物联网和机器人导论
VIPLE工作流编程

Introduction to IoT and Robotics,
based on Visual Programming
Experiments

Yinong Chen
Arizona State University, U.S.A.

Lectures of the Course

http://neptune.fulton.ad.asu.edu/VIPLE/

- **ASU VIPLE** can be used as the lab environment in Introduction to Engineering's fr
. They can be used together with the VIPLE tutorial, which is a lab manual for writi
  - L01 - About the Course and Syllabus
  - L02 - CS Related Disciplines
  - L03 - VIPLE - Visual IoT/Robotics Programming Language Environment
  - L04 - ALU Simulation in VIPLE
  - L05 - Number systems
  - L06 - Finite State Machine and Programmig
  - L07 - Algorithms
  - L08 - Event Driven Programming
  - L09 - Programming Langauges
  - L10 - Operating System
  - L11 - Unix and Edison
  - L12 - IoT and RaaS
  - L13 - IoT and Augmented Reality
  - L14 - from OOC to SOC
  - L15 - SOC and Web Software
  - L16 - Presentation Techniques
  - L17 - Big Data
  - L18 - Cloud Computing
  - L19 - Amdahls Law
  - L20 - Ethics Theories

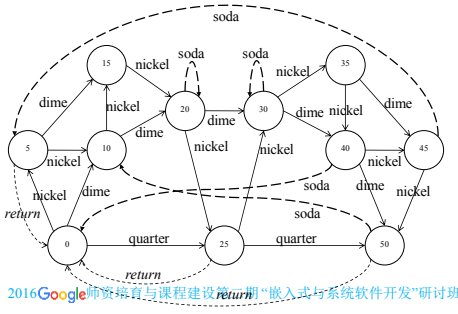
Lecture Outline

Finite State Machine (FSM)

- A Finite State Machine (FSM) is a mathematical model consisting of a finite number of states, transitions between states, inputs, and outputs.
  - An input causes a transition from one to another state
  - An output is associated with a state and an input
- As a programming model, FSMs are best to be applied to respond to a **sequence** of inputs (events), such as
  - Coin insertion into a vending machine;
  - Timer expires for traffic lights
  - Mouse-clicks/key strikes during a program's execution
  - Arrival of individual characters from a string

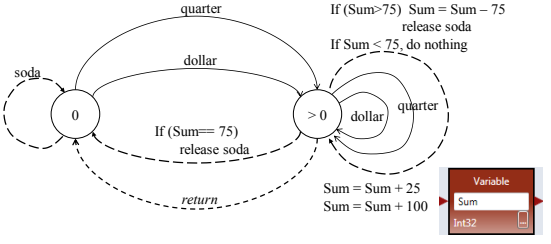
A Classic Application of FSM: Vending Machine

- Soda costs 40 cents
- Possible inputs: Nickel, Dime, Quarter
- Not all “return” transitions are shown

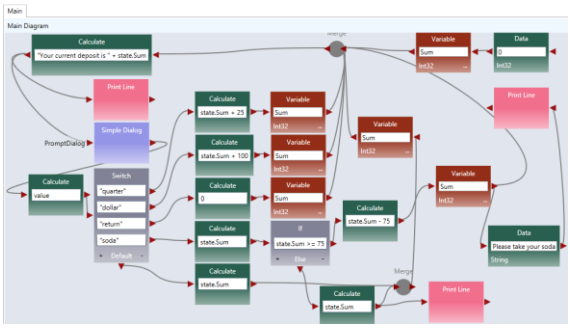


FSM with Variable

- Allowing additional variables for storing values



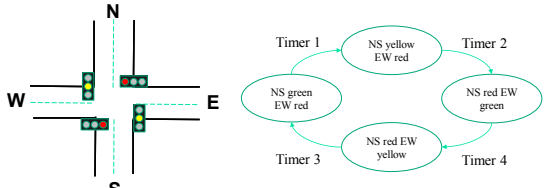
7 VIPLE Implementation of the Vending Machine



8 FSMs are used for designing hardware and software

For example:

- Sequential circuit design;
- Parity generation in storage and communication;
- Parentheses matching in expressions;
- Garage door controller;
- Traffic lights controller.



