Team 26: Bomb Squad

University of Massachusetts Amherst

BE REVOLUTIONARY™

Team Members

Ethan LaFleur - Electrical Engineer

Krishna Vijayakumar - Computer Engineering

Edward “Matt” Buiser - Computer Engineering

Background: Keep Talking and Nobody Explodes

- **Our game will be modeled after the online video game: “Keep Talking and Nobody Explodes”**
- **This game is available for Consoles, PCs, and mobile devices**
- **There is also a virtual reality version**
- **Rules of the game:**
  - One player has a bomb in front of them
  - The other player has a manual for how to defuse the bomb
  - Player with bomb must describe modules on the bomb to player with manual
  - Player with manual instructs how to defuse

Problem Statement

Technology such as video game consoles, or virtual reality systems are not available to many people. Also many people who did not grow up with this type of technology have a difficult time traversing the digital landscape. This limits the exposure of online games. Our solution will bring an online game: Keep Talking and Nobody Explodes to the tabletop. It will eliminate the need for technology and will be more appropriate for all types of people.

System Specifications

- Uses a physical “bomb” with modules already installed
- Randomizes each individual module every time a new level starts
- Randomizes which modules are active on each run
- Includes difficulties in the form of levels that a user can set before each run
  - Automated in code for the game, affects each game differently, overall time may be lower as it increases
- Includes at least 3 regular modules and 1 “needy” module
  - Simon Says
  - Password Game
- Some modules will be adapted for our implementation
- Modular approach
  - One master module to keep track of all modules
  - Each module contains its own information and ruleset
  - Communicate Via I2C (SDA and SCL wires) to Master Module - latency within a few seconds
- Manual will be converted to an app
  - Selectable options for modules on app
  - Send Difficulty via Bluetooth
    - Wireless, distance between “diffuser” and “expert” within 15 feet

MDR Deliverables

- **Prototypes of select few modules that demonstrate critical aspects of our project**
- **A working prototype for a module defined as:**
  - Capable of receiving necessary data (random seed, difficulty level, etc.) via I2C
  - Capable of sending necessary data (Strikes) via I2C
  - Contains own ruleset
  - Indication of incorrect/correct attempts, and indication when the game is completed
  - Difficulty level changes game and solution
- **Randomness generated each time through for whole system**
- **Evaluation of power for the entire system**
  - Theoretical measurements of power consumption
- **Basic development of app capable of the following:**
  - Communication over bluetooth
  - Sending the difficulty level to the Master Module
  - Displaying the Manual for the games

