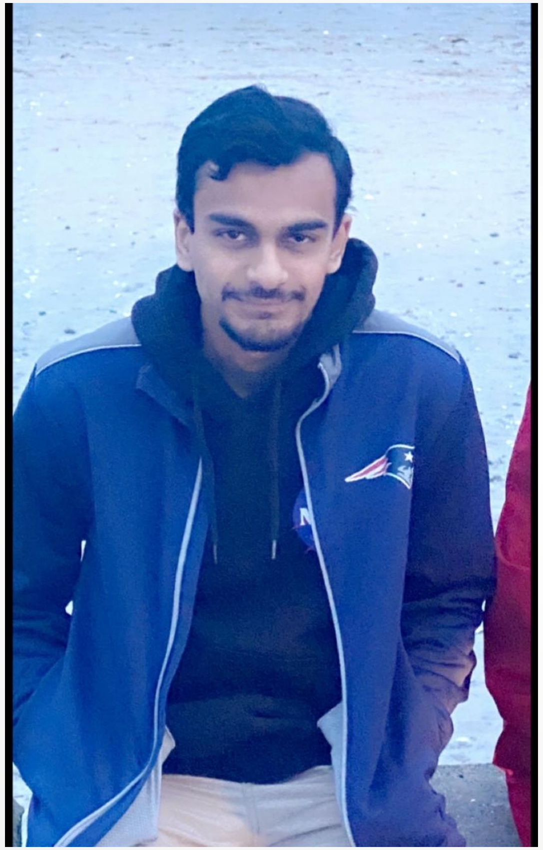
Team 26: Bomb Squad

University of Massachusetts Amherst BE REVOLUTIONARY™

Team Members

Ethan LaFleur - Electrical Engineer

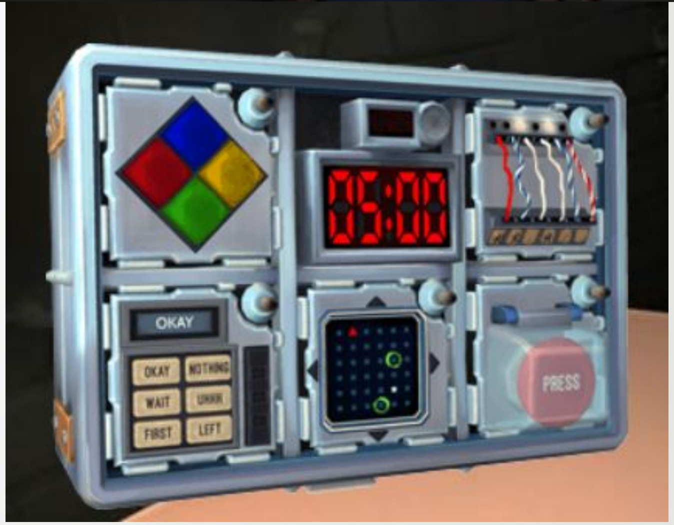


Krishna Vijayakumar - Computer Engineering

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
  - Must feel random to player
  - At least over 20+ level configurations for all modules
- Includes difficulties in the form of levels that a user can set before each run
  - Automated in game, affects each game differently, overall time may be lower as it increases
  - 5+ difficulty levels
- Includes at least 2 regular modules and 1 “needy” module
  - Simon Says
  - Password Game
- Modular approach
  - One master module to keep track of all game modules
  - Each module contains its own information and ruleset
  - Communication between modules
- Rechargeable power source capable of 30 hours of gameplay at a time
- Manual will be converted to an app
  - Selectable options for modules on app
  - Send difficulty via app

Intended CDR Deliverables

- ✓ All modules on solderless breadboard/PCB without unexempted hardware
- ✓ Custom PCB designs for all modules
- ▲ Restructured code without unexempted libraries (all code written in C)
- ✓ Report on randomness implementation in new system
- ✓ Description of plan for power in system

Test Plan

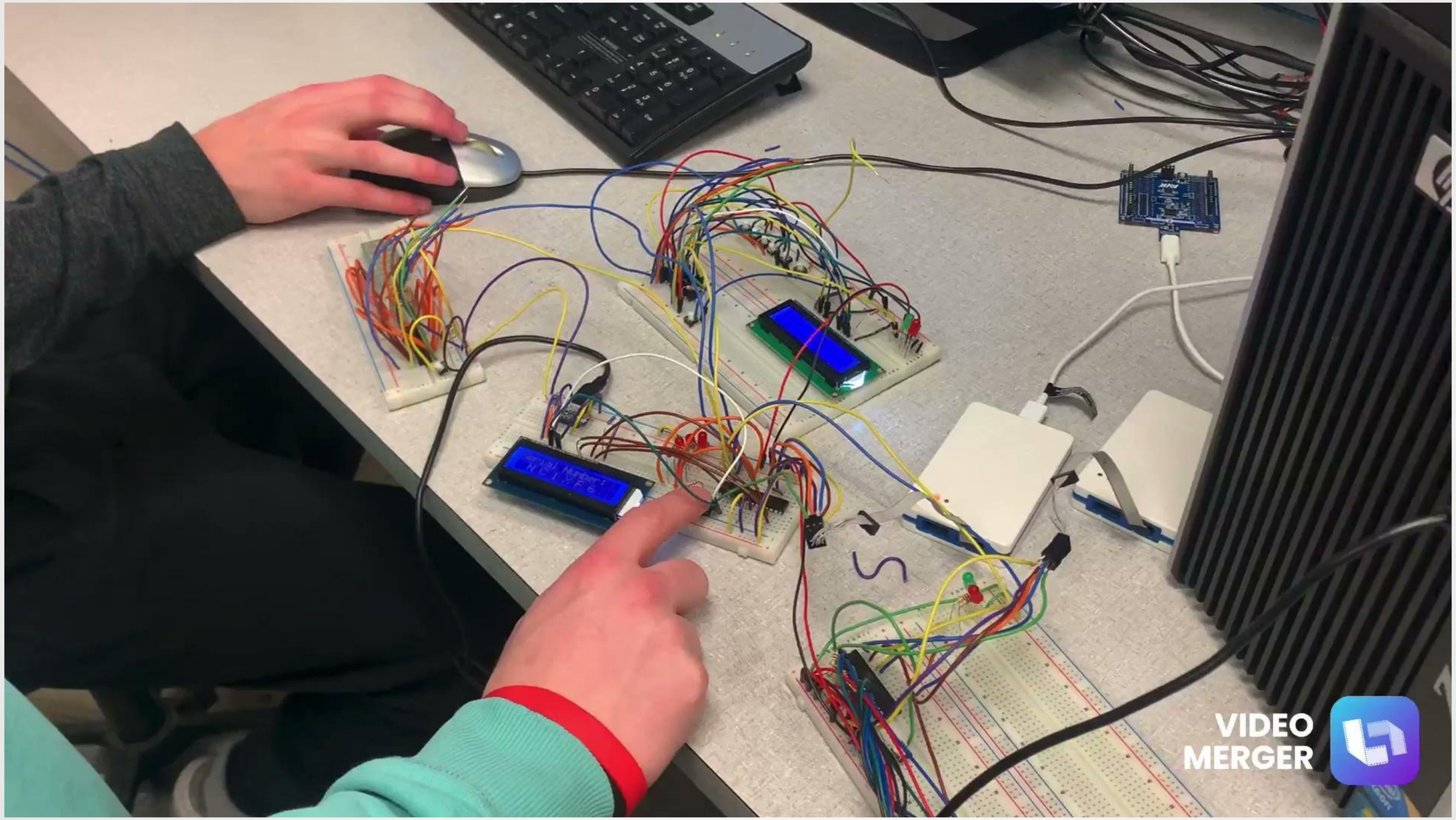
- **Demonstration of game being played in front of evaluators**
  - Play game on multiple different difficulty levels

- **Solder and test each PCB with specific peripherals**
  - Elaborate on components of PCBs we are satisfied with
  - Outline prospective features for next revisions

