Final Project Report and Submittal

Written Report

1. Summary of the project

The system delivered shall be able to run four devices that use AC power as well as provide a graphical user interface that can be used to set the parameters of the operation. This system will be used to automate the process of cleaning and hardening resin molds for Dr. Swapp.

2. Description of the objectives of the project
    a. The system should provide the ability to safely and conveniently apply power to the four devices.
    b. The system should provide a manual switch for each component so that each component can be turned on or off with this switch.
    c. The system should provide a graphical user interface with the following capabilities:
        i. Individual component on/off control
        ii. Individual component start/stop times and/or durations. This aspect of the user interface needs more thought.
        iii. Ability to set a program and allow the program to run at particular time intervals or at particular times.

3. Description of the chosen hardware platform and why you chose it

The Beagle Bone Black wireless was chosen due to its JavaScript and webserver capabilities. It also was chosen for its ability to utilize a Wi-Fi soft access point. This was helpful so that Dr. Swapp didn't have to jeopardize his office Wi-Fi security.

4. Description of the hardware design

The logic-level converter and a 6-pin header, to provide the inputs for the relay module, were soldered on a ProtoCape for the BBBw. A 6-wire ribbon ran from the header on the ProtoCape to the relay module's header pins. The power to the logic-level converter and the relay module were both provided from the power pins on the BBBw. All the wiring for these connections was soldered underneath the ProtoCape to keep the top clear of clutter.

The power to the device was provided by a standard wall outlet, and the power was distributed in the device through terminal strips. The first terminal strip split the power from the single input into four line, neutral, and earth wires for the four outputs. The earth and neutral wires go straight to the output terminal strips, and the line goes to the 3-position switches. From the 3-position switch it is split to either go directly to the output, for manual control, or to the relay module, for BBBw control. After the line input comes out of the relay or three position switches, they come together on another terminal strip alongside the earth and neutral. From here there are four outlets that are powered from the strips.

5. Description of the chosen software platform and why you chose it

The software used was HTML for the client, JavaScript for the server, and sockets.io for the communication between them. The HTML and JavaScript interacted well and were very easy to work with. Helpful information was provided online making it very easy to teach oneself how to use this software.

6. Description of the software design

HTML Design

The HTML utilizes tables and forms in order to create the input system. Each device has its own row in the table. There is also a column devoted to the pin state of each row. The user can input numbers (in seconds) into the column and can then submit them to the server by pressing the submit button on the bottom of the form. The submit button calls a JavaScript function that handles the input from the user and sends the information to the server using socket.emit. The socket.emit function sends a specific text string to the server identifying the coming data object which is made up of the user specified information. There is also a clickable box on the bottom of the webpage for the user to click to stop the system. This calls a socket.emit function that simply sends a systemStop string to the server.

The handlePinUpdate function is called whenever the client receives a pinUpdate message. This message is accompanied with an object that contains all of the information for the updated pin states. These are sent at a specified interval by the server. The data received from the server is a pin value and its corresponding pin value. The client interprets this object and updates the webpage values using the document object model, which is an API for modifying the values output on the webpage.

JavaScript Server Design

The start of the JavaScript includes going through each of the required libraries and including them. This creates the bonescript, io, app, and fs object. The bonescript object object allows for interacting with the pins and specific hardware components of the beaglebone. The app object sets up the http server and handler. The io object uses sockets to enable communication between the client and server on a specific port of the ip address. The fs object allows for the use of the Node.js file system.

From there, the pins are initialized to high (1) to start. High is off for our switch system. This helps to keep devices from turning on randomly on start up. Variables are also made with string values to denote each device. This helps later with identifying which pins match up to each device. There are also timer and off variables for each device, these are used to keep track of the setTimeout callbacks.

Next is the pinStates object. This object stores the pin states and is updated when the pin is changed. This approach is used in alternative to interrupts as the interrupt system isn't reliable in bonescript. There is also a variable that is the file path to the index.html values that is to be sent to the client. There is a soc variable that is defined and used later to send data to the client. The app.listen function is used to determine which port to use for communication. In this program, it is the 8085 port. When used with a soft access point, this webpage can be accessed by typing 192.168.8.1:8085 into the address bar of a web

browser. The handler function takes incoming requests and handles them. If there is an error, it springs an error code. The handler function is not modified for this project.

