FINAL PROJECT DEMO

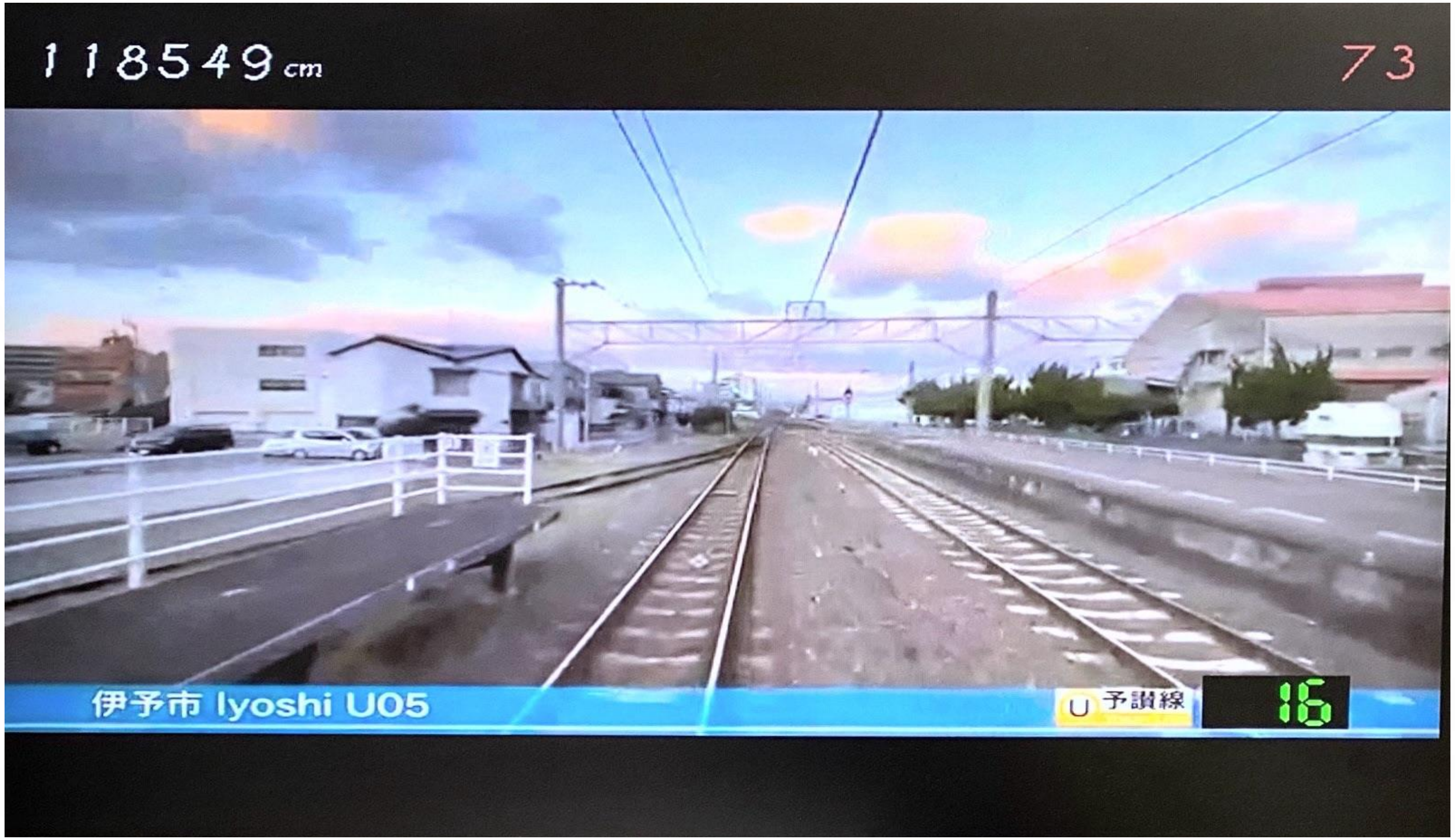
DC LAB SPRING 2025

TEAM 09 / 丁睿濂、林靖軒、劉代恩

OUTLINE

- Introduction
  - Train simulator
  - Speed control
- Implementation Details
  - Block diagram
  - Flow chart
  - Connections
  - Memory
  - Display
  - Video play speed control
- Novelty

