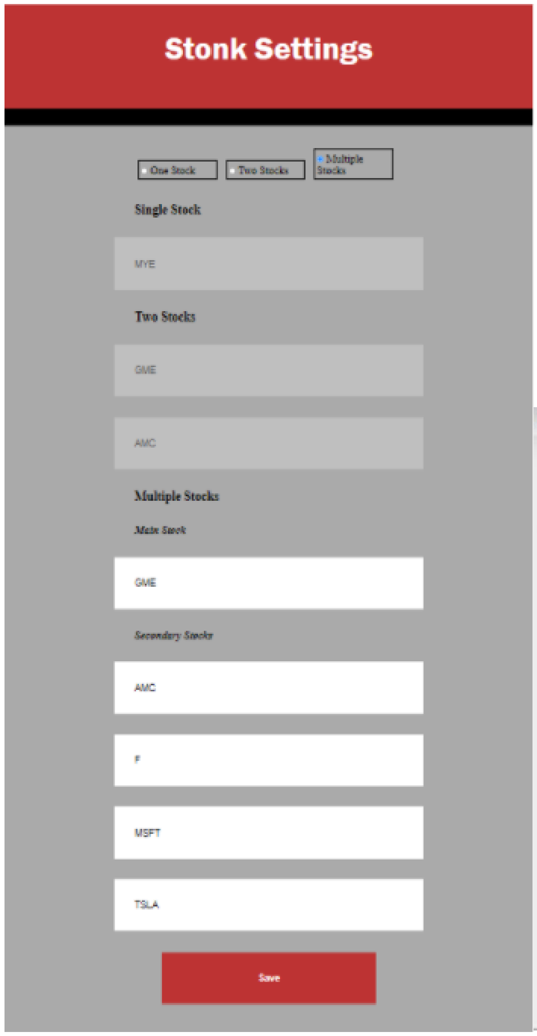
Portfolio

Virtual Stock Ticker – Front End – Java GUI – Virtual LED Matrix



Stonk Settings

One Stock [✓ Multiple Stocks] Two Stocks

Single Stock

MYE

Two Stocks

GME

AMC

Multiple Stocks

Main Stock

GME

Secondary Stocks

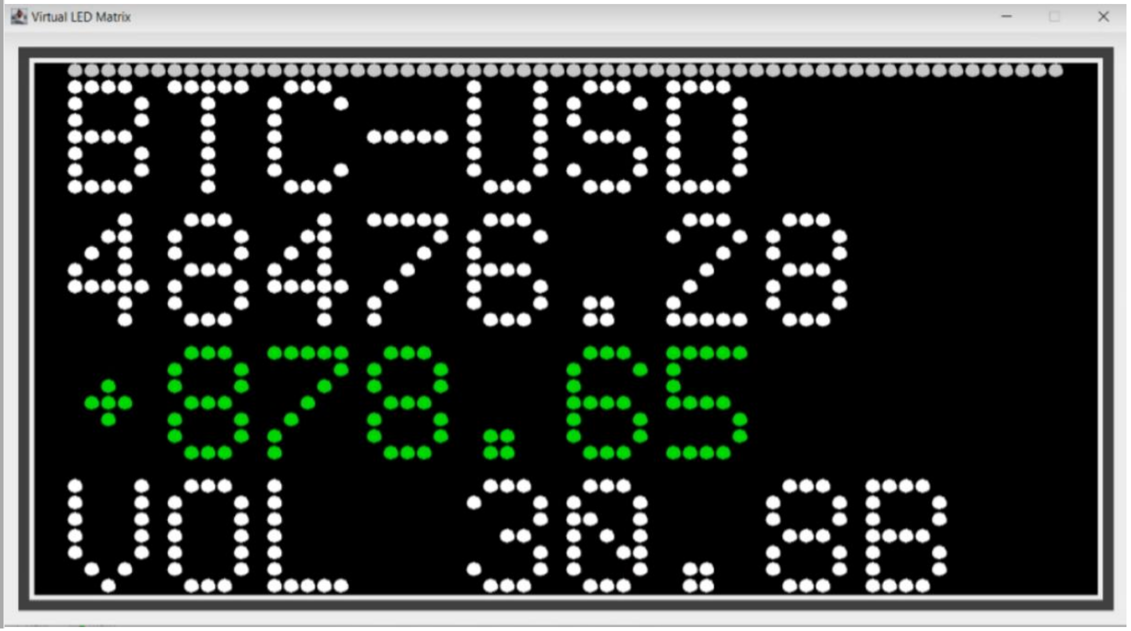
AMC

F

MSFT

TSLA

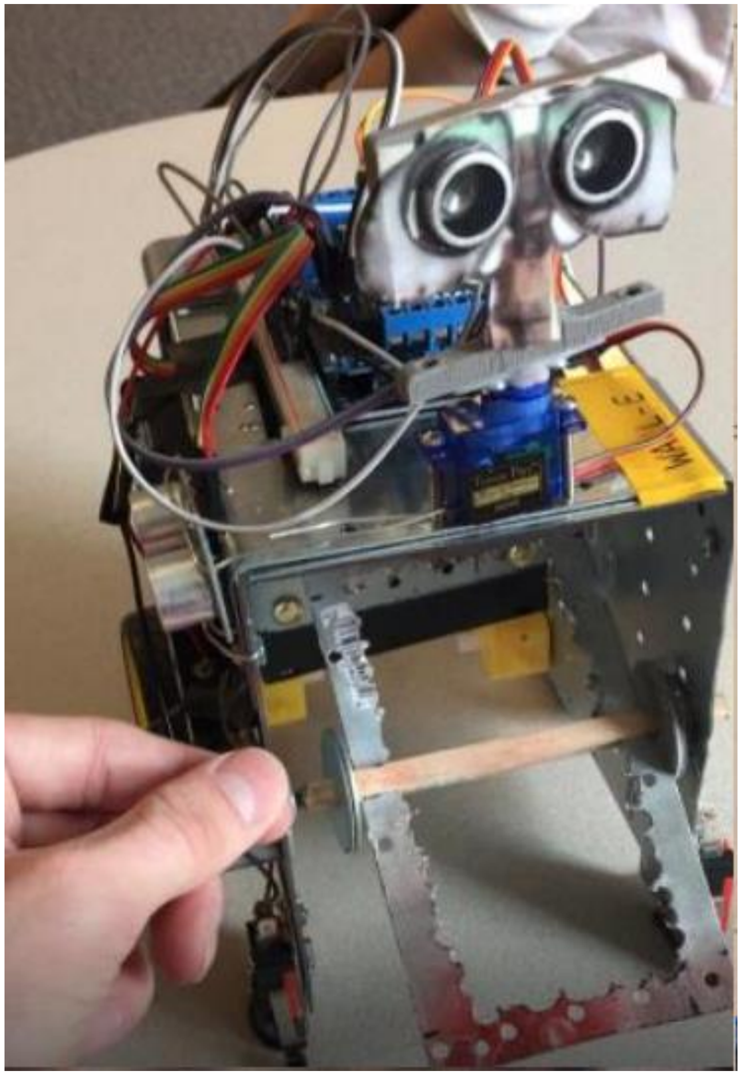
Save



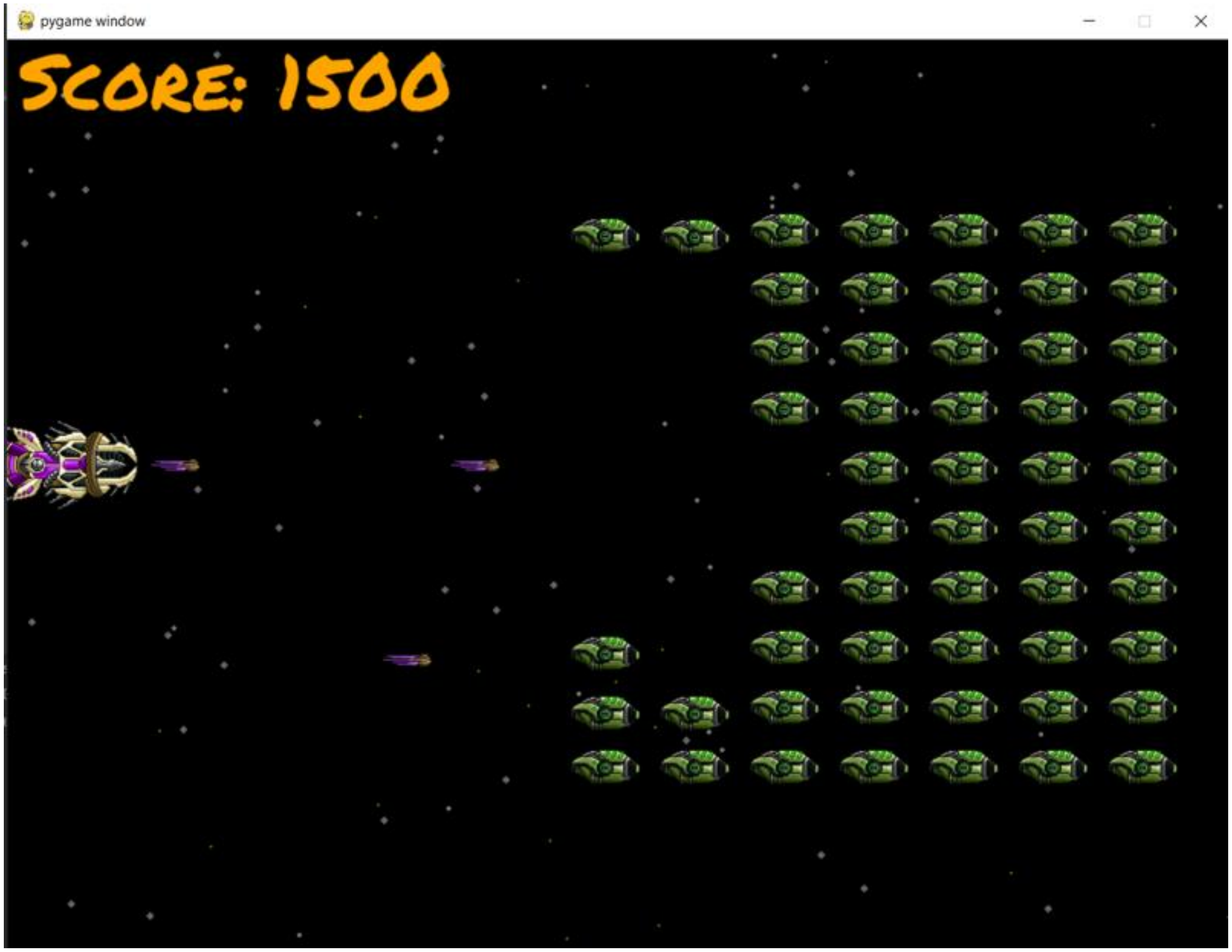
Virtual LED Matrix

```
BTC-USD
48476.28
+878.65
VOL   38.8B
```

