# Dance Dance Revolution: Follow The Light

Custom Project Final Report

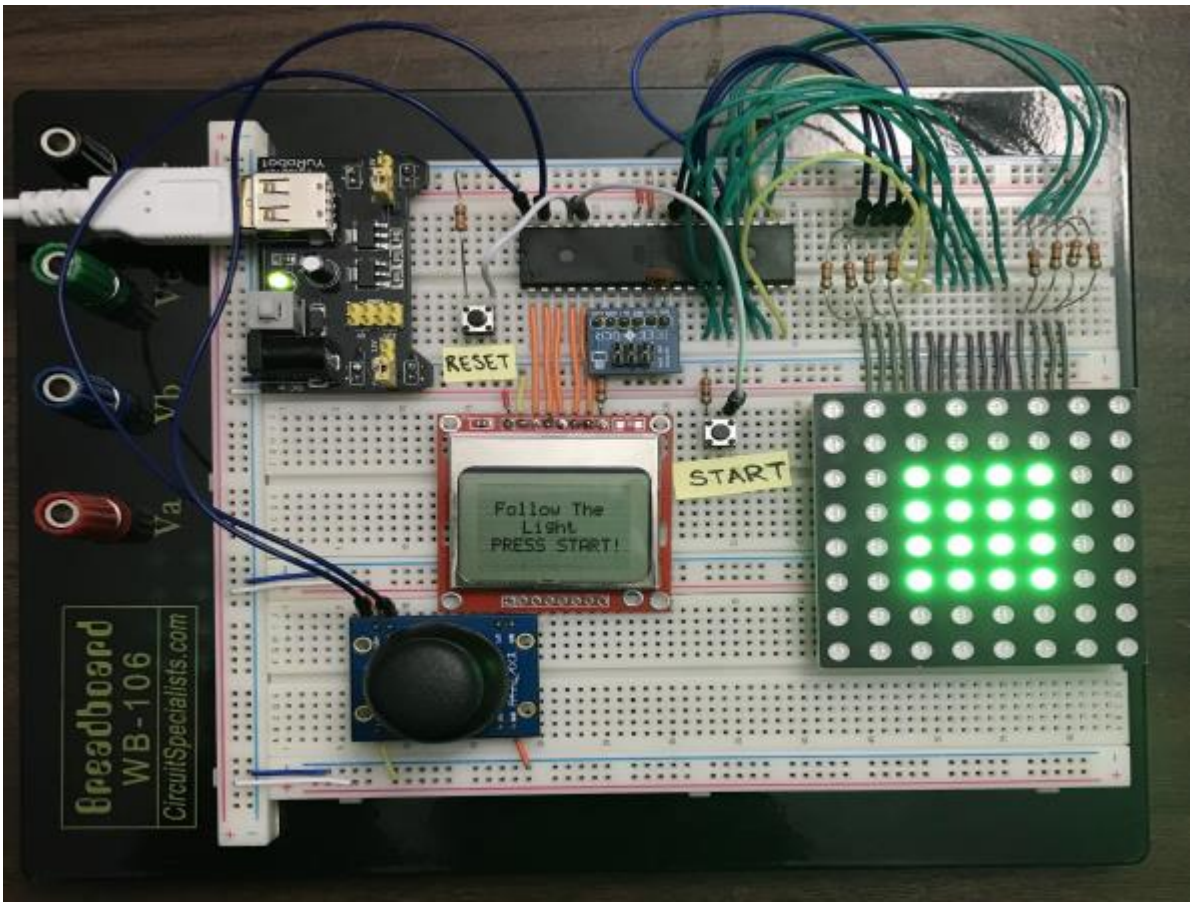
Spring 2017

Srisri Gokanapudy

# Table of Contents

# Introduction

Correctly move the joystick to the directions displayed on the LED Matrix. Messages, total correct, and total wrong will be displayed on a NOKIA 5110 LCD screen. You have 13 tries and if you get more correct than wrong you win.




# 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.

- ATMega1284 microcontroller

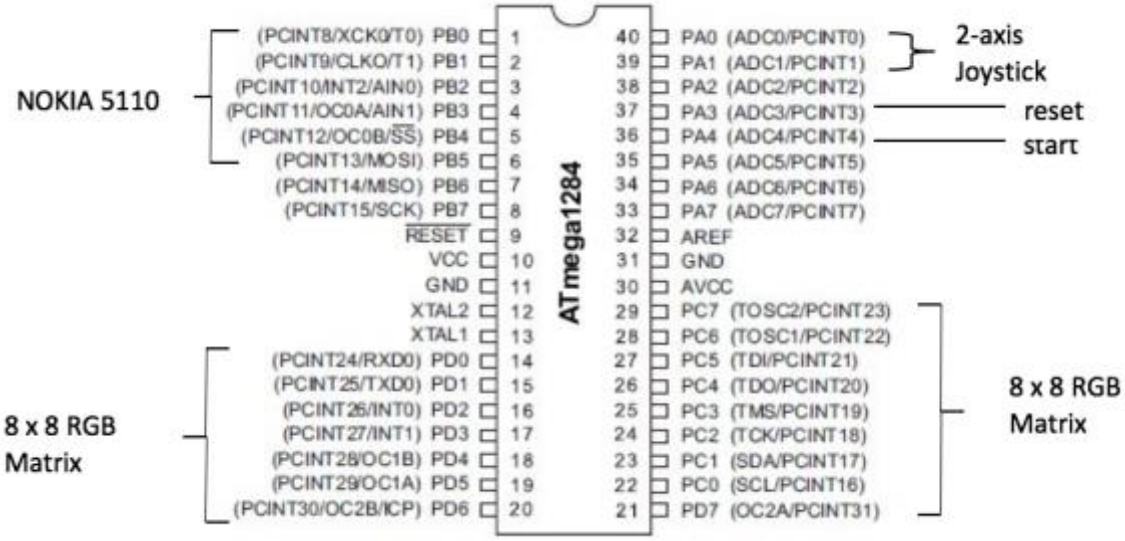
**Inputs:**

- **2-axis Joystick** for user input
- 2 Buttons for Start and Reset

**Outputs:**

- **Nokia 5110 LCD screen** to display intro message and score
- **8 x 8 RGB LED Matrix** which displays game instructions

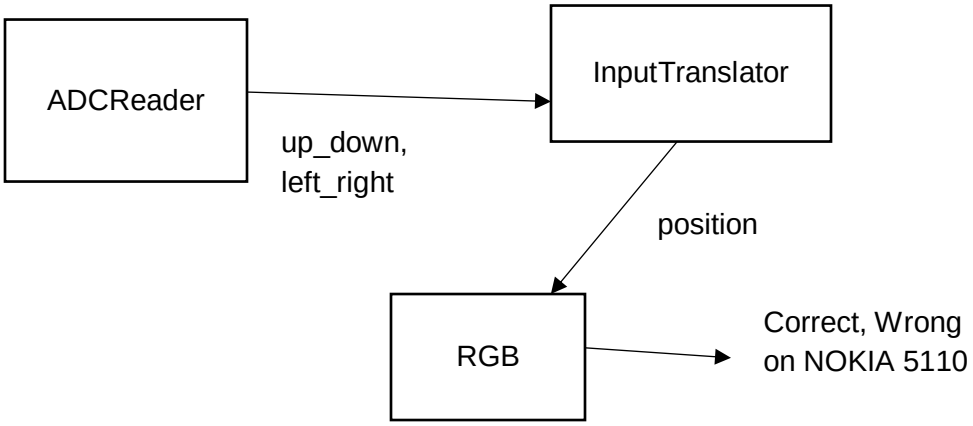
*Pinout*




Reset and Start are buttons.

# Software (Task Diagram)

The software designed for this project was implemented using the PES standard. The overall design as a task diagram is included below:



Now I will write a short description of the tasks in the project. The appendix will include SM's that I designed:

**ADCReader:** reads in input from the user and stores the x-axis and y-axis values in left_right and up_down respectively.

**InputTranslator:** reads in the up_down and left_right values and sets position accordingly. For example if the up_down value is greater than 165, this sets position to 1 indicating that the user has moved up.

**RGB:** Displays the instructions (up, down, left) as 1, 2, 3 and checks if the desired position matches the instruction and increments the number correct if correct or the number wrong if wrong. Correct and wrong are updated after every user input.

# Complexities

## *Completed Complexities:*

- Integrating and calibrating the joystick
- Writing to and display on the NOKIA 5110
- Display custom patterns on the RGB matrix

No Uncompleted Complexities.

# YouTube link

https://youtu.be/rbXPbxeJ4FI

# Known Bugs and Shortcomings

- Rarely, some joystick movements aren't accounted for and some are accounted for twice. This happens when the movements are made "too slow" or "too fast" for the joystick to track. I tried adjusting the time periods so that input waits after checking once and found a 3 count wait to be optimal. Perhaps adjust my polling to use registers is something I could do in the future.

# Future work

- I will add a randomizer to randomize which direction the joystick must point to. This would make each game unique.
- I will incorporate the “right” direction on the joystick

# References

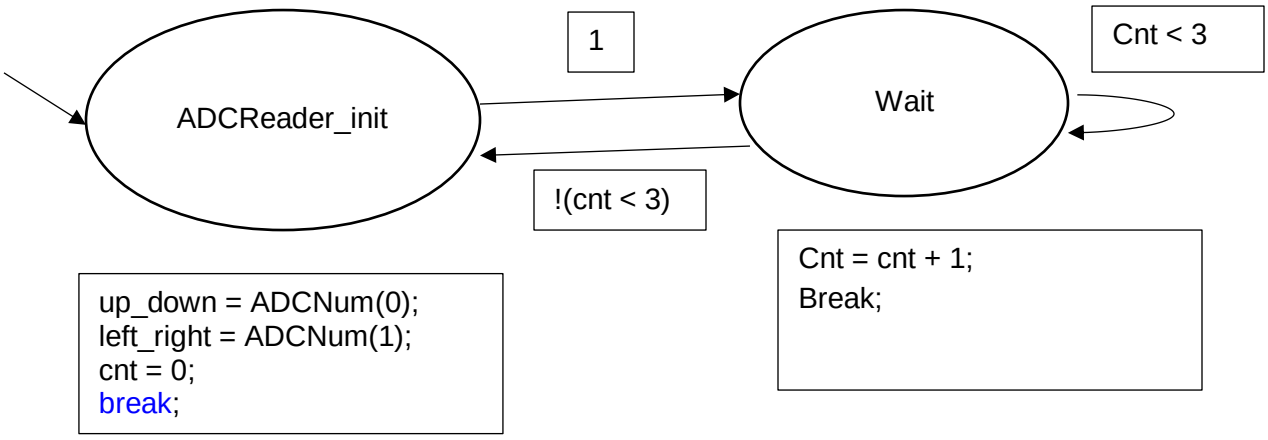

- ADC_init() and ADCNum() were acquired from: http://maxembedded.com/2011/06/the-adc-of-the-avr/
  I used these functions to convert my joystick’s analog signal to a usable digital signal. This is how I set the values of up_down and left_right in the ADC_Reader task.
- NOKIA_5110.H and NOKIA_5110.C were acquired from: https://github.com/LittleBuster/avr-nokia5110
  I used the functions from these files to write messages, countdown and score to my NOKIA 5110 Display.


# Appendix

Images of all of the SM’s and other relevant work:

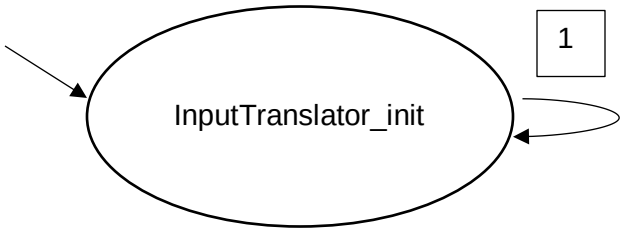
## *ADCReader SM:*

Reads in input from the user and stores the x-axis and y-axis values in left_right and up_down respectively.



## *InputTranslator SM:*

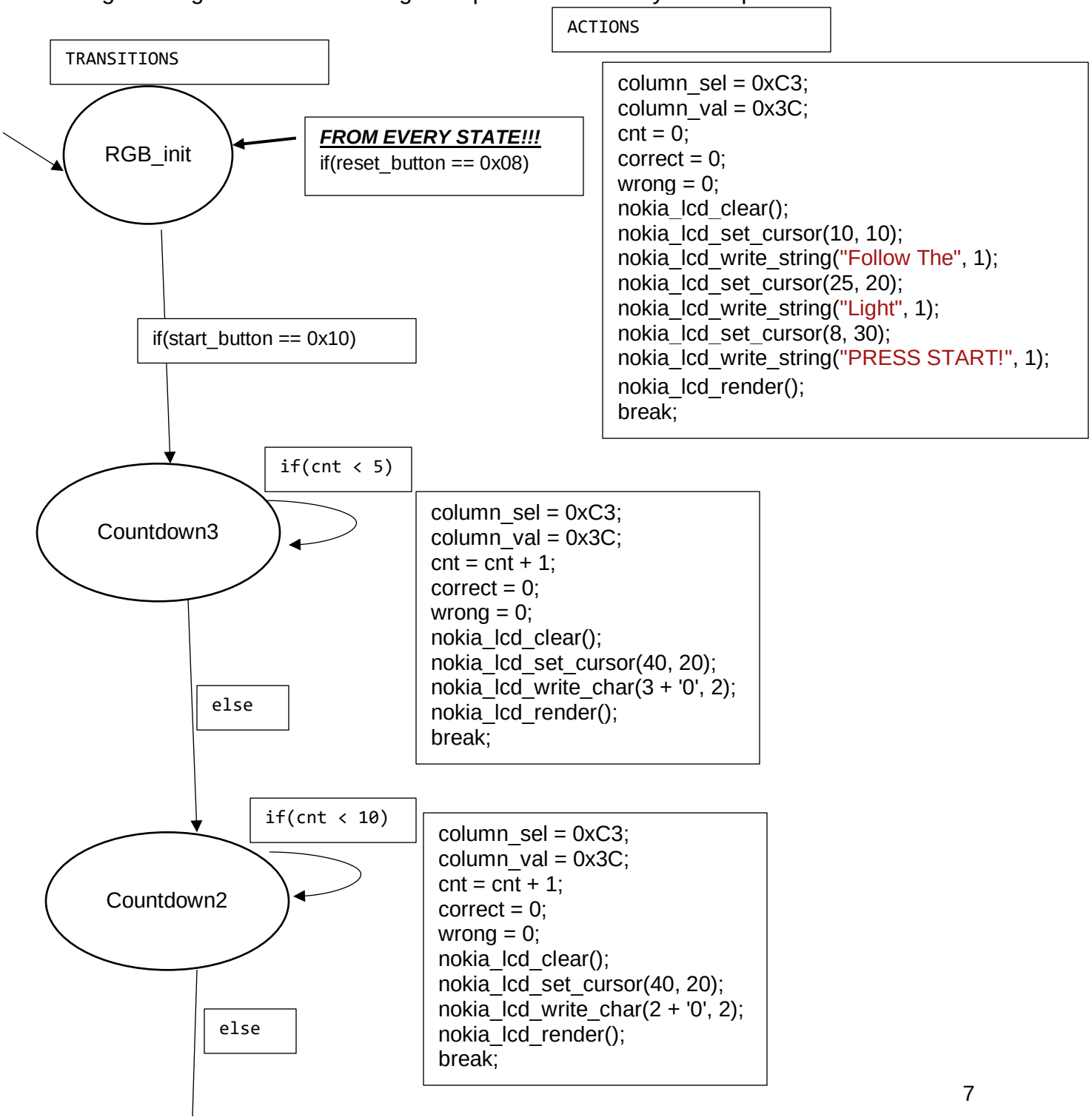
Reads in the up_down and left_right values and sets position accordingly. For example if the up_down value is greater than 165, this sets position to 1 indicating that the user has moved up.

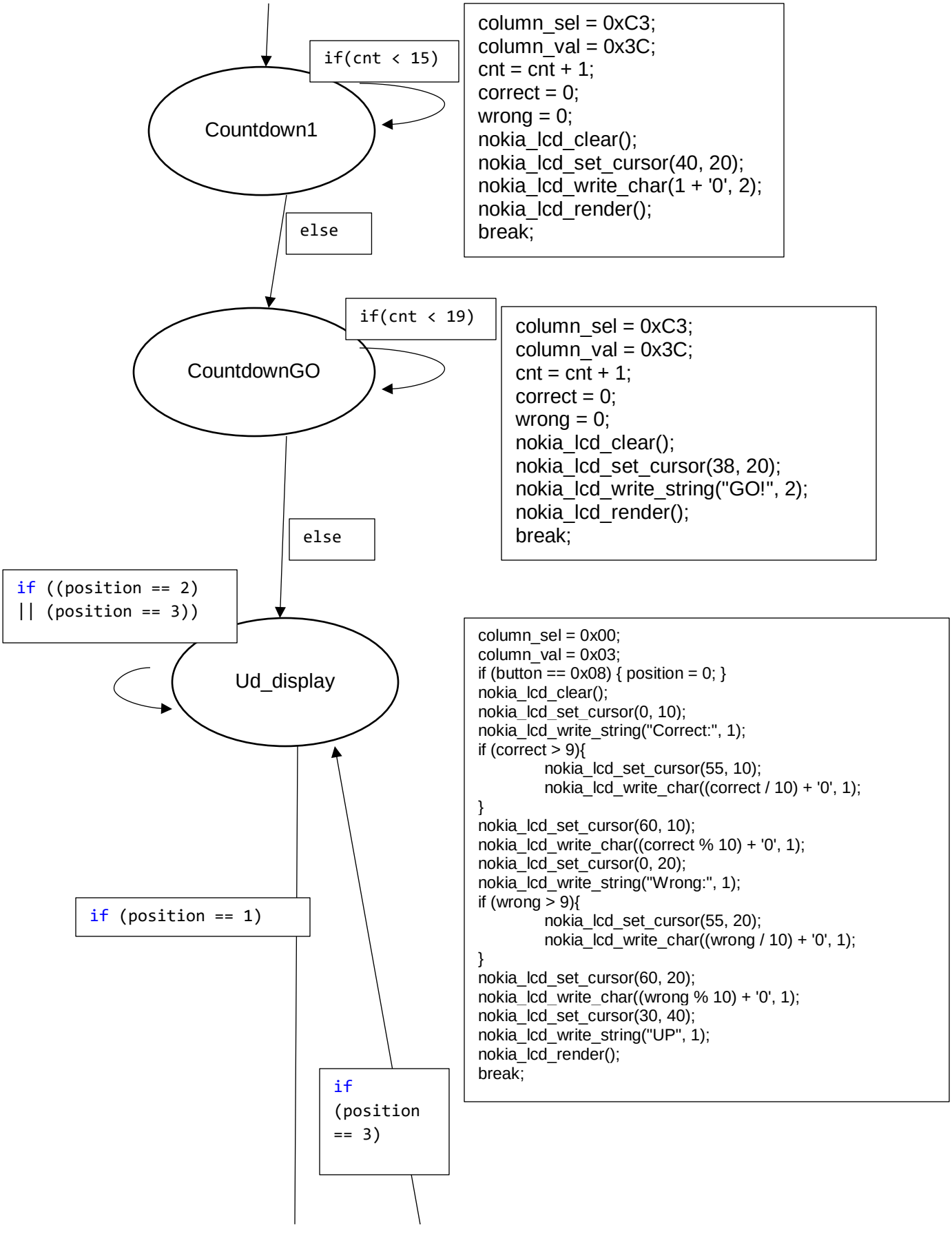


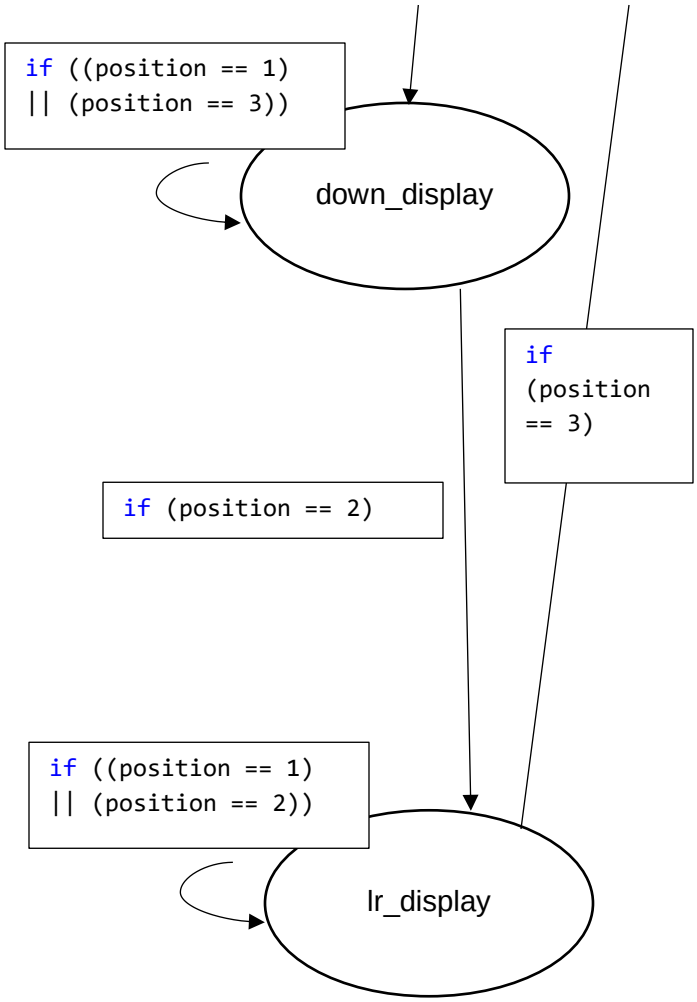

```
if (left_right > MAX) { position = 3; } //left
else if (up_down > MAX) { position = 1; } //up
else if (up_down < MIN) { position = 2; } //down
else if (left_right < MIN) { position = 4; } //right
else { position = 0; }
break;
```

## *RGB SM:*

Displays the instructions (up, down, left) as 1, 2, 3 and checks if the desired position matches the instruction and increments the number correct if correct or the number wrong if wrong. Correct and wrong are updated after every user input.

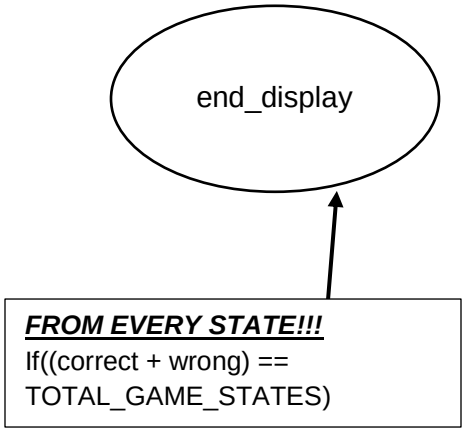



if(cnt < 15)
Countdown1
column_sel = 0xC3;
column_val = 0x3C;
cnt = cnt + 1;
correct = 0;
wrong = 0;
nokia_lcd_clear();
nokia_lcd_set_cursor(40, 20);
nokia_lcd_write_char(1 + '0', 2);
nokia_lcd_render();
break;
else
if(cnt < 19)
CountdownGO
column_sel = 0xC3;
column_val = 0x3C;
cnt = cnt + 1;
correct = 0;
wrong = 0;
nokia_lcd_clear();
nokia_lcd_set_cursor(38, 20);
nokia_lcd_write_string("GO!", 2);
nokia_lcd_render();
break;
else
if ((position == 2)
|| (position == 3))
Ud_display
column_sel = 0x00;
column_val = 0x03;
if (button == 0x08) { position = 0; }
nokia_lcd_clear();
nokia_lcd_set_cursor(0, 10);
nokia_lcd_write_string("Correct:", 1);
if (correct > 9){
nokia_lcd_set_cursor(55, 10);
nokia_lcd_write_char((correct / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 10);
nokia_lcd_write_char((correct % 10) + '0', 1);
nokia_lcd_set_cursor(0, 20);
nokia_lcd_write_string("Wrong:", 1);
if (wrong > 9){
nokia_lcd_set_cursor(55, 20);
nokia_lcd_write_char((wrong / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 20