- **Report and evaluate the randomness implemented in our system**
  - Detail how randomness is implemented in C
  - List number of combinations possible
  - Take measurements (record how many games played before repeat and find total unique configurations) to determine if randomness is satisfactory for our purposes

- **Description of current power usage and plan for power going forward**
  - Explain our plan to power the system for FPR

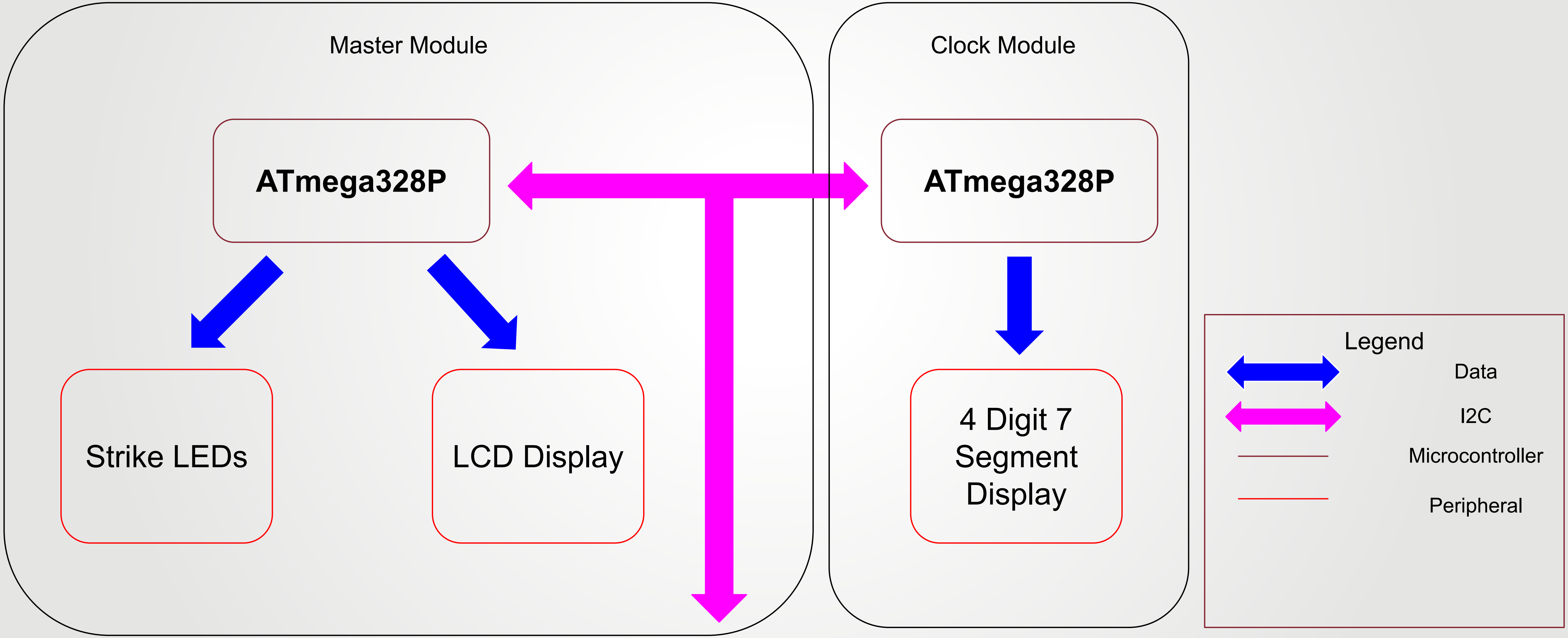
Demo Video



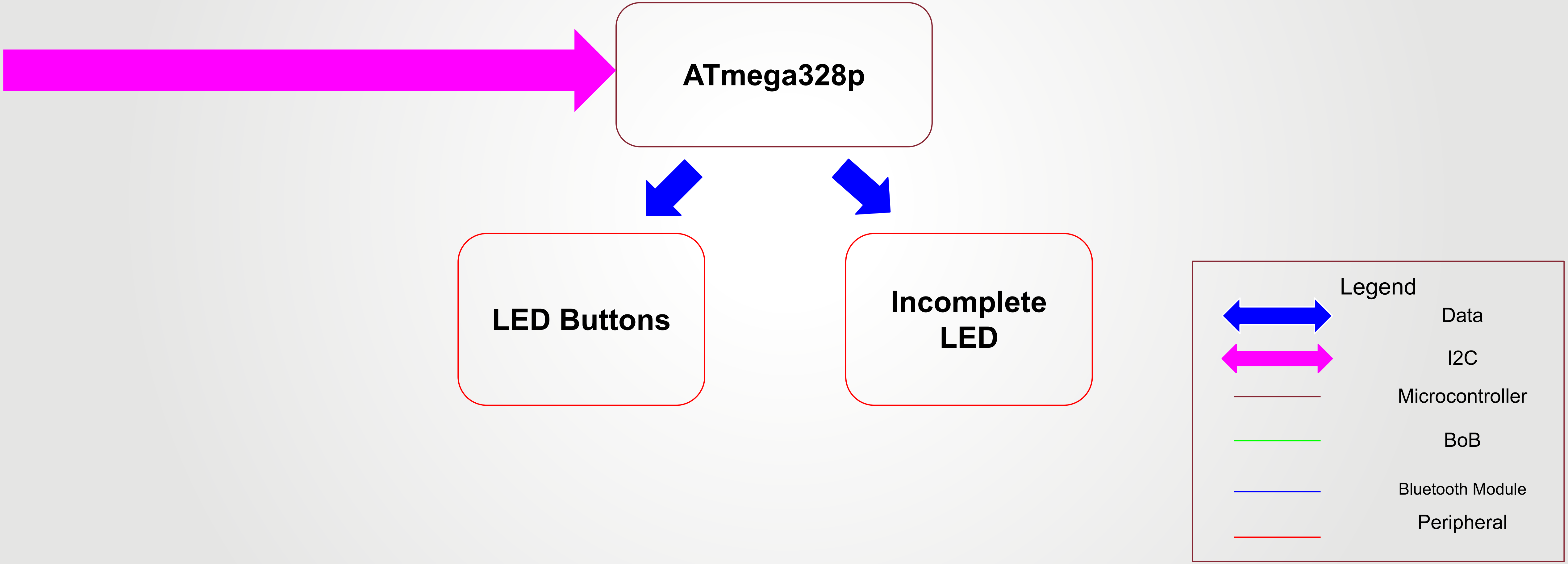
Current Status of Specifications

- ✓ Uses a physical “bomb” with modules installed
- ✓ Randomizes each module each time a new level starts (20+ configurations)
- ✓ Includes difficulty in the form of levels (5+ different levels)
- ✗ Includes at least two regular modules and one needy module
- ✓ Capable of 30 hours of gameplay before recharge
- ✓ Manual will be converted to an app

