DCLab Group1 Presentation

Table of Contents

Table of Contents

Table of Contents

Table of Contents

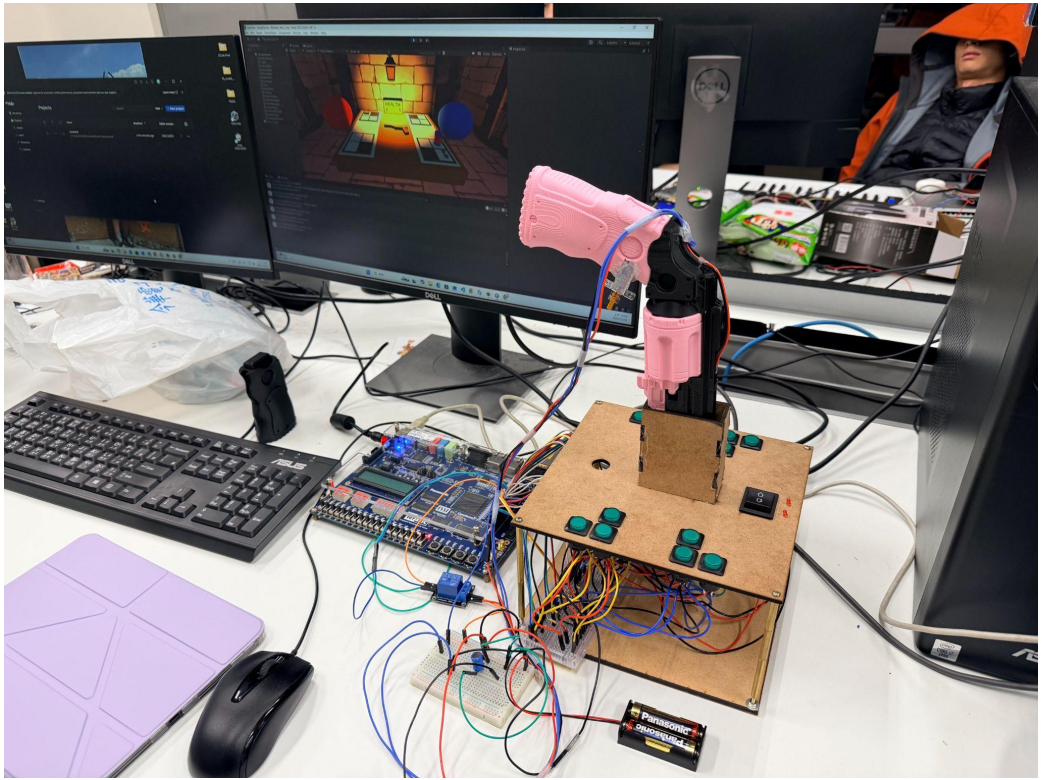
Introduction

Game Rules - Shoot

Game Rules - Shoot

Game Rules - Item

Make it reeeeeeeal !!



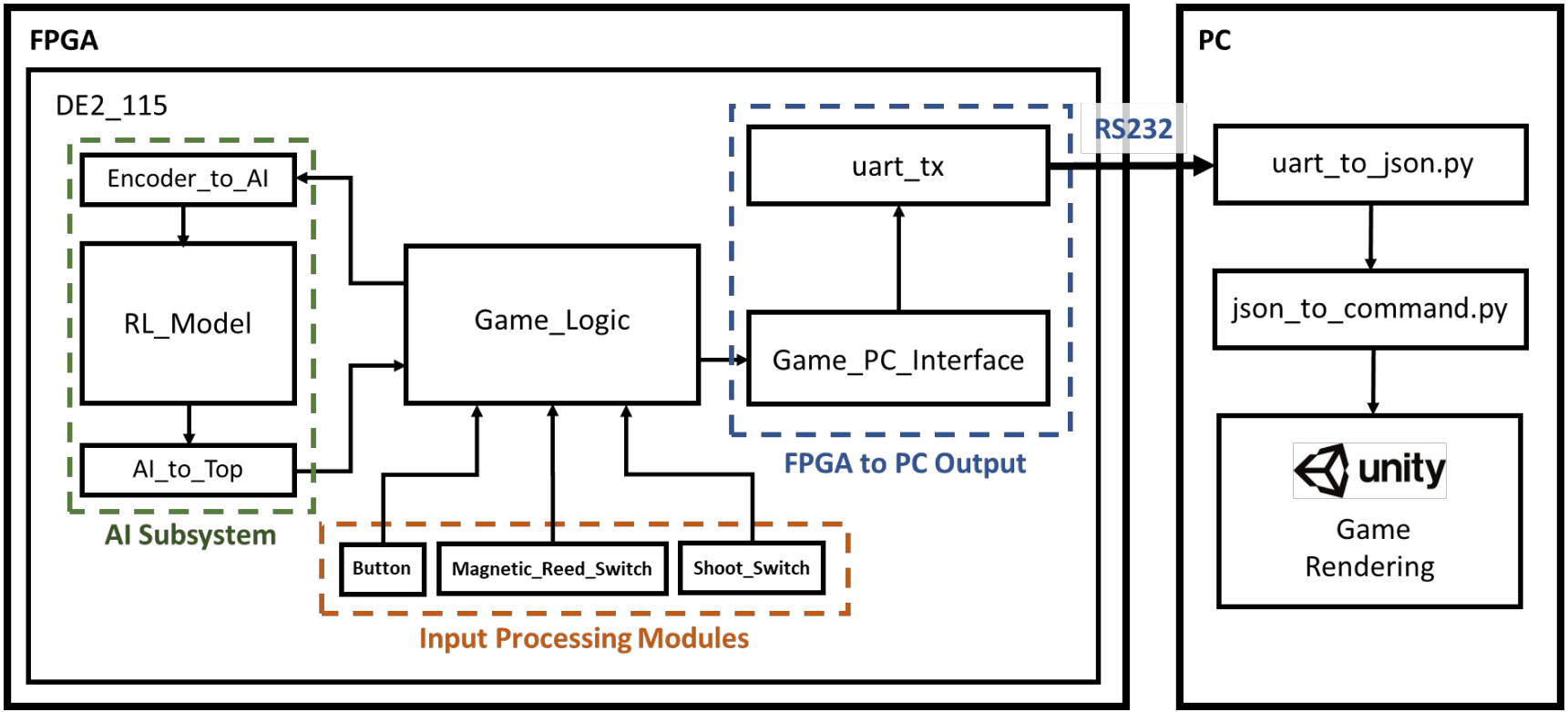
What we have done

- Physical System Implementation
  - Gaming Platform
  - Input Buttons and Switches
  - Electric Shocking Gun

What we have done

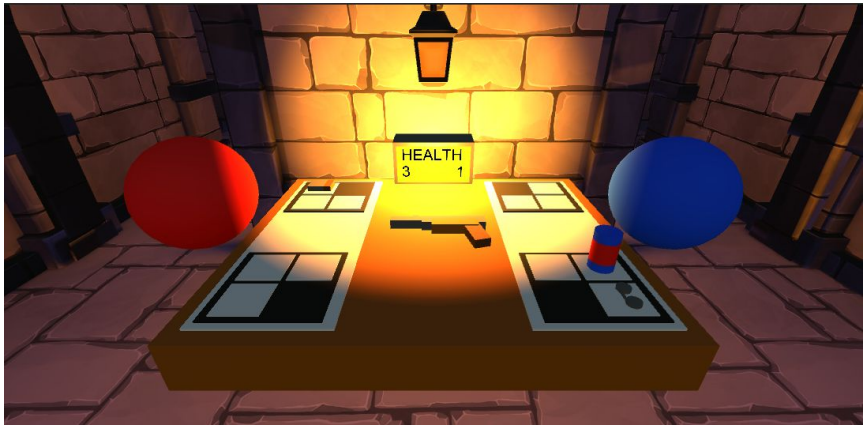
- Gaming Mode
  - Player v.s. Player
  - RL model v.s. Player
  - RL model v.s. RL model

Block Diagram



Game Rules

- Arbitrary Bullets and items
- Shoot yourself to extend your turn.
- Shoot your opponent to defeat them.




