6.111 Introductory Digital Systems Laboratory

Final Project Proposal: AirPong

Sabina Chen | Bret Heaslet | David Mueller

30 October 2018

1 Project Overview

For our final project, we will be moving the game of pong into the real world. Pong is an arcade game that simulates table tennis. Players compete against each other by controlling in-game paddles on the left or right side of the screen to hit a ball back and forth. Points are earned when a player fails to return the ball to the other.

In our real-world version of Pong, *AirPong*, we will be using a drone as the ball and people as the paddles. A camera hanging from the ceiling will use computer vision to track the real-time locations of the drone and players within a defined playing field. The physical positions of the players will represent the locations of the paddles, and the physical position of the drone will represent the location of the ball. We will use an FPGA to process camera and sensor data, create game logic, and define controls to send commands to the drone in 3D space. While the game is in motion, a visualization of the real world pong game will appear on the VGA monitor.

2 Block Diagram

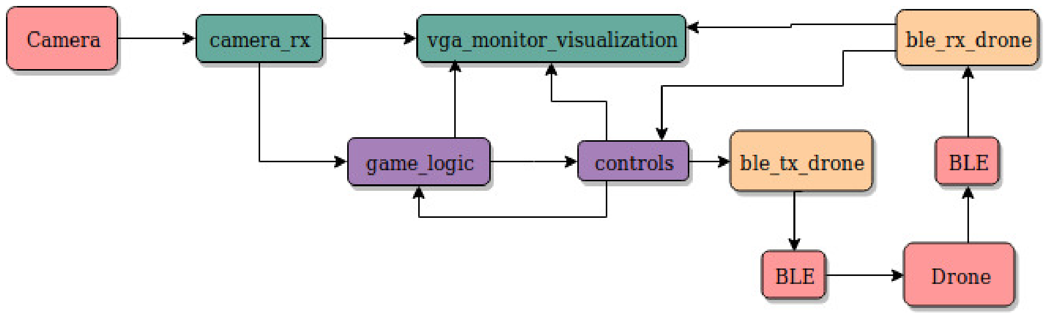

Figure 1: AirPong Initial Block Diagram

3 Modules

3.1 Communications

3.1.1 ble_rx_drone

Functionality:
The ble_rx_drone module receives data from the HM-10 Bluetooth Low Energy (BLE) communications module that interfaces with the drone. This module will sample and concatenate the data bits into packets. Once all data bits have been received, the module will translate the packets into meaningful data. (e.g. accelerations, quaternions, etc.)

Inputs/Outputs:
This module will have a clock input in order to sample the data at the proper frequency, as well as the data bit input from the BLE module. As for outputs, there will be three accelerations, one for the x, y, and z, respectively, three coordinates, again, one for the x, y, and z, one sonar reading (height above ground), and one barometer reading (ambient pressure).

Complexity:
This module will not store anything in memory past what is necessary for operation. The largest register in the module will be 12 bytes. This single

data register will store the data bits as they come in and then the data will be extracted from them on transmission completion. The data will then be stored and forwarded on to the next module(s).

Test Plan:
This module will be tested in several phases. Initially, test benches will be used to simulate with ModelSim. Once, we've made all the headway available from simulation, we will test the module on the FPGA, communicating with the BLE module. Once communication can be confirmed with the BLE module (the module sends acknowledgements as commands are sent), we will connect a smartphone to the BLE module and send messages back and forth between the smartphone and FPGA to ensure proper communication. Lastly, we will run an operational test with the drone. Sending the "take off" command, the drone will hover, then sending the "Land" command, the drone will land.

Person Responsible: Bret

3.1.2 ble_tx_drone

Functionality:
The ble_tx_drone module will take motor speeds from the controls module, translate them into a BLE packet and then transmit that packet one bit at a time over serial connection with the HM-10 BLE chip.

Inputs/Outputs:
As input, this module will have both a clock and the 4 motors speeds from the controls module. As output, the module will produce a command packet that will be transmitted one bit at a time, using asynchronous serial communication, to the BLE module and then forwarded onto the drone.

Complexity:
The intricacies of this module lie in translating the motor speeds to proper commands that the drone can understand, as well as, packetizing the command properly. The commands for the all of Parrot's drones are well documented, so this should only be a tedious, not impossible task. Packetizing the command properly will require a few iterations of trial and error before we get it completely correct.

Test Plan:
This module will be tested in a similar fashion and alongside the ble_rx_module. We will use testbenches with ModelSim to debug. When we've accomplished all we can with simulation, we will begin testing communication with the HM-10 BLE module. After communication is established with the HM-10, we will use a smartphone to receive and send data to and from the HM-10 module/FPGA. The last step of testing will be an operational test with the drone. Send a command to make it hover and then land.

Person Responsible: Bret

3.1.3 camera_rx

Functionality:
This module uses computer vision to process camera information in order to identify the locations of the drone and paddles.

Inputs/Outputs:
The input will be pixel data from the camera, and the output will be the x and y coordinates of the drone and paddles.