Master Module Block Diagram



Simon Says Block Diagram



ATmega328p

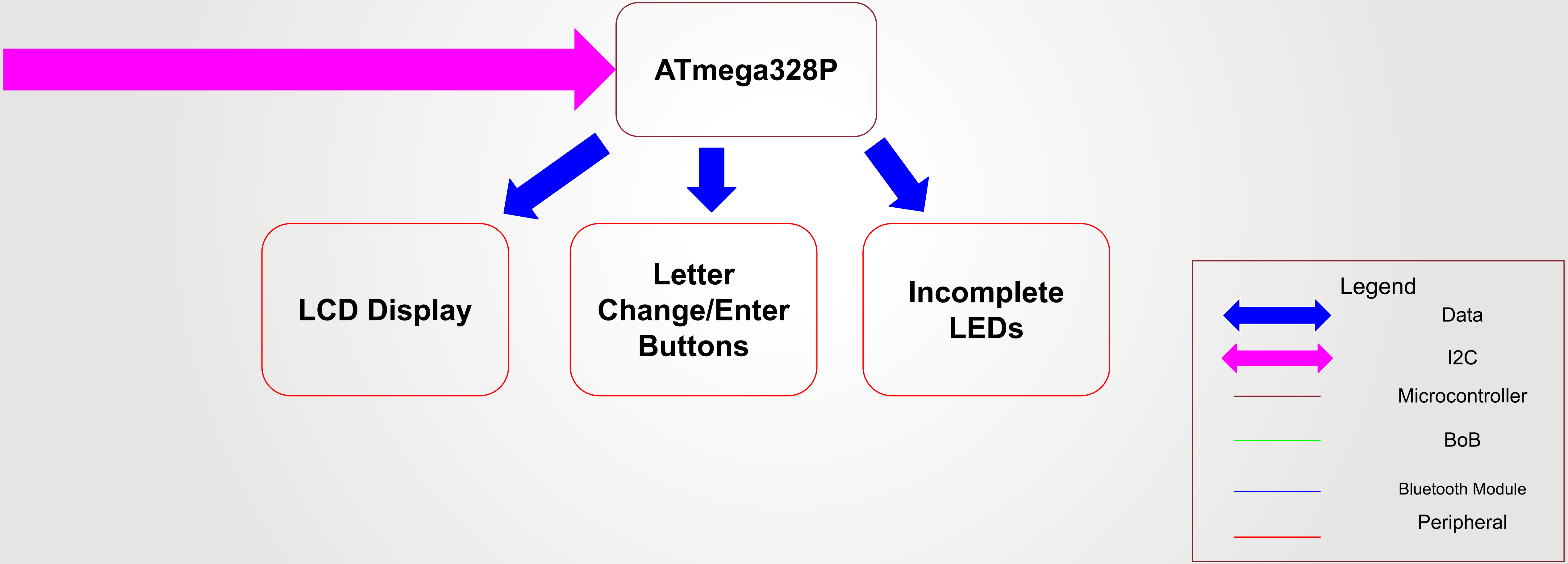
LED Buttons

Incomplete LED

Legend

⟷	Data
⟷	I2C
—	Microcontroller
—	BoB
—	Bluetooth Module
—	Peripheral

Password Game Block Diagram



ATmega328P

LCD Display

Letter Change/Enter Buttons

Incomplete LEDs

Legend

Data

I2C

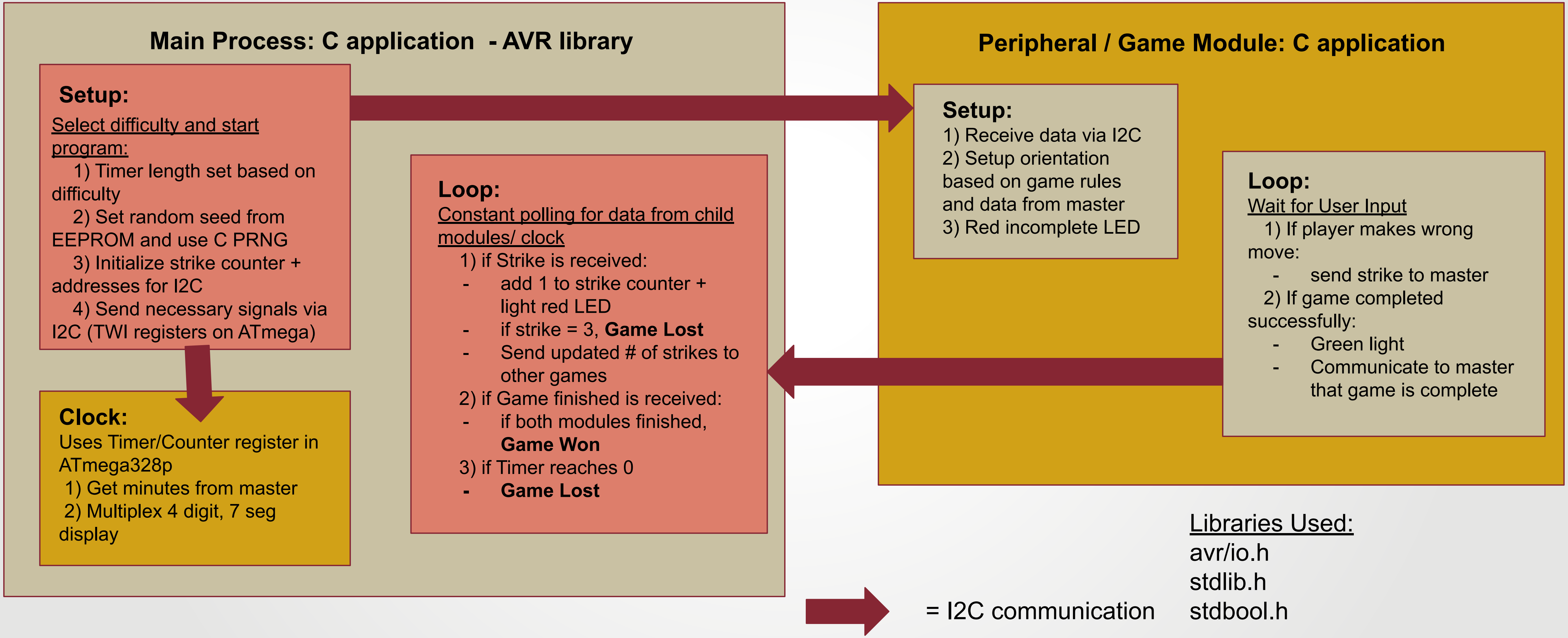
Microcontroller

BoB

Bluetooth Module

Peripheral

Software Block Diagram - Game State



Main Process: C application - AVR library

Setup:

<u>Select difficulty and start program:</u>

1) Timer length set based on difficulty
2) Set random seed from EEPROM and use C PRNG
3) Initialize strike counter + addresses for I2C
4) Send necessary signals via I2C (TWI registers on ATmega)

Clock:

Uses Timer/Counter register in ATmega328p

1) Get minutes from master
2) Multiplex 4 digit, 7 seg display

Loop:

<u>Constant polling for data from child modules/ clock</u>

1) if Strike is received:
   - add 1 to strike counter + light red LED
   - if strike = 3, **Game Lost**
   - Send updated # of strikes to other games
2) if Game finished is received:
   - if both modules finished, **Game Won**
3) if Timer reaches 0
   - **Game Lost**

Peripheral / Game Module: C application

Setup:

1) Receive data via I2C
2) Setup orientation based on game rules and data from master
3) Red incomplete LED

Loop:

<u>Wait for User Input</u>

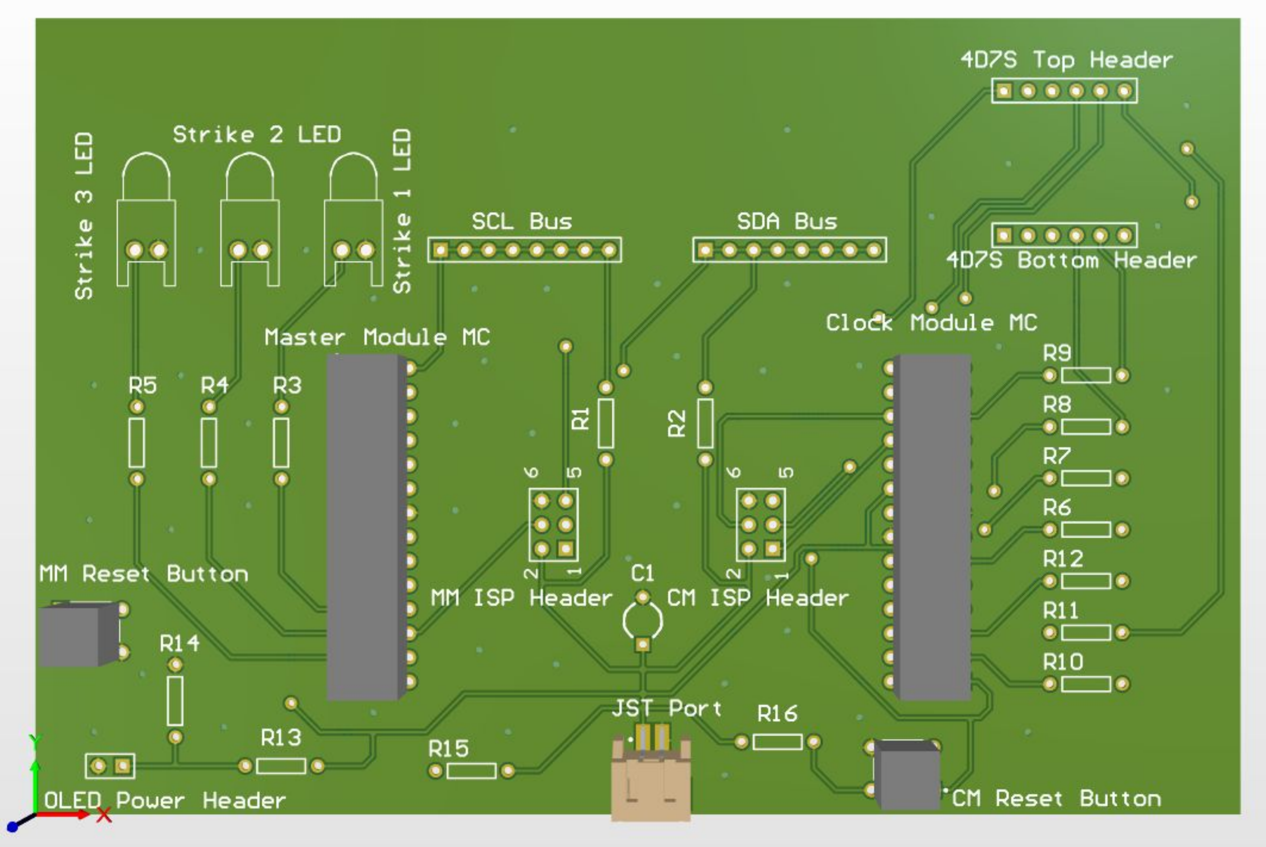
1) If player makes wrong move:
   - send strike to master
2) If game completed successfully:
   - Green light
   - Communicate to master that game is complete

