EE 478 Capstone Final Report
RFID Interaction Suite

Alyanna Castillo
Patrick Ma
Ryan McDaniels

Contents

List of Tables

List of Figures

1 ABSTRACT

2 INTRODUCTION

These are some example of how to cite and use bullet points A reference to Table 4 and one to the Design Specification, Section 3.

- Overall summary description of the module - 2-3 paragraphs maximum (explanation of use cases goes here)
  - Specification of the public interface to the module
    - Inputs
    - Outputs
    - Side effects
  - Psuedo English description of algorithms, functions, or procedures
  - Timing constraints
  - Error handling

3 DESIGN SPECIFICATION

3.0.1 Design Overview

This system is a RFID-based gaming system. A user is able to play alone against a computer, or can play with other users owning a system using a multiplayer connection feature. Additionally, users can create their own customized cards using a third built-in function of the device.

3.1 Design Requirements

3.1.1 Environmental Requirements

The device must operate in an indoor/outdoor environment with relative humidity consistent with desert and tropical environments. It must be durable enough to withstand various types of users, especially small children. The unit is to be portable and battery operated.

3.1.2 System Input and Output Requirements

The system must be capable of accepting several different types of signals and inputs:

- Standard frequencies used for close-range or Near-Field Communication Radio Frequency Identification (RFID) devices
- Standard frequencies used for commercial wireless communication standards such as wifi networks
- User input from a keypad to navigate menus, enter commands, and provide other alphanumeric information

- When in multiplayer mode, communication and commands from other players must be accepted and responded to.

The system must be capable of providing the following outputs:

- Display information about the current game state through an LCD screen
- Provide status information of individual cards placed on the game board
- Send commands to other systems when in multiplayer mode
- Send wireless signals to program RFID-enabled cards

3.1.3 User Interface

The system must also have the following buttons, switches or interface devices:

- 16-button keypad with alphanumeric character labels
- Reset button that when pressed causes a soft reset of the system
- Power button that turns the system on and off
- An LCD screen viewable from several angles. The screen must be viewable indoors and in overcast conditions

An example User interface is shown in Figure 1.

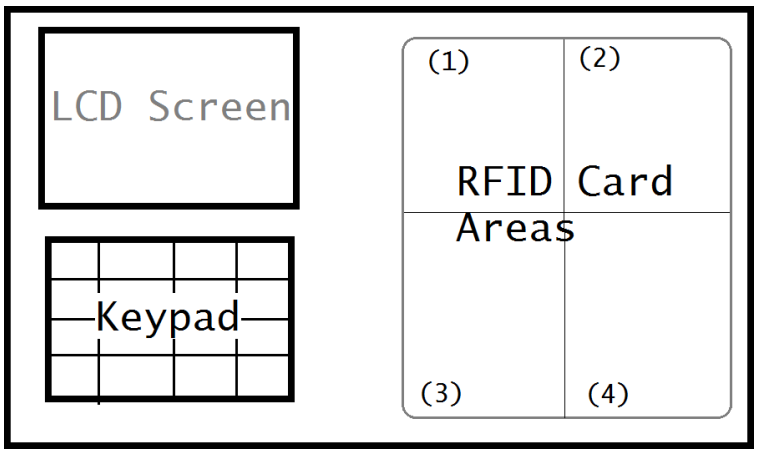


Figure 1: An example front panel layout for the system

3.1.4 Functional Requirement

The system must support the following modes of operation:

Single Player: Allows the user to select from, and play, single player games. Upon entering Single Player mode, the system prompts the user to select a game to play from those stored in memory. Selecting a valid game loads the game from memory and begins game execution.

Multiplayer: Allows the user to select and play multiplayer games. Upon entering Multiplayer mode, the system activates wireless communication and attempts to connect to other users in Multiplayer mode. Once a connection is established, the users select a multiplayer game on both systems and gameplay begins.

Build Card: Allows the user to create custom cards for a game. To create a custom card, the user will input the data via the keypad. The data stored will vary based on game requirements. All cards must contain unique card serial numbers or identifing ID numbers, and a code that specifies which games the card is valid for.

In all three modes, the user will be instructed by the programmed game to place a card in the RFID reading area to interact with the game and control events.

3.1.5 Operating Requirements

The system must operate in a standard commercial or household environment. The system must be portable, wearable on the user's arm, and operate on an internal power supply.

3.1.6 Reliablility and Safety Requirements

The system must be compliance with communication and electromagnetic radiation standards including those from the U.S. Federal Communication Committee, and applicable state and federal child safety laws.

3.2 Identified Use Cases

The user interacted with the system through the game. In the figure below, a user would interact with the system through single player, multiplayer, or build card mode. In a single player game, the user would play against a computer. In a build card game, the user would be working with the system to create/modify cards. Should a user be in multiplayer mode, they would interact with another using using the system during gameplay.

3.3 Detailed Specifications

3.3.1 System Description

This specification describes and defines the basic requirements for a wearable digital card-gaming device. The device will house 1 DLP-RFID2 reader module that will take data from RFID cards. As cards are placed on the device, a 4-inch linear servo will move the reader to scan each card

and use it in a game of the user's choosing. The user has the option of doing a single player, multiplayer, or build their own card deck. In multiplayer mode, the user would be able to wirelessly communicate with one other player in possession of another device via the Xbee wireless module.

3.3.2 Specification of External Environment

The device is to be operated in an indoor/outdoor environment with relative humidity consistent with desert and tropical environments. It must be durable enough to withstand various types of users, especially small children. The unit is to be portable and battery operated.

Temperature Range: 0-60C (32-140F)

Humidity: Humidity 30-60

3.3.3 System Inputs

The system must support the measurement of the following kinds of signals:

- Frequency in the standard 13.56 MHz range for close range RFID signals
- Frequency in the 2.4 GHz range for wireless communication with Xbee modules
- It will follow protocols from ISO 15693 for RFIDs
- User input via the 16-character keypad.

Reading of RFID cards will be done by sending standard control strings, such as "0108000304FF0000" (for "ping") to the DLP-RFID2. These strings will allow control over read and write functions on the device. These strings will be programmed into a driver on a microcontroller. Each RFID tag will have an identifier so that its information can be looked up in the system's memory. These strings are sent through a 115200 baud UART connection and are detailed in Table 1.

DLP RFID-2 Commands

Description	**String**
Inventory (Find cards in range)	010B0003041406010000000
Set data in/out to user memory	010C00030410002101020000
AGC mode (Alternating Gain Control)	0109000304F0000000
Write to block	0113000304182220[UID][BLOCK][DATA]
Read block	0117000304186221

Table 1: Command strings to control the RFID reader.

The system will also accept user input from a keypad that will determine which mode to set the game in: single player, multiplayer, or card building. It will also allow the user to navigate the menu screens and select their next move. The controls for navigation can be found in Table 2.

If the system is in multiplayer mode, it must receive game data from the device of another player and modify the user's game state based on that data. The shared data consists of players' scores, current cards on the field, etc.

