Electric Betting System for Poker

Team 62 – Anand Giridharan, Umang Chavan, Varun Pitta
ECE 445 Design Document – Spring 2019
TA: Nicholas Ratajczyk

1. Introduction

Objective

Texas Hold'em is a variant of the card game of poker. Each player in the game is dealt two cards face down and the dealer goes through three rounds of community betting flipping over 5 cards in total. Each player then makes bets on each round based on the strength of their hand. Each player in the game starts with the same amount of money to use while betting. The money is represented using poker "chips," which are small discs that are often made of plastic or clay. Without these chips, playing poker becomes very difficult.

If a group of people wish to play poker, they would need a deck of cards and a poker chip set. The issue is that not everyone owns a poker chip set and those who do only have a limited amount of chips. This restricts the number of people that can play and how much money can be bet. Poker sets can be rather expensive and sometimes if the chips are made from authentic clay, they can chip and break easily. The problem statement that we are addressing is to see if there is a more effective way of playing poker for cheap without worrying about chips.

Our proposed solution is an electronic betting system that completely eradicates the need of physical poker chips. The idea is to have a centralized unit where all betting happens, and each player can see the community pot. Each player will also have their own device that allows them to see their own money and make poker actions such as raising the bet or calling the bet. By taking away the need to use poker chips, everyday people can play poker without having to worry about the financial restraints and the game restraints.

Background

The idea of removing accessories from games and using an electric alternative has been around for a while. Monopoly Electronic Banking Edition by Hasbro eliminated the need for paper cash that is normally used in games [1]. Instead they created a debit card system, where each player swipes a machine to perform all transactions with the bank or with other players. The purpose of this was to remove the hassle of having so much paper laying around. All the math and counting is done by the middle unit and the player only has to know what action they want to take. The debit cards in this game use magnetic strips to take care of player identification

Our goals are similar to this, but we also wanted to add the feature where each player knows the amount of money they hold as well as the community pot. Instead of using magnetic strips, our player identification will be handled with RFID readers. The hope for the end-product is that it will be efficient enough to eliminate poker chips, be affordable, and still provide enjoyment to the game.

High-Level Requirements

- Players should be able to join at the beginning of any round and see the amount of money other players have as well as the amount of money in the community pot.
- Players should be able to raise, call, or fold from their device, and the central hub should display the chosen action.
- All devices including the central display must be powered with a battery pack. The battery pack must last at least 2.5 hours, which is enough to cover the duration of a poker game.

