MAY1735

ISU Department of ECprE

Dr. Daniels

Kyle Fischer - Team Lead

Michael Linthicum - Communications Lead

Daniel Shauger - Concept Holder Lead

Sam Neff - Webmaster

Nick Juelsgaard - Schedule and Planning Lead

may1735@iastate.edu

http://may1735.sd.ece.iastate.edu/

Revised: 12/6/16

CyMote for CprE 185

Design Document

Contents

[1 Introduction 2](#_Toc468780911)

[1.1 Project statement 2](#_Toc468780912)

[1.2 Purpose 2](#_Toc468780913)

[1.3 Goals 2](#_Toc468780914)

[2 Deliverables 3](#_Toc468780915)

[3 Design 3](#_Toc468780916)

[3.1 System specifications 3](#_Toc468780917)

[3.1.1 Non-functional 3](#_Toc468780918)

[3.1.2 Functional 3](#_Toc468780919)

[3.2 Proposed Design/Method 4](#_Toc468780920)

[3.2.1 Hardware Design 4](#_Toc468780921)

[3.2.2 Software Design 7](#_Toc468780922)

[3.3 Design Analysis 9](#_Toc468780923)

[4 Testing/Development 9](#_Toc468780924)

[4.1 Interface specifications 9](#_Toc468780925)

[4.2 Process 10](#_Toc468780926)

[5 Conclusions 10](#_Toc468780927)

[6 References 10](#_Toc468780928)

# 1 Introduction

## 1.1 Project statement

CprE 185 currently utilizes a device called the Arduino Esplora. The Esplora is a handheld controller with a simple set of I/O on board, including some buttons, a joystick, Bluetooth, and, most importantly, a three dimensional accelerometer. The students in CprE 185 collect real-world data from the Esplora to solve interesting programming and physics-based challenges. Unfortunately, the Esplora has been discontinued.

Our Senior Design group will design and build a beta model for a new device for CprE 185 specifically designed for this class. This new device, called the CyMote, will provide all the functionality previously provided while costing less. The design will be on a PCB with student generated control code.

## 1.2 Purpose

The department has specifically designed CprE 185 to challenge freshmen students to think like engineers. It is important for students to associate the work they do in class to solving problems in the real world. The Esplora was that connection. It allowed both user input in the form of the joystick and buttons, but it also allows for collection of the effect of the laws of physics through the accelerometer.

The device that we develop will allow students a meaningful pathway into solving real-world problems. This device will reliably provide the desired I/O, withstand constant handling, and be easy enough for a freshman student to figure out. The three main tenants of the project design are:

1. Reliability
2. robustness
3. Simplicity for End User

## 1.3 Goals

The goal of the project is to create a beta model of a demonstrably robust CyMote device that will be capable of all required functions. Our hardware goal is to become proficient enough with MultiSIM and Ultiboard to create a usable, defect free, PCB. This controller must be easy for a TA or graduate student to purchase, assemble, and test. The device must have an easy to use programming interface and on-board I/O to allow testing.

Our software goal is to create easily adjustable code for the MCU. We will be providing a copy of the source code with enough internal commenting that a junior/senior level computer engineering student will be able to follow the code logic. There will also be a PC wrapper program that can be specifically tied to individual CyMotes to allow multiple CyMotes to work together in the same room. The PC wrapper program will take the data from the CyMote over BLE and present it to the CprE 185 student in a way that they can use C code to manipulate.

# 2 Deliverables

There are two main deliverable items for this project: (1) A physical beta model of the CyMote for display and presentation, (2) a PC wrapper that will provide a student with information from the CyMote for use in class, and (3) a full set of documentation that will allow future development.

# 3 Design

The CyMote will emulate the Arduino Esplora in all facets that pertain to the CprE 185 class. The CyMote will be loaded into a soft casing and be dropped from the third floor of the Coover Atrium. This is the main use case. The CyMote will be self-powered and will send info back to a computer over BLE. It will send enough data fast enough for the students to get see visibly how the acceleration of the device is affected by the drop and the landing.

