The LazerBoy Entertainment System

Brian Sumner, Kamil Adylov, Phi Huynh.
In association with Salim Lakhani, PhD.
CU Denver CSCI 4287 - Spring 2019

Special thanks to Donald Vukovic from TinkerMill, for technical consulting and sourcing of hardware components.

Project Description

The LazerBoy Entertainment System is a game platform which allows players to shoot lasers at photosensitive targets and enjoy a variety of game modes.

https://github.com/lazerboy-entertainment-system

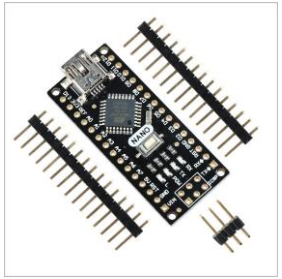
Disclaimer - It's just a toy!

The LazerGun M9B2 is a harmless instrument that is not intended to cause fear in or assault to another person.

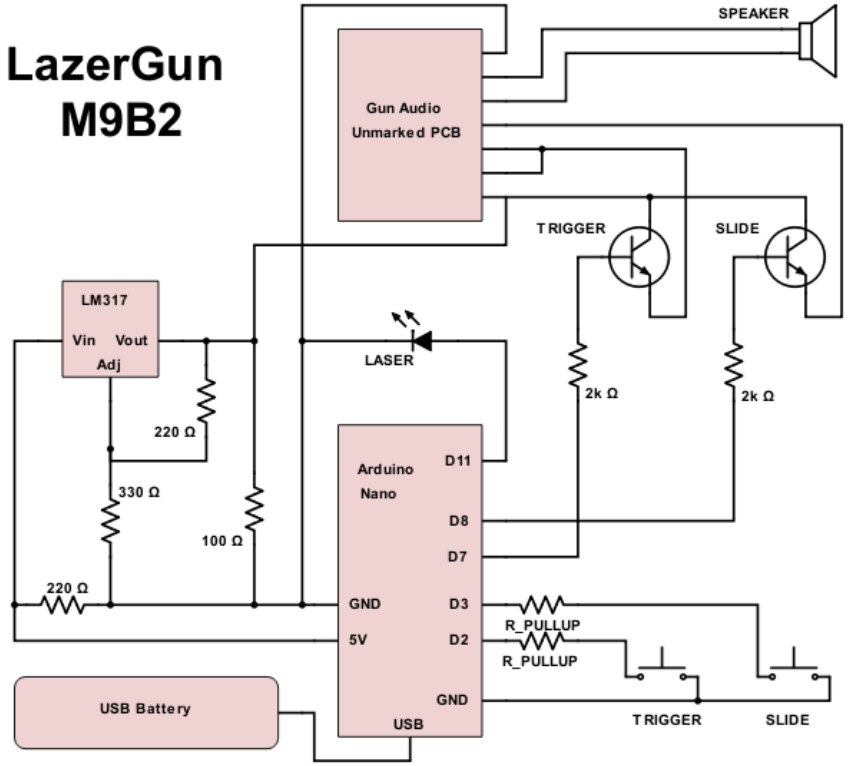
Pursuant to CU Denver Student Code of Conduct §E.21

Hardware Used for the LazerGun M9B2

- Children’s Roleplay Toy Gun
- Arduino Nano
- 6 mm Class IIIa Red Laser
- Custom Audio Interface Board
- Audio Controller PCB
- USB Power Bank
- Speaker
- Switches for the Trigger and Slide



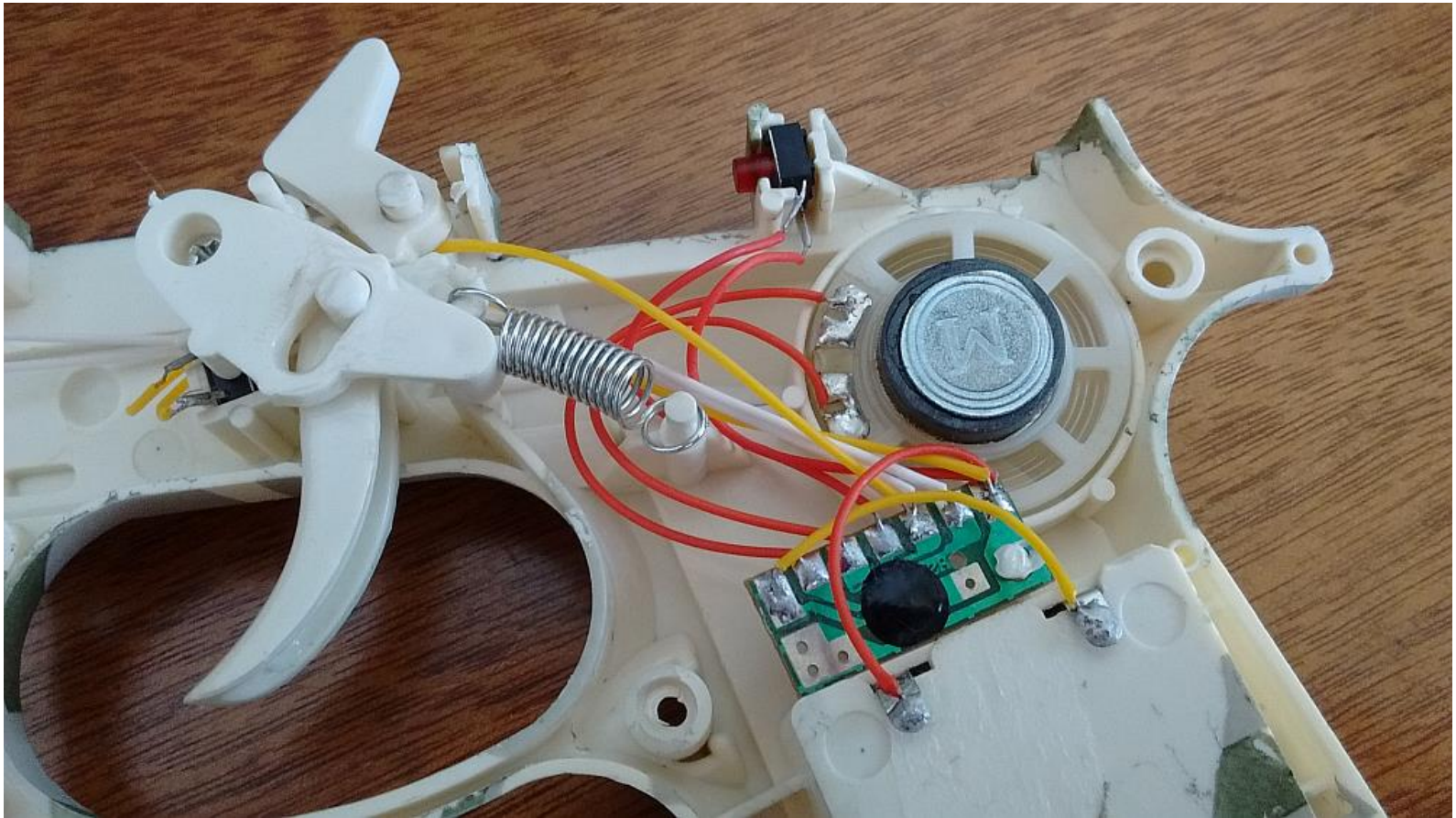
Hardware Schematic for the LazerGun M9B2



