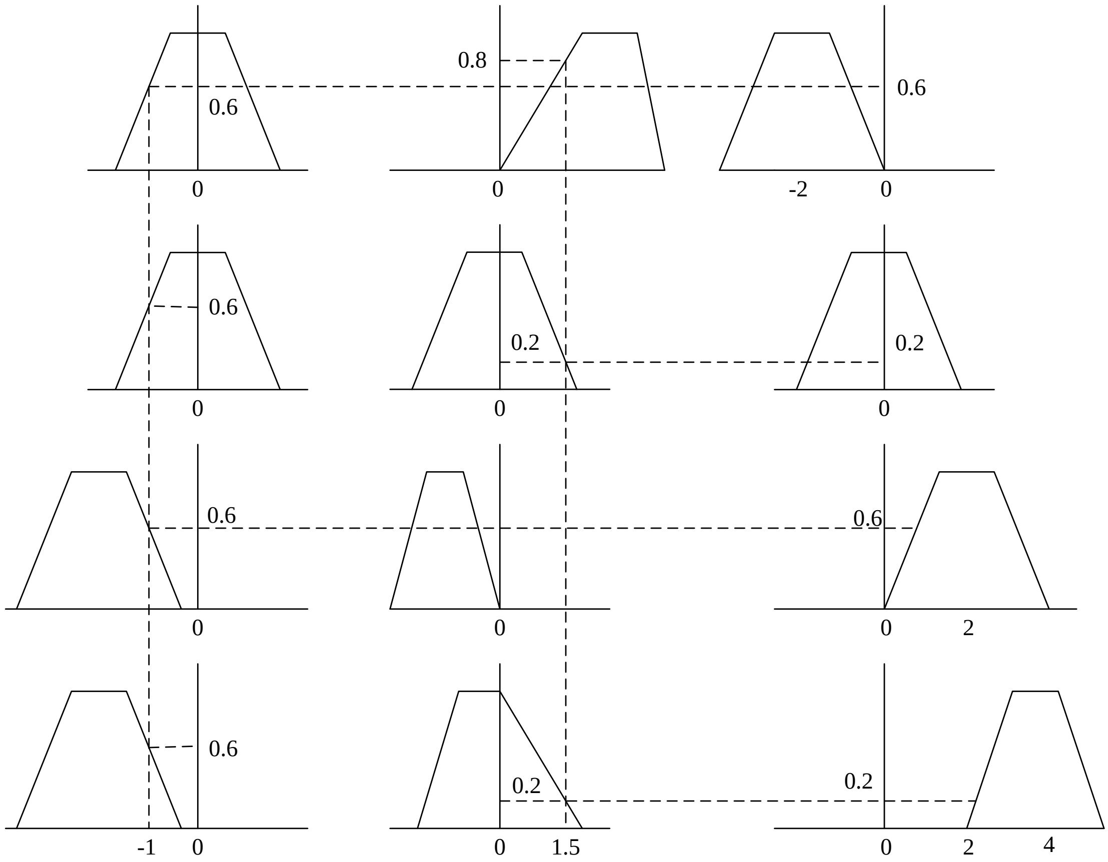
The Center of Area Method (CoA)

$$z_0 = \frac{\sum_{j=1}^{n} \mu_z(w_j) w_j}{\sum_{j=1}^{n} \mu_z(w_j)}$$

n : the quantization levels of the output.

Also : center of gravity (centroid) of the area. (The distribution of a control action)

An example is illustrated below:



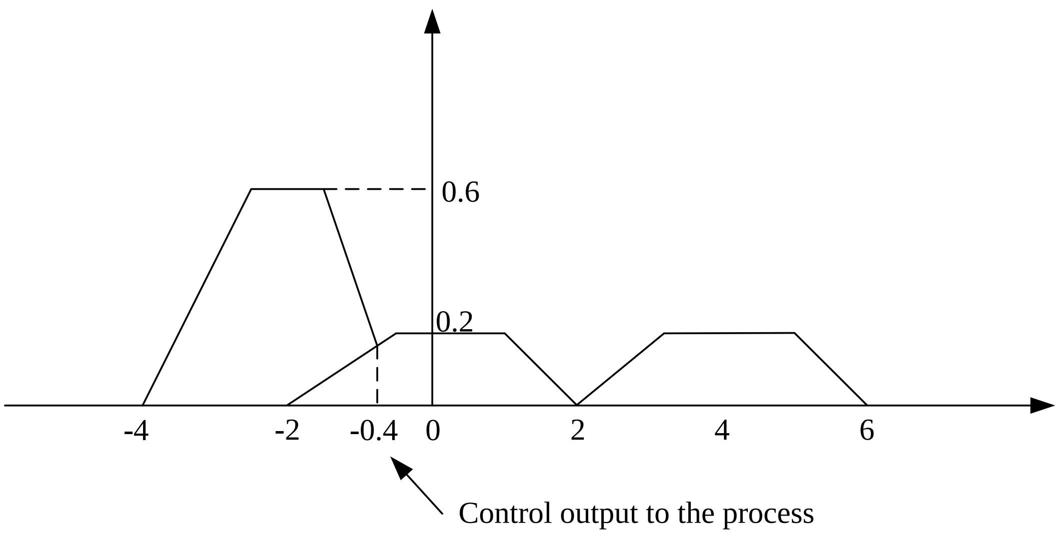


Fig. 8-12 Defuzzification by center of gravity (Centroid) method

$$z_0 = \frac{0.6\times(-2)+0.2\times 0+0.2\times 4}{0.6+0.2+0.2} = -0.4$$

7). A practical system

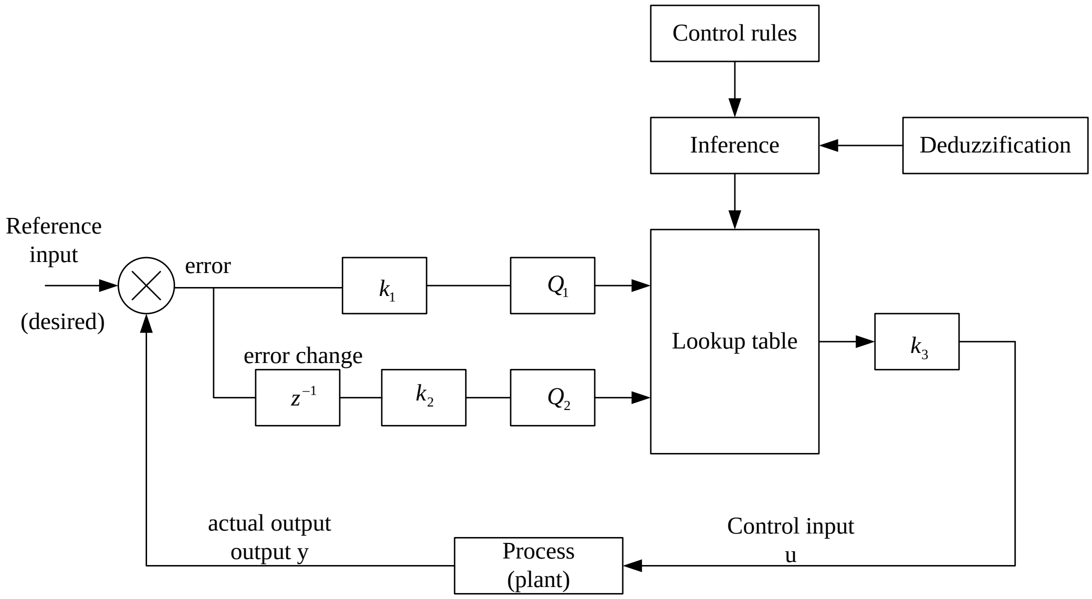


Fig. 8-13 Implementation of a fuzzy control system

8). Mobile robot navigation using Fuzzy Control

- Mobile robot needs to handle a dynamic changing environment with static and moving objects.
- Obstacle avoidance design using imprecise (incomplete) environmental information.
- There is no perfect sensor, sensor data are uncertain, the capacity of onboard sensors is limited.
- "Common sense" can be good enough for navigation in an indoor environment.

