CSE446 Project 3 (Assignments 5 and 6, 50+50 Points)
Spring 2020

A5 Due: March 21, 2020 by 11:59pm (Arizona Time), plus one day grace period

A6 Due: March 28, 2020 by 11:59pm (Arizona Time), plus one day grace period

---

Introduction

The aim of this project is to make sure that you have read the slides and the text and have understood the concepts covered in the lectures and in the text, including messaging, caching, and VIPLE. By the end of the assignment, you should have a good understanding of the concepts and have applied these concepts in developing operational programs.

Practice Exercises and Preparation (No submission required)

1. Reading: Textbook Chapters 4 and 5.4.2 on XML file operations, Chapter 9 and Appendix B on IoT and VIPLE.
2. Practice exercises: Questions at the end of Chapters 8. Complete the multiple choice questions. The solution is available in the course web page.
3. Practice exercises: Questions at the end of Chapters 9. Complete the multiple choice questions. The solution is available in the course web page.
4. Follow Lecture 2-8 to implement a Website with caching.
5. Read Text Chapter 9 and download VIPLE (http://neptune.fulton.ad.asu.edu/VIPLE/) to develop workflow programs for robotics applications.
6. Follow Text Chapter 9 and Appendix B. Implement different applications using finite state machines.
7. Follow the tutorial to run the drive-by-wire program using the Unity simulator. Note, in order to start you program, you need to:
    1) After you started the Unity Simulator, you need to go back to VIPLE, to click the "Start" button", the green arrow. Then, the Control window will appear, showing that the robot is running.
    2) When you are controlling the robot using the keyboard, you must keep the Control window on the top in active, not in background.

Assignment 5: Messaging (Submission Required, 50 points)

Due: March 21, 2020 by 11:59pm (Arizona Time)

Create a simplified messaging service to buffer messages in an XML file and a Web client to allow users to send and receive messages. You will develop your service and application on IIS Express localhost.

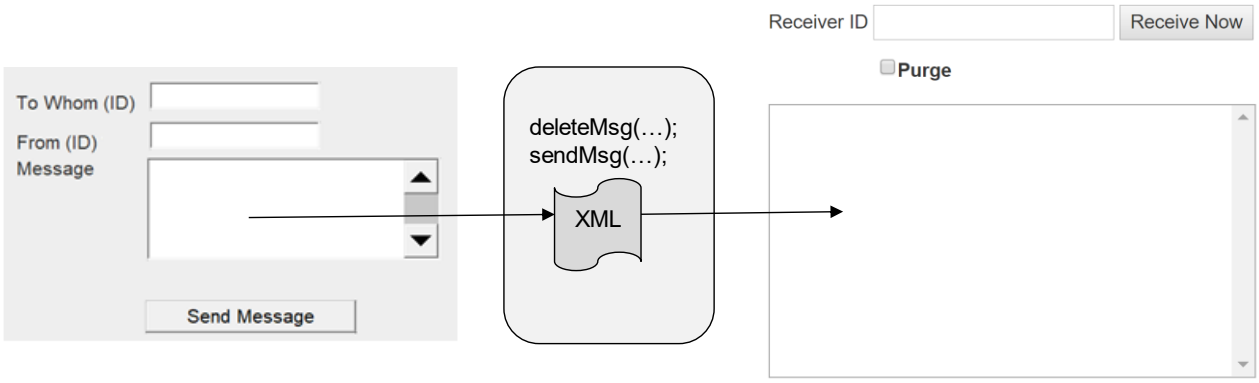
1 Messaging service: Develop a service (WSDL service, RESTful service, or Workflow service) that can buffer messages before the receivers fetch the messages. The messages must be saved in an XML file (or JSON file) that the service can access. The service must offer the following operations. You may add more parameters for additional functions. [25]

- Operation 1: sendMsg: It allows the client to send a string message to the messaging service, and the message will be stored in the database (XML file) with senderID, receiverID, and a time stamp.
  Inputs: string senderID, string receiverID, string msg.
  Output: void
  Note: a timestamp will be added by the program.

- Operation 2: receiveMsg: It allows the client to receive all the messages send to the receiverID.
  Inputs: string receiverID, boolean purge
  Output: string[] an array of messages, with each element containing the related information: senderID, sending time, and message.
  Note: The receiver should receive the new messages that have not been received in the previous receive call. If purge == true, the service will delete all messages of the receiver.

To read and write an XML file on disk, you may want to read text Chapter 4 on XML processing and Section 5.4.2 on reading and writing XML files. You may also read Text Chapter 10 and using LINQ to XML methods.