##  3.1 System specifications

The beta model of the CyMote will be laid out on a PCB. It will use the ATSAMB11 to control various peripherals. The peripherals used are:

* Joystick w/ button
* 4 buttons
* Power Switch
* 9° of Freedom sensor
* 3.7V Battery
* RGB LED

The software component will be written in C using Atmel Studio and ASF project structure. There will be a wrapper program on the CyMote’s corresponding PC. This program will be a Windows executable. The wrapper will handle communication with the CyMote and will present the desired information to the student in the command line.

### 3.1.1 Non-functional

1. Look like a game controller
2. Stream data in real time – fast enough to register current movement on screen as having no lag
3. Beta model built on a handheld size PCB
4. Permanently securable to the PCs in the CprE 185 lab to ensure they don’t “walk away”

### 3.1.2 Functional

1. Droppable – survive a 30’ drop in a soft container of some kind
2. Robust enough to have people plugging/pressing/slamming/charging often
3. Communicate with a partner PC over Bluetooth wirelessly
4. Low power consumption from onboard rechargeable batteries
5. I/O components
	1. 9° of Freedom – 3D accelerometer
	2. Joystick – (x,y analog inputs, button press)
	3. 4 game controller style input buttons
	4. Status LEDs for all the devices
	5. Tricolor LED for display/status/fun
6. Power on/off switch
7. Eight hours of battery life
8. Communicates without loss or bugs over USB and BLE
9. Streams >30 accelerometer records per second
10. Permanently named (physical, BLE, code) for paring with computers

## 3.2 Proposed Design/Method

We have divided this project up into several phases. First we would like to pick devices that we know will work. Then we would like to get them all together and design an Alpha model that will allow us to determine if our pieces will work together. After that we will begin designing the PCB for the beta model. As we develop the hardware for the beta model, we will begin creating the software for the board. Once we know that we can use the board to communicate over BLE and USB to all the on board sensors, then we will begin creating the wrapper program.

### 3.2.1 Hardware Design

The hardware design is driven by several key factors:

* Battery life
* Compact physical design
* ATSAMB11 controller
* 9° of Freedom I/O device
* BLE transmission
* SPI communication
* User-friendly design, others must be able to re-create

We need to take all these concepts into account as we design our CyMote PCB.

#### 3.2.1.1 MCU

Because the ATSAMB11 fits our project so well, the circuits are significantly simplified. See the following two tables from the ATSAMB11 data sheet. \*Note that all table and graphics in this section are pulled from Atmel’s ATSAMB11-MR210CA-MR510 datasheet.



Table 1: ATSAMB11 suggested operating voltages

 The MCU can accept input voltages from 2.3 to 4.3, which is more than the full range of typical 3.7V Li-Ion batteries anyway. This dovetails with our decision to purchase a single 3400 mAh Li-Ion battery for each CyMote. We can power the MCU directly from the battery with some simple voltage detection to set up an elegant death for the controller when it is near the regulator dropout voltage. Because we specifically chose the rest of the components to work on 3.3V, we can supply a 3.3V rail with a LDO switching regulator that will operate efficiently. The next table shows the typical current draw of the MCU.



Table 2: Typical current draws from the ATSAMB11

Because the typical current from the battery to supply the MCU is so low (even under constant BLE Rx/Tx conditions) we are able to specify a battery that doesn’t have to be terribly large. In fact, if we take into account current draw from the 9° of Freedom sensor, and the leak currents through the joystick, and even if we assume that we will be turning on the LEDs nearly constantly, the total current draw we expect is less than 50mA. Theoretically we can power this device for 60 hours on a 3,400mAh battery. We will be testing the actual longevity of the battery on a single charge, and we may resize the battery smaller if we find it acceptable.