TRAIN SIMULATOR



remaining distance

remaining time

speed

BASIC FUNCTIONS

- Input:
  - Joystick (acceleration, brake, …) → FPGA
  - Computer (video frames & speed labels) → FPGA
- Function:
  - Receive control from joystick and calculate speed correspondingly
  - Communicate with PC to receive frames and display video accordingly
- Output:
  - FPGA → Monitor (show video, speed, distance, and score)

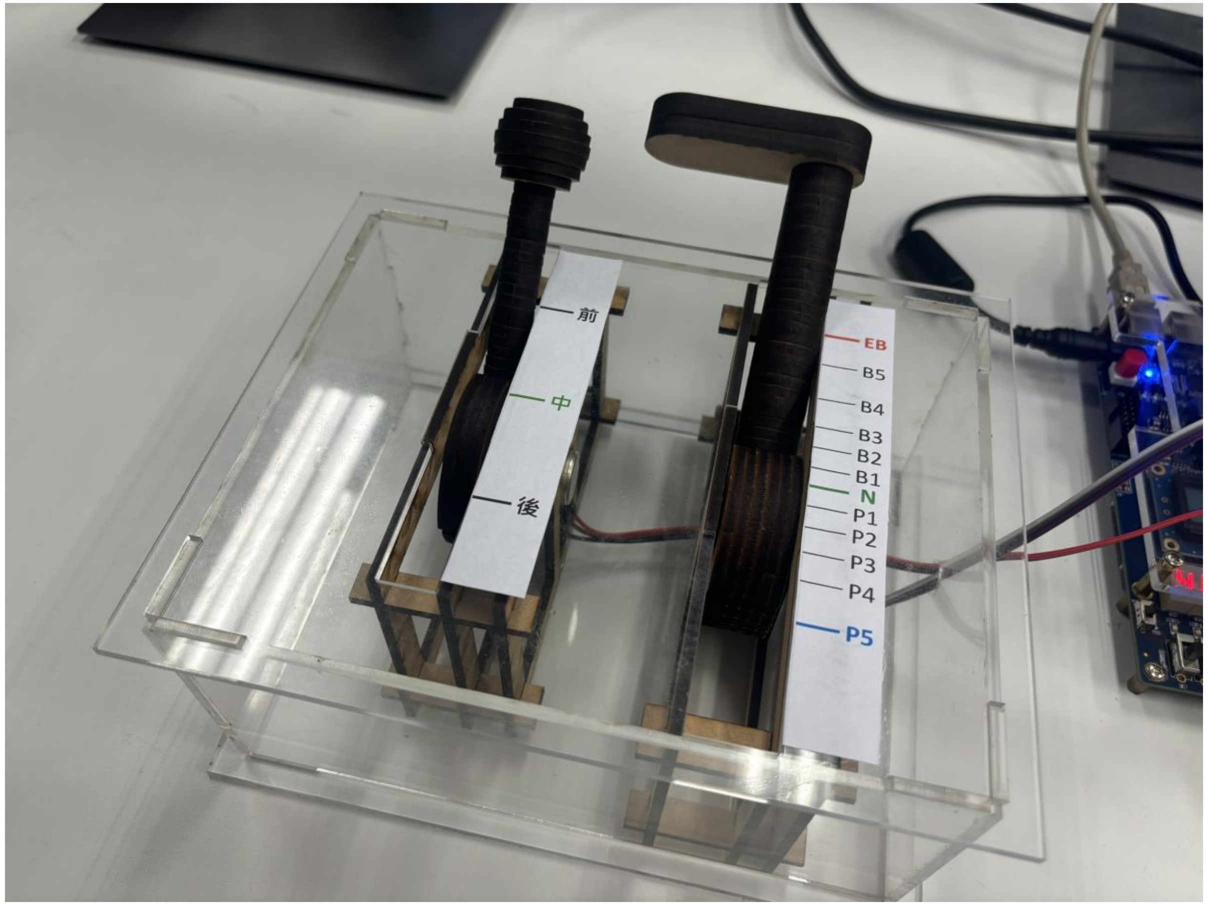
SPEED CONTROL

- control by joystick
- acceleration and brake

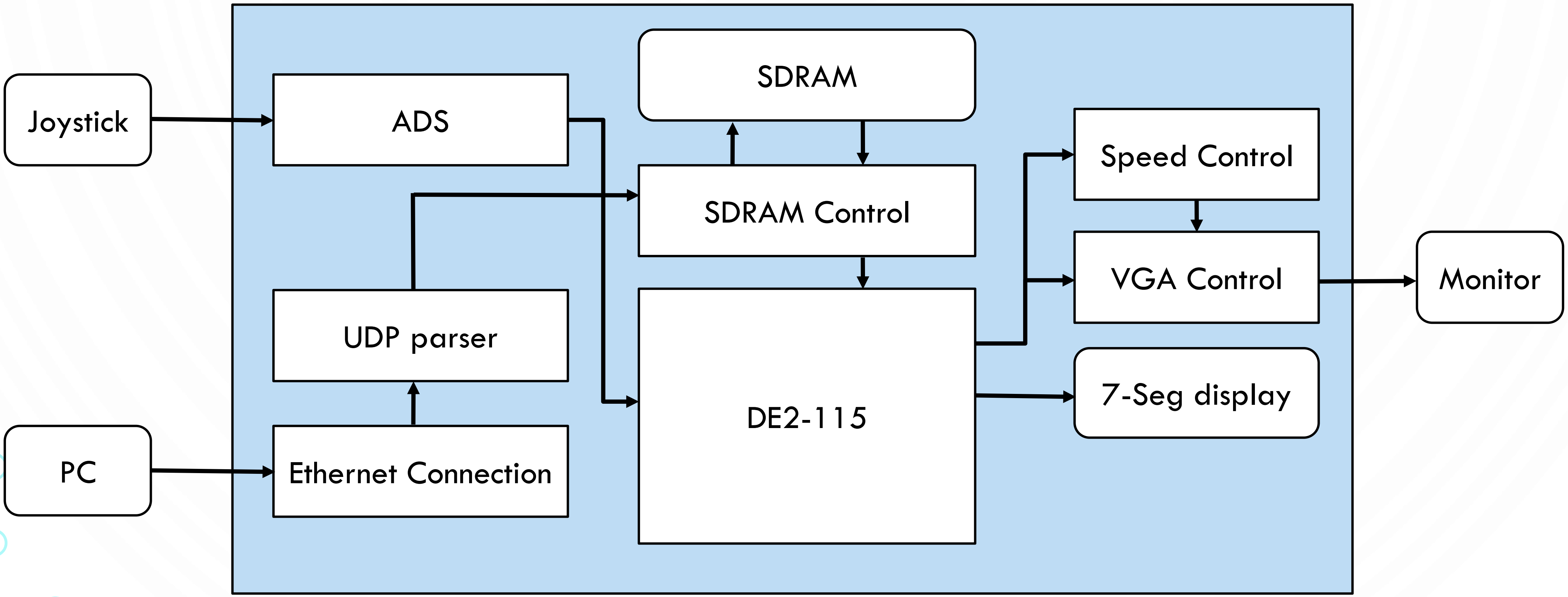
- Joystick:

  Gears + Clip

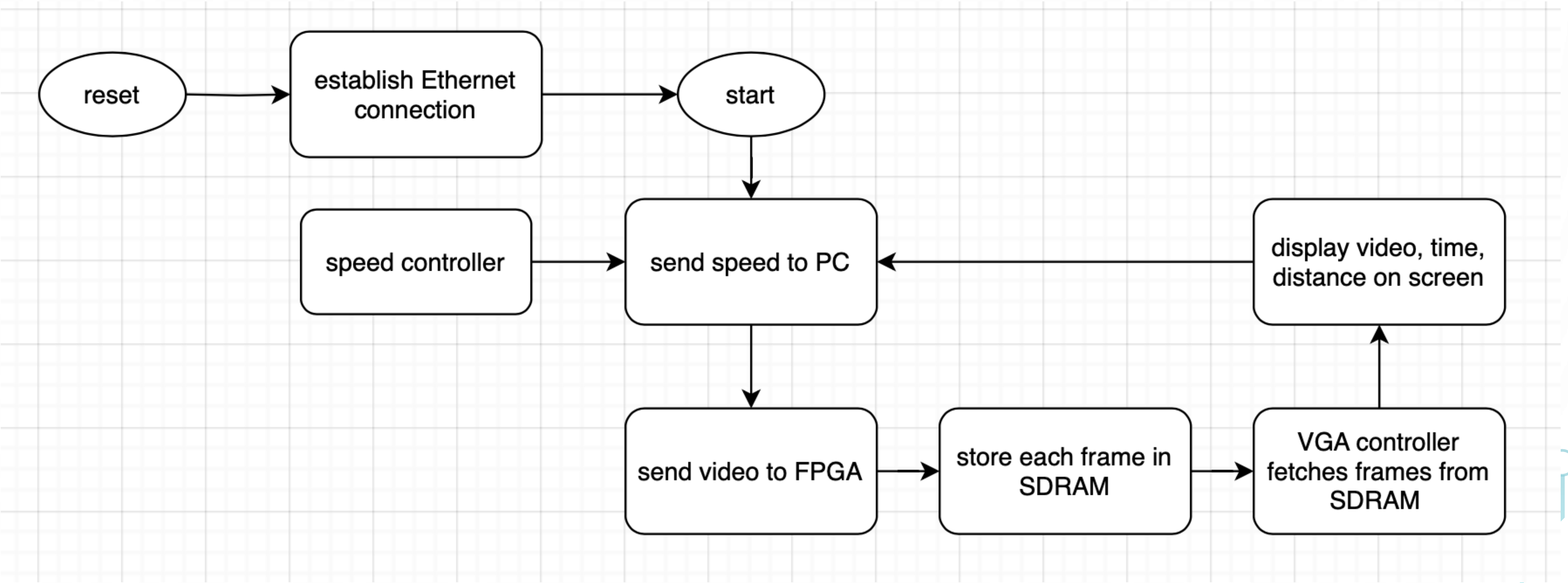
  → Discrete value control



BLOCK DIAGRAM



FLOW CHART


reset
establish Ethernet connection
start
speed controller
send speed to PC
display video, time, distance on screen
send video to FPGA
store each frame in SDRAM
VGA controller fetches frames from SDRAM