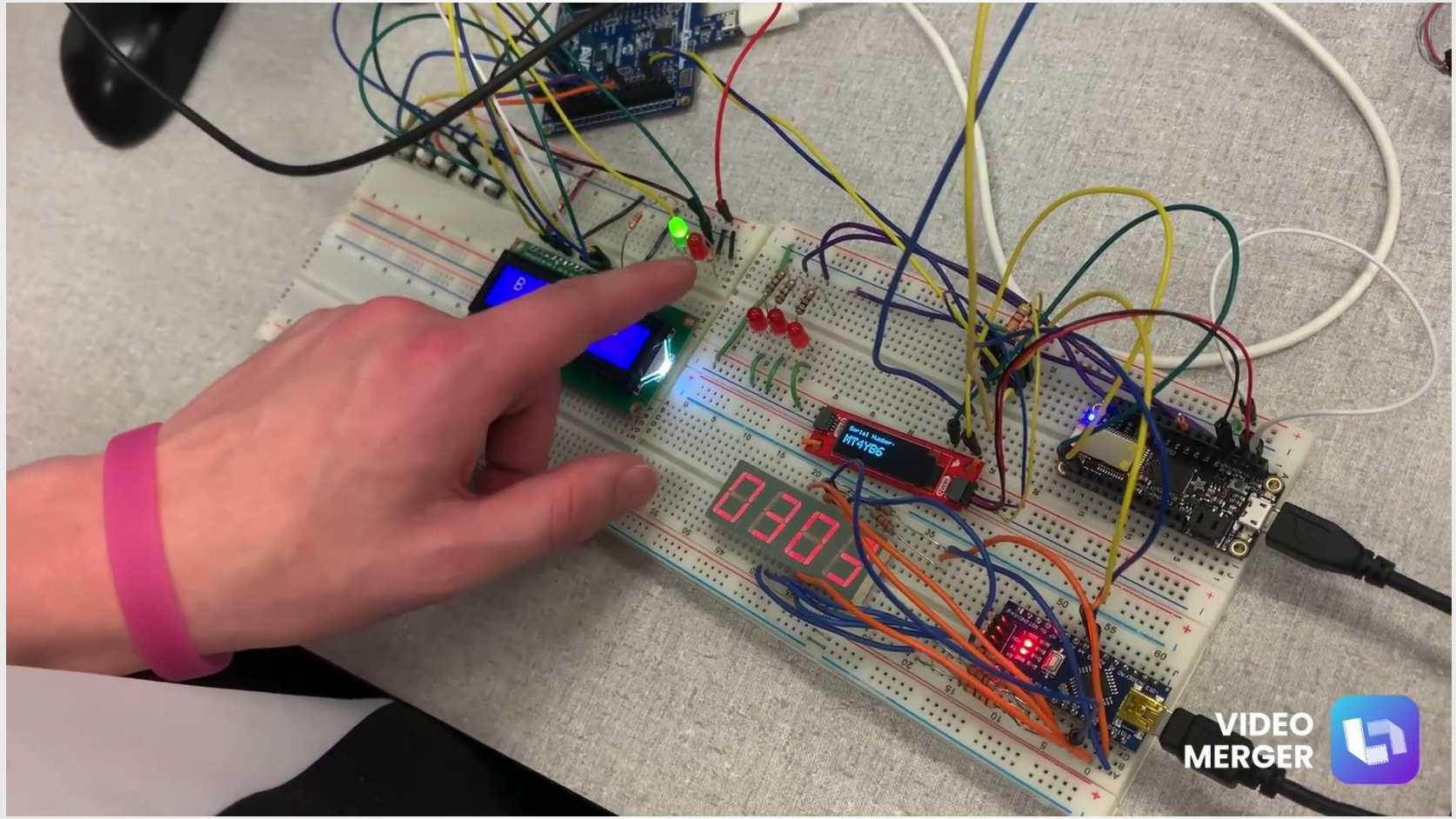
Test Plan

- **Demonstration of multiple playthroughs for overall system keeping note of the critical features being present:**
  - Randomness
  - Difficulty Incorporation
  - Microcontroller Communication (sending data over I2C)
  - Child modules containing individual ruleset
- **Report of Randomness that includes:**
  - How we generate randomness
  - Theoretical number of combinations for every aspect that incorporates randomness
  - Measurements of how random each game is

Test Plan (Cont.)

- **Report of the Power Consumption of the System:**
  - Take Measurements of Power on a per Module Level
  - Hand Calculations for Rough Estimate of Total Power Consumption
  - Evaluation of if the System could be Powered by Rechargeable Power Source (battery)
- **Demonstration of Bluetooth Application Capable of the Following:**
  - Bluetooth Connectivity via developed application
  - Sending Difficulty Level to Microcontroller
  - Pulling up Manual for Included Modules
- **Display of the User Interface of Bluetooth Application**

Demo Video



0:00 to 2:35
Demonstration of playing and completing the game successfully

2:35 to end
Demonstration of both Child Modules able to send strikes to Master Module

Randomness Report

- Randomness is a critical aspect of the project
- Arduino and Microchip studio do not have a good way to randomize
- They start with the same seed every time it is initialized
- How to change that seed?

```
rSeed = analogRead(A3)*micros();
```

- Read noise and multiply
- Serial Number consists of: AA#AA#
- Over 45 Million unique possibilities
- Random seed is sent to the CM's using I2C

Randomness Report (Cont.)

Simon Says:

Sequence Length	# of Unique Combinations
3	64
4	256
5	1024
6	4096
7	16,384

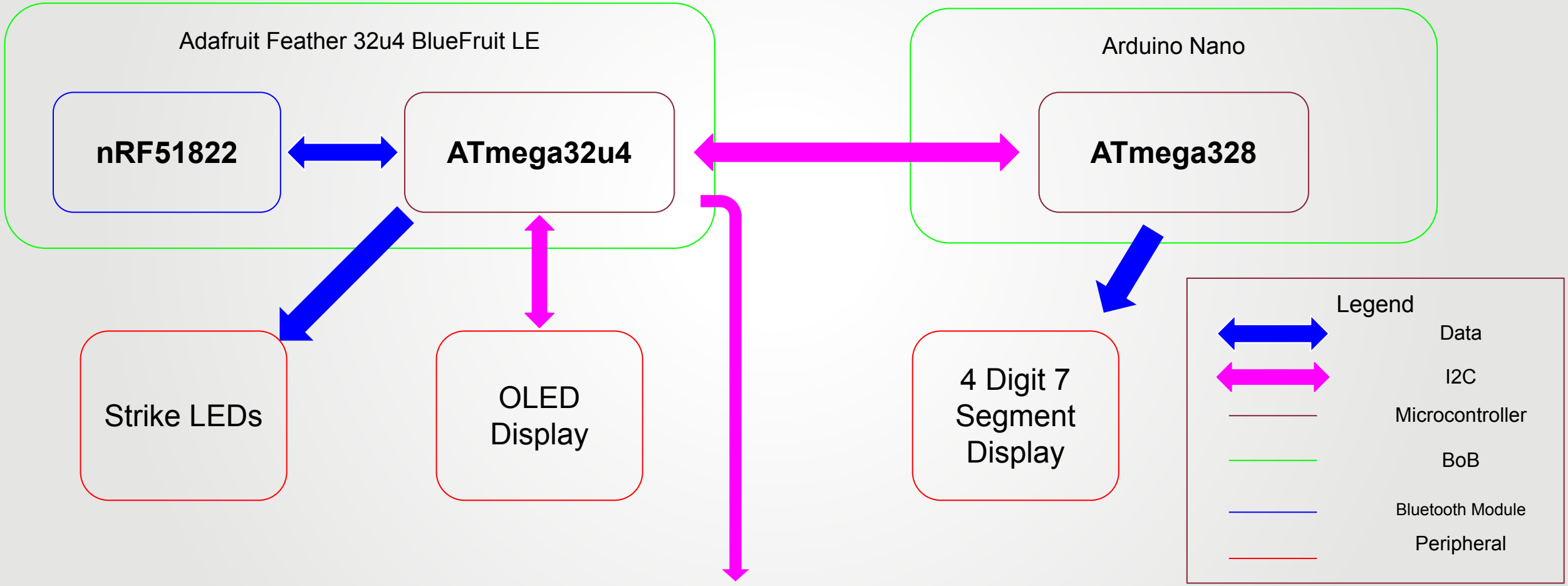
Password Game:

Extra Letters Per Slot	# of Unique Combinations
4	17,500
5	21,875
6	26,250
7	30,625
8	35,000

Randomness Report Cont.