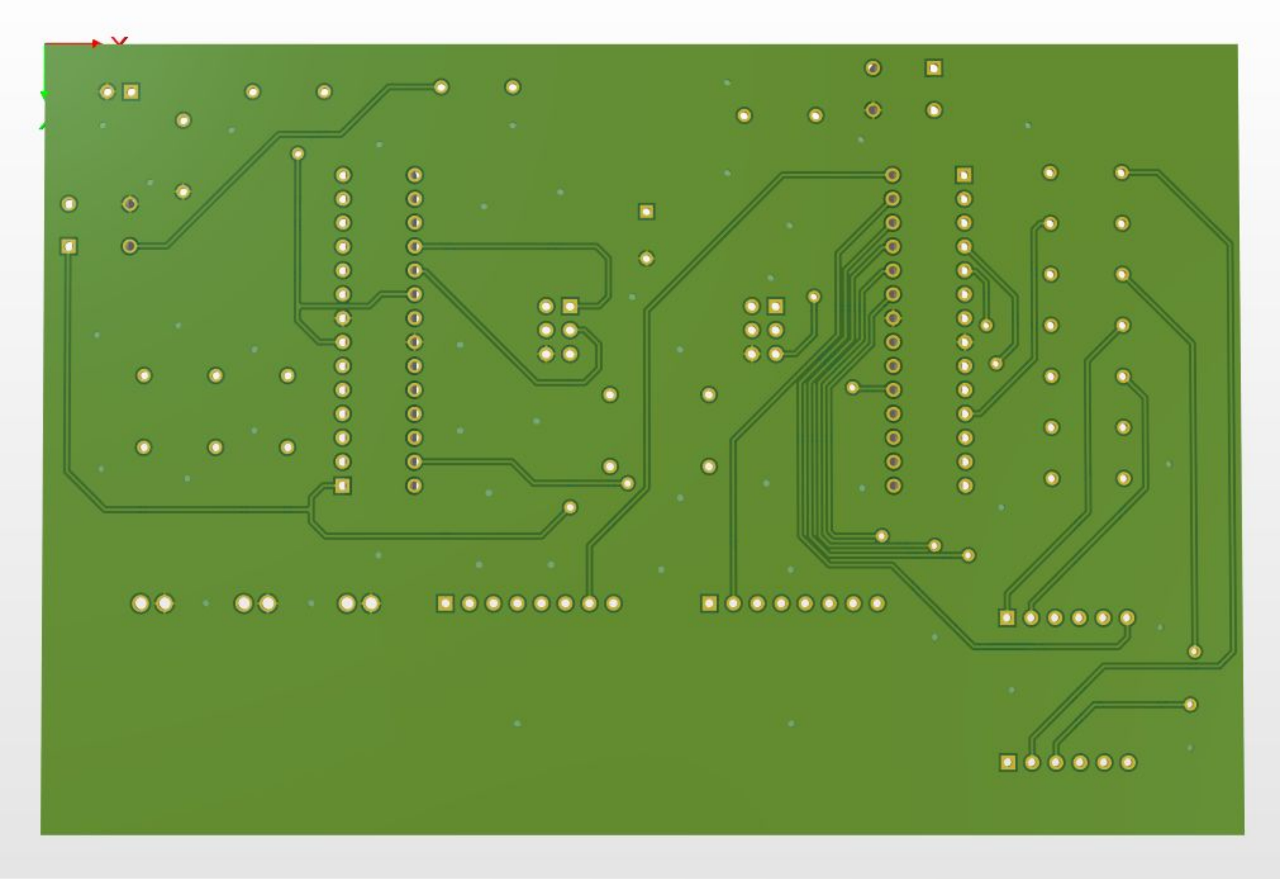
= I2C communication

<u>Libraries Used:</u>

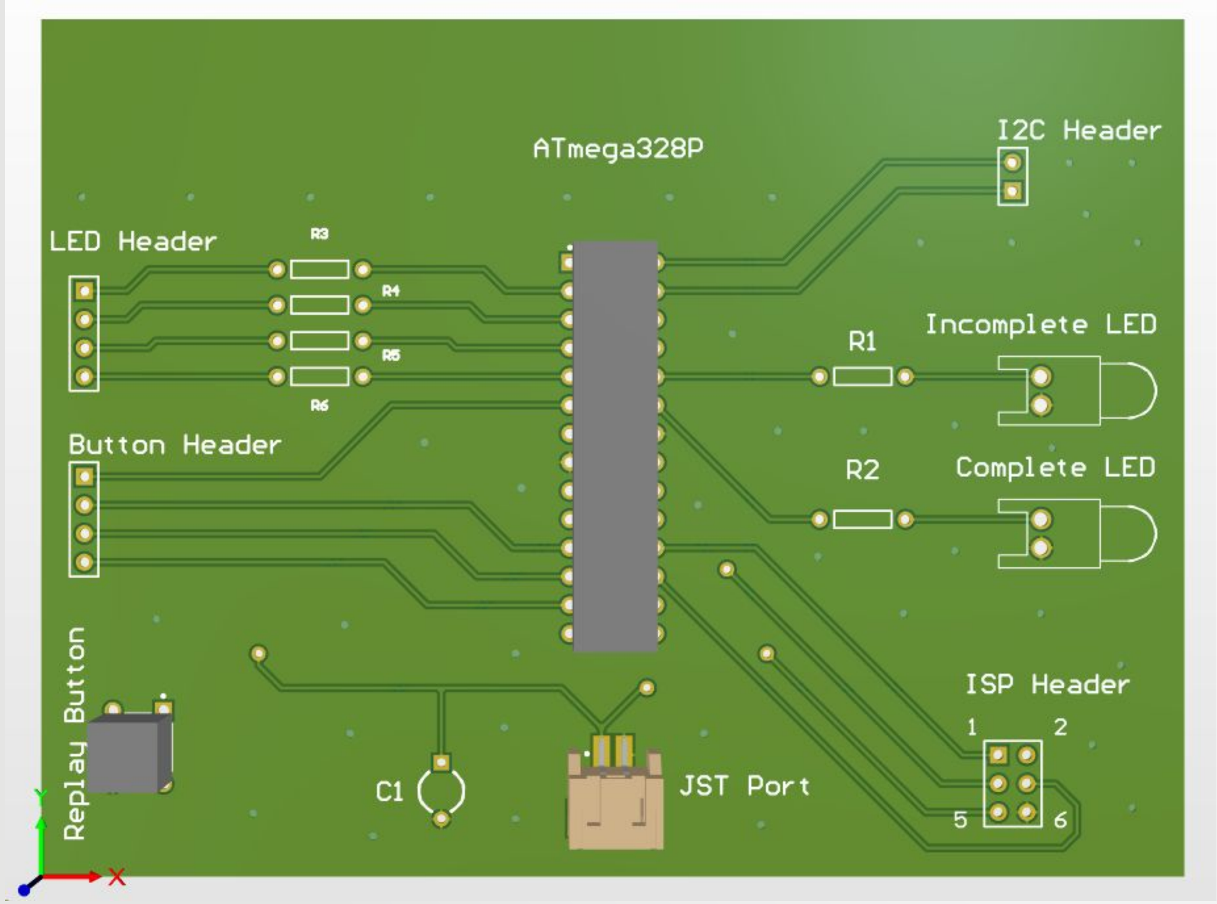
avr/io.h
stdlib.h
stdbool.h

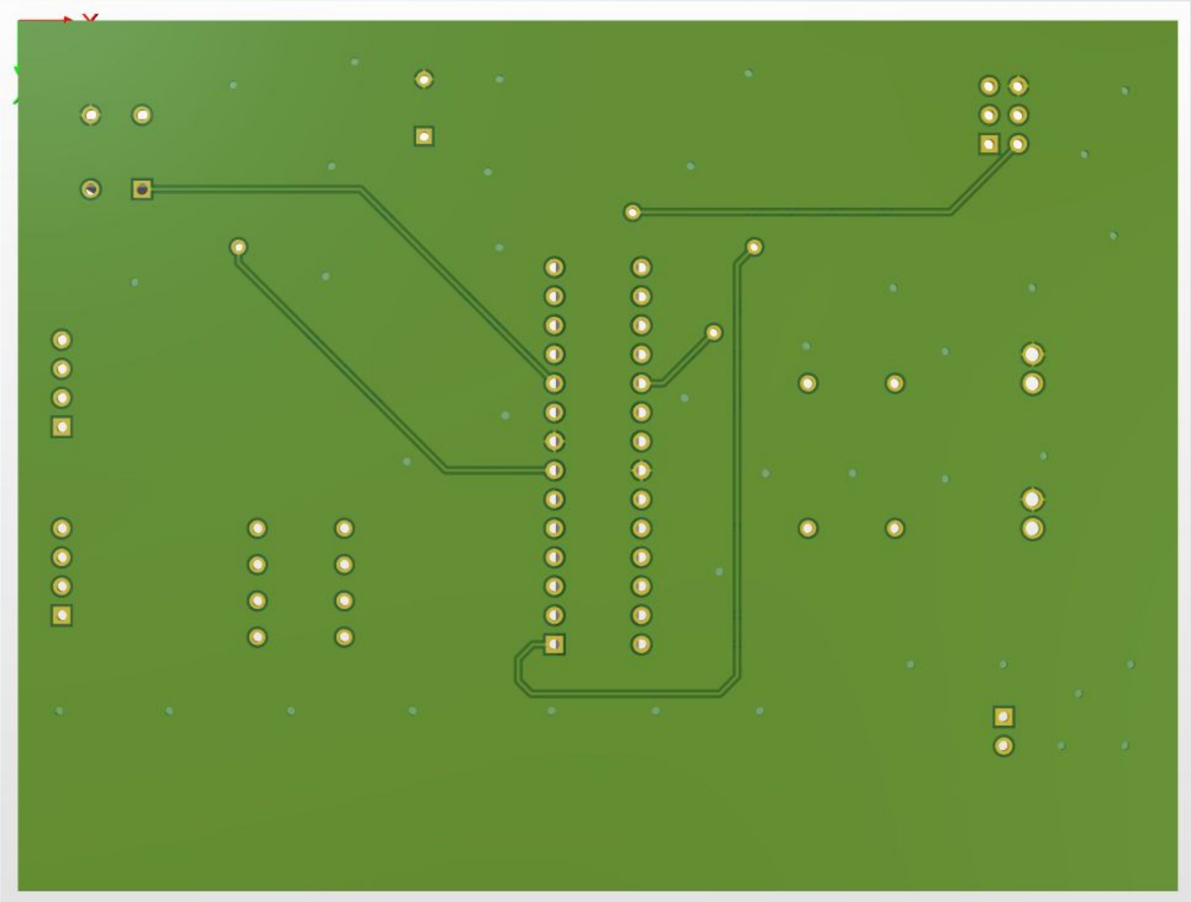
Master Module Custom PCB



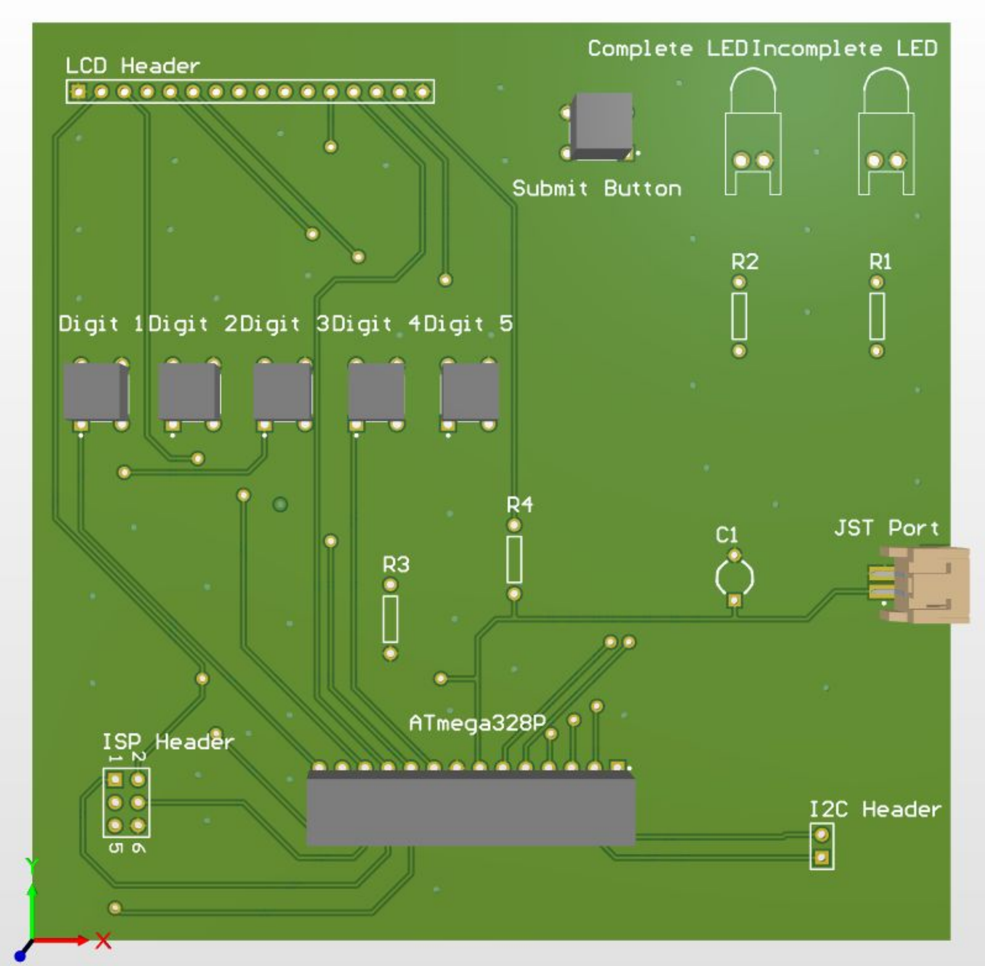


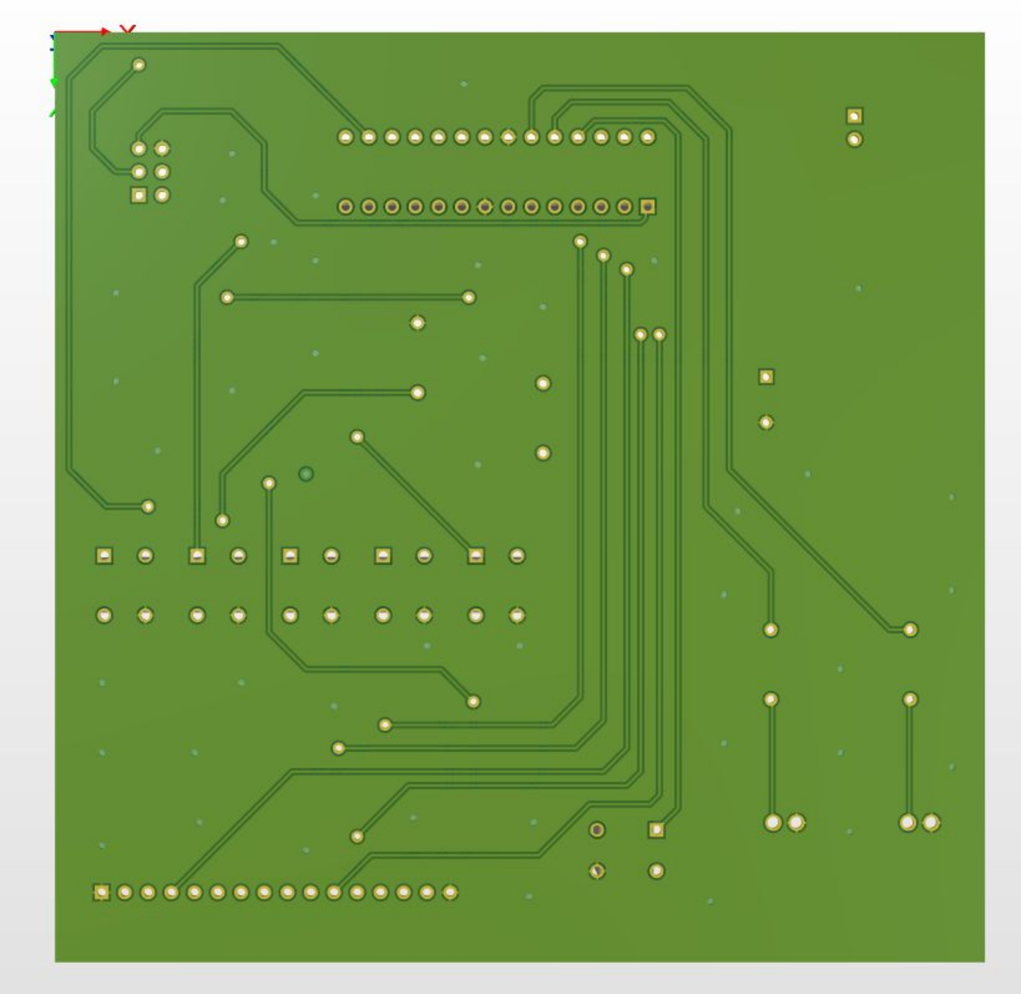
Simon Says Custom PCB





Password Game Custom PCB





Master Module Custom PCB Next Revisions

- Remove OLED components
- Add LCD component
- Create more space for 4-digit 7-segment display
- Add JST Port for power/ground on either side
- Add connections for bluetooth module