User Keypad Input	
Command	**Key**
Move Up	2
Move Down	8
Move Left	4
Move Right	6
Select/Enter	D
Back/Return	B

Table 2: Controls to navigate through the system's menus.

The connection between two systems must be implement the IEEE 802.15.4 wireless networking protocol. To identify potential connections, the system must send a "ping" command followed by a unique system identification number (a serial number). The system then waits three seconds for any response. If no valid response occurs, the system recommences the search sequence up to five times. A system responding to a connection request waits until the connection request ends then sends the ID of the system it wishes to connect followed by its own system ID. The system then waits for confirmation of the request from the other system. During game play, the system sending data transmits the ID of the second system's Xbee, followed by the data.

3.3.4 System Outputs

The system must display information about the current game state and the reading from the RFID signal on an LCD display. The display will show a menu that will allow a user to navigate through options using a keypad. When a game is being played, the display will show information relevant to the game, such as simple graphics, scores, instructions, and player moves.

The system will respond to cards being placed in the card reader area. The card will be physically "locked in" until the user decides to place another card in its place. An LED will also indicate that a card has been correctly placed and read, according to Table 3.

LED Color Indications	
Card Status	**Color**
Card placed and recognized	Green
Card placed, not read	Yellow
Error reading card	Red
No card placed	Off

Table 3: Status of LED lights based on RFID read state

When the system is in multiplayer mode, data must be sent wirelessly to another device. This data consists of players' scores, current cards in play for the game, etc. The LCD display will show information about the second player, like player name, in addition to information relevant to the game.

3.3.5 User Interface

The user must be able to select the following using the keypad on the front panel of the instrument:

- Mode
  - Singleplayer
  - Multiplayer
  - Build Cards
- Card Reader
  - Physical space to place cards

Game mode can be selected through a menu screen on the LCD display – the user will be able to navigate the menu with a keypad. Similar to a gaming device, the power will be controlled by a switch and reset will be a push button. Game output will be displayed on an LCD display. Such a display must be readable from several angles. The front panel of the system can be found in Figure 1.

3.3.6 System Functional Description

A user will be provided with a deck of RFID cards that contain data about whatever game the user wants to play. These cards would be used in the RFID reading device that would be worn by the user on their arm. First, the user must choose using the keypad what game mode they would like to play: single player, multiplayer, or card building. In all three modes, the user will be instructed by the programmed game to place a card in the RFID reading area to interact with the game and control events. In multiplayer mode the user would have to find the other user they would like to play with and connect to their game using the wireless features of the system. In card building mode, the user would be able to enter data via the keypad to program their own data onto whichever RFID card they choose. A block diagram for the system can be found in Figure 3. The system functionality is described below.

- Read/Write Cards:
  - Communicate w/ DLP: The chosen RFID reader is the DLP-RFID2 reader/writer module. The system must communicate with this reader using the RS-232 protocol at TTL voltage levels. The communication link must be at 115200 Baud, 8-bits, 1 stop bit, and no parity check. The commands to the RFID reader are specified in TRF7960_ISO15693_protocol.pdf
  - Traverse card data: The system must read up to five cards
  - Quantity of Cards: The Reader must be able to read and write to 5 cards at a time.
  - Write Card: Data must be written to the card according to the ISO 15693 standard. The structure of data contained on the card may vary with the game associated with the card. The following data locations must be present:
    - Unique card Identifier (UID) – A serial number uniquely identifying the card from all other cards

        * Card name – The name of the card
        * Display name – The name of the card as displayed to the user
        * Associated Programs – A code that identifies the card as valid for a specific game set.

- Access Database
    - Read Data to SRAM: The system must be capable of requesting data from the SRAM using a byte-addressing scheme. The returned data must be received as an 8-bit word in parallel.
    - Write data to SRAM: The system must be capable of requesting data from the SRAM using a byte-scheme. Data must be sent as an 8-bit word in parallel.
    - Lookup/reference Table:

- User Interaction
    - Read Keypad: The system must read user keypad input with a maximum lag time of 250 ms.
    - Display: The system must display the following
        * Simple graphics
        * Score
        * Game instructions
        * Player options
    - Card Reactions: When a card is placed on the reader, the system must acknowledge the card through a series of LEDs. If the card is read/accepted as valid, green appears. If the card cannot be read, red appears. Additional colors may indicate other statuses

- External Communications (ExtCom): a. Xbee: The system must be capable of maintaining a connection with another system using the XBee wireless standard over a minimum distance of 5 feet. Data transmission must occur at 250kbs. b. RS-232 connection: The system must be capable of maintaining an EIA-232 serial connection with an external computer system. Data sent must be at 9600 baud, 8-bit, 1-stop bit, and no priority. c. Wireless Driver: The driver must be capable of sending commands to initiate connections, acknowledge received data, sending data, and terminating a connection d. RS-232 Driver: The driver must be capable of sending strings of data up to 20 characters in length. The encoding must use the extended ASCII table.

3.4 Functional Decomposition

Figure 6 in Appendix D gives a hierarchy of the system software functions. Each function in the functional diagram has a corresponding software task.

The functions are explained below:

Play Game Loads and begins execution of the requested game.

System Scheduler The system scheduler monitors the state of the system and calls system functions as necessary. On system power on or power on reset, the scheduler intitializes system variables and calls the setup() function of each task.

Read/Write Cards Formats and sends commands to the RFID reader. The task also processes the raw data from the RFID reader and stores the card UID and card memory data for later retrieval by other tasks.

Access Database Reads and writes data to the local SRAM. The Access Database task determines where each byte of data is written and writes or requests the data from the memory. The Access Task also maintains a reference table of game data corresponding to the moves, types, etc. corresponding to data encoded on the RFID cards.

User Interaction User interaction involves three functions: Keypad, Display, and Card Reaction.

Keypad The keypad driver determines which key the user has pressed, if any.

Display The display driver interfaces with the LCD over an Serial Peripheral Interface (SPI) to draw game menus and screens. The display driver also formats text strings and numbers from the game for display on the screen.

Card Reactions The Card Reaction task determines when a card had been placed onto the board, tells the system to read and process the card, and sets the card status LEDs.

LED Driver This task changes the output of the LEDs based upon the read status of the cards.

Communication Communication includes three tasks: external serial communication via RS-232, external wireless communication via 2.4GHz wifi, and internal InterPIC communicaton via I^2C

Serial Driver The serial driver sets up the serial RS-232 connection port. The task also processes incoming serial data, saving the result to system memory and alerting the scheduler to run subsequent tasks.

Wireless Driver The wireless driver configures the XBee module, sets up the person-to-person communication link and formats incoming and outgoing data between the game and XBee module.

InterPIC Driver The InterPIC driver configures the I^2C port and sends data over the I^2C connection. The driver wraps and unwraps data sent over the connection in the I^2C protocol. Each once unwrapped, the command is interpretted, data is stored for system use, and the scheduler is alerted to call other tasks.

Build Card The Build Card function prompts the user to input customizing options then encodes the responses for storage on an RFID card. Once data is properly encoded, the task instructs the InterPIC Driver to write the data to the RFID card through the RFID reader.

