Stepmania

Custom Project Final Report

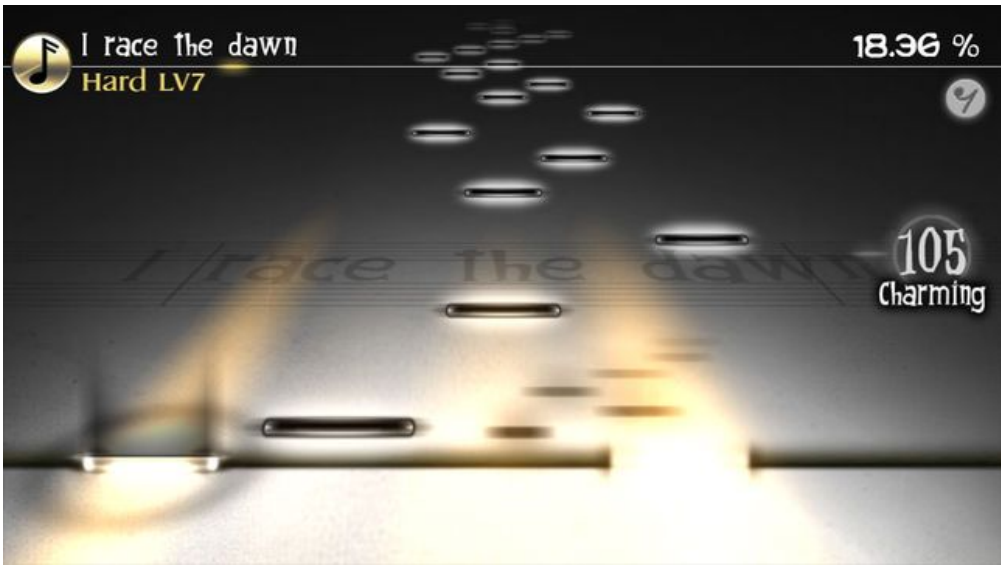
Spring 2018

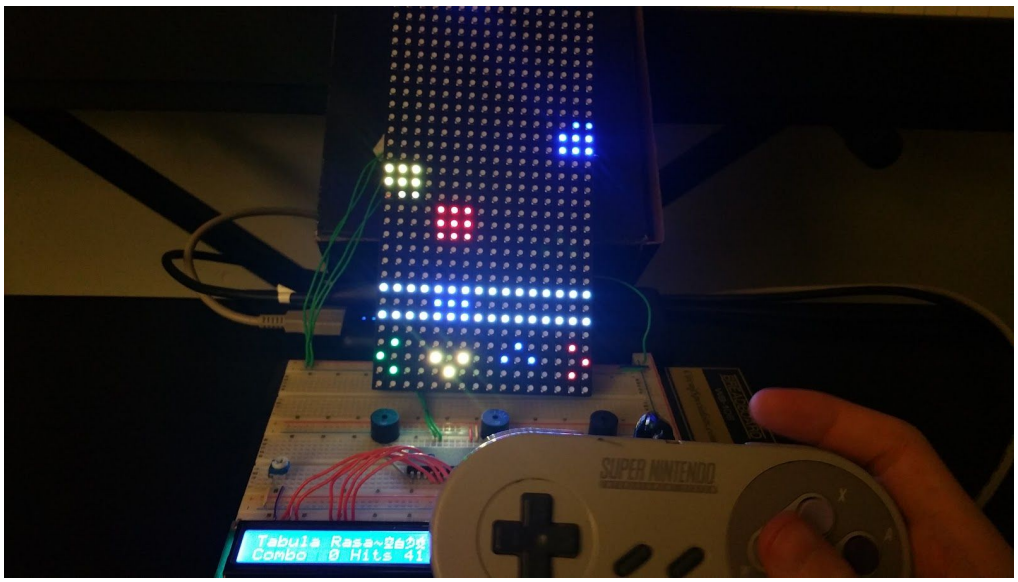
Jonathan Oaks

Table of Contents

Introduction

Stepmania is a rhythm game where sequences of arrows scroll across the screen while the player taps on the corresponding button in time with the music. Players aim to hit the sequences accurately to maintain a combo meter and score points for hitting notes.