Figure 1: Basic schematic for powering the ATSAMB11

#### 3.2.1.2 9° of Freedom

The other main component that is non-trivial and required is the 9° of Freedom sensor. This sensor has nine total axes of sensing: 3-D accelerometer, 3-D magnetometer, and 3-D gyroscope. We are only concerned with the accelerometer for this senior design project. We have chosen the LSM9DS0 device for the CyMote. This chip is powered by 3.3V and uses at most 4.2mA of current to operate. It speaks Serial Peripheral Interface (SPI) communication, so it was very important that we pick a controller that was capable of communicating over the SPI bus. Atmel chips are well suited for this bus, and the ATSAMB11 can actually manage two unique SPI buses.

#### 3.2.1.3 PCB

Because we were able to find such well-fitted components, we were able to scope in PCB design. The PCB will contain all the components and provide a stable location for the joystick and buttons for user operation. There are some important design rules that we are taking into account in special regards to the ATSAMB11. Because of the sensitive and low-energy nature of the BLE communication protocol, the board must be specifically cut and designed to avoid any noise that might interfere with the antenna. To this end, we will ensure that the only very noisy device: the LDO voltage regulator, is placed sufficiently far from the BLE antenna. Atmel also suggests that the board is physically cut to help facilitate the communication.



Figure 2: PCB Cutout design

#### 3.2.1.4 Battery Charging and Power Management

To provide the necessary power for our system, we need a battery. We found that a Li-Ion battery was more than sufficient for our design. Because the ATSAMB11 can take a large range of input voltages and we only need one dedicated voltage rail at 3.3V, we have chosen a 3.7V Li-Ion battery. The battery will charge at a value of 4.2V via a yet-to-be picked charging IC.

#### 3.2.1.5 System One-Line



Figure 3: One-line Diagram of the system

This is our proposed one-line diagram of the system. You can see that the components are powered by the internal battery, which is in turn powered by either a USB or 5V DC interface. The MCU is directly powered by the battery while the rest of the components tap into a regulated 3.3V rail. The MCU talks to the PC over the internal BLE connection and also through a special Atmel programming/debugging interface called In-System Programming (ISP) that communicates over the SPI bus. The 9° of Freedom also communicates on this bus.

### 3.2.2 Software Design

The ATSAMB11 will be programmed with Atmel Studio 7, which is a free software suite for programming all Atmel controllers. This will allow the future super users of the CyMote to easily update MCU code and make bug fixes and feature adds. Because of the nature of the ATSAMB11 and the SPI protocol, this system will be fairly easy to add devices to. We foresee the necessity/desire to add devices such as ADC/DAC converters and an audio codec into the CyMote, so this will make the modularity of the device very handy for future developers.

The PC wrapper program will be a Windows executable and will interface with a BLE dongle either permanently embedded in a computer or secured to the computer in some way. The PC wrapper will handle the BLE communication from the PC to the CyMote and request data only when the user wants to collect the data. This way we avoid using energy in the CyMote when it’s not necessary. The PC wrapper will also handle simple error communication to the user to allow freshmen to troubleshoot the device if it’s not working properly.

The CyMote itself must be able to handle all on-board devices and communicate when it is working correctly and when it is malfunctioning. We will do some significant testing to prove out reliability. Below is the basic program diagram.



Figure 4: CyMote data flow

The two main external communication paths are the BLE communication from the BLE dongle on the PC to the ATSAMB11 BLE antenna and the debugger connection from the PC (through USB) to the ISP programming interface on the CyMote. Each CyMote will have a permanent name that will link it to a corresponding PC and BLE dongle.

#### 3.2.2.1 Digital and Analog I/O

The CyMote must handle the input of four game controller style buttons. These are digital inputs and the CyMote must be able to communicate these button presses to the PC as Boolean values. The joystick has two axes and a button built in. The two analog values (ranging from 0-1023) along with the digital signal of the button must also be communicated. The RGB LED is driven by PWM outputs from the device as well, and these values must be able to be set by the user.

#### 3.2.2.2 9° of Freedom

The accelerometer data is polled from the LSM9DS0. The data can be gathered as fast as the SPI can be used to request it. Because of the difficult Atmel libraries, we have decided to create our own register reading implementation. This implementation is entirely simulated through software on variable GPIO instead of in dedicated hardware registers, and is, therefore, a little slower. However, the speed of polling is still much faster than the specified transmit rate of the data over BLE.