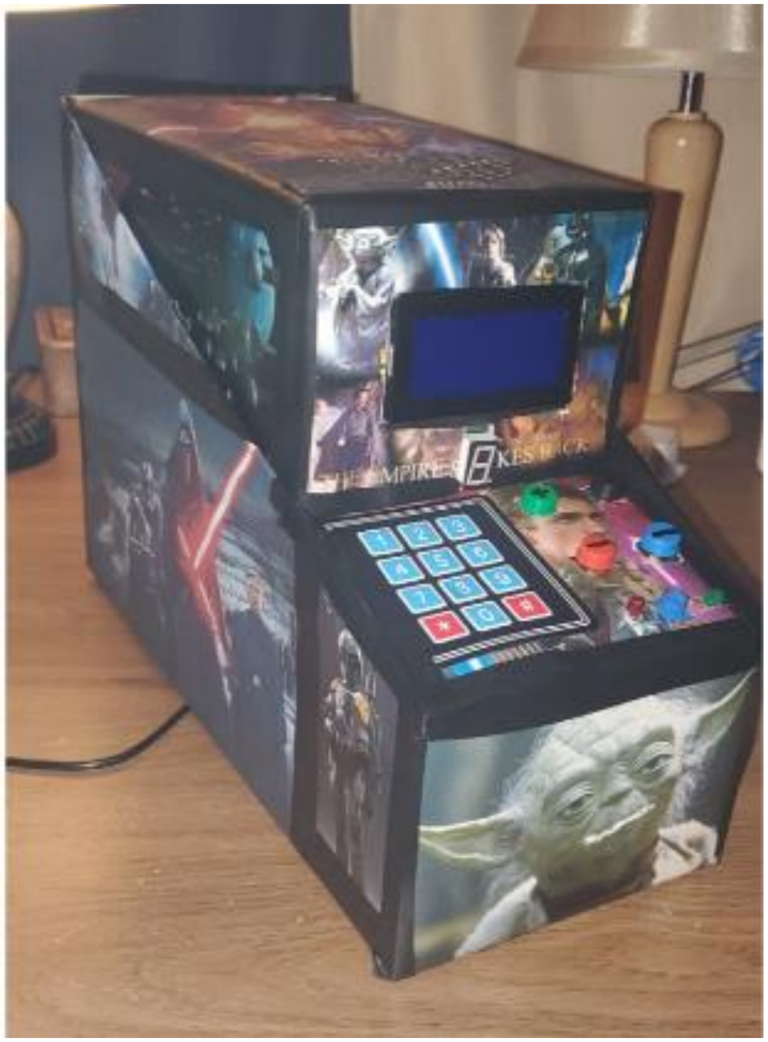
Robot Soccer Competition – Robot

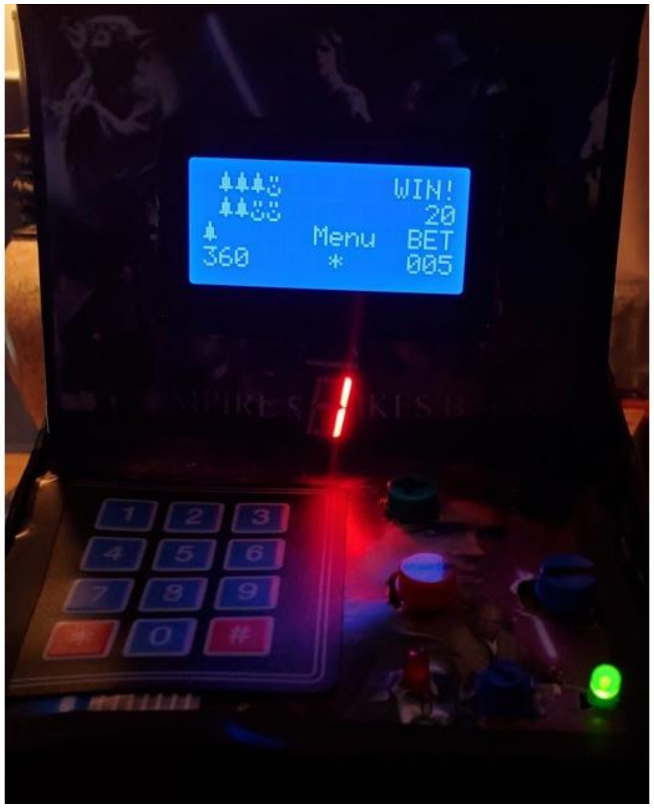


PyGame – Galaga

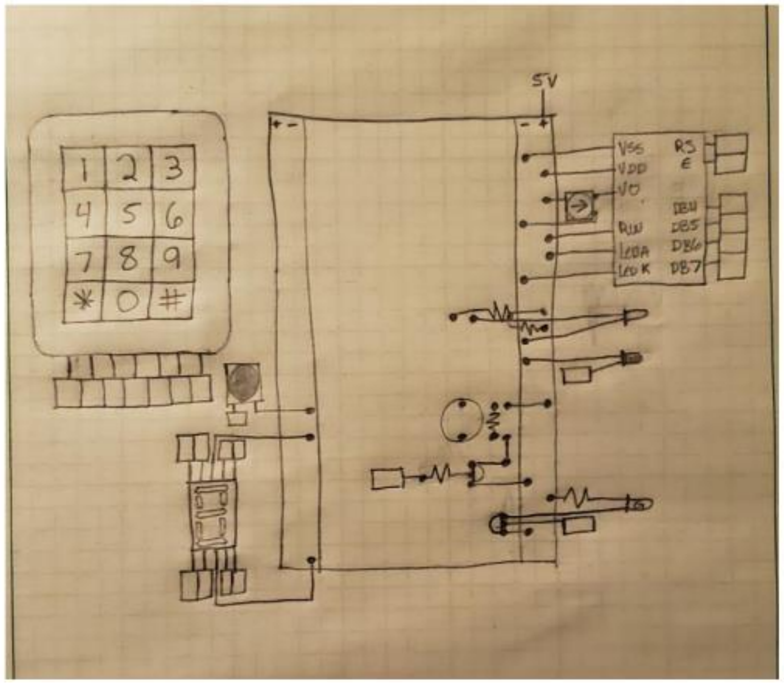


FPGA – Slots Machine

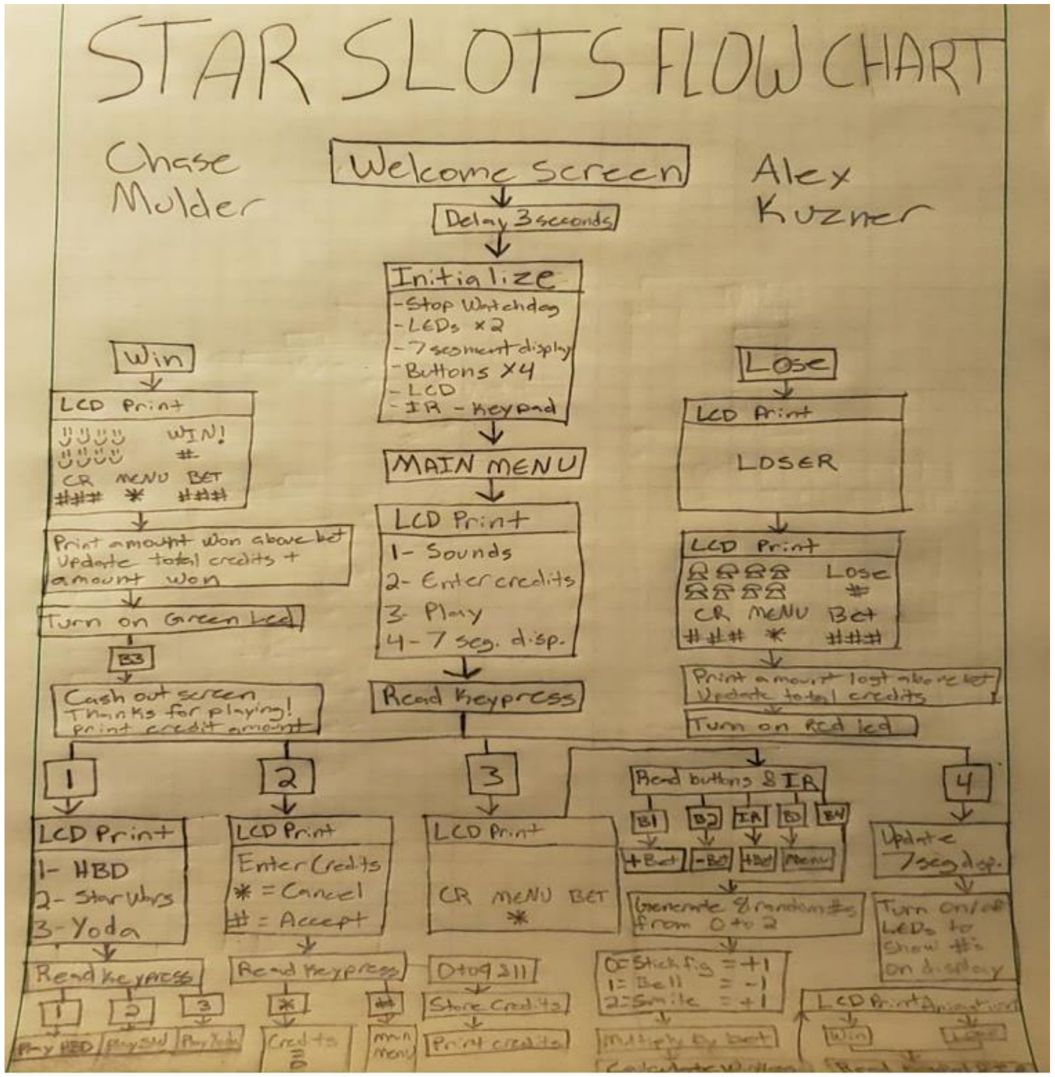




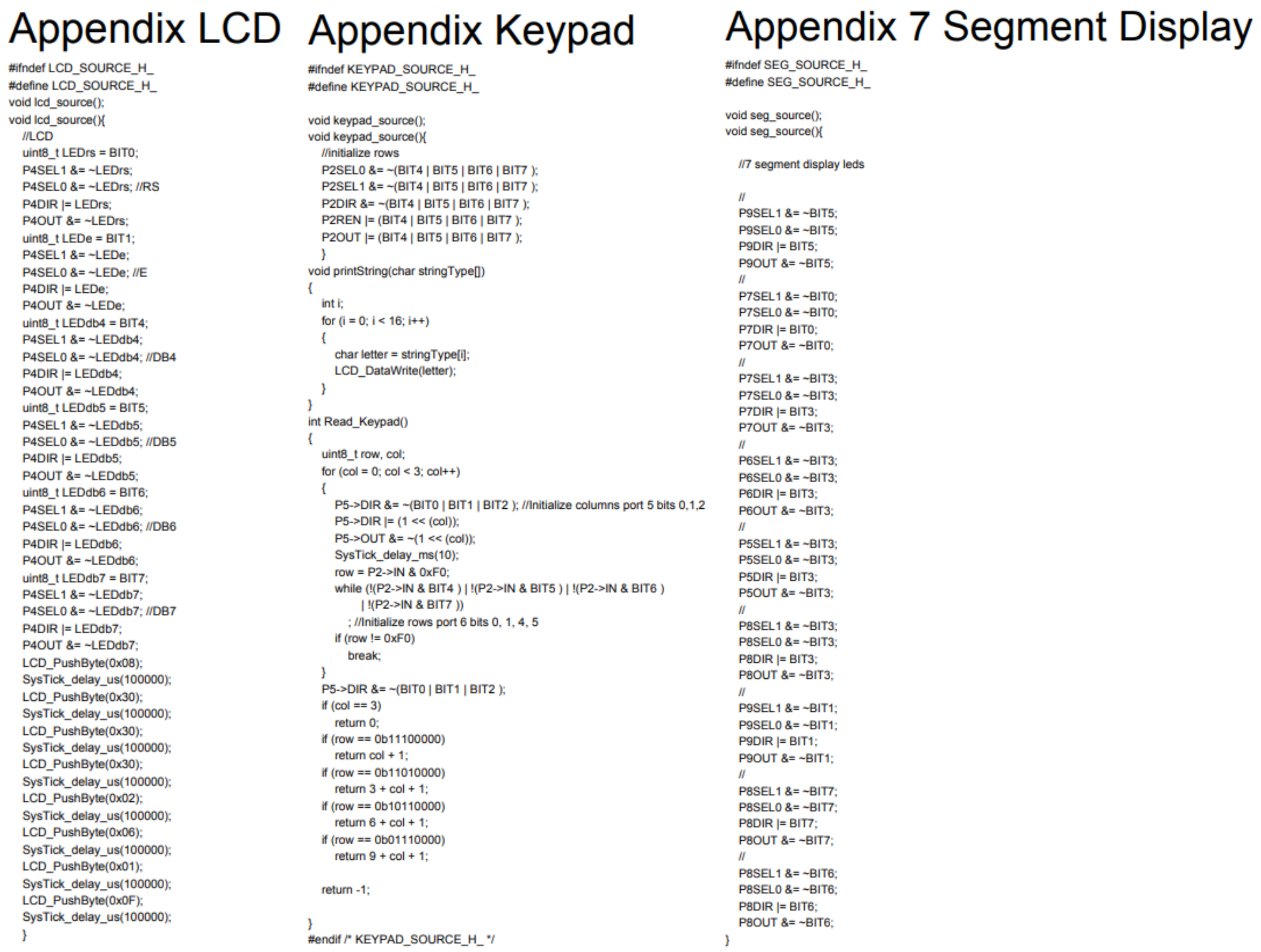
FPGA – Slots Machine – Hardware



FPGA – Slots Machine – High-Level Drawing



FPGA – Slots Machine – Vivado TCL Code



Appendix LCD

```
#ifndef LCD_SOURCE_H_
#define LCD_SOURCE_H_
void lcd_source();
void lcd_source(){
    //LCD
    uint8_t LEDrs = BIT0;
    P4SEL1 &= ~LEDrs;
    P4SEL0 &= ~LEDrs; //RS
    P4DIR |= LEDrs;
    P4OUT &= ~LEDrs;
    uint8_t LEDe = BIT1;
    P4SEL1 &= ~LEDe;
    P4SEL0 &= ~LEDe; //E
    P4DIR |= LEDe;
    P4OUT &= ~LEDe;
    uint8_t LEDdb4 = BIT4;
    P4SEL1 &= ~LEDdb4;
    P4SEL0 &= ~LEDdb4; //DB4
    P4DIR |= LEDdb4;