4 HARDWARE IMPLEMENTATION

4.1 Top Level Design

4.2 Low Level Design

5 SOFTWARE IMPLEMENTATION

5.1 Top Level Design

The software implementation for the RFID interaction suite consisted of developing the game for a user to play $and for the developers to test$. The system took input from the Xbee, RFID tags, and keyboard. Data would be outputted through the LCD screen. Below is a figure depicting the overall software for the RFID interaction suite.

- Play Game
  - Load Game: The program would have to load the correct game based off of user input through the keypad. The system is told which game to select based off of a set flag.
  - Execute Game: After a game flag has been set, the game would run until a player has finished.
- Read/Write Cards
  - Scan for New Cards: During game mode or build card mode, the system would scan for new cards via the RFID reader/writer.
  - Write Card: During the ”build card” part of the game, the user would be able to write data to the cards through the RFID reader/writer. The user would have the ability to add a name and attack moves to the card through the external keypad and the in-game keyboard.
  - Read card: During game mode, the system would check for available cards to work with. If a player was in the middle of a duel, the system would scan for new cards to use for gameplay. If a reader was trying to build new cards, the system would look for available cards to be modified.
- Access Database
  - Read Data to SRAM: Reads game-relevent data. For example, when a game is to be loaded, the system load the game from the SRAM. Or if the computer were to read in a monster during a single player duel game, it would access the database of monsters.
  - Write Data to SRAM: Writes new game data. If new monsters or games were to be added to the system, it would be written to the SRAM for future use.
  - Look Up / Reference Table: Used as a reference for memory locations of data in the SRAM $i.e. ''Phonebook''$.
- External Communication

  - RS232 Driver: The back end of the system uses the USART to communicate with the Xbee wireless chips and RFID drivers. Information was sent out or recieved by the system through these peripherals.
  - Wireless Driver: In addition to the RS232, a driver for the Xbee wireless had to be implemented to allow the system to communicate with other users during the multiplayer game.

- User Interaction
  - Read Keypad: A driver was made to read key presses from the user. Different parts of the system could read the keypad using functions built in this driver.
  - Display: An LCD driver was modified to develo
  - Card Reaction: An LED driver was made to control the LED lights of the RFID interaction suite. This would indicate to the user whether a card had been properly registered or not, based off of the color of the lights.

5.2 Low Level Design

A major software component of the RFID interaction suite were the built in games.

- LCD.c and LCD.h
- The majority of documentation for our ST7735 1.8 TFT LCD screen did not include example code for how to use the screen. Most tutorials or example code $i.e. Adafruit$ assumed that the user would be programming on an Arduino UNO, which had built in libraries for fonts and shapes. The link below features code found on Google which interfaces the LCD with a PIC microcontroller using the chip's SPI function. However the code was not optomized, thus LCD.c and LCD.h are modified versions of the code. The original drawBox function in LCD.c created a box pixel by pixel - it would take a beginning pixel location and an ending pixel location, causing load times to take longer. The new box function takes the beginning and end coordinates for the box and fills everything in between by printing out pixels all at once. Another function printrs was made to take a char pointer to a string and print it out on the LCD. A major advantage to using this example code was that it already built a font library for the LCD screen. Had the group not utilized this, they would have taken time defining the pixel structure for every possible character.
- Original Code (unmodified): https://sites.google.com/site/arduinomega2560projects/microchip-pic/level-3/st7735-1-8-tft
- Game.c and Game.h
  - Single Player: Single player mode followed the direction of the state machine diagram. All random variables were seeded to a system timer on the PIC.
  - Multiplayer:
  - Build Cards:

- Multiplayer Game
  - Specification of the public interface to the module
    - Inputs
    - Outputs
    - Side effects
  - Psuedo English description of algorithms, functions, or procedures
  - Timing constraints
  - Error handling
- Build Card Game
  - Specification of the public interface to the module
    - Inputs
    - Outputs
    - Side effects
  - Psuedo English description of algorithms, functions, or procedures
  - Timing constraints
  - Error handling

6 PRESENTATION, DISCUSSION, AND ANALYSIS OF THE RESULTS

6.1 Results

6.2 Discussion of Results

6.3 Analysis of Any Errors

The biggest problem with the final version of this project that was present at demo time was the fact that the multiplayer features were not implemented. There was test code that correctly configured the Xbee modules for multiplayer, but they were not completely implemented with the game. The reason for this was because when the I^2C system was implemented, the entire game had to be modified to run within the scheduler, when it was just a single function before. I^2C communication is controlled by the system's interrupt handler, and certain flags are set depending on whether data is being sent or received. Those flags have to be processed, and a game that is running in a function and taking control of the entire system would not allow for those flags to be processed. The time it took to port the game over to running completely within the scheduler made it impossible to get the multiplayer functions completely implemented and working in time.

Another problem at the time of the demo was that card reading was not 100% functional. The system had four distinct sets of data that could be successfully written to a card using the "Build Cards" option from the system's main menu, but the data was not being displayed properly in the singleplayer game. Data coming from the card over the I^2C connection was confirmed to be correct, but the game was not interpreting and displaying the information correctly to the user. This

was also a matter of running out of time. For the same reasons as above (converting the game to be run within the flag-based scheduler) the RFID reading still had some kinks to work out at the time of the demo. Those problems were that:

- Monster type was being read incorrectly
- Monster name was being displayed incorrectly
- Complete attack list was not implemented

For demo purposes, only the first attack would be read and it would be copied to all three attack slots. Only the "FAIL" attack ID was programmed to be read, and if the ID did not match the "FAIL" attack's ID, then it was interpreted as a "SCRATCH" attack. This function worked correctly, but the rest of the attack IDs were not implemented. The monster's level was correctly read as well.

Finally, there was an error when finishing a singleplayer game. When the game was complete, the main menu was not being properly displayed. Once again, the problem was time, and the main menu's controls were still functional. Another menu could be loaded by navigating the invisible menu and choosing an option, letting that screen load, and then pressing the "B" key to return back to the main menu and redraw it.

6.4 Analysis of Implementation Issues and Workarounds

7 TEST PLAN

7.1 Test Specification

7.2 Test Cases

8 SUMMARY AND CONCLUSION

8.1 Final Summary

8.2 Project Conclusions

This project had many hardware components that were new to the group. There was little documentation to gooff of and previous capstone projects had not done anything similar to our design requirements. Much of the project was spent reading documentation and figuring out how to design drivers compatible with the PICs, since most available documentation was for Arduinio shields. Using the LCD and Xbee was more difficult since online tutorials referenced the Arduino shield. RFID tags were a completely new piece of hardware to work with with little documentation that the group had to contact the company for help.

With all this new technology, the group broke down building and testing into separate components. The intention was to put everything together at the end and make a scheduler to fire off every function at a particular time. This was the first development flaw - assuming everything would work when put together. There was miscommunication over how the scheduler would work that many functions with while loops had to be redone using flags. A better method would have been to consistently putting components together once their functionality was verified separately.