2. Design

Block Diagram

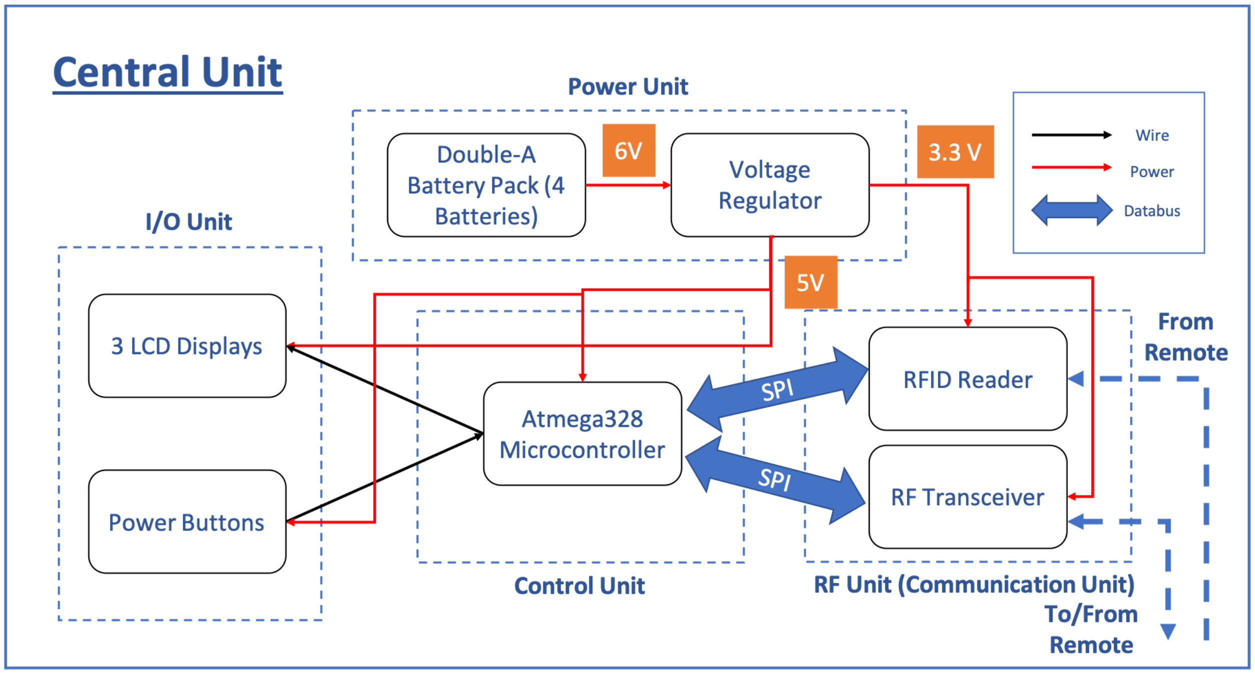


Figure 1: Block Diagram of Central Unit

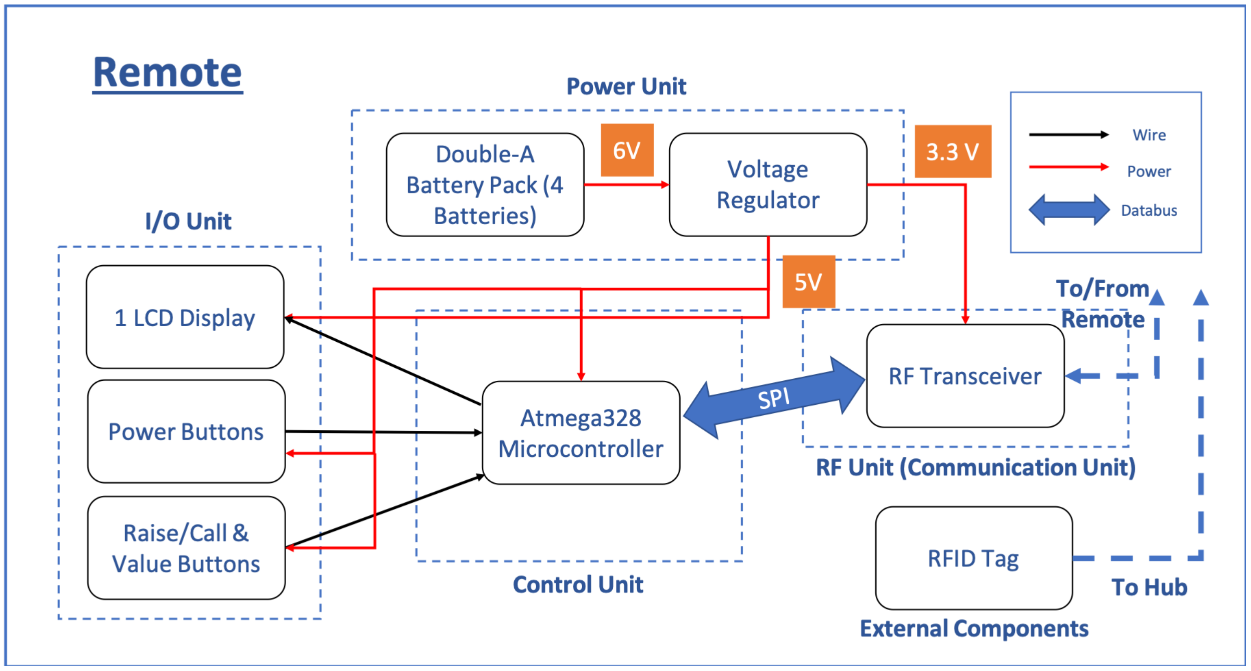


Figure 2: Block Diagram of Remote Unit

Physical Design

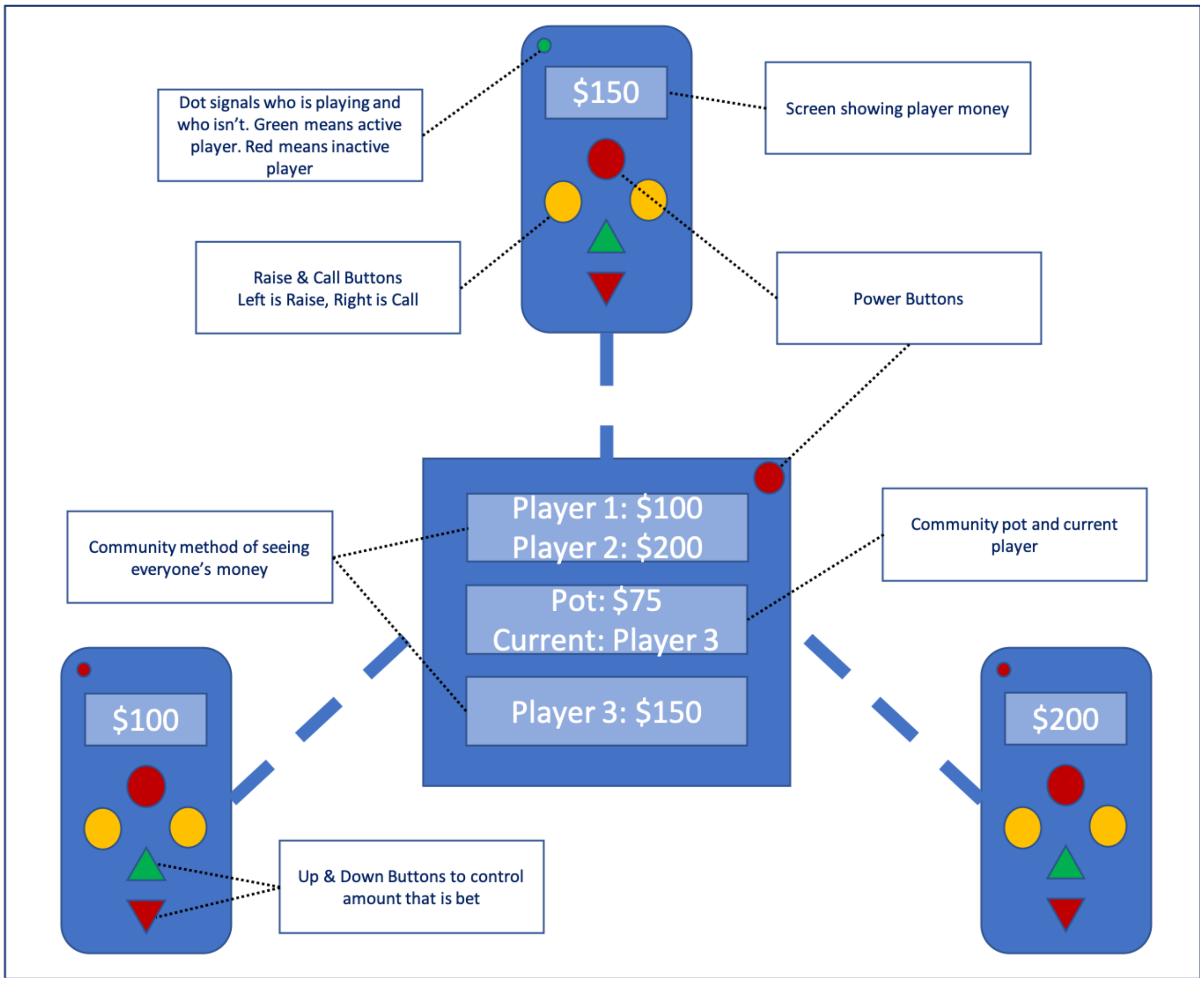

Dot signals who is playing and who isn't. Green means active player. Red means inactive player
$150
Screen showing player money
Raise & Call Buttons Left is Raise, Right is Call
Power Buttons
Player 1: $100
Player 2: $200
Pot: $75
Current: Player 3
Player 3: $150
Community method of seeing everyone's money
Community pot and current player
$100
$200
Up & Down Buttons to control amount that is bet

Figure 3: Physical Layout of Full Apparatus

Our physical design consists of essentially a main display device and the remote(s) for each player. On the main display we plan to display:

1) Each Players Total Money
2) Total Pot Money
3) Current Player

The main display will also contain a power button to turn the device on and off. For the remote(s) we plan to display:

1) Player's total money
2) Amount being raised/called after player bet

We will also have various buttons and indicators such as:

1) Call push button
2) Raise push button
3) Up push button (to increase amount you'd like raise by)
4) Down push button (to decrease amount you'd like to raise by)
5) Power button

Functional Overview

Power Supply

We require a power supply for nearly all our units. We will be using a battery pack with four AA batteries for our power supply.

