Final Report

Senior Design Group: May1624

Advisor/Client: Dr. Tom Daniels

Assistant: Brian Gillenwater

Team Members: Niklas Jorve, Branden Sammons, Jonathan Krueger, Yidong Liu, Xinian Bo

Table of Contents

Project Plan

Problem Statement

As an entry-level programming class, CprE 185 has designed labs that seek to combine real-time hardware readings with beginner level programming in “C” language for computer engineering, software engineering, and electrical engineering students. The current labs use an Arduino Esplora game pad or a Nintendo Wii remote to gather data from infrared sensors, gyroscopes, accelerometers, and button inputs and transmit the information back to a computer via USB or Bluetooth. Due to the limited mobility of the Esplora and the connectivity issues with the Wii remote, the goal of this project will be to investigate, develop, and implement a low-cost, wireless sensor package that is programmable and accessible to lab computers. The final product should include a wireless sensor package similar to what is currently being used in the lab along with a robust application that can connect devices to the computer and provide wrapper functions to manage data collection.

Preliminary Research

Before deciding to build our own custom sensor controller, we researched what tools were available on the market that could be used to fulfill the goals of the project.

Arduino Esplora

Dr. Daniels had showed us the Arduino Esplora that he has been using in the last few years as the main teaching instrument for the Computer Engineering 185 course. But the drawbacks are what has lead to our project. Thus the additions to the already commercially produced board was to add bluetooth capabilities, a rechargeable battery with circuit, and some case protection. After further investigations we found that Arduino was ‘retiring’ the Esplora. Thus we could not use the commercial product. So we would have to build our own version if we wanted to continue down this path, since product availability and maintainability are two major components of our requirements.

Texas Instruments Keychain Sensor Package

Loosely based on the idea of the wiimote, the sensor package was a simple polling system that returned different values from the various sensors. With a blank slate a chip and a printed circuit board would have to be made and programed. Problems start when you ask about the usability of this package. With only a polling type of system you make product very narrow scoped for only a few exciting labs. With the basic coding only being input parsing we thought this project wouldn’t have a big enough scope for our needs.

Sphero Robot

This Idea came about with the release of the Ti-88 Sphero robot. With it came the idea of a robot for the 185 labs. Adding functionality to the robot to support Infrared and Sonar would make traversal of the commercial product more interesting. However even with additions we decided that there wouldn't be enough product to produce a large quantity of uses in the forms of labs and projects.

iRobot Create 2

Since making a robot seemed like an fun and noticeable product, the idea to mimic the Computer Engineering 288 courses labs with the IRobot Create package made sense. With the new and improved platform we figured the labs could start by learning the basics in 185 and transition that knowledge into 288. Building a similar system seemed easy, however with a budget reaching our maximum amount this project wasn't in a viable scope. Also on the same usability scale as the Sphero the scope didn't seem big enough to proceed.

Playstation 4 Controller

Looking at more commercially available products the Idea of repurposing a already existing controller was brought up. With the expected availability of the PS4 being around for a minimum of 5 years the ability to maintain and or replace a broken controller would be as simple as ordering another be sent to the college. With many sorts of inputs and outputs the controller seemed like a perfect match. However with software development being around the 3 month estimation this didn't seem like a possible solution. Other concerns were brought up about controller updates and bluetooth compatibility. With these things in mind the Project would constantly be ongoing.

Smart Pi Controller

The Idea of having a reprogrammable system that is still wireless and hands, brought the Raspberry Pi to the front table. With only lacking an analog to digital converter the Pi offers us a wide range of possibilities. Ideas have been sketched out for a printed circuit board addition that would change a Pi into a wireless controller that satisfies all the clients needs. Connecting via the 40 exposed pins on the Pi 2 model b the board would have room for joysticks, buttons, accelerometers, and the ADC needed to run all those inputs. Designed around the shape of the Arduino Esplora we feel that we can make all the needed circuits for charging and communication so that we have an all in one system. Current worries are the Pi's power draw making our battery time short.