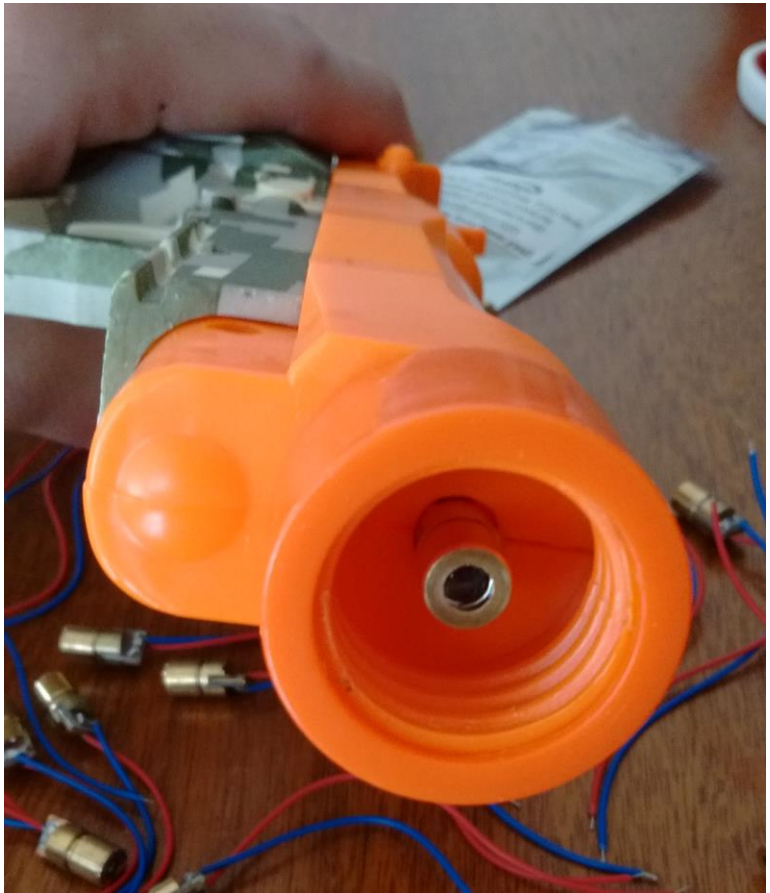
Building the LazerGun M9B2 - Part 1

Building the LazerGun M9B2 - Part 2

Building the LazerGun M9B2 - Part 3



Building the LazerGun M9B2 - Part 4

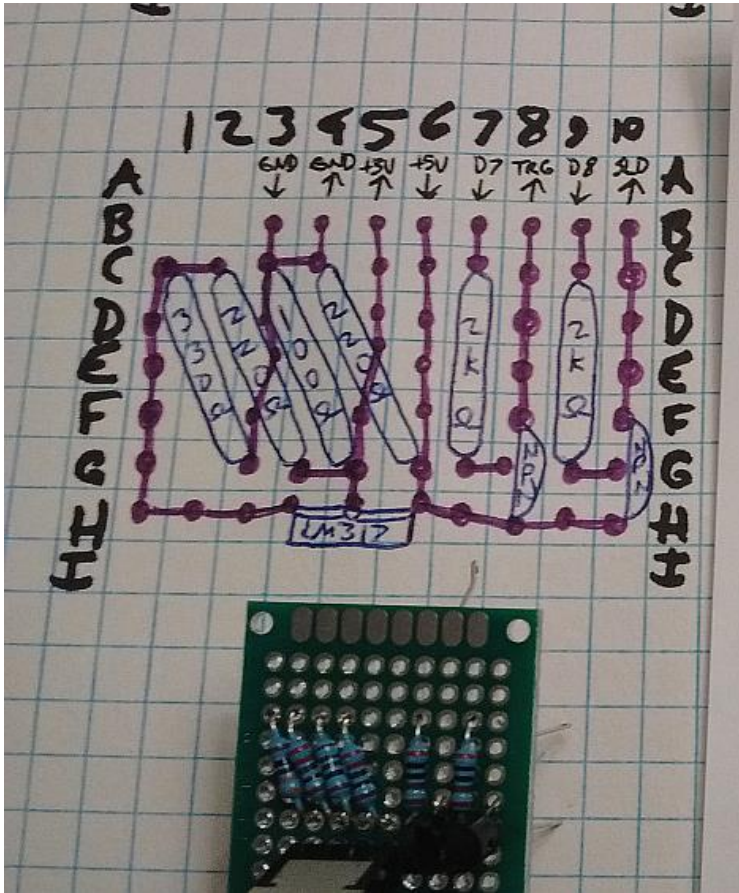


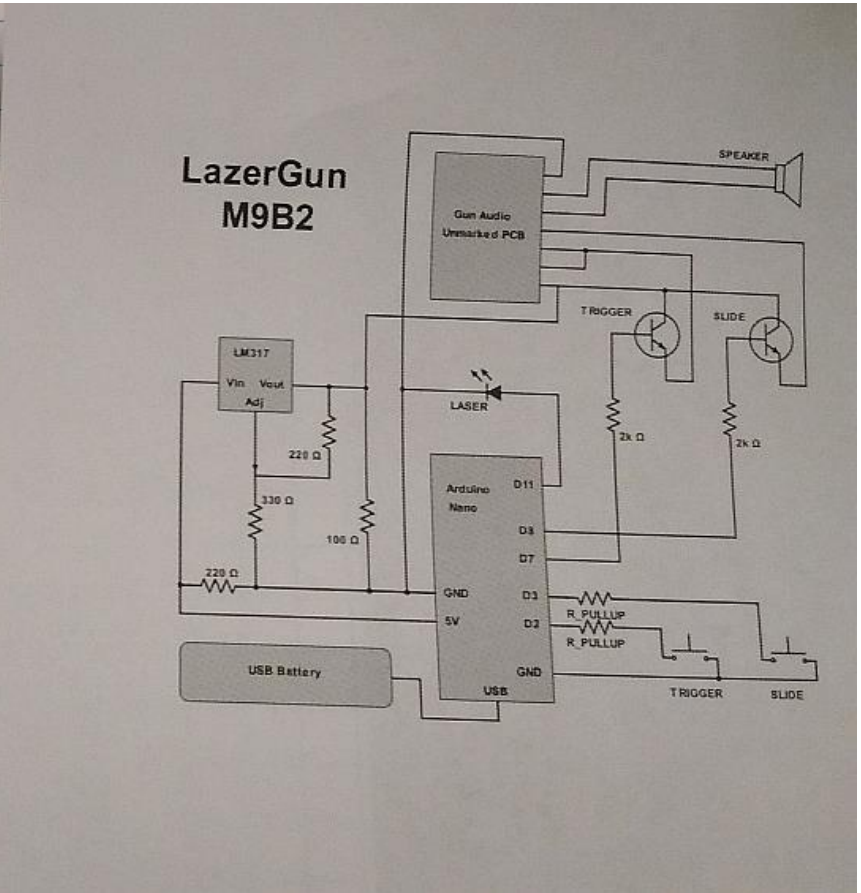
Building the LazerGun M9B2 - Part 5

Building the LazerGun M9B2 - Part 6

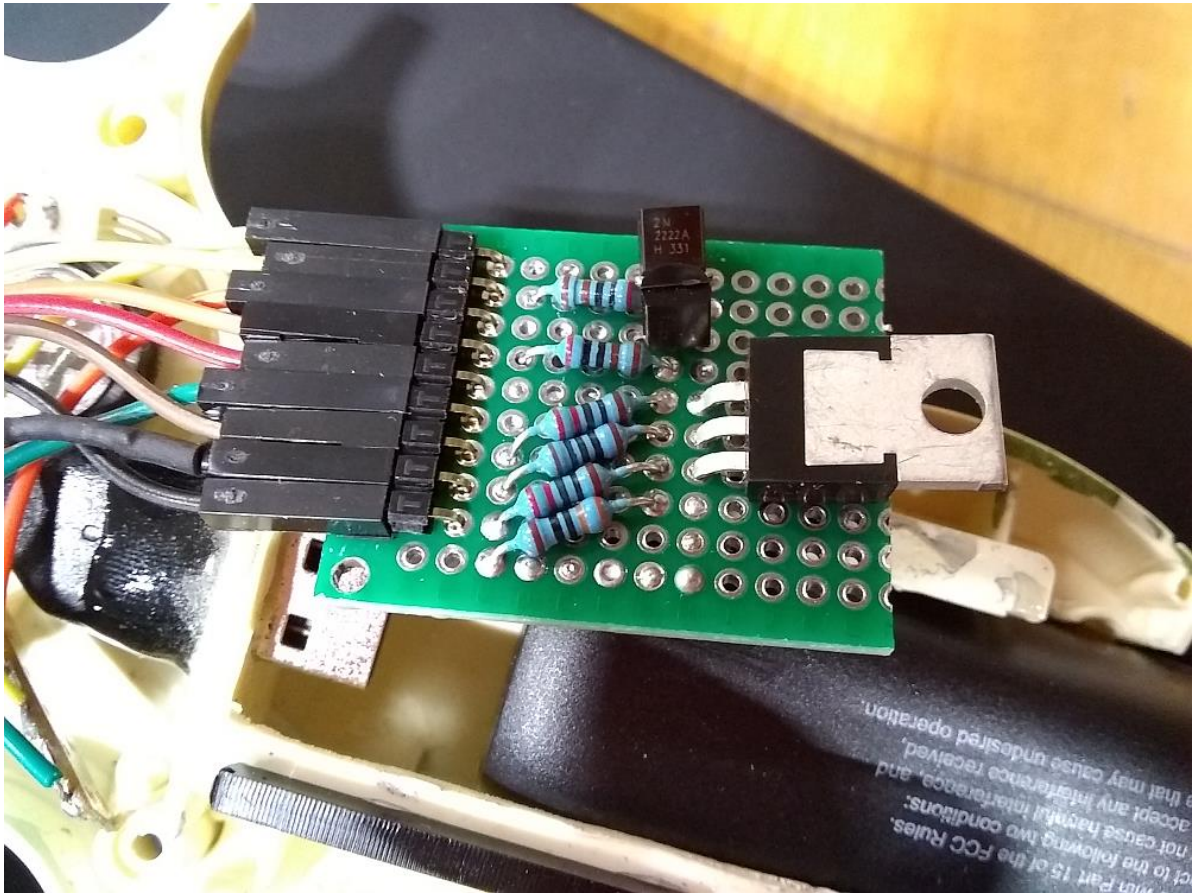
