Software Formalization

Year: 2015 **Semester:** Spring **Team:** 2 **Project:** R.I.S.K.
Creation Date: February 19, 2015 **Last Modified:** October 7, 2015
Author: <redacted> **Email:** rdacted@purdue.edu

Assignment Evaluation:

Item	Score (0-5)	Weight	Points	Notes
Assignment-Specific Items				
Third Party Software				
Description of Components				
Testing Plan				
Software Component Diagram				
Writing-Specific Items				
Spelling and Grammar				
Formatting and Citations				
Figures and Graphs				
Technical Writing Style				
Total Score				

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1.0 Utilization of Third Party Software
We are using no third party software on our microcontroller, outside of the basic interfacing header files and C standard library provided by Microchip.

The Raspberry Pi Libraries being used are as follows:

Name	License	Description	Use
BCM2835 library	GPL V2	"This is a C library for Raspberry Pi (RPi). It provides access to GPIO and other IO functions on the Broadcom BCM 2835 chip, allowing access to the GPIO pins on the 26 pin IDE plug on the RPi board so you can control and interface with various external devices." [1]	We will be using this library for our SPI communication on the Raspberry Pi's end.
Apache2	Apache License	"The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards." [2]	We will be using the web server to host our web application.
Socket.IO	MIT	"Socket.IO enables real-time bidirectional event-based communication. It works on every platform, browser or device, focusing equally on reliability and speed." [3]	We will most likely be using socket.io in our project to allow bidirectional communication which will make our updates faster to the client devices.
iScroll 5	MIT	"iScroll is a high performance, small footprint, dependency free, multi-platform javascript scroller." [4]	We will be using iScroll to provide a side scrolling view on the web app that allows more cards to exist than the screen can display at once. The additional cards will be hidden off of the screen until they

			are scrolled onto the screen.

Table 1. Raspberry Pi Libraries

2.0 Description of Software Components

The software components for the project are as follows: Game Logic, Basic I/O, and Raspberry Pi.

The game logic component is responsible for keeping track of various aspects of the game state, including troop counts and country ownership, and for advancing the game state based on user input. The game logic takes the form of a single large state machine that is "clocked" each time input is provided by a player pressing buttons or making a card exchange in the web application. The current game state is repeatedly read by the Basic I/O system to convert the game state into a format that can be displayed on the country board LEDs. This component was written entirely by our team and has been completely developed, so all that remains is its final testing.

The Basic I/O component is responsible for displaying game state information to the players via the RGB LEDs, 7-segment displays, and LCD screen, and for detecting user input via the buttons and rotary encoder. Data will be shifted out to the LEDs and 7-segment displays using SPI (which the microcontroller natively supports), and a simple LCD driver will control communication with the LCD screen. This component will be developed entirely by our team, and we do not plan on using any third-party software for it.

The Raspberry Pi component is responsible for running a mobile web application (written in HTML, CSS, JavaScript, and PHP) that will be used by the players to manage and use their country cards. There will be several libraries used here, all of which are mentioned in Table 1 in the previous section. The website itself will be developed completely by the team and will not be borrowing from other projects or third-party applications. The Pi will also have a program running to send and receive data from the microcontroller using the SPI protocol. This data will determine what the web application displays. All of the web server functionality will be delivered over an ad-hoc WiFi network that is hosted by the Pi. The Raspberry Pi will be running all of these applications using Linux as an operating system.

3.0 Testing Plan

The Basic I/O component will be tested on a piecewise basis. Each country board must be able to display any two-digit number and one of six potential colors - therefore, a test will be written that cycles through all of these. Additionally, it is crucial that the SPI be working properly to send the country information to each of the territories when they are all chained together, which will require a test similar to the previous one but which shifts out data for all 42 boards. Small-scale tests will be run to ensure the LCD is able to receive and display messages - these will consist simply of sending each line of the LCD text and checking that the right output is displayed. To check the rotary encoder, a test will be set up to emulate scrolling between countries: the selected country will be "flashing" (alternating between some color and white), and when the knob is turned the next country in sequence will begin flashing. Button testing is as simple as displaying a message on the LCD which describes which button was pressed. This is the most important component to test - without displays, the game cannot be seen, and without buttons, the game cannot be played.