    P4OUT &= ~LEDdb4;
    uint8_t LEDdb5 = BIT5;
    P4SEL1 &= ~LEDdb5;
    P4SEL0 &= ~LEDdb5; //DB5
    P4DIR |= LEDdb5;
    P4OUT &= ~LEDdb5;
    uint8_t LEDdb6 = BIT6;
    P4SEL1 &= ~LEDdb6;
    P4SEL0 &= ~LEDdb6; //DB6
    P4DIR |= LEDdb6;
    P4OUT &= ~LEDdb6;
    uint8_t LEDdb7 = BIT7;
    P4SEL1 &= ~LEDdb7;
    P4SEL0 &= ~LEDdb7; //DB7
    P4DIR |= LEDdb7;
    P4OUT &= ~LEDdb7;
    LCD_PushByte(0x08);
    SysTick_delay_us(100000);
    LCD_PushByte(0x30);
    SysTick_delay_us(100000);
    LCD_PushByte(0x30);
    SysTick_delay_us(100000);
    LCD_PushByte(0x30);
    SysTick_delay_us(100000);
    LCD_PushByte(0x02);
    SysTick_delay_us(100000);
    LCD_PushByte(0x06);
    SysTick_delay_us(100000);
    LCD_PushByte(0x01);
    SysTick_delay_us(100000);
    LCD_PushByte(0x0F);
    SysTick_delay_us(100000);
    }
```