Building the LazerGun M9B2 - Part 7

Building the LazerGun M9B2 - Part 8

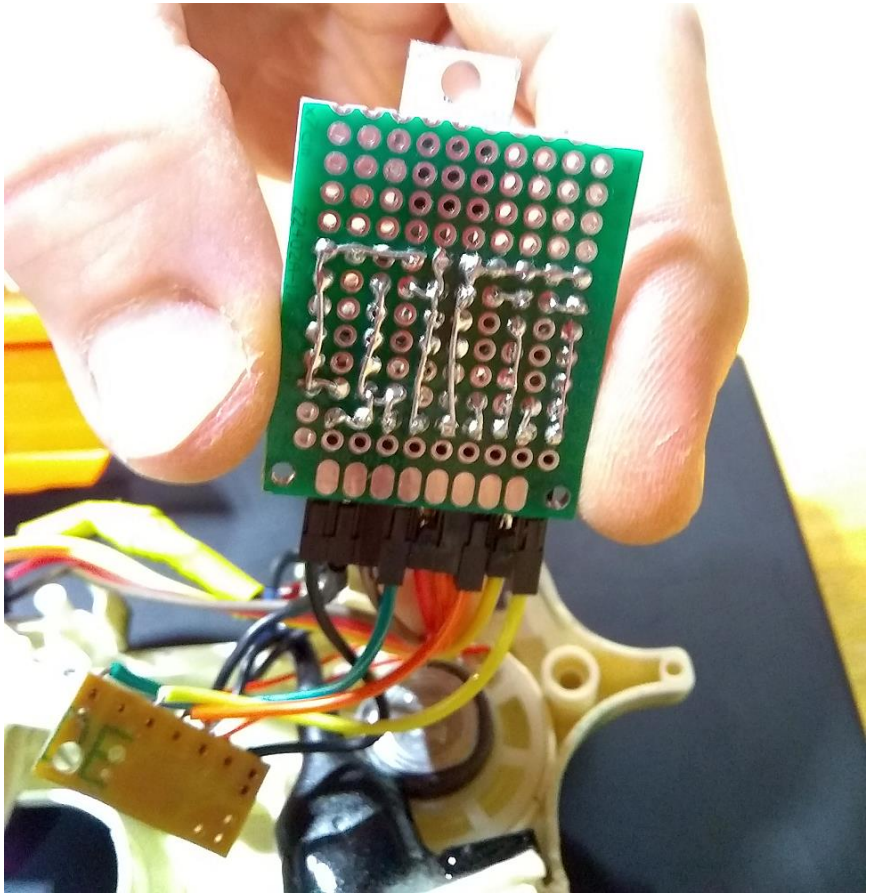




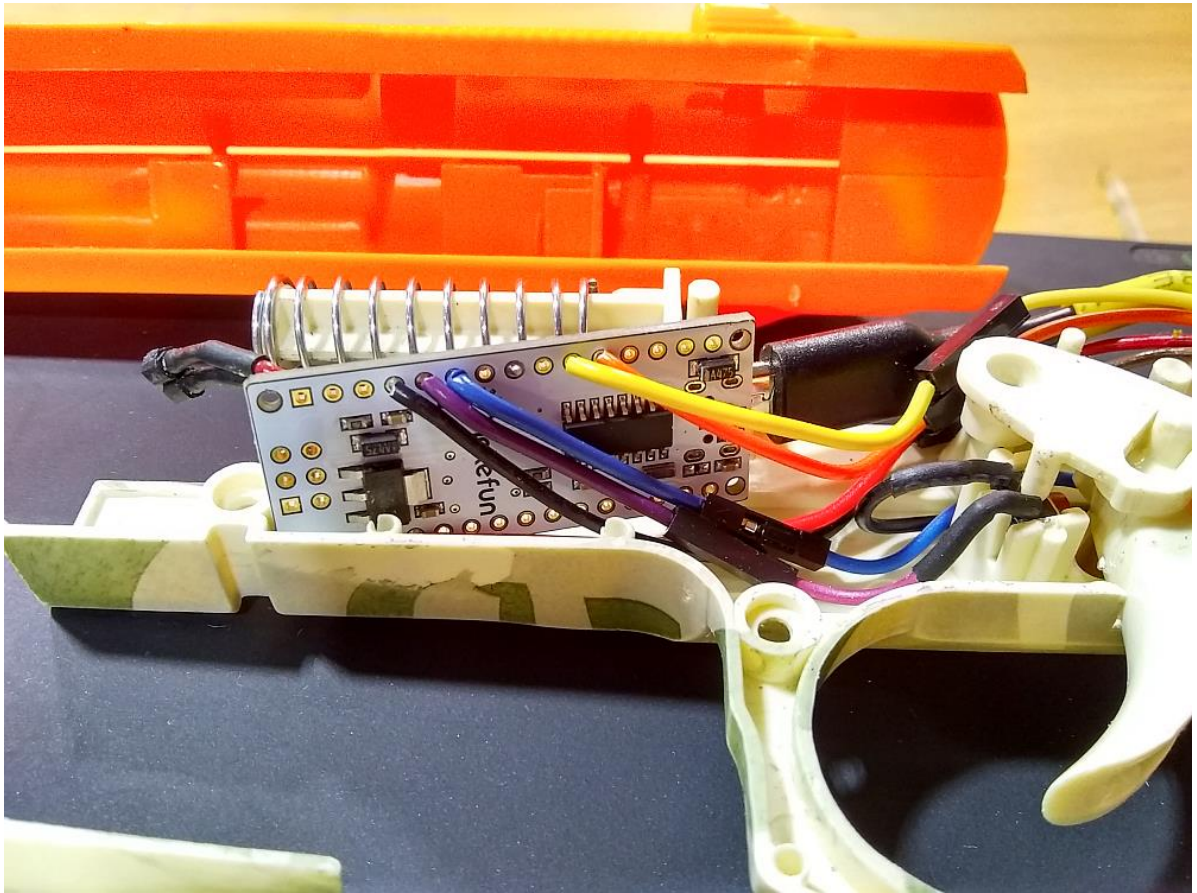
Building the LazerGun M9B2 - Part 9



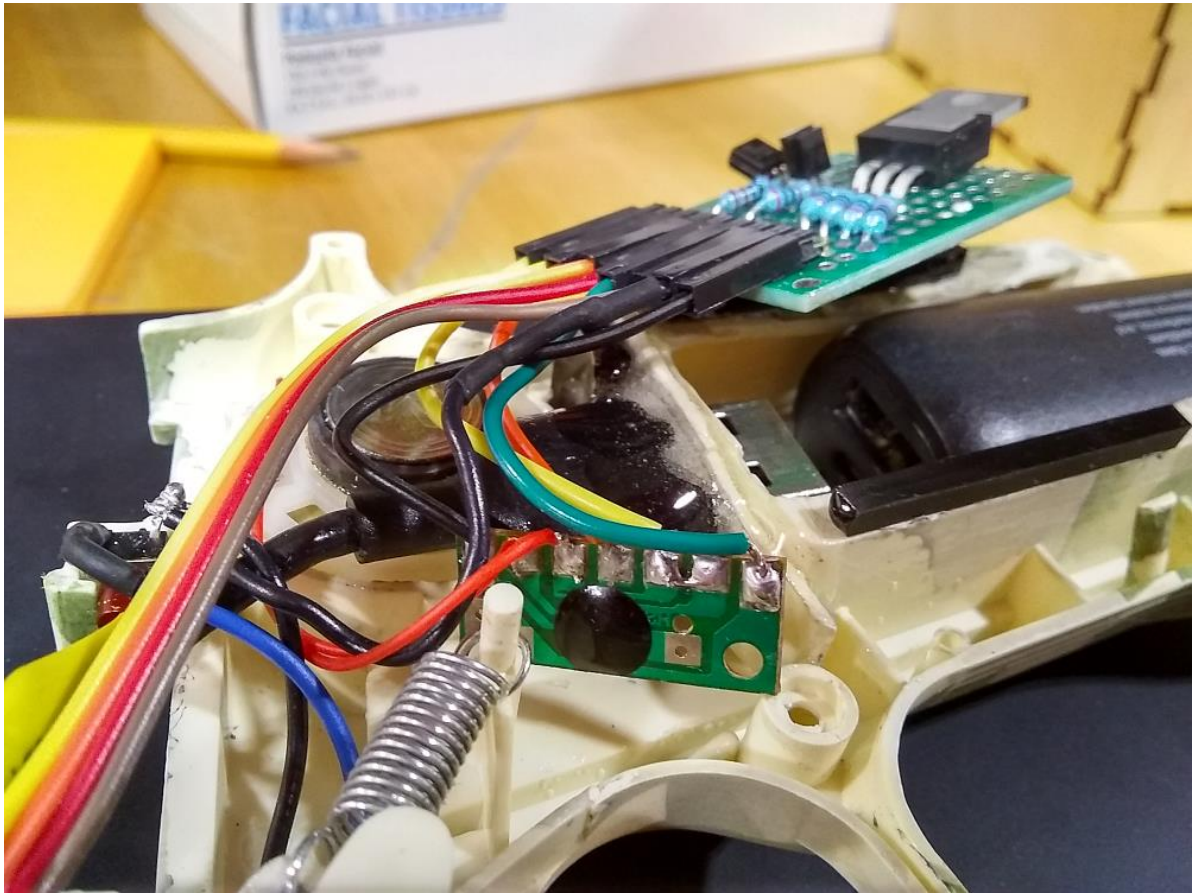
Building the LazerGun M9B2 - Part 10



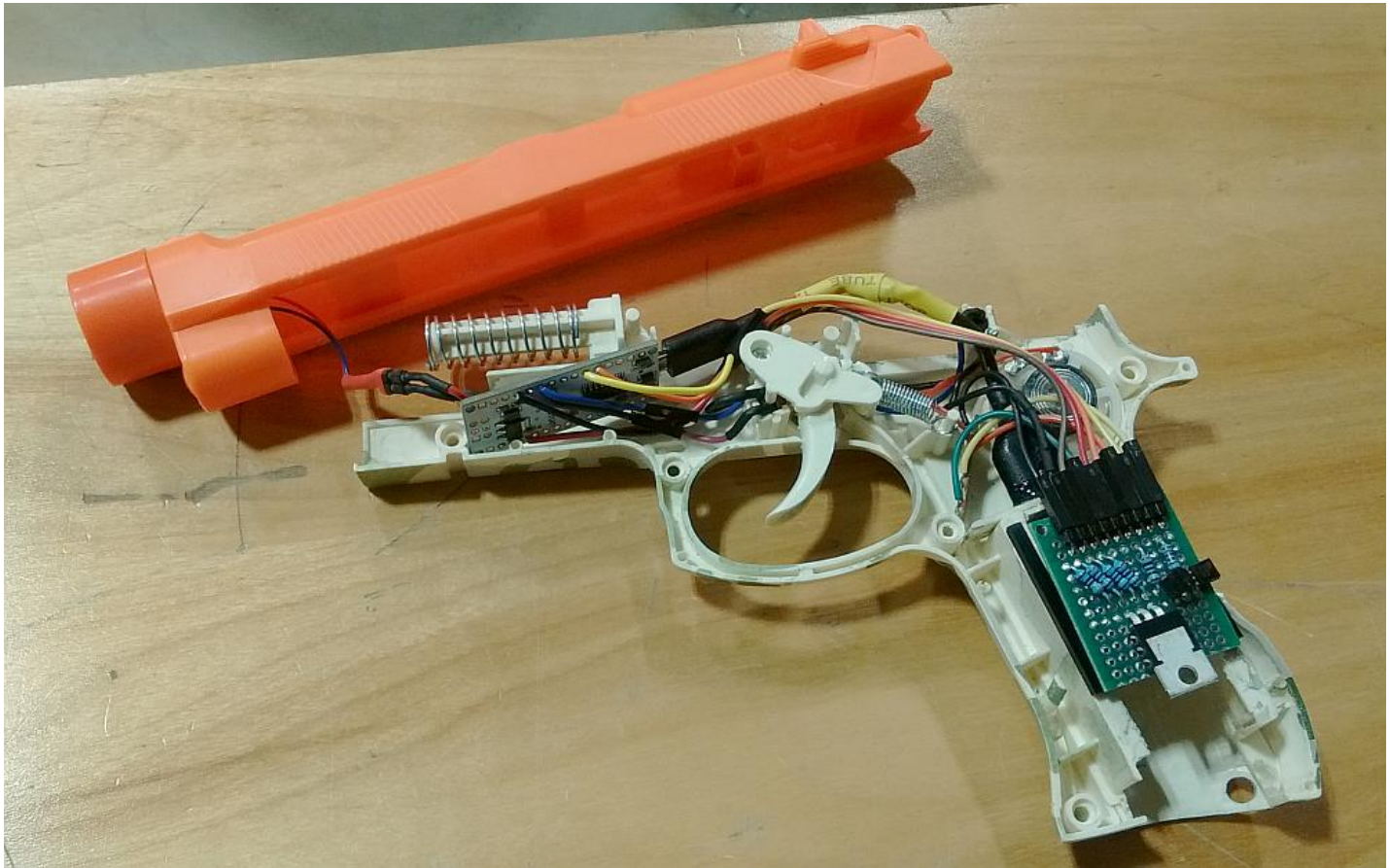
Building the LazerGun M9B2 - Part 11



Building the LazerGun M9B2 - Part 12



Building the LazerGun M9B2 - Part 13



Building the LazerGun M9B2 - Part 14

Code Implementation For LazerGun M9B2

LazerGun Generic Timer Structure

```
// TYPE DEFINITIONS
typedef struct timer16_t timer16_t;
struct timer16_t
{
  uint16_t flag_isEnabled : 1;
  uint16_t count          : 15;
  uint16_t maxCount       : 15;
  uint16_t flag_doEvent   : 1;
};
```

