ECEN 4213:
Embedded Computer System Design

Final Project: Web-based Remote Robot Control

Ex #	Max Points	Points Earned	Grading criteria	Instructor Initial
1a	1		Program entered correctly, compiles, and runs as specified (1.0) _____	
1b	2		Program entered correctly, compiles, and runs as specified (2.0) _____	
2a	4		Program entered correctly, compiles, and runs as specified (4.0) _____	
2b	4		Program entered correctly, compiles, and runs as specified (4.0) _____	
3	3		Program entered correctly, compiles, and runs as specified (3.0) _____	
4a	5		Program entered correctly, compiles, and runs as specified (5.0) _____	
4b	5		Program entered correctly, compiles, and runs as specified (5.0) _____	
Bonus 1	5		Program entered correctly, compiles, and runs as specified (5.0) _____	
Bonus 2	4		Program entered correctly, compiles, and runs as specified (4.0) _____	
Bonus 3	3		Program entered correctly, compiles, and runs as specified (3.0) _____	
Report	2			
TOTAL:				

DESCRIPTION

This lab will introduce the concept of Internet of Things (IoT) with a focus on web-based remote robot control, which involves the design of a webpage and robot codes that allow a mobile device (such as a smart phone) to remotely control a Kobuki robot and display sensor data including camera video from it.

What you will learn are:

- HTML programming
- Inline CSS
- JavaScript for front end interaction
- Flask web application framework
- jQuery and Ajax for messaging
- UDP for live streaming

The overall structure of the project is as follows:

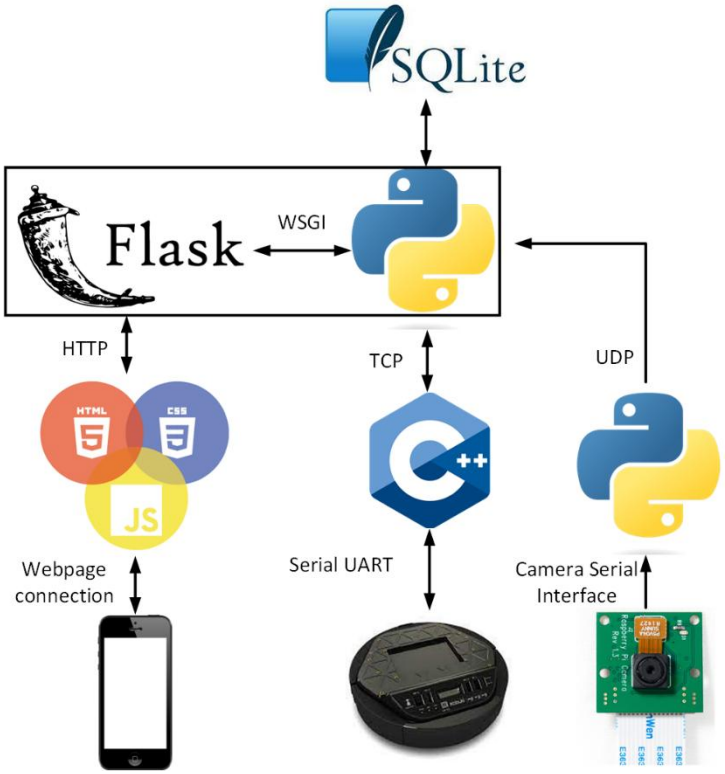


Figure 1. The overall structure of the project

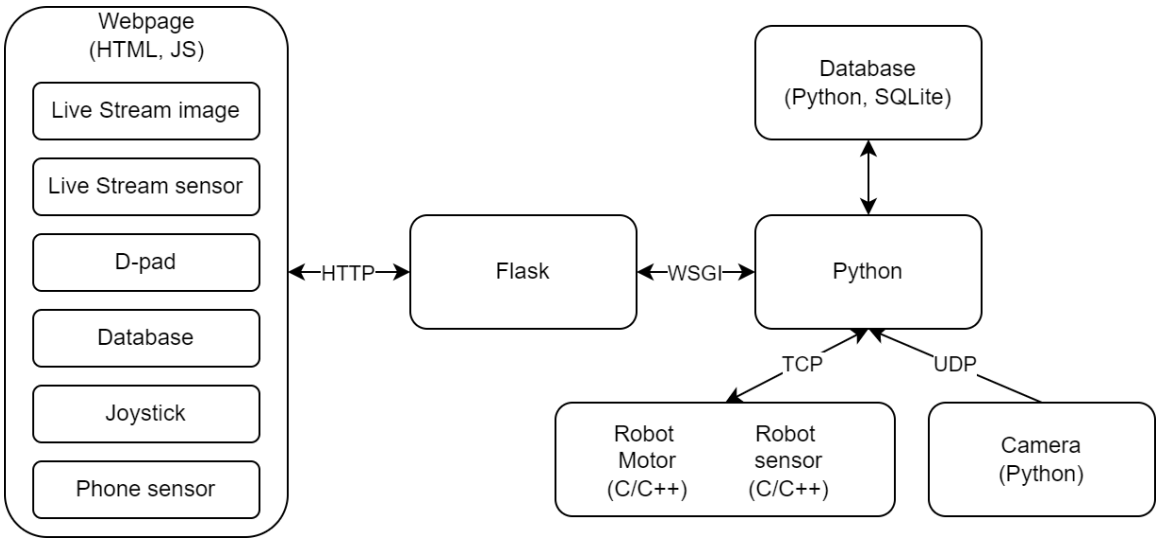


Figure 2. The overall block diagram of the web-based remote robot control system.

Internet of Things (IOT)

"IoT" is a term used to describe a network of physical devices, or 'things', that use embedded sensors, software, and other technologies to share data with each other either locally or across the internet. Examples of these can be as simple as a thermostat that reads the data back to your phone, or a very complex system such as a smart home, which can have many different sensors such as temperature and humidity sensors, cameras, and actuators such as door locks, air conditioners and other smart appliances. A remote-controlled robot is another example of IoT in which a mobile device like a smart phone can access the sensor data from the robot and issue control commands to the robot.

Web Programming for IoT

Web programming can be used for establishing communication channels in an IoT project. Webpages can be accessed on any device as long as it is connected to the Internet. Different from traditional statis websites, IoT-oriented websites should support user interactions on the client side and dynamic content generation at the server side.

The webpage that we will create uses both HTML and JavaScript (JS). Although it does have some dynamic functionality, HTML is mostly used to create static content on a webpage, whereas JavaScript is used to give more interactive functions to the webpage such as pressing a button on a webpage which brings up new text. HTML shows you what the button looks like, while JavaScript makes the button do something. While on the server side, a Flask web framework allows the web pages to interface to user-defined

Python programs which can access the sensor data from the robot and send the user commands to the robot. The overall block diagram of the project is shown in Figure 2.

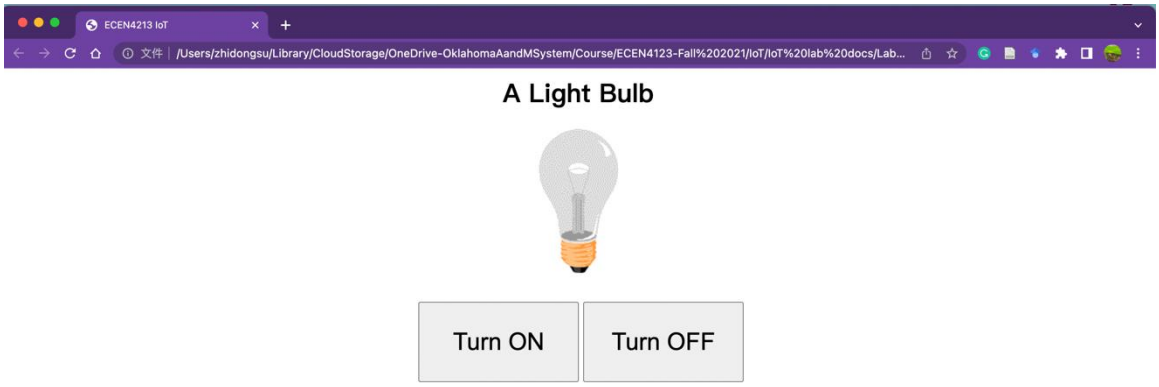
LIST OF DOCUMENTS/CODES USED IN THIS LAB