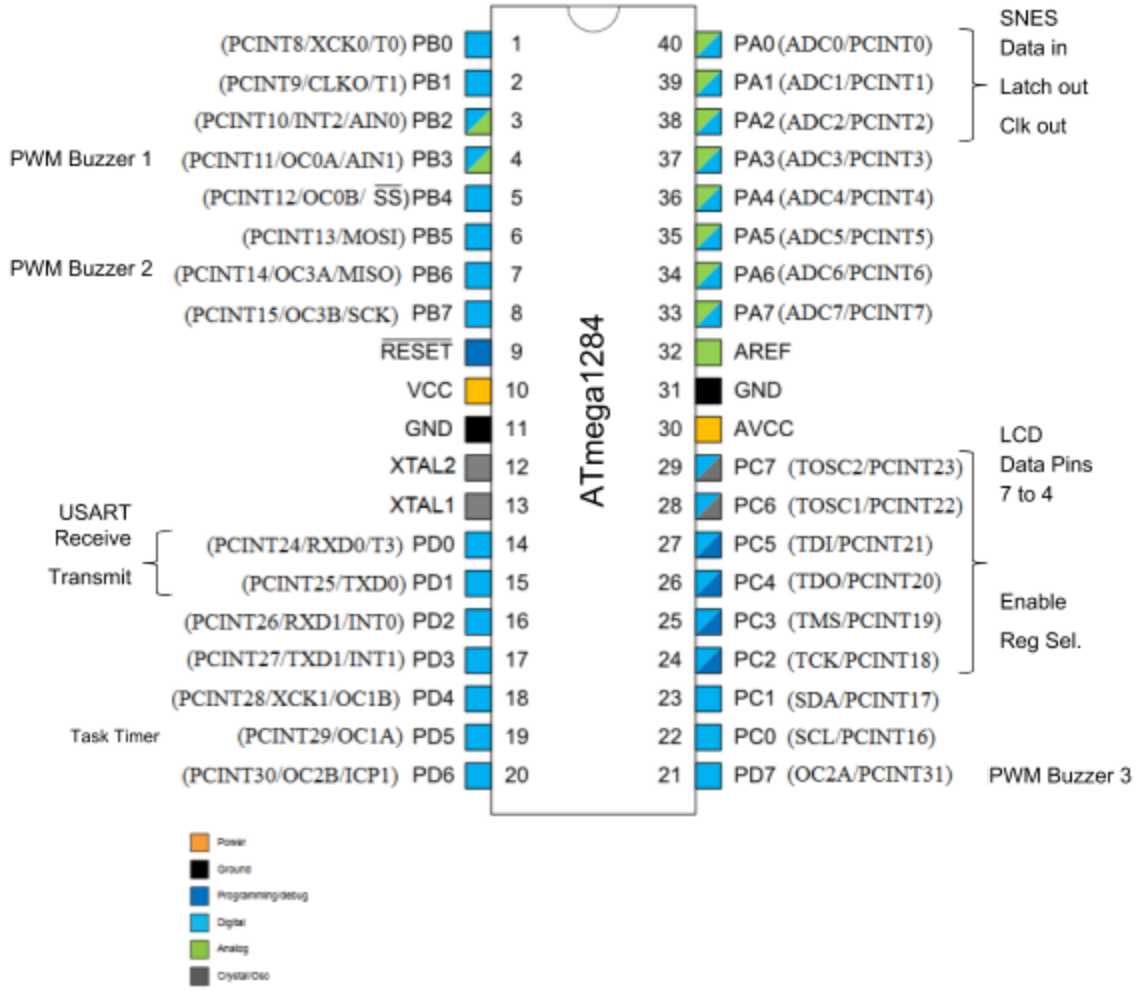
Youtube Link

https://www.youtube.com/watch?v=CP1OC6pawgQ

Hardware

- ATMega1284 Microcontroller
- SNES Controller
- 16x2 LCD Display
- Piezo Buzzer x3
- 16x32 RGB LED Matrix
- Arduino Uno (ATMega328p)