Beer

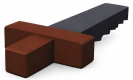

Cigarette

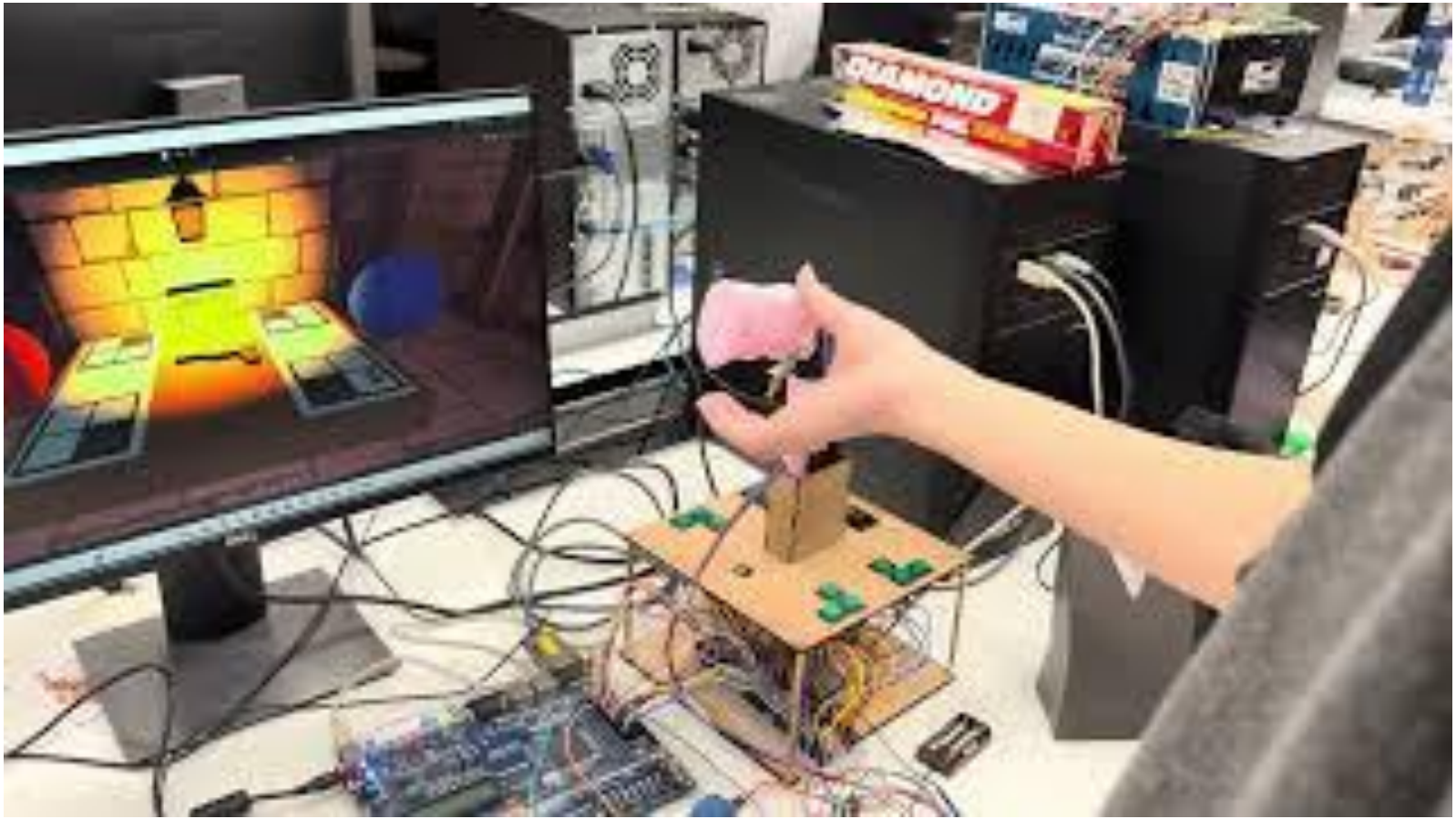

Handcuff


Magnifier


Phone


Reverse


Saw

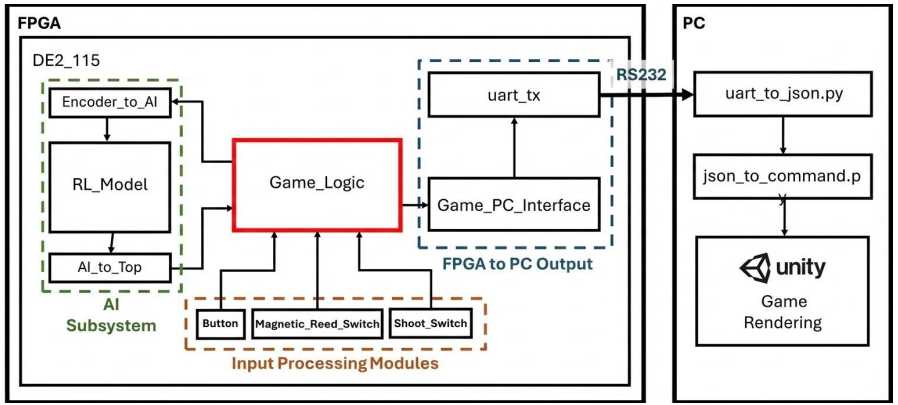

DIAMOND

Game Logic

Game Logic

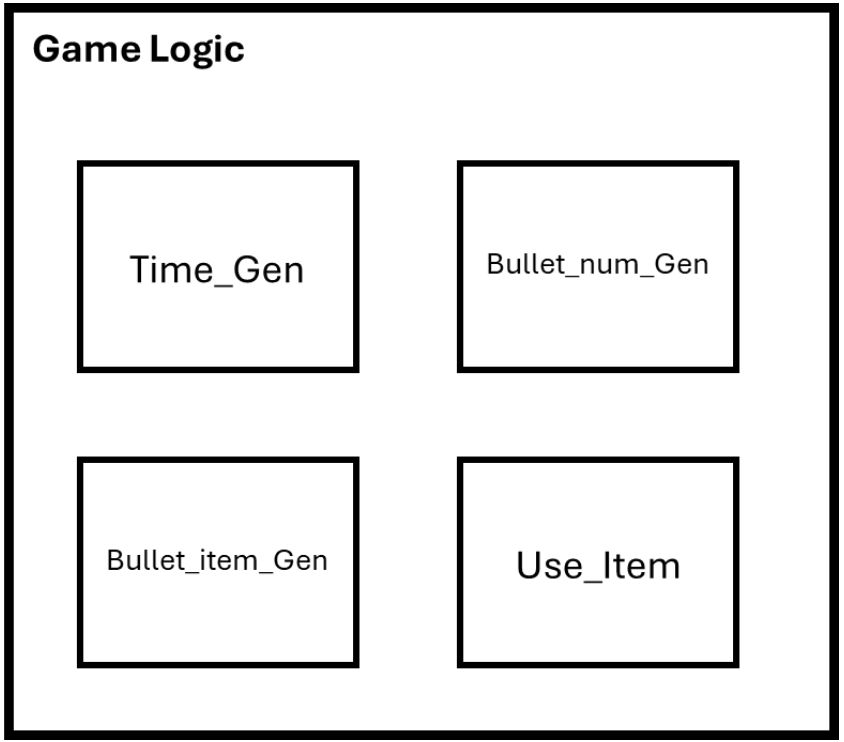
Actual Game Logic Implementation

- Inforces game rules, manages turns, health, items, and bullet logic
- Submodules :
  - Time_Gen
  - Bullet_num_Gen
  - Bullet_Item_Gen
  - Use_Item



Game Logic

- Submodules :
  - Time_Gen
  - Bullet_num_Gen
  - Bullet_Item_Gen
  - Use_Item

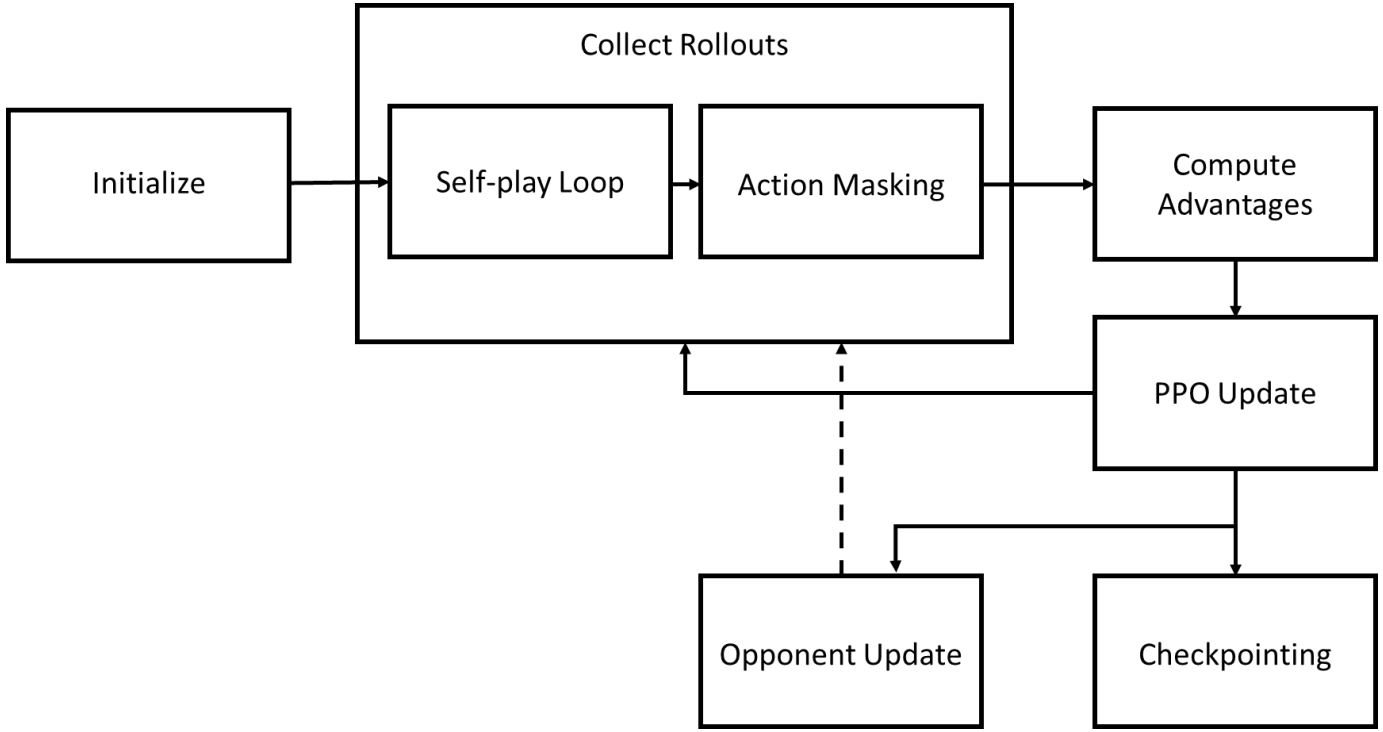


RL Model

RL model-Model Architecture

- **Algorithm**: MaskablePPO (PPO with action masking support)