9 FSMs are often used for Traffic Lights System

- When the system is simple, e.g., with two to four states, we may not need to explicitly use an FSM
- When the system becomes more complex, it is harder without using FSM.
- For example, design the controller for this traffic lights:

Design controller for this traffic controller

Traffic Lights by Canary Wharf Tower, East London

10 Use FSM to design a garage door control system

No states required

Required states.
When button pressed:
If state==open
then close
else open

Open Close

11 Use FSM to design a garage door control systems

If the door is closed and the button is pressed (touch sensor), the door begins to move up.

When it reaches the top, the door activates a limit switch (a touch sensor) and stops.

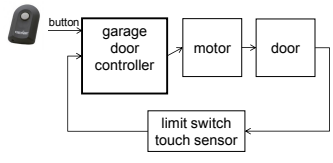
If the door is open and the button is pressed, the door begins to move down.

When it reaches the bottom, the door activates another limit switch and stops.

12 Use FSM to design a garage door control systems

A garage door opener system

block diagram



...we want to design the controller...

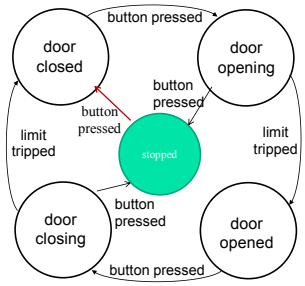
13 Use FSM to design a garage door control systems

A garage door opening system

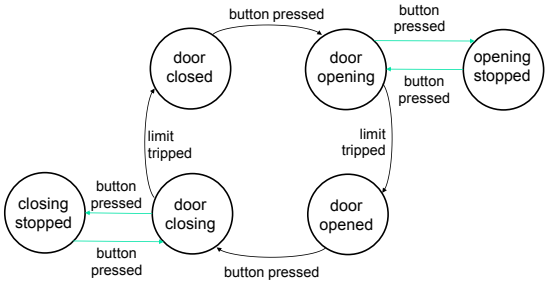
states
- door closed
- door open
- door closing
- door opening

events
- button press
- limit switch touched
  (closing finished or opening finished)

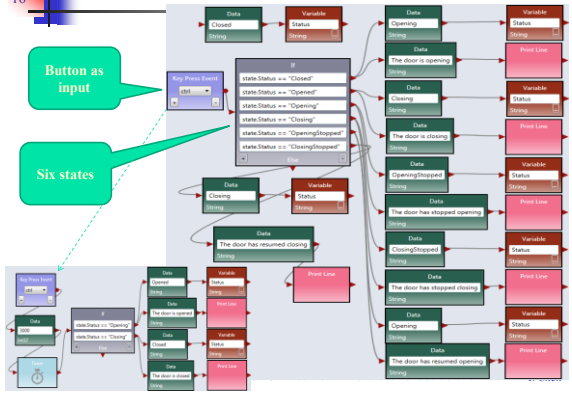
14 Use FSM to design a garage door control systems



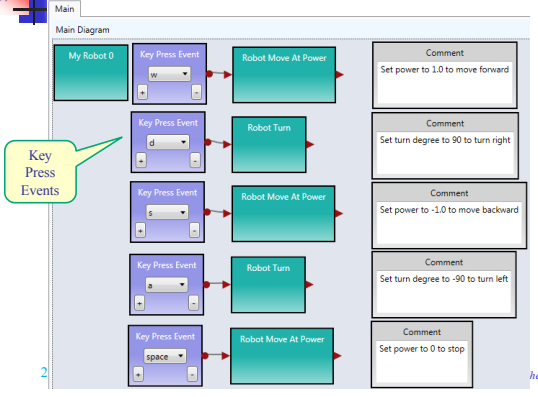
15 Use FSM to design a garage door control systems