Serial Number	Simon Says Sequence	Password
SL4LV8	GYBR	First
QU3LI9	GBYY	After
CK2NR4	GBBR	Found
EB6NE3	GYBY	Learn
EO2YQ0	GGGB	Could
AD5ZS1	GYRY	House
UV9YP5	GRRB	About
AD5ZS1	GYRY	House
MT4YB6	GYGG	Below
GL1AY0	GRRG	First

Master Module Block Diagram



Master Module Software

Uses Arduino libraries on Adafruit Feather

Setup

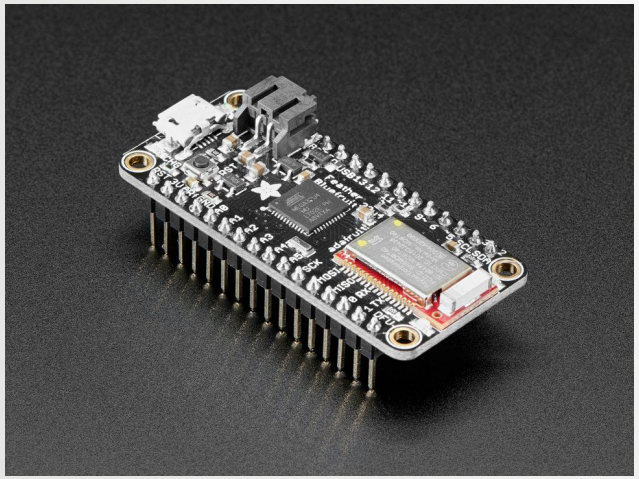
1. **Wait for App to send level data via Bluetooth - Uses BLE library**
2. **Generate random seed**
   a. Uses built in PRNG
3. **Pass necessary data to the ATmega328Ps via I2C - Uses I2C library**
   a. Waits for Acknowledgement
4. **Pass level data to Arduino Nano via I2C**
   a. Arduino Nano sets timer based on level
      i. Multiplexes 4-digit display, uses built-in time function ***millis()*** to change display every second

Loop

1. **Poll for data sent from peripherals every second**
   a. Peripherals send strike and if module is complete
   b. Use *millis()* to keep track of time
2. **If strike sent, update red LEDs, check if strikes count = 3**
   a. if true, break and game is lost ; continue if false
   b. update other peripherals on info
3. **If module is complete, update completed list; Game won when list full**
4. **Stop when timer is at 0**