1. *TutorialsPoint HTML https://www.tutorialspoint.com/html/index.htm*
2. *HTML_CSS_JavaScript_Python_Tutorial.pdf*
3. *Flask_jQuery_Ajax_Tutorial.pdf*
4. *py_client.py*
5. *py_server.py*

Exercise 1a: HTML Basics

In this exercise, you will create a basic webpage. To begin with, we will create a simple webpage that shows a webpage title (ECEN4213 IoT), a description sentence (A Light Bulb), an image and two buttons in the center of the webpage as shown below. Finish the code *FinalEXE1a.html* in the **templates folder** to implement the described function. Use an editor to open the code so you can edit it. Use a browser to open the code to see the render effect.

Learn from the provided document TutorialsPoint HTML.pdf about the HTML basics. You are suggested to look at Chapter 1, 2, 3, 4 and 9. You can focus on the highlighted contents. You can also refer to HTML_CSS_JavaScript_Python_Tutorial.pdf.



Exercise 1b: JavaScript Basics

The JavaScript is a small piece of program that can add interactivity to your website. In this exercise, you will write the JavaScript to enable the two buttons "Turn ON" and "Turn OFF" to turn on or turn off the light. Namely, when you push the "Turn ON" button, the image "pic_bulbon.png" will be displayed. When you push the "Turn OFF" button, the image "pic_bulboff.png" will be displayed. Finish the code *FinalEXE1b.html* to implement the described function.
You can refer to highlighted content in Chapter 26 of the document TutorialsPoint HTML.pdf about the JavaScript basics. For a detailed instruction, please refer to https://www.tutorialspoint.com/javascript/index.htm.

After finishing the HTML code, we will use a web framework named Flask to build a web application so you can use your cell phone to remotely access it. Run the Python code named "*FinalEXE1b.py*" using command "python3 *FinalEXE1b.py*". If your IP is "10.227.1.1" and the Port is "8080", you can use the link http://10.227.1.1:8080 to access the webpage. Make sure your RPI and cell phone are connected to the same WIFI. The following picture shows a running example.

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. Please refer to https://www.tutorialspoint.com/flask/index.htm for more information.

Exercise 2a: Design a Kobuki Interface Webpage

Design a webpage that has three modules: 1: Live stream image, 2: sensor status, 3: control pad, as shown below. We will use the <input> and tags to implement module 2 and 3. You can change the "style" attribute of the <input> tag to change the font size and button size. Include each module into the " " tag to isolate each module. The following shows the code architecture of module 2 and 3.

```
<div>
        <!— module 2 sensor tag: Bumper, wheelDrop, Cilff -->
        <!—sensor tag: Bumper, wheelDrop, Cilff -->
 <input style="font-size:40;height:150; width:300;" type = "button" id = "but1" value = "OFF" />
</div>
<div>
        <!— module 2 sensor status: on, off,… -->
 <input style="font-size:XX;height:XX; width:XX;" type = "button" id = "XX" value = "XX" />
</div>

<div>
        <!— module 3 D-pad -->
        <div>
        <!—up button -->
 <input style="font-size:XX;height:XX; width:XX;" type = "button" id = "XX" value = "XX" />
        </div>

        <div>
        <!—left, stop and right button -->
 <input style="font-size:XX;height:XX; width:XX;" type = "button" id = "XX" value = "XX" />
        </div>

        <div>
        <!—down button -->
 <input style="font-size:XX;height:XX; width:XX;" type = "button" id = "XX" value = "XX" />
        </div>
</div>
```

Finish the code *FinalEXE2a.html* to implement the three modules. After finishing the HTML code, run the Python code named "*FinalEXE2a.py*" and use your cell phone to open the webpage.

