Gem Protector

Custom Project Final Report

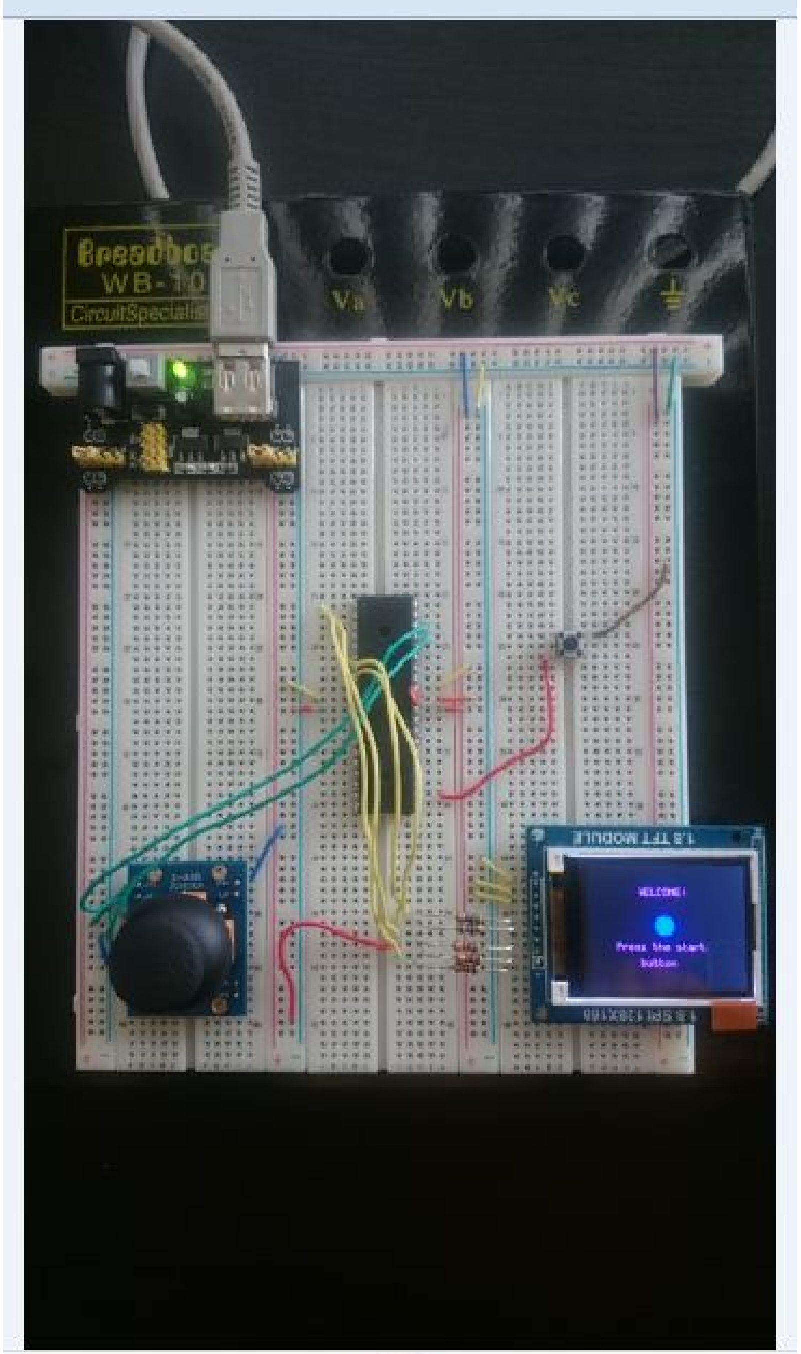
Winter 2018

Tasmiyah Qazi

Table of Contents

Introduction:

Gem Protector is a game where a player controls a character to protect 4 gems from incoming thieves. The player controls the character with a joystick and has to “eat” the thieves before they can reach any gems. The objective of the game is to try and keep your gems from getting stolen for as long as possible which will be represented by a score. Every time you “eat” a thief, you gain 1 point.



User Guide:

When the game turns on, the player will be greeted by a welcome screen. The player can press the start/restart button to start the game. The player has to move the character on screen with the use of a joystick and “eat” the incoming thieves trying to steal the gems. The objective of the game is to last as long as you can before all gems are stolen. The player can also press the start/restart button to stop the game in the middle of gameplay. When all the gems are stolen or if the player presses the start/restart button, the player sees a game over screen where their score will be displayed as well. The player can press the start/restart button to go back to the welcome screen.