Does the robot need to know the distance to an object with a value 1.15 meter?
Far or near is used in common sense to drive a vehicle.

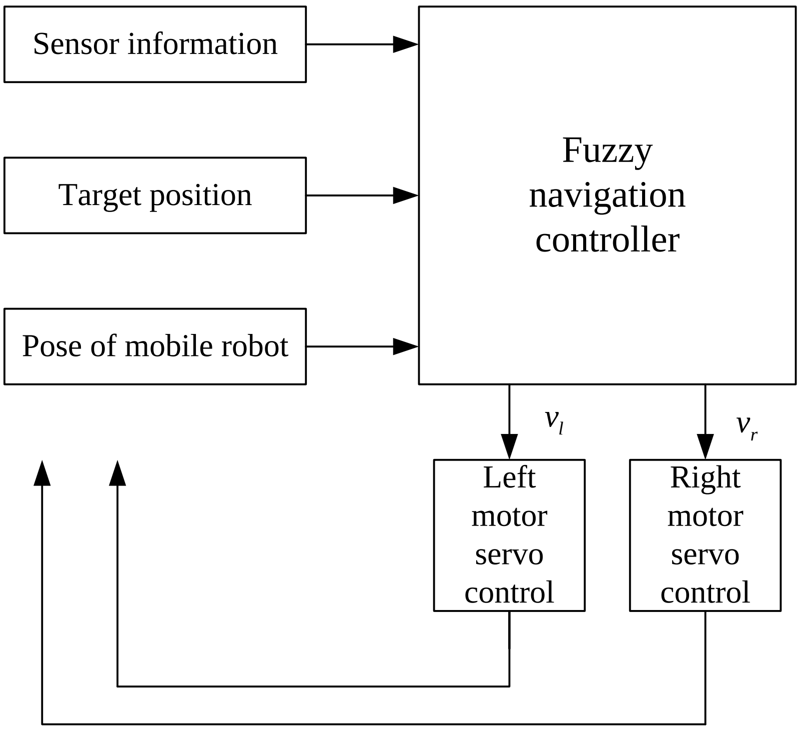


Fig. 8-14 Control structure of a fuzzy navigation system

Chapter 9 Power and Batteries

Converts chemical energy to electrical energy.

- Properties

  1) Rechargeability

     Not rechargeable – primary battery

     Storage battery

  2) Energy Density

     The maximum energy per unit mass Watt-hours/kg (Wh/kg).

  3) Capacity

     Energy stored in cell, usually in mA-hrs. (some A-hrs).

     Capacity = energy density x mass of battery.

  4) Voltage

     Voltage of a single cell depends on type of battery, voltage also depends on the state of charging of the cell.

  5) Internal resistance

     Determines maximum current a battery can supply.

  6) Shelf life

     How quickly a battery discharge with no external load.

For mobile robot, three choices of rechargeable batteries :

Type	Energy density	Cell voltage	Internal resistance	Type capacity
Lead-Acid	40 Wh/kg	2.0V	0.006Ω	1.2-120Ah (for large robot)
NiCd (memory effect)	38 Wh/kg	1.2V	0.009Ω	500mAh 1800 4000 (環保)
NiMH (nickel-metal-hydride)	57 Wh/kg	1.3V	?	1100 2300 (expensive)

- gasoline has > 400 x energy density of NiCds (~80 x if gasoline converted to electricity)
- batteries can be cost effective despite the high absolute costs of the energy they supply.
- Fuel cell, or other type of power supply will become available in the future.

Voltage : depends on states of charge

NiCd : ~1.2V when fully charged.

1.1V when at 10-20% of capacity.

Lead-acid : 2.1V when fully charged.

1.8V when at low capacity.

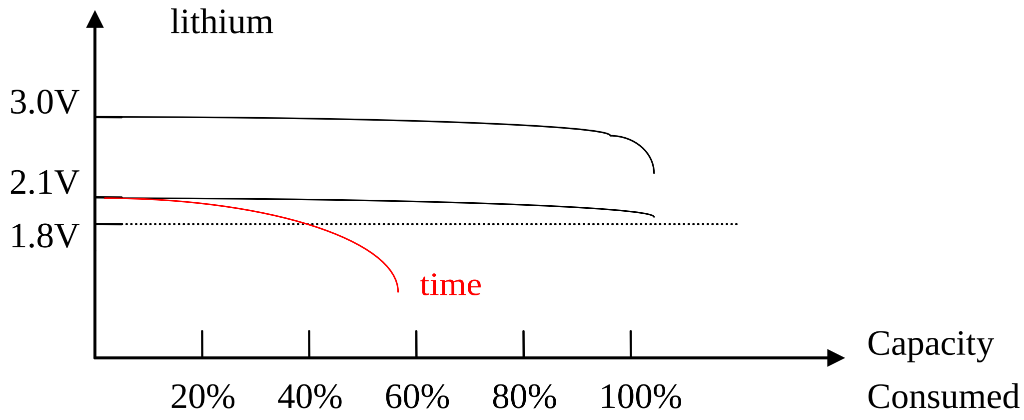


Fig. 9-1 The voltage of a battery cell

Capacity : Amp-hours (?)

mA-hrs,

mA-hrs x voltage = energy (mW-hrs).

rated capacity under favorable conditions.

Only about 60%-80% is available under rapid discharge conditions.

Internal resistance :

If the positive and negative terminals of a battery are shorted together, the current is limited by the internal resistance of the battery.

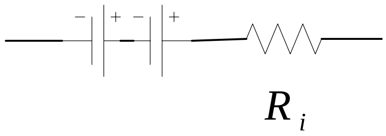
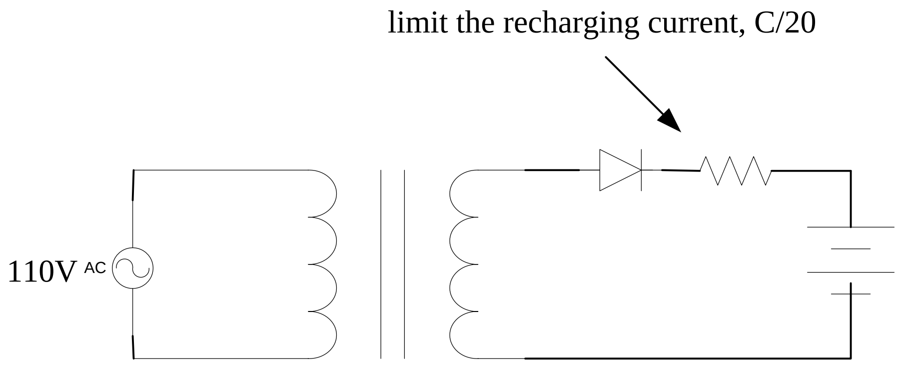


Fig. 9-2 The battery cell model

NiCd cells have small R_i

Despite its lower energy density, NiCd is more suitable for application of high surge current. But more dangerous to use.

If NiCd battery is short-circuited, the high current may melt insulations and cause fire.



Charging circuit (IC chips)

Fig. 9-3 Recharging of the battery

Power regulation :

Battery voltage changes with state of charge.

Want constant voltage for IC power supply.

Want constant voltage as load charges (or current demand changes).

May want several voltages from single battery.

Linear regulators : LM7805 -5V regulator.