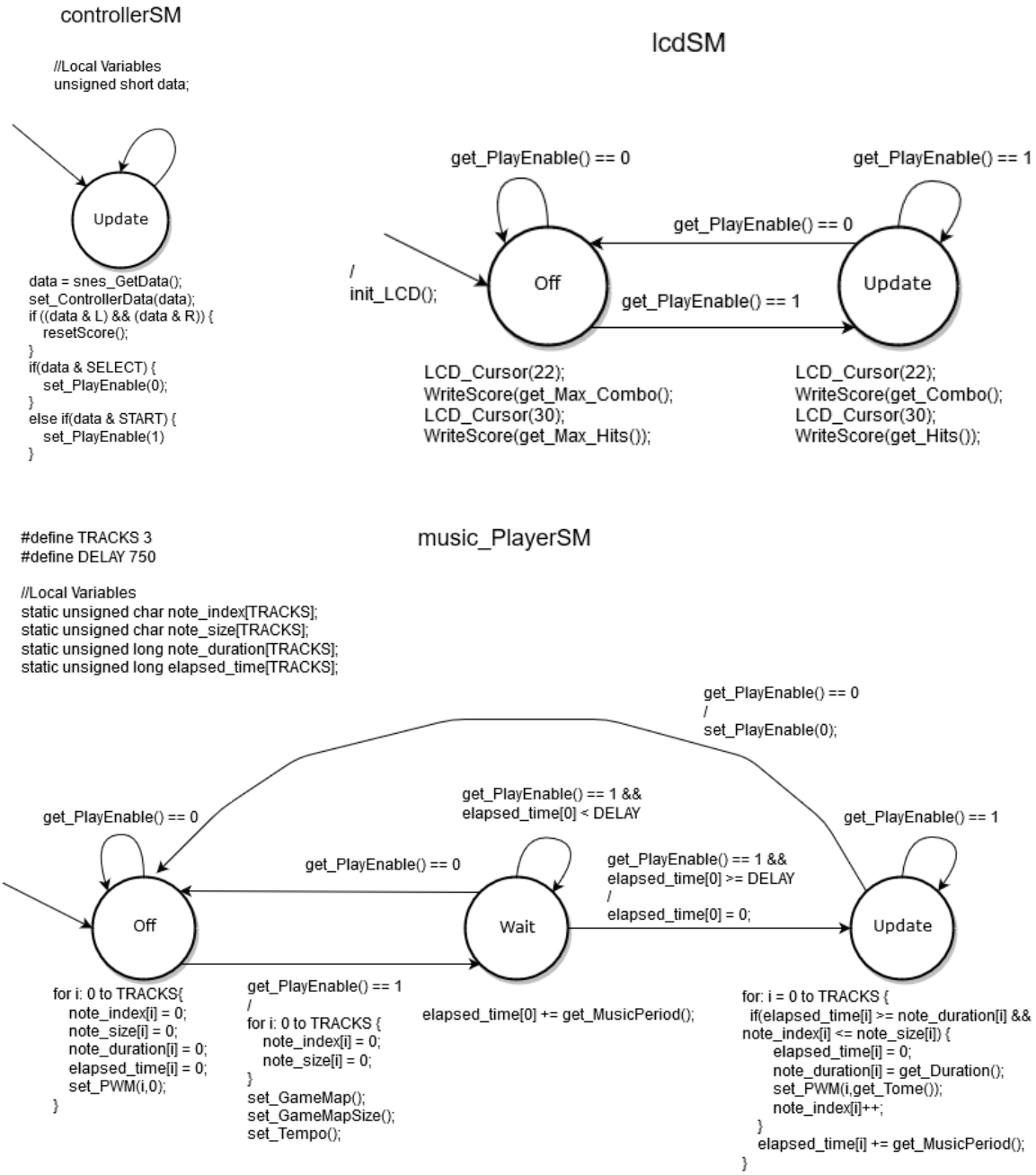
ATmega1284 Pin Layout



Software/State Machines

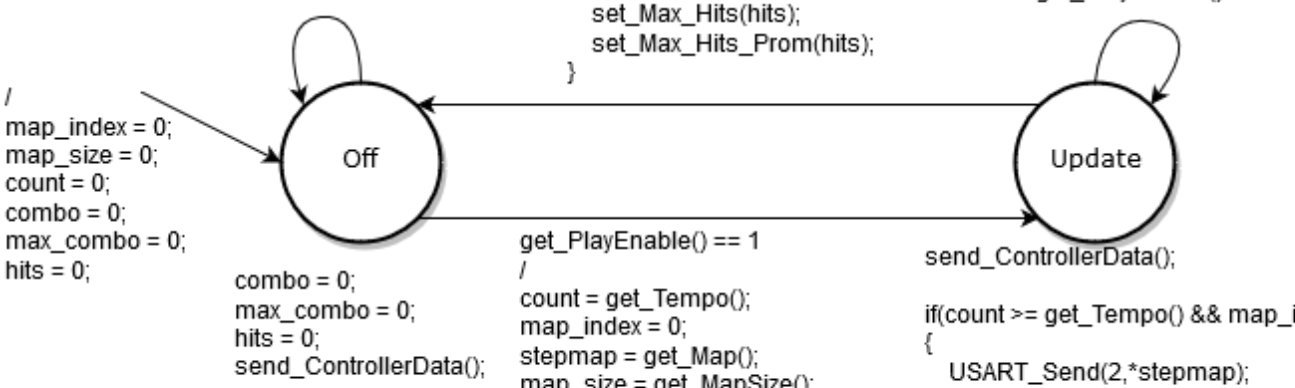
For detailed code, visit the github page for this project: https://github.com/nailcliper/CS120B-Stepmania

ATMega1284



usartSM

```
//Local Variables
static unsigned char *stepmap;
static unsigned short *map_index;
static unsigned short map_size;
static unsigned char count;
static unsigned char combo;
static unsigned char max_combo;
static unsigned char hits;
```




```
/
map_index = 0;
map_size = 0;
count = 0;
combo = 0;
max_combo = 0;
hits = 0;
```

```
get_PlayEnable() == 0
/
USART_Send(3,0);
combo = 0;
max_combo = 0;
hits = 0;
send_ControllerData();
```

```
get_PlayEnable() == 0
/
USART_Send(3,0);
if(max_combo > get_Max_Combo()) {
   set_Max_Combo(max_combo);
   set_Max_Combo_Prom(max_combo);
}
if(hits > get_Max_Hits()) {
   set_Max_Hits(hits);
   set_Max_Hits_Prom(hits);
}
```

```
get_PlayEnable() == 1
/
count = get_Tempo();
map_index = 0;
stepmap = get_Map();
map_size = get_MapSize();
sendMapBuffer();

USART_Send(3,1);
USART_Send(4,get_Tempo());
combo = 0;
set_Combo(combo);
hit = 0;
set_hits(hits);
```