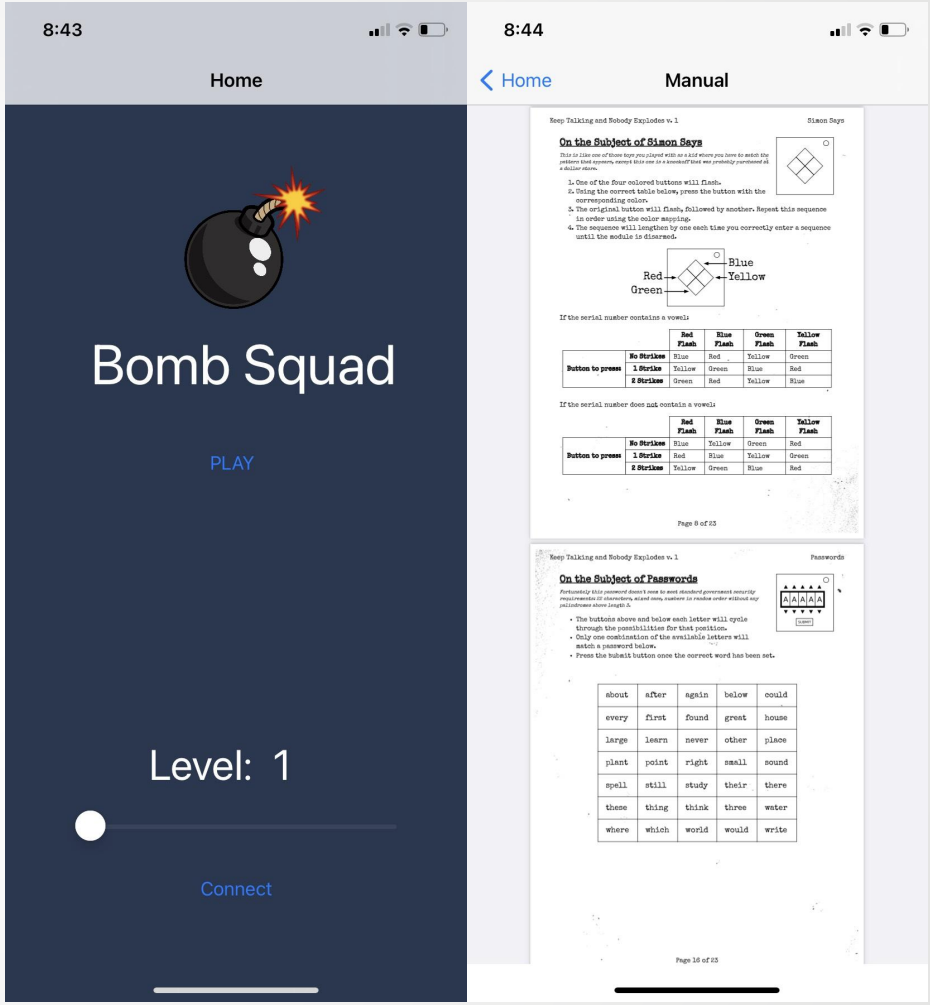
Justification of Adafruit Feather 32u4

- **Built-in Bluetooth Low Energy Module allows for Bluetooth communication, rapid prototyping, and an easy command set**
- **I2C communication**
- **Use Arduino libraries because board is only compatible with the Arduino IDE**
- **Small memory and low processor computation needed → save power with 8 bit**

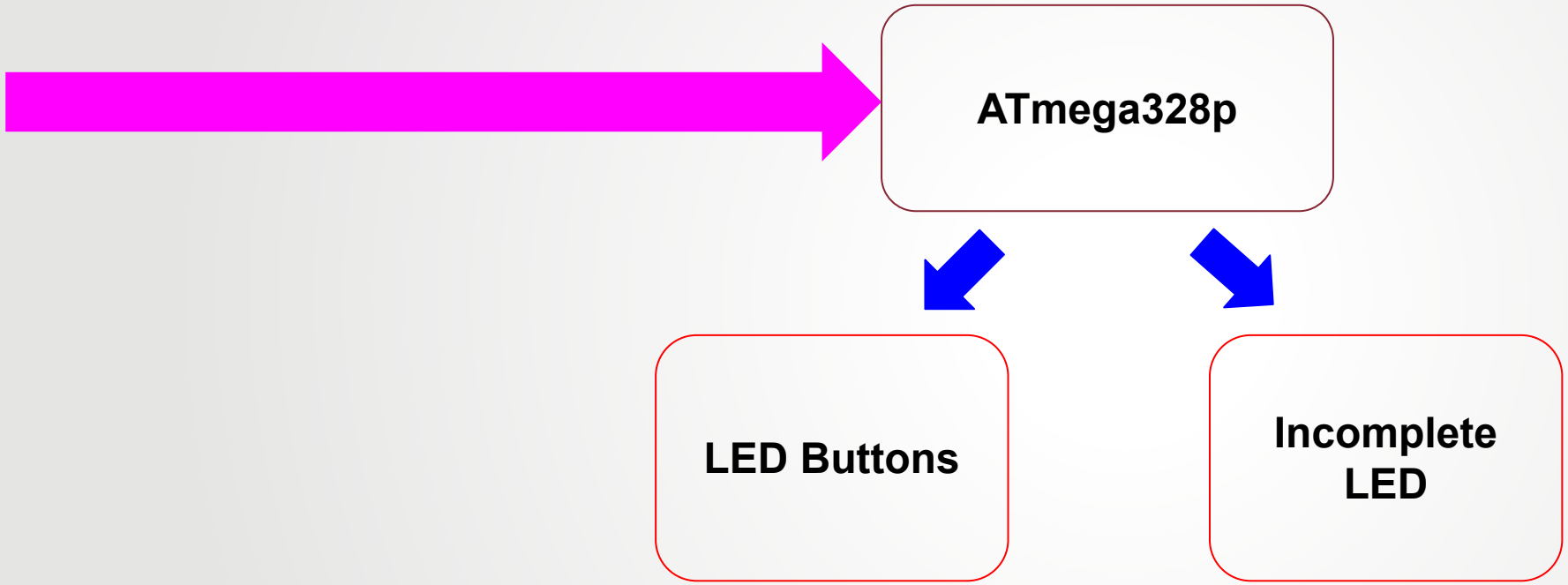


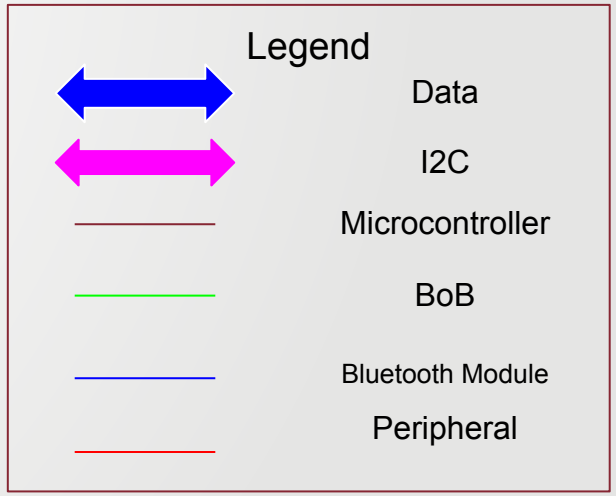
Bluetooth and the App

- XCode development tool has defined framework for Bluetooth called Core Bluetooth
- Arduino IDE has helper libraries for using communication with a Bluetooth peripheral.
- app uses basic features to control the level being sent via Bluetooth
- Message from the app uses ASCII encoding of bits in message



