ECEN 4910 – Integrated Systems Programming

Final Project: Web-controlled Fish Feeder

Authors: Riley Hester & Michael Goberling

Due: 12/14/2017

1 Objective

The objective of this project was to demonstrate a growth in knowledge of IOT devices and applications by integrating multiple systems to achieve a desired outcome. This project is an implementation of a web-controlled fish feeding mechanism that combines knowledge and skill in Debian Linux, BeagleBone Black (BBB) hardware, JavaScript, NodeJS, Python, HTML, and PWM.

2 Background

Home automation has become universally popular. From connected cleaning robots to wireless light fixtures, there are many neat devices that allow us to control various features in our homes from our phones, computers, or tablets. The project team identified a need to control tending to aquatic pets and decided to design, model, and implement a web-controlled automated fish feeder using BeagleBone Black (BBB). A preliminary system design model is shown below.

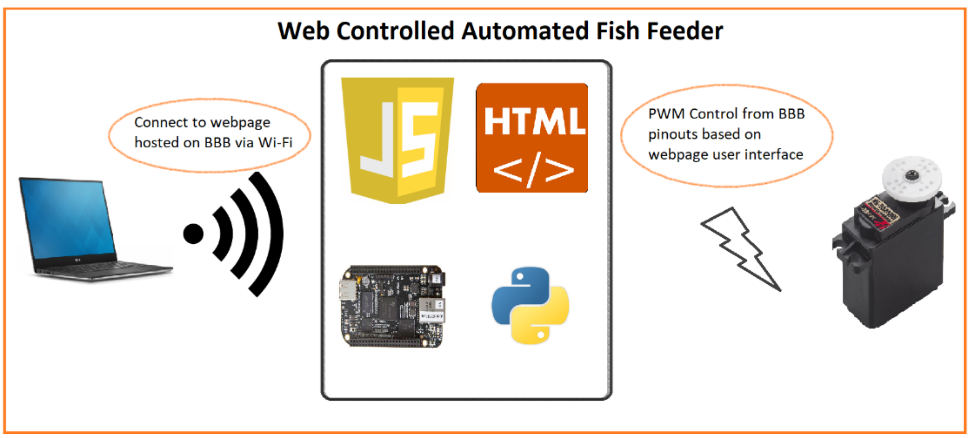

Figure 1: System Design

To achieve this system, the project team researched implementing a web server on the BeagleBone Black (BBB) using JavaScript. The team used NodeJS for these purposes. To facilitate this project's IOT standards, the project team decided to implement connection to the web server via Wi-Fi.

Web server handling was performed using JavaScript. The web server was a simple HTML page. PWM control was implemented via an Adafruit Python library built for BeagleBone Black (BBB). The Python script was triggered via a shell command that our JavaScript code would tell to execute given some user input.

Pulse Width Modulation (PWM) is a power control method that allows for adjustable throughput to control and manipulate the intensity, position, or speed of lighting or electromechanical components. Designers can control these attributes by adjusting the percentage duty cycle of a digital signal to a component over time.

2.1 Resources

The following resources were used to complete this project:

- BeagleBone Black
    - Cloud9
    - Nano
    - Vim
    - Debian Linux
- JavaScript
    - NodeJS
    - JSON
- Python
    - BBB Adafruit Library
- HTML
- Edimax EW-7811Un 150Mbps USB Wi-Fi Adapter
- HITEC HS-422 Servo Motor

3 Procedure

As neither team member had prior exposure to BBB, the first step in this project was to connect to the device and become comfortable navigating its filesystem. The BeagleBoard website features an easy guide to follow for connecting to the BBB, which was followed until connection of the board was successful.

Next, it was important to confirm that PWM could work on the BBB. To achieve this, the team consulted a BBB pinout diagram to find a GPIO pin that could handle PWM duties. The group decided to use Pin 14 on P9 (*EHRPWM1A*) for PWM, Pin 2 on P9 (*DGND*) for ground, and Pin 6 ON P9 (*VDD 5V*) for power.

