CS Capstone Mid-Winter Progress Report

March 20, 2018

Project LOOM

Prepared for

Kevin McGrath
Kirsten Winters

Prepared by

Luke Goertzen
(Group 36)

Abstract

This document is an individual report of my progress for Winter Term. The report briefly reiterates the purpose and goals of the project before presenting its current status and a weekly breakdown of how this state was reached. Also included is an evaluation of myself and my group members.

Contents

List of Figures

List of Listings

1 Recap of Project Purpose and Goals

With Project LOOM, we aim to create an open-source, plug-and-play, suite of modular building blocks, the extensible and easy programmability of which expands the demographic of people capable of implementing Internet of Things solutions. For users with limited technical expertise to create complex systems, we aim to build a system that abstracts out the more technical details, allowing them to focus on their system more than the implementation of the modules. The system should also be usable by higher level students and experts by allowing them to modify or write their own firmware, and create new modules.

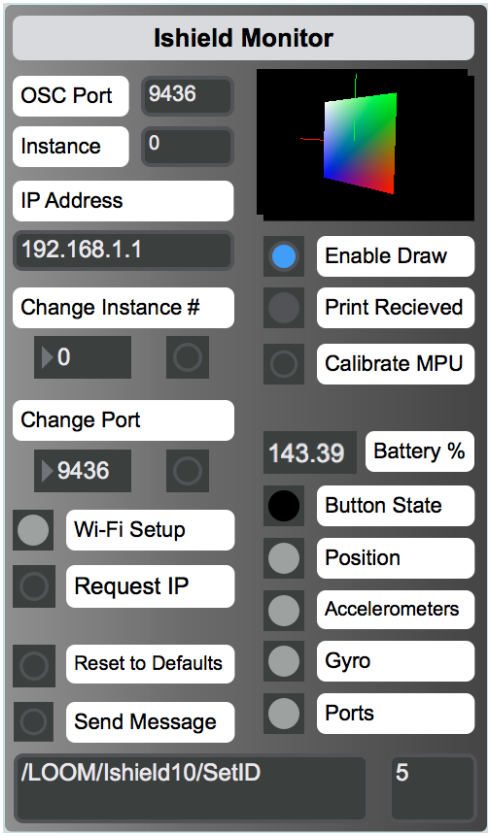
2 Current Project Status

As usual, progress this term has continued to be steady, as all associated parties of LOOM – that is, the CS team, the ECE team, our client (Chet Udell), and other members of his lab – work diligently and help each other in meeting the expectations of our client and respective classes. This is despite Chet being mostly out of the lab lately, due to an injury. In the last progress report, we noted that the current focus of the teams' bandwidths were towards preparing for the OPEnS Lab open house and Internet of Things workshop. These went well, and as a result of this preparation and the feedback from the events, we now have a reliable and feature rich master code, configurable for whichever devices (among the ones we support: Ishield with gyroscope, Neopixel, relay, servo, soil moisture sensor, and arbitrary analog sensors) are being used, a clean and refined set of interfaces for the different devices, and a number of flexible demos (which are easily created or modified) to show off the current extent of the project. These events were helpful both in terms of getting feedback on our project from a user's perspective, and for the practice demonstrating and explaining our project to a variety of people.

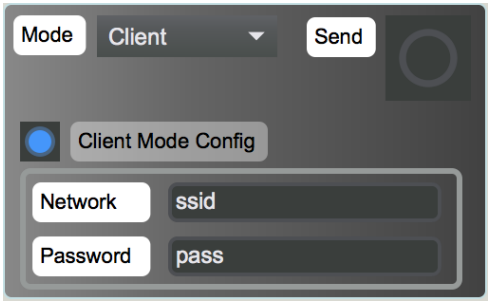
Below, I will go into further detail of the current state of various parts of project I have worked on (note, however, that roles in our team tend to be rather flexible, and as such, few components of the project have been worked on by exclusively one person).

2.1 Max Interfaces

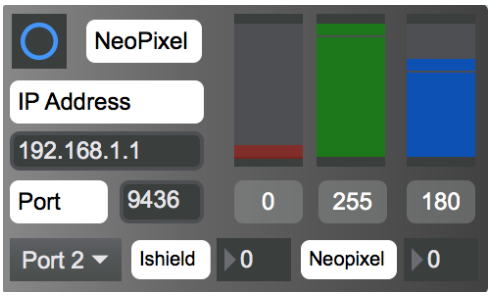
The following images present the current state for the primary interfaces of the devices we use. The Ishield Monitor is the centralized interface for viewing the status and inputs, and controlling the configuration of, a Feather M0 with an Ishield. The Neopixel control module is used for controlling Neopixels plugged into Ishields. The servo and relay modules control those types of devices, respectively. The WiFi configuration window can be opened for the Ishield Monitor, servo, and relay controllers.



