Final Project

Goals

The goal of the final project is to demonstrate that you know how to build a simple IoT product. Your aim will be to build a home automation-style device, such as a smart door bell, security camera, thermostat, etc.

Project ideas

Lighting

Control your home lighting remotely. Have multiple room lights (a few LED's) that are easily controlled for brightness or RGB color value (remember HTML5 has a color picker element!). The lighting system can also report back how bright it is inside your home using the photoresistor in real-time. A nice interface would be a web page where clicking on an image of a home's floor plan toggles that specific room's light on/off. This can be done using the <area> element.

Thermostat

Monitor the temperature (thermistor) of your home in real-time and set furnace status on/off. The web page displays the temperature and a control or input for the set-temperature. You can simulate the furnace turning on using an LED (make sure to shut it off after a while using a Timer object.) Optionally, instead of displaying the current temperature, you could plot the temperature as a function of time (using chart.js or plotly.js).

Doorbell

Pressing a physical button tells the webpage to play a sound and begins streaming video from the camera. A button on the web page turns on the buzzer simulating that you are letting them into your apartment building. The video should then stop after some time, until the next person presses the button. Follow Chapter 16 to set up the Camera.

Remote door lock

Make a webpage that can remotely lock/unlock a door by controlling a servo (rotate it to a certain degree to specify locked/unlock). Locally you should be able to push a button or turn the potentiometer to control the door. The webpage should also display the correct status of the door.

Room occupancy counter

You can keep track of the number of bluetooth devices in a room - getting a rough count of the number of occupants. Check out this blog post. The data gathered would be sent to a web page that could plot the time history (using chart.js or plotly.js). The controls (web and local) could be a button restart the the count, refreshing the list of available devices, or clearing the plot. This would be a great project to keep running in the Makerspace!

Create your own!

You have a lot of freedom for an IoT project. Just check in with me if you have your own idea and **make sure it meets the requirements below!**

Requirements

- Interact with **one new component**:
  - Chapter 11 Photoresistor (light sensor)
  - Chapter 12 Thermistor (temperature sensor)
  - Chapter 16 Camera
  - Bluetooth software stack
  - Servo!
- Have **local controls** (buttons, rotary encoder, potentiometer, etc) and **local display** (oled, led bar, etc)
- An **interactive web interface**. Use **websockets** to send and receive data from the esp32. Local and web controls and displays should **stay in sync**.
- **Name your main script `main.py`**, that way you don't need Thonny to start your app - you just plug in the USB cable to power it and your program launches automatically. Micropython looks for `main.py` to run, otherwise it starts the shell.
- **You must demo your product during the final exam slot and clean up your lab kit**. Not doing this will result in a zero on the final project.

What to turn in

- All the code needed to run your app: python + html + any other files.
- A 1-2 page product summary. You don't need to walk me through how you made it, just explain the functionality, some images to show it off, and most importantly some advertising for why everyone should buy it ($$$)!

Rubric

Item	Percentage
Runs without error	30
New component	10
Local interface	20
Web interface	20
Product summary	10
Code quality	10

New component
: Did you use a new electronics component or bluetooth?

Local interface
: Do the local controls allow hardware changes? Does the local display accurately show the state of the hardware?

Web interface
: Do the web elements allow you to control hardware? Do you display the hardware status? Do the controls and display stay in sync with the local interface/hardware?

Product summary
: Make sure its clearly written. Explain its main functionality and how it works.

Code quality
: Is the code well-organized, with **separate components in separate files**? Do you have helper

function and classes to make the main script easy to read? (see my advice below)

Advice

- Have fun! You have a lot of room for creativity! Try to make a polished product - this is a great project to put on your resume!
- Make little classes for each major component in your product and place them in separate files. This is nice organization and makes it easy to test individual components of your product. **Write small scripts to test the individual parts of your program.** Then the final product is gluing together components!
- Work incrementally! Build one component, test it locally, then add a web interface, repeat!