Requirements	**Verification**
1. Batteries power all our LCDs and Microcontrollers the duration of the game	1. Test this by using a Multimeter (Voltmeter specifically) to see how much voltage is outputted from N number of batteries and see if they reach the minimum needed to power all the components (need ~3.3V and ~5V to power all parts).

Voltage Regulator

We are using two voltage regulators, one for our RF transceiver and one for our microcontroller/LCDs. This is necessary to be able to step down our 6 V input (4 batteries) to 3.3 V max for our RF transceiver (STMicroelectronics LD1117AV33), and 5 V max for our microcontroller and LCDs (Sparkfun L7805).

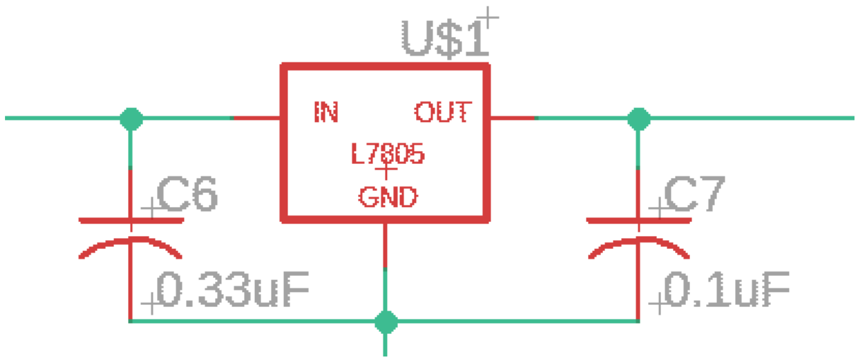


Figure 4: 5V Regulator

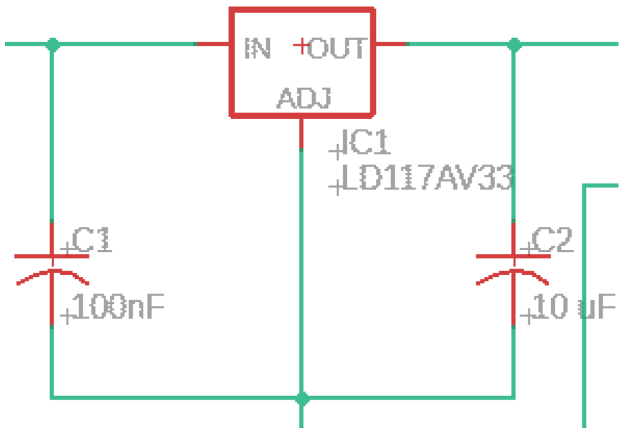


Figure 5: 3.3V Regulator

Requirements	**Verification**
1. Must ensure that the step down from 6 V to 3.3 V (+/- 0.3v) is successful through the use of voltage regulator 2. Must ensure that the step down from 6 V to 5 V (+/- 0.2V) is successful through the use of voltage regulator	1. Send a 6 V input voltage through the voltage regulator and measure the output to see if it is 3.3 V with a +2%/-2% margin of error as specified in data sheet, using a Voltmeter 2. Send a 6 V input voltage through the voltage regulator and measure the output to see if it is 5 V with +4%/-4% margin of error as specified in the data sheet, using a Voltmeter

<u>Microcontroller</u>

We are going to use an ATmega328p microcontroller in order to manage storing data and data transfer, as well has handle signals from various control buttons (raise, call, power buttons). The microcontroller will be programmed via UART and will read the signals from the RF module via SPI.

Requirements	**Verification**
1. Should be able to store data (see Memory Section for more details) 2. Should process user signals and update game states nearly instantly, ideally under ~0.5 seconds	1. Send simple data and see if it is able to reproduce it when it called 2. Press a button on the player device/central hub and see if it produces the desired action (e.g. FOLD, CALL, RAISE), under the desired time.

Power Button

This will simply serve as a switch in our circuit to disconnect the voltage source from the rest of the components.

Requirements	**Verification**
1. Switch successfully turns on or off the console	1. Test the switch for a set number of trials and considered success if all trials work

Call Button

This button will only be on the player device, and it will be used to send a signal to the central hub display via the RF module. The central hub display will then automatically update the call amount to the community pot.

Requirements	**Verification**
1. When CALL button is pressed a HIGH signal must be received in the output, else it will go to 0.	1. Using a Multi-meter we will check if the current is >0 Amps when the button is pressed and 0 if it is not pressed.
2. The microcontroller on player device should receive a signal from the CALL button > 95% of the time and store it to be sent to the central hub by the RF transceiver	2. Press CALL button 100 times and see if a signal is received at least 96 times

Raise Button

This button will be only on the player device, and it will be used to send a signal to transfer the data from the player device microcontroller to the central display microcontroller via the RF module.

Requirements	**Verification**
1. When RAISE button is pressed a HIGH signal must be received in the output, else it will go to 0. 2. The microcontroller on player device should receive a signal from the RAISE button > 95% of the time and store it to be sent to the central hub by the RF transceiver	1. Using a Multi-meter we will check if the current is >0 Amps when the button is pressed and 0 if it is not pressed. 2. Press RAISE button 100 times and see if a signal is received at least 96 times

<u>Up & Down Button</u>

These buttons will only be on the player devices, and they will be used to control how much the player wants to increase or decrease the current bet.

Requirements	**Verification**
1. When UP button is pressed a HIGH signal must be received in the output, else it will go to 0. 2. The microcontroller on player device should receive a signal from the UP button > 95% of the time and store it to be sent to the central hub by the RF transceiver	1. Using a Multi-meter we will check if the current is >0 Amps when the button is pressed and 0 if it is not pressed. 2. Press UP button 100 times and see if a signal is received at least 96 times
1. When DOWN button is pressed a HIGH signal must be received in the output, else it will go to 0.	1. Using a Multi-meter we will check if the current is >0 Amps when the button is pressed and 0 if it is not pressed.

2. The microcontroller on player device should receive a signal from the DOWN button > 95% of the time and store it to be sent to the central hub by the RF transceiver	2. Press DOWN button 100 times and see if a signal is received at least 96 times

<u>Fold Button</u>

This button will only be on the player device, and it will be used to send a signal to the central hub display via the RF module. The player remote will be disabled for the rest of the round, where round is when N-1 players have folded.