The onConnect function comes next in the code and redirects to specific functions depending on the message received from the client. The two options inside onConnect for this program is systemStart and systemStop. The systemStop message calls the handleSystemStop function. This function turns each pin to high (off), updates the pinstates variable, and clears any timeout that has been called.

The handleSystemStart starts each device using setTimer interrupts attached to a variable. This variable calls the startComponent function, this function is called when the interrupt reaches the start time. The startComponent function sets the pin of the device in question to low (on) and then sets another timer for the stopComponent function. Once the timer interrupt occurs, the stopComponent function is called which would set the pin high (off).

There is also a sendOutput function that is called every 50 milliseconds that sends the pin states to the client for processing by using soc.emit in order to send data to the client.

7. Test Results – what worked and what didn't

After extensive testing, everything operates as expected. The manual control for the devices works and so does the Web interface control through the BBBw. Also, all four devices can be run simultaneously. One of the only issues is that the devices have to have a start time in order to work properly. If one device isn't run, it causes issues with the other devices.

8. Lessons learned – things you would've done differently if you were starting over

From the hardware side, the original plan was to include an opto-isolator between the logic-level converter and the BBBw, but that was removed because it was discovered that the relay module already had built-in optoisolators to protect any microcontroller that was connected to it from back emf from switching the relays. Also, although it was better to be cautious, originally the circuit included a barrel jack for a 5V input that was going to power the relay module and logic-level converter, but after experimentation it was discovered that the BBBw power pins were able to provide the power without burning out.

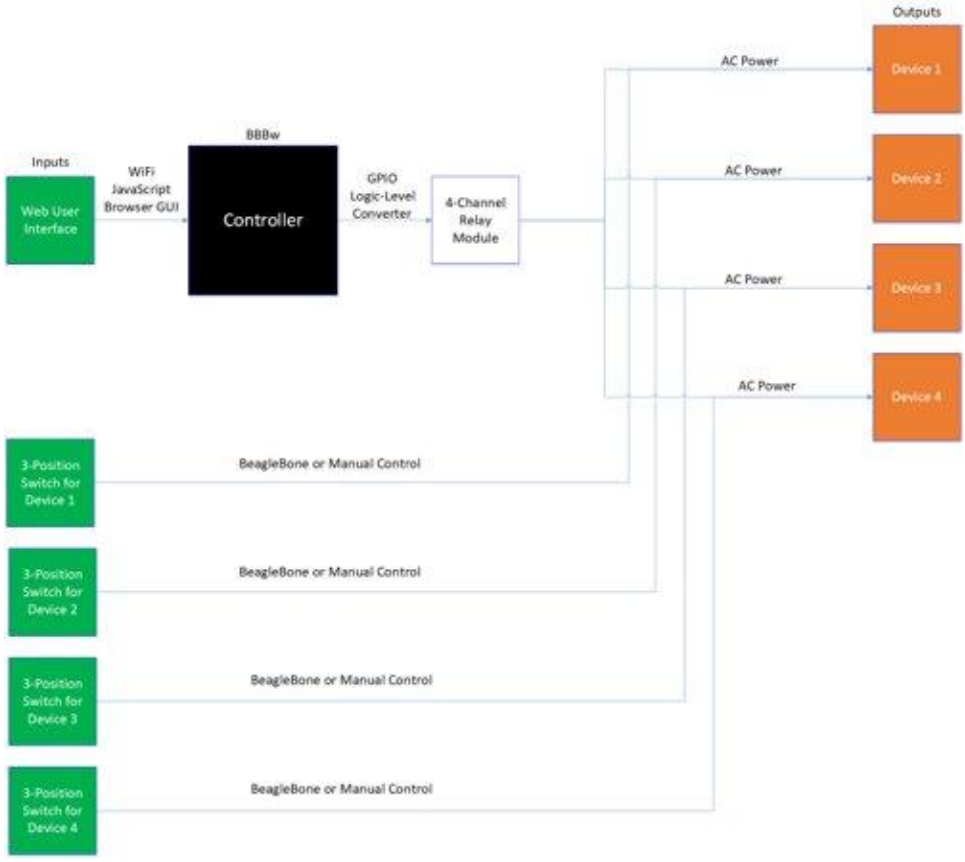
On the software side there was a lot to be learned but I found out that there is a lot of resources for HTML and JavaScript on the internet. I would have made sure to do more research and read through the programming BeagleBone book earlier. In general, I didn't have a whole l wish I would have figured out earlier. I do wish that I would have been more aware of the object-oriented nature of JS, but other than that everything progressed quite smoothly.