Re-programmable Arduino Due Controller

Using a Open source controller with a large quantity of input and output pins we have the Smart Pi Controller except in a easy to use environment that includes analog to digital converters and the inverse

Digital to analog converters too. Still with a PCB designed around the base Arduino Esplora we end up with a smart controller on a Arduino environment for reprogrammability. Flaws are the needed exact 3.3 volt current the board uses over the time the controller is on. Serious damage can occur with improper voltage.

Programmable Intel Edison Controller

Running on the same Idea as it's two predecessors the Intel Edison allows us this great interface to work with and also the ability to make the controller smart, programmable, and wireless. However the Edison also packs a onboard bluetooth and wifi which has all been compacted into the size of a quarter. This with the fact that it runs on two 500 Mhz cores means that we have all the processing power of a computer onboard our controller but we still don't have to give up the space to have a true computer. Running Linux the Edison can give students the ease of an OS to backup their system and make it familiar like the programming on their personal computers. These features make it the perfect match for our project and the current Idea to be implemented.

Economics

We have a soft budget limit around 250 dollars per unit according to the project description from Dr. Daniels. That would be sufficient for the current options on the table:

Current ideas

- Intel Edison Compute Module. Can be purchased from Amazon for $53
- Intel Edison Breakout Board: Can be purchased from Amazon for $75
- Battery Packs: ~$50 To be determined
- PCB: ~$100 To be determined

Past ideas

- Sphero BB-8. 149.99 each on online retailers.
- Arduino Esplora. Currently not available on official site, but on some certified online stores, it costs 65 dollars.
- Raspberry Pi 2 model b. It can be purchased on Amazon for 41.58 dollars.
- Ps4 controller which is also available on Amazon, 49.95 for a controller with cable
- iRobot Create II is fairly expensive, 200 dollars for each one on its official site, moreover the maintenance cost would be higher than other options.

Requirements

Hardware:

- Must use Intel Edison as microcontroller board
- Must be able to communicate wirelessly
- Must survive on battery power
- Reprogrammable
- Multiple sensors:
    - Accelerometer
    - Joystick
    - Buttons
    - Audio input/output

Software:

Functional

1. Controller can be turned off.
2. Controller can be reprogrammed.
3. If the Controller has not been reprogrammed the Default Controller software should be loaded.
4. Computer and Controller will communicate via Bluetooth.
5. Controller, upon being powered, should begin looking for its connection.
6. The client side API should allow the user to:
    a. Establish a connection.
    b. Close a connection.
    c. Read from a sensor.
    d. Read from multiple sensors at once.
    e. Read audio data.
    f. Send audio data.
    g. Release used resources.
7. The controller side API should allow the user to:
    a. Establish a connection.
    b. Close a connection.
    c. Read data.
    d. Write data.
    e. Send data.
    f. Release used resources.
    g. Get sensor data.

Non-Functional

1. Power coming in NEEDS to be 3.3V

2. Default controller must be…
    a. Stored into nonvolatile memory
    b. Loaded at power on
    c. Loaded at reset
3. Controller can be powered via cable or batteries.
4. While plugged in the Controller should run on cable power.
5. While plugged in batteries should charge.
6. Batteries should last for 1 hour and 30 minutes before needing charged.
7. Batteries should take 20 minutes to fully charge.
8. The response time between sending a command from the client and receiving feedback should be no more than 50 milliseconds.
9. If the Controller does not respond to a command within 100 milliseconds:
    a. Resources should be released.
    b. Appropriate error shown.
10. If user software stops, expectedly and unexpectedly, our communications software should:
    a. Release used resources.
    b. Close.
11. User should be warned when they make calls to an unconfigured connection
12. Controller uptime should be in line with battery functionality
13. Software uptime should be in line with programs using it
14. All software will have proper documentation.

Risks

Risks	**Probability**	**Criticality**	**Risk Factor (P * C)**	**Mitigation Strategy**
Damage caused by overheating	0.15	80	12	Cooling system added to device, or space out components more.
Battery can not supply needed power to Intel Edison	0.10	100	10	Plug device in permanently. This makes the system wired, but these steps are necessary to stop harm to the device.
Damage caused by external impact	0.05	80	4	Tutorial should be included. Acrylic protector should be used.
PCB production takes longer than	0.25	20	4	Push delivery date back and focus on other aspects that need to be

planned				accomplished.
Modules can't be acquired when needed	.05	40	2	Look for alternative modules
Bluetooth connectivity issues with lots of devices in the same room	.01	100	1	Switch to wifi as our platform for communication.