Exercise 2b: Integrate JavaScript with the Kobuki Interface Webpage

After finishing the Kobuki Interface Webpage, you need to use the JavaScript to handle the D-pad pushing event and send the control command to the Python Flask code. You also need to send the sensor data from the Python Flask code to the HTML webpage so the webpage can update the sensor status in real-time.

Step 1: Send control command to the Flask server and print out the event. To handle a button pushing event, in *FinalEXE2b.html*, the following JavaScript code monitors the event triggered by the button with the id of "upbutton" and calls the Python function named "UpFuntion" defined in the Python code "*FinalEXE2b.py*".

```
<!-- callback function for the upbutton pushing event -->
<script type=text/javascript> $(function() { $("#upbutton").click(function (event) { $.getJSON('/UpFunction', { },
function(data) { }); return false; }); }); </script>
```

(JavaScript code)

```python
@app.route('/UpFunction')
def UpFunction():
    print('In UpFunction')
    return "Nothing"

# define four funtions to handle the left, right, down and stop buttons
@app.route('/function_name')
def function_name():
    print('In XXFunction')
    return "Nothing"
```

(Python code)

Follow this example to handle the button pushing event of the Down, Left, Right and Stop button, respectively.

Step 2: Send the sensor data from the Flask server to the HTML webpage. Finish the JavaScript code located in the static folder named "parse_data.js" to handle the sensor data. The function of the JS code is to update the value of the display button according to the obtained sensor data. The following code shows an example to update the values. Make sure you choose the correct button ids and handle all button status.

```javascript
// finish the code to handle the bumper status
  if (bumper=="0")
    {
      document.getElementById("but1").value = "OFF";
    }
  if (bumper=="1")
  {
    document.getElementById("but1").value = "Right";
  }


  // finish the code to handle the wheel drop status
if (drop=="0")
  {
    document.getElementById("the id of button where you need to display the sensor status").value = "OFF";

  }
```

Exercise 3: Live Stream Display

In this exercise, you will implement the live stream display function through which you can observe the real-time video obtained from the Raspberry Pi camera in your browser. Before implementing this function, please read the example codes "py_server.py" and "py_client.py" which show a Python version of UDP client and server communication. Run the two codes in different terminals to see what you get.

In your Flask server "FinalEXE3.py", configure the socket server and finish the following function to receive the image sent by the client.

```
def gen(camera):
    max_len = 65507
    frame = ''
    while True:
        # receive image to the client: frame = .....

        yield (b'--frame\r\n'
            b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

In your Raspberry Pi camera client "camera_pi.py", configure the socket client and finish the following function to send image to the server.

```
def get_f():
    global camera,connection
    image = camera.get_frame()
    print(len(image))
    try:
        # send image to the server
        pass
    except:
        print("something happened in client.sendto(image,server_address)")
```

Run the two codes in different terminals, open the link in your browser and you will see the live stream.

Exercise 4a: D-pad Control of Kobuki

In this exercise, you will use the D-pad in the webpage to control the robot movement. In Exercise2b, we are able to print out the message when different D-pad buttons are pushed. In this exercise, we will send the control command to a C++ program to control the robot movement. In "FinalEXE4a.py", follow the example to send the control command of the five D-pad buttons.

```python
@app.route('/UpFunction')
def UpFunction():
    print('In UpFunction')
    cmd = 'u'
    connection.send(cmd.encode('utf-8'))
    return "None"

# define four funtions to handle the left, right, down and stop buttons
@app.route('/function_name')
def function_name():
    print('In XXFunction')
    cmd = 'XXXXX'
    connection.send(cmd.encode('utf-8'))
    return "None"
```

In the CPP file "FinalEXE4a_CPP.cpp", finish the "read_socket" which is used to read the socket data and control the robot.