Simon Says Block Diagram



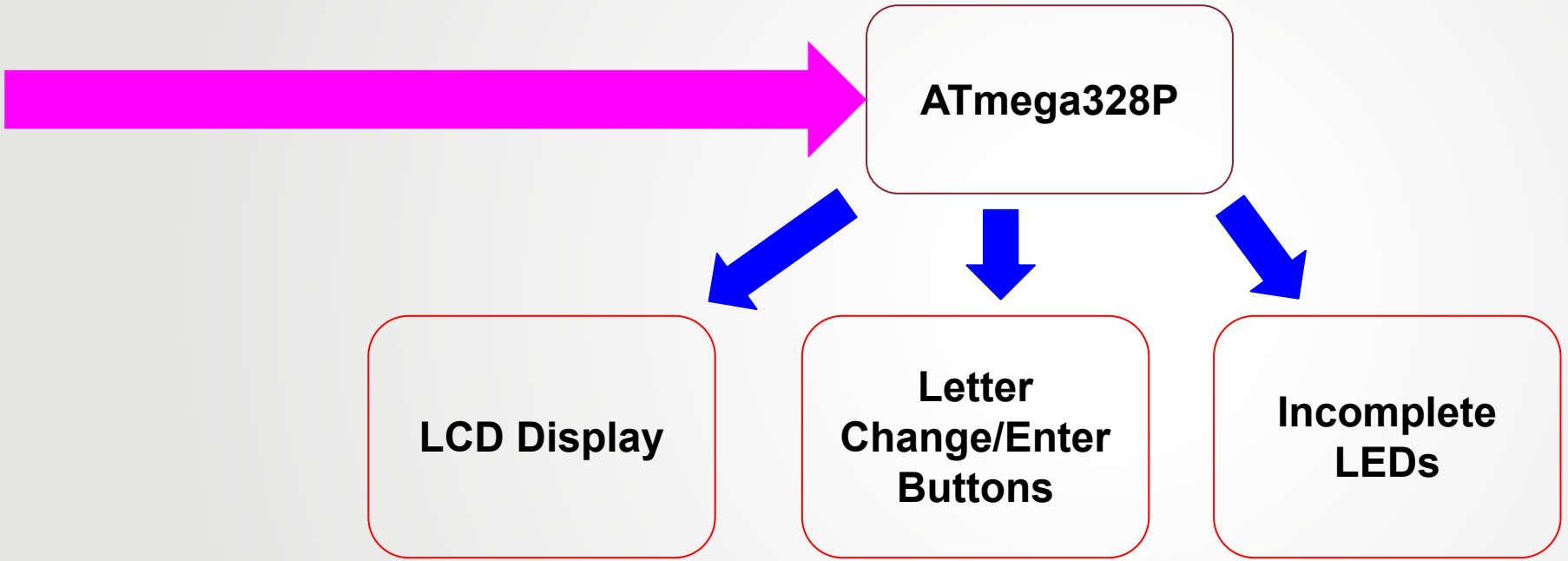


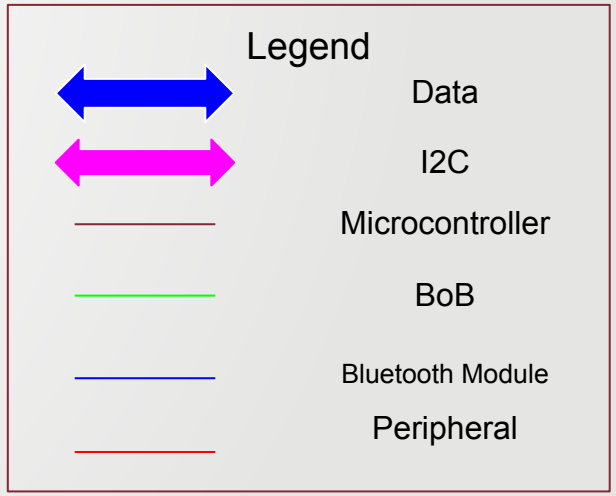
Simon Says Software

Embedded C program on ATmega328P (Microchip Studio)

1. **Receive level, random seed, # of strikes, and serial number via I2C - use TWI registers**
2. **Use level number and random number to generate sequence length**
   a. Random number generated through **random()** PRNG in avr-libc standard library
   b. Save flashed sequence in array
3. **Generate correct sequence array based on # of strikes and properties of serial number**
4. **Wait until user inputs sequence length number of button presses**
   a. Save user response in array
5. **Compare arrays**
   a. **if correct sequence == input sequence, flash green**
      i. if round != last, add 1 more random flash to flashed sequence and go to step 3
      ii. if last round, stay green and break
   b. **if correct sequence != input sequence, flash red**
      i. Add a strike, signal to master module with I2C as ST
      ii. Loop to step 3

Password Game Block Diagram





Password Game Software

Embedded C program on ATmega328P

Setup:

1. **Initialize and clear LCD Display**
2. **Get random seed and difficulty from master**
3. **Use avr-libc PRNG to choose 1 of 35 possible 5 letter words**
4. **Difficulty determines the number of possible letters game cycles through for each character**
5. **Initially display one of random letters in the cycle for each character**
    a. Created LCD_Print function abstracts away bit-level operations when displaying characters