9. Final notes – this should include anything else necessary for me to reproduce your project without any problems.
    a. AC power can be complicated and dangerous. When working on a device, make sure that power is not supplied if you are messing with wires.
    b. Make sure that the AC power is contained and not able to be touched easily by anyone.

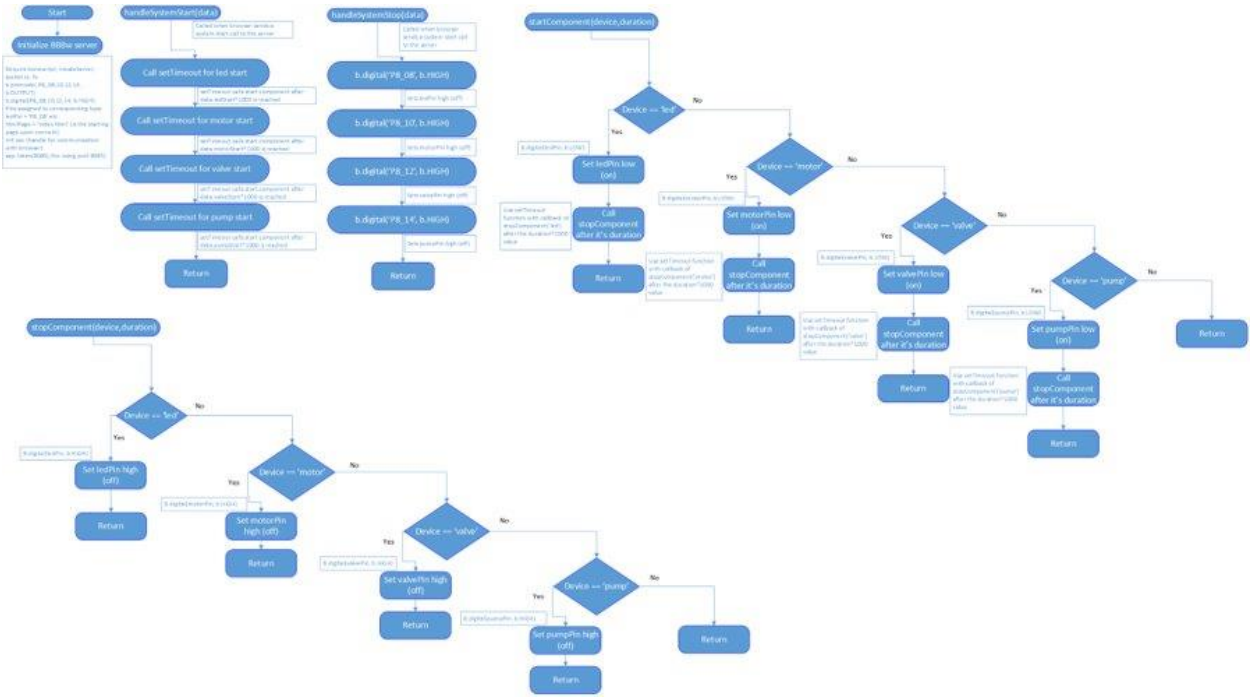
c. Be careful when powering the BeagleBone as it can be easily destroyed by too much current
d. Ensure that the ortho.js file is placed in the autorun folder of cloud9 in order to run this program on startup. Also make sure to have the correct path in ortho.js to the index.html file. I prefer to use the entire pathway so that I know it is correct. This may need to be changed on your BeagleBone in order to operate properly.