```cpp
void read_socket(){
    char buffer[100];
    while(1){
        read(sock , buffer, 50);
        /*Print the data to the terminal*/
        cmd = buffer[0];
        printf("received: %c\n",cmd);

        // use cmd to control the robot movement


        //clean the buffer

    }

}
```

After finishing the code, run the three codes "FinalEXE4a.py", "FinalEXE4a_CPP.cpp" and "camera_pi.py" in different terminals. Open the link in your browser to control the robot.

Exercise 4b: Read and Display the Kobuki Sensor Data

In this exercise, you will read the sensor data sent by the robot and display the data in the webpage in real-time. In "FinalEXE4b.py", replace the string 'b0c0d0' with a correct variable so that the sensor data can be sent to the webpage.

```
@app.route('/')
def index():
    if request.headers.get('accept') == 'text/event-stream':
        def events():
            for i, c in enumerate(itertools.cycle('\|/-')):
                yield "data: %s\n\n" % ('b0c0d0')
        return Response(events(), content_type='text/event-stream')
    return render_template('FinalEXE3.html')
```

In the CPP file "FinalEXE4b_CPP.cpp", finish the code in the main function to send the sensor data.

```
int main(){
    setenv("WIRINGPI_GPIOMEM", "1", 1);
    wiringPiSetup();
    kobuki = serialOpen("/dev/kobuki", 115200);
    createSocket();
    char buffer[10];
    std::thread  t(read_socket);

    while(serialDataAvail(kobuki) != -1)
    {
        // Read the sensor data.


        // Construct an string data like 'b0c0d0', you can also define your own data protocal.


        // Send the sensor data through the socket


        // Refer to the code in previous labs.
    }
    serialClose(kobuki);

    return(0);
}
```

After finishing the code, run the three codes "FinalEXE4b.py", "FinalEXE4b_CPP.cpp" and "camera_pi.py" in different terminals. Open the link in your browser to control the robot.

Upon finishing Exercise 4, you will be able to see the live stream, control the robot movement and monitor the real-time sensor status through the webpage.

Bonus 1: Joystick Control of Kobuki

In this exercise, you will use the virtual joystick in the webpage to control the movement of the Kobuki.

In the HTML file "FinalB1.html", finish the following code to acquire the 'X' and 'Y' positional data from the joystick object and format this into a JSON message with the format: {"x": xpos, "y": ypos}

```
<script type="text/javascript" >
    var joy = new JoyStick('joy1');//,joy1Param);


    function sendJoystick(){
        // get the x axis position xpos
        var xpos = joy.GetX();

        // todo: get the y axis position ypos

        const xhttp = new XMLHttpRequest();

        xhttp.open('POST', "/joydata",false);
        xhttp.setRequestHeader("Content-Type", "application/json");

        // todo: format xpos and ypos into a JSON message with the format: {"x": xpos, "y": ypos}
        const json = {    };

        xhttp.send(JSON.stringify(json));
    }
    setInterval(sendJoystick ,500);
</script>
```

In the Python file "FinalB1.py", the following code receives the JSON data from the HTML webpage and sends to the C++ code. No coding needed for this part.

```
@app.route('/joydata',methods = ['POST', 'GET'])
def JoystickFunction():
    content_type = request.headers.get('Content-Type')
    if (content_type == 'application/json'):
        json = request.get_json()
        connection.send(str(json).encode('utf-8'))
        return "Content supported\n"
    else:
        return "Content not supported\n"
```

In the C++ file "FinalB1_CPP.cpp", finish the following code to parse the *xpos* and *ypos* from the received buffer and control the robot movement.

