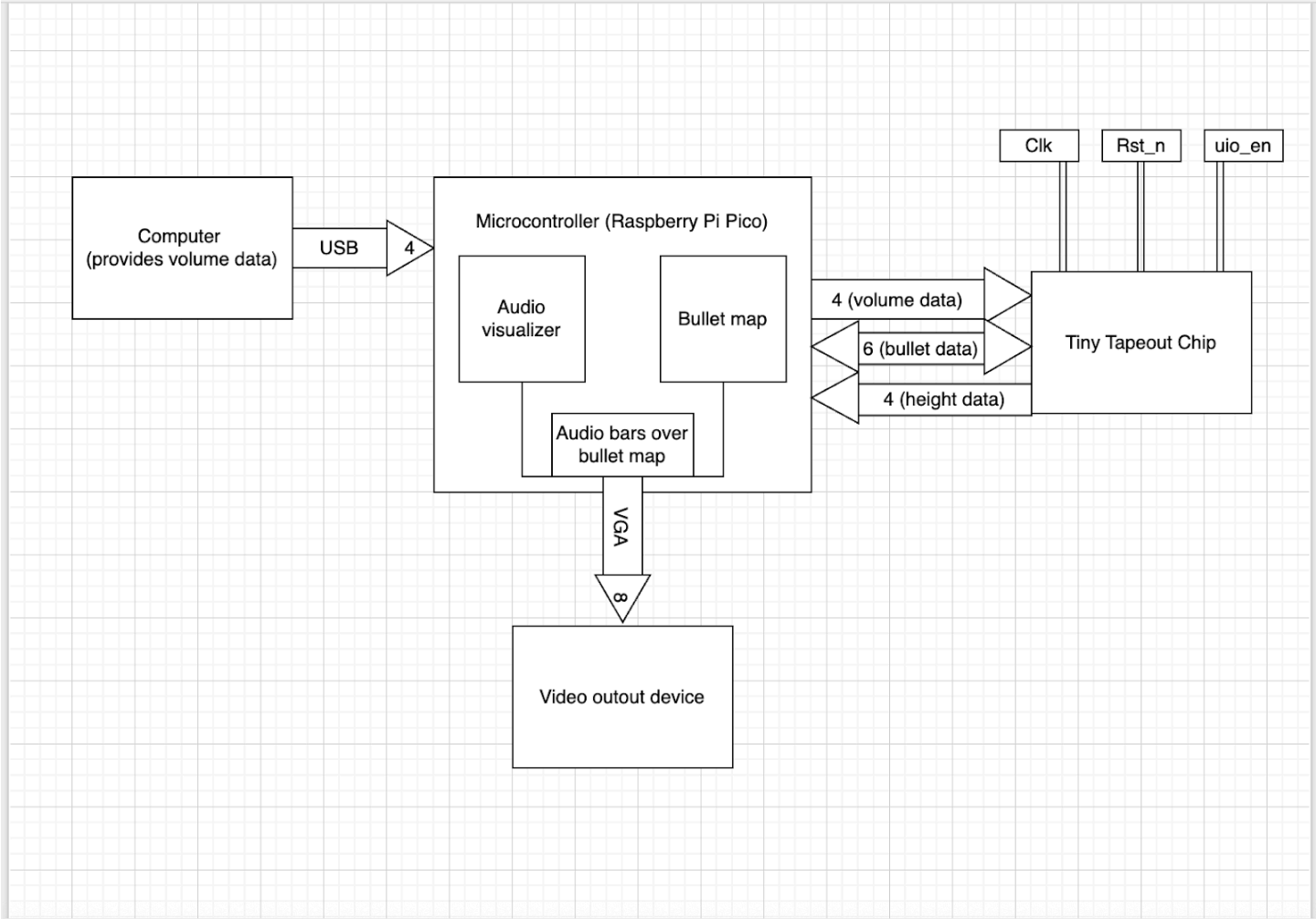
Summary of Idea: Create an audio visualizer using the letters in UWATERLOO as the growing/shrinking bars. The text (bars) will be colored "waterloo gold". Our demoscene will use Bad Apple (or another touhou song) as audio, though it will be configured in a way such that it can adapt to any audio. For the background, we will use the chip to generate patterns similar to those in the Touhou bullet hell games.

Block Diagram:



I/O:

#	Input	Output	Bidirectional
1	clk		X pos (in/out n times)*
2	rst_n		Y pos (in/out n times)*
3	uio_enable		X vel (in/out n times)*
4	volume	Stretch height	Y vel (in/out n times)*

5	volume	Stretch height	Type of bullet
6	volume	Stretch height	Type of bullet
7	volume	Stretch height	
8			