PseudoCode of Timer ISR for LazerGun M9B2

```
ISR(TIMER1_COMPA_vect){

// IF TRIGGER DEBOUNCE TIMER ENABLED
        // IF COUNT <= 0
                // DISABLE TIMER
                // SET DO EVENT FLAG
        // ELSE
                // DECREMENT COUNT
// IF SLIDE DEBOUNCE TIMER ENABLED
        // IF COUNT <= 0
                // DISABLE TIMER
                // SET DO EVENT FLAG
        // ELSE
                // DECREMENT COUNT
.....
}
```

PseudoCode of ISR Function for Trigger Button

```
void ISR_pin_trigger_in()
{
  // IF MAGAZINE CAPACITY IS EMPTY
                // SET FIRING MODE TO SAFETY
  // IF FIRING MODE IS SAFETY
                // DISABLE TRIGGER INPUT
  // IF TRIGGER INPUT ENABLED
                // IF FIRING MODE IS SEMI AUTOMATIC
                                // SET FIRE LASER FLAG
                // DISABLE TRIGGER INPUT
                // SET TRIGGER DEBOUNCE TIMER COUNT AS MAX COUNT
                // ENABLE TRIGGER DEBOUNCE TIMER
                // SET LASER RESET TIMER COUNT AS MAX COUNT
                // ENABLE LASER RESET TIMER
}
```

PseudoCode of ISR Function for Slide Button

```
void ISR_pin_slide_in()
 {
 // IF SLIDE INPUT ENABLED
                // IF MODE SELECTION WINDOW TIMER IS NOT ENABLED
                                // SET FIRING MODE TO SEMI AUTOMATIC
                // ELSE
                                // INCREMENT FIRING MODE
                // SET MAGAZINE CAPACITY TO MAGAZINE MAX CAPACITY
                // SET MODE SELECTION WINDOW TIMER COUNT AS MAX COUNT
                // ENABLE MODE SELECTION WINDOW TIMER
                // DISABLE SLIDE INPUT
                // SET DO RACK SLIDE FLAG
                // SET SLIDE DEBOUNCE TIMER COUNT AS MAX COUNT
                // ENABLE SLIDE DEBOUNCE TIMER
}
```

How to Use the LazerGun M9B2

Aiming:

- Line up posts flat and centered
- Laser hits at top of center post

To Put LazerGun into Battery:

- Put battery into LazerGun
- Wait for sound to indicate laser system is active
- Rack slide to release safety

Magazine Capacity:

- Colorado-compliant
- Only holds 15 rounds in the magazine at a time

Reloading:

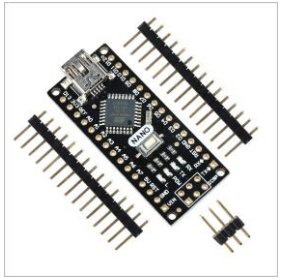
- Pull slide back (i.e. “rack slide”) to reload the magazine to full capacity

Select-Fire Modes:

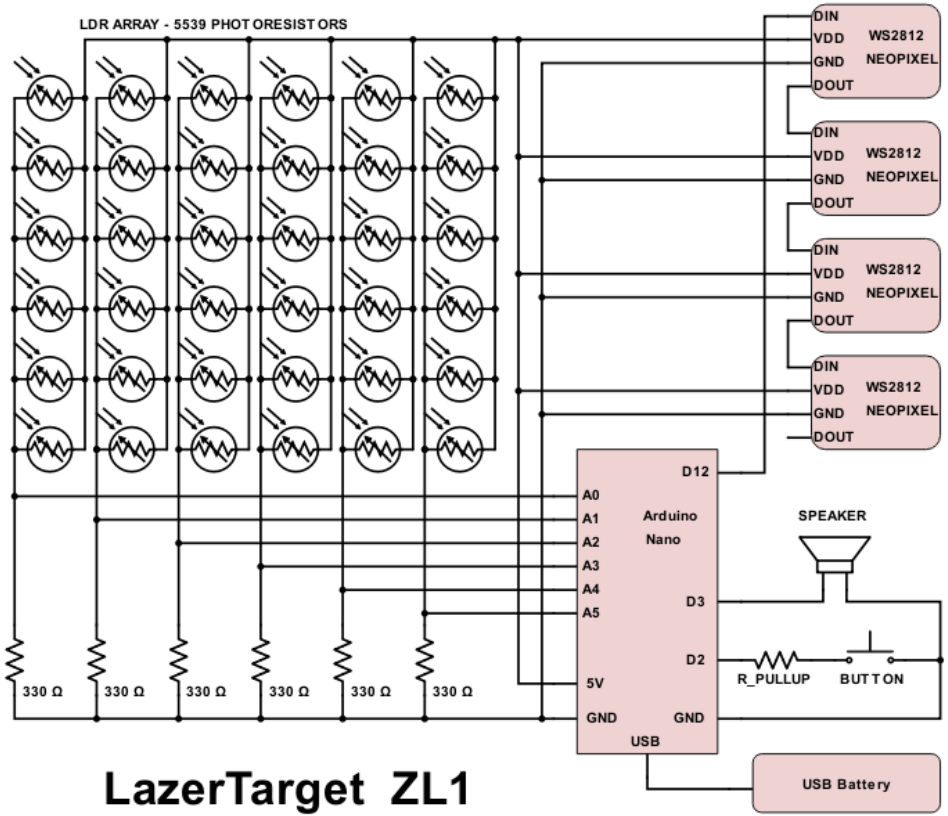
- Rack slide once for Semi-Automatic
- Rack slide twice for Three-Round Burst
- Rack slide three times for Fully-Automatic

Hardware Used for the LazerTarget ZL1

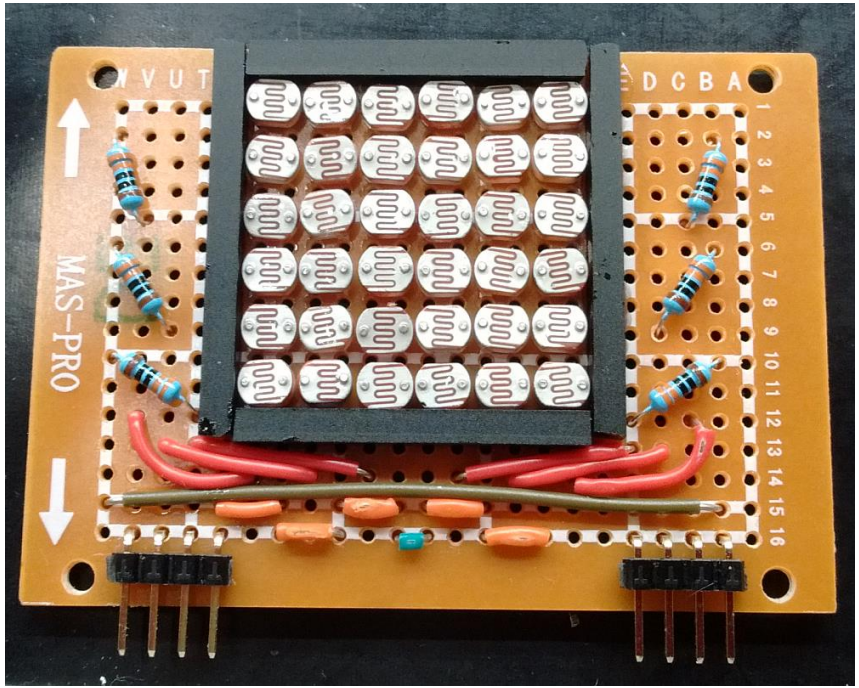
- Arduino Nano
- 36pcs LDR 5539 Photoresistors
- Custom Laser Detector Board
- USB Power Bank
- NeoPixel LED - RGB Addressable
- Momentary Pushbutton Switch
- Amplified Speaker
- State-of-the-Art Ziploc Enclosure