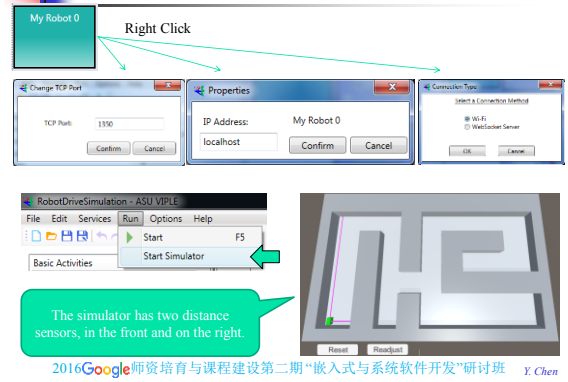
16 VIPLE Implementation if the Garage Door Opener



17 Drive-by-Wire Simulation

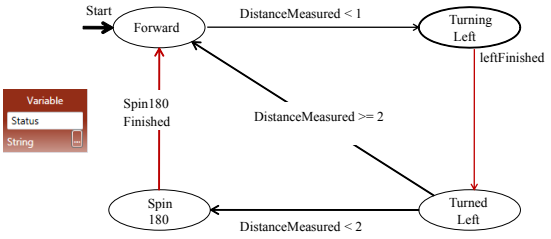


Unity Simulator Configuration



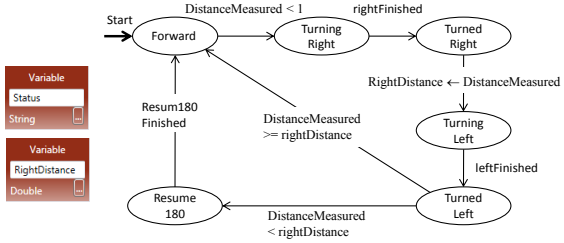
19 Maze Navigation Algorithms

- Greedy Algorithm: Take the first working solution



20 Maze Navigation Algorithms

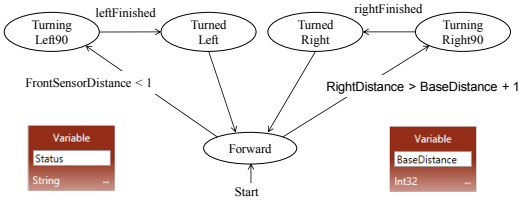
- Local Best: Measure two distances and choose the direction with the longer distance



21 Maze Navigation Algorithms

Right-Wall-Following:

- Move forward;
- Turn right if the right side is open;
- Turn left if the front is not open.
- It assumes that forwards and turns are accurate.



22 Improving the Navigation Algorithm

- For example:
- Detect the cu-de-sac

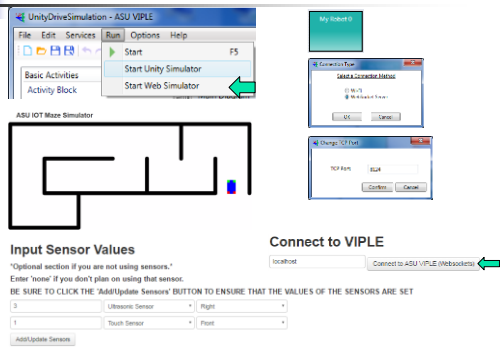
Given implementation

turn 90 degree;
move forward;
turn 90 degree

A possible enhancement

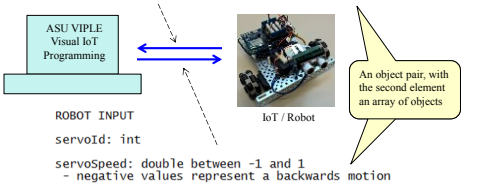
If cu-de-sac
turn 180 degree

23 Web Simulation



24 Communication between VIPLE & Device: JSON Object

```
ROBOT OUTPUT

name: string (touch, distance, sound, light, color, motorEncoder)
id: int
value: For touch sensor, value will be an int (0 = not pressed and
                                                1 = pressed).
       For other sensors, value will be a double

{"sensors": [{"name":"touch", "id":0, "value":0},
             {"name":"distance", "id":1, "value":12.8}]}
```