Requirements	**Verification**
1. When FOLD button is pressed a HIGH signal must be received in the output, else it will go to 0.	1. Using a Multi-meter we will check if the current is >0 Amps when the button is pressed and 0 if it is not pressed.
2. The microcontroller on player device should receive a signal from the FOLD button > 95% of the time and store it to be sent to the central hub by the RF transceiver	2. Press FOLD button 100 times and see if a signal is received at least 96 times

<u>Central Hub Display</u>

The central display device will contain three LCD displays, with two of them containing the current players, their current chip amounts. The third one will contain the community pot,the current bet, what the current player's action is, and who is going. The data being displayed on the LCD screens will be coming from the microcontroller storage.

Requirements	**Verification**
1. Should be able to display characters with enough brightness, so user does not strain to read the displayed information; must be visible to read ~5 ft radius	1. Try various potentiometer settings until reaching a brightness that

Player Device Display

The player device will have one LCD screen so that the user can see how much money he/she has left and how much money he/she wants to raise for the current turn. The data being displayed on the LCD screen will be coming from the microcontroller storage.

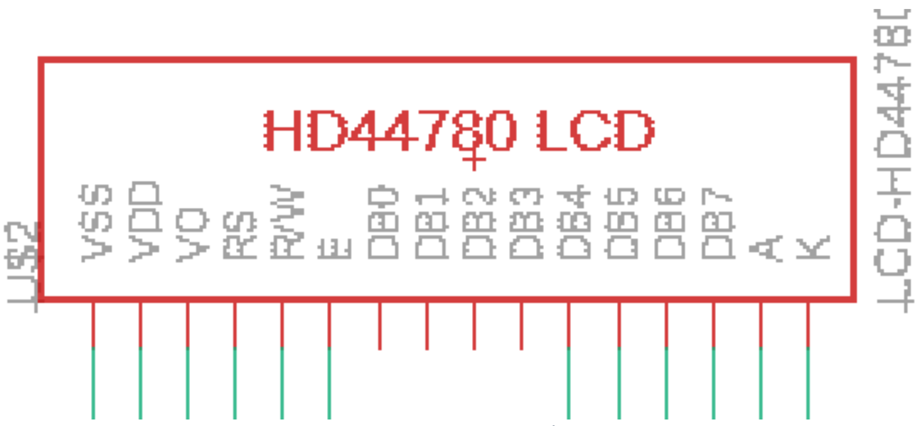


Figure 6: LCD 16x2 Display

Requirements	**Verification**
1. Should be able to display characters with enough brightness, so user does not strain to read the displayed information; should be visible ~2 feet away.	1. Try various potentiometer settings until reaching a brightness that

RFID Tag

The player device will contain the RFID tag that will be attached to each player device/remote to allow a player to successfully join the game and begin the mode of communication between the central hub and the player device. To join the game the player must scan his or her RFID on the back of the player device on the RFID reader on the central hub.

Requirements	Verification
1. The tag should properly be attached to each player device and be accessible to be read by the RFID reader 2. Should be unobtrusive	1. Test the RFID tag and reader by attaching it to an LED and having it light up if the RFID is successfully read; test this with all the RFIDs we will have 2. Ensure the RFID Tag is barely visible, yet functional on the player device

<u>RF Transceiver</u>

The RF Transceiver is necessary for communication between the Player Device and Central Hub, as we must pass data to be displayed almost every I/O action. The model we will be using is the NRF24L01. This has a transmit power of 12mA, and operates at a 1.9 - 3.6 V range. The frequency bandwidth in which this transceiver operates is 2.4 GHz. The amount of baud (how many times the signal changes per second) ranges from 250 Kbps to 2 Mbps.

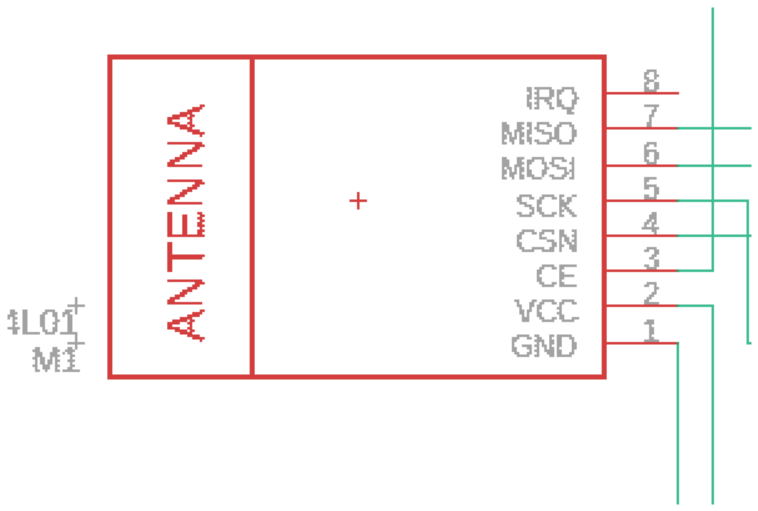


Figure 7: NRF24L01 Transceiver

Requirements	**Verification**
1. The transfer of data from the Player Device transceiver and Central Hub transceiver (and vice versa), should consistently be correct and instant. 2. The transfer of data should occur properly within a radius of at least 5 feet	1. Send simple data between the Central Hub and Player Device, both hard-coded and dynamically, to be displayed on the LCD display. Do this several 100 times to ensure the communication channels work to our specified speed (~250 Kbps) and with almost 100% accuracy. 2. Go to various points around the device (<= 5 feet) and see if the transfer of data is still successful between player device and central hub

RFID Reader

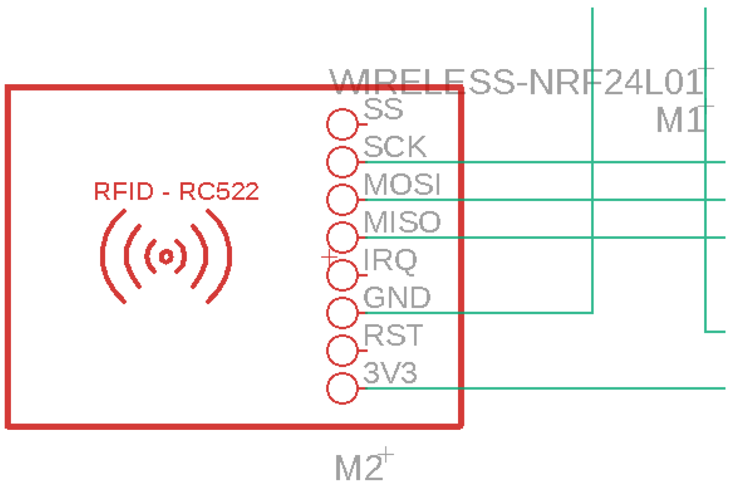
The RFID Reader serves the purpose of reading the individual player's RFID tags and allows them to enter the game. This will only be on the Central Hub, and will only be used once by each player then the software will keep the rotation/turns of each player going.