The Game Logic component will be tested by enacting all of the various gameplay scenarios that can happen within RISK - including battles, troop movements, troop reinforcements, using cards, and many others. A simple GUI program has already been developed to test the game logic code independently of the microcontroller; this has allowed for the code to be developed and run through initial tests. Once the displays are set up, the game code will be run on the microcontroller through the various scenarios mentioned in order to test that the main loop is running properly on it. This is the second most important component to test - though it is crucial to running the game, it is less important than getting the displays to work.

The Raspberry Pi is the least critical piece of the game. It will first be tested by making sure that data can be sent to and received from the microcontroller over SPI. Once that connection is established it will have its ad-hoc network tested to verify that mobile devices can connect to the web application. The third test will be to receive card data from the microcontroller and to display that data on the mobile devices. The fourth and final test will be to test the user of the mobile device choosing to play their cards and having the Pi send that data over SPI to the microcontroller. This component is the third most important, since the game can be played without cards.

4.0 Sources Cited:

[1] McCauley, Mike. "C library for Broadcom BCM 2835 as used in Raspberry Pi." Internet: http://www.airspayce.com/mikem/bcm2835/, [Feb. 26, 2015].

[2] "HTTP SERVER PROJECT." Internet: http://httpd.apache.org/, 2015 [Feb, 26, 2015].

[3] "socket.io." Internet: http://socket.io/, [Feb. 26, 2015].

[4] Spinelli, Matteo. "ISCROLL 5." Internet: http://cubiq.org/iscroll-5, Jan. 10, 2014 [Feb. 26, 2015].

Appendix 1: Software Component Diagram

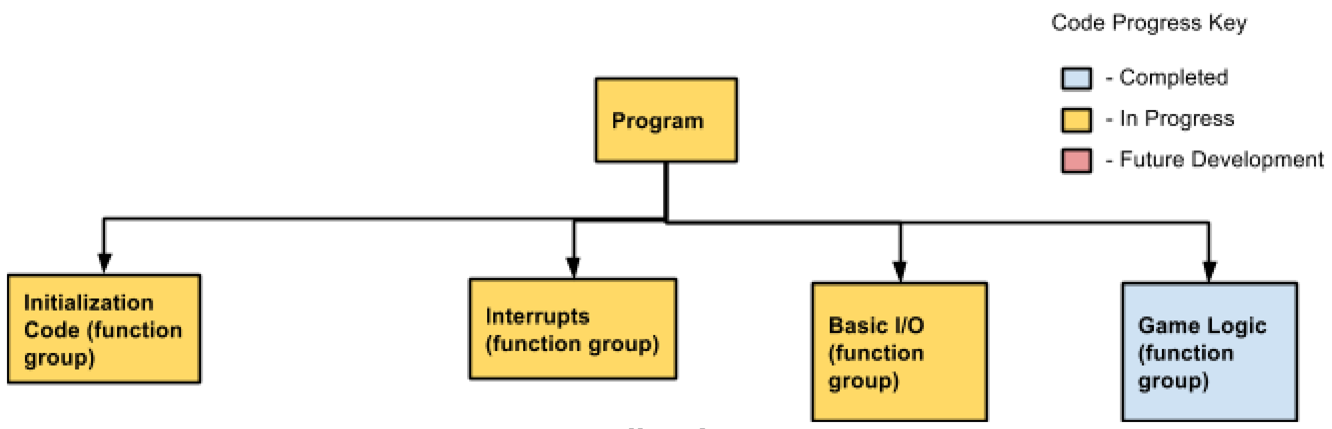
Code Progress Key
- Completed
- In Progress
- Future Development
Program
Initialization Code (function group)
Interrupts (function group)
Basic I/O (function group)
Game Logic (function group)