Hardware:

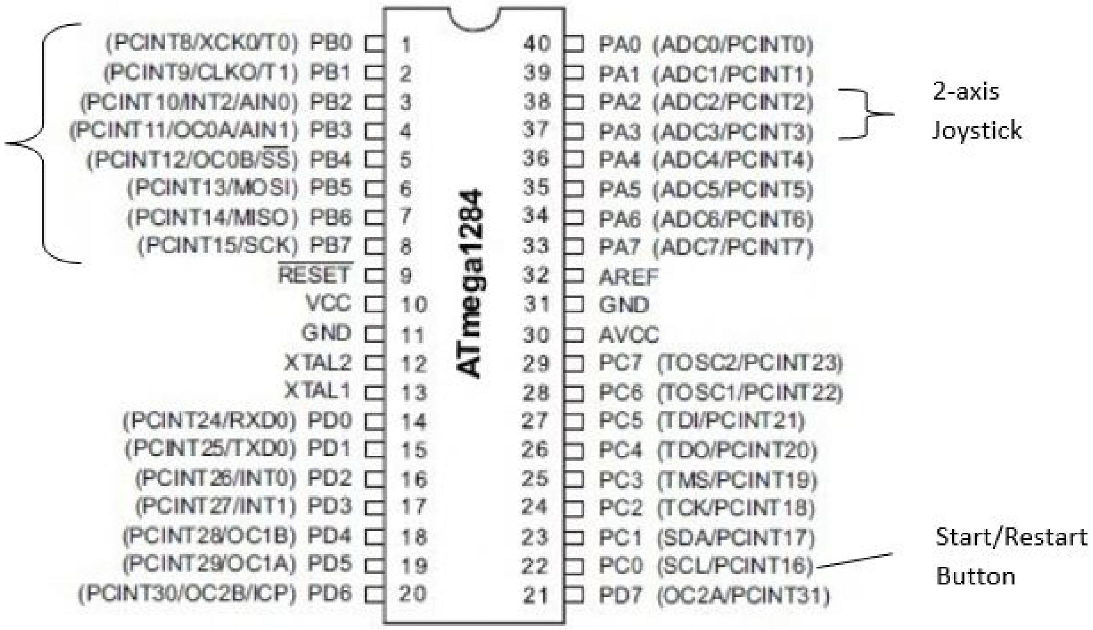
Parts List:

The hardware that was used in this design is listed below. The equipment that was not taught in this course has been bolded.

- ATMega1284p microcontroller
- Button
- **2-axis joystick**
- **1.8 inch, SPI 128x160 TFT display module**

Pinout:

A more detailed picture can be found here:
https://drive.google.com/file/d/13ZF-pQyr1qubTRvv1nshkmLfimip-pdy/view?usp=sharing



Software:

Atmel Studio 7 software was used.

A scheduler.h file is included in the project which helps create task structures for the main c code:
https://drive.google.com/file/d/1eygkPrKXWyFx4WjmJ4PPD79ALj38bZqq/view?usp=sharing