Requirements	**Verification**
1. Ensure the Player Devices can be read into the game through the use of the RFID reader	1. Test each individual RFID tag on the reader and ensure the reader is able to identify each separately and open a method of communication between the Player Device and Central Hub. If successful, all RFID tags will be saved on the Central Hub's microcontroller
2. Ensure the RFID tag is read on touch (less than 1 Inch distance)	2. Move the RFID tag closer and farther from the RFID Reader and see at what distance the reader recognizes the RFID Tag
3. Ensure there is at least a 99% accuracy of the tag being read	3. Tap the RFID tag on the reader 100 times and if it is successful 99 times, it is successful

Figure 8:

NRF24L01

Transceiver



Circuit Schematics

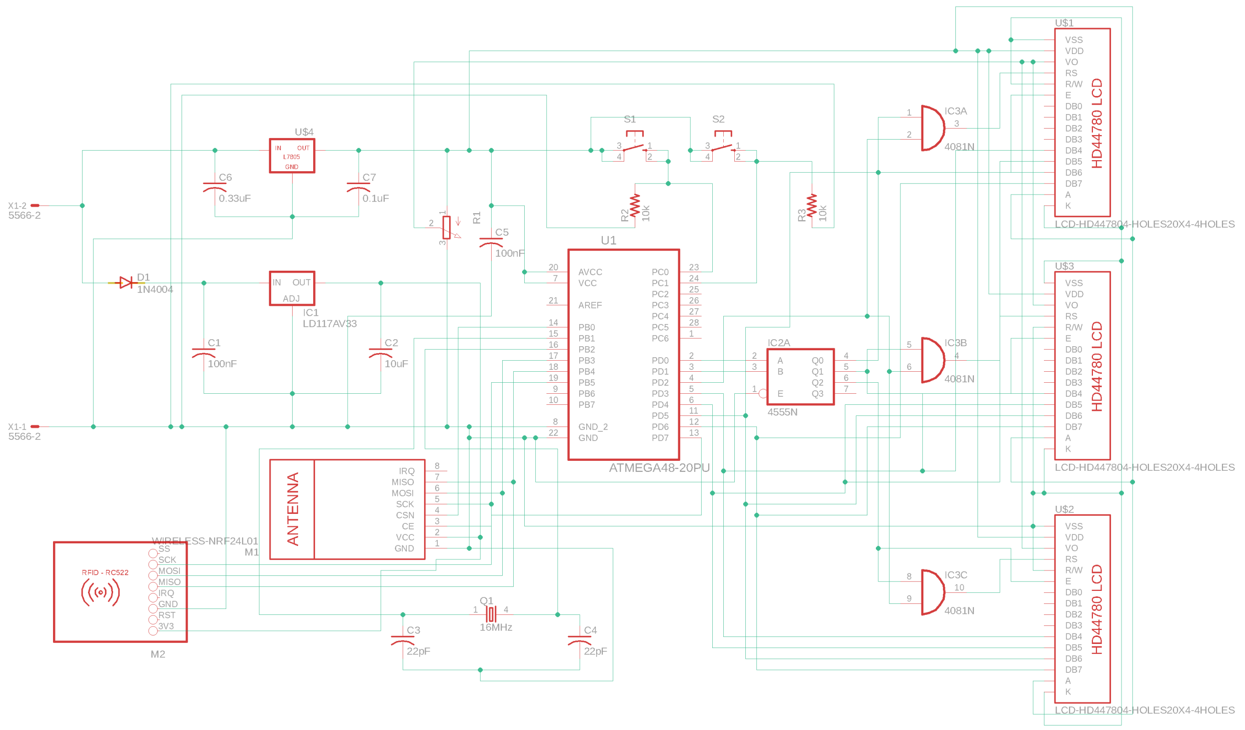


Figure 9: Central Hub Schematic

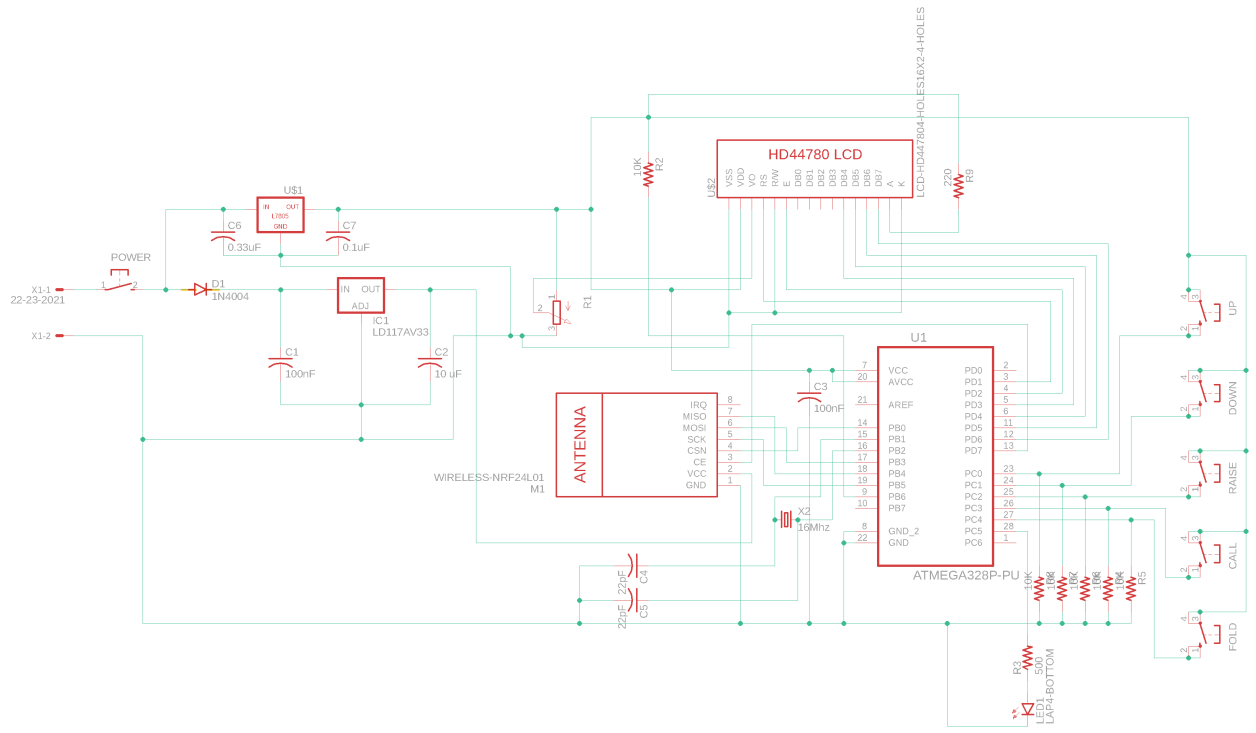


Figure 10: Player Remote Schematic

Circuit Elements

- Capacitors
  - The capacitors in the circuit primarily serve the purpose of suppressing high frequency noise from sources and loads
- Resistors
  - The resistors in the circuit are used to ensure that the current entering the microcontroller is regulated. Since this is one of the most critical components to the device, we must ensure it doesn't fry or get damaged.