Appendix Keypad

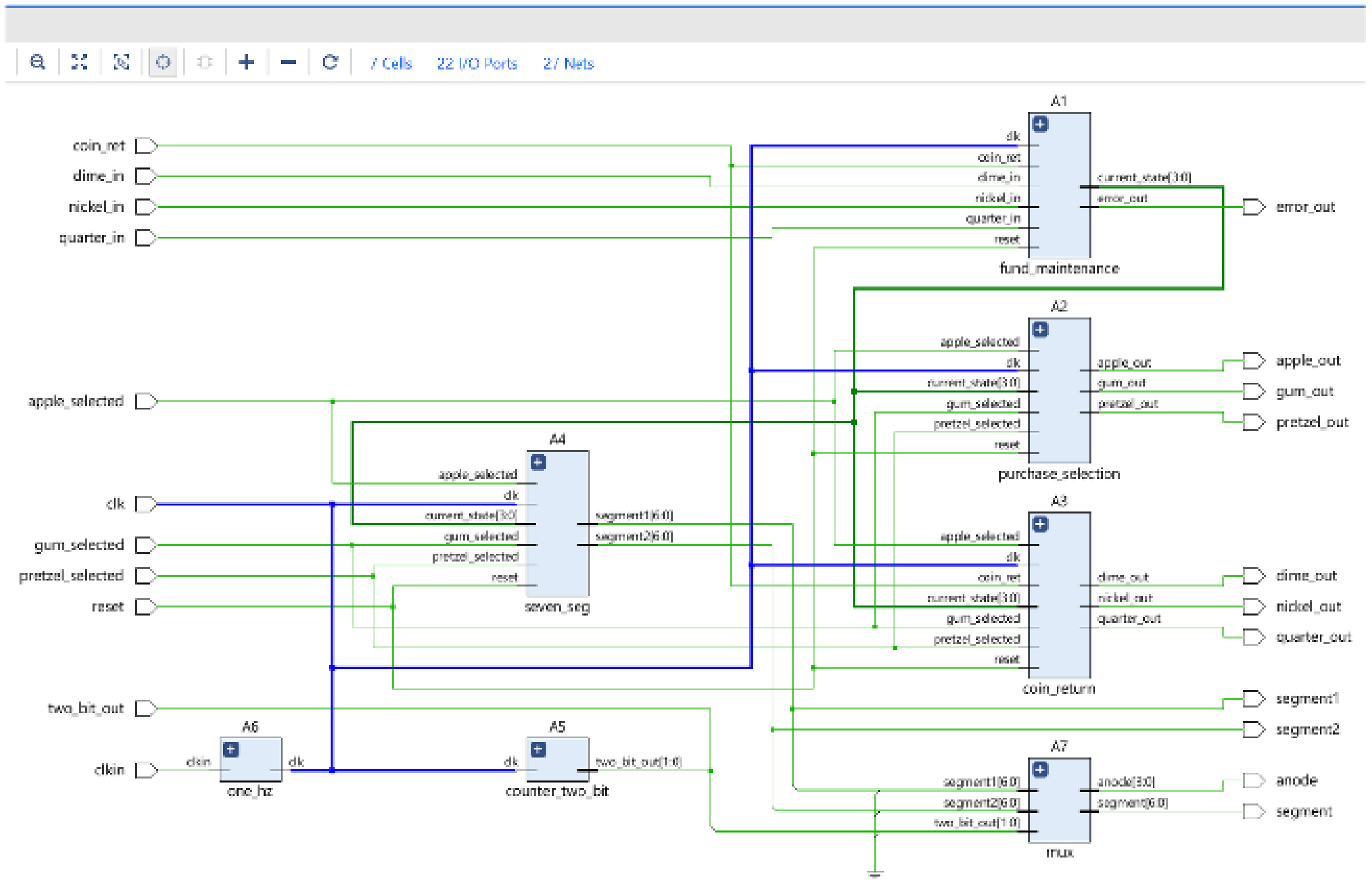
```
#ifndef KEYPAD_SOURCE_H_
#define KEYPAD_SOURCE_H_

void keypad_source();
void keypad_source(){
    //initialize rows
    P2SEL0 &= ~(BIT4 | BIT5 | BIT6 | BIT7 );
    P2SEL1 &= ~(BIT4 | BIT5 | BIT6 | BIT7 );
    P2DIR &= ~(BIT4 | BIT5 | BIT6 | BIT7 );
    P2REN |= (BIT4 | BIT5 | BIT6 | BIT7 );
    P2OUT |= (BIT4 | BIT5 | BIT6 | BIT7 );
    }
void printString(char stringType[])
{
    int i;
    for (i = 0; i < 16; i++)
    {
        char letter = stringType[i];
        LCD_DataWrite(letter);
    }
}
int Read_Keypad()
{
    uint8_t row, col;
    for (col = 0; col < 3; col++)
    {
        P5->DIR &= ~(BIT0 | BIT1 | BIT2 ); //Initialize columns port 5 bits 0,1,2
        P5->DIR |= (1 << (col));
        P5->OUT &= ~(1 << (col));
        SysTick_delay_ms(10);
        row = P2->IN & 0xF0;
        while (!(P2->IN & BIT4 ) | !(P2->IN & BIT5 ) | !(P2->IN & BIT6 )
              | !(P2->IN & BIT7 ))
            ; //Initialize rows port 6 bits 0, 1, 4, 5
        if (row != 0xF0)
            break;
    }
    P5->DIR &= ~(BIT0 | BIT1 | BIT2 );
    if (col == 3)
        return 0;
    if (row == 0b11100000)
        return col + 1;
    if (row == 0b11010000)
        return 3 + col + 1;
    if (row == 0b10110000)
        return 6 + col + 1;
    if (row == 0b01110000)
        return 9 + col + 1;

    return -1;

}
#endif /* KEYPAD_SOURCE_H_ */
```

Appendix 7 Segment Display

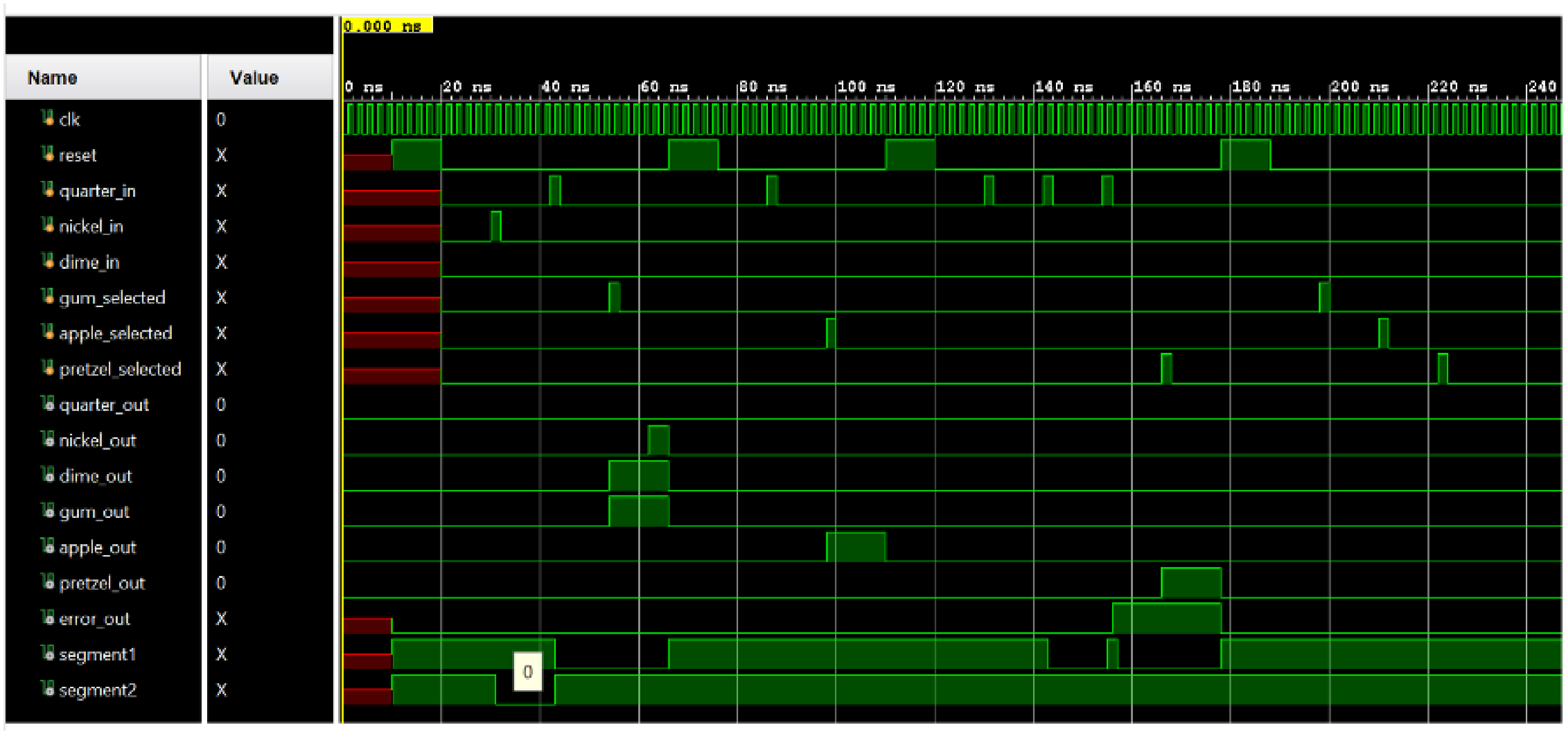
```
#ifndef SEG_SOURCE_H_
#define SEG_SOURCE_H_

void seg_source();
void seg_source(){

    //7 segment display leds

    //
    P9SEL1 &= ~BIT5;
    P9SEL0 &= ~BIT5;
    P9DIR |= BIT5;
    P9OUT &= ~BIT5;
    //
    P7SEL1 &= ~BIT0;
    P7SEL0 &= ~BIT0;
    P7DIR |= BIT0;
    P7OUT &= ~BIT0;
    //
    P7SEL1 &= ~BIT3;
    P7SEL0 &= ~BIT3;
    P7DIR |= BIT3;
    P7OUT &= ~BIT3;
    //
    P6SEL1 &= ~BIT3;
    P6SEL0 &= ~BIT3;
    P6DIR |= BIT3;
    P6OUT &= ~BIT3;
    //
    P5SEL1 &= ~BIT3;
    P5SEL0 &= ~BIT3;
    P5DIR |= BIT3;
    P5OUT &= ~BIT3;
    //
    P8SEL1 &= ~BIT3;
    P8SEL0 &= ~BIT3;
    P8DIR |= BIT3;
    P8OUT &= ~BIT3;
    //
    P9SEL1 &= ~BIT1;
    P9SEL0 &= ~BIT1;
    P9DIR |= BIT1;
    P9OUT &= ~BIT1;
    //
    P8SEL1 &= ~BIT7;
    P8SEL0 &= ~BIT7;
    P8DIR |= BIT7;
    P8OUT &= ~BIT7;
    //
    P8SEL1 &= ~BIT6;
    P8SEL0 &= ~BIT6;
    P8DIR |= BIT6;
    P8OUT &= ~BIT6;
}
```