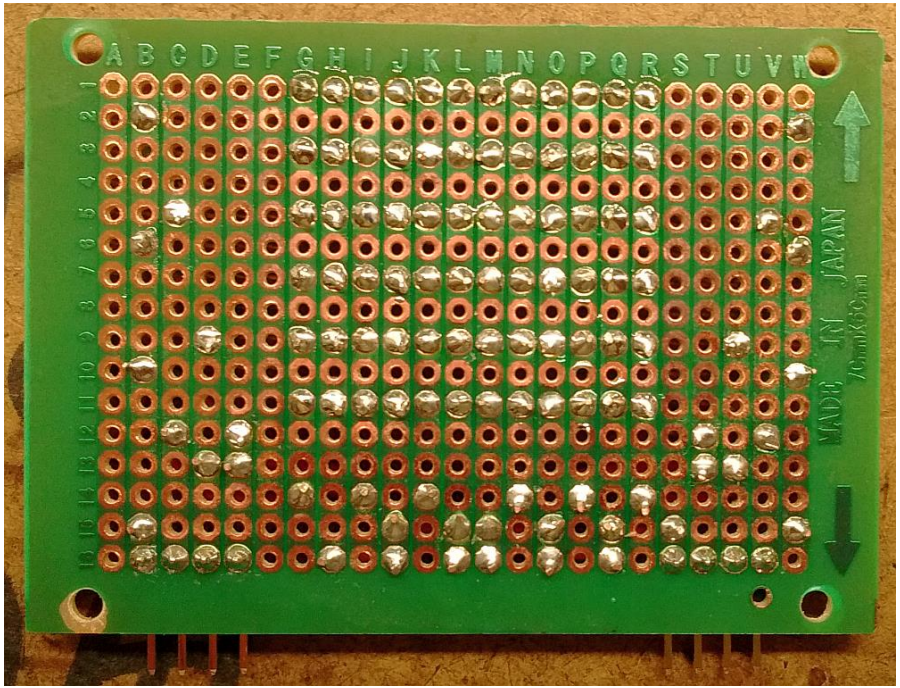


Hardware Schematic for the LazerTarget ZL1

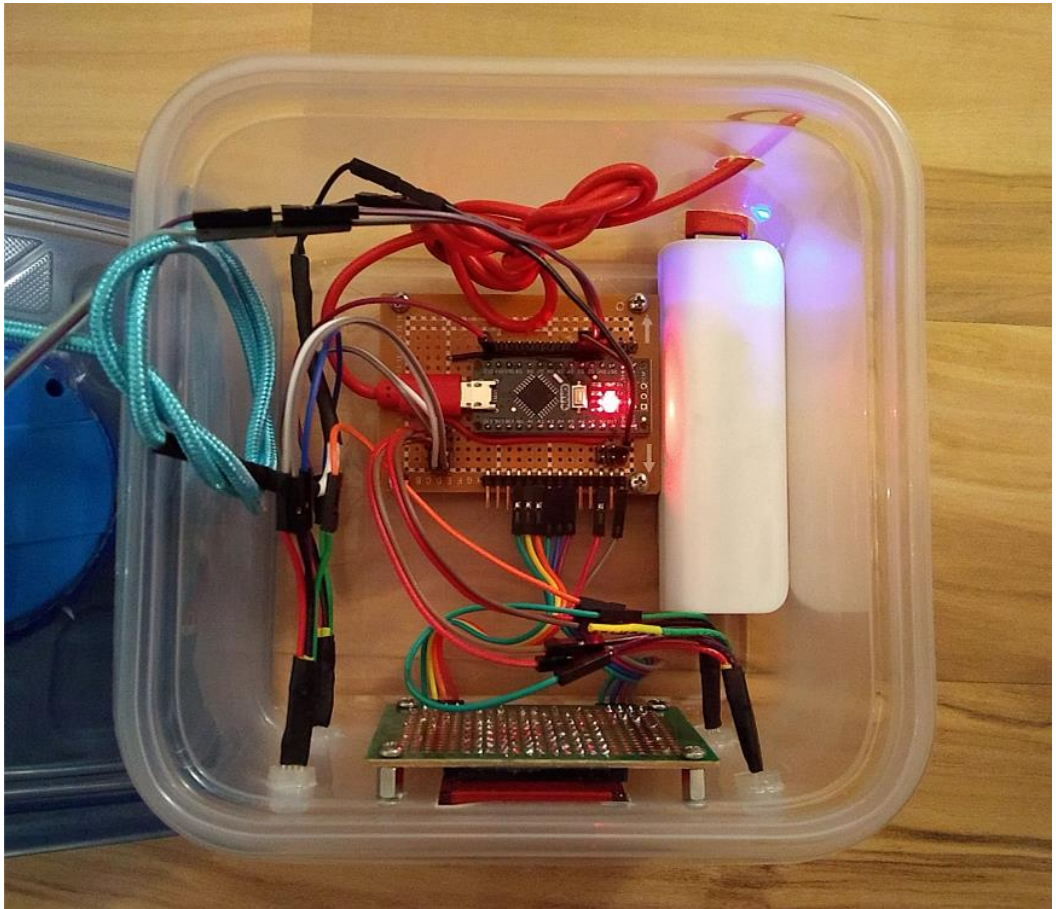


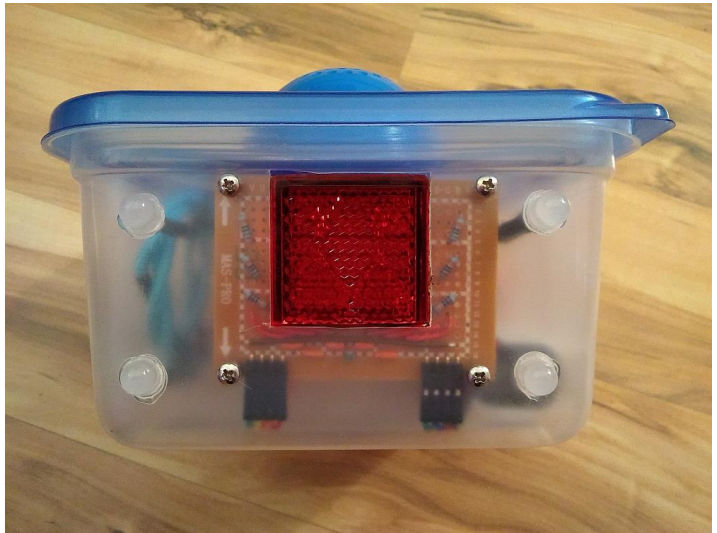
Laser Detector Board for the LazerTarget ZL1