- Diodes
  - The diodes are used to ensure that the polarity of the current is consistent throughout the circuit and is not being reversed, which could cause circuit bugs
- Potentiometer
  - The potentiometer is used to control the brightness of the LCD display. Our goal is to set this at a certain value and keep it consistent so that it doesn't change throughout the game.
- AND Gates
  - The AND gates are primarily used to ensure that only certain LCD displays are getting the data at once. We don't want all the displays to get the data at once because we want to display different items, so this will control the logic for that

Flow Diagrams

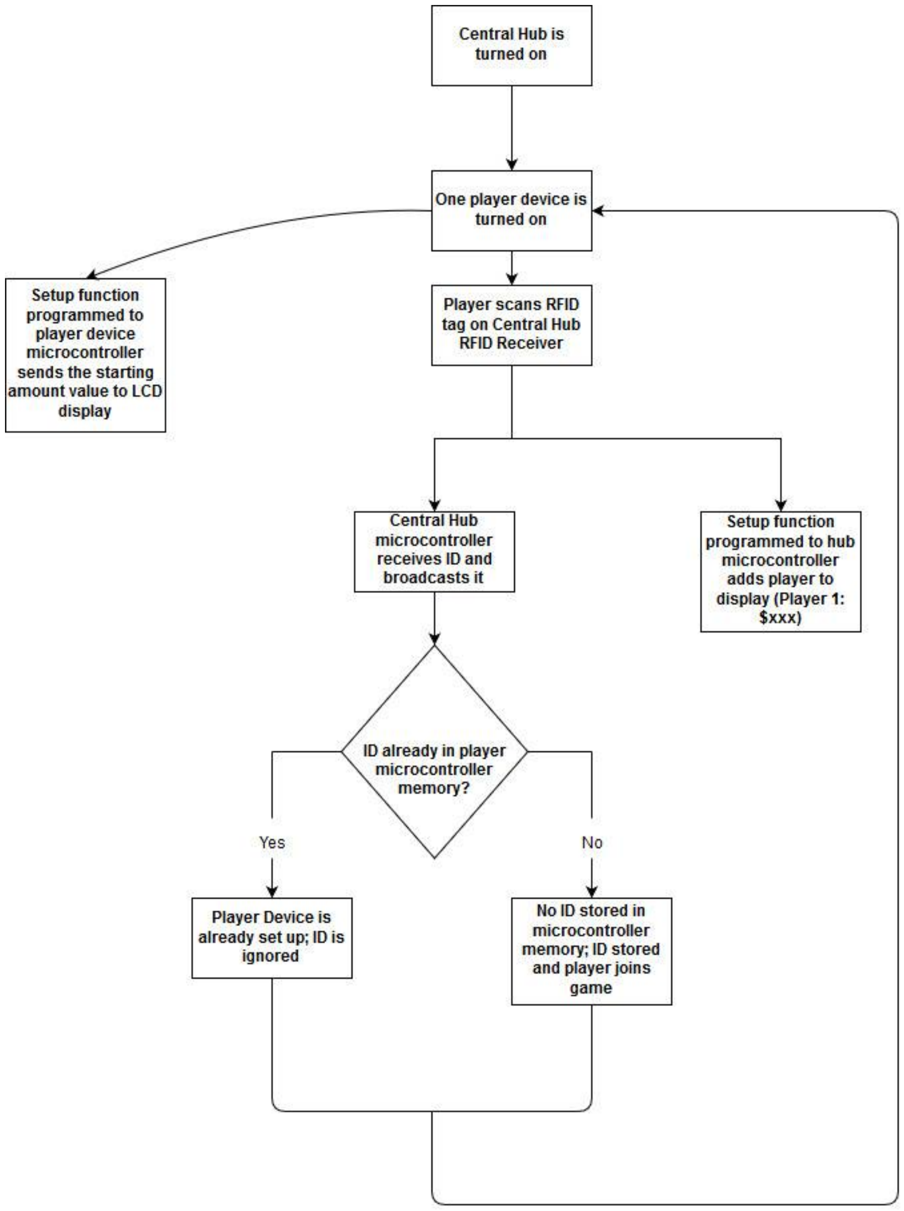

Central Hub is turned on
One player device is turned on
Setup function programmed to player device microcontroller sends the starting amount value to LCD display
Player scans RFID tag on Central Hub RFID Receiver
Central Hub microcontroller receives ID and broadcasts it
Setup function programmed to hub microcontroller adds player to display (Player 1: $xxx)
ID already in player microcontroller memory?
Yes
No
Player Device is already set up; ID is ignored
No ID stored in microcontroller memory; ID stored and player joins game