- Add mounting holes

Child Module Custom PCB Next Revisions

- **Add reset button**
- **Add power/ground headers (Simon Says)**
- **Create more space for LCD display (Password)**
- **Mounting Holes for enclosure**

Randomness Revisited

- **Use EEPROM on ATMega328P (Master Module)**

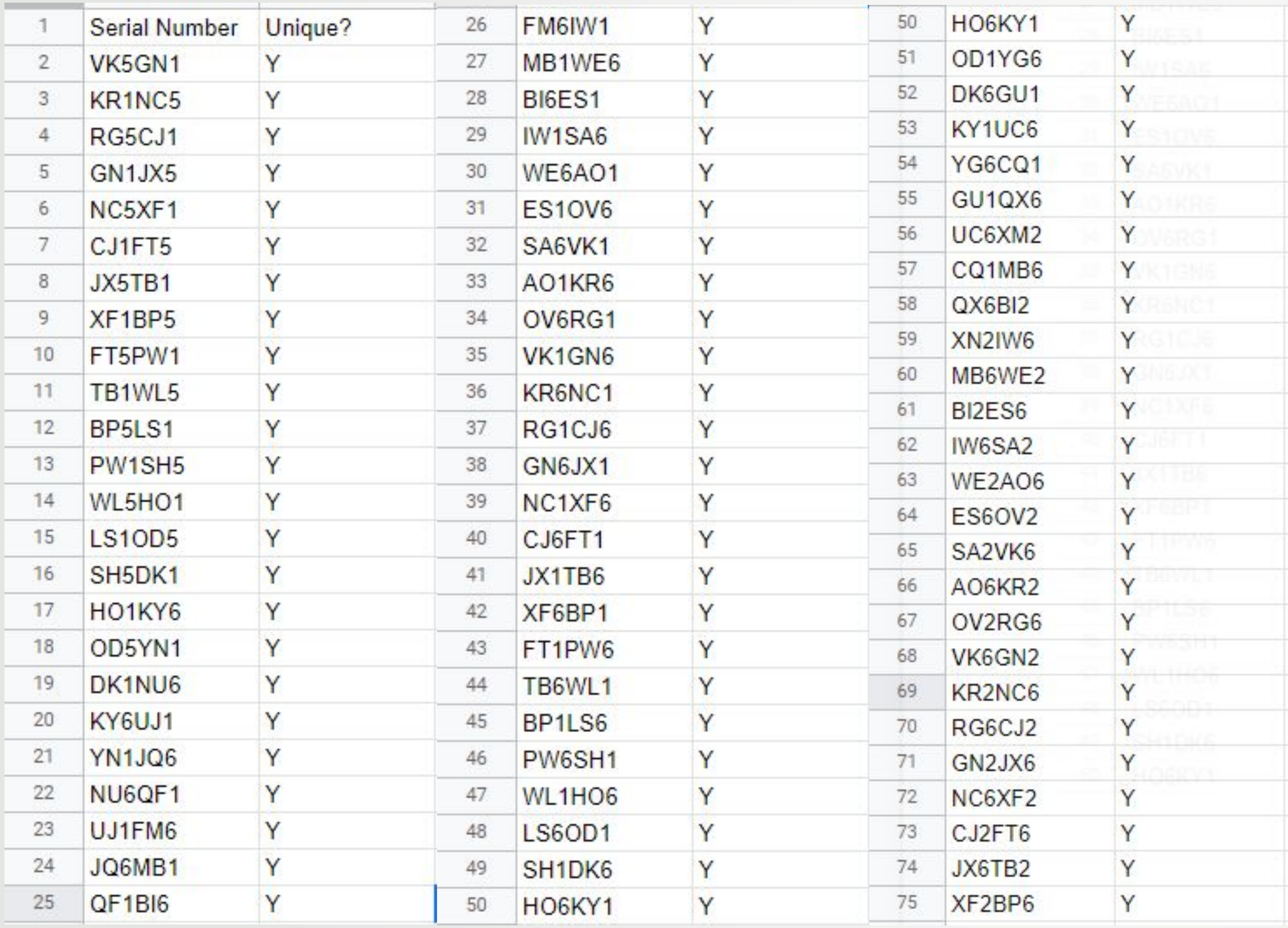
- **Store random seed in EEPROM**

- **Increment random seed each time**

- **8 bit register ~ 256 different seeds**

- **rand() - built in pseudorandom function**
    - no strong guarantees about quality of random numbers