GQM

Goal	**Question**	**Metrics**
Durable Battery	How long does the battery last per charge?	Time system can run before needing recharged.
	How long does the battery take to recharge?	Time needed to charge battery from expended to fully charged.
	How many recharges can the battery do?	Maximum number of recharges before battery stops accepting charge.
Hardware	What kind of sensors need to be included?	Temperature Light Acceleration Stress IR
Software	Which function should be included?	Sensor data output display PC compatibility PC programmable
	Wireless communication does not interrupt use of software on client side?	Impact on run times of client programs using communications with controller.

Wireless	How to make the device wireless	use integrated bluetooth module which can communicate with connected users

Project Design

Stories

As a user, I want:

1. To connect to the controller from my computer.
    a. Must use bluetooth
    b. Simple commands
2. To be informed when the controller disconnects.
3. Errors to be informative and protect me from making illegal calls.
4. Documentation to clearly define the necessary steps, without me needing prior experience.
5. Controllers uptime to be inline with the controller being powered.
6. Computer API's uptime to be inline with my own code.
7. To be able to use the controller while it is charging.
8. To request data from a sensor.
9. To request data from multiple sensors.
10. To get audio data picked up by the controller.
11. To send audio data to be played by the controller.

As an instructor, I want:

1. The ability to turn off the device to save battery life.
2. Setup to take less than 15 minutes per device.
3. Batteries to survive 2 labs without needing to charge
    a. 2 labs is approximately 4 hours
4. To power and connect the device via USB-B connection.
5. To create labs focused on using the core libraries to teach students.
6. To create lab tools built around the core libraries.

As an intro programmer, I want:

1. Easy to understand libraries
2. Simple to set up systems
3. Descriptions that are down to my level

As a programmer, I want:

1. To ensure system resources are freed upon termination.

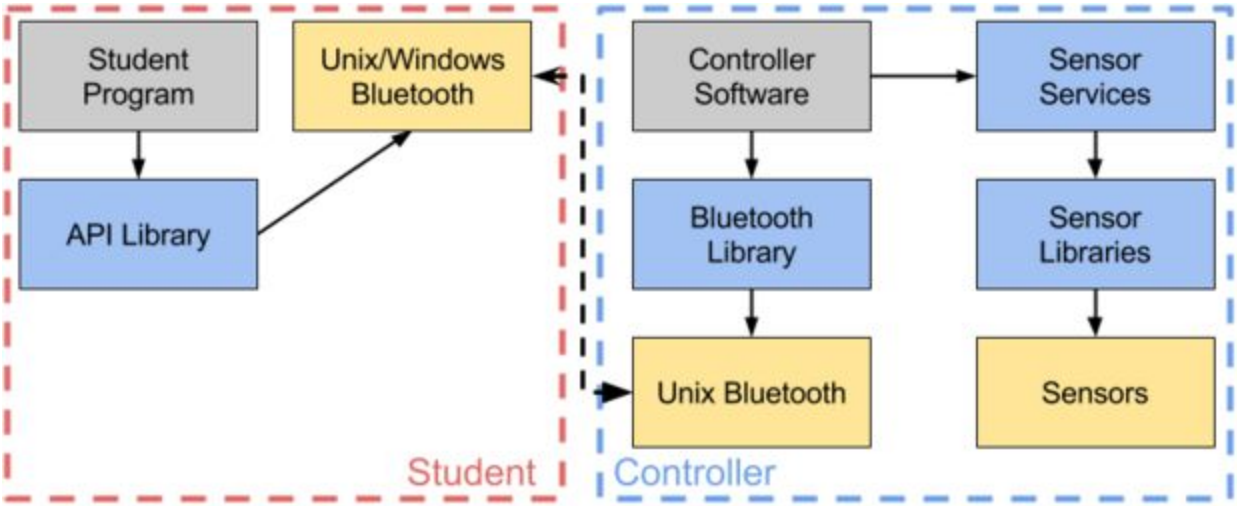
As an advanced programmer, I want:

1. To build my own bluetooth communication interface on the controller.
    a. Utilizes sensor libraries
2. To build my own sensor readers on the controller.
    a. Utilizes bluetooth libraries.
3. To write my own controller program using onboard libraries.
    a. Utilizes bluetooth and sensor libraries
    b. May write their own format for reading, or use a prebuilt format.
4. To write and run controller software from scratch.
    a. Default software does not run.

Block Diagrams (Software)

Software Architecture

Figure 1: Software Block Diagram



System Analysis

Battery

The battery circuit will consist of either LiPo or Li-Ion batteries that can supply 5V of power at 800mA. The battery circuit will operate in 3 settings including power mode (powering the Arduino and PCB), charging mode, and turned off. The way that the battery will be charged is with a barrel power connector that is plugged into a standard outlet. The power circuit could contain one large battery, or several smaller batteries connected in parallel (for charging purposes).

Printed Circuit Board

The PCB will be designed to attach to the top of the Intel Edison board. The PCB will interact with the Due through the I/O pins on both boards. The PCB will consist of a variety of sensors, buttons, and a joystick. The sensors will be designed to poll data at a designated frequency and send the information over serial bus to the Intel Edison.

Intel Edison

The Intel Edison will be the board that we will use with the microprocessor and power control. The board will be powered with the battery and will regulate the power for the PCB. The board has built in bluetooth and wifi and will communicate with the lab PC. The Edison will be connected by a serial bus to the PCB for data transfer and communication.

Lab Computers

The lab computers will interface with the PCB sensors through a bluetooth connection. The Bluetooth connection used will act as a serial port and transmit and receive bytes of information that it will interpret. The lab computers will also contain the software that we will implement to parse the input bytestream.

Specifications

Input/Output

The main input for the final controller will include a power connector that will connect to the battery's charging circuit. There might also be a serial connection available to connect a lab PC directly to the controller instead of using Bluetooth (for testing purposes). The battery will be connected internally to both the Intel Edison and the PCB which will have no external I/O. The output from the device will be piped directly to the PC over Bluetooth connection.

Interfaces

Hardware

- micro-usb
- usb
- buttons
- joystick
- accelerometer
- mic
- on/off switch

Software

Client Side API

- Start Connection
- Establish Connection
- read data from controller
- close connection

Controller Side API

- Get incoming commands
- get sensor data
- report data
- Set up bluetooth

General

Hardware

Intel Edison:

- dual core processor
- Integrated Wi-Fi and Bluetooth 4.0 support
- 1 GB DDR and 4GB FLASH memory

Intel Edison Breakout board:

- micro USB connector
- Battery charger
- DC power supply jack

PCB board:

- gyroscope
- accelerometer
- buttons
- Joystick
- battery supply circuit

Implementation Details

Hardware

The major component of the device is intel edison, which allows the controller to communicate with all the sensors on the PCB, also provides the bluetooth function to communicate with the computers in labs. It also allows students to reprogram the controller using C++.

A PCB is designed to connected the edison to all the components. The PCB includes 4 layers. A ground plane is included. All the sensors are placed on the top layer. The intel edison, USB type-B connector and audio jack are place on the bottom layer in order to hide them in the acrylic case. The battery holder is attached to the back cover of the acrylic case. Students can simply removed the back cover to change the batteries. Students can also use the USB type-B connector the charge the whole device.

On the sensor side, A 9 DOF sensor contains an accelerometer, a gyroscope and a compass will be used to gathering motion data; on the control side, 4 buttons and a joystick will be the channel for students to input command. As a part of the communication, RGB leds will give the feedback to students about basic status of the controller.

An audio codec is also included in the design. There are three audio jack placed on the lower part of the controlled. They provide the basic audio function: audio in, audio out and microphone. The audio function of the device can provide more practical function.