Additionally, another PICKit 3 should have been bought for use during testing - with only one PICKit, components had to be tested and debugged one at a time after programming. The ability to consistently load the program into PICs and observe the results would have reduced the time for debugging and testing each individual part.

The original intent of the project was to have a working motor that would move the RFID reader to read multiple cards. Another feature was that he user would also be able to make custom cards with their own ID, attack names, moves, and type. Due to time constraints and testing, these idea had to be scrapped in the final implementation. Future implementations should include these features.

A BREAKDOWN OF LAB PERSON-HOURS (ESTIMATED)

Person	Design Hrs	Code Hrs	Test/Debug Hrs	Documentation Hrs
Patrick	x	x	x	x
Alyanna	x	x	x	x
Ryan	x	x	x	x

By initializing/signing above, I attest that I did in fact work the estimated number of hours stated. I also attest, under penalty of shame, that the work produced during the lab and contained herein is actually my own (as far as I know to be true). If special considerations or dispensations are due others or myself, I have indicated them below.

B BILL OF MATERIALS

Table 4 lists the bill of materials for the construction of one system.

Bill of Materials

Item	Unit Cost	Quantity	Total Cost
TI HI-Plus RFID Tags	0.91	20	18.20
DLP-RFID2 RFID Reader/Writer	50	3	150
Xbee S1 Wireless Chips	30	2	60
PLA Makerbot Filament	48	1	48
GAL22V10D	3.5	4	14
PICKit 3	45	2	90
CY7C128A SRAM	4	2	8
3.3 Volt Linear Regulator	3.22	2	6.44
Lever Switch, micro SPDT, momentary	2.5	6	15
16-key Numeric Keypad	7.5	2	15
128x169 Color LCD	17	2	34
PIC18F46K22 Microcontrollers	7.7	4	30.8
RGB LED, Common Cathode	4	8	12
(EXTRA)			
(EXTRA)			
(EXTRA)			
Total Cost			

Table 4

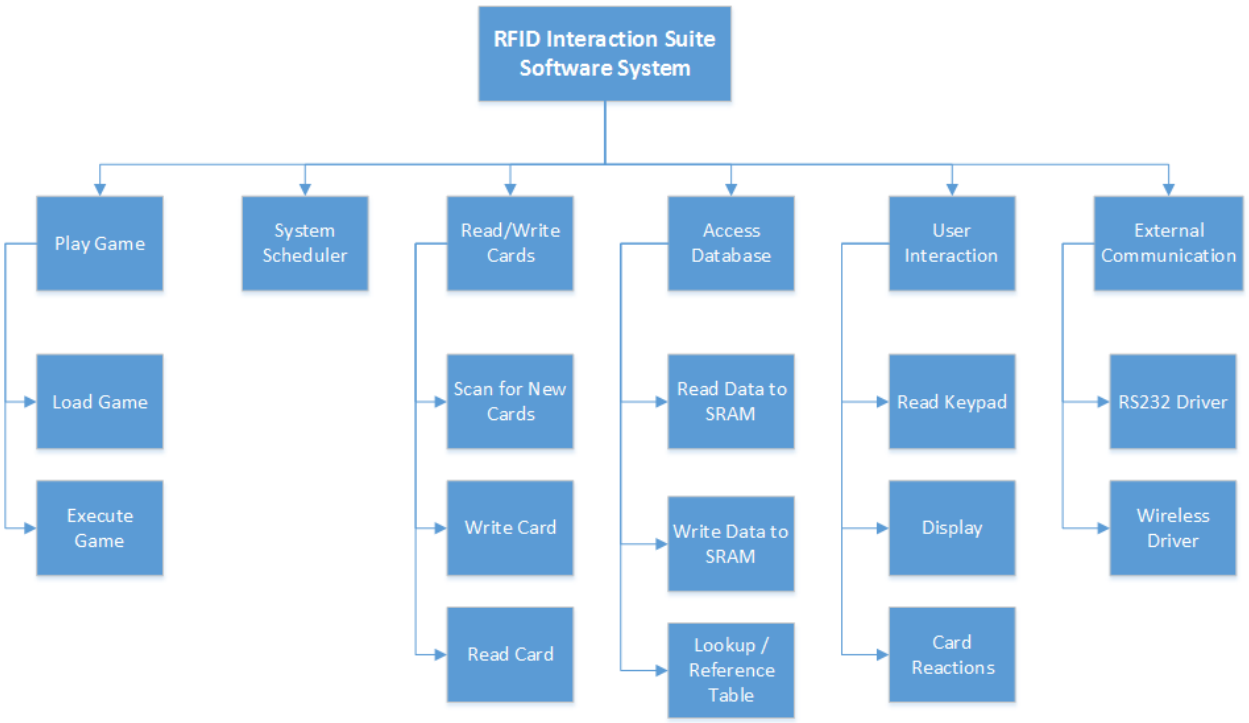


Figure 2: Software functional decomposition showing the major functional divisions and tasks