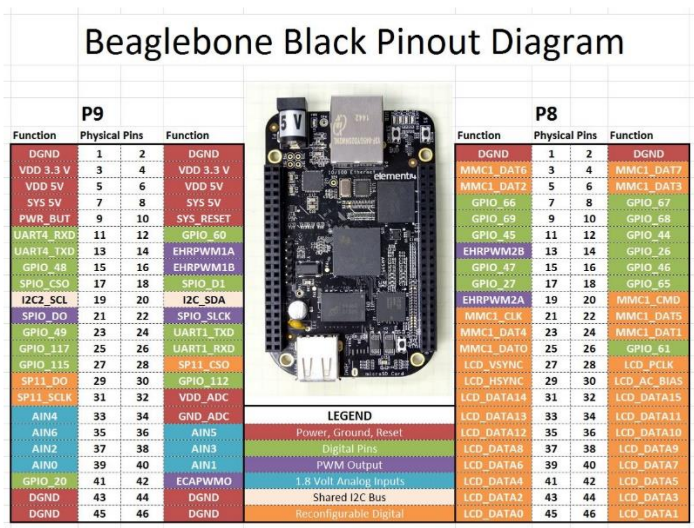

Figure 2: BBB Pinout Diagram

After selecting the appropriate pins for PWM purposes, the team wrote a Python script using the well-documented Adafruit BBIO library to control the PWM to confirm its operation. To feed an aquatic animal, all the team needed was to essentially tip the food-holding mechanism, which meant finding two extreme values for the servo to rotate to. After experimentation, the team found that writing the duty cycle to 12% and 3% allowed the servo to turn to positions that were suitable for dumping the contents of a proverbial food-holder. To facilitate position destination arrival, the team added a 1 second delay between position switching. The Python code is shown below.

```python
import Adafruit_BBIO.PWM as PWM
import time
import sys
servoPin="P9_14"
PWM.start(servoPin, 5, 50)
PWM.set_duty_cycle(servoPin,12)
time.sleep(1)
PWM.set_duty_cycle(servoPin,3)
```

Figure 3: PWM Python Script

Once the PWM-controlled servo motor was operating properly, the next step was to set up the BeagleBone to be controlled from the web. A Javascript file was created that allowed the BeagleBone to host an HTML page on a web server that would be used to send the command to activate the servo routine. This Javascript used the *http* module to create the server and the *socket.io* module to communicate between the server and the BeagleBone.

```javascript
// fish_server.js

var app = require('http').createServer(handler);
var io = require('socket.io').listen(app);
var fs = require('fs');
var bb = require('bonescript');
const exec = require('child_process').exec;

var htmlPage = '09_02_pwm_fishfeeder_2.html';

app.listen(8085);

function handler (req, res) {
  fs.readFile(htmlPage,
    function (err, data) {
      if (err) {
        res.writeHead(500);
        return res.end('Error loading file: ' + htmlPage);
      }
      res.writeHead(200);
      res.end(data);
    });
}

function onConnect(socket) {
    socket.on('exec', handleExec);}

function handleExec(message) {
        var data = JSON.parse(message);
        exec(data.command);
}

io.sockets.on('connection', onConnect);
```

Figure 4: PWM Javascript file

The accompanying HTML file consisted of a title and a button that would execute a script function named *feed()*. The *feed()* function would then communicate a JSON formatted command to the BeagleBone to execute the Python file stored on it.

```
<html>
        <head>
                <script src = "/socket.io/socket.io.js", "time" > </script>
                <script>
                        var socket = io.connect();
                        function feed(){
                                socket.emit('exec', '{"command":"python feedertest.py"}');
                        }
                </script>
        </head>

        <body>
                <h1 style="font-size:60pt;">Fishy Hungry???</h1>
                <input type="button" onclick="feed()" value="Feed da Fishy!"/>
        </body>
</html>
```

Figure 5: PWM HTML file

5 Outcomes

After setting up the BBB-hosted web server and connecting to the accompanying web address, the servo motor can be moved by interacting with the button on the webpage. The servo motor simply turns to one polar extreme, and back to another.

The project team gained valuable experience regarding BBB hardware, Debian Linux, JavaScript, HTML, Python, PWM, and web hosting. The team feels more prepared to attack IoT problems using embedded hardware.

6 Problems Encountered

Originally, the plan was to implement the web server and send outputs to the BBB hardware strictly via JavaScript by sending and parsing JSON to the server. After attempting to implement this solution and uncovering that the implementation had consistency issues, the team decided to make a switch. The project team theorizes that the issues were related to the default frequency of the PWM and its inability to be changed by JavaScript.

Additionally, the BBB can experience USB port transmission issues due to interference from the HDMI port conveniently placed just beneath the USB port on the board. To remedy these issues, the HDMI port was disabled via software.

7 Conclusion

Execution of this project demonstrated a growth in knowledge of IOT devices and applications on behalf of the project team by integrating multiple systems to achieve a desired outcome. The fish feeder concept was successful in nature as a minimum viable system.

7.1 Future Work

The project team would be interested in continuing to develop this solution. To achieve this, the team would need to find or develop a food-holder that could be fastened to the servo motor and devise a way attach the servo motor to the accompanying aquatic tank.

8 Appendix

Required files have been compressed into a zip file with this report.