Software

Student run libraries will be coded in C++ using preprocessor directives that allows us to target Windows and Unix systems used in labs. C++ ensures that we can efficiently code in a clean style. The multi-system targeting allows our program to be used in any classroom environment available. This multi-system approach is necessary as bluetooth communication is handled differently on each system.

Controller libraries will be programmed in C++ and compiled on a Linux based system, as this is a static environment that is unchanging.

A default program will be implemented in C++ on the controller. This program is used to facilitate basic use of the controller, interfacing with student programs. This utilizes our controller libraries.

All libraries will be compiled into C++ dlls as they will be easy for students to import and use in their projects. This also allows us to push new versions of the library easily without disrupting student programs.

Testing Process and Results

Stage 1: Component Testing

After determining the components needed within the project, a sample set of products were purchased for our first stage, hardware testing. During this phase we put the components on breakout boards and integrated them with the edison utilizing its breakout board. The Results of this testing showed

to be very beneficial. Findings about the edison using little endian form and lower than expected logic levels would have been detrimental within later forms of testing if this stage was left out. Over ambitious time scheduling could not be follow as our testing within this phase took much longer than expected.

Stage 2: Software Testing

Beginning with the start of software production Black box and White box testing were implemented. This ensures that the Software was completing the needed tasks with the expected outcomes. Following the production of the first PCB we had planned to do Integration testing. These plans were dropped when the PCB was unsuccessful. Due to the time delays caused by Stage 1 we do not currently have very many results from the second iteration of Integration testing done on the second PCB since we have just gotten it back.

Stage 3: User Testing

After both of the other stages, We had planned to finish up all the testing with some real human testing. This round of testing was to determine test whether our product was simple to use and intuitive. Given to a user and asked to complete a task we could determine how a sample user base would use our system. Using the knowledge collected we could refine documentation or change the system depending on the issues. However due to the time delays from Stage 1 we had no time to complete this Stage of testing.

Appendix I: Operation Manual

Turning on the Device

Figure 2: Powering Device

Before the device is turned on, make sure that the battery pack leads are plugged into the back of the board correctly. Failure to connect the battery pack or not correcting the leads properly could lead to damage of the board. Once the batteries are plugged in, simply flip the switch as shown above and wait for the green LED next to the switch to light up.

API Commands

Connect

Connect is used to establish a connection to a controller.

Format:

char Connect(char* bluetoothAddress)

Arguments Required:

bluetoothAddress - Set of characters that represents a bluetooth address. This is in the format of “XX:XX:XX:XX:XX”

Returns:

-1 if the input was incorrect, or an error occurred processing the input
0 if no connection could be established
1 if the connection was established

Disconnect

Disconnects from the current bluetooth connection.

Format:

char Disconnect()

Arguments Required:

Returns:

-1 if disconnect failed, or an error occurred

1 if the disconnect succeeded, or no connection was present upon calling

ReadSensor

Reads data from a specified sensor.

Format:

double ReadSensor(int sensorID)

Arguments Required:

sensorID- The ID of the sensor we want to read from. (Refer to Appendix V for list of sensor IDs.)

Returns:

Data from the sensor, typed as a double.

ReadSensors

Read data from a set of specified sensors.

Format:

char ReadSensors(double[] buff, SensorSet sensorSet)

Arguments Required:

buff - buffer of allocated memory that data can be stored in **(?)**

sensorSet - A struct that contains an array of ids along with a size for the array. (Refer to Appendix V for list of prepackaged sensor sets.)

Returns:

0 if read failed

1 if read succeeded

Buffer buff filled with requested data from sensors, in-order of their listing in sensorSet.

ConfigureLED

Sends commands to configure the RGB of the LED.

Format:

char ConfigureLED(char R, char G, char B)

Arguments Required:

R - Toggles the red light off on 0, on for any other number

G - Toggles the green light off on 0, on for any other number

B - Toggles the blue light off on 0, on for any other number

Returns:

0 if configure failed

1 if configure succeeded

ReceiveAudioData

Connect is used to establish a connection to a controller.