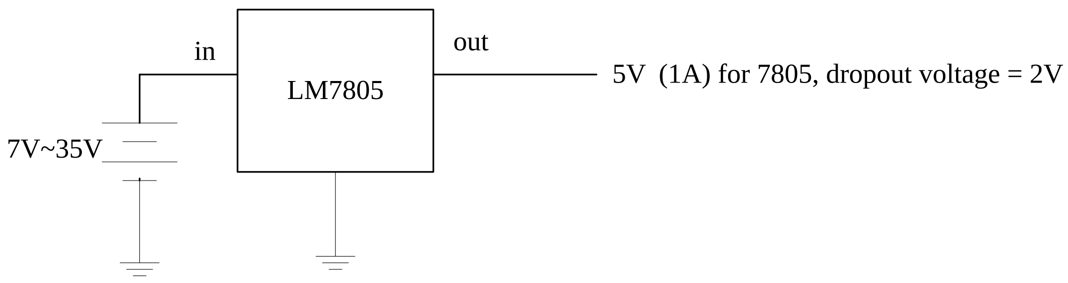


Fig. 9-4 Voltage regulator

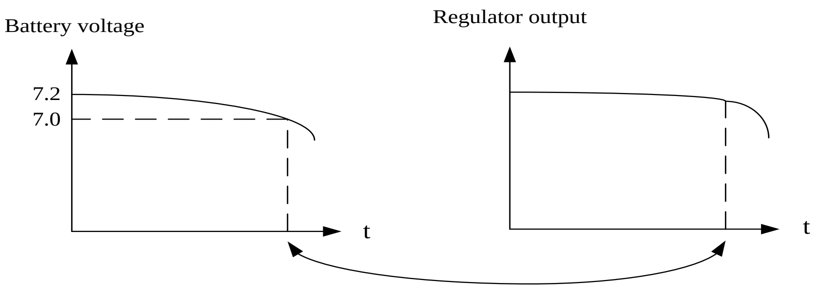


Fig. 9-5 Battery cell voltage and regulator output voltage

Can have 0.2V drop in battery voltage before regulator output charge!(Basic stamp will work to ~4.5V)

As long as the input voltage is higher then the required output voltage by a certain amount (dropout voltage). The output voltage will be constant.

This may correspond to a cell has only used up a small portion of battery capacity. (1.2V cell x 6 = 7.2V, each battery cell will still have 1 volt when used up.)

Can use 8.4V battery, but this will increase power loss.

Power loss : $P = I(V_{in} - V_{out})$

V_{in} : supply voltage

V_{out} : output voltage

I : current output

P : power dissipated by the linear regulator

Circuit uses $5.0\,I$

Regulator consumes : $3.4\,I$

$\Rightarrow$ Linear regulator with a smaller dropout

LM2940CT - 5.0 (more expensive)

If voltage higher than the battery voltage or a voltage with a polarity opposite that of the battery is required $\rightarrow$ DC – DC converters.

Isolation

- Power supplier noisy : changing state of digital IC, motor brushes, motor starting, stopping ; $\rightarrow$ current/voltage spikes on power supply.
- Noise from external electrical/magnetic fields

1) Set power supply between logic circuit and motor.

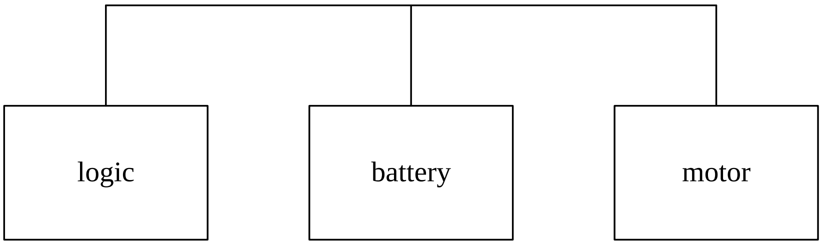


Fig. 9-6 Isolation of logic circuitry and power stage circuitry

2) Single point ground. (avoid ground loop)

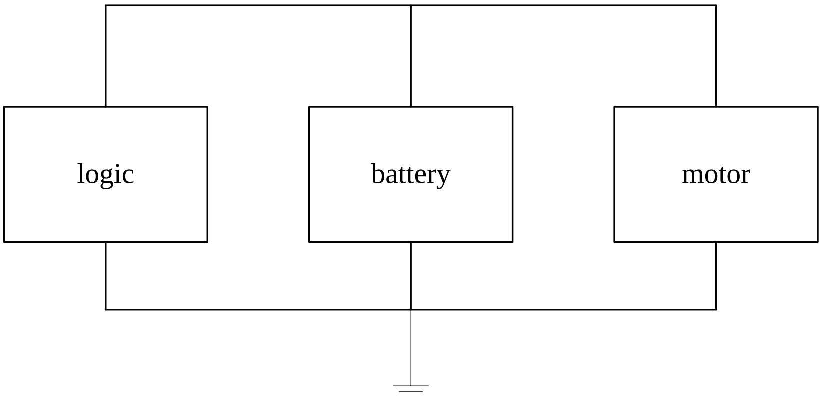


Fig. 9-7 Single point grounding connection

3) Connect small capacitors (0.1 μF), across the power and ground of each digital chip, to combat the state charge transient noise.

4) To deal with motor switching, starting, stopping large current spikes,

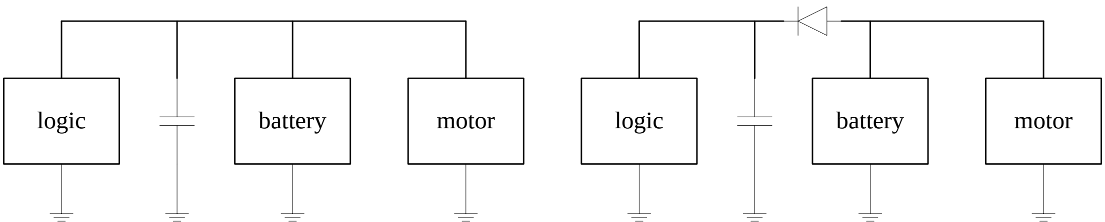


Fig. 9-8 Protection from motor spikes

a. a large (10 μF) capacitor protects the other circuit from the motor spikes.
b. Diode prevents reverse polarity spikes from motor starts/stops from getting into logic supply.

5) Opto-isolator

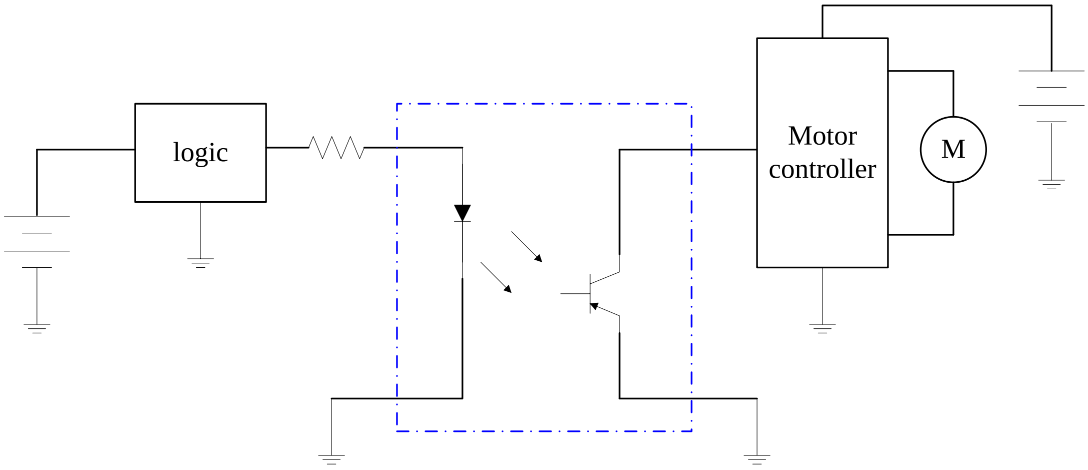


Fig. 9-9 Use of optical isolation

Part II laboratory

After understanding the theory, it is time to implement it. First, the system software is carried out. The OS images is flashed to the SD card by using OS image flasher. Please install the ubuntu 18.04 operating system on the embedded development board, Raspberry Pi, and then install the ROS melodic. The ROS system is used for communication and the mobile robot will be controlled via ROS in all labs.