Figure 1: Overall Code Structure

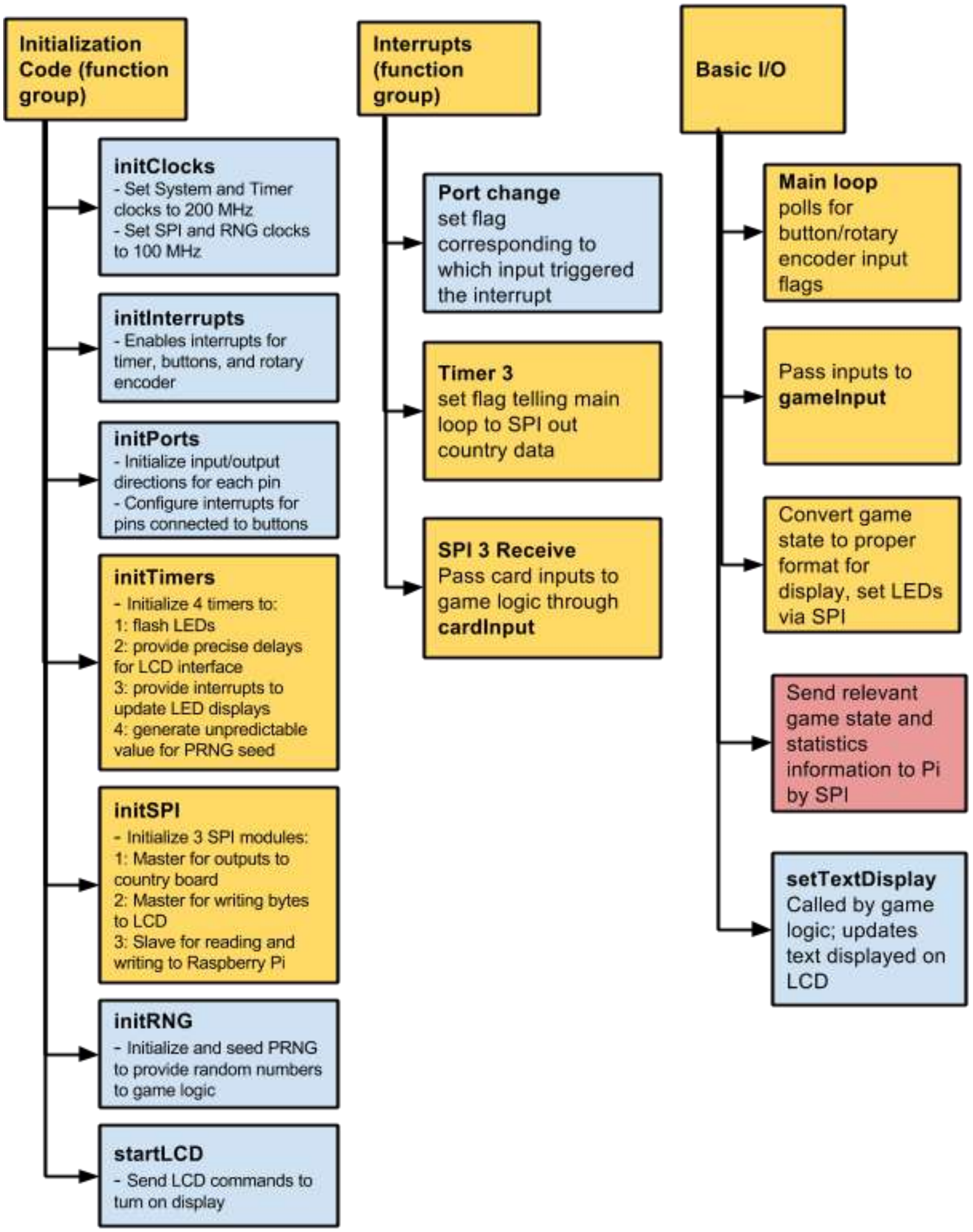
Initialization Code (function group)
initClocks
- Set System and Timer clocks to 200 MHz
- Set SPI and RNG clocks to 100 MHz
initInterrupts
- Enables interrupts for timer, buttons, and rotary encoder
initPorts
- Initialize input/output directions for each pin
- Configure interrupts for pins connected to buttons
initTimers
- Initialize 4 timers to:
1: flash LEDs
2: provide precise delays for LCD interface
3: provide interrupts to update LED displays
4: generate unpredictable value for PRNG seed
initSPI
- Initialize 3 SPI modules:
1: Master for outputs to country board
2: Master for writing bytes to LCD
3: Slave for reading and writing to Raspberry Pi
initRNG
- Initialize and seed PRNG to provide random numbers to game logic
startLCD
- Send LCD commands to turn on display
Interrupts (function group)
Port change
set flag corresponding to which input triggered the interrupt
Timer 3
set flag telling main loop to SPI out country data
SPI 3 Receive
Pass card inputs to game logic through cardInput
Basic I/O
Main loop
polls for button/rotary encoder input flags
Pass inputs to gameInput
Convert game state to proper format for display, set LEDs via SPI
Send relevant game state and statistics information to Pi by SPI
setTextDisplay
Called by game logic; updates text displayed on LCD