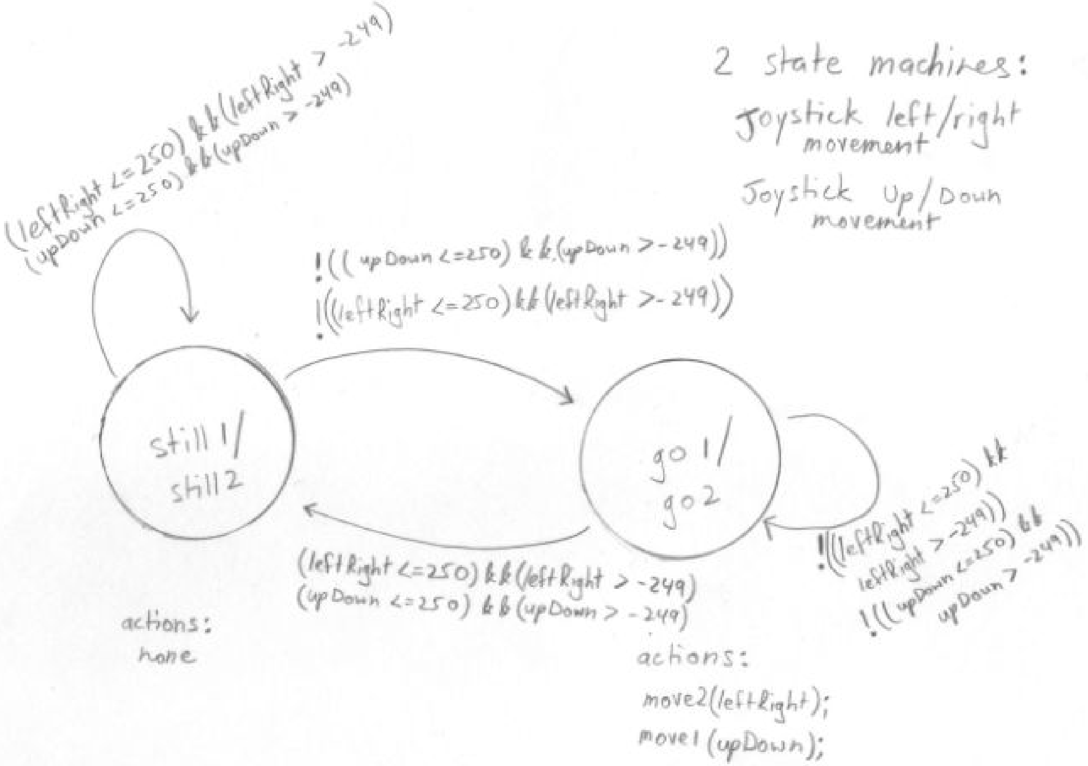
A timer.h file is included in the project which makes use of the ATmega1284's built-in timers for running state machines concurrently:
https://drive.google.com/file/d/1DpbwBiChBDaiASscs6bTzaYFECfDjWZc/view?usp=sharing

An SPI_LCD.c file is included in the project which makes use of the ATmega1284's built-in SPI serial port. This is used for the TFT display module: https://drive.google.com/file/d/1euHuRN_G-2QnH3V_4zdcqW1JTNybYdFc/view?usp=sharing

The main c code for game logic can be found here: https://drive.google.com/file/d/1jI6ksAxJNqejenkesq9oghy_nEESgy_-/view?usp=sharing