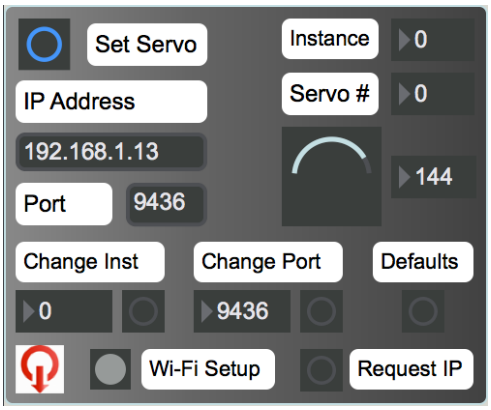
(a) Ishield Monitor



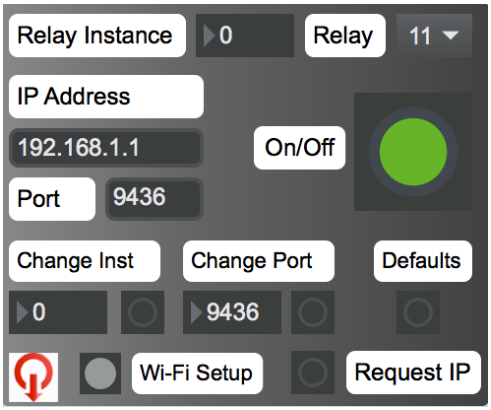
(b) WiFi Config Window



(c) Neopixel Control Module



(d) Servo Control Module



(e) Relay Control Module

Fig. 1: Max/MSP Control Module Interfaces

2.2 Firmware

While my focus within the project is on the software and interfaces, I still work a fair amount on the firmware for our devices, often to add functionality corresponding to an update or addition to the Max modules. Development of these features is generally done in relatively isolated files, to be merged into the master file later. As the project stands now, nearly all of the code we have worked on is now contained within a core set of files (one main file, a configuration file, and and few auxiliary files containing extra functions). Soon, these files will be further organized as to separate

functions specific to a given device into their own header files. The following sections of code are a few of the more interesting functions that I have worked on.

The routing function is primarily of Trevor's design, but I have added to it as new types of messages that can be expected are added. If a bundle that is received by the device has the correct family and device header prefix (the expected header is dynamically constructed based on config file), the message gets passed to this function. The function shows a number of common functions that the message may be routed to, for operations such as changing the WiFi information or port numbers, as well as device-specific actuator control functions. The latter are enclosed within #ifdef's so that the preprocessor will eliminate unnecessary code when compiled and flashed to the devices.

Listing 1: OSC Bundle Routing Function

```
void msg_router(OSCMessage &msg, int addrOffset){
  #if DEBUG == 1
  char buffer[100];
  msg.getAddress(buffer,addrOffset);
  Serial.print("Parsed ");
  Serial.println(buffer);
  #endif
  #ifdef is_servo
  msg.dispatch("/Servo/Set",set_servo,addrOffset);
  #endif
  #ifdef is_relay
  msg.dispatch("/Relay/State", handleRelay, addrOffset);
  #endif
  #ifdef is_mpu6050
  msg.dispatch("/MPU6050/cal",calMPU6050_OSC,addrOffset);
  #endif
  #ifdef is_neopixel
  msg.dispatch("/Neopixel",setColor,addrOffset);
  #endif

  msg.dispatch("/Connect/SSID",set_ssid,addrOffset);
  msg.dispatch("/Connect/Password",set_pass,addrOffset);
  msg.dispatch("/wifiSetup/AP",switch_to_AP,addrOffset);
  msg.dispatch("/SetID",set_instance_num,addrOffset);
  msg.dispatch("/SetPort",set_port,addrOffset);
  msg.dispatch("/requestIP",broadcastIP,addrOffset);
}
```

The broadcast IP address function is an interesting protocol I designed to address a problem that arose in practical use, but was not present in typical development. Essentially, when a device connects to an existing WiFi network (after transitioning from access point mode or power off) an IP address assigned to it. In development, this behavior is fine, as the device is plugged into a computer and reports back the IP that is was assigned via Serial. However, in normal use, the information would not be visible, and the device cannot automatically tell the computer what its address is, especially if they are not even be on the same network yet. My solution is that once a user connects to the network, they press a button to request the device's IP, and only the correct device on the network will respond, sending back a

message with its IP. In Max, both the module that requested the data and any linked modules will automatically update with this new information.