2 MsgApp: Develop a Web application (client) in ASP .Net or MVC that contains at least two Web pages: Sender page and Receiver page. The sender sends messages to the Messaging service and the receiver receives messages from the service. A sample GUI is shown below. The GUI must link to all components that need to be implemented. You can add additional items in your GUI. [25]



To test your clients (sender and receiver) on localhost/IIS Express: Right click the service project and choose View in Browser to run the service. Then, start the client, which will open another bowser window to run the client. Copy and paste the URL into a new browser window. Now, you have three sessions of your application: one service session and two client sessions. You can test your application by sending a message in one browser and receiving the message in another browser. Submit a screenshot of your testing.

Submission: You can put the service and the client in one solution or in two separate solutions. You must place all the files into one folder and zip the folder for submission. Make sure that your solution and project can run on TA's computer.

Assignment 6: IoT Application Composition and Integration (Submission Required)

Due: March 28, 2020 by 11:59pm (Arizona Time)

Before you start this part of the assignment, you need to follow the lecture slides and the tutorial in Text Appendix B and Chapter 9 to get started with programming in VIPLE workflow.

1 An ASCII code consists of 7 bits of 0s or 1s. The 8th bit is often generated for parity checking. Write a VIPLE application to generate the even-parity bit for an ASCII code. **[20 points]**

1.1 Use the keypress event to enter a string of seven 0s and 1s. The VIPLE diagram counts the number of 1s entered. After seven digits are entered, it outputs the string of eight digits, with the eighth digit to be 1 if the number of 1s entered is an odd number. Otherwise, the eighth digit will be 0. Use Print Line to print the entire eight digits. [5]

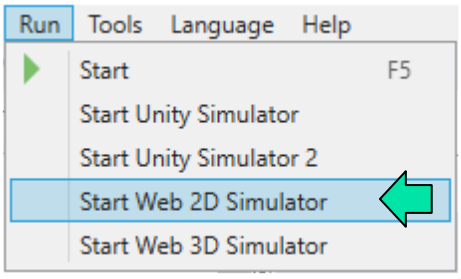
1.2 Use a WSDL service or a RESTful service (for example, in ASU Repository) to encrypt the eight-digit value generated in the previous question, save the encrypted value into a variable, and use Print Line to print the encrypted string. [5]

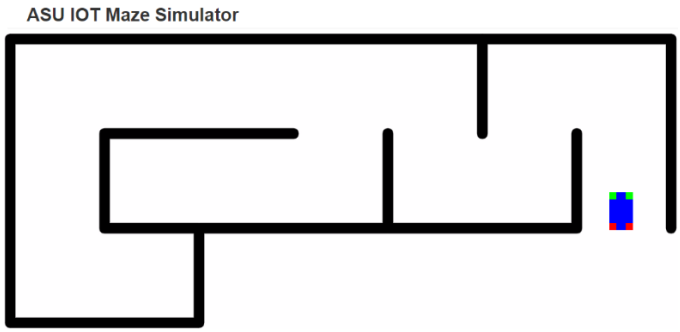
1.3 Write a code activity to read the encrypted value and decrypt the value by calling the corresponding decryption service. You can use either a WSDL service or a RESTful service in this question. [5]

1.4 Generate a random binary number of 7 bits, instead of entering via keyboard. Use the activities in the previous questions (1.2 and 1.3) to process the number. Put these steps in a loop and iterate the steps for N time, where you can set N = 10. [5]

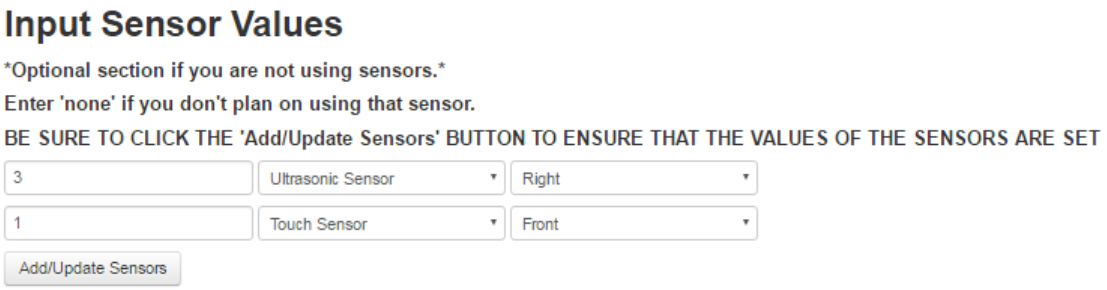
Submission: The VIPLE code for questions 1.1, 1.2, 1.3, and 1.4.