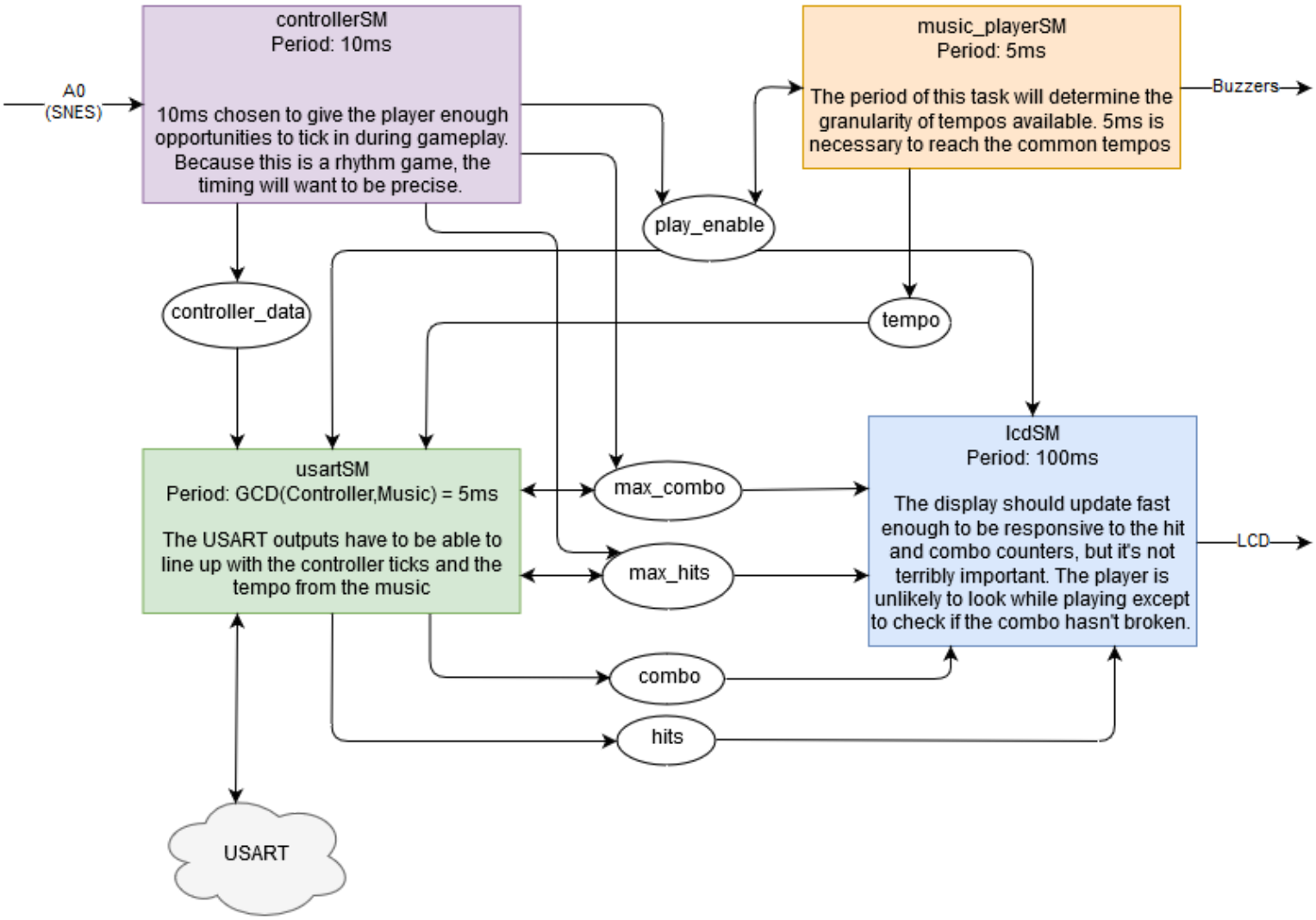
```
get_PlayEnable() == 1
send_ControllerData();

if(count >= get_Tempo() && map_index < map_size)
{
   USART_Send(2,*stepmap);
   stepmap++;
   count = 0;
   map_index++;
}

while(USART_HasReceived()) {
   unsigned char data = USART_Receive();
   if(data == HIT) {
      hits++:
      set_Hits(hits);
      combo++;
      set_Combo(combo);
      if(combo > max_combo) {
         max_combo = combo;
       }
   }
   else if(datta == MISS) {
      combo = 0;
      set_Combo(combo);
   }
}
```

ATMega1284 Task Diagram

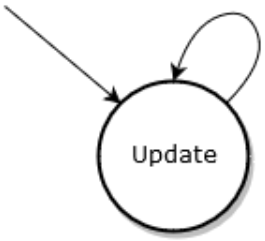
System Period: 5 ms



Arduino

inputSM

```
//Local Variables
unsigned char rx[2];
```



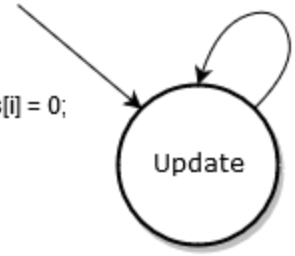

```
while(UART_available() {
   UART_Read(rx,2);
   switch(rx[0]) {
      case 0x01: // Button Data
         set_ControllerData(rx[1]);
         break;
      case 0x02: // Map Data
         mapList.add(rx[1]);
         break;
      case 0x03: // Enable Data
         set_PlayEnable(rx[1]);
         break;
      case 0x03: // Tempo
         set_Tempo(rx[1]);
         break;
      default: break;
   }
}
```

buttonSM

```
//Global Variables
unsigned char button_states[4][

//Local Variables
unsigned char controller_data;
unsigned char button_state;
```