C HARDWARE DIAGRAMS

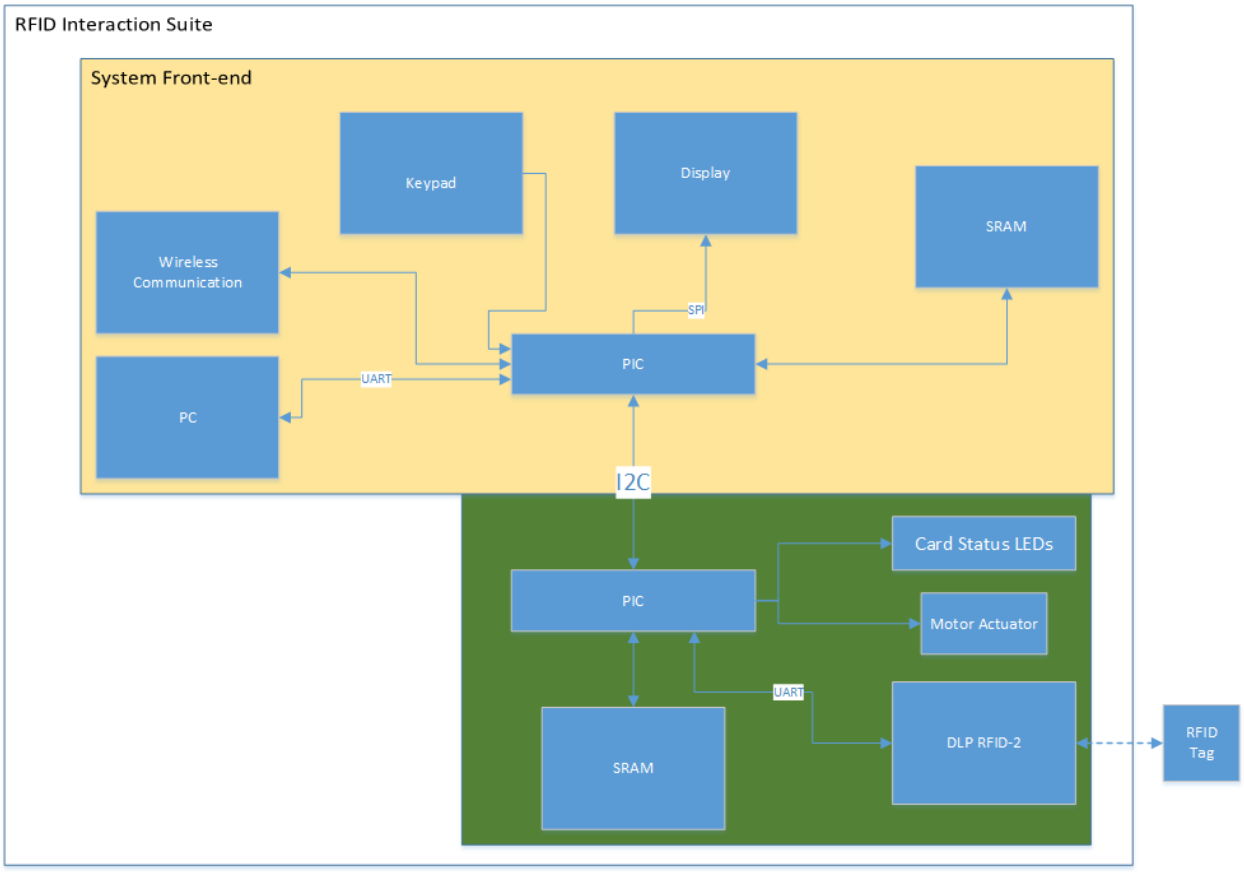


Figure 3: High level block diagram of the system hardware components

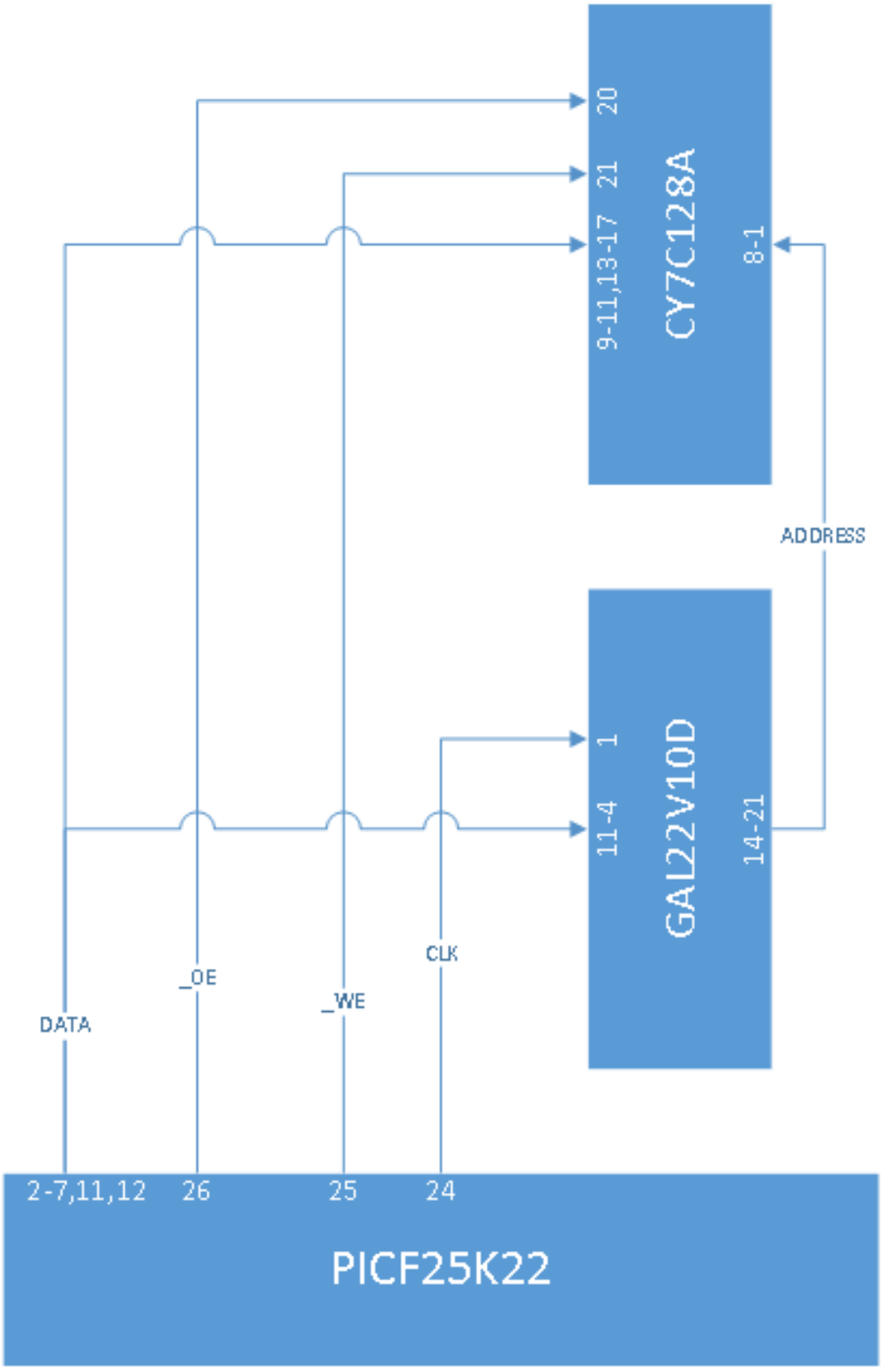


Figure 4: Block diagram of the SRAM hardware system

Figure 5: Pinouts to the front- and back-end microcontrollers

D FUNCTIONAL DECOMPOSITION DIAGRAM