2 Robot maze navigation using VIPLE and an HTML5 simulator: Web 2D simulator. Configure your devices following these diagrams. **[30 points]**

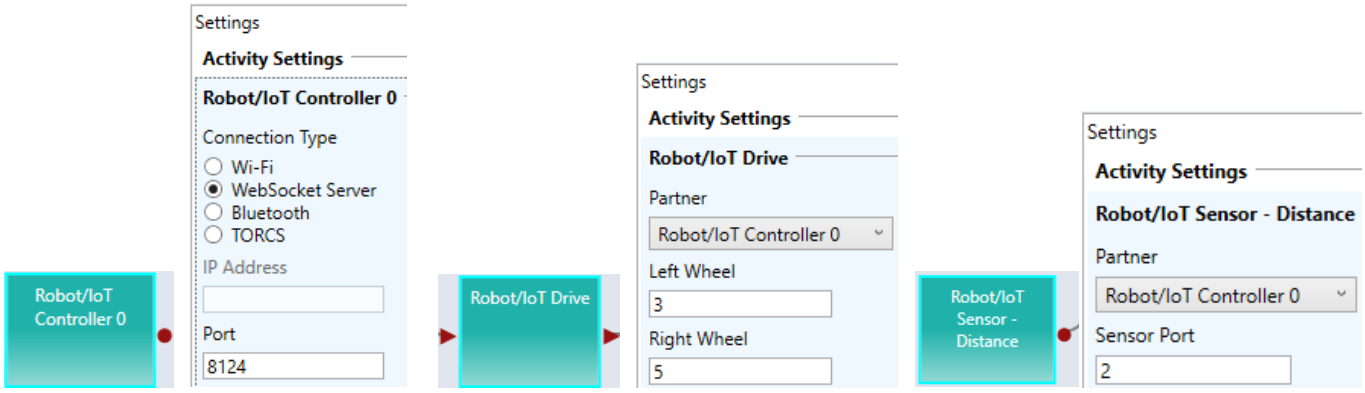




Read the instructions on the web page to configure the simulator. Note, you need to change the right wall to left wall!



On the VIPLE side, you need to configure your devices to match with what you have configured on the Web simulator side.



Your tasks are to implement the **left**-wall-following program. You can follow the textbook Section 9.5.4. Note, the book shows the sample code for the **right**-wall-following program.

Submission: The code for question 2.

Put all the codes in all questions in one folder, zip the folder, and submit the zip file.

General Submission Requirement

All submissions must be zipped into a single file electronically submitted to the Canvas assignment folder. All files must be zipped into a single file for submission.

For programming assignments, the entire solution folder with all the files must be included. To make sure that you have all the files in the zip file, unzip the file on a different machine or in a different directory, and test if you can run the project in a different location.

Submission notice: A programming assignment typically consists of multiple distributed parts. They may be stored in different locations when you create them. You must copy these projects into a single folder for Canvas submission. To make sure that you have all the files included in the zip file and they work together, download your own submission from the Canvas. Unzip the file on a different machine, and test it and see if you can run the solution in a different location because the TA will test your application on a different machine.

Grading and Rubrics

Each sub-question (programming tasks) has been assigned certain points. We will grade your programs following these steps:

(1) Compile the code. If it does not compile, 50% of the points given for the code under compilation will be deducted. Then, we will read the code and give points between 50% and 0, as shown in right part of the rubric table.

(2) If the code passes the compilation, we will execute and test the code using test cases. We will assign points based on the left part of the rubric table.

In both cases (passing compilation and failed compilation), we will read your program and give points based on the points allocated to each sub-question, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

Please notice that we will not debug your program to figure out how big or how small the error is. You may lose 50% of your points for a small error such missing a comma or a space!

We will apply the following rubrics to **each sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

Major	Code passed compilation				Code failed compilation		
Points	pts * 100%	pts * 90%	pts * 80%	pts * 70% - 60%	pts * 50% - 40%	pts * 30% - 10%	0
Each sub-question	Meeting all requirements, well commented, and working correctly in all test cases	Working correctly in all test cases. Comments not provided to explain what each part of code does.	Working with minor problem, such as not writing comments, code not working in certain uncommon boundary conditions.	Working in most test cases, but with major problem, such as the code fail a common test case	Failed compilation or not working correctly, but showing serious effort in addressing the problem.	Failed compilation, showing some effort, but the code does not implement the required work.	No attempt

Late submission deduction policy:

- Grace period (Sunday): No penalty for late submissions that are received within 24 hours of the given due date;
- 1% grade deduction for every hour after the first 24 hours of the grace period!
- No submission will be allowed after Tuesday midnight. The submission link will be disabled.