FPGA – Vending Machine – Software

Figure 11: Vending Machine Schematic

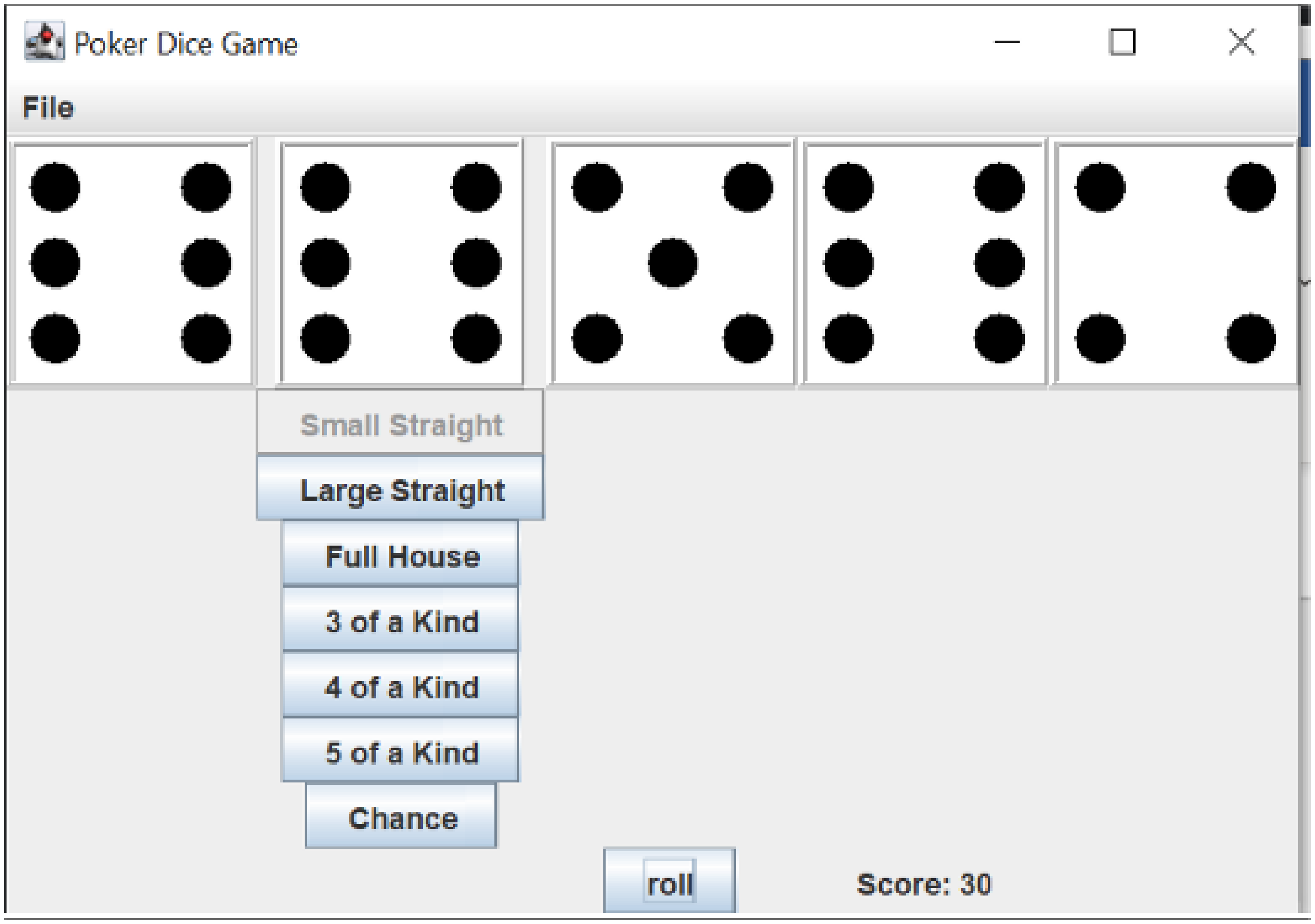


FPGA – Vending Machine – Software Simulation

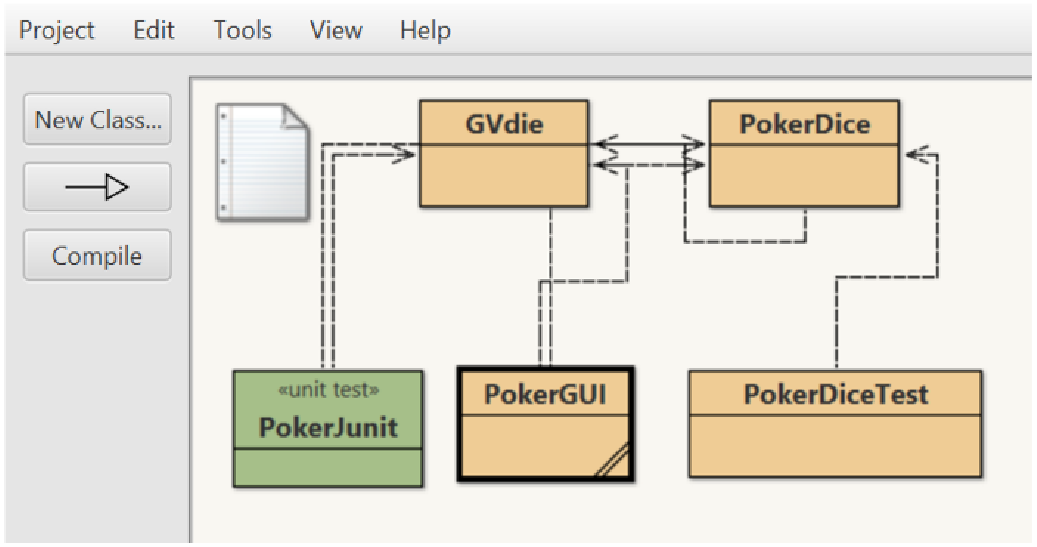


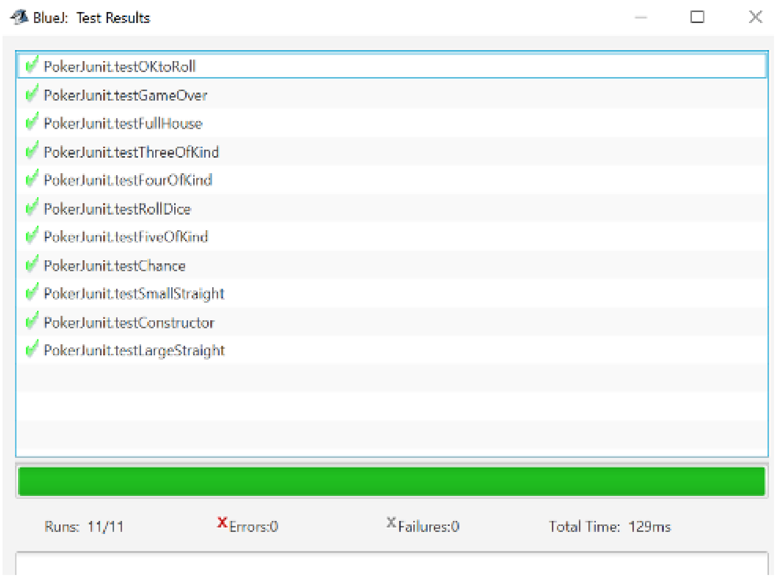
Name	Value
clk	0
reset	X
quarter_in	X
nickel_in	X
dime_in	X
gum_selected	X
apple_selected	X
pretzel_selected	X
quarter_out	0
nickel_out	0
dime_out	0
gum_out	0
apple_out	0
pretzel_out	0
error_out	X
segment1	X
segment2	X