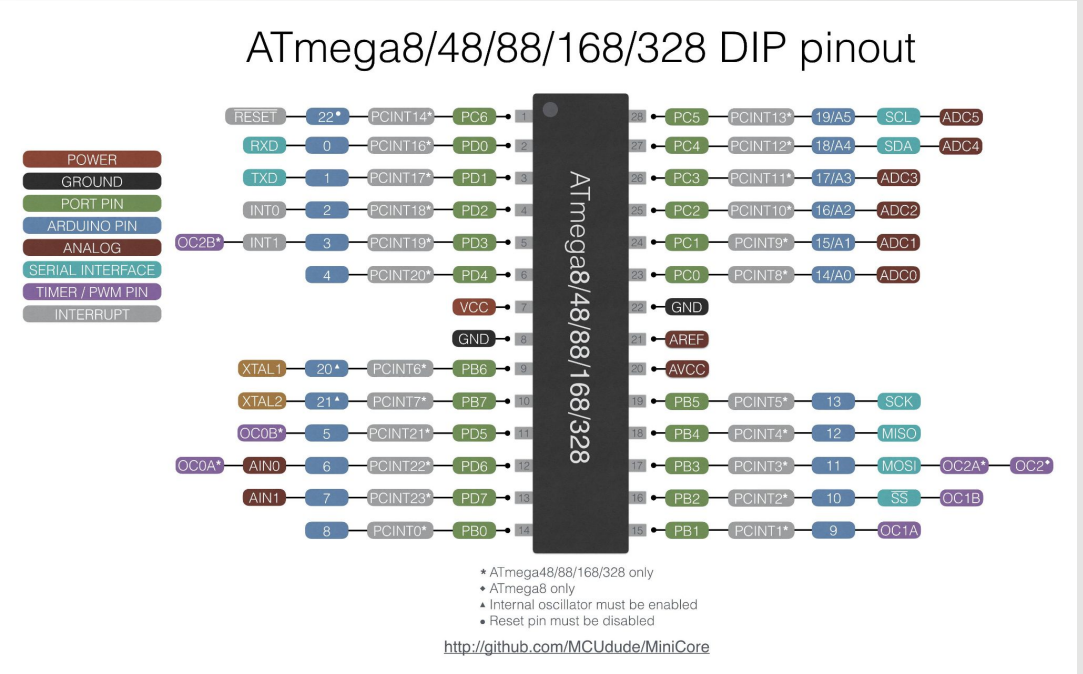
Loop:

6. **5 buttons for each character**
    a. A button press changes the displayed letter for the corresponding character position
7. **Submit button**
    a. If displayed string == original word, game won and turn green light on
    b. if displayed string != original word, add strike, send signal to master via I2C, flash red

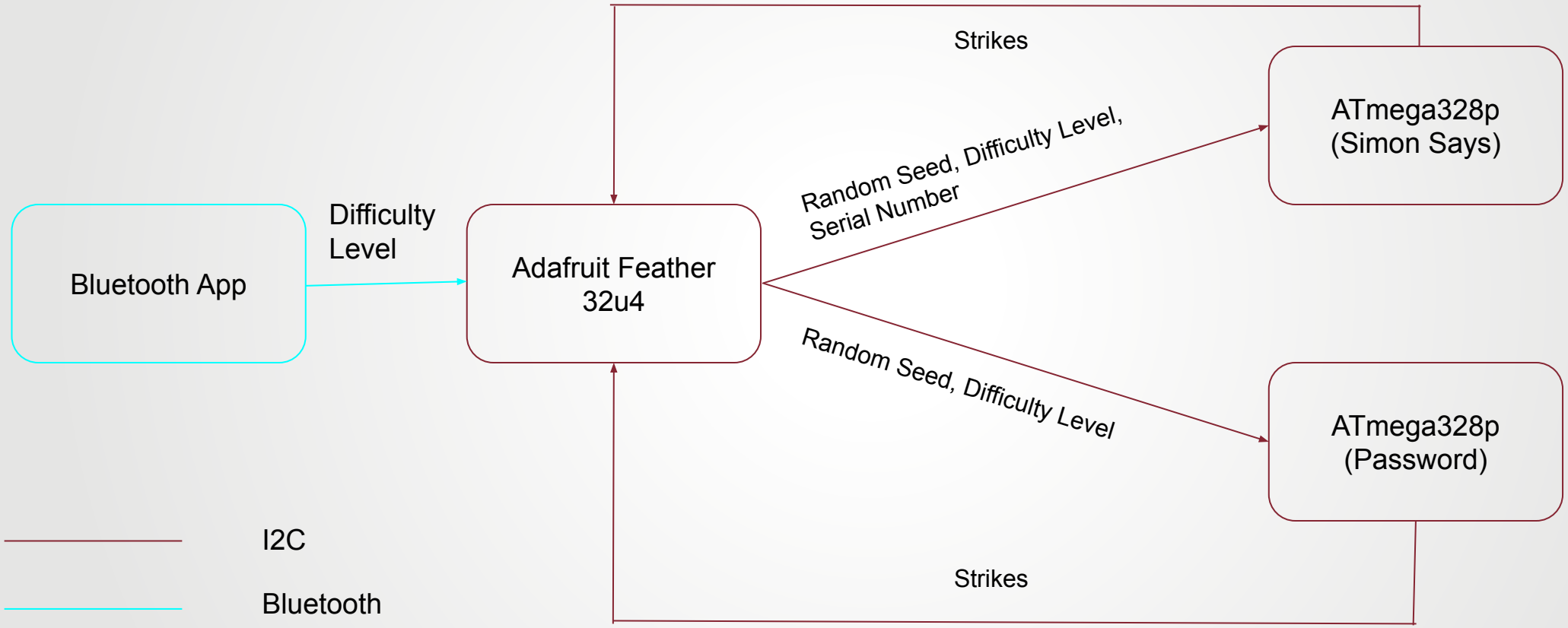
Justification of ATmega328P

- 8-bit Microcontroller
  - Fairly low memory and processor intensive tasks - Save Power
  - Lower numbers = lower data width necessary