- **Network**: MLP — Input(30) → Dense(128) → Dense(128) → Output(9)
- **Parameters**: ~21k trainable parameters
- **Input** (33 dims): global info (5), phase (2), self info (9), opponent info (9), bullet knowledge (8)
- **Output** (10 actions): discrete actions (0: Shoot enemy, 1: Shoot self, 2–8: Use items, 8: Ready)

RL Model-Training Procedure



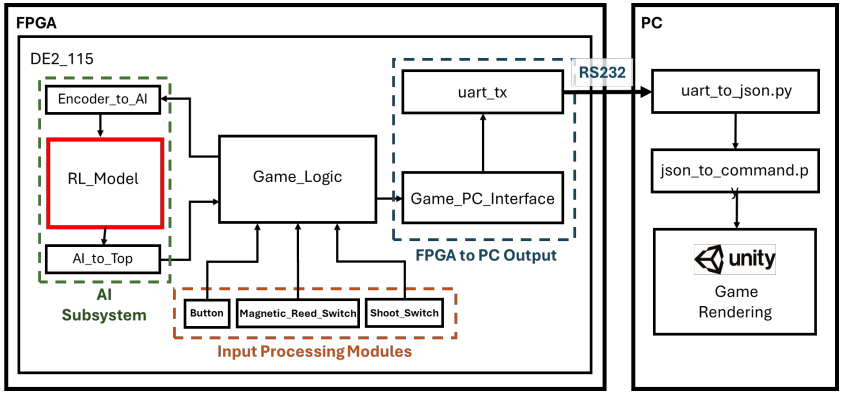
RL Model-Hyperparameter

Parameter	Default	Description
`total_timesteps`	1,000,000	Total training steps
`n_envs`	4	Parallel environments
`learning_rate`	3e-4	Learning rate
`batch_size`	64	Batch size for updates
`n_steps`	2048	Steps per environment before update
`opponent_update_freq`	10,000	Steps between opponent updates