#### 3.2.2.3 BLE

The main way we will communicate with each device is over BLE. We have verified that we can poll the MCU for data over BLE from a third party BLE device. The next steps will be to implement communication via our PC Wrapper and switch to a subscription system instead of a request system.

#### 3.2.2.4 PC Wrapper

The PC wrapper is still being designed. It will definitely include the following.

* BLE dongle interface to allow communication with the CyMote
* Windows executable
* Provides raw CyMote data to a user designed C program
* Allows for simple error checking
* Tied to a permanently named and pair CyMote

## 3.3 Design Analysis

So far we have spent a decent amount of time finding appropriate pieces, defining the scope of our project, and testing our chosen MCU.

We have chosen the Atmel ATSAMB11 as a controller. We purchased three development board that Atmel provides to start working with these controllers. We have used startup documentation provided by Atmel as well as some pre-designed programs to develop our BLE communication and SPI communication. So far we have gotten buttons, LEDs, the joystick, SPI, and BLE to fully function with their “Hello World” programs.

For the hardware, we decided that we will be putting the beta model on a PCB. We have talked to Lee about how to get this done. He had a lot of good ideas, and we’re planning on trying to get our first board cut before Christmas if possible. As for power management, we’ve done some basic research on battery power and had a good conversation with Dr. Tuttle about how to choose the correct power management ICs. We are in the process of ordering some test pieces.

# 4 Testing/Development

## 4.1 Interface specifications

 The over hardware is the board. We will be communicating with the MCU over USB and BLE. The MCU will be our entry point to the 9º of Freedom. The board will also charge its batteries over USB.

## 4.2 Process

We will be testing each hardware piece on its own first. We have already tested almost all the pieces alone. The software team can talk to the MCUs, talk to the 9° of Freedom over SPI, and can get feedback from the buttons and the joystick. They have also made the LED light up.

We are developing our first alpha board schematic right now. Once we get all the pieces installed we will be testing the robustness of the design. Once we have a robust design, we will begin design on the PCB. Once we have a PCB cut, we will try out hand at surface mount soldering. We will verify the board and make any necessary changes, and then order our final boards and try to make a solid beta model.

# 5 Conclusions

So far we have developed our scope, spent time specifying our parts, and started making progress with the hardware design. We have already made very good progress with the software. We need to make significant progress in the first part of the spring semester in order to have the project end positively. If we can continue our good progress on the design and start getting PCB cut by the first couple of weeks in January, we will have a good chance to have our hardware complete by spring break, which will in turn give us a good chance to finish our software by May.

# 6 References

Datasheet for the LSM9DS0:

[http://www.st.com/content/ccc/resource/technical/document/datasheet/ab/2a/3b/45/f0/92/41/73/DM00087365.pdf/files/DM00087365.pdf/jcr:content/translations/en.DM00087365.pdf](http://www.st.com/content/ccc/resource/technical/document/datasheet/ab/2a/3b/45/f0/92/41/73/DM00087365.pdf/files/DM00087365.pdf/jcr%3Acontent/translations/en.DM00087365.pdf)

Datasheet for the ATSAMB11:

<http://www.atmel.com/Images/Atmel-42426-SmartConnect-SAMB11-SOC_Datasheet.pdf>

Datasheet for the X-plained Pro Development Board:

<http://www.atmel.com/Images/Atmel-42664-ATSAMB11-Xplained-Pro_UserGuide.pdf>

Atmel Software Framework (ASF):

<http://asf.atmel.com/docs/latest/>

SparkFun LSM9DS0 Arduino Library:

<https://github.com/sparkfun/SparkFun_LSM9DS0_Arduino_Library/blob/master/src/SFE_LSM9DS0.cpp>

 Official description of CprE 185

<http://catalog.iastate.edu/azcourses/cpr_e/>