Randomness Data



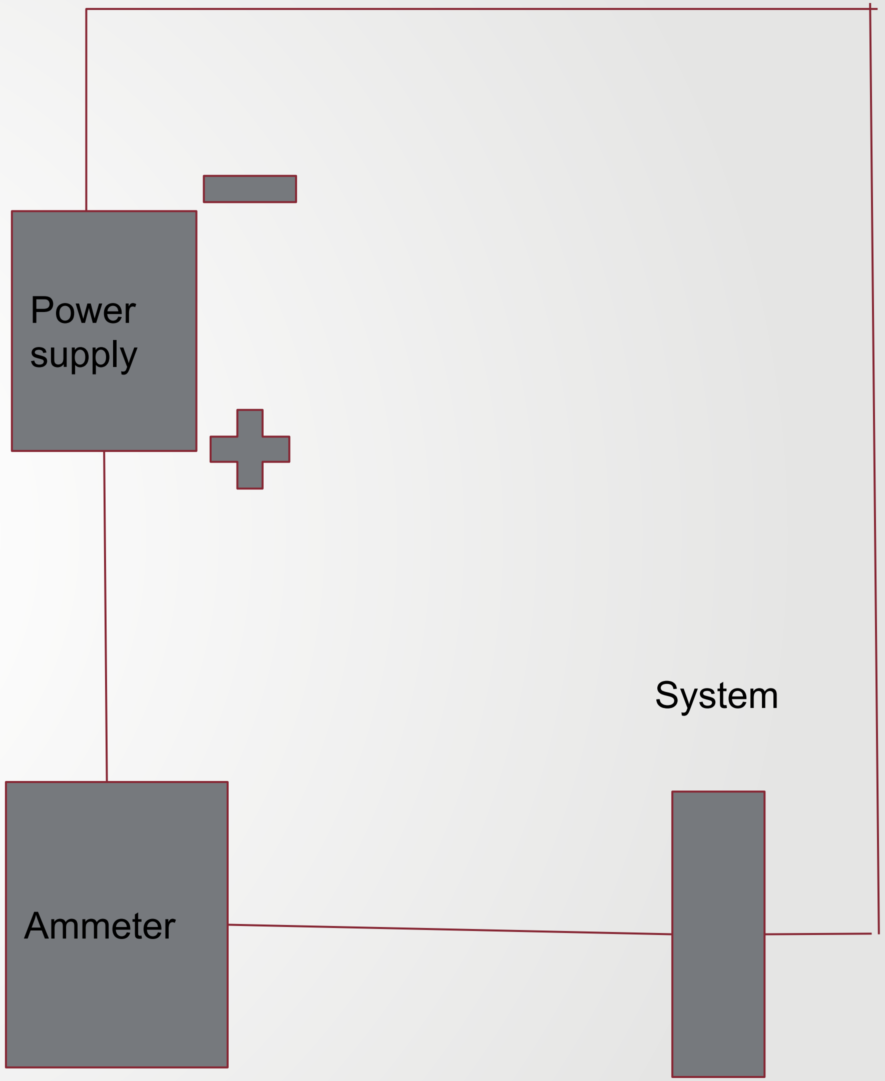
1	Serial Number	Unique?
2	VK5GN1	Y
3	KR1NC5	Y
4	RG5CJ1	Y
5	GN1JX5	Y
6	NC5XF1	Y
7	CJ1FT5	Y
8	JX5TB1	Y
9	XF1BP5	Y
10	FT5PW1	Y
11	TB1WL5	Y
12	BP5LS1	Y
13	PW1SH5	Y
14	WL5HO1	Y
15	LS1OD5	Y
16	SH5DK1	Y
17	HO1KY6	Y
18	OD5YN1	Y
19	DK1NU6	Y
20	KY6UJ1	Y
21	YN1JQ6	Y
22	NU6QF1	Y
23	UJ1FM6	Y
24	JQ6MB1	Y
25	QF1BI6	Y
26	FM6IW1	Y
27	MB1WE6	Y
28	BI6ES1	Y
29	IW1SA6	Y
30	WE6AO1	Y
31	ES1OV6	Y
32	SA6VK1	Y
33	AO1KR6	Y
34	OV6RG1	Y
35	VK1GN6	Y
36	KR6NC1	Y
37	RG1CJ6	Y
38	GN6JX1	Y
39	NC1XF6	Y
40	CJ6FT1	Y
41	JX1TB6	Y
42	XF6BP1	Y
43	FT1PW6	Y
44	TB6WL1	Y
45	BP1LS6	Y
46	PW6SH1	Y
47	WL1HO6	Y
48	LS6OD1	Y
49	SH1DK6	Y
50	HO6KY1	Y
50	HO6KY1	Y
51	OD1YG6	Y
52	DK6GU1	Y
53	KY1UC6	Y
54	YG6CQ1	Y
55	GU1QX6	Y
56	UC6XM2	Y
57	CQ1MB6	Y
58	QX6BI2	Y
59	XN2IW6	Y
60	MB6WE2	Y
61	BI2ES6	Y
62	IW6SA2	Y
63	WE2AO6	Y
64	ES6OV2	Y
65	SA2VK6	Y
66	AO6KR2	Y
67	OV2RG6	Y
68	VK6GN2	Y
69	KR2NC6	Y
70	RG6CJ2	Y
71	GN2JX6	Y
72	NC6XF2	Y
73	CJ2FT6	Y
74	JX6TB2	Y
75	XF2BP6	Y

- Same random seed shared between all modules, thus repeat SN = repeat game config
- Srand() clearly not perfect
- There are 256 unique configurations for the overall system
- However, it is not perfectly random
- Repeat two letter combinations
- VK appears in test run 2, 32, and 35
- Different number combinations for those runs, technically unique
- Same 2 numbers used for each 34 run bunch
- E.g. 5 and 1 for runs 2-34
- 256 unique configurations satisfies system spec

Power Report

- Estimated atmega328p current draw and operating voltage: ~15mA (idle minimum) & 5V
- Four atmegas drawing power from same source
- 4 * 15mA * 5V = 300mW

- Measured current draw (all microcontrollers) when idle (waiting on user input): ~59mA
- Idle power: 59mA * 5V = 295mW
- Peaks to no more than 80mA when user input
  - does little computation
- How long can our device be powered?
  - 6600 Wh / 0.295 W = 22,372 hours



Power Plan Going Forward

- “How will we power the whole system”
- One battery (3.7V LIB)
- Connected via JST port to Master PCB
- Spread to other modules via JST

FPR Plan

- **Revise and finalize PCBs**
  - Entire system migrated to PCBs
  - Bluetooth introduced on PCBs

- **Demonstrate game**
  - Create enclosure for game
  - Game played multiple times on multiple difficulties
  - Play game with someone other than the team

- **Large sample size and analysis of randomness in game**
  - Does it meet system requirements?
  - Metrics: Approximate Entropy, NIST statistical test suite

- **Report the final power usage of PCBs**
  - How long can the game be played?

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, as well as designing the custom PCB’s.