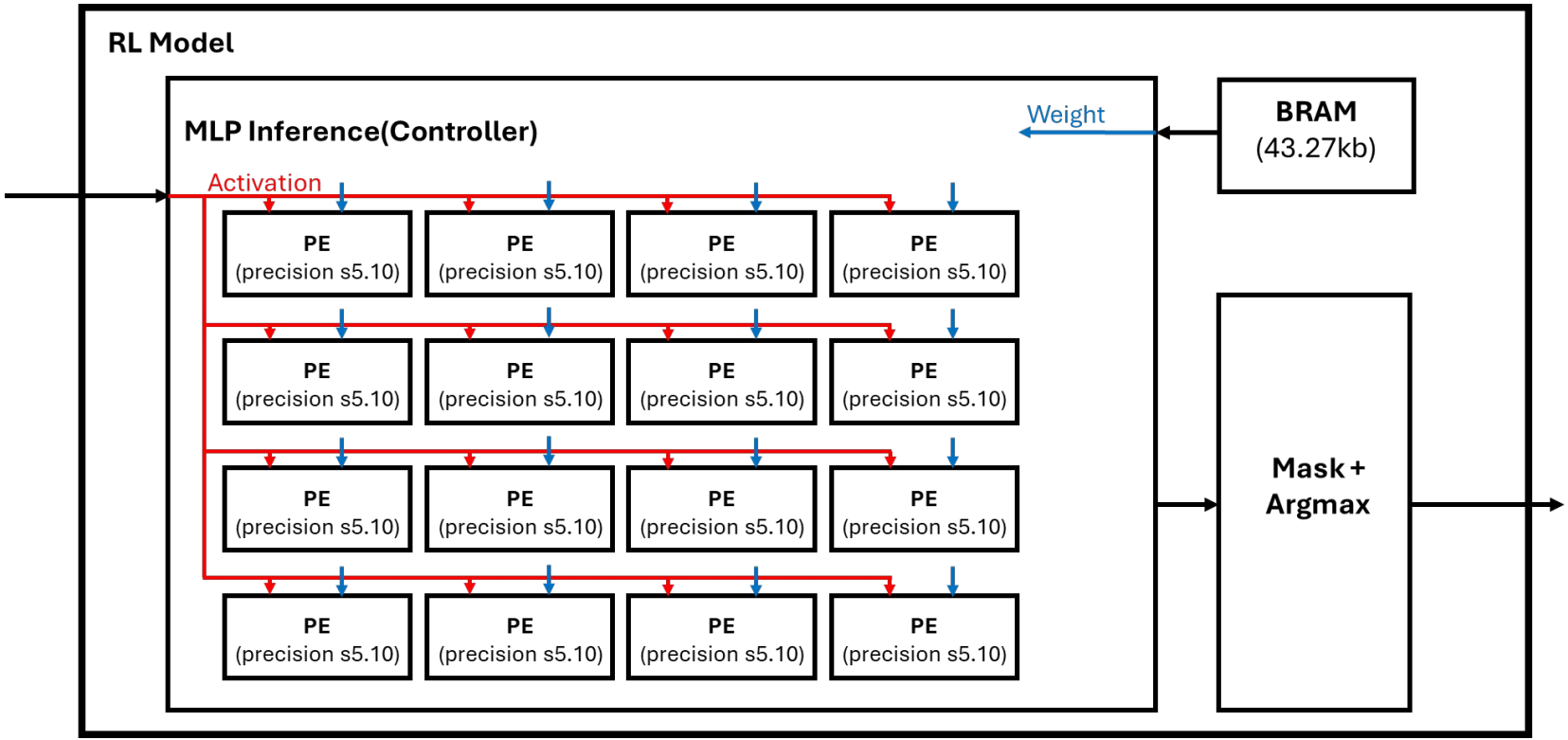
RL Model-Hardware

Advantages

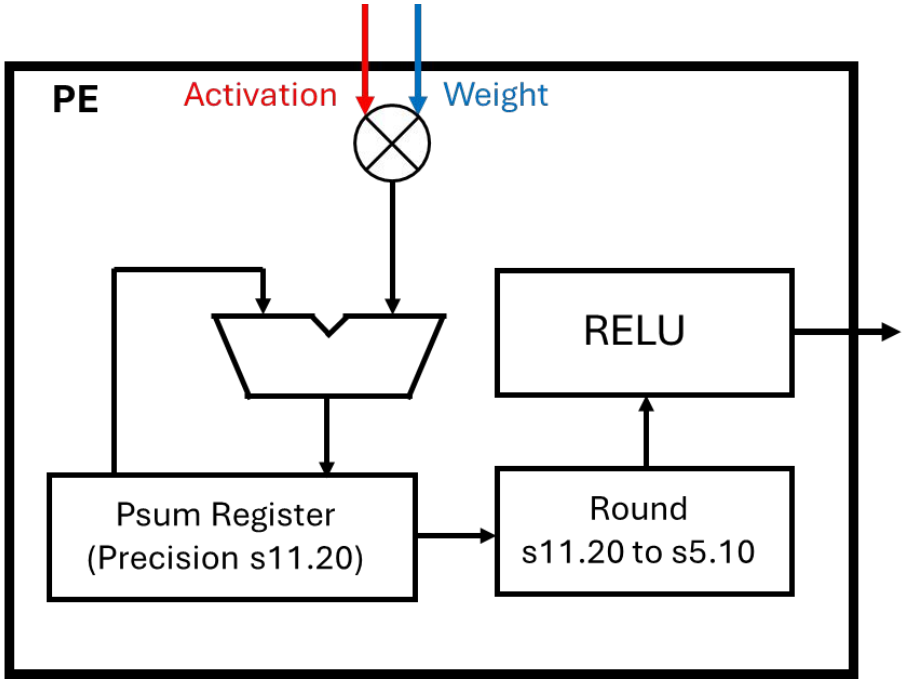
- Fixed-point (Quantization FP32→s5.10)
- Parallel PEs
- Hardware Masking
- Faster Inference Speed



RL Model-Hardware Architecture



RL Model-Hardware-PE (Processing Element)