After completing the installation of Raspberry Pi and ROS, connect the L298N motor driver module to the motor for controlling the speed of the motor. Through the communication of ROS, the Raspberry pi sends commands to the Arduino, and the Arduino sends the PWM control signal to the L298N. Finally, the control of the front, back, left and right movements of the mobile robot is completed.

When the car is moving, it must have the ability to avoid obstacles and know the moving target point. Therefore, the sensor is added to make the mobile robot have the function of avoiding obstacles and tracking the target point. Through the design of data acquisition and processing, the processed information is obtained to control the motion of the mobile robot. Use touch sensor and photo resistor sensor to obtain environmental information, and control the motion of the mobile robot according to the processed information, so that the mobile robot can avoid obstacles and track the position of the light ball.

Finally, through the design of Cognition and Behavior-based Control, the IR receiver sensor is integrated, the sensing data is calculated by the embedded computer, the processed information is obtained, and the motion of the mobile robot is determined through the decision of the state machine, so that after the mobile robot finds the light ball, it can send the light ball to the designated position to achieve the required task.

After completing all the checkpoints, you can participate in the robot hockey contest and demo your unique mobile robot.

Checkpoint #1 Raspberry Pi and ROS

- **Purpose:**

  setting the development environment for robot and using ROS system.

- **Task1:**

  Setting up the environment and remote control the Raspberry Pi

  1. Install Ubuntu mate 18.04 in Raspberry Pi
  2. Install ROS Melodic in Raspberry Pi
  3. Using the Raspberry Pi by remote control it

- **Task2:**

  Using ROS to communicate

  1. Communicate between Raspberry Pi and Arduino by using ROS

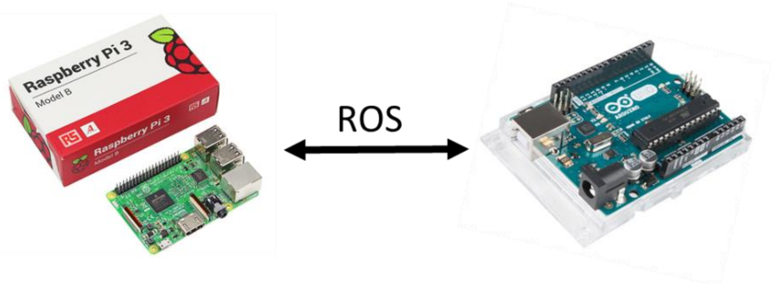
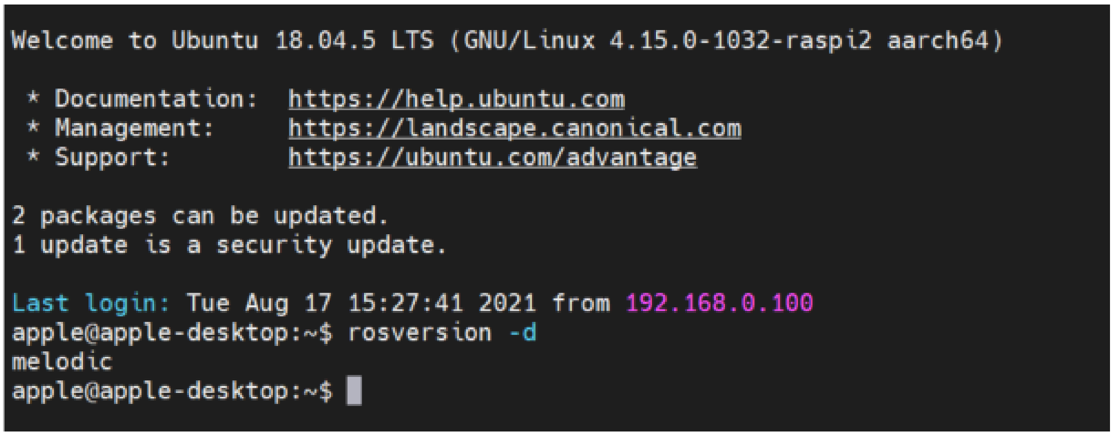


Figure 1 Raspberry Pi and Arduino

- **Materials list**

	Material	Number
1	Raspberry Pi	1
2	Arduino UNO	1
3	16g SD card	1
4	USB A to B	1
5	USB A to micro B	1
6	Power back	1

Task1

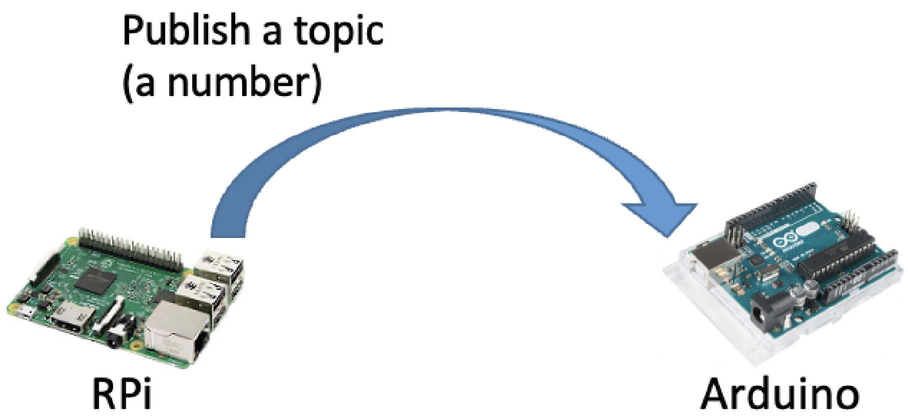


```
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-1032-raspi2 aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

2 packages can be updated.
1 update is a security update.

Last login: Tue Aug 17 15:27:41 2021 from 192.168.0.100
apple@apple-desktop:~$ rosversion -d
melodic
apple@apple-desktop:~$
```