Java GUI – Yahtzee



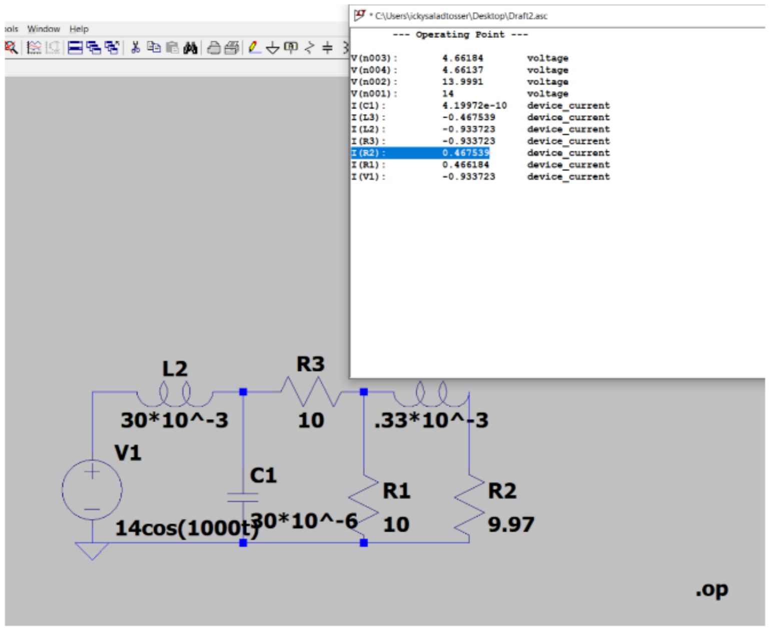
Java – Testing



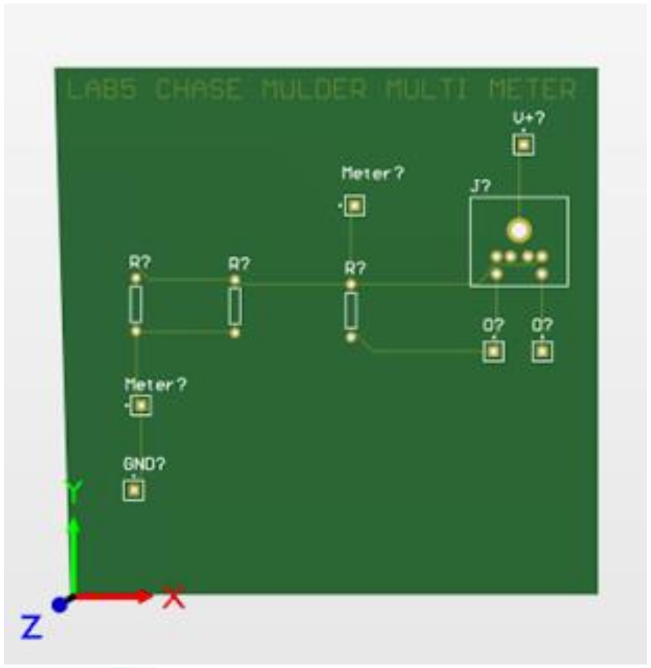


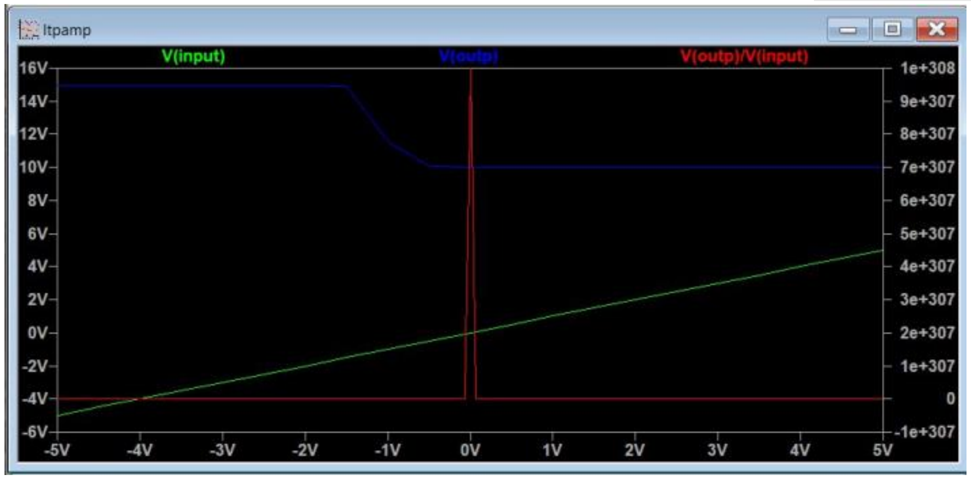
```
✓ PokerJunit.testOKtoRoll
✓ PokerJunit.testGameOver
✓ PokerJunit.testFullHouse
✓ PokerJunit.testThreeOfKind
✓ PokerJunit.testFourOfKind
✓ PokerJunit.testRollDice
✓ PokerJunit.testFiveOfKind
✓ PokerJunit.testChance
✓ PokerJunit.testSmallStraight
✓ PokerJunit.testConstructor
✓ PokerJunit.testLargeStraight

Runs: 11/11     Errors:0     Failures:0     Total Time: 129ms
```

Circuits I – PCB – Testing

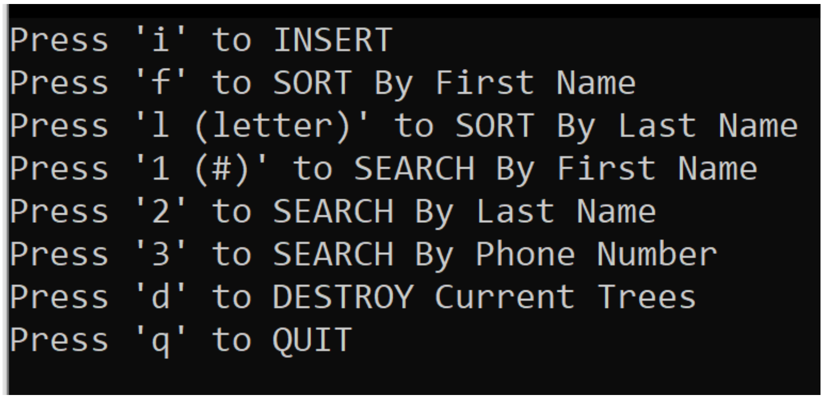


```
        --- Operating Point ---

V(n003):         4.66184        voltage
V(n004):         4.66137        voltage
V(n002):         13.9991        voltage
V(n001):         14             voltage
I(C1):           4.19972e-10    device_current
I(L3):           -0.467539      device_current
I(L2):           -0.933723      device_current
I(R3):           -0.933723      device_current
I(R2):           0.467539       device_current
I(R1):           0.466184       device_current
I(V1):           -0.933723      device_current
```