1.Generate Address

2.Access weIght

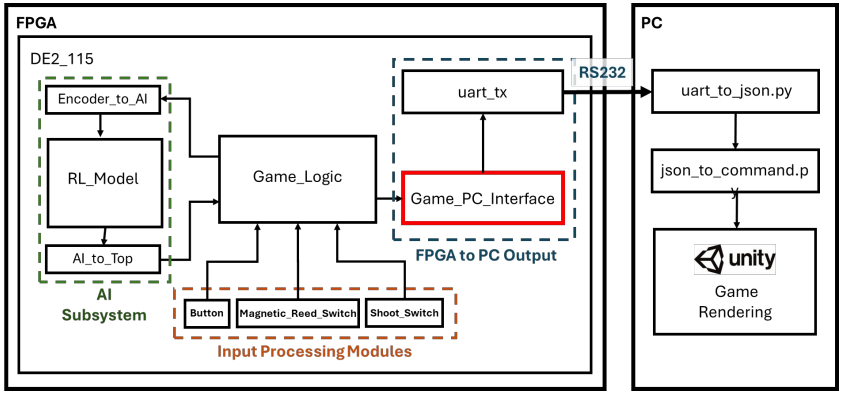
3.Multiplication

4.addition

Game PC Interface

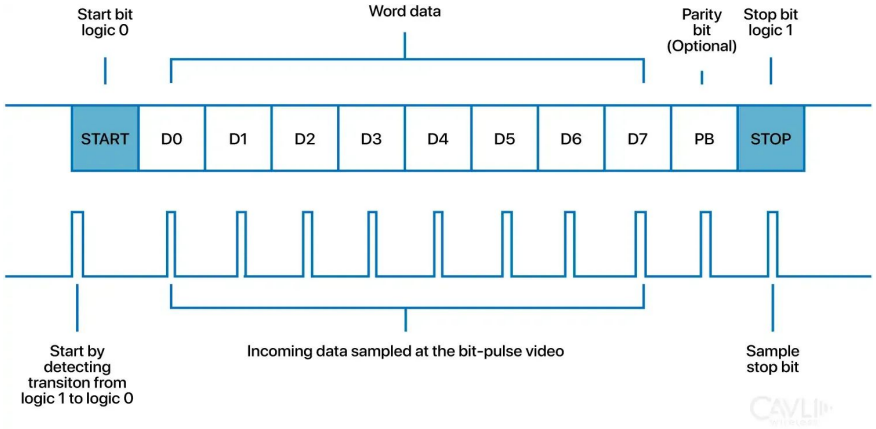
Game PC Interface

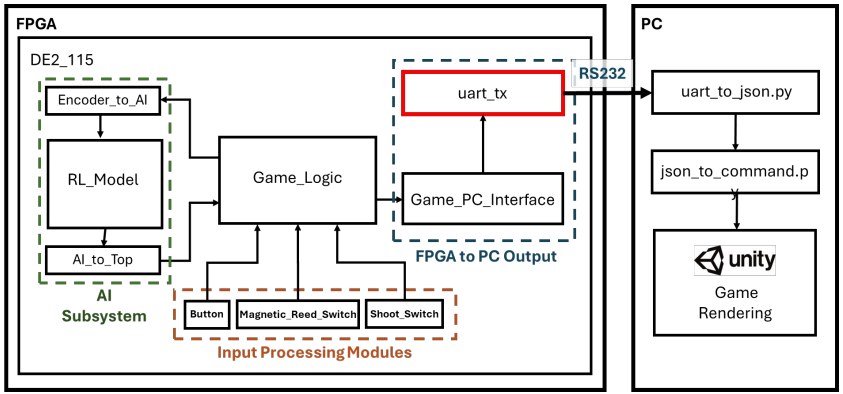
- **Wrap GameState into 14 byte**
  - Byte Index 0 : Header (0xA5)
  - Byte Index 1~13: Gameinfo



Uart TX

- **Baud rate:115200**
- **Transmitt the gamestate packet to PC**

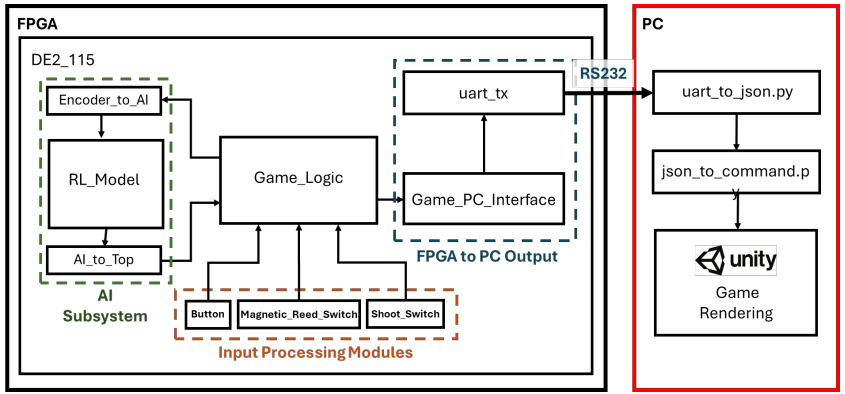




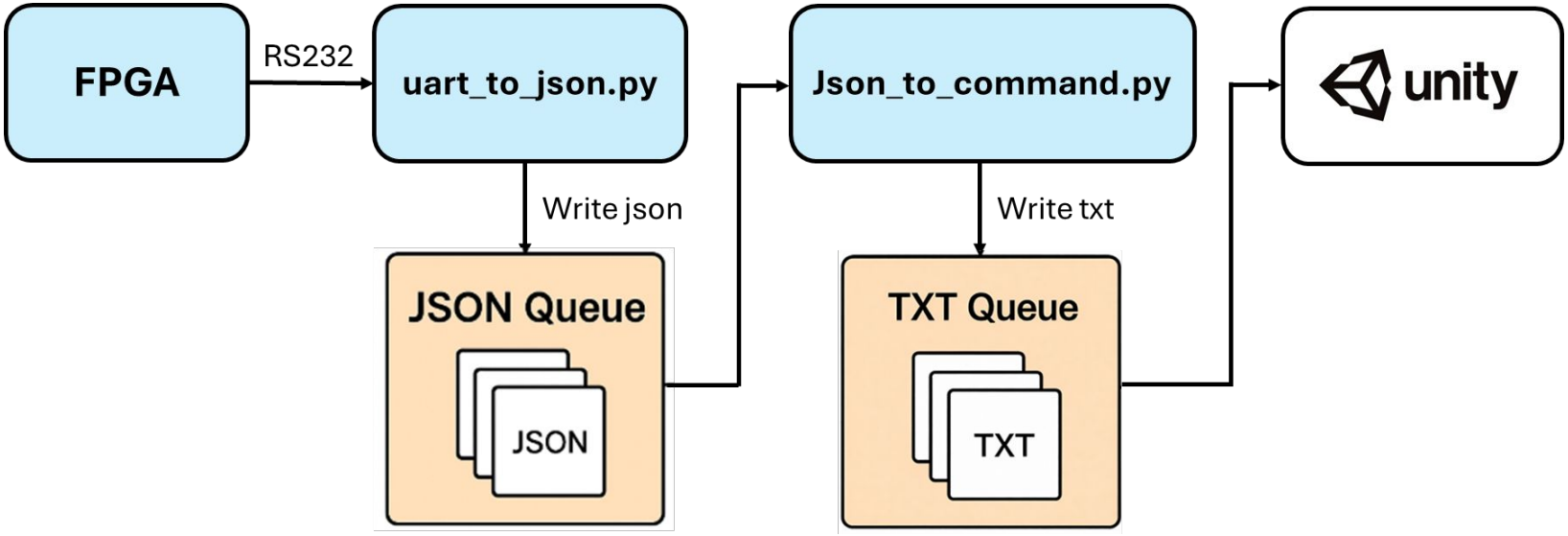
PC Rendering the Game

- Use Unity Engine to Render our game

- Solve the problem that Inference speed is much faster than game rendering speed