Figure 2: Initialization Code, Interrupts, and Basic I/O Structure

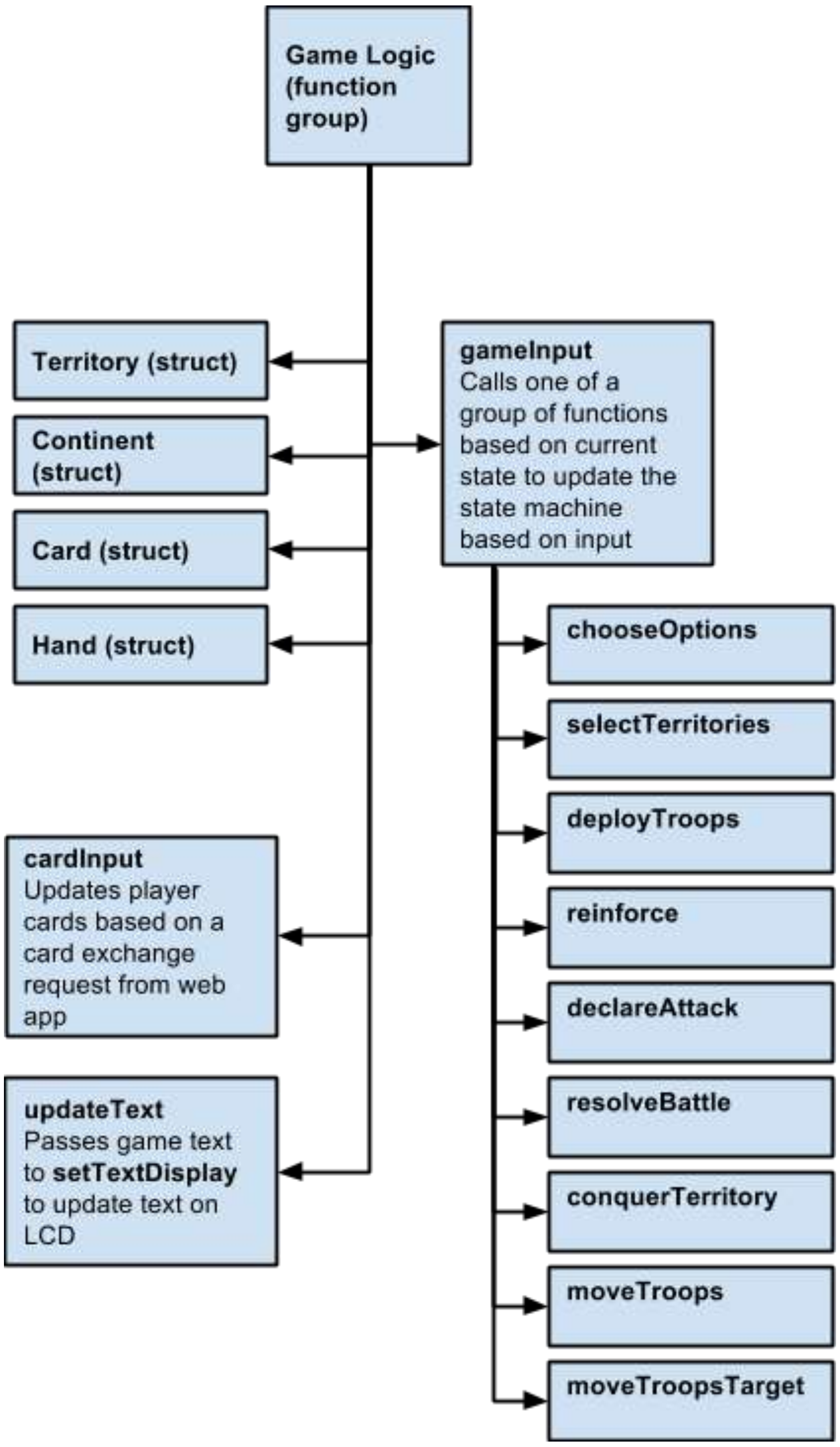
Game Logic (function group)
Territory (struct)
Continent (struct)
Card (struct)
Hand (struct)
gameInput Calls one of a group of functions based on current state to update the state machine based on input
chooseOptions
selectTerritories
deployTroops
reinforce
declareAttack
resolveBattle
conquerTerritory
moveTroops
moveTroopsTarget
cardInput Updates player cards based on a card exchange request from web app
updateText Passes game text to setTextDisplay to update text on LCD