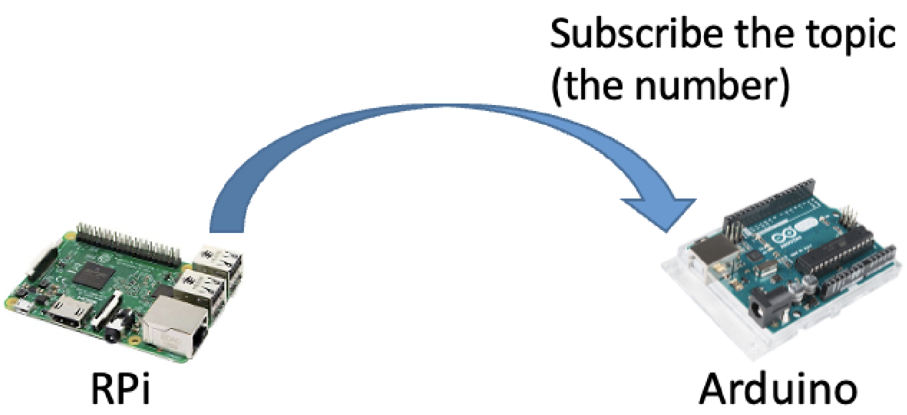
Figure 2 Task1 result

Task2

Step1



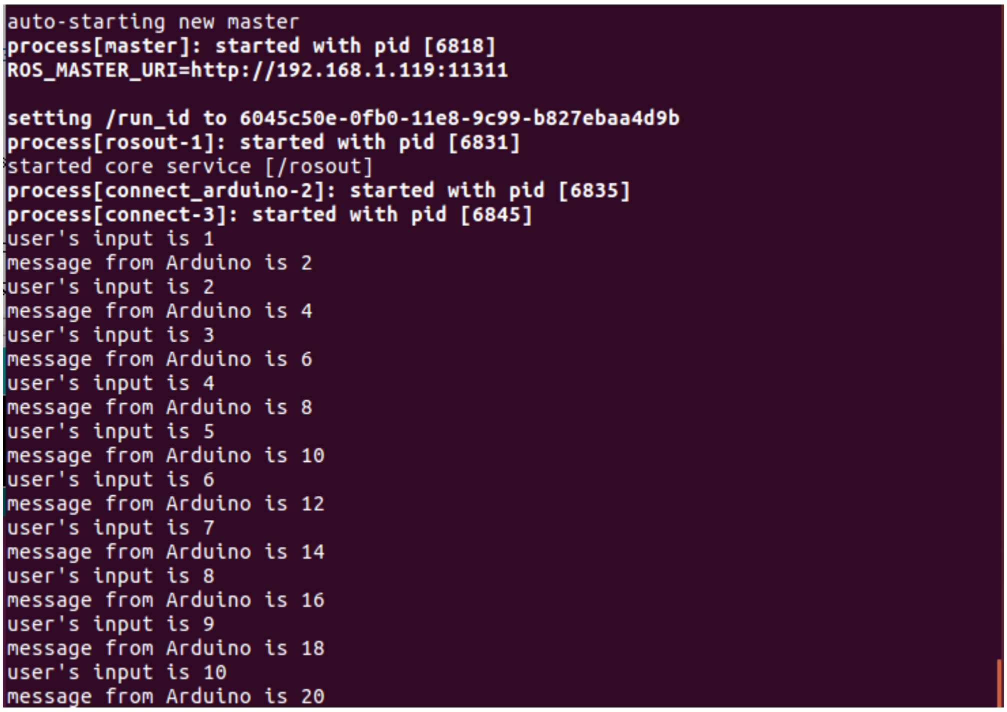
Step2



Step3

Step4

Task2 Result



```
auto-starting new master
process[master]: started with pid [6818]
ROS_MASTER_URI=http://192.168.1.119:11311

setting /run_id to 6045c50e-0fb0-11e8-9c99-b827ebaa4d9b
process[rosout-1]: started with pid [6831]
started core service [/rosout]
process[connect_arduino-2]: started with pid [6835]
process[connect-3]: started with pid [6845]
user's input is 1
message from Arduino is 2
user's input is 2
message from Arduino is 4
user's input is 3
message from Arduino is 6
user's input is 4
message from Arduino is 8
user's input is 5
message from Arduino is 10
user's input is 6
message from Arduino is 12
user's input is 7
message from Arduino is 14
user's input is 8
message from Arduino is 16
user's input is 9
message from Arduino is 18
user's input is 10
message from Arduino is 20
```

Figure 2 Task2 result

Checkpoint #2 Motors Control

- **Purpose:**

  Motion Control of basic DC motors by encoder with Raspberry Pi and Arduino.

- **Tasks:**

  To control two motors by encoder signals.

  - Move forward. (25%)
  - Move backward. (25%)
  - Turn right. (15%)
  - Turn left. (15%)
  - How straight the robot can move when moving forward. (20%)

```
setting /run_id to 0fc9125a-1011-11e8-9501-b827ebaa4d9b
process[rosout-1]: started with pid [2832]
started core service [/rosout]
process[connect_arduino-2]: started with pid [2835]
process[checkpoint2-3]: started with pid [2836]
user's right is 120
user's left is 120
user's right is -100
user's left is 50
user's right is 100
user's left is 200
user's right is 0
user's left is 0
user's right is 100
user's left is 100
user's right is -50
user's left is 50
user's right is 0
user's left is 0
```

Figure 1 User's command terminal

- **Materials list:**

	Material	Number
1	Chassis	1
2	DC motor	2
3	Wheel	2
4	Support wheel	1
5	L298N motor control module	1
6	Li-po battery	1
7	Low voltage alarm	1
8	Screw driver	2

Figure 2 Materials list:

- **L298N**

- Double H-bridge driver module
- When the input voltage is given around 7V to 12V, can supply 5V for motors
- IN1, IN2, IN3 and IN4 : High/Low pulse for rotation direction
- ENA, ENB: PWM for speed control

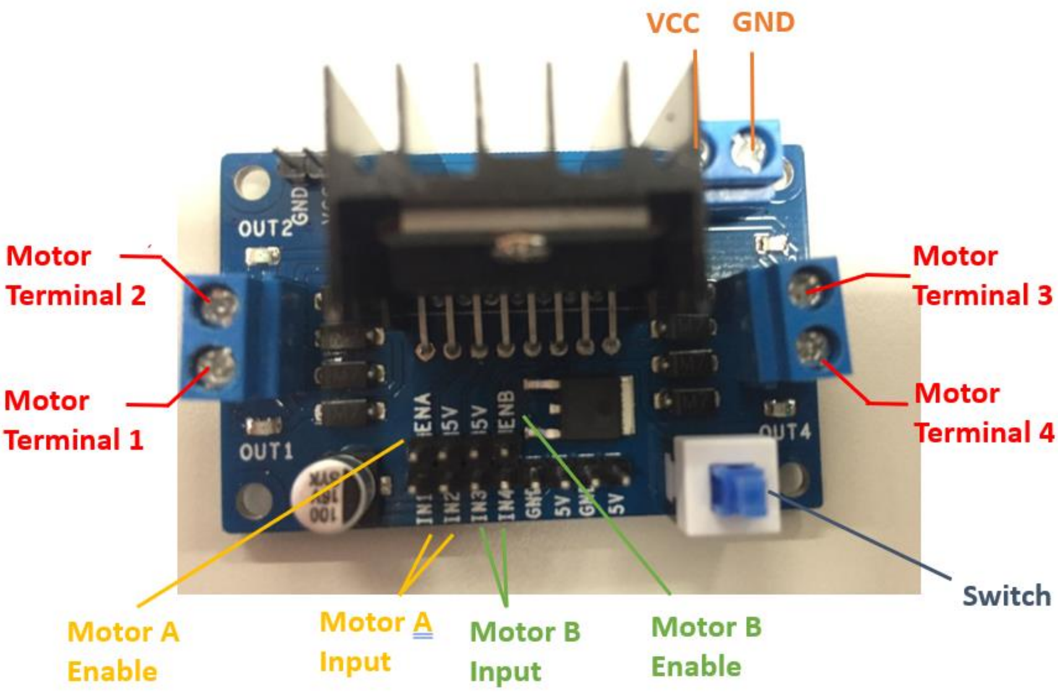


Figure 3 L298N motor driver module

- ## Motor with Encoder
  - The motor with a 120:1 gearbox and an integrated quadrature encoder that provides a resolution of 16 pulse single per round.
  - Pin Description

Pin	Name	Description
A	Encoder A phase output	Changes square wave with the output frequency of Motor speed
B	Encoder B phase output	Changes square wave with the output frequency of Motor speed(interrupt port)
C	Encoder supply GND	
D	Encoder supply +	4.5-7.5V