- I2C Communication
- Familiarity
- Bare Metal Coding for Easy Transfer to PCB



Module Communication



Power Report

Power Consumption by Module (Worst Case 5 Minute Game)

- **Master Module - 4.2767W**
- **Password Game - 0.281 W**
- **Simon Says Game - 0.176 W**
- **Total = 4.7337 W**
- **Per 5 minute game: 4.7337 W * 5 min = .394 W per game**

For Proposed LIB from PDR:

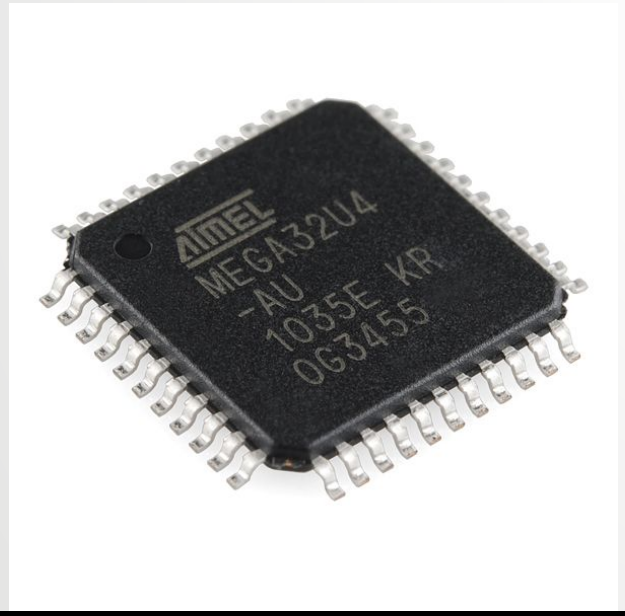
- **6600 mAh**
- **6.6 * 5 = 33 Wh**
- **Would result in around 83 games**

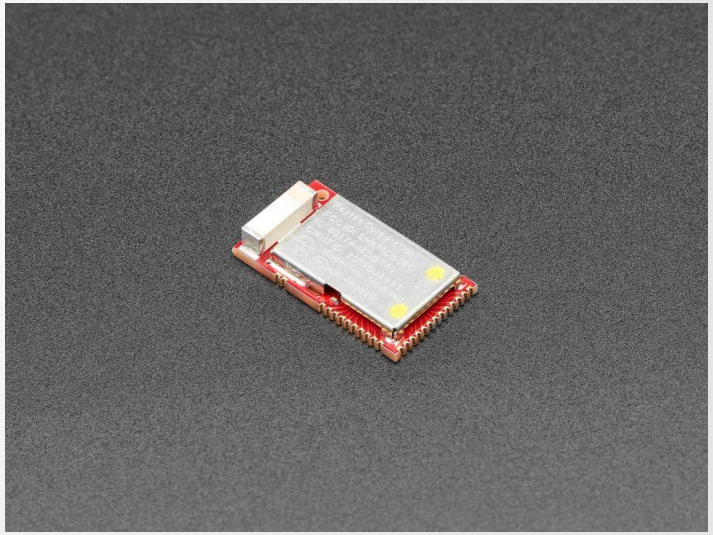
MDR Achievements

- Fully developed 2 Mini Games and the Master Module including ruleset for each game
- Fleshed out the interactions between modules using I2C, system capable of sending and receiving strikes/data
- Determined way of using a random number generator to satisfy needs of the system
- Coded Mini Games Bare metal for easy transfer to PCB models
- Incorporated level difficulty into each child module
- Developed a basic version of a bluetooth app capable of meeting needs of the system

Custom PCB Plan (Master Module)

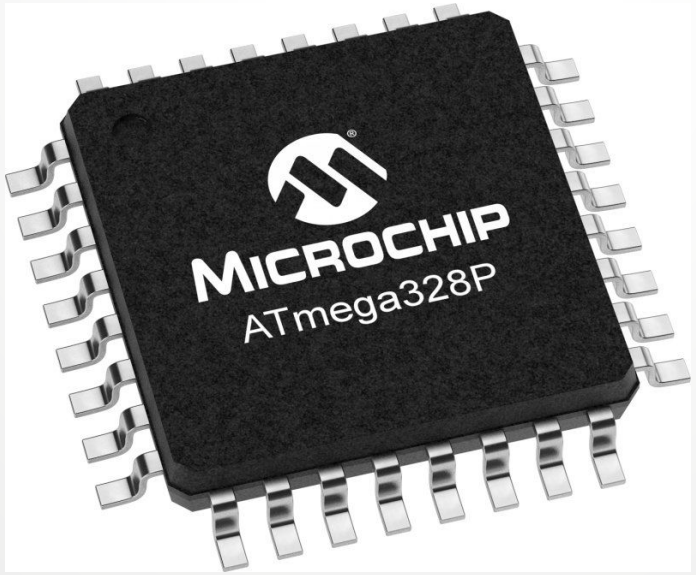
- MCU: ATmega328p (Clock) and ATmega32u4 (Strikes, I2C, etc.)
- Sensors: Our Project Does not include Sensors
- Special ICs: nRF51822 bluetooth module
- Power Supply of 5V, and regulator down to 3.3V





Custom PCB Plan (Child Modules)

- MCU: ATmega328P
- Sensors: No sensors will be used
- Special ICs: No special IC’s will be used, only GPIO
- Power Supply of 5V



Team Member Responsibilities

Ethan LaFleur:

- Team Coordinator: In charge of organizing and running weekly meetings both with the team and also with advisor. Coordinate our presentations/demonstrations with team evaluators.
- Hardware Lead: Determine the hardware that will be used for each module and how to assemble efficiently.