Projected Schedule

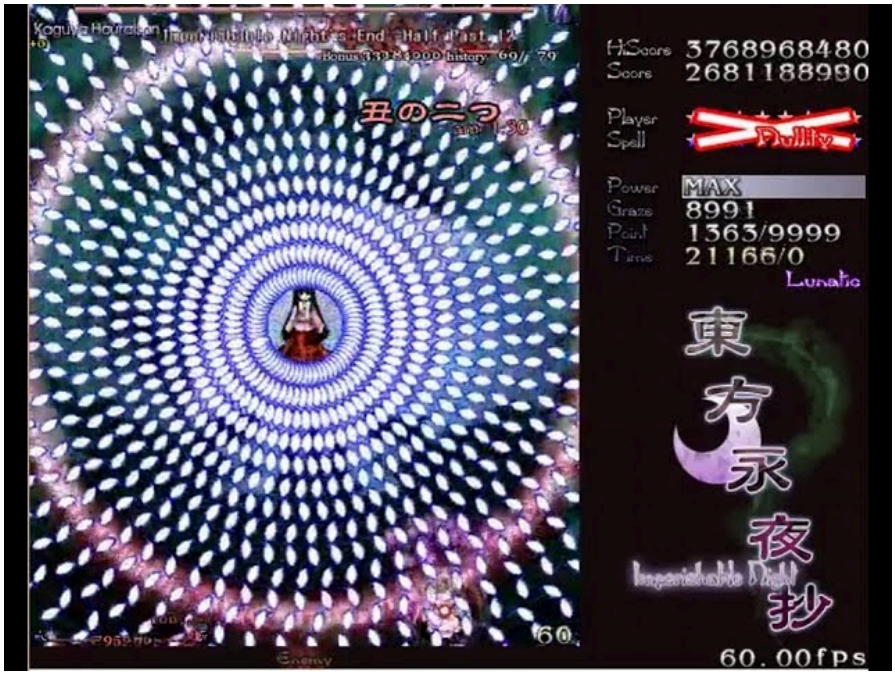
- October 1:
  - Finish counter project, block diagram, I/O
  - Complete code to transform mp4/mp3 audio into input volume values
  - Decide how many "volume bars" the audio visualizer will have
- October 8:
  - Implement bullet types & their corresponding chip calculations (velocity, size, color)
  - Program micro-controller to accept usb input (to receive volume data)
  - Program micro-controller to prepare to output volume and bullet data to chip
- October 15:
  - Start verilog implementation of the chip
- October 22:
  - Midterms
- October 29:
  - Finish verilog implementation of the chip
- November 5:
  - Join microcontroller logic (interface with chip, storing bullet data, display) and I/O with chip logic and I/O
- November 12:
  - Connect micro-controller to external interface (vga output)
- November 19:
  - Start testing, fix any errors
- November 26:
  - Grace period for more testing/updating
- December 3:
  - Finish demoscene, test successful
  - Finals

Proposal

- ***Design docs resources for examples on last page**
- **Summary of Idea:** Create an audio visualizer using the letters in UWATERLOO as the growing/shrinking bars. The text (bars) will be colored "waterloo gold". Our demoscene will use Bad Apple (or another touhou song) as audio, though it will be configured in a way such that it can

adapt to any audio. For the background, we will use the chip to generate patterns similar to those in the Touhou bullet hell games.

- **Specifics:**
  - Focus on shapes (color later if we have space) for the background and bullets
    - At the very least though, we want to have black, white, yellow (for audio visualizer)
  - Avoid bullet overlap
  - ~~Avoid having more than 10 bullets on screen moving at once~~
    - Data can be stored in the microcontroller's RAM, which is on the order of megabytes. Thus, we can store many dozens of bullets



- Combined image:
  - Audio visualizer using UWATERLOO instead of generic bars (font will be smaller to provide more points)
    - 

Requirements

- Must be able to correctly display Bad Apple at minimum at 720p and the original frame rate (30fps).

- Must be able to correctly display text at minimum the same resolution as the video (720p original frame rate, i.e. 30fps)
- Must accept audio input
- Must be capable of displaying at least 20 bullets on the screen at the same time

Design

- **Chip I/O:** Takes volume input and outputs stretch height based on volume.
    - **Inputs**:
        - Clk
            - 20 Mhz?
        - Rst_n (reset)
        - Volume_in (15 possible values)
            - This value is scaled by the software then provided to the chip
            - Is a waveform
        - Bullet location
            - Max 8 bytes for 1 bullet (depending on output screen resolution)
    - **Outputs**:
        - Stretch_height (15 possible values)
            - This value is generated by the chip and then scaled by the software
        - New bullet location
            - One pin for X, one pin for Y
            -
    - **Outputs**
        - Vga output will come from the microcontroller
    - **Timing**
        - Bullets will update location every ~16 frames (time for calculations)
        - Audio visualizer will update once every time every bar's new stretch height is calculated (assume 240 frames, 30 bars)
        - We can treat the connection between the microcontroller and the chip as a UART connection
- **Micro-controller:** Raspberry Pi Pico
    - 264 kb SRAM
    - 2 mb flash memory (for code)
- **External Functions (Software, MC):**
    - Python/C++ to convert video volume to values readable by the chip
    - Values are sent to and received from the chip through a microcontroller
    - Python/C++ then uses the stretch height to modify the video
    - Python/C++ runs the video with the text on the computer