Appendix

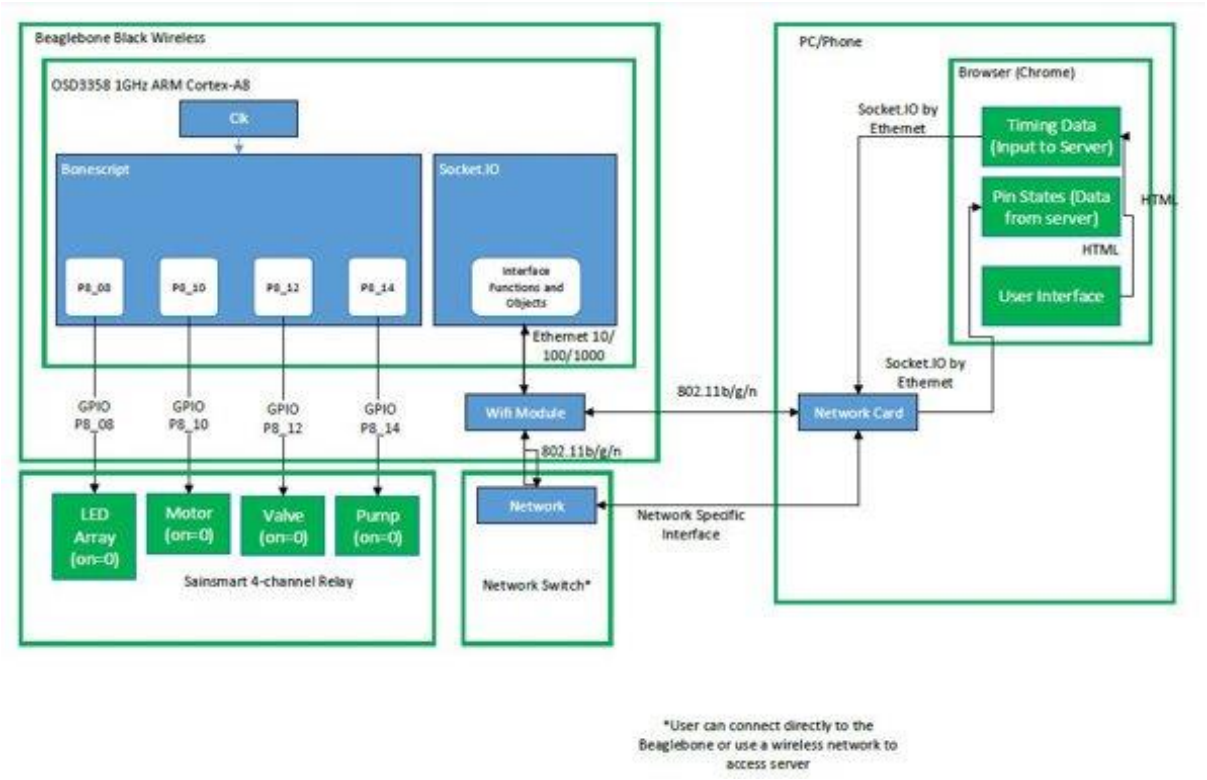
1. Software Design Documentation
    a. Up to date block diagram



    b. Up to date activity diagram(s)