Flow Chart

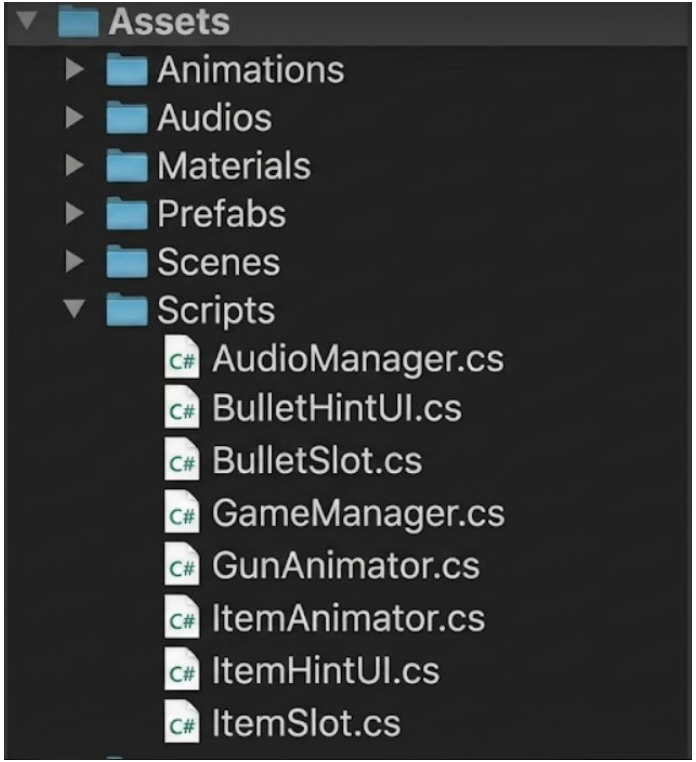

FPGA
RS232
uart_to_json.py
Write json
JSON Queue
JSON
Json_to_command.py
Write txt
TXT Queue
TXT
unity

Python code

- **uart_to_json.py**
  - Listens to COM4 (Baud 115200) for hardware signals
  - Decodes 14-byte hex packets (Header 0xA5) using bitwise operations to extract game variables.
  - Serializes parsed data (Winner, HP, Items, Bullets) into sequential JSON files (state0.json, state1.json)

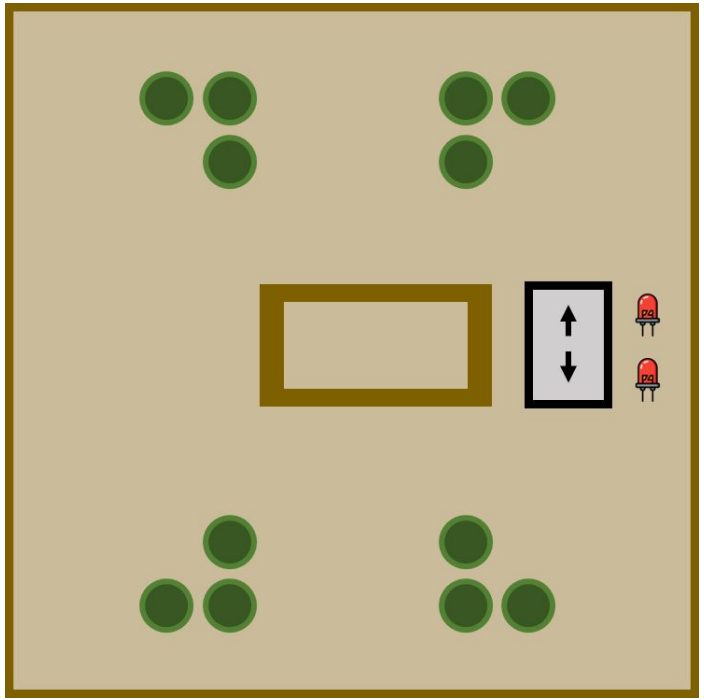
- **json_to_command.py :**
  - Continuously watches for new JSON logs generated by the UART script.
  - Compares old_state vs. new_state to detect changes in HP, item usage, or turn switching.
  - Translates state changes into specific string commands (e.g., "ML" for Magnifier, "Q" for Shooting) for Unity to render.