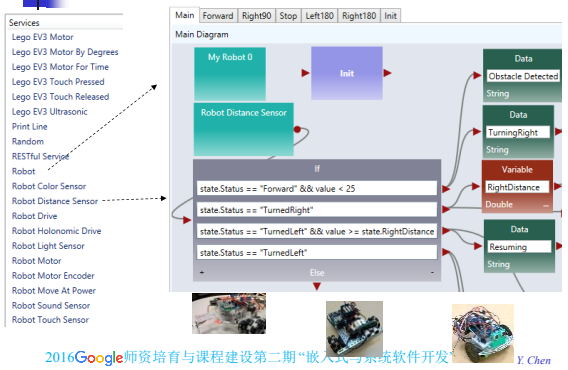



```
ROBOT INPUT

servoId: int

servoSpeed: double between -1 and 1
 - negative values represent a backwards motion

{"servos": [{"servoId":3, "servoSpeed":0.5},
            {"servoId":5, "servoSpeed":-0.5}]}
```

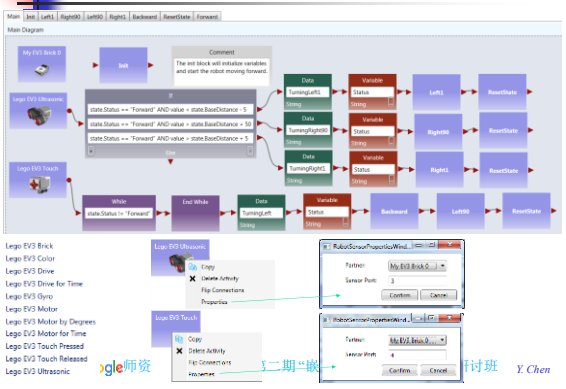
25 Interfacing with Different Devices

- For any open architecture robot, we can program the robot:
  - to interpret the JSON objects from VIPLE
  - to wrap sensor data into JSON object
- For vendor-specific robots, such as Lego EV3 and iRobot,
  - We cannot program the robots to interface with VIPLE
  - We program VIPLE to interface with each type of such robots. I have Lego robots programmed so far.

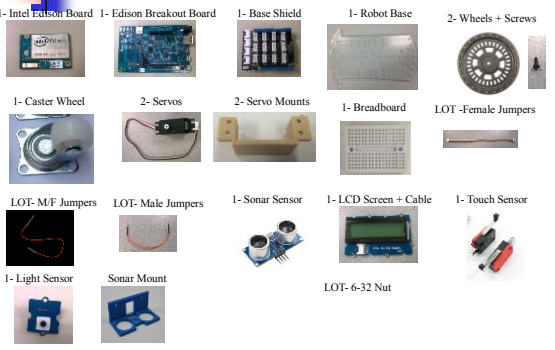
26 Connecting to Physical Robot: Edison Robot



27 Controlling Physical Robot: Lego EV3 Robot



28 Physical Robot Construction



29 Physical Robot Servo Assembly

Note: Make sure servos are calibrated prior to installation

1. Gather servo and mounts

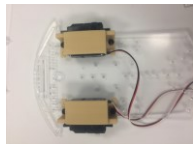

2. Slide servo into mount


3. Screw in the top hole for the servo

30 Physical Robot Body Assembly

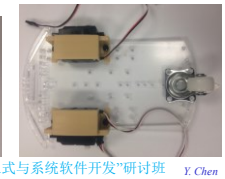
1. Gather base and Servo mounts


2. Align mounts on the base and fasten with screw and nuts



3. Gather caster wheel and hardware

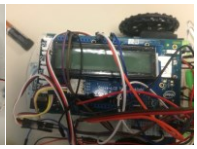
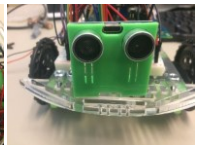
4. Align and fasten Caster Wheel to base as shown

31 Physical Robot Final Assembly





32 Lecture Summary

- Use FSM model for event-driven programming
  - Vending machine
  - Traffic light controller
  - Garage door opener
- Autonomous maze navigation algorithms and programs
  - VIPLE simulation environment
  - First working solution
  - Local best
  - Wall following
- Physical Robot Construction