CONNECTIONS

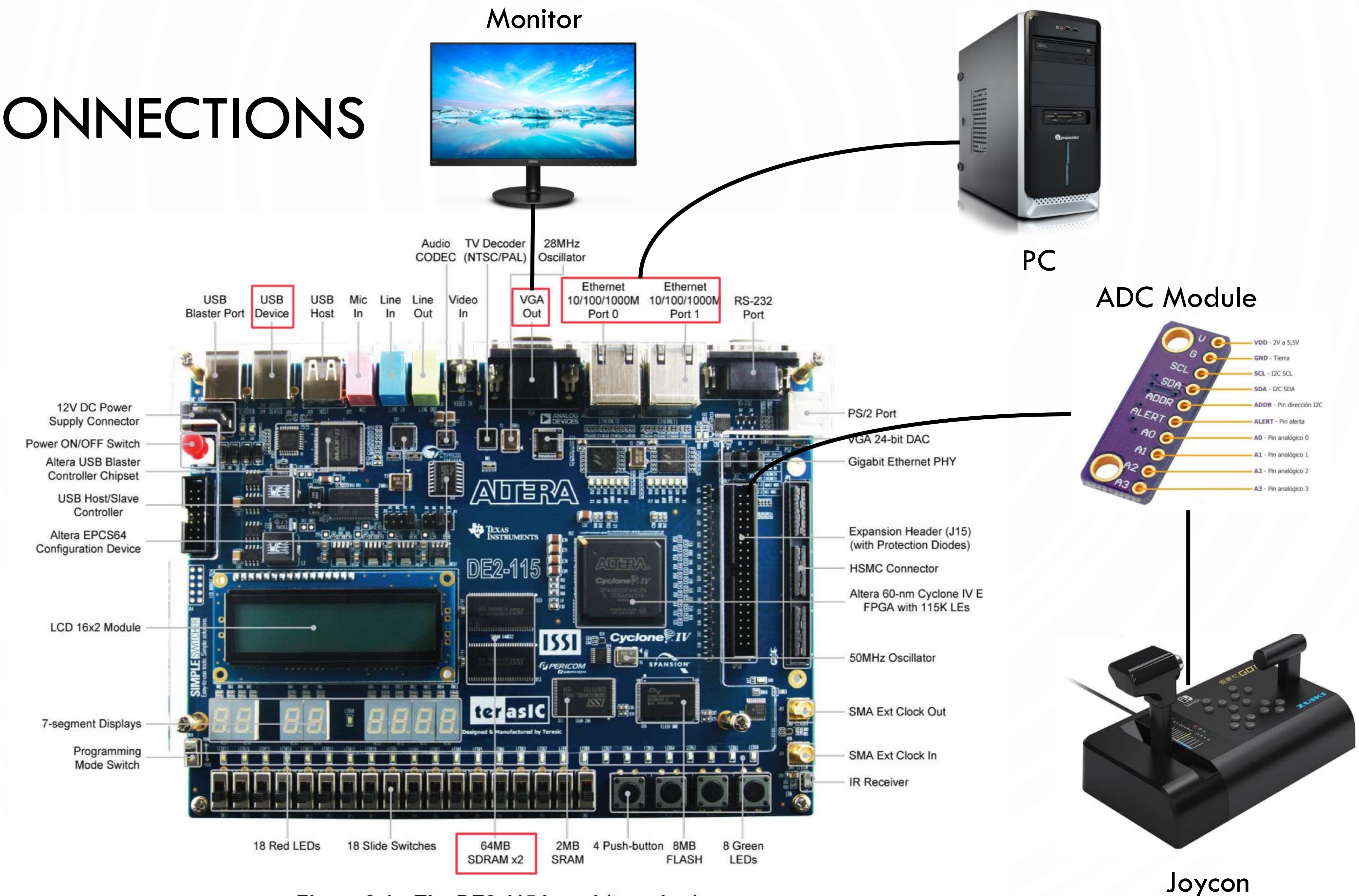


Figure 2-1 The DE2-115 board (top view)

CONNECTIONS

- GPIO (joystick → FPGA): control playback speed by acceleration and brake
    - Use potentiometer and ADC module (ADS1115) to communicate with FPGA
    - Support I2C protocol
    - Use Arduino and logic analyzer to understand the protocol:
        - Initialize
        - Set Config Register
            - Select Config Register
            - Write Config Register Settings (channel)
            - Read Config Register Settings
        - Read Conv Register
            - Select Conv Register
            - Read Conv Register

CONNECTIONS

- Ethernet (PC → FPGA): send video frames and return speed labels
  - ○ https://github.com/alexforencich/verilog-ethernet
  - ○ Each frame is converted into RGB24 format, pack multiple pixels, then send data packet to FPGA
  - ○ Parse data packets on FPGA based on data format to handle data correctly
  - ○ Add header to packets can help FPGA distinguish different data type
  - ○ IP configuration:
    - PC: 192.168.50.7
    - FPGA: 192.168.50.8

    Both PC and FPGA have to be under the same subnet.
  - ○ Also supports FPGA → PC connection (send speed labels)

CONNECTIONS

- VGA (FPGA → Monitor): output corresponding video
  - Support resolution: 640*480, 30FPS
  - Able to display numbers, such as speed, distance and time on screen

MEMORY

- SDRAM
  - ○ Store video frames in SDRAM
  - ○ Ethernet and VGA module requires different clock rate, so we need a buffer to store image
- Register
  - ○ Store digits for display

DISPLAY

- Value of basic features
  - Speed: calculated from current speed and joystick's value
  - Distance to next station: send labels from PC through Ethernet
  - Remaining time: counted in FPGA
- Display
  - Store images of 0~9 in registers
  - Cover VGA's pixel at image's position with image's pixel matrix.
  - Scale RGB value with numbers, e.g. higher speed, less time → more red

    desired stop distance → green

VIDEO PLAY SPEED CONTROL

- Label each frame with its original speed label → video speed
- Player control acceleration and brake to adjust speed → user speed
- If user speed > video speed
  - use higher FPS
  - skip some frames if FPS exceeds its upper limit (>70)
- If user speed < video speed
  - use lower FPS
  - send the same frame if FPS too slow (<30)

NOVELTY

- Flexibility
  - Download any video and start playing after few processing
  - Deal with protocols between FPGA and joystick
- Low latency & real-time data handling
  - FPGA provides faster response times, which is a fit for real-time data processing

REFERENCE

- DE2-115 User Manual
  - https://www.terasic.com.tw/wiki/images/f/f2/DE2_115_manual.pdf
- Ethernet
  - https://medium.com/@Frank_pan/how-to-use-ethernet-components-in-fpga-altera-de2-115-26659da06362
  - https://github.com/alexforencich/verilog-ethernet
- SDRAM
  - 邏輯電路設計DE2-115實戰寶典
- Vehicle Speed Estimation
  - shafu0x/vehicle-speed-estimation: Vehicle Speed Estimation from Video using Deep Learning and Optical Flow in PyTorch.