Figure 3: Game Logic Structure

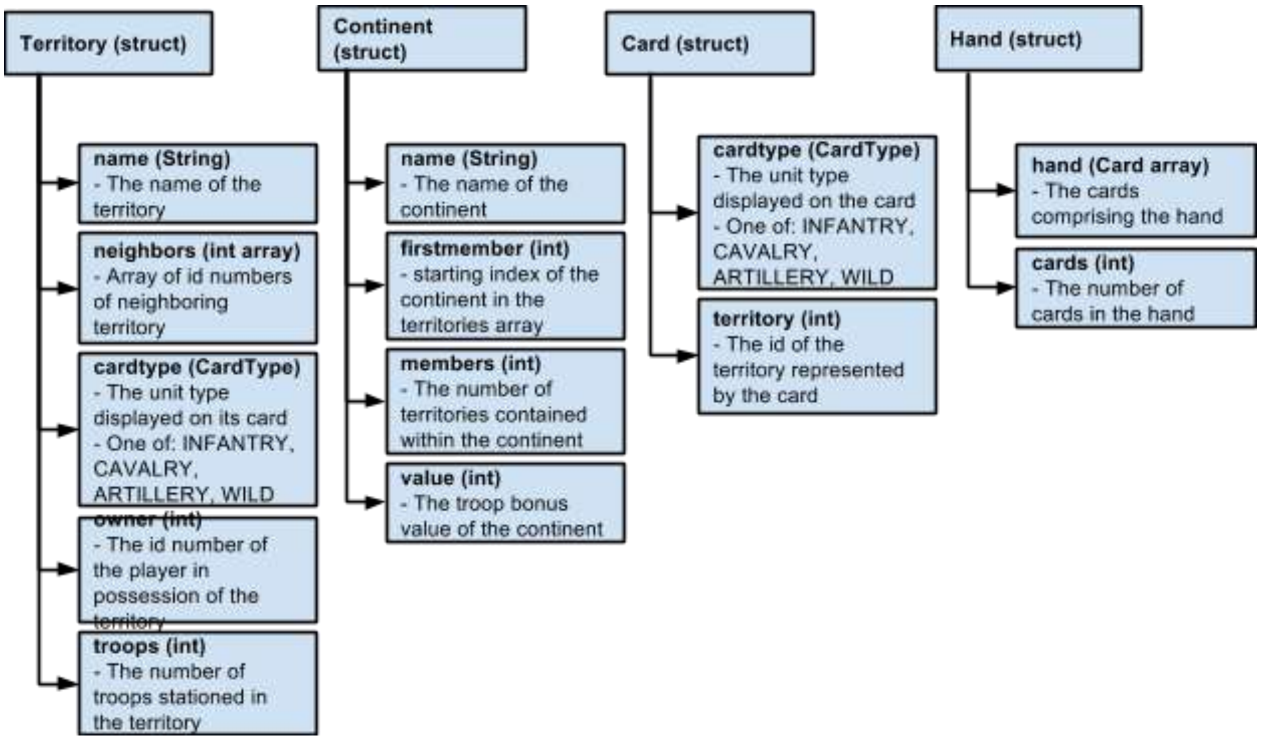
Territory (struct)
name (String) - The name of the territory
neighbors (int array) - Array of id numbers of neighboring territory
cardtype (CardType) - The unit type displayed on its card - One of: INFANTRY, CAVALRY, ARTILLERY, WILD
owner (int) - The id number of the player in possession of the territory
troops (int) - The number of troops stationed in the territory
Continent (struct)
name (String) - The name of the continent
firstmember (int) - starting index of the continent in the territories array
members (int) - The number of territories contained within the continent
value (int) - The troop bonus value of the continent
Card (struct)
cardtype (CardType) - The unit type displayed on the card - One of: INFANTRY, CAVALRY, ARTILLERY, WILD
territory (int) - The id of the territory represented by the card
Hand (struct)
hand (Card array) - The cards comprising the hand
cards (int) - The number of cards in the hand