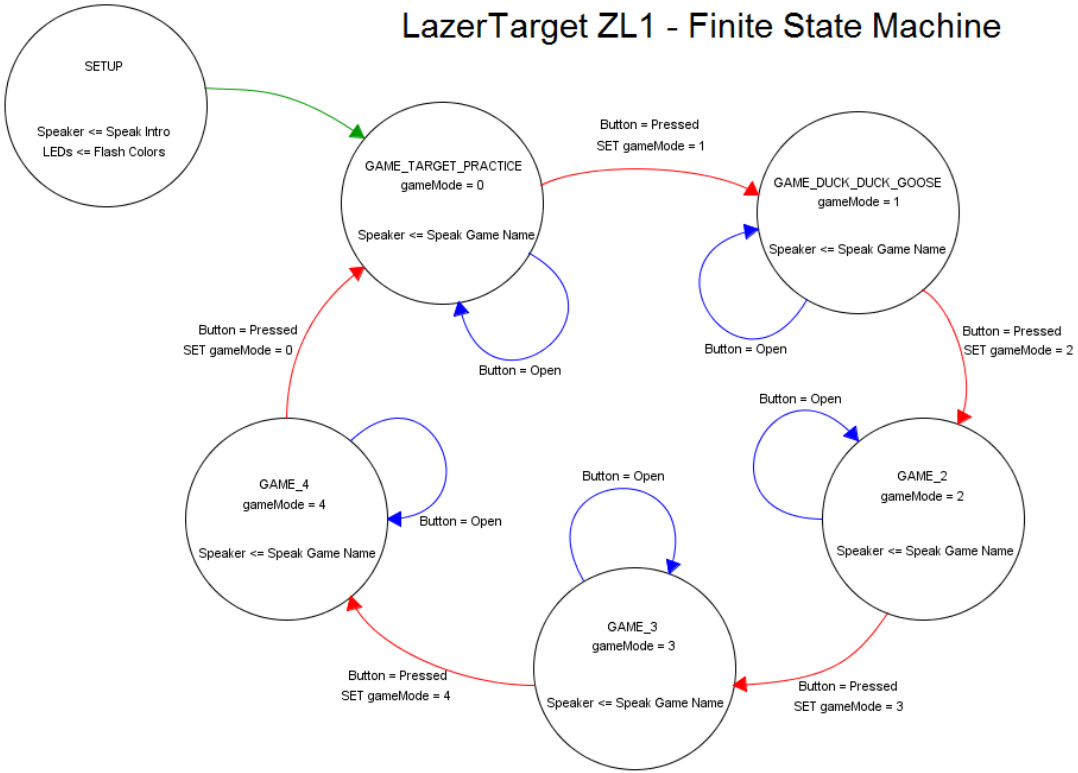


LazerTarget ZL1 Views





LazerTarget ZL1 State Diagram



Code Implementation for the LazerTarget ZL1

```
void loop()
{
  // It notifies the player of current mode once after the switch button is pressed
  // Then, one of the game types will be played according to the game mode
  switch (gameMode)
  {

    case GAME_TARGET_PRACTICE:

      // Target Practice
      Serial.println("GAME:  TARGET PRACTICE");
      game_targetPractice();
      break;

    case GAME_DUCK_DUCK_GOOSE:

      // Duck Duck Goose game mode
      Serial.println("GAME:  DUCK DUCK GOOSE");
      if (gameMode == GAME_DUCK_DUCK_GOOSE) voice.say(spDUCK);
      if (gameMode == GAME_DUCK_DUCK_GOOSE) voice.say(spDUCK);
      if (gameMode == GAME_DUCK_DUCK_GOOSE) voice.say(spGOOSE);
      game_duckDuckGoose();
      break;


    case GAME_DRAW:

      // Draw!
      Serial.println("GAME:  DRAW!");
      voice.say(spDRAW);
      game_draw();
      break;

    default:

      // Back to Target Practice (Game0)
      Serial.println("GAME:  RESET TO DEFAULT");
      voice.say(spTARGET);
      gameMode = GAME_TARGET_PRACTICE;
  }

  // BLOCK UNTIL VOICE DONE TALKING THEN DISPLAY TRANSITION EFFECT
  while (voice.talking());
  if (flag_displayTransitionEffect)
  {
    flag_displayTransitionEffect = false;

    FastLED.setBrightness(LED_BRIGHTNESS_HIGH);
    colorIndex = 0;
    timer_delay(LED_GP_TIMER_NUMBER, DELAY_LED_TRANSITION_EFFECT);
    while (timer_isActive(LED_GP_TIMER_NUMBER))
    {
      for (ledIndex = 0; ledIndex < NUMBER_OF_LEDS; ++ledIndex)
      {
        leds[ledIndex] = ColorFromPalette(RainbowColors_p, colorIndex,
                                          LED_BRIGHTNESS_HIGH, LINEARBLEND);
        colorIndex += 30;
        FastLED.show();
      }
    }
  }
  leds_setColor(CRGB::Black, LED_BRIGHTNESS_HIGH);
}
```

```
bool isTargetHit() {

  flag_isTargetHit = true;
  LDR_register = 0;

  if (analogRead(PIN_LDR_0) >= LDR_LUX_THRESHOLD)
    LDR_register |= B00000001;

  if (analogRead(PIN_LDR_1) >= LDR_LUX_THRESHOLD)
    LDR_register |= B00000010;

  if (analogRead(PIN_LDR_2) >= LDR_LUX_THRESHOLD)
    LDR_register |= B00000100;

  if (analogRead(PIN_LDR_3) >= LDR_LUX_THRESHOLD)
    LDR_register |= B00001000;

  if (analogRead(PIN_LDR_4) >= LDR_LUX_THRESHOLD)
    LDR_register |= B00010000;

  if (analogRead(PIN_LDR_5) >= LDR_LUX_THRESHOLD)
    LDR_register |= B00100000;

  if (LDR_register == B00111111)
  {
    voice.say(spROOMS_TOO_BRIGHT);
    Serial.println("ROOM IS TOO BRIGHT");
    flag_isTargetHit = false;
  }
  else if (LDR_register == 0)
    flag_isTargetHit = false;

  return flag_isTargetHit;
}
```

Interrupt Function For Pushbutton Switch

```
// INTERRUPT SERVICE ROUTINE FOR INPUT BUTTON
// activates debounce timer and switches to the next game mode once the push button is pressed
void ISR_BUTTON_PRESSED()
{
  if (flag_isButtonEnabled)
  {
    flag_isButtonEnabled = false;

    timer_buttonDebounce.count = BUTTON_DEBOUNCE_COUNT;
    timer_buttonDebounce.flag_isEnabled = 1;

    ++gameMode;
    flag_displayTransitionEffect = true;
  }
}
```

Timer Structure For LazerTarget

```
// TYPE DEFINITIONS
typedef struct timer32_t timer32_t;
struct timer32_t
{
  uint32_t flag_isEnabled     : 1;
  uint32_t flag_doEvent       : 1;
  uint32_t count              : 30;
};
```

```
// INTERRUPT SERVICE ROUTINE FOR TIMER1
ISR(TIMER0_COMPA_vect){

    // After switch button is pressed, the debounce flag of the timer will be enabled
    // The debounce timer will be disabled when its count reaches to zero
    if (timer_buttonDebounce.flag_isEnabled)
    {
        if (timer_buttonDebounce.count <= 0)
        {
                timer_buttonDebounce.count = BUTTON_DEBOUNCE_COUNT;
                timer_buttonDebounce.flag_isEnabled = 0;
                flag_isButtonEnabled = true;
        }
        else
          --timer_buttonDebounce.count;
    }


    // ISR IMPLEMENTATION FOR GENERAL PURPOSE TIMERS
    for (gpTimerIndex = 0; gpTimerIndex < NUMBER_OF_GP_TIMERS; ++gpTimerIndex)
    {
        if (timer_gpArray[gpTimerIndex].flag_isEnabled)
        {
            if (timer_gpArray[gpTimerIndex].count <= 0)
            {
                timer_gpArray[gpTimerIndex].flag_isEnabled = 0;
                timer_gpArray[gpTimerIndex].flag_doEvent = 1;
            }
            else
                --timer_gpArray[gpTimerIndex].count;
        }
    }
}
```