Listing 2: Broadcast IP on Request

```
void broadcastIP(OSCMessage &msg){
    configuration.ip = WiFi.localIP();
    char addressString[255];

    sprintf(addressString,"%s%s",configuration.packet_header_string,"/NewIP");

    OSCBundle bndl;
    bndl.add(addressString).add((int32_t)configuration.ip[0])
                           .add((int32_t)configuration.ip[1])
                           .add((int32_t)configuration.ip[2])
                           .add((int32_t)configuration.ip[3]);

    Udp.beginPacket(configuration.ip_broadcast, configuration.localPort);
    bndl.send(Udp);  // send the bytes to the SLIP stream
    Udp.endPacket(); // mark the end of the OSC Packet
    bndl.empty();    // empty the bundle to free room for a new one

    #if DEBUG == 1
    Serial.print("Broadcasted IP: ");
    Serial.println(configuration.ip);
    #endif
}
```

Recall that the Neopixel is a digital LED with variable color, and has been a common means of visually indicating correct functionality of some code. The function and associated data structures below are used to control such Neopixels, on different ports of an Ishield, or with daisy chaining. The code as it is now is a notable improvement over past iterations, using arrays for organization and easy extensibility, with the configuration file being used to specify where and how many Neopixels are plugged in. This prevents trying to set the color of a Neopixel where there is none plugged in, which could be problematic if there was instead another device connected to the specified port.

Listing 3: Refined Neopixel Control Function

```
#ifdef is_neopixel
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel * pixels[3];
bool pixel_enabled[3] = {NEO_0, NEO_1, NEO_2};
int colorVals[3][3] = { {0,0,0},
                        {0,0,0},
                        {0,0,0} };

void setColor(OSCMessage &msg){
    int port     = msg.getInt(0);
    int pixelNum = msg.getInt(1);
    int color    = msg.getInt(2);
    int val      = msg.getInt(3);

    ... Debugging prints ...

    if (pixel_enabled[port])
        colorVals[port][color] = val;

    if (pixel_enabled[port])
          pixels[port]->setPixelColor(pixelNum, pixels[port]->Color(
                                                colorVals[port][0],
                                                colorVals[port][1],
                                                colorVals[port][2]));
    for(int i = 0; i < 3; i++) {
        if (pixel_enabled[i])
            pixels[i]->show();
    }
}
#endif //END OF NEOPIXEL FUNCTIONS
```

3 Weekly Breakdown

3.1 Week 1

Summary - This week, we reviewed where the project was at the end of Fall term, what progress had made over break, and arranged meeting times for the term with our TA, client (Chet Udell), and the ECE team of Project LOOM. We identified which aspects of the project needed further development or refinement before we moved on, ensuring that the current state of the project was a fairly reliable foundation. Chet also informed us that the upcoming faculty demo would likely be pushed back a few weeks.

3.2 Week 2

Summary - This week, we had our first TA meeting of the team; worked on improving the state of our servo code and functionality; and began incorporating relays into the project, adding relevant new interfaces and code.

3.3 Week 3

Summary - My work this week was focused on refining and adding new Max control and processing modules for servos, relays, and soil sensors. I also helped Trevor with the the process of implementing client mode WiFi into our device firmware (starting with code for the Ishields). That is, instead of the device creating its own WiFi access point that a user can connect to, the device and computer can both join an existing WiFi network. This will also enable a computer to communicate with multiple devices at once, which is not possible in AP mode.

3.4 Week 4

Summary - This week, I directed my attention towards redesigning and significantly improving the Ishield monitor, also ensuring that it had the functionality to let us specify the WiFi mode of connected devices, sending network information as needed. On the firmware side, I ported the servo code into a template that Trevor made for adding client mode functionality. To demonstrate that we had client mode working, we connected two devices to the lab's WiFi network, using the gyroscope on the Ishield to control the rotation of a servo.

3.5 Week 5

Summary - This week, I further improved the Ishield monitor, merging the functionality of other modules into it, thus reducing the total number of modules a user might have to use. I also added the Wifi config from this to the servo module. Other progress included further porting of existing code to support client WiFi mode, and, as a side project, I got Neopixel daisy chaining to work. We also began working on the poster.
Problems - I cannot get Neopixel to change color on more than one port (daisy chaining is fine, however). The servo code seems to have something wrong with it (lag and freezing) after working previously.
Solution - Between Trevor and I, none yet.