Figure 11: Game Setup Flow

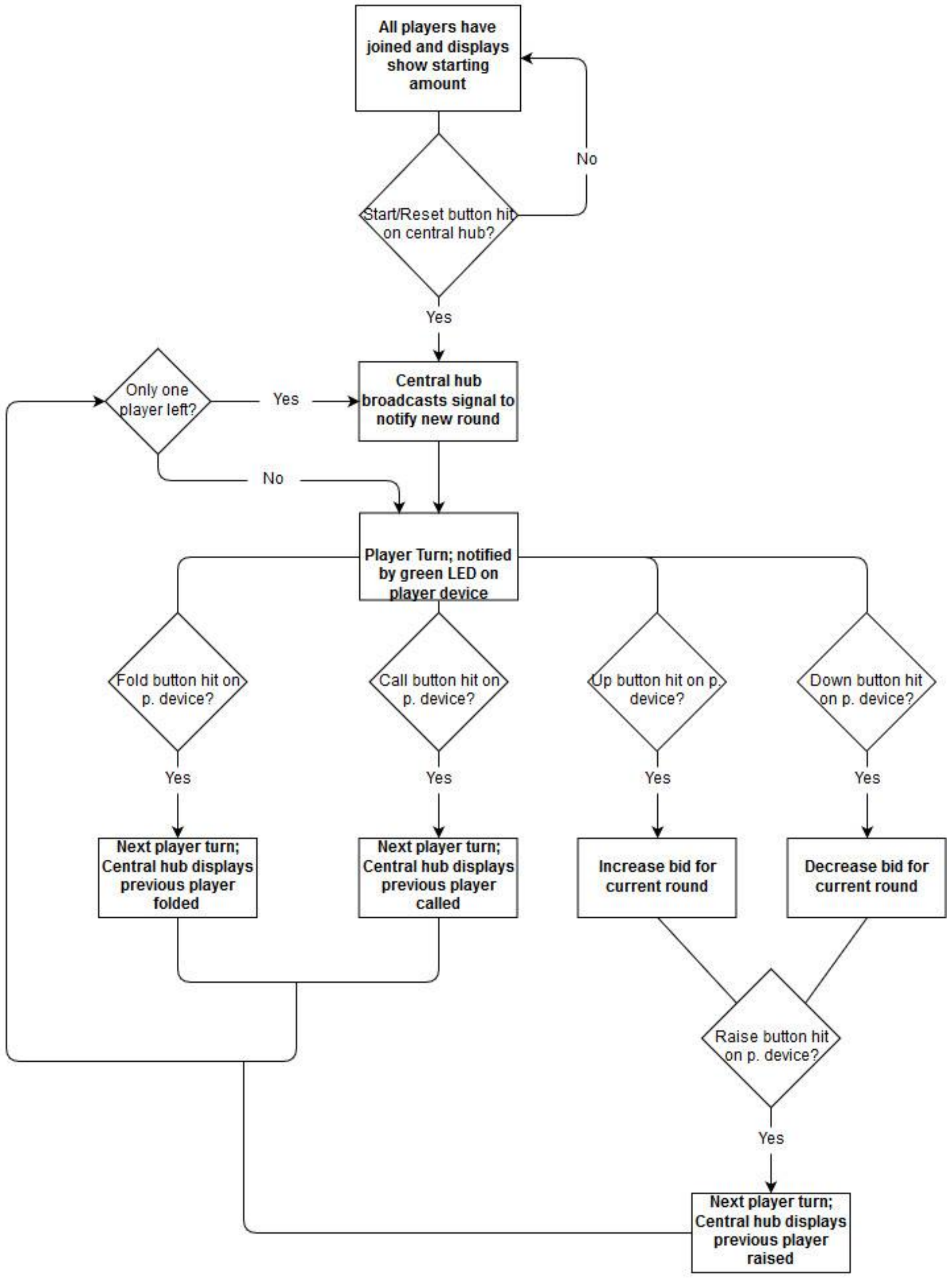

All players have joined and displays show starting amount
Start/Reset button hit on central hub?
No
Yes
Only one player left?
Yes
No
Central hub broadcasts signal to notify new round
Player Turn; notified by green LED on player device
Fold button hit on p. device?
Call button hit on p. device?
Up button hit on p. device?
Down button hit on p. device?
Yes
Yes
Yes
Yes
Next player turn; Central hub displays previous player folded
Next player turn; Central hub displays previous player called
Increase bid for current round
Decrease bid for current round
Raise button hit on p. device?
Yes
Next player turn; Central hub displays previous player raised

Figure 12: Game State Flow

Flow Diagram Brief

The flow diagrams describe how the game progresses during setup and how it works during the actual gameplay. The setup and game state is an iterative process that keeps looping until players can begin playing the game. If a player was to join once the game has already begun, the game setup flow will have to be run again but all that changes with the game state is that another player joins the fray.

Memory

The ATmega328p comes with 32k bytes Flash memory (0.5k is used for the bootloader, 2k bytes of SRAM, and 1k byte of EEPROM [8]. Flash and EEPROM are non-volatile, therefore they persist even if power is suddenly turned off [8]. In our game, some player could accidentally turn off their device, or the device could run out of battery. In this case, we still want to store all the relevant information on the microcontroller, so that we can access it after the power is on again.

On the central hub, the important values to keep track off are everyone's money and their bids, the amount in the community pot, whose turn is it, and who has folded. The good thing about the data we are storing is that we can generally assume that the casual player does not play in a game with over $1000, so our data does not consume a lot of space. The three players will have a max of 3 digits for both their current money and bid, which is a total of 18 bits needed for those values among 3 players. We will also have three bits for the community pot, safely assuming that players will not exceed the amount. All players will be identified by a number, so keeping track of whose turn it is will only be one digit. Our prototype design will be for three players, so there can be a max two people that fold (according to our game logic) and therefore two bits for that as well. This brings us to a total of 18+3+1+2 = 24 bits that need to be stored. Even if we store each number as a character, we have 192 bytes minimum that need to be stored in the EEPROM out of 1k bytes. Additionally, the microcontroller will need to store the 96 bits of data for the RFID tag, and with three players, this is 288 bits of data, which is 36 bytes of data. In total, the microcontroller needs to store about 228 bytes of data to persist when the power turns off. This leaves a lot of room for expansions and even more data storage.

For each player device, the value that need to be stored is like what the central hub has to store, but it only needs to keep track of its individual money count, bid, whether it's the current player's turn and the RFID tag associated to the device. Therefore, there is significantly less than 228 bytes of data storage in the EEPROM needed to persist, which allows for more expansion for other data to be stored (personal names, etc.)

Tolerance Analysis

One variable factor that will determine our project success is if the device and central hub battery life lasts the duration of the game. One of our main tasks to handle is to ensure that the current power solution we have now will be sufficient in any given situation. Therefore, we are testing the tolerance of our battery pack solution, as well as the power consumption of the components of our circuit, to ensure that we are allowing enough runtime for the game. The max duration for game time we are going to set is 2.5 hours.