Sample Voice Audio Definitions (Speak & Spell Format)

```
const uint8_t spTARGET[] PROGMEM =  {0x0A, 0xD8, 0x5C, 0x4D, 0x03, 0x25, 0x8D, 0xA9, 0x24, 0x5A, 0x52, 0xB6,
                                     0x22, 0x85, 0x31, 0x1F, 0xDC, 0xD2, 0xF2, 0xB4, 0x4C, 0xDB, 0xE5, 0xCA,
                                     0xC8, 0x52, 0x0B, 0xEE, 0xA6, 0xC7, 0x2D, 0xCF, 0x53, 0x69, 0x43, 0x6E,
                                     0xA5, 0xBA, 0x94, 0x80, 0x2A, 0xAA, 0x65, 0xFA, 0x1C, 0x88, 0x36, 0x23,
                                     0x51, 0x1B, 0xEB, 0x30, 0xF4, 0xB0, 0x36, 0x6B, 0xA8, 0x51, 0x24, 0x3D,
                                     0xD6, 0xAC, 0xA1, 0x84, 0x44, 0x4F, 0x7F, 0xD4, 0xE6, 0x41, 0x46, 0x70,
                                     0x62, 0x23, 0x02, 0xA7, 0x28, 0x55, 0xA1, 0x1A, 0x00, 0xA0, 0x80, 0x21,
                                     0xDD, 0x18, 0xB0, 0xB9, 0xDA, 0xFF, 0x03
                                    };

const uint8_t spFIRE[] PROGMEM =    {0x04, 0x18, 0xCE, 0x4D, 0x02, 0x1A, 0xD0, 0x80, 0x04, 0x46, 0x91, 0x55,
                                     0x57, 0x07, 0x6D, 0xD9, 0xCD, 0xAE, 0x4F, 0x55, 0x5D, 0x59, 0x87, 0xAE,
                                     0xB9, 0xD5, 0x6D, 0x5B, 0xDB, 0x7D, 0x93, 0xB6, 0xED, 0xEE, 0xE3, 0x5A,
                                     0x6B, 0x6A, 0xF4, 0x91, 0xD5, 0x73, 0x6B, 0x67, 0xF5, 0x47, 0xBC, 0xD4,
                                     0xA7, 0x9C, 0xA5, 0x34, 0xE4, 0xD0, 0xA6, 0xF0, 0xE4, 0xAA, 0xB8, 0x2D,
                                     0xAB, 0xC3, 0x9B, 0x62, 0xC2, 0xAC, 0x74, 0xF6, 0x9F, 0xFB, 0x72, 0x0B,
                                     0xEC, 0x92, 0xCD, 0xEE, 0xCF, 0x43, 0x69, 0x4C, 0x5B, 0xFF, 0x3F
                                    };

const uint8_t spSECONDS[] PROGMEM = {0x04, 0xF8, 0xC5, 0x51, 0x01, 0xBF, 0xA6, 0x6A, 0x40, 0x03, 0x16, 0xD0,
                                     0xC0, 0xCA, 0xAB, 0x75, 0x2D, 0xCD, 0x25, 0x37, 0xBB, 0xD9, 0xCA, 0xDA, 0x54,
                                     0x0F, 0xEE, 0xD9, 0x29, 0x6B, 0x47, 0x30, 0xD8, 0xE3, 0x80, 0x00, 0x6A, 0x26,
                                     0x6D, 0x55, 0xEB, 0xCA, 0x21, 0xB9, 0xE4, 0xD4, 0xDD, 0x26, 0xA5, 0xF9, 0xE3,
                                     0x3D, 0xB6, 0x75, 0x38, 0xA3, 0x31, 0x5B, 0x9A, 0xB6, 0x11, 0x51, 0x32, 0xD2,
                                     0xAA, 0x3F, 0xFC, 0x21, 0xCE, 0x22, 0xD1, 0xD7, 0x2D, 0x9E, 0x39, 0x0B, 0x37,
                                     0x4E, 0xD7, 0x26, 0xE1, 0xFA, 0xC4, 0x55, 0x42, 0xFD, 0x85, 0xFB, 0x7B, 0x77,
                                     0x13, 0xA3, 0x27, 0x80, 0x03, 0xD0, 0x25, 0x20, 0x01, 0x0A, 0x20, 0x20, 0x69,
                                     0xD6, 0xFF, 0x07
                                    };
```

Sample Game Code: Game “Draw!”

```
6  void game_draw() {
7
8    while (gameMode == GAME_DRAW) {
9      // BLOCKING DELAY FOR GAME TO START
10     timer_delay(1, DELAY_GAME_START);
11
12     while (timer_isActive(1) && gameMode == GAME_DRAW);
13
14     if (gameMode != GAME_DRAW) return;
15
16     voice.say(spREADY, false);
17     leds_setColor(CRGB::Yellow, LED_BRIGHTNESS_LOW);
18
19     timer_delay(1, DELAY_READY);
20     while ((voice.talking() || timer_isActive(1)) && gameMode == GAME_DRAW);
21
22     if (gameMode != GAME_DRAW) return;
23
24     voice.say(spDRAW, false);
25
26     leds_setColor(CRGB::Blue, LED_BRIGHTNESS_LOW);
27
```

Sample Game Code: Game “Draw!”

```
28    timer_delay(5, DELAY_MAX_ROUND_WINDOW);
29    while (timer_isActive(5) && gameMode == GAME_DRAW) {
30
31      if (isTargetHit()) {
32        timer_stop(5);
33        leds_blinkColor(CRGB::Green, LED_BRIGHTNESS_HIGH,
34                  LED_FAST_BLINK_CYCLES, LED_FAST_DELAY_TIME, GAME_DRAW);
35        break;
36      }
37    }
38
39    double seconds = (DELAY_MAX_ROUND_WINDOW - (timer_getCount(5) *
40                      TIMER_INTERVAL_MILLISECONDS)) / 1000.0;
41
42    Serial.print("SECONDS SPENT:  ");
43    Serial.println(seconds);
44
45
```

Best Embedded Systems Project?

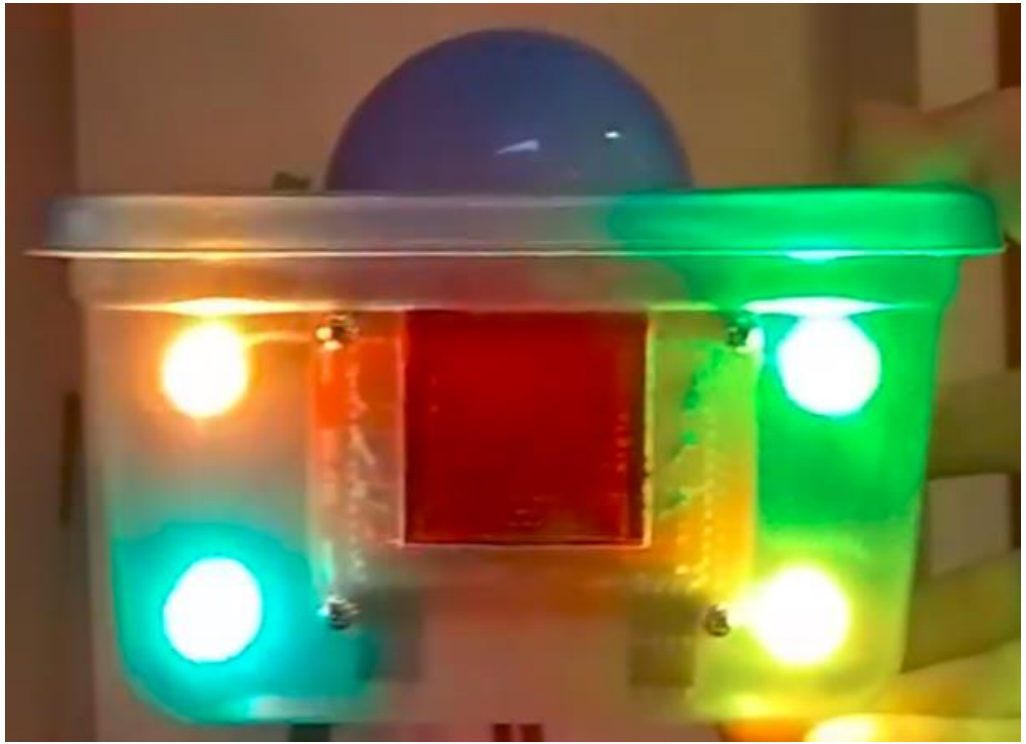
- **Most Complex Project:**
  - Advanced electronic circuit design
  - Retrofitting existing objects to house electronic circuitry and components
  - Advanced firmware programming techniques

- **Most Well-Built Project:**
  - All electronic components soldered securely in place
  - Durable housings for components
  - Hardwired headers with disconnectable leads

- **Most Concurrent Project:**
  - Multiple non-blocking timer delays
  - Simultaneous speech and LED usage
  - Simultaneous polling of input pins and use of timer delays

- **Most Fun Project:**
  - It's definitely fun...
  - But is it the most fun? You decide:
  - **We will now need a volunteer** (somebody who is accurate with a pistol)

Demo Video: LazerTarget ZL1 / LazerGun M9B2



https://www.youtube.com/watch?v=4LuswPReQVM