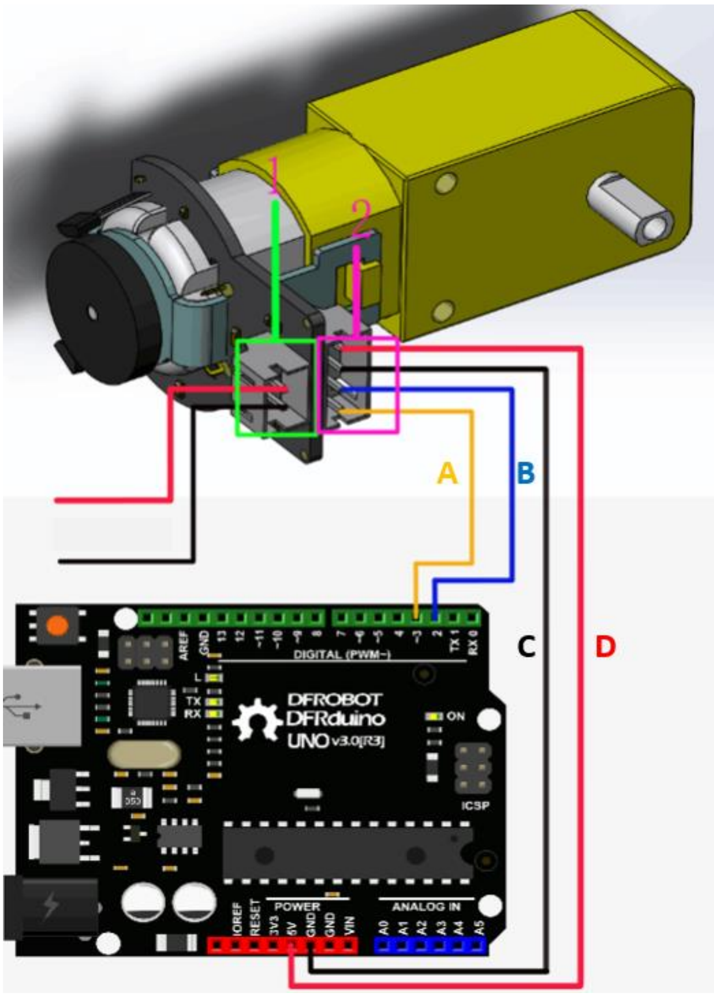


Figure 4 Motor layout

Checkpoint #3 Obstacle Avoidance

- **Purpose:**

The purpose of this checkpoint is to make sure you can control your robot to move in the arena. The mobile robot needs to detect an obstacle in front of it and take action to avoid the obstacle in order to continue its motion.

Finally, your robot can find the assigned target. In this checkpoint, the target is a ring of LED lights.

- **Tasks:**

Please demonstrate your robot performing the following actions:

1. Please start to arrange the space configuration of your robot, make sure every and each component such as circuit boards and sensors is settled firmly and stable on the chassis and all the robot functions will not be affected by wires. **(15%)**
2. Make sure that your robot can move freely. It means that you do not need to use keyboard to control it anymore. **(20%)**
3. Integrate a light sensor and two touch sensors to the robot and program your robot to find and move toward the LED light. **(30%)**
4. The time to find the LED light. (in 90 sec). **(35%)**

Arena:

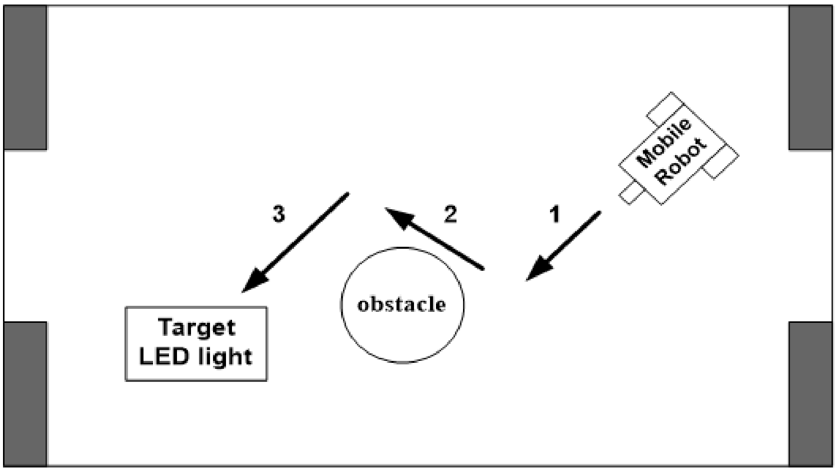


Figure 1 Arena

- **Materials list:**

	Material	Number
1	Photo resistor sensor	1
2	Touch sensors	3
3	Resistances	3
4	Breadboard	1

- **Photo resistor sensor:**

Use Photo resistor sensor to detect the LED light.

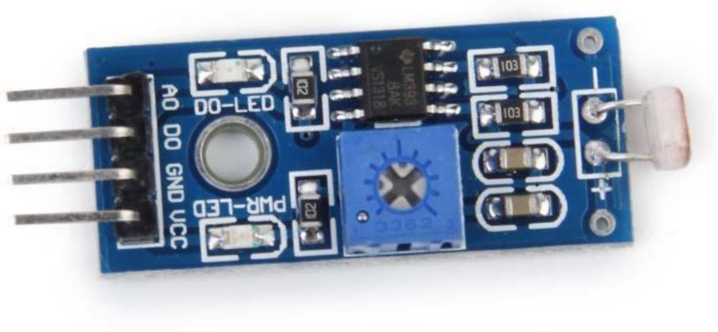


Figure 2 Photo resistor sensor

1. V_{cc} connect to Pi3's 5V.
2. GND connect to Pi3's GND.
3. D_0 connect to GPIO Pin.
4. You can change the Variable Resistor to increase the sensitivity of the sensor. If the brightness is bright enough, D_0 will be 0

- **Touch sensor:**

Integrate touch sensors to avoid an obstacle and walls of the area.