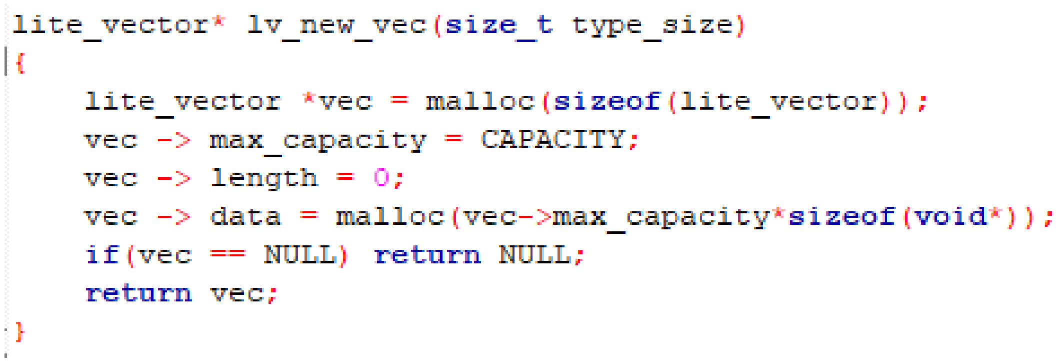


Phone Directory – Linux - Vim – C code



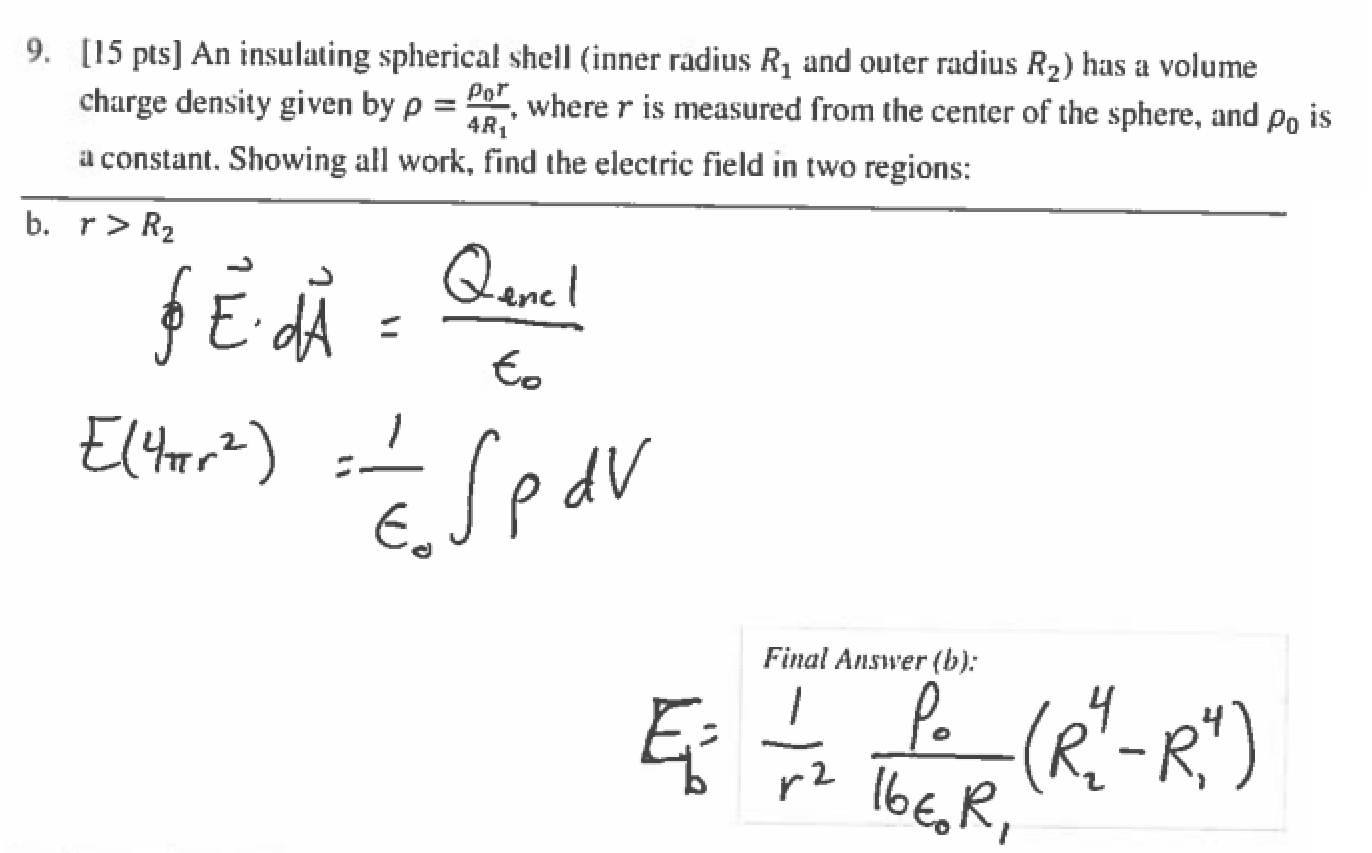
```
Press 'i' to INSERT
Press 'f' to SORT By First Name
Press 'l (letter)' to SORT By Last Name
Press '1 (#)' to SEARCH By First Name
Press '2' to SEARCH By Last Name
Press '3' to SEARCH By Phone Number
Press 'd' to DESTROY Current Trees
Press 'q' to QUIT
```

Dynamically Sized Vector – Java



```
lite_vector* lv_new_vec(size_t type_size)
{
    lite_vector *vec = malloc(sizeof(lite_vector));
    vec -> max_capacity = CAPACITY;
    vec -> length = 0;
    vec -> data = malloc(vec->max_capacity*sizeof(void*));
    if(vec == NULL) return NULL;
    return vec;
}
```

Hardest Engineering Physics II Problem



9. [15 pts] An insulating spherical shell (inner radius R_1 and outer radius R_2) has a volume charge density given by $\rho = \frac{\rho_0 r}{4R_1}$, where r is measured from the center of the sphere, and ρ_0 is a constant. Showing all work, find the electric field in two regions:

b. $r > R_2$

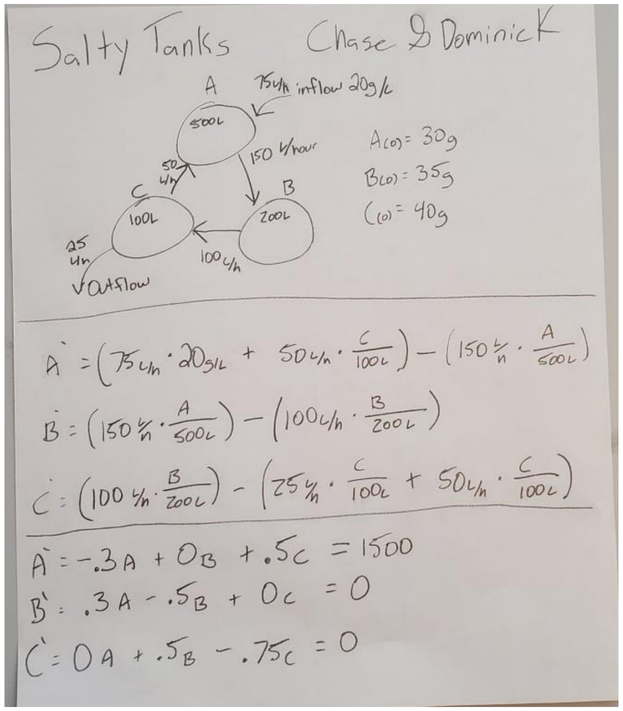
$$\oint \vec{E} \cdot d\vec{A} = \frac{Q_{encl}}{\epsilon_0}$$

$$E(4\pi r^2) = \frac{1}{\epsilon_0} \int \rho \, dV$$

Final Answer (b):

$$E_b = \boxed{\frac{1}{r^2} \frac{\rho_0}{16\epsilon_0 R_1} (R_2^4 - R_1^4)}$$

Diffy Q – Final Project



Salty Tanks   Chase & Dominick

A
75 L/h inflow 20g/L
500L
150 L/hour
B
200L
100 L/h
C
100L
50 L/h
25 L/h
Outflow

$A(0) = 30g$

$B(0) = 35g$

$C(0) = 40g$

$$\dot{A} = \left(75\,L/h \cdot 20\,g/L + 50\,L/h \cdot \frac{C}{100L}\right) - \left(150\,L/h \cdot \frac{A}{500L}\right)$$

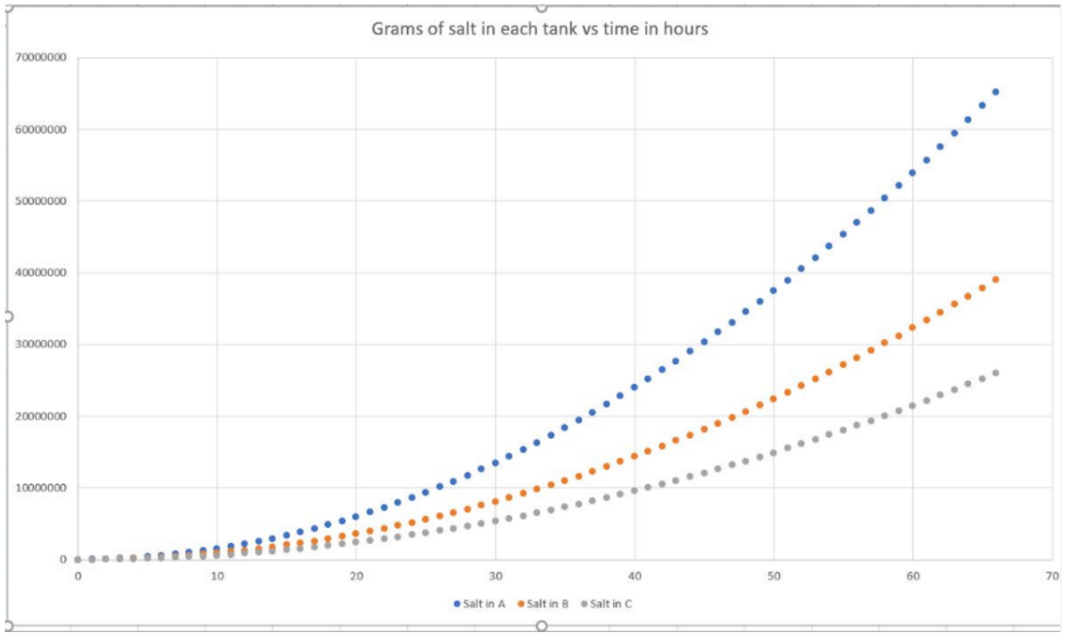
$$\dot{B} = \left(150\,L/h \cdot \frac{A}{500L}\right) - \left(100\,L/h \cdot \frac{B}{200L}\right)$$

$$\dot{C} = \left(100\,L/h \cdot \frac{B}{200L}\right) - \left(25\,L/h \cdot \frac{C}{100L} + 50\,L/h \cdot \frac{C}{100L}\right)$$