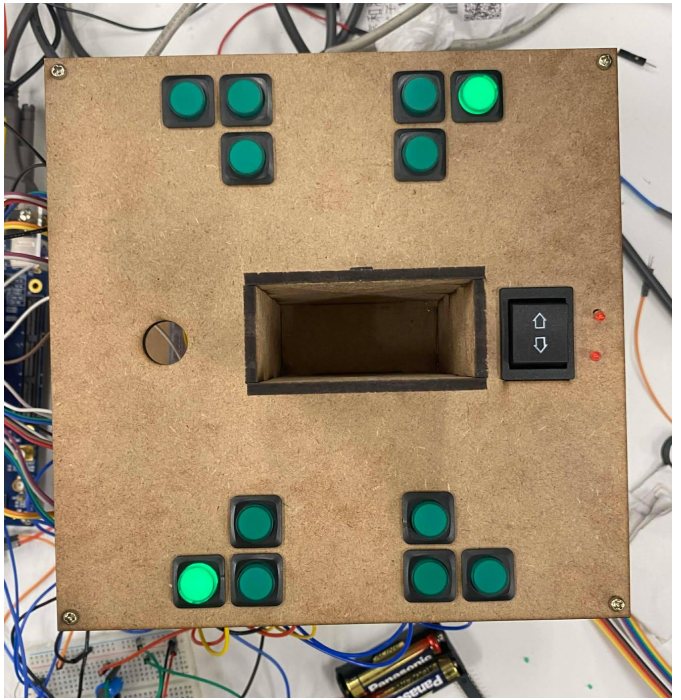
Unity Render



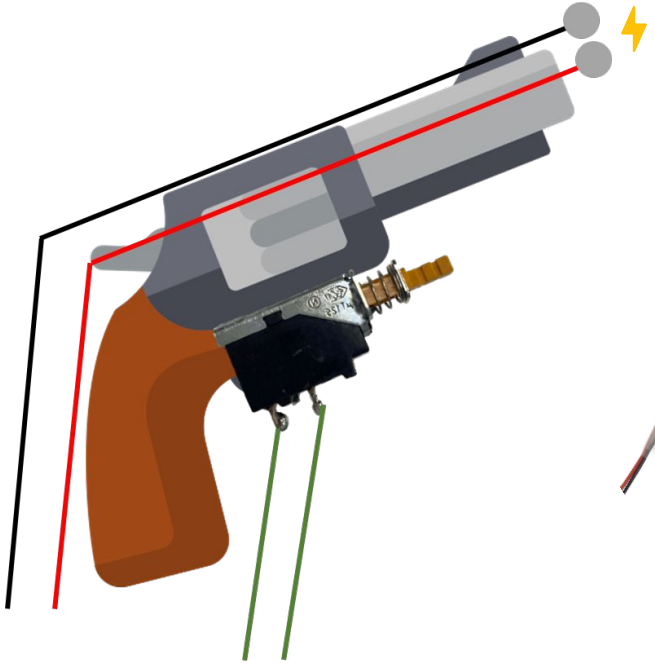
Human Player UI - Circuit

Mechanical Structure - Game Platform

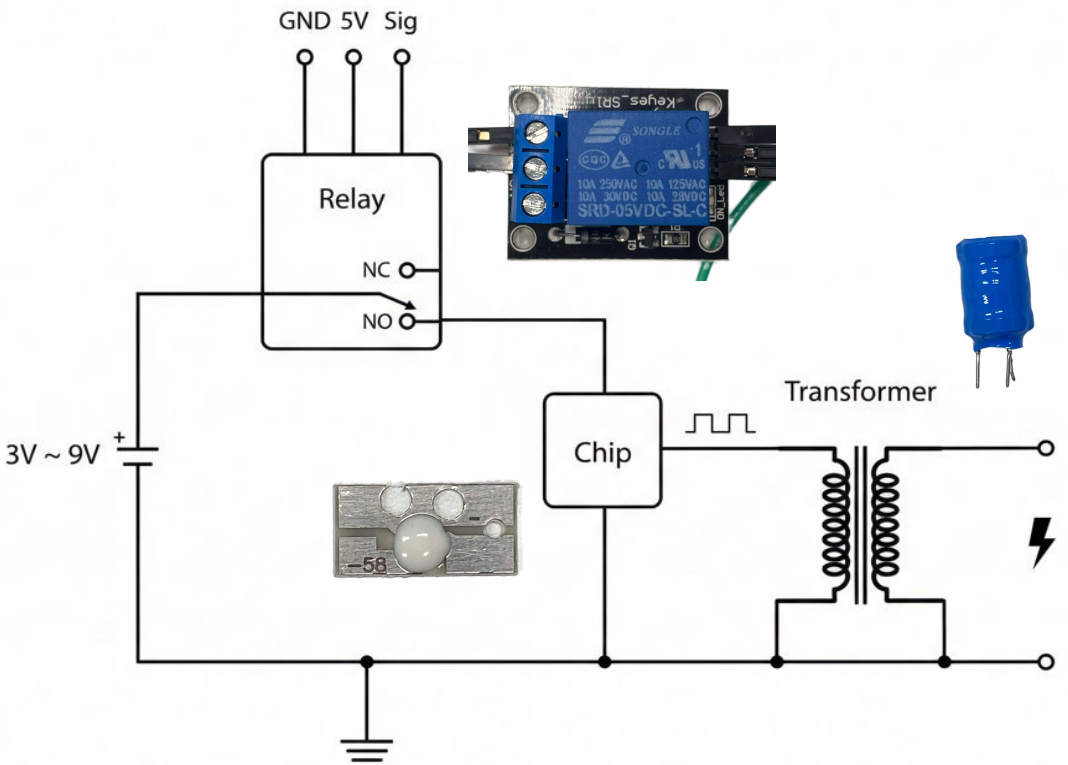




Mechanical Structure - Gun



Shocking Gun Circuit

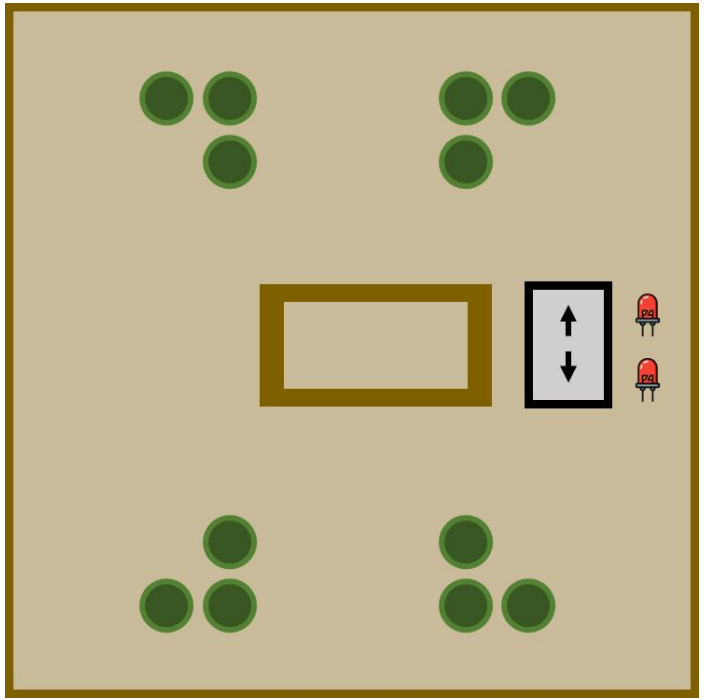


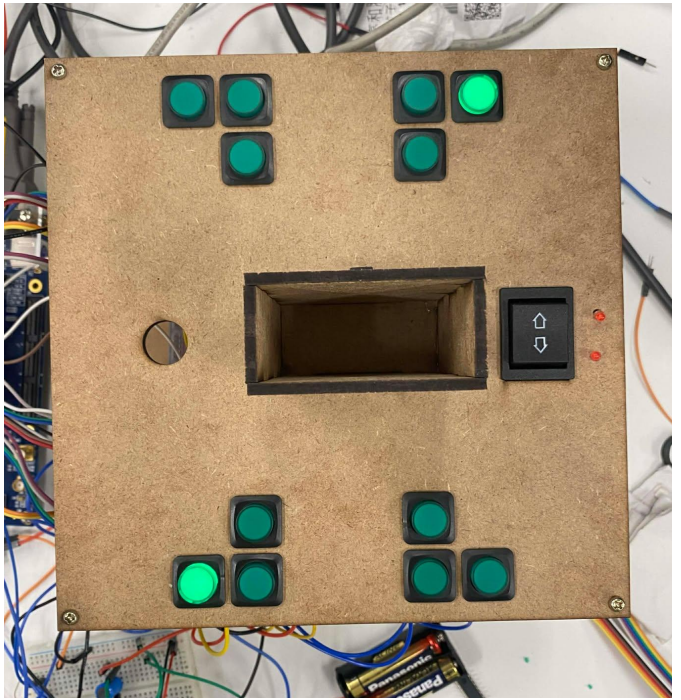
Human Player UI - Signal Process

Input Processing Modules

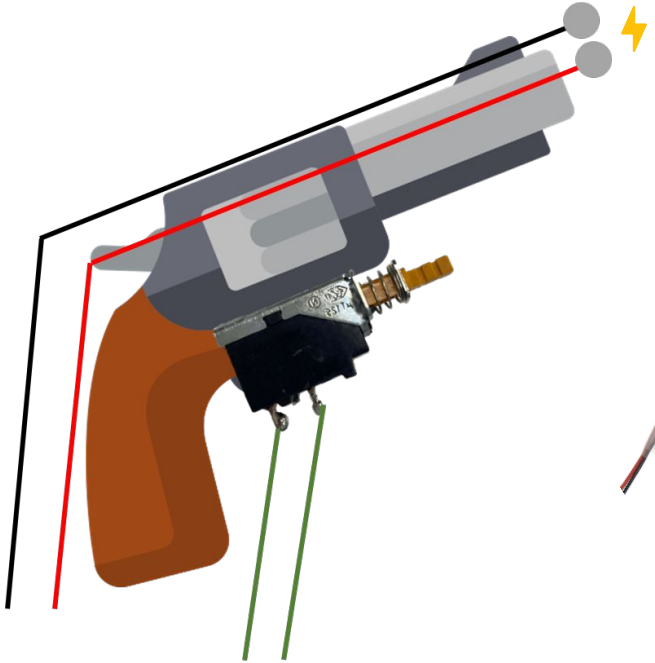
- Modules for processing human player inputs :
  - LED buttons -> item selection
  - Trigger button -> trigger detection
  - Magnetic reed switch -> shooting stage detection
  - Switch -> target selection