Complexity:
The camera will be hanging from the ceiling in order to get a top-down view of the game. The camera will be wired directly to the FPGA. The module will process the pixel data from the camera by first using a computer vision algorithm to define the playing field via corner markers on the ground. Within this playing field, we will then find and output the x and y coordinates of the drone and the paddles. We plan on having the computer vision algorithm identify the corner markers, drone, and players via color masking by sticking fiducials or fluorescent tape on the objects we want to identify.

The top left corner of the playing field will be (0,0). The width and height of the playing field will be discretized into 1023 by 767 respectively, in order to match the pixel discretization of the VGA monitor for easier processing of coordinate information between the VGA monitor and the camera.

Test Plan:

This module can be tested independently by wiring the camera to the FPGA and verifying the correctness of the x and y coordinates of the drone and paddles by outputting the values on the labkit hex display.

Person Responsible: Sabina

3.2 Data Processing

3.2.1 game_logic

Functionality:
Use current position of the drone and paddles to update game state. Game state determines commanded velocities used by the `controls` module. Send state information such as puck velocity to `vga_monitor_visualization`. Update and keep track of player scores.

Inputs:

- `drone_x, drone_y`
  Drone x,y position in discrete coordinates.
- `paddle_1x, paddle_1y`
  `paddle_2x, paddle_2y`
  x,y coordinates for player 1 and 2's paddles.

Outputs:

- `drone_vx, drone_vy`
  x,y vector components of commanded drone velocity in *world frame*.
- `state`
  Current game logic FSM state.
- `score_1, score_2`
  Player 1 and 2's scores.

Complexity:
Nearly identical to the Pong lab. Group already has code implementing 2-player pong and a parameterized architecture which can be leveraged.

Test Plan:
Use test benches already developed from Pong Lab.

Person Responsible: David

3.2.2 controls

Functionality:
Provides full control of all 4 motors based on commanded velocity and altitude. The commanded velocity and altitude serve as reference signals for closed-loop control which determines commanded motor speeds. Pitch and roll angles are also estimated based on sensor inputs, and the these are compared against varying commanded pitch and roll angles determined by the control loop. This module also controls drone takeoff and landing sequences.

Inputs:

- `ref_vx, ref_vy`
  x,y components of commanded velocity vector, in *world frame*. Needs to be converted to body frame coordinates.
- `accel_x, accel_y, accel_z`
  Accelerations reported by the x,y,z accelerometers.
- `gyro_x, gyro_y, gyro_z`
  Angular velocities reported by the x,y,z gyroscopes.
- `altitude`
  Distance above ground reported by the ultrasonic sensor.
- `pressure`
  Ambient pressured reported by the barometer
- `flow_x, flow_y`
  Optical x,y flow

Outputs:

- `pitch, roll`
  Estimated pitch and roll angles, in radians.

- `motor_1, motor_2, motor_3, motor_4`
  Commanded speeds for all 4 drone motors.

Complexity:
High computational complexity. Algorithms are be based in linear algebra, so many matrix multiplications are needed. Computations are used to compute the commanded speed of mechanical motors, so there is no concern of excessive pipeline latency.

Algorithms will also be moderately complicated, but several students from 16.30 will be helping with this aspect, so the risk here is greatly reduced.

Test Plan:
The controller will be modeled and simulated using Simulink. As the controller is developed in Verilog, its outputs will be verified against the simulation inputs and outputs.
Person Responsible: David

3.2.3 vga_monitor_visualization

Functionality:
This module displays a 2D visualization of the real-world game of pong onto the VGA monitor.

Inputs/Outputs:
The inputs of this module are the x and y coordinates of the drone and paddles from the `camera_rx` module, game state information from `game_logic` module, and flight telemetry from the `ble_rx_drone` and `controls` modules.

Complexity:
While the game is in session, we plan to show a live video stream of the game from the camera onto a vga monitor. Overlayed on top of the live video feed will be a 2d visualization of the ball and paddle, as well as game state information, such as an arrow showing the current direction and velocity of the ball/drone. We will also display drone telemetry and estimated state information in the corner of the monitor, such as accelerometer, gyro, sonar, optical flow, and estimated pitch and roll.

Ultimately, we aim to augment the game visualization on the VGA so that bystanders can explicitly see the game logic in action. This module will also be an invaluable tool for debugging the game implementation if needed, as all the game states are visualized explicitly on the monitor. Tasks that could be complex include rendering 2d visualization/alpha blending on top of the live video feed, and displaying text on the screen.

Test Plan:
This module can be tested by using the camera and vga monitor to display the live video feed directly. Data from other modules can be displayed and tested as each of the respective modules become written.

Person Responsible: Sabina

4 External Components

We plan on using four external devices: one camera, one drone, and two BLE devices.

4.0.1 Parrot Mambo Mini-drone

The parrot mambo is a small 8 inch by 8 inch mini-drone that can be controlled using a game controller or smart-phone application via Bluetooth. We plan on communicating to the parrot mambo using two BLE devices, one transmitter and one receiver. The BLE devices will be connected directly to the FPGA.

4.0.2 Camera

We will be borrowing a camera that can be wired directly to the labkit FPGA.

4.0.3 BLE

The BLE module will transmit/receive BLE signals to/from the drone. The particular BLE module we will be using is the HM-10 BLE module. This module takes UART as input and then uses the L2CAP protocol to communicate over BLE.