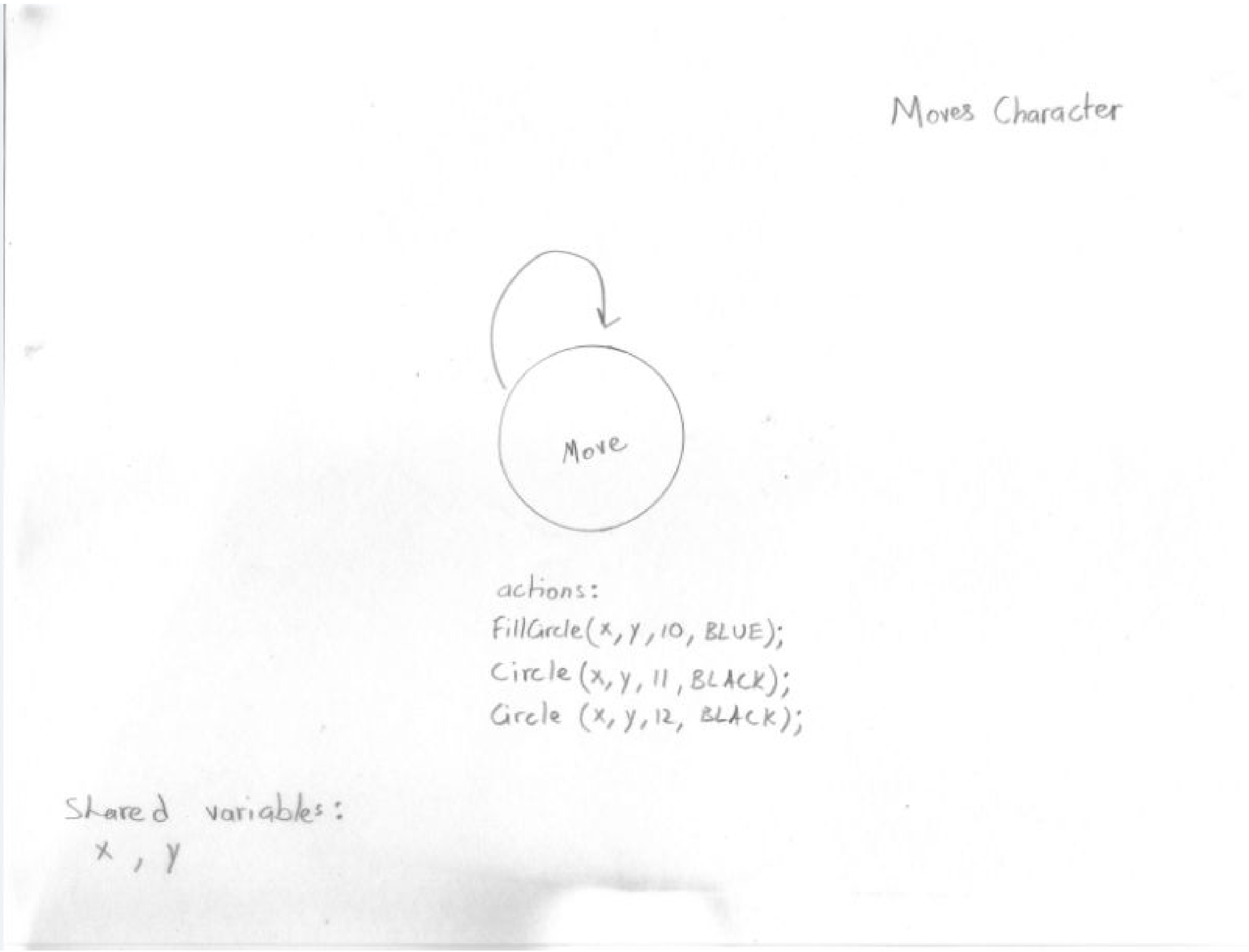
Sources Cited:

- To help me get the joystick working I had a little help from: https://github.com/Dave864/Ascii-Avatar-Maker
- For the SPI protocol I had a little help from: http://w8bh.net/avr/AvrTFT.pdf


(leftRight <=250) && (leftRight > -249)
(upDown <=250) && (upDown > -249)
!((upDown <=250) && (upDown >-249))
!((leftRight <=250) && (leftRight >-249))
still1/
still2
go1/
go2
(leftRight <=250) && (leftRight > -249)
(upDown <=250) && (upDown > -249)
!((leftRight <=250) &&
leftRight > -249))
!((upDown <=250) &&
upDown > -249))
actions:
none
actions:
move2(leftRight);
move1(upDown);