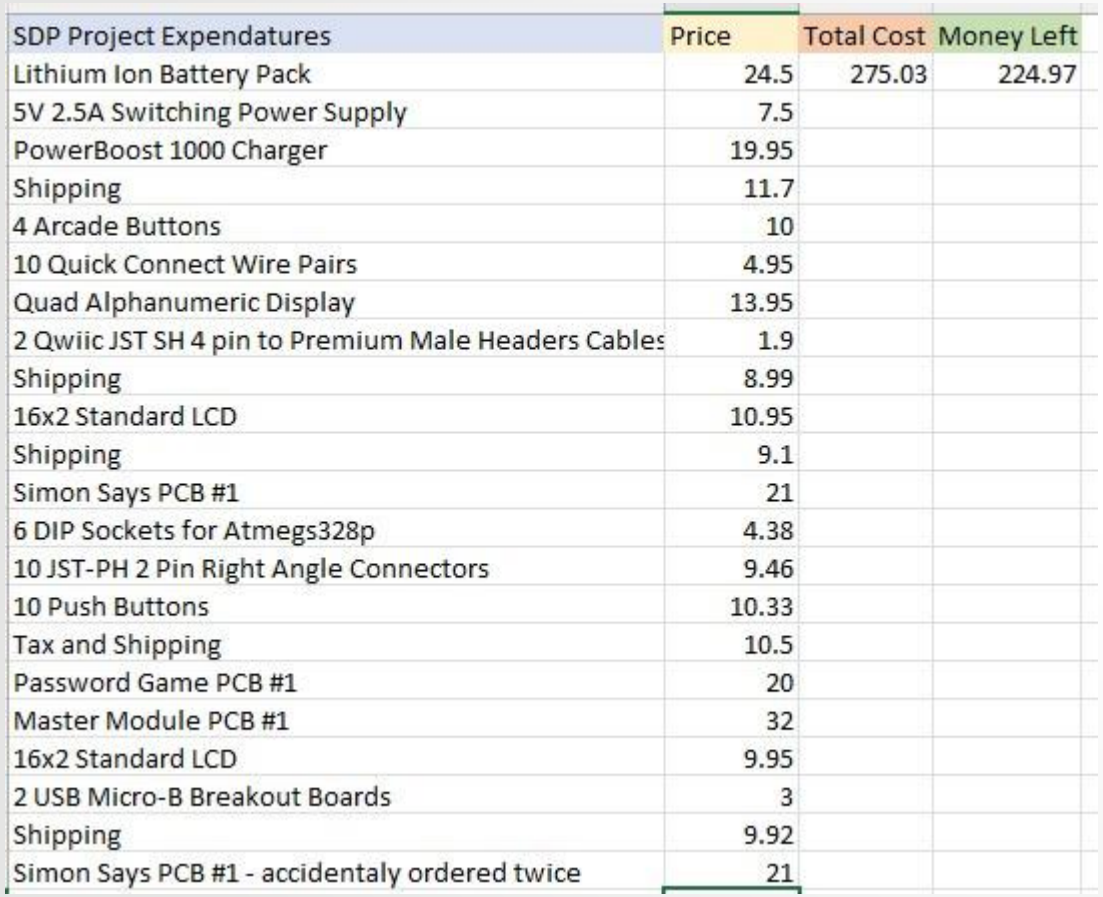
Krishna Vijayakumar:

- 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.

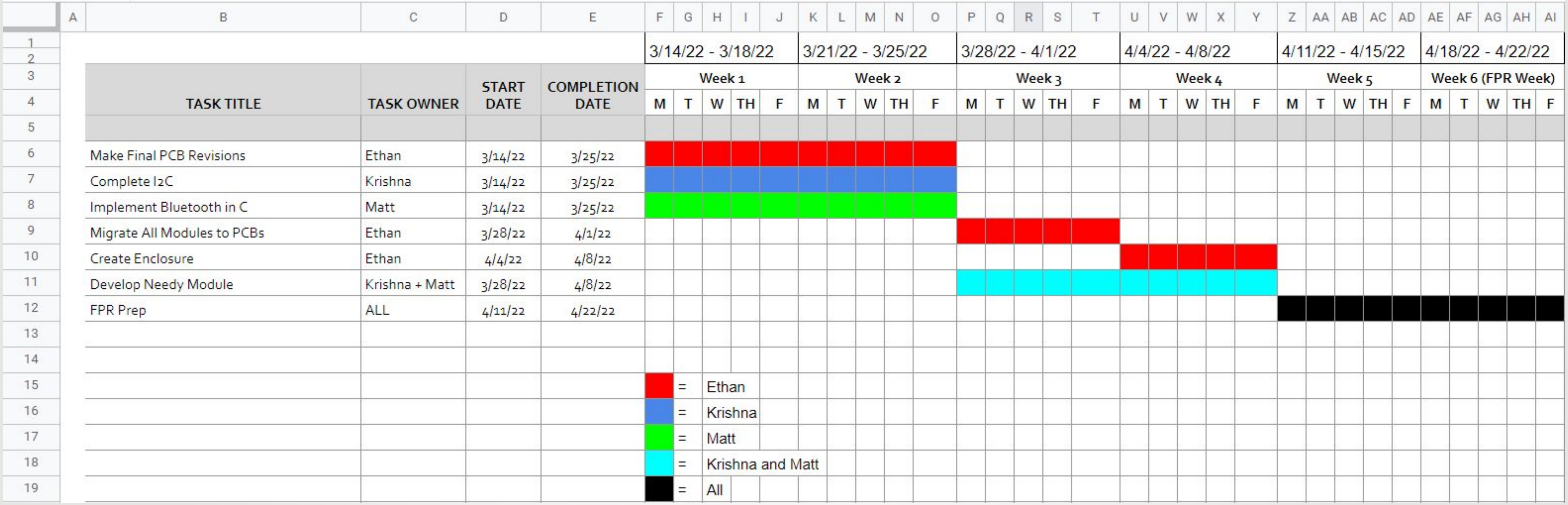
Budget



SDP Project Expendatures	Price	Total Cost	Money Left
Lithium Ion Battery Pack	24.5	275.03	224.97
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 Cable	1.9		
Shipping	8.99		
16x2 Standard LCD	10.95		
Shipping	9.1		
Simon Says PCB #1	21		
6 DIP Sockets for Atmegs328p	4.38		
10 JST-PH 2 Pin Right Angle Connectors	9.46		
10 Push Buttons	10.33		
Tax and Shipping	10.5		
Password Game PCB #1	20		
Master Module PCB #1	32		
16x2 Standard LCD	9.95		
2 USB Micro-B Breakout Boards	3		
Shipping	9.92		
Simon Says PCB #1 - accidentaly ordered twice	21		

- PCBs need revisions
- $225 to spend mainly on PCB revisions, smaller hardware parts, and an encasing

Gantt Chart



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
1 2						3/14/22 - 3/18/22					3/21/22 - 3/25/22					3/28/22 - 4/1/22					4/4/22 - 4/8/22					4/11/22 - 4/15/22					4/18/22 - 4/22/22				
3		TASK TITLE	TASK OWNER	START DATE	COMPLETION DATE	Week 1					Week 2					Week 3					Week 4					Week 5					Week 6 (FPR Week)				
4						M	T	W	TH	F	M	T	W	TH	F	M	T	W	TH	F	M	T	W	TH	F	M	T	W	TH	F	M	T	W	TH	F
5																																			
6		Make Final PCB Revisions	Ethan	3/14/22	3/25/22	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																				
7		Complete I2C	Krishna	3/14/22	3/25/22	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																				
8		Implement Bluetooth in C	Matt	3/14/22	3/25/22	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																				
9		Migrate All Modules to PCBs	Ethan	3/28/22	4/1/22											🟥	🟥	🟥	🟥	🟥															
10		Create Enclosure	Ethan	4/4/22	4/8/22																🟥	🟥	🟥	🟥	🟥										
11		Develop Needy Module	Krishna + Matt	3/28/22	4/8/22											▨	▨	▨	▨	▨	▨	▨	▨	▨	▨										
12		FPR Prep	ALL	4/11/22	4/22/22																					⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
13																																			
14																																			
15						🟥	=	Ethan																											
16						🟦	=	Krishna																											
17						🟩	=	Matt																											
18						▨	=	Krishna and Matt																											
19						⬛	=	All																											

Thank you for your time!

University of Massachusetts Amherst

BE REVOLUTIONARY™