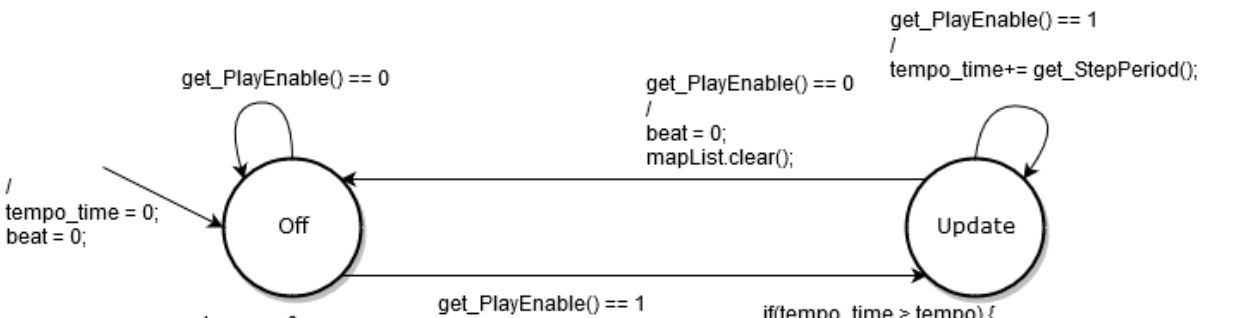
```
/
for i = 0 to 4 {
   button_states[i] = 0;
}
```



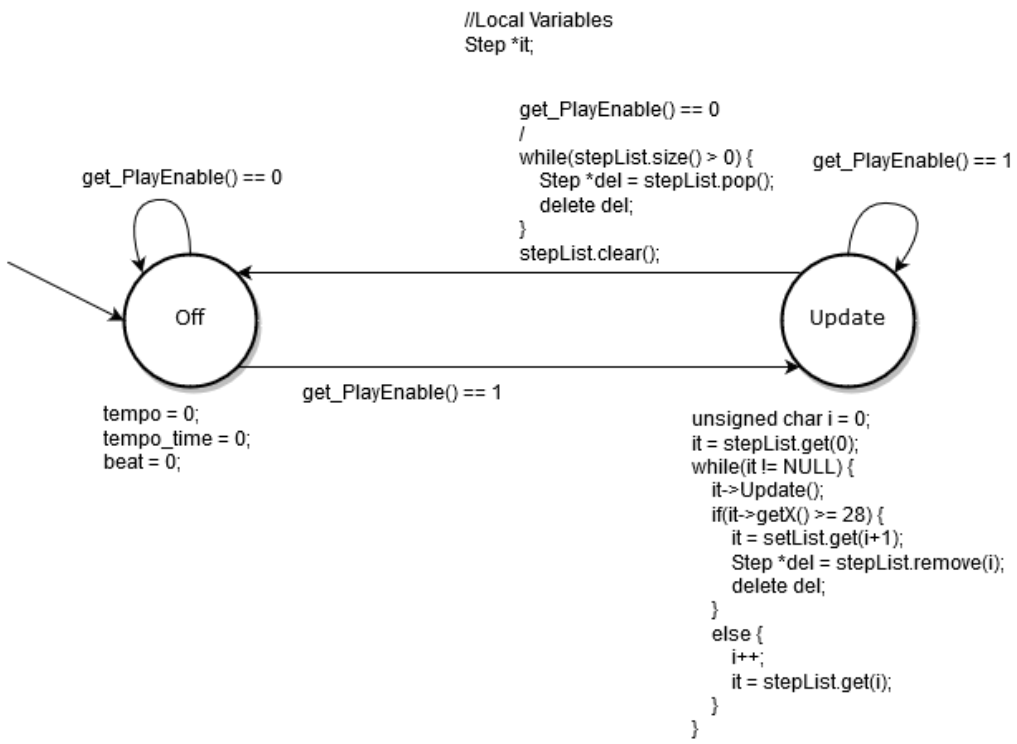

```
controller_data = get_ControllerData();
DrawButtons();
```

stepSM

```
//Local Variables
static unsigned short tempo;
static unsigned char tempo_time;
static unsigned char beat;
```