Format:

char ReceiveAudioData(char[] data, int size)

Arguments Required:

data - buffer of allocated memory that data can be stored in

size - number of bytes of data that need to be read.

Returns:

0 if read failed

1 if read succeeded

Buffer data filled with requested audio data.

SendAudioData

Connect is used to establish a connection to a controller.

Format:

char SendAudioData(char[] data, int size)

Arguments Required:

data - audio data to be played by the controller.

Size - number of bytes of data to be sent.

Returns:

0 if send failed

1 if send succeeded

Compiling a Program on the Edison

1. Open up your FTP program
2. Enter in this data:
    a. Host: sftp://192.168.2.15
    b. User: Student
    c. Password:
    d. Port: 22
3. Transfer your files over into the default directory (Look up how in your FTP program)
4. Open up a ssh tool, such as Putty
5. Enter this data:
    a. Address: 192.168.2.15
    b. Port: 22
    c. Connection type: SSH
6. When prompted enter in the login information:
    a. Username: Student
    b. Password:

7. Now compile your code
    a. g++ -o <Program Name> <Source Files> -l “Controller.h”
    b. gcc -o <Program Name> <Source Files> -l “Controller.h”

Turning the Controller Off

Figure 3: Turning The Controller Off

Turning the device off is as simple as flipping the power switch back to its original state. The reset button can also be pushed to turn off the device and power it back on. If neither of these steps are working properly, the controller can also be turned off by unplugging the battery pack on the back of the board.

