TEAM 18 – VALIDATION & INTEGRATION SCHEDULE

Milestone 3

TESTABLE COMPONENTS:

- Routing software (Rpi)
- Line Following Array (pic32)
- AI Engine / Pathfinding (RPi)
- PixyCam I/O software (RPi)
- PacMan Rover GUI (Rpi)
- Statistics Display (Rpi)

ROUTING SOFTWARE VALIDATION - JOHN:

(See validation checksheet)

- Test Case #1, Subroutine Test:
    - SHOW that the subroutines for filling rover data frame and opening sockets work correctly.
    - CORRECT if sockets for read/write open correctly in 5 tries
    - CORRECT if rover frame filler returns the correct data frame for multiple speeds and directions
    - VERIFIED by matching strings sent to rover sockets to a known correct string
        - Test program waits for I/O from receiving component and does comparison, pass/fail printed to output
- Test Case #2, Single I/O Tests:
    - SHOW that server can handle I/O between different routing paths
    - CORRECT if component that was intended to receive data did receive correct data after send from another component
        - e.g. If Pacman sends a debug message it should be received by statistics
        - e.g. If the GUI sends a left or right it should package the data frame correctly and send to Pacman rover

        - e.g. If the AI engine sends a left or right it should package the data frame correctly and send to Ghost rover
    - VERIFIED by pass/fail comparison in test program
- Test Case #3, Continuous Input Check:
    - SHOW that the server can handle lots of inputs quickly
    - CORRECT if no failures occur in test program, and no errors or exceptions are raised on the server
    - VERIFIED by zero error count at end of test program

STATISTICS VIEW SOFTWARE VALIDATION – JOHN:

- Test Case #1, Rover Direction:
    - SHOW that statistics view can receive and display left/right information about Pacman and Ghost
    - CORRECT if data received and displayed
    - VERIFIED by visual confirmation in output log
- Test Case #2, Rover Debug:
    - SHOW that statistics view can receive and display debug information in raw format and add to tracked stats
    - CORRECT if debug received and displayed
    - VERIFIED by visual confirmation of output log and tracked stats displayed
- Test Case #3, Message Missing:
    - SHOW that statistics view can correctly detect missing message numbers from rover
    - CORRECT if number of missing messages is displayed in tracked stats view
    - VERIFIED by visual confirmation of tracked stats view

LINE FOLLOWING ARRAY VALIDATION - BEN:

- Test Case #1:
    - SHOW the LFA can detect the presence or absence of a black line
    - CORRECT if the data bits on the LFA are only those which are over the line and no others

    - VERIFIED by visual confirmation of which sensors are occluded vs. what data is being returned

AI ENGINE / PATHFINDING VALIDATION - BEN:

- Test Case #1:
    - SHOW that the AI algorithm can find the shortest path between two nodes using the A* (A-star) search
    - CORRECT if for each map, starting node, and end node, there will be a shortest path
    - VERIFIED by testing multiple maps and goals to ensure handling of edge cases. (More complex maps could be verified using an open-source implementation of the algorithm)

PIXYCAM I/O SOFTWARE VALIDATION - ANDREW:

- Test Case #1, Data Input:
    - SHOW that fake PixyCam data can be read from a text file
    - CORRECT if data assigned to correct rover and playing field marker
    - VERIFIED by test program text output
- Test Case #2, Data Translation:
    - SHOW that data can be translated into 2D playing field representation
    - CORRECT if algorithm adjusts for viewing angle
    - CORRECT if map is being correctly read
    - VERIFIED by comparing output with test text file
- Test Case #3, Rover Node Assignment:
    - SHOW that rover locations are being assigned to the correct node
    - CORRECT if raw data points from file are adjusted, (x,y) values are correct
    - CORRECT if (x,y) value of each rover is assigned to correct node
    - VERIFIED by test program output
- Test Case #4, Pacman/Ghost “Captured”:
    - SHOW that Pacman and Ghost can be marked “captured” when within a certain distance
    - CORRECT if input data points that are known to be close enough produce captured flag
    - VERIFIED by test program output of capture flag

PACMAN ROVER GUI VALIDATION - DANNY

- Test Case #1, GUI Components:
  - SHOW that buttons output rover commands to textbox
  - CORRECT if left, right, and 180 display the right messages
  - VERIFIED by testing buttons manually, GUI framework does not have a testing module.
- Test Case #2, Network Output:
  - SHOW that buttons output rover commands over network to message router
  - CORRECT if messages on textbox match messages received on message router
  - VERIFIED by running a test message router to capture any messages
- Test Case #3, Network Input:
  - SHOW that messages received from message router display on the textbox
  - CORRECT if messages from router match messages printed to the textbox
  - VERIFIED by running a test message router to send arbitrary messages
- Test Case #4, Fruit Mode:
  - SHOW that GUI reacts to certain status messages, specifically the fruit mode
  - CORRECT if GUI receives a fruit status message and displays "FRUIT OBTAINED for a predetermined amount of time.
  - VERIFIED by using test message router to send status message and then checking GUI for reaction

Milestone 4

TESTABLE COMPONENTS:

- LFA Decision Engine (pic32)
- PixyCam I/O software (RPi)
- AI Engine / Pathfinding (Rpi)
- Motor Control (pic32)
- I2C Interface (pic32)

Integration Schedule

RASPBERRY PI INTEGRATION COMPONENTS:

- Routing Software
- AI Engine
- GUI
- Statistics View
- PixyCam I/O

PIC32 INTEGRATION COMPONENTS:

- I2C Tasks/ISRs
- LFA Decision Algorithm
- Color Sensor Algorithm
- Motor Control Algorithm
- UART/WiFly TX-RX
- Debug Task

SCHEDULE FOR INTEGRATION

WEEK	INTEGRATION TASK	RESULT
NOV 13 - 19	Integrate AI engine with PixyCam I/O	AI Engine should be able to receive information from PixyCam I/O and process real results from physical board
NOV 13 - 19	Integrate GUI, Stats, AI Engine with Router	All on-Pi communication should be ready to go
NOV 20	Get RX/TX framework finished	
NOV 20 - 27	Integrate I2C task, LFA decision engine	Make sure I2C can be read and fed to LFA decision engine to get direction results (with emulated motor movement)
NOV 20 - 27	Integrate I2C and Color Sensor Algorithm	Make sure Color Sensor can read and interpret I2C
NOV 20 – 27 (weekend)	Integrate LFA decision engine with motor control	Get this thing moving down the line!
Rest… Testing!		