2 state machines:
Joystick left/right movement
Joystick Up/Down movement

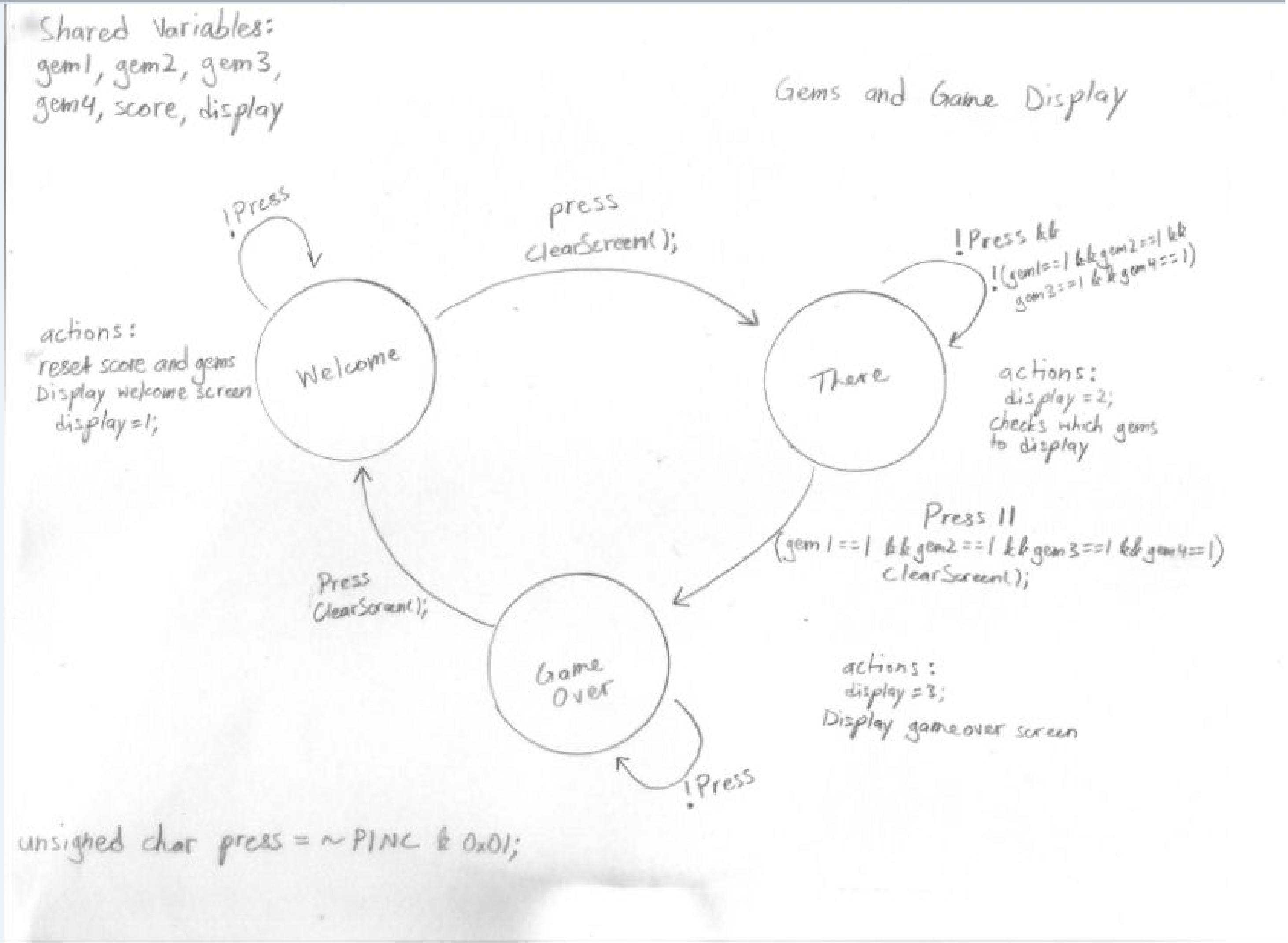
leftRight = ReadADC(2)
upDown = ReadADC(3)