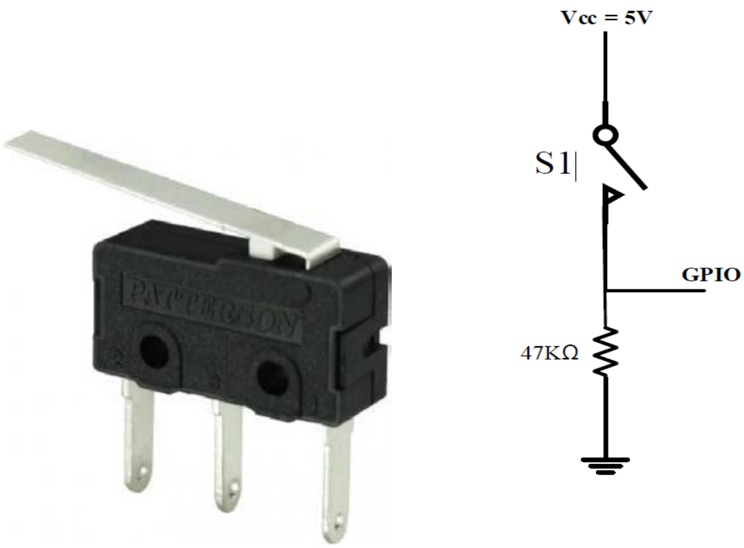


Figure 3 Touch sensor

- **Example Hardware Configuration**

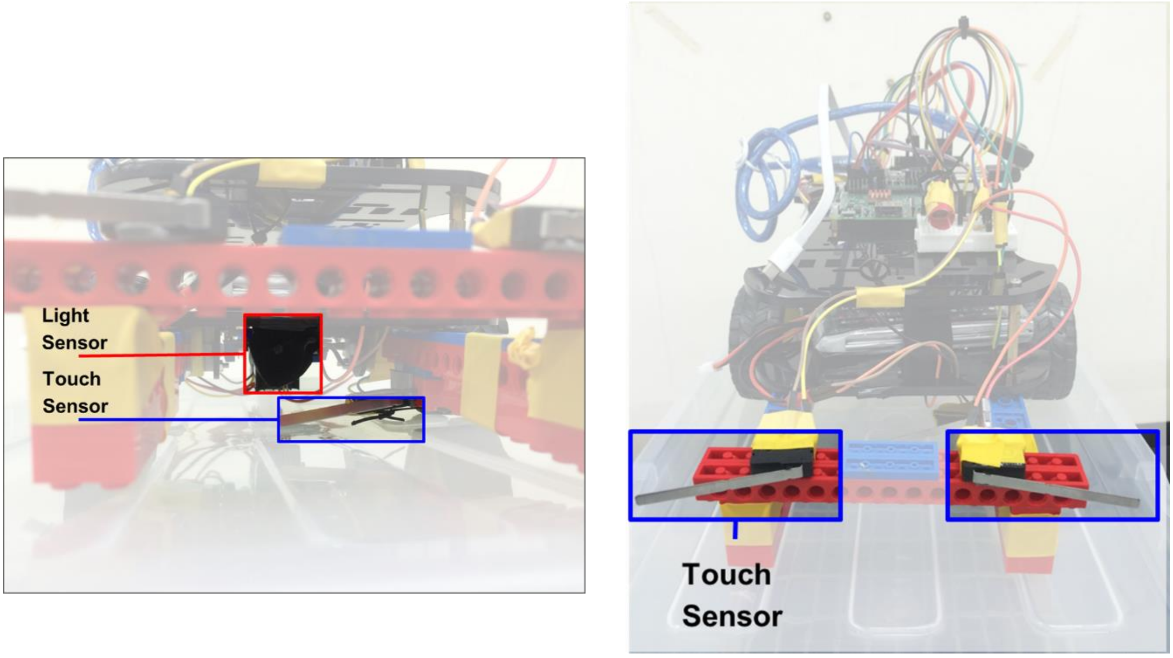


Figure 4 Example Hardware Configuration

- **GPIO pin**

Reference：https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi

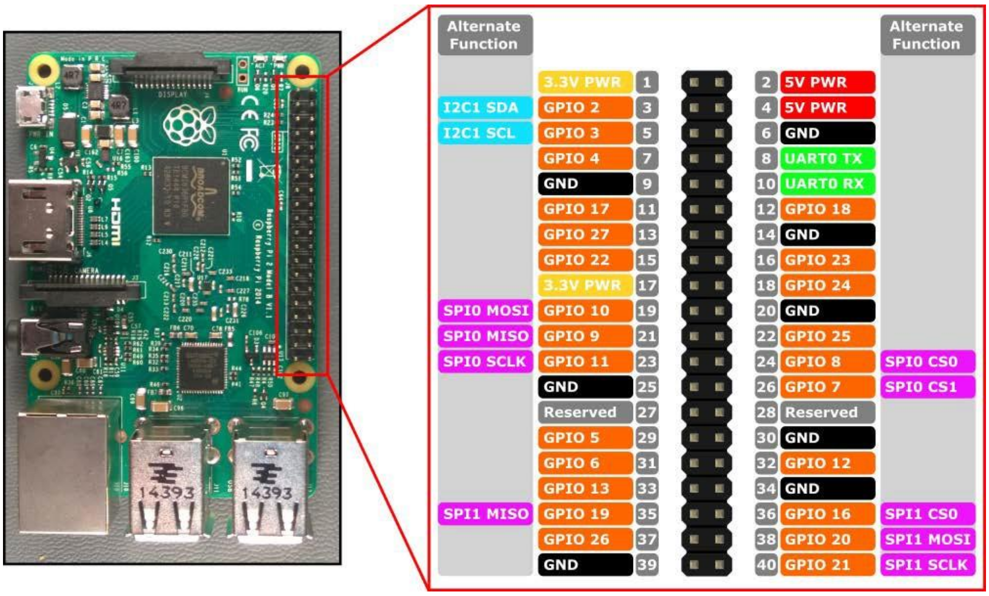


Figure 5 Hardware interfaces for the Raspberry Pi 2 and Raspberry Pi 3

- **Wiring Pi**

WiringPi is an GPIO interface library for the Raspberry Pi and it attempt to bring Arduino-wiring-like simplicity to the Raspberry Pi.The goal is to have a single common platform and set of functions for accessing the Raspberry Pi GPIO across multiple languages

.

1. Test wiringPi's installation

- **Run the gpio command to check the installation.**

```
$ gpio -v
```

Ps. If it's not working, you have to reinstall.

Installation steps Reference：http://wiringpi.com/download-and-_install/

- **To install**

If you get something, then you have it already installed. You will need to purge the package first.

- C version:

```
$sudo apt-get purge wiringpi
```

- Python version:

```
$sudo pip install wiringpi2
```

If you do not have GIT installed, you can install it with：

```
$sudo apt-get update
$sudo apt-get install git
```

To obtain WiringPi using GIT:

```
$ cd
$ git clone git://git.drogon.net/wiringPi
$ cd ~/wiringPi
$ git pull origin
```

The new build script will compile and install it all for you.

```
$ ./build
```

Check the WiringPi have already installed.

```
$ gpio -v
```

2. Check WiringPi's Pin number

```
$ gpio readall
```

Prints a table of WiringPi's Pin number on terminal.

P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Figure 6 WiringPi's pin

3. WiringPi with ROS

If you can run this example code successfully, your program will display 1 when your light sensor to detect the light.

- **Example cpp code**

```
#include "ros/ros.h"
#include <wiringPi.h>
#include <iostream>
#include <std_msgs/Int16.h>

//light receive pin 3
const short int lightpin = 3;

ros::Time previous_time; ros::Time current_time;

int main (int argc, char **argv){
ros::init(argc, argv, "light_receive_data");
ros::NodeHandle n;
ros::Publisher light_pub = n.advertise<std_msgs::Int16>("light_data", 1);

unsigned short int light_rev = 0;
std_msgs::Int16 light_data;

//use this command whithout sudo
setenv("WIRINGPI_GPIOMEM", "1", 1);

//library setup function wiringPiSetup ();
pinMode (lightpin, INPUT);

//10hz
ros::Rate loop_rate(10);
while(ros::ok()){
light_rev = digitalRead(lightpin) ;
light_data.data = light_rev;
ROS_INFO("light_receive : %d ", light_rev);
light_pub.publish(light_data);
ros::spinOnce();
loop_rate.sleep();
}
return 0 ;
}
```

- **Example cmake code**

```
cmake_minimum_required(VERSION 2.8.3)
project(light_receive_data)

find_package(catkin REQUIRED COMPONENTS roscpp
rospy std_msgs
)
FIND_LIBRARY(WIRINGPI_LIBRARY wiringPi /usr/local/include)

catkin_package(
CATKIN_DEPENDS roscpp rospy std_msgs
)

include_directories(
${catkin_INCLUDE_DIRS}
)

add_executable(lightreceivedata src/light_receive_data.cpp)
target_link_libraries(lightreceivedata ${catkin_LIBRARIES}
${WIRINGPI_LIBRARY})
```

- **Example launch code**

```
<launch>

<!--connect arduino-->
<!--node name="connect_arduino" pkg="rosserial_python"
type="serial_node.py">
<param name="~baud" type="int" value="57600" />
<param name="~port" type="string" value="/dev/ttyACM0" />
</node-->

<!--node name="name" pkg="your package" type="Executive file"/-->

<node name="light_receive_data" pkg="light_receive_data" type =
"lightreceivedata"></node>
</launch>

```

Checkpoint #4 Full Function Demonstration

● Purpose:

The purpose of this checkpoint has two goals. First, making sure that your robot can detect a beacon signal and move towards it. Second, combine all the function as Obstacle Avoidance, Hockey Seeking (Light-ball detection) and Goal Seeking (IR signal receiving and moving toward goal) together for robot hockey contest.

For this assignment, two infrared diodes will be set up at opposite ends of an arena. Each diode will be emitting light modulated at 38 KHz, but their pulse width are different when received by IR receiver module.

You will need to demonstrate your robot's capabilities under relaxed conditions with no other robots in the arena. The arena will be the actual contest arena.