3.6 Week 6

Summary - This week, the team primarily worked on getting our progress report put together, presentation recorded, and poster designed, somewhat limiting the time we had to spend on more progress. However, we did verify the functionality of the gyro-to-pixel demo across two devices both running in client mode on the same WiFi network. The servo version of this demo has continued to be somewhat troublesome though, and we do not know the source of the problem yet.
Problems - The servo code still broken, but we have narrowed it down to the changes implemented for client mode WiFi.
Solution - Still not sure what exactly is the issue.

3.7 Week 7

Summary - This week, I focused on getting demos ready for the upcoming open house of the lab and the IoT demo, in addition to continuing the process of get everything working in client mode. I also worked on a number of general improvements, retroactively adding some of the Ishield monitor features to the actuator modules, and adding code/Max code to have a device reply with its IP address upon request - which is very helpful when changing to client mode and the device gets assigned an IP address (I based this on the ARP protocol). The WiFi "template" now has the code for all major devices we currently use in a one location.
Problems - How to send IP address structure from devices.
Solution - Trevor noted that the structure is, roughly, just an array. As such, I can just send the four numbers of the IP address as an OSC message with four arguments.

3.8 Week 8

Summary - I made a significant amount of progress this week. A number of refinements were made to the primary Max modules, and dynamic instance numbers and ports should now work for each device / Max module pair. I also have more elaborate demos (for use next week) prepared using these improved modules.

3.9 Week 9

Summary - This week was somewhat atypical in that we had two demonstrations / workshops of our project, the open house of the the lab, and an IoT workshop (which Kirsten attended). Preparing for these helped us resolve issues we were not yet aware of and make sure that the master version of the WiFi template worked for all devices (by only changing the config information). The demos indicated the reliability and capabilities of our system, as well as helping us find errors in edge cases we had not thought of ourselves. The devices are now configurable to start up with a provided default network to try to connect to, and can store state to seamlessly restart after being turned off. I also resolved an issue in which Max would persistently crash for Windows users.

3.10 Week 10

Summary - This week, I made a handful of small changes to implement suggestions made by people who attended the IoT workshop or fixes that we noticed should to be made. I talked to Chet about making alternate views of the interface and operation so that people with low technical expertise can have a simplified means of setting up and controlling their devices with minimal difficulty. We are also making sure to go back through code and ensure that everything is sufficiently commented so that people can easily understand it. I also began working on the term progress report and presentation.

4 RETROSPECTIVE

Positives	Deltas	Actions
The collective group of Project LOOM continues to be well organized and communication is good, keeping progress steady	Not essential, but we would like a config file generator with a user interface and/or command line executable	Trevor and I will continue investigating this. He has the start of python script that will generate configs, I've been looking into running this and compiling and uploading code to devices without the use of the Arduino IDE, possibly via Max
Devices successfully run in client WiFi mode	Need code for non-WiFi communication to be more refined	Work on it more (not my focus though)
Devices can store configuration to resume seamlessly after restarting	Do more interesting this will linked software / services	Now that the code pretty solid, I can work on improving the external logging / visualization features
MaxMSP modules are fully functional on everyone's computers now	Further modularize the main code	Put functions of specific devices/shields in their own files to simplify preprocessor statements and for ease of finding relevant code

5 First User Study

This term, we had two demos/workshops of our project, allowing us to explain our project to potential users and non-users alike. The first of these was the open house for the OPEnS lab (the Openly Published Environmental Sensing lab, which Chet is the director of) and the projects worked on within it. Demonstrating our project here helped us figure out how to better explain our project, especially to people who are not already at least vaguely familiar with our project, or are not particularly technically knowledgeable. The event went well enough for us, but it brought to our attention the improvements would need to be implemented for better reliability and simpler setup configuration, especially in the context of demos.

The second demonstration of our project, the Internet of Things workshop, was held a few day after the open house and focused specifically on the Project LOOM, without the other work of the lab present. This event was intended primarily for faculty members interested in the project, including Kirsten, and those with plans to use the devices, in courses or otherwise. In the days between the two events, we were able to resolve a number of reliability issues, implement simpler configurability (flashing the configured firmware to the nine demo boards took but several minutes), and get the devices' configuration to persist after being powered off. As such, the demos at this event were better organized and more comprehensive of the work we have done. This included all the sensors and actuators that we have worked with so far. Additionally, we ran the devices on battery power instead of our typical development setup with power

being supplied from a computer. We had demonstrations of how to design a Max ”program” to easily design or alter the functionality of a network of devices, and how to understand the flow of data streams present.

In preparation for and as a result of these two events, our project notably improved in refinement and usability. We have further discussed what else we may want to add in terms of new features, or to provide simplified views of the existing system for those who do not need or want as many of the details provided (e.g. ports and IP addresses).