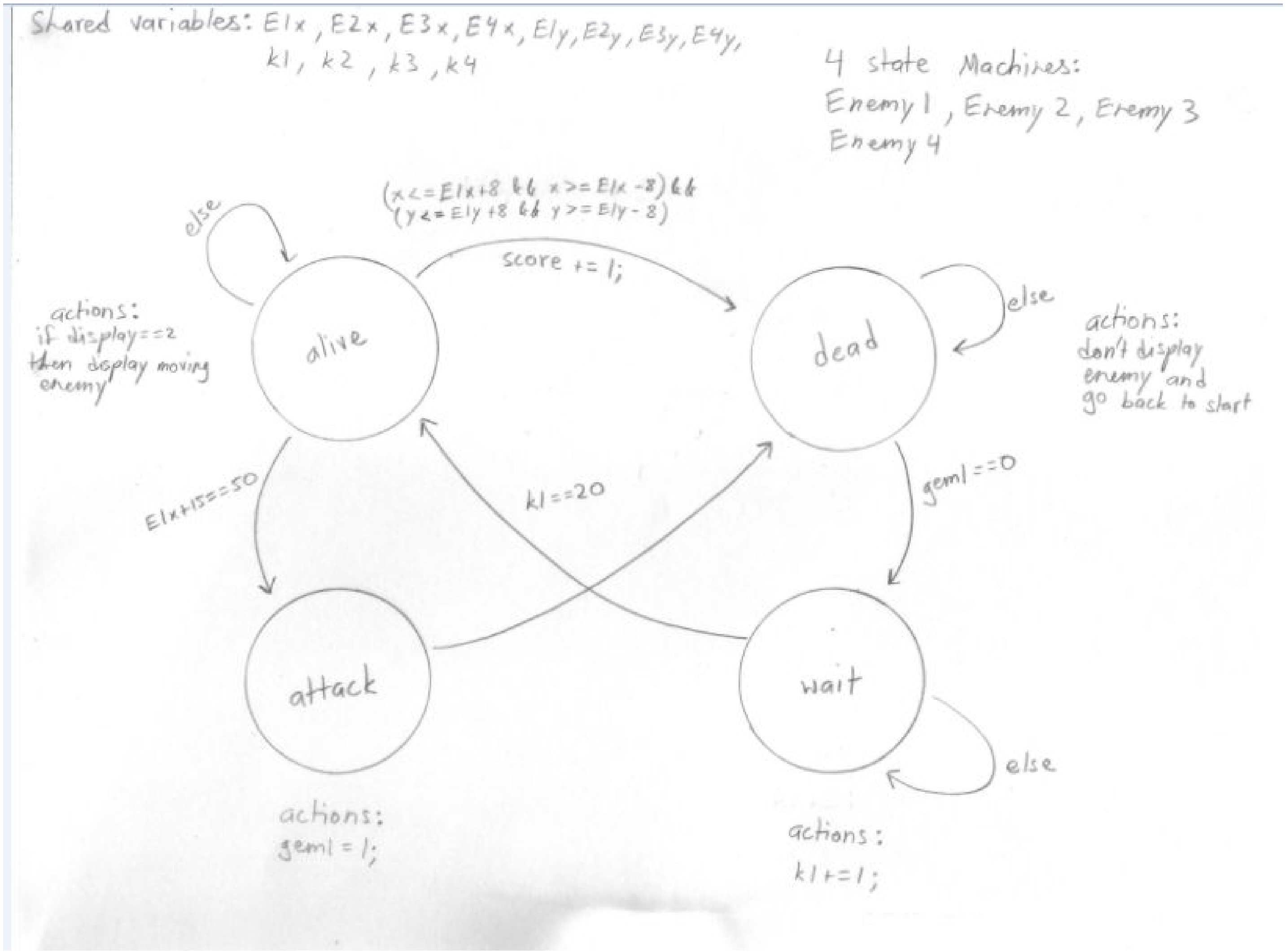


Moves Character

actions:
FillCircle(x, y, 10, BLUE);
Circle (x, y, 11, BLACK);
Circle (x, y, 12, BLACK);

Shared variables:
x , y


Shared Variables:
gem1, gem2, gem3,
gem4, score, display
Gems and Game Display
!Press
press
ClearScreen();
!Press &&
!(gem1==1 && gem2==1 &&
gem3==1 && gem4==1)
actions:
reset score and gems
Display welcome screen
display=1;
Welcome
There
actions:
display=2;
checks which gems
to display
Press ||
(gem1==1 && gem2==1 && gem3==1 && gem4==1)
ClearScreen();
Press
ClearScreen();
Game
Over
actions:
display=3;
Display gameover screen
!Press
unsighed char press = ~PINC & 0x01;


Shared variables: E1x , E2x , E3x , E4x , E1y , E2y , E3y , E4y ,
k1 , k2 , k3 , k4
4 state Machines:
Enemy1 , Enemy 2 , Enemy 3
Enemy 4
else
(x<=E1x+8 && x>=E1x -8) &&
(y<=E1y +8 && y>=E1y - 8)
score += 1;
else
actions:
if display==2
then display moving
enemy
alive
dead
actions:
don't display
enemy and
go back to start
E1x+15==50
k1==20
gem1==0
attack
wait
else
actions:
gem1 = 1;
actions:
k1+=1;

Complexities:

Completed Complexities:

- Joystick
- TFT display module
- Using SPI
- Game Logic

Incomplete Complexities:

- n/a

Youtube Link:

A demo video can be found here: https://youtu.be/X34kf1jgkk4

Known Bugs and Shortcomings:

- The start/restart button has to be pressed a little longer than you would expect. I believe this is because of the period set for the state machine. To fix this I can add a release state to to button press, which can switch over to the next state after the button is released only.

Future Work:

- Include more intricate looking custom characters
- Add random variables for enemy respawn rate
- Add random variables for enemy periods (for their speed)
- Add another TFT display module for a larger playing area