Mechanical Structure - Game Platform



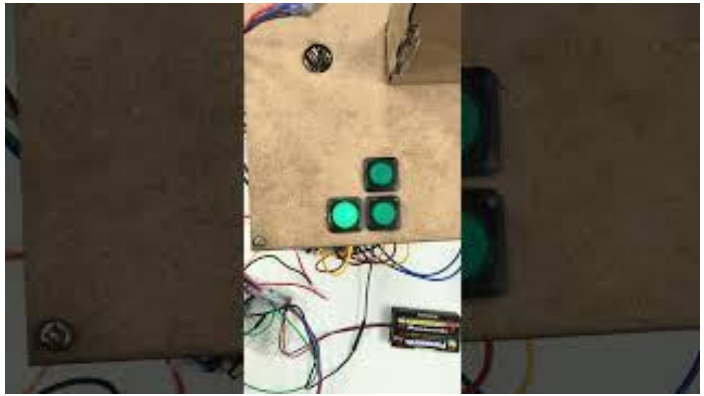


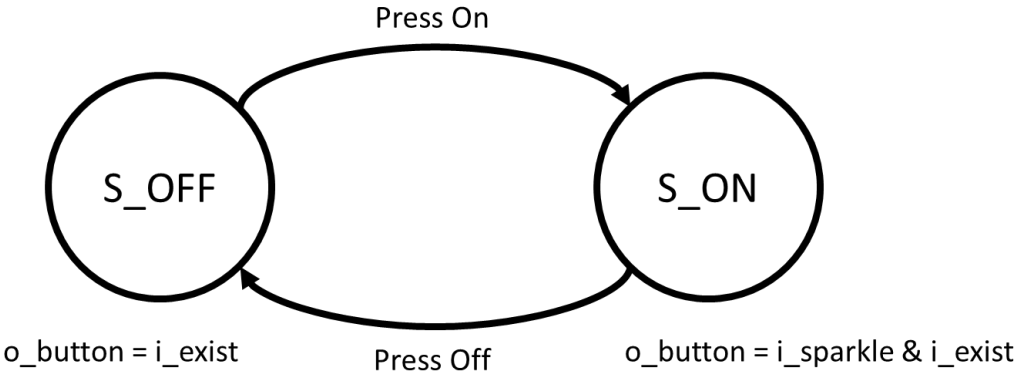
Mechanical Structure - Gun

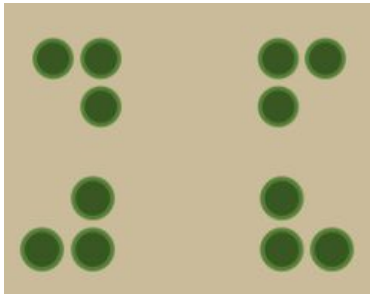


LED Buttons

- Item selection
- o_button : LED signal
- i_sparkle : Square wave
- i_exist : Existence of the item



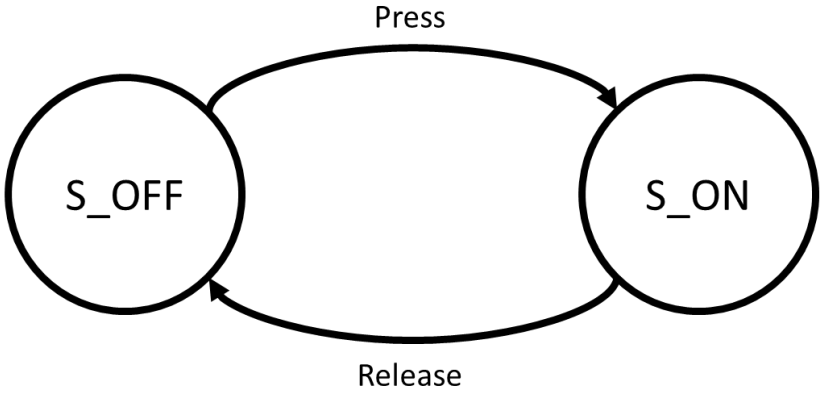


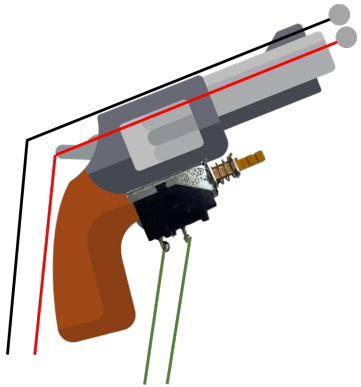


Trigger Buttons

Trigger Buttons

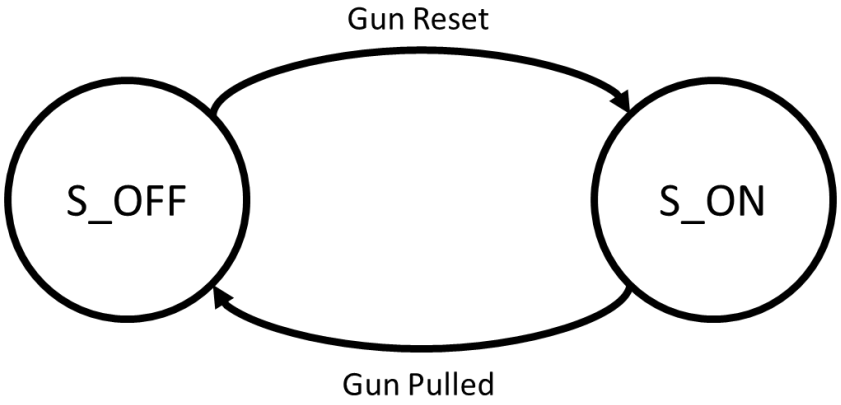
- Gun trigger detection
- Long wires make the signal glitches severely
- The button input signal is debounced
- The gun is triggered when the button pressed for a short amount of time





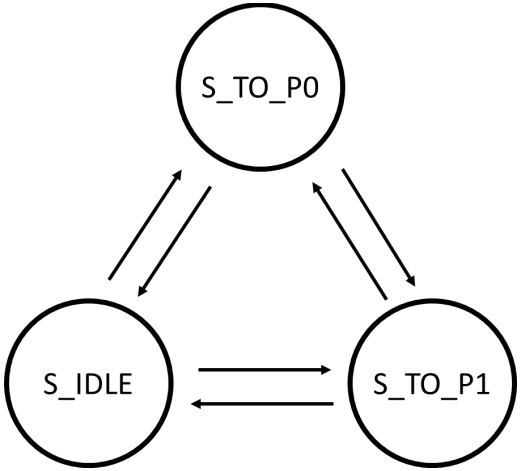
Magnetic Reed Switch

- Shooting stage detection
- The signal is debounced to keep the state stable
- When the gun’s is pulled and the magnetic on the gun is removed from the sensor for a short amount of time, player into shooting stage



Switch

- Target selection
- The LED will flash correspondingly
- When the gun is triggered, the state will be reseted to IDLE state



DEMO


HEALTH
0 4