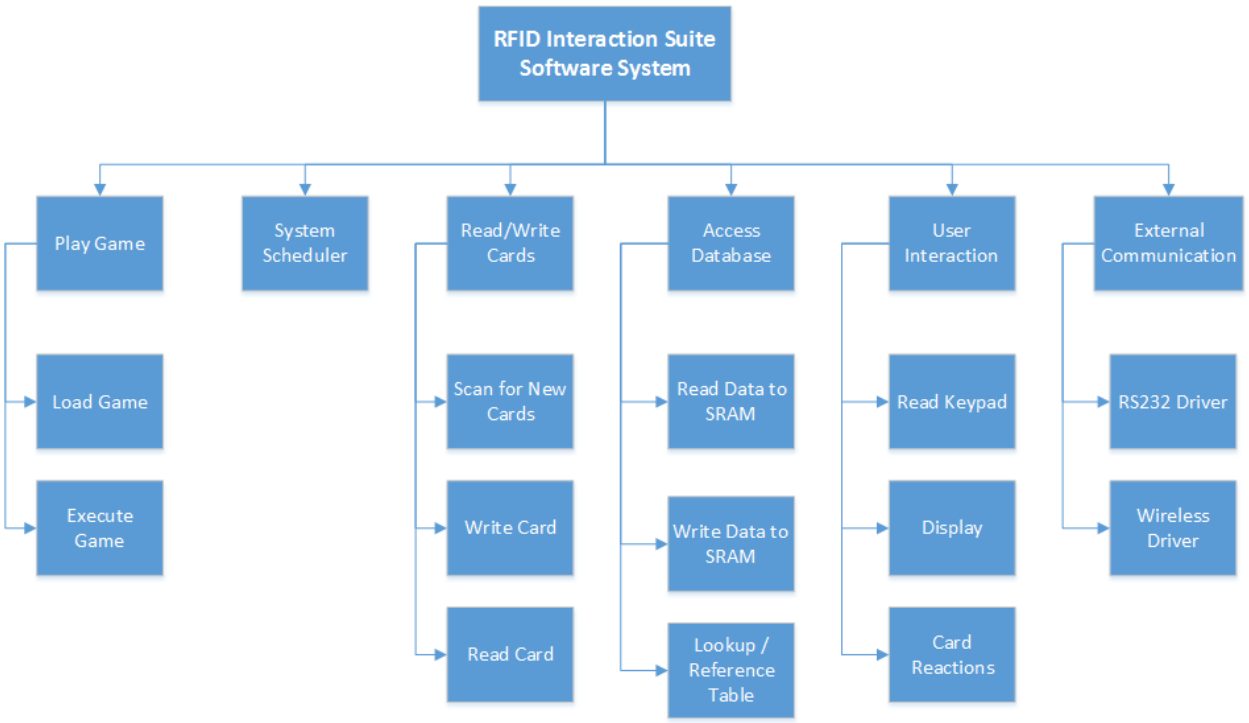
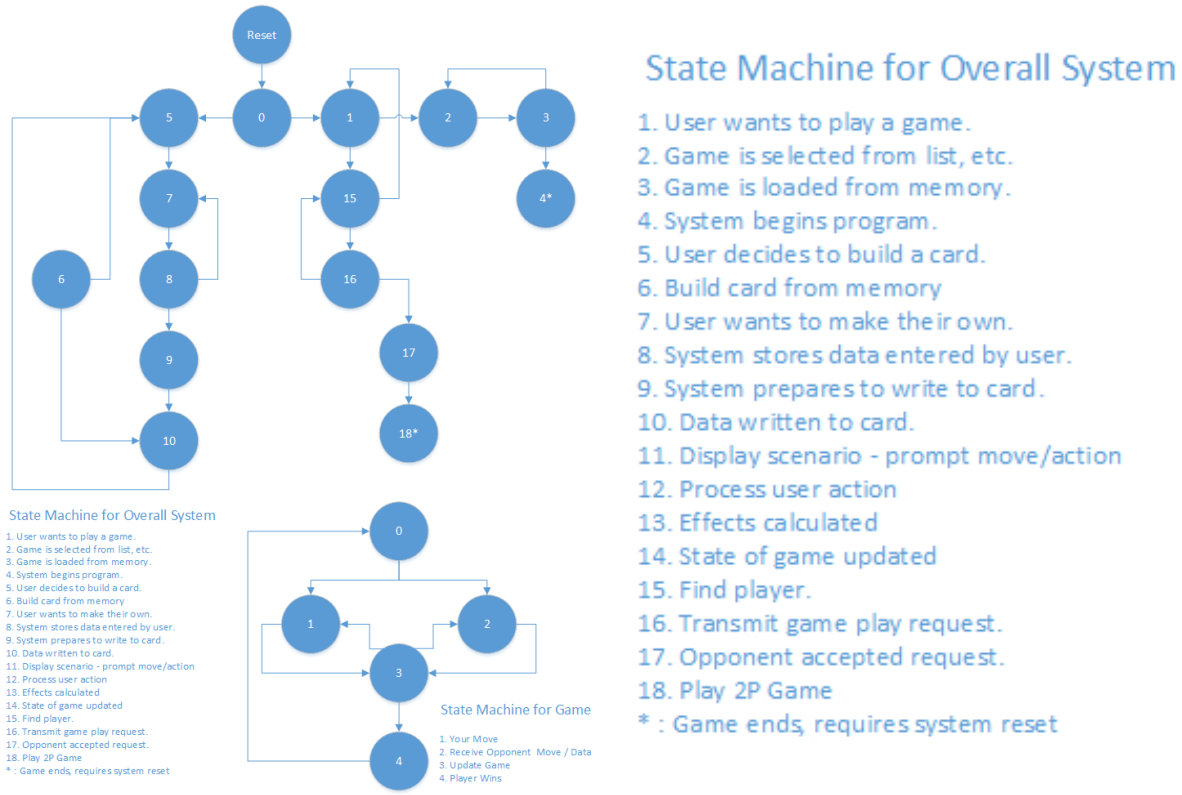


Figure 6: Software functional decomposition showing the major functional divisions and tasks

E STATE DIAGRAMS

E.1 System State Diagram



(a) (b)

Figure 7: State diagram of the primary operating system. Figure 7a shows the states while Figure 7b provides a legend