● Tasks:

Please demonstrate your robot performing the following actions:

1. Have the ability to avoid all the obstacle in the arena. (20%)
2. Capture the hockey puck. (20%)
3. Your robot should be able to find **two different beacons (Beacon-1 600 and Beacon-2 1500)** and move to the specified beacon in the arena and bring the puck into the goal , respectively, of Beacon1 and Beacon 2. (25%)
4. The time to complete the goal of **Beacon-1 600** and the goal of **Beacon-2 1500 (25%)**.
5. Should complete the mission in 120sec. (The completing time will be counted for grading.)(10%)

Arena:

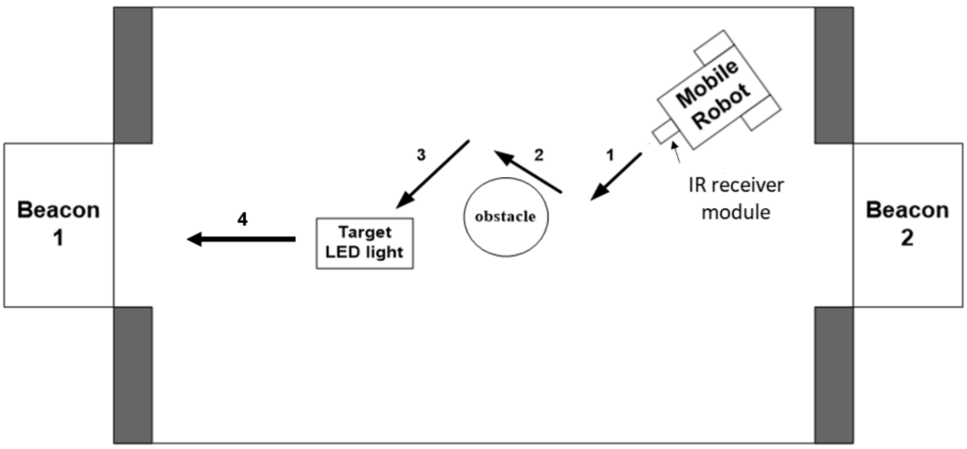


Figure 1 Arena

● Materials list:

	Material	Number
1	PIC-428 LM IR receiver	1

Using IR receiver to find beacons(Beacon-1 600 and Beacon-2 1500) which is the goal.

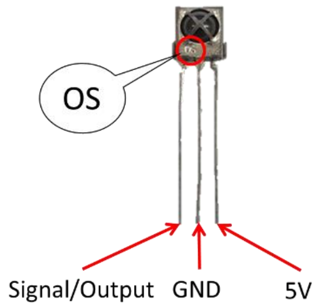


Figure 2 IR receiver

- **Beacon (Provided by TA):**

Two infrared diodes are set up at opposite ends of the arena. Each Beacon will emit light modulated at 38KHz, but their pulse widths are different when received by the IR receiver module.

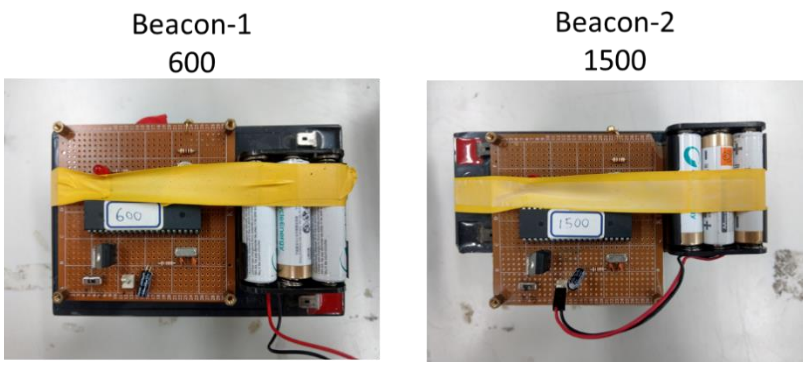


Figure 3 Beacon emitter

- **Search Beacon:**
  - Beacon-1 600
  - Beacon-2 1500

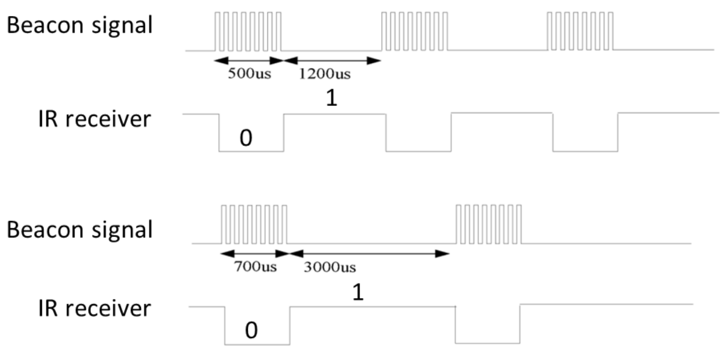


Figure 4 The signals emit by Beacon

Receive IR data for a period of time(≥0.1 seconds) and calculate the pulse proportion.

$$\textbf{Ratio} = \frac{\boldsymbol{number\ of\ 0}}{\boldsymbol{total\ data(include\ 0\ and\ 1)}}$$

If your goal door is 1500, the ratio is between 0.17 and 0.22.

If your goal door is 600, the ratio is between 0.27 and 0.32.

Robot Hockey contest

1. The contest is one-on-one fashion. A march consists of an upper half and a lower half sets. Each set has 90sec.
2. A robot which brings the puck into the opponent goal wins 2 scores. If the puck is in the own goal, there is no score for either side.
3. As time is up in any half march, a robot which holds the puck wins 1 score.
4. If a robot does not move for 15sec in the game, the game ends immediately. The opponent side wins 1 score.
5. The complete contest contains two runs. Four teams with the highest scores in run 1will be selected to the 2ndrun. The 2nd run consists of two semi-finals and the final.
6. **There are three start positions for both robots(see below). The start position is determined by throwing a dice in the march. The original puck position will be at the center.**
7. During the march, if a robot is broken, upon the student request, the march may be suspended for 60secfor repairing.
8. **Each and every robot is not allowed to lock the puck during the contest, for example to use a magnet or a fence.**
9. The judge gives the final decision for any other conditions occurring during the contest.

Arena:

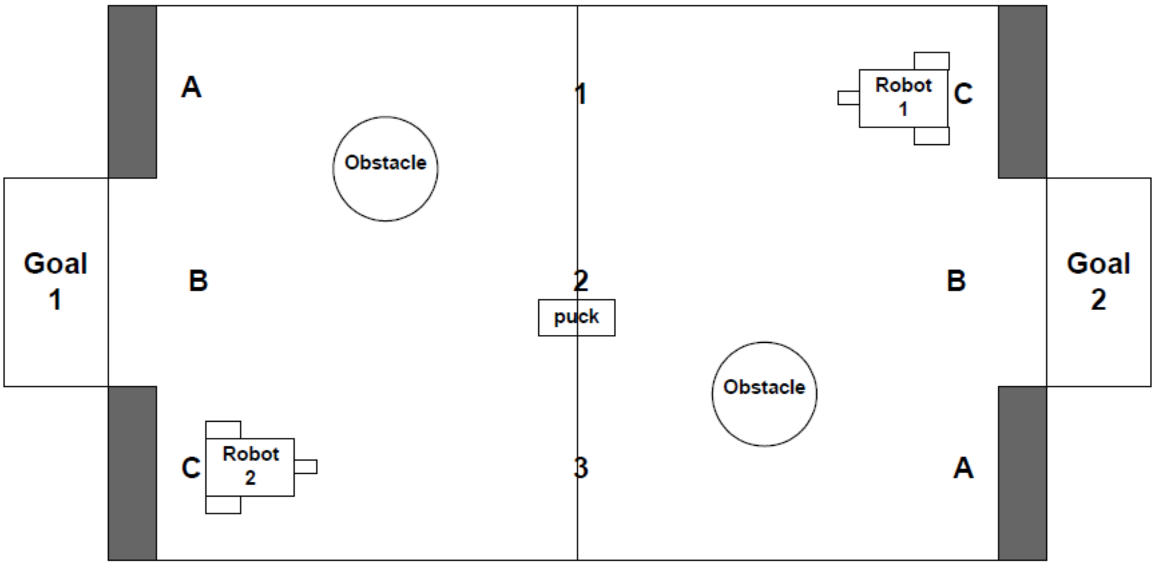


Figure 1 Arena