Krishna Vijayakumar:

- Budget Lead: Ensure budget is spent effectively and keep track of total money spent. Keep track of orders.
- On Board Programmer: Develop software to run based on game/ system specifications

Matt Buiser:

- PCB Lead: Tasked with tracking parts and coordinating with companies, making sure parts will arrive in a timely manner
- Off Board Programmer: Design IOS application and implement bluetooth communication from app to master microcontroller.

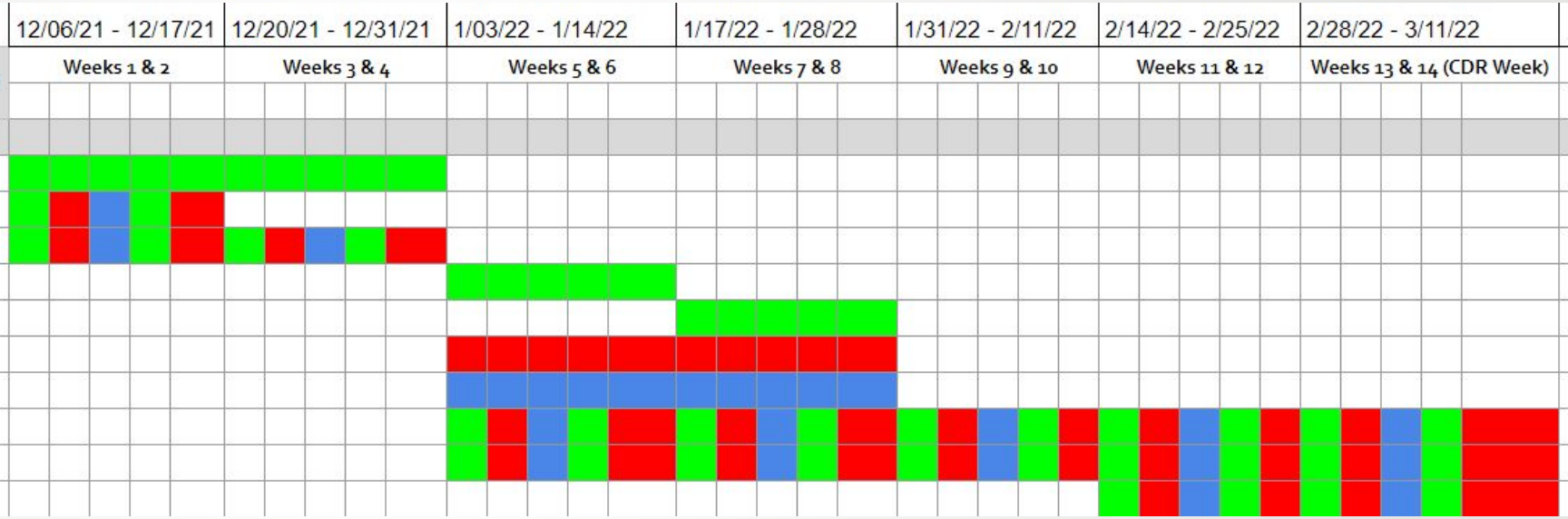
Budget

SDP Project Expendatures	Price	Total Cost	Money Left
Lithium Ion Battery Pack	24.5	114.39	385.61
5V 2.5A Switching Power Supply	7.5		
PowerBoost 1000 Charger	19.95		
Shipping	11.7		
4 Arcade Buttons	10		
10 Quick Connect Wire Pairs	4.95		
Quad Alphanumeric Display	13.95		
2 Qwiic JST SH 4 pin to Premium Male Headers Cables	1.9		
Shipping	8.99		
16x2 Standard LCD	10.95		
Shipping	9.1		

- Our estimated PCB costs + revisions total to $230
- Leaves us with over $150 to spend on other modules + encasing

Gantt Chart

TASK TITLE	TASK OWNER	START DATE	COMPLETION DATE
Restructure Code	Matt	12/6/21	12/31/21
Learn Altium	All	12/6/21	12/17/21
Initial PCB Revisions	All	12/6/21	12/31/21
Order PCB	Matt	1/3/21	1/8/21
Migrate MM to Bare Metal	Matt	1/3/22	1/28/22
Migrate to PCB	Ethan	1/3/22	1/28/22
Develop Game 3	Krishna	1/3/2022	1/28/22
Test PCBs	All	1/31/22	3/11/22
Test Game 3	All	1/31/22	3/11/22
CDR Prep	All	2/14/22	3/11/22




Thank you for your time!
University of Massachusetts Amherst
BE REVOLUTIONARY™