E.2 General Gameplay State Diagram

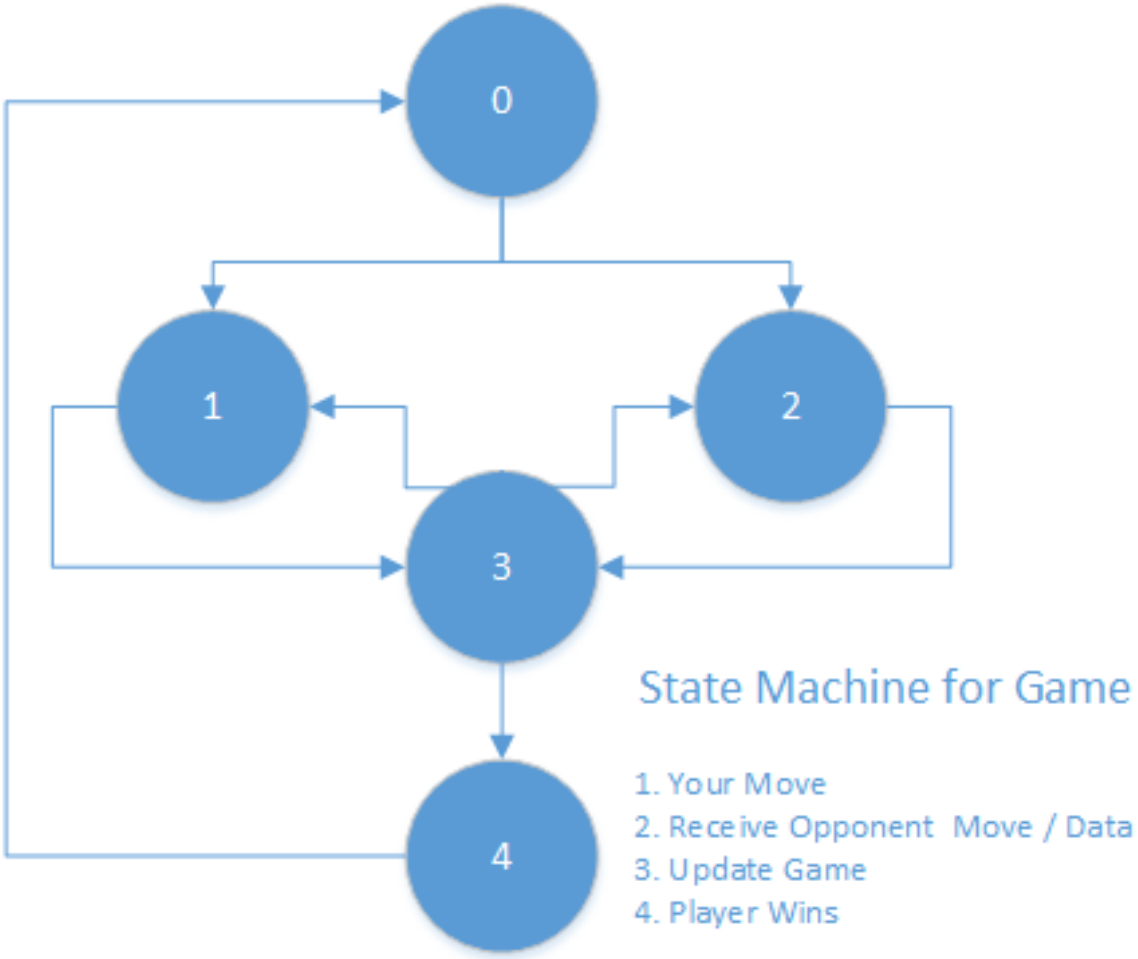


Figure 8: State diagram of basic turn-based game

F CONTROL FLOW DIAGRAMS

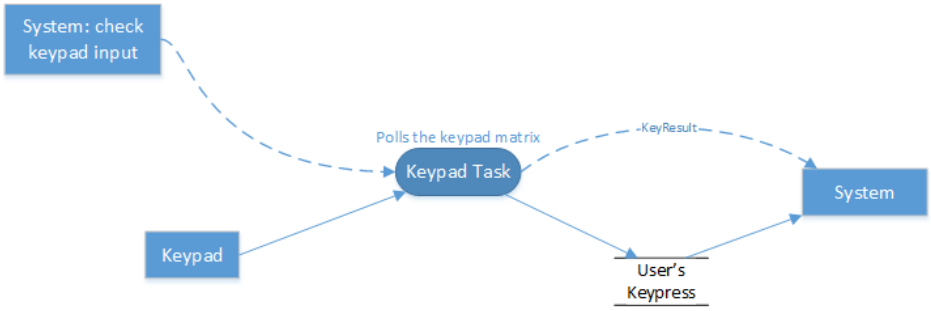


Figure 9: Keypad response

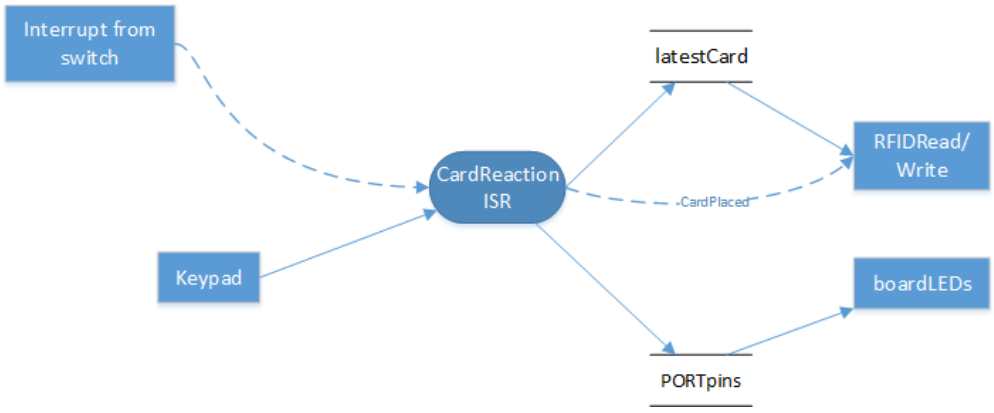


Figure 10: Card Reaction LEDs

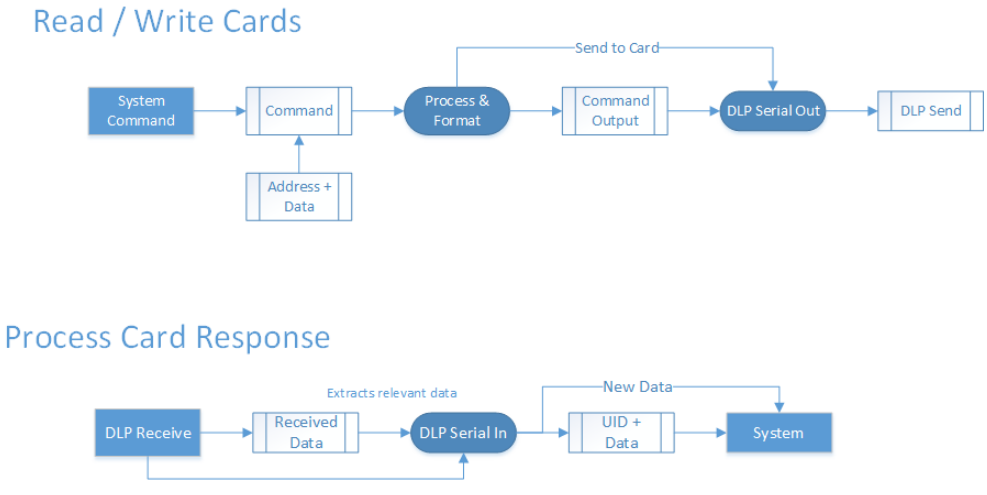
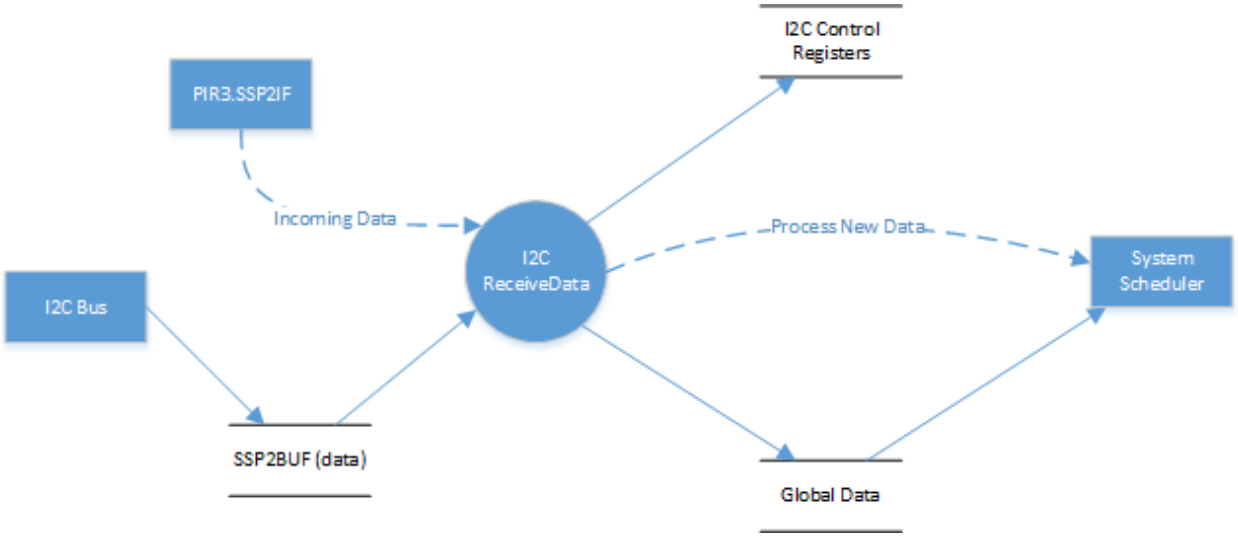
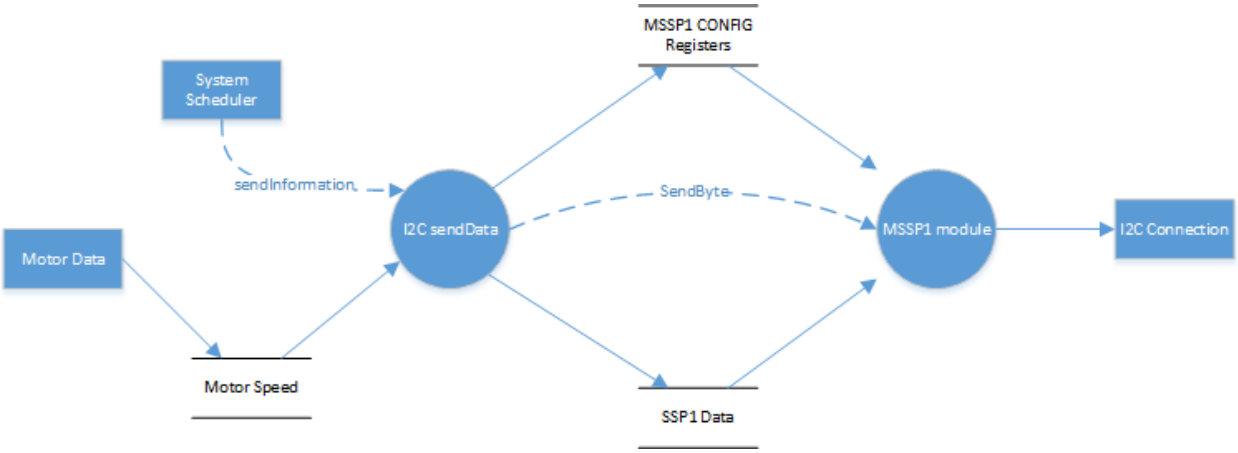