Appendix II: Previous Prototype Designs

Prototype 1.0

Design

Figure 4: Prototype 1 Pin Layout

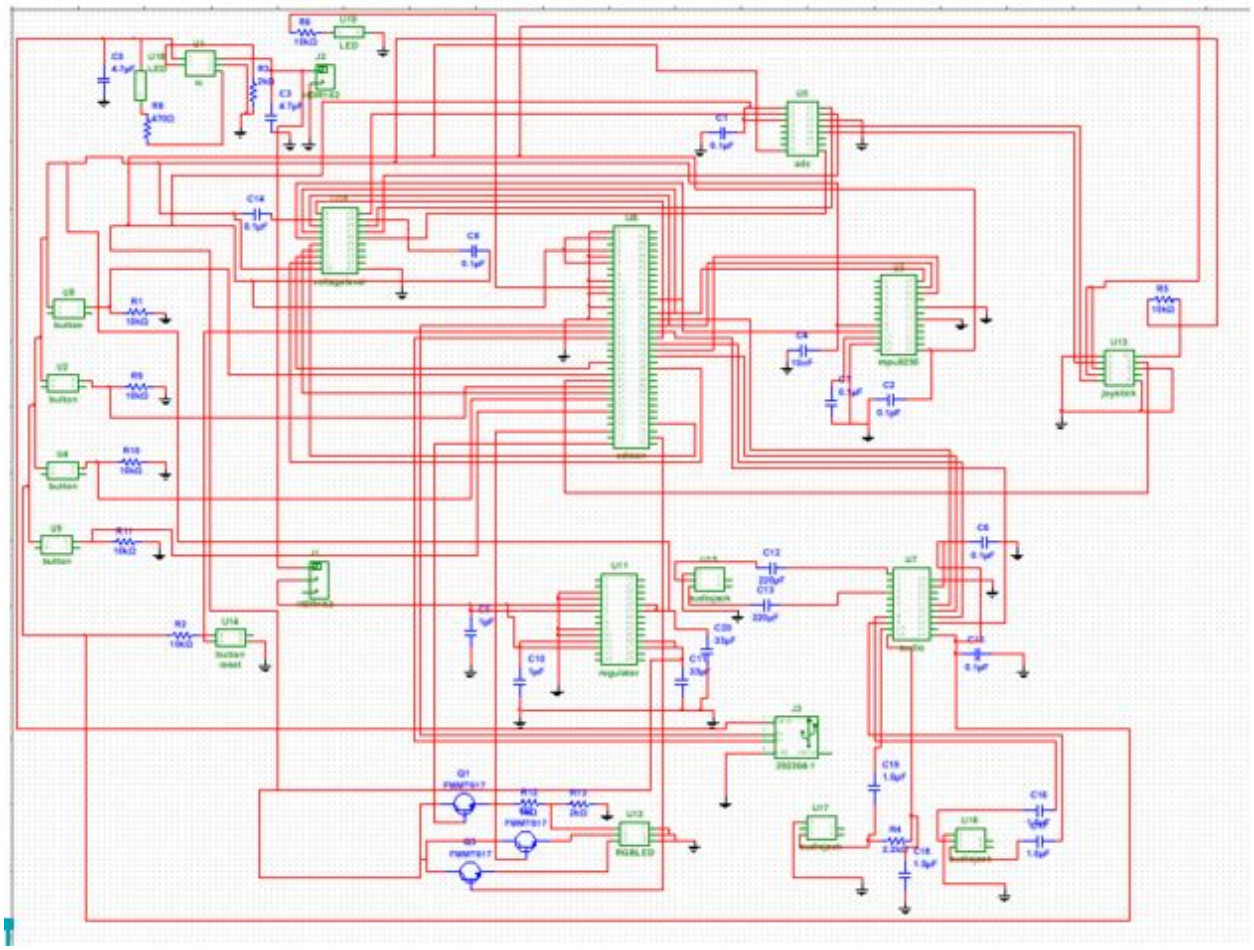
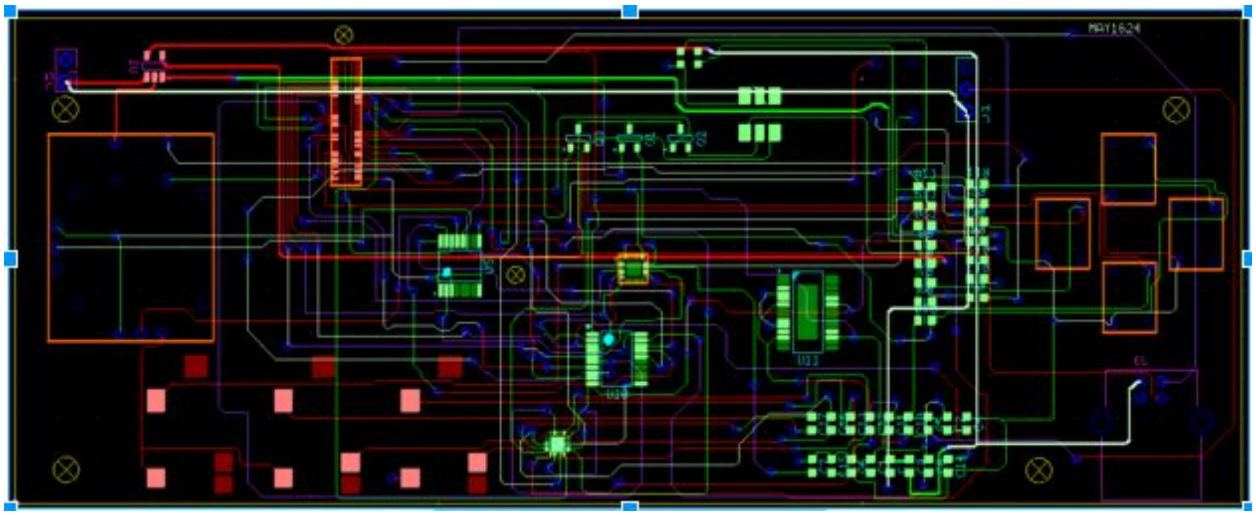


Figure 5: Prototype 1 CAD Design



Changes for Prototype 2

- Silkscreen layer needed to be moved to the top of the board and changed to ink material instead of copper
- Move USB-B and Mic ports to center of board
- Add header pin out locations
- Moved the Edison so that the antenna would be closer to the edge of the board
- Changed Edison 70-pin connection from male to female

Appendix III: Other Considerations

Challenges

Component Testing

When we first conceptualized the project, our timeline was constructed for a pre-designed solution. After researching the market, it became clear to us that a usable solution was not available. At that time, we decided that a custom designed printed circuit board (PCB) would be our best option. The timeline and scope of the project was shifted to more accurately reflect the difficulty of the project. Unfortunately since we did not have PCB design experience, our timeline did not do a good job of estimating the time cost of testing and placing each component on the board. This challenge required us to modify the overall scope of the project to put more emphasis on the PCB design.