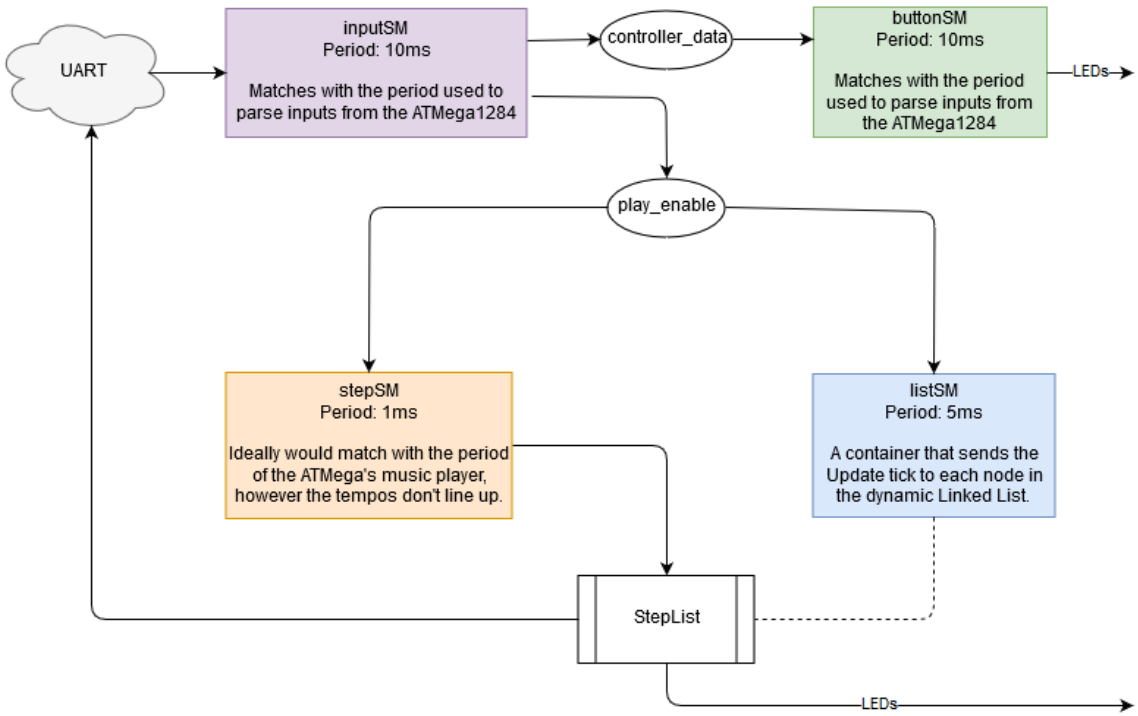


listSM



Arduino Task Diagram

System Period: 1 ms



Complexities

- Configuring 16x2 LCD Display in 4-bit mode
- Programming custom characters onto the LCD for display
- Use of SNES Controller for inputs
- Use of EEPROM for storing high scores
- Configuring three available timers (one 16-bit and two 8-bit) to be capable of playing all 88 notes in the range of A0 to C8.
- Communication with an Arduino using USART

Known Bugs and Shortcomings

- Some state machines write to the same shared variables, which could cause undefined behavior. In testing, the behavior was not a significant problem. Holding the start button when the song finishes starts the song again, but the scores are saved. Holding the reset scores buttons when the song exits might not reset the scores until the buttons are pressed again. I don't consider these fatal enough flaws to consider fixing at the moment.
- The tempo is not behaving as expected. In initial testing, the formula 60000 / (prescaler * original tempo) would produce an accurate duration in milliseconds for the notes to match the original tempo in beats per minute with a beat granularity decided by the prescaler. As development went on, the tempo would drift slower. Some of the function calls may be delaying the count. This problem carried on into the Arduino, where the tempo passed in would not produce a sequence of notes in line with the supposed tempo. As a result, the tempo values are hardcoded in.
- The periods for the state machines might be too low. This is definitely the case for the Arduino.
- The initial power-on may not initialize correctly. Because the Arduino has power and ground lines connecting into the breadboard, it can partially power the ATMega if it is provided power first, causing initialization to fail on the ATMega. Even in a normal startup, the Arduino may not communicate properly with the ATMega and remain stagnant. This can be tested by pressing the arrow/buttons on the SNES controller. If the arrows at the bottom of the Matrix don't light up, then the Arduino's reset button needs to be pressed until it does.
- To make it more portable, I power the device by plugging into a USB power bank, however, the order of the devices changes the amount of power they receive. If the order isn't correct, the Arduino doesn't receive enough power and it takes longer to update. The result is that the lights don't appear as strongly and the tempo lags behind the song.

Future Work

In the future, I'd like to be able to play different songs. The method for creating maps is relatively simple, however the issue involving tempo slightly complicates things. Not all tempos are available and the mathematical formula for converting tempo in beats per minute to milliseconds per beat doesn't produce an accurate result.

I would also like to include the held-note type, where the player must hold the note for some duration. I had attempted to do this originally, but it seems that I've hit the limit for RAM on the Arduino., which is where the notes are dynamically created.