Figure 11: RFID tag reader subsystem



(a)



(b)

Figure 12: I2C communication between microcontrollers. Figure 12a shows the flow for received data and Figure 12b shows the flow for transmitted data.

G PROJECT SCHEDULE

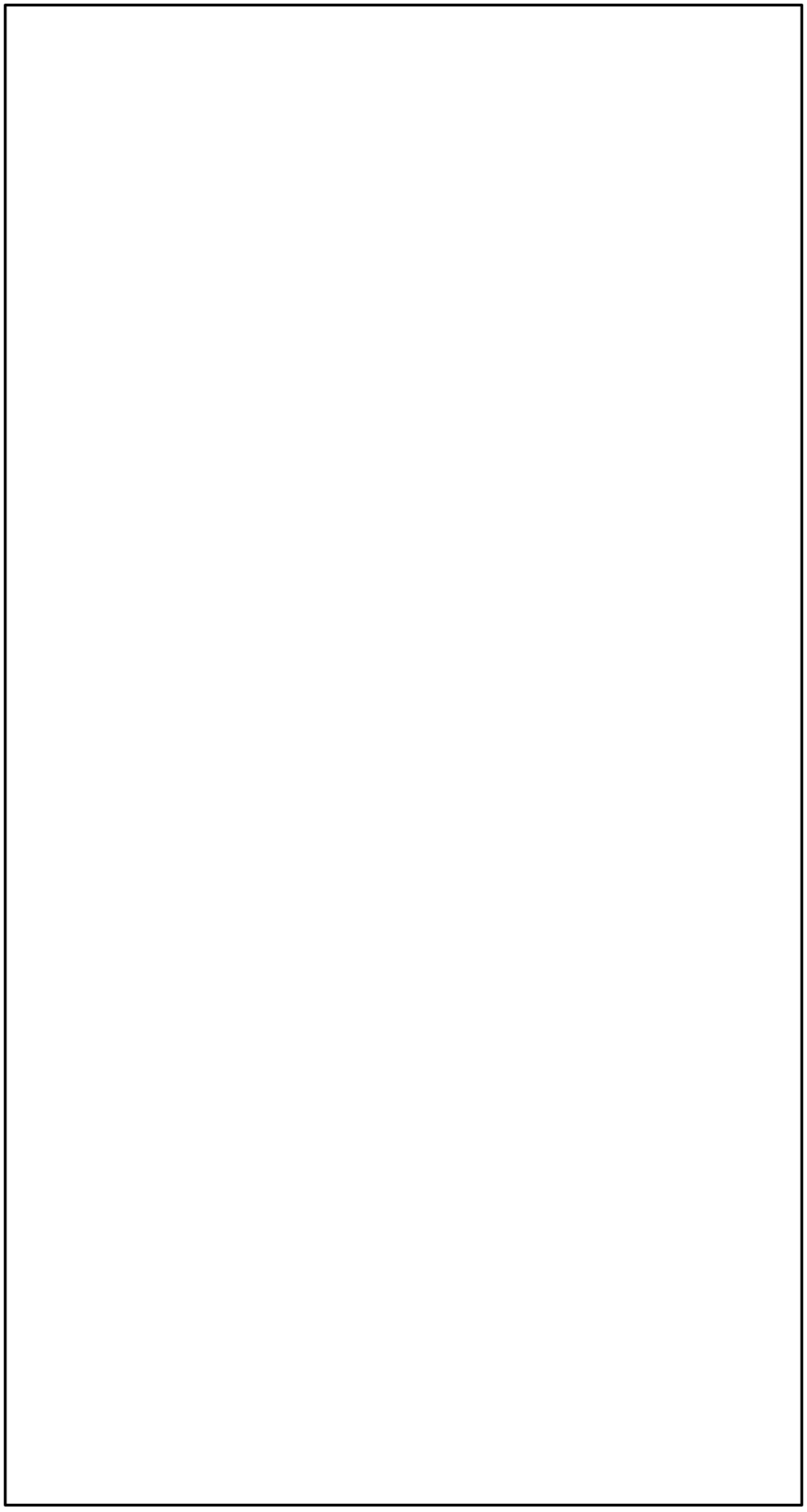


Figure 13: Project Gantt Chart

H SOURCE CODE

Source code for this project is provided below.