SPI Communication

Most of the components on the PCB communicate with the Edison over the Serial Peripheral Interface Bus (SPI). We tested the sensors by connecting them to power and using the oscilloscope to get readings from the device. Unfortunately due to some complications with the software, we ended up sending incorrect messages to the sensors and ended up bricking them. We figured out the error was that we were planning for a big Endian system and the sensors turned out to be little Endian. Once we fixed the code, we were able to successfully communicate with the sensors.

Edison I/O Voltage

The Intel Edison is meant for an ultra-low power consumption project. When we started using the GPIO pins on the Edison, we weren't getting the correct data that we were expecting. We found out that the GPIO pins of the Edison run at 1.8V while most of the sensors required at least 3.3V. We fixed this issue by adding a logic leveler to the system which steps up the output voltage of the Edison from 1.8V to 3.3V.

Case

The case was a big concern for our project because we knew we needed some protection for the PCB. We did a ton of market research to see if there was a viable option, however we could not find a case in the correct dimensions that also fit our price range. In the end, we decided that we would create our own acrylic case to protect the device. The battery pack can then also be screwed into the case instead of onto the board.

What We Learned

Communication

With a fabrication project, communication is essential between the different layers of work. This project required seamless integration between hardware and software to be completed. Our group found that the best way to communicate was through weekly meetings, detailed meeting notes, text messaging, and emails. We met once a week with our advisor to discuss the project, and we also met once more per week to work on project components. During the second semester, we decided to have a 3 hour “lab” meeting where everyone would come in and work on the project at the same time. This allowed for our group to address issues quickly and efficiently. We also had a good stream of communication between the team and our advisor who was extremely helpful in clearing roadblocks for the team.

Technical Documentation

One important aspect of the project was learning how to effectively read technical documentation. Since we used multiple parts on the PCB, it was important to understand the voltage needed to power the component and the communication bus that the component could talk to the Edison was. Our strategy was to print out all of the datasheets for each component and go through each one highlighting important information such as input voltage, communication bus, output voltage, update frequency, etc. Whenever a question came up about a component, the technical documentation was available for a quick consultation.

Appendix IV: Code

All of the code for the project can be accessed through the following link:
https://git.ece.iastate.edu/jdkruege/May1624-Wireless-Controller.git

Appendix V: Sensor ID Chart

Controller Sensor IDs	
ID	Service Name
Core (0)	
0	Main
9 Degree of Freedom (100)	
101	Gyroscope X
102	Gyroscope Y
103	Gyroscope Z
111	Accelerometer X
112	Accelerometer Y
113	Accelerometer Z
121	Magnetometer X
122	Magnetometer Y
123	Magnetometer Z
ADC (200)	
No default ADC services, user must provide own	
Joystick	
201	Joystick X
202	Joystick Y
203	Joystick Button
Buttons (300)	
301	Top
302	Left

303	Right
304	Bottom
Audio (400)	
401	Headphone out
402	Mic In
403	Line in
LED (500)	
501	RGB LED

Sensor Sets			
Gyro Set			
101	102	103	
Accelerometer Set			
111	112	113	
Magnetometer Set			
121	122	123	
9DoF Set			
Gyro Set	Accelerometer Set	Magnetometer Set	
Joystick Axis Set			
201		202	
Button Set			
301	302	303	304

Appendix VI: Acronyms

Acronym	**Definition**
API	Application Program Interface
CprE	Computer Engineering
Edison	Intel Edison
FTP	File Transfer Protocol
GPIO	General Purpose Input Output
Hz	Hertz
LED	Light Emitting Diode
LiPo	Lithium polymer
Li-Ion	Lithium Ion
PCB	Printed Circuit Board
SPI	Serial Peripheral Interface Bus
SSH	Secure Shell
V	Voltage
9DoF	Nine Degree of Freedom