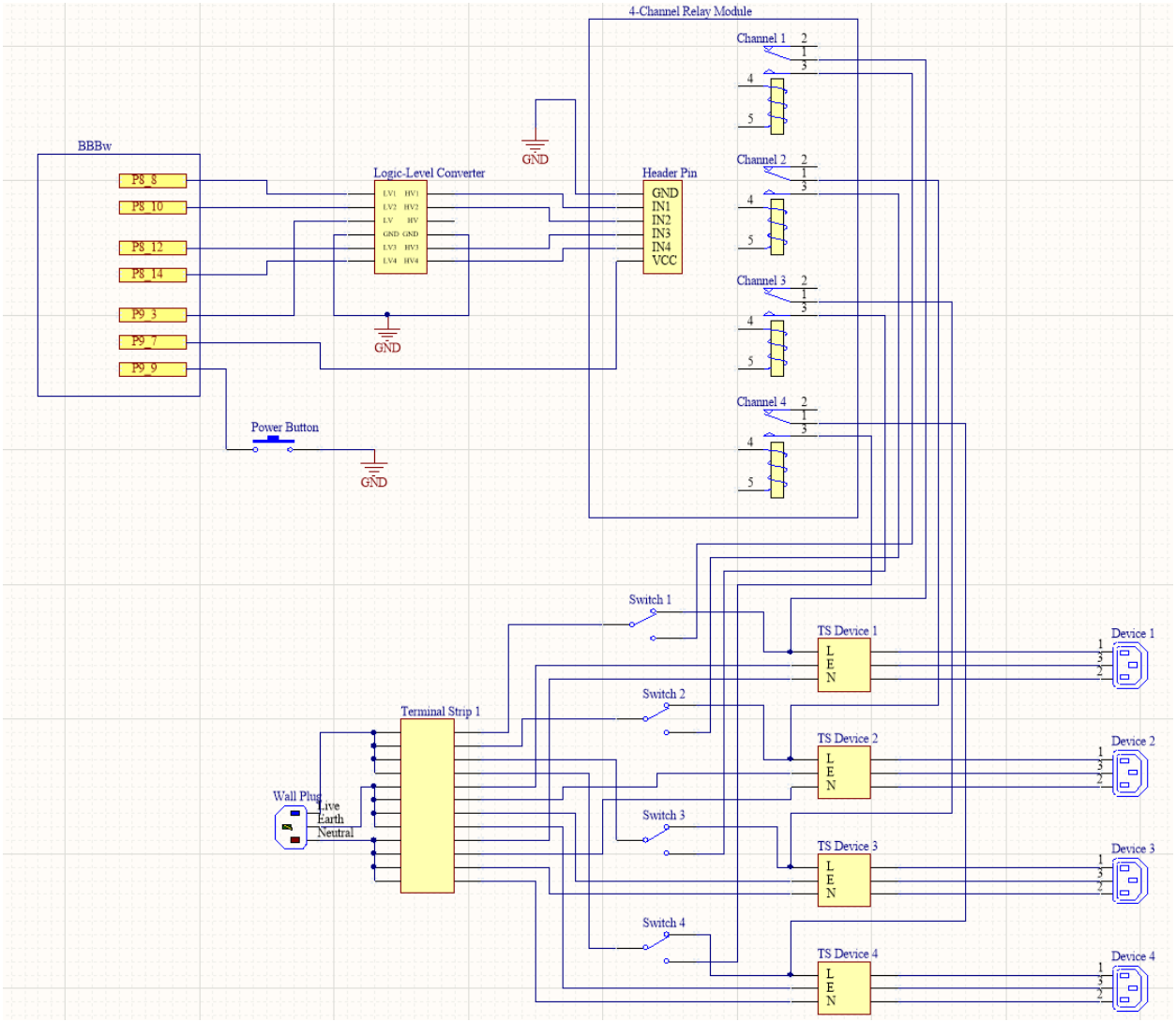


c. Up to date functional architecture



1. Source code
    a. All code needed to get this project working can be found at https://github.com/OsoGrandisimo/ortho. This hyperlink includes the sockets.io module that is required for server/client communication

2. Hardware Design Documentation
    a. A schematic of hardware implementation



    b. Parts list

Qty	Descriptive Name of Part	Vendor	Part Number	Unit Cost	Total Cost
1	4-Channel Relay Module	Amazon		9.99	9.99
4	3-Position Switch	Digi-Key	EG2350-ND	2.24	8.96
1	Logic Level Converter	SparkFun	BOB-12009	2.95	2.95
2	Terminal Strip	Baylor		0	0
1	ProtoCape for BBBw	DigiKey	1568-1261-ND	11.95	11.95

1	BeagleBone Black Wireless	DigiKey	BBBWL-SC-562-ND	76.13	76.13
1	5V DC Wall Adapter	Amazon		3.09	3.09
2	3-Prong Power Cord	Amazon		7.43	14.86

c. Datasheet links
   i. 4-Channel Relay Module Link: https://www.amazon.com/SainSmart-101-70-101-4-Channel-Relay-Module/dp/B0057OC5O8
   ii. 3-Position Switch: https://www.e-switch.com/system/asset/product_line/data_sheet/129/100.pdf
   iii. Logic Level Converter: https://cdn.sparkfun.com/datasheets/BreakoutBoards/BSS138.pdf
   iv. ProtoCape for BBBw: https://cdn.sparkfun.com/datasheets/Dev/Beagle/CAT24C256-D.PDF
   v. BeagleBone Black Wireless: https://github.com/beagleboard/beaglebone-black-wireless

3. Hardware Implementation
   a. Outside of the box

   b. Inside of the box

i. It is important to note that the components will be screwed down and secured before submitting the final device.