```
void read_socket(){
    char buffer[100];
    while(1){
        read(sock , buffer, 50);
        /*Print the data to the terminal*/
        cmd = buffer[0];
        printf("received: %c\n",cmd);

        // parse xpos and ypos from the buffer


        // use xpos and ypos to control the robot movement


        //clean the buffer

    }

}
```

After finishing the code, run the three codes "FinalB1.py", "FinalB1_CPP.cpp" and "camera_pi.py" in different terminals. Open the link in your browser to control the robot.

Bonus 2: Phone Sensor Control of Kobuki

A different way of controlling Kobuki, instead of using a Joystick, is to use motion sensors in a smartphone. Most smartphones have high resolution motion sensors such as accelerometers, gyroscopes, and magnetometers which different mobile applications use. These sensors can be accessed using JavaScript on a webpage. Remember, not all devices have these sensors.

In this exercise, you will use the phone sensor accessed by the webpage to control the movement of Kobuki.

In the HTML file "FinalB2.html", refer to the POST method used in the HTML code in "FinalB1.html" to send the JSON data pjson to the Flask server.

```
<script>
  function handleOrientation(event) {
    updateFieldIfNotNull('Orientation_a', event.alpha);
    updateFieldIfNotNull('Orientation_b', event.beta);
    updateFieldIfNotNull('Orientation_g', event.gamma);
    incrementEventCount();
  }
  function sendOrientation(){
    const xhttp2 = new XMLHttpRequest();
    var alpha = parseInt(document.getElementById('Orientation_a').innerHTML);
    var beta  = parseInt(document.getElementById('Orientation_b').innerHTML);
    var gamma = parseInt(document.getElementById('Orientation_g').innerHTML);
    const pjson = {"d": String('p'), "x": String(beta), "y": String(gamma), "z": String(alpha)};

    // refer to the POST method used in the HTML code in Lab6EXE6.html to
    // send the JSON data pjson to the Flask server.
    // todo: set up the POST method
    if (is_running){

    // todo: send pjson;

    }

}
```

In the Python file "FinalB2.py", the following code receives the JSON data from the HTML webpage and sends to the C++ code. No coding needed for this part.

```
@app.route('/phonedata',methods = ['POST', 'GET'])
def PhoneFunction():
    content_type = request.headers.get('Content-Type')
    if (content_type == 'application/json'):
        json = request.get_json()
        print("Data ", json)
        connection.send(str(json).encode('utf-8'))
        return "Content supported\n"
    else:
        return "Content not supported\n"
```

In the C++ file "FinalB2_CPP.cpp", finish the following code to parse the phone sensor data from the received buffer and control the robot movement.

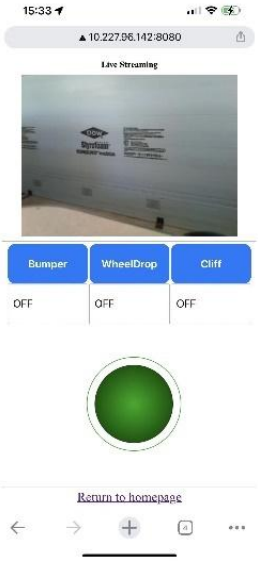
```
void read_socket(){
    char buffer[100];
    while(1){
        read(sock , buffer, 50);
        /*Print the data to the terminal*/
        cmd = buffer[0];
        printf("received: %c\n",cmd);
        // parse sensor data from the buffer


        // use the sensor data to control the robot movement


        //clean the buffer

    }

}
```

After finishing the code, run the three codes "FinalB2.py", "FinalB2_CPP.cpp" and "camera_pi.py" in different terminals. Open the link in your browser to control the robot.

Bonus 3: Combination of All Functions

Main page

Phone sensor control



Joystick control

We have learned the live stream display, real-time Kobuki sensor status display, database operation and three control methods of Kobuki. In this exercise, we will combine all functions that we have implemented into one application. In the main page, we can navigate to the other three subpages. We can also return to the main page in each subpage. The functions in different subpages remain the same as before.

Supplemental Questions

1. Briefly summarize what you learned from this lab.
2. What is the function of the JavaScript code used in this lab?
3. What is the function of the Flask used in this lab?

Submission

Submit your lab report through **Canvas**. Your lab report should include code, supplemental questions and answers of all exercises, along with the screenshots or pictures of the circuit. Put your supplemental questions and answers of all exercises, along with the screenshots in a word file or PDF file. Make sure your code is thoroughly commented. Zip all files into one when you submit.