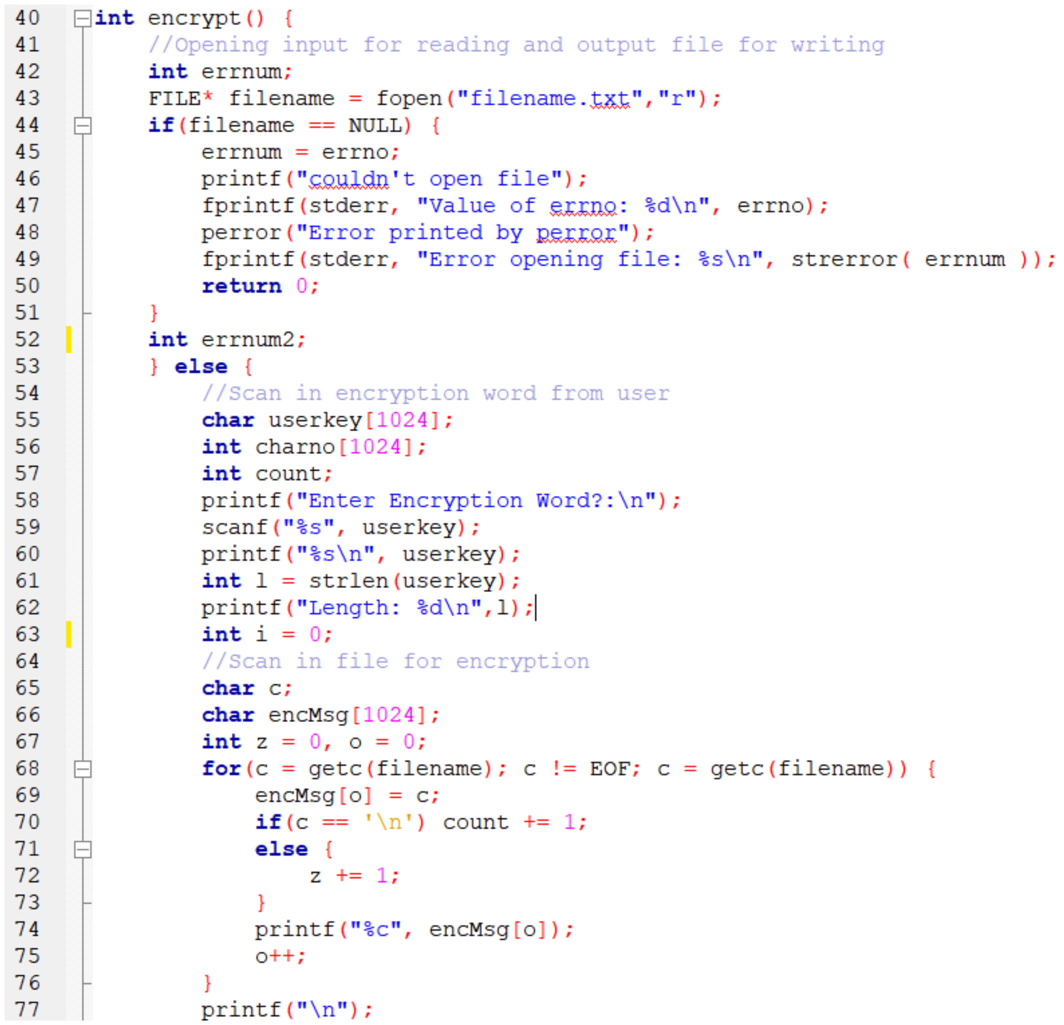
$$A' = -.3_A + 0_B + .5_C = 1500$$

$$B' = .3_A - .5_B + 0_C = 0$$

$$C' = 0_A + .5_B - .75_C = 0$$

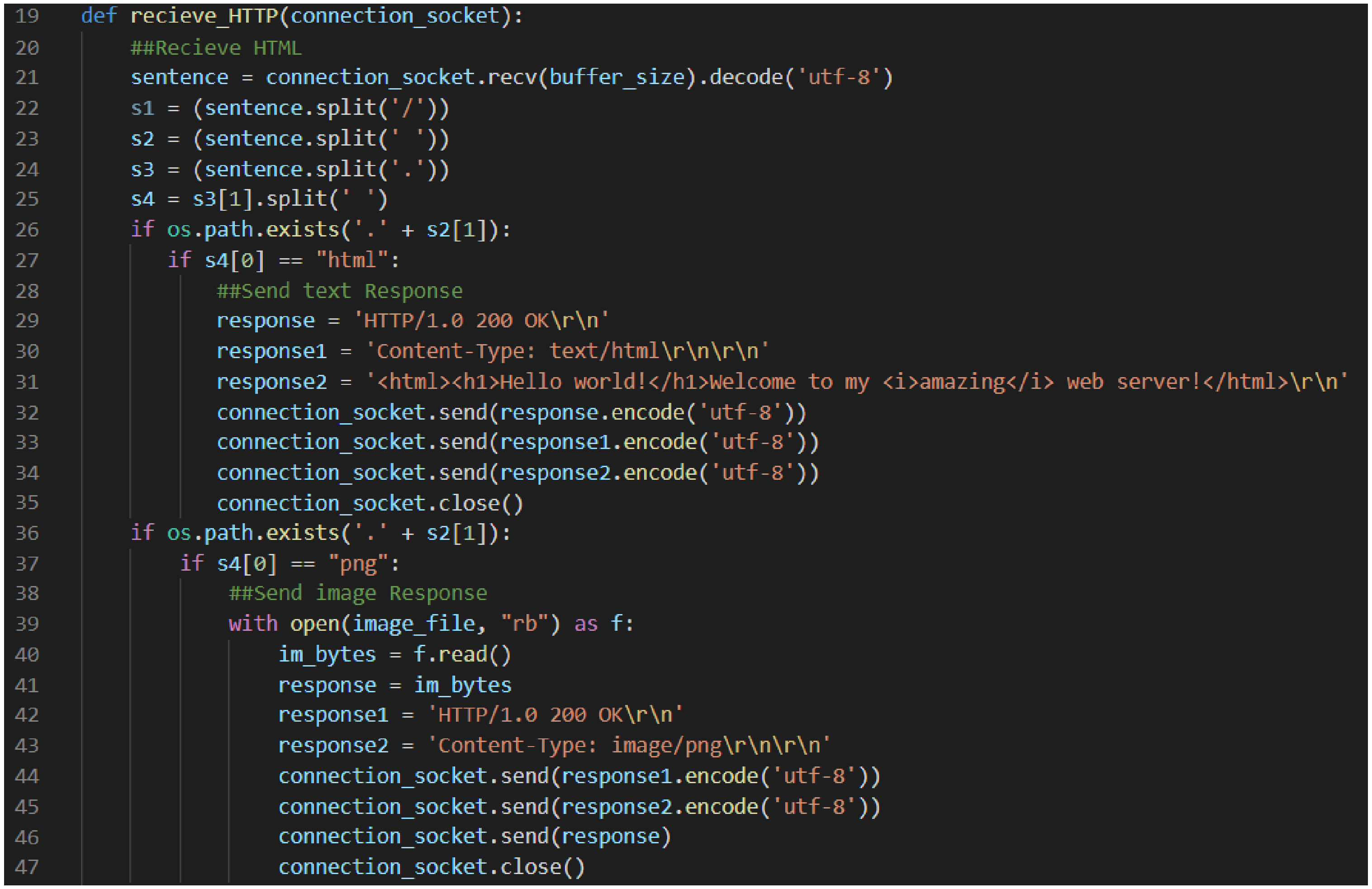


C Code – Encryption From User



```c
int encrypt() {
    //Opening input for reading and output file for writing
    int errnum;
    FILE* filename = fopen("filename.txt","r");
    if(filename == NULL) {
        errnum = errno;
        printf("couldn't open file");
        fprintf(stderr, "Value of errno: %d\n", errno);
        perror("Error printed by perror");
        fprintf(stderr, "Error opening file: %s\n", strerror( errnum ));
        return 0;
    }
    int errnum2;
    } else {
        //Scan in encryption word from user
        char userkey[1024];
        int charno[1024];
        int count;
        printf("Enter Encryption Word?:\n");
        scanf("%s", userkey);
        printf("%s\n", userkey);
        int l = strlen(userkey);
        printf("Length: %d\n",l);
        int i = 0;
        //Scan in file for encryption
        char c;
        char encMsg[1024];
        int z = 0, o = 0;
        for(c = getc(filename); c != EOF; c = getc(filename)) {
            encMsg[o] = c;
            if(c == '\n') count += 1;
            else {
                z += 1;
            }
            printf("%c", encMsg[o]);
            o++;
        }
        printf("\n");
```

TCP Server Multithreading - Python



```python
def recieve_HTTP(connection_socket):
    ##Recieve HTML
    sentence = connection_socket.recv(buffer_size).decode('utf-8')
    s1 = (sentence.split('/'))
    s2 = (sentence.split(' '))
    s3 = (sentence.split('.'))
    s4 = s3[1].split(' ')
    if os.path.exists('.' + s2[1]):
       if s4[0] == "html":
           ##Send text Response
           response = 'HTTP/1.0 200 OK\r\n'
           response1 = 'Content-Type: text/html\r\n\r\n'
           response2 = '<html><h1>Hello world!</h1>Welcome to my <i>amazing</i> web server!</html>\r\n'
           connection_socket.send(response.encode('utf-8'))
           connection_socket.send(response1.encode('utf-8'))
           connection_socket.send(response2.encode('utf-8'))
           connection_socket.close()
    if os.path.exists('.' + s2[1]):
        if s4[0] == "png":
            ##Send image Response
            with open(image_file, "rb") as f:
                im_bytes = f.read()
                response = im_bytes
                response1 = 'HTTP/1.0 200 OK\r\n'
                response2 = 'Content-Type: image/png\r\n\r\n'
                connection_socket.send(response1.encode('utf-8'))
                connection_socket.send(response2.encode('utf-8'))
                connection_socket.send(response)
                connection_socket.close()
```