Figure 4: Game Logic Structure Definitions

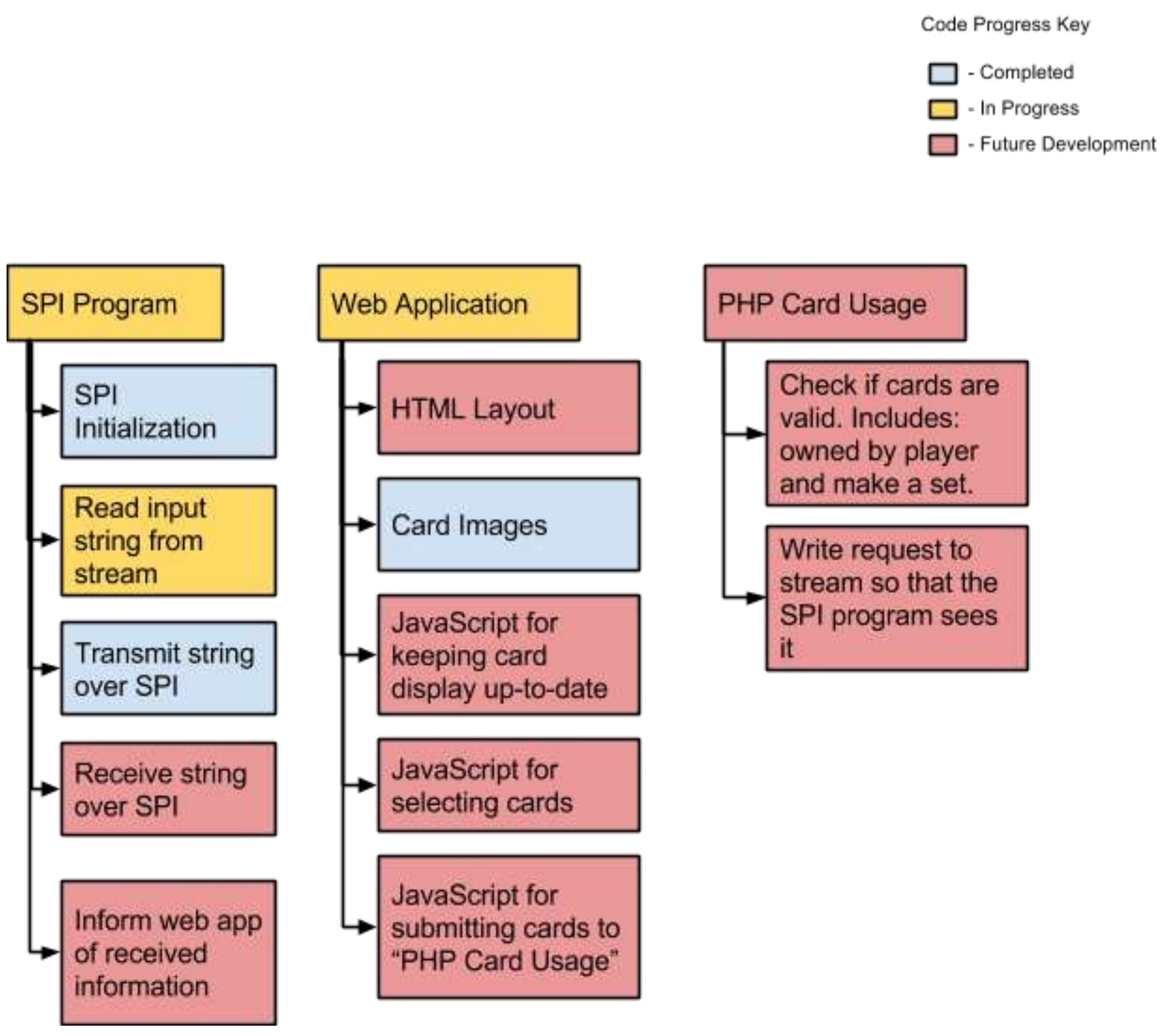
Code Progress Key
- Completed
- In Progress
- Future Development
SPI Program
SPI Initialization
Read input string from stream
Transmit string over SPI
Receive string over SPI
Inform web app of received information
Web Application
HTML Layout
Card Images
JavaScript for keeping card display up-to-date
JavaScript for selecting cards
JavaScript for submitting cards to "PHP Card Usage"
PHP Card Usage
Check if cards are valid. Includes: owned by player and make a set.
Write request to stream so that the SPI program sees it

Figure 5: Web Application Code Diagram