In our circuit, we have our 6V battery pack connected in parallel with two voltage regulators in order to bring it down to 5V and 3.3V respectively. Since our initial voltage is fairly close in value to the dropped down voltages, the regulator efficiency will be fairly good, and less heat will be dissipated. If our initial voltage was larger, we may have to consider switch converters (such as the buck converter) which dissipates less heat overall. In terms of battery life, the microcontroller is going to require the most power as it provides and receives signals from all of the other components in our device. Using various parameters provided in the ATmega328p data sheet, we did a tolerance analysis on battery life in order to see if the various parameters still enabled the 2.5-hour gameplay that we intend to aim for [5].

$$Efficiency = \frac{Power_{Out}}{Power_{In}} = \frac{Voltage_{out}}{Voltage_{in}}$$

Equation 1

$$Runtime = Charge_{AA} * \frac{Efficiency}{load_{powerdraw}}$$

Equation 2

The ATmega328p has an internal oscillator of 8 MHz. In our scenario, we can provide either 3.3V or 5V to the microcontroller, but we will be providing 5V to the microcontroller for better performance. Using the parameters of 8 MHz and Vcc = 5V for the ATmega328p, the voltage regulator efficiency is around 75%. The microcontroller, for these parameters, draws about 6.3mA. We can then use the above equations to get our results [5].

$$\frac{5V}{6V} = 0.83333$$

$$2500mAH * \frac{0.83333}{6.93mA} = 300.624\ hours$$

Although the microcontroller has an internal oscillator, we will be using a 16 MHz crystal oscillator to provide the near-constant frequency to the microcontroller. According to the ATmega328p data sheet, for an input voltage of 5V and a frequency of 16 MHz, the microcontroller draws about 12mA [5].

$$\frac{5V}{6V} = 0.83333$$

$$2500mAH * \frac{0.83333}{12mA} = 173.61\ hours$$

We will be doing additional testing with the individual components integrated in our device, and we will re-do an analysis to ensure that our battery life still remains above the 2.5-hour time we have allocated.

3. Cost & Schedule

Cost

Since we are expecting to finish the entire project during the duration of the semester, we don't need to account for partial work calculations. We estimate our development costs to be \$30/hour, 15 hours/week for 3 people. Therefore, our total development costs will be:

$$3 * \frac{\$30}{hr} * \frac{15\,hrs}{week} * 16\,weeks = \$21{,}600$$

Part	Cost (Individual)	Cost (Project)
LCD Display (HD44780)	\$2.71 (16x2) \$5.61 (20x4)	\$19.16
Microcontroller (ATMega328P)	\$4.30	\$17.20
RF Transceiver (NRF24L01)	\$1.75	\$7.00
RFID Reader (RC522)	\$12.99	\$12.99
Voltage Regulator	\$0.95 (L7805) \$0.54 (LD1117AV33)	\$5.96
Logic Gates (SN74HC08AN)	\$0.44	\$1.76
Decoder (CD4555BEE4)	\$0.46	\$1.84
Resistors/Capacitors Pack	\$15.90	\$15.90
Battery Pack (4xAA)	\$1.95	\$7.80
Push Buttons	\$0.25	\$4.00
Potentiometer	\$0.95	\$3.80
16 MHz Crystal	\$0.95	\$3.80
Total		\$101.21

We are planning on building only one unit, therefore our total development costs including work hours is \$21,701.21

Schedule

Week	Anand	Umang	Varun
2/25/19	Requirements and verifications	Remote design review	Central hub design review
3/4/19	Soldering training Purchase of components	Soldering training Initial framework	Soldering training Initial framework
3/11/19	PCB analysis and testing	Remote PCB construction	Central hub PCB construction
3/18/19	**Spring Break**		
3/25/19	Revisions/edits to PCB design		
4/1/19	Test microcontroller capabilities	Construct software scheme	Construct and test RF module
4/8/19	Demonstration and testing of individual modules	Construct display module	Construct I/O module
4/15/19	Compile all modules for demonstration		
4/22/19	Final edits		
4/29/19	Final presentation/demo Final Report		

4. Ethics & Safety

With any electrically heavy product, safety concerns are quite apparent. In our product specifically, the biggest safety concerns we have are a potential power overload and potential moisture/water short circuits.

Since we have multiple AA batteries (up to 4) powering our devices, it is possible that we could have voltage overload, which will eventually lead to a power overload, causing a potential explosion. We are attempting to regulate the voltage outputted by these batteries using both using a voltage regulator and the resistive capabilities available in the PCBs.

We are also concerned with the potential for moisture/water to get within the devices, whether intentionally or unintentionally. If water gets into the device(s) it will inevitably lead to a short-circuit as the components will be damaged. Since the product is ideally to be used indoors, or in an area where water cannot get into the device, a normal casing should be enough.

The goal of our product is to bring a fun, and fair system to the common or even professional poker player, by eliminating chips from the game and allowing for a fair system of money representation. By doing so we eliminate potential cheating from the game which satisfies IEEE Code of Ethics, #2: “to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist” [7]. Thus, this product serves to not only allow for more players to participate at once, but also allows for a cheat-free system by getting rid of the need for physical chips.

While our product is meant to be harmless, as poker is simply a card game, it may aggravate certain illnesses such as a gambling addiction. This is in violation of IEEE Code of Ethics, #9: “to avoid injuring others, their property, reputation, or employment by false or malicious action”[7]. With the potential ease of this product, there is a chance of further aggravation to a person’s already harmful addiction. We do not have a means to solve such a health issue - we are assuming that players are playing responsibly and are careful with their money.

5. References

[1] N/A, N/A. “Monopoly Game: Ultimate Banking Edition” *Hasbro*, 26 Nov. 2016, https://monopoly.hasbro.com/en-us/product/monopoly-game-ultimate-banking-edition:292A13F3-5056-9047-F5EC-64DBA290A02B.

[2] N/A, N/A. “What Are ESC, UBEC and BEC.” *Oscar Liang*, 26 Nov. 2016, oscarliang.com/what-is-esc-uber-bec-quadcopter/.

[3] Publications, IEEE. “IEEE Code of Ethics.” *IEEE - Advancing Technology for Humanity*, www.ieee.org/about/corporate/governance/p7-8.html.

[4] ATMega, ATMega. "ATMega628." *Sparkfun*. N.p., 2016. Web. 21 Feb. 2019.

[5] Igor, Ardunio. "Arduino's ATMega328 Power Consumption." *Gadget Makers' Blog*. N.p., 14 Dec. 2013. Web. 22 Feb. 2019.

[6] Julius, Bob. "Powerdis." *Low Pass Filters*. N.p., 2015. Web. 22 Feb. 2019.
N/A, N/A. "What Are ESC, UBEC and BEC." *Oscar Liang*. N.p., 26 Nov. 2016. Web.

[7] Publications, IEEE. "IEEE Code of Ethics." *IEEE - Advancing Technology for Humanity*. N.p., n.d. Web.

[8] QU1500_US_UL1, QU1500_US_UL1. "QU1500_US_UL1." *D2ei442zrkqy2u.cloudfront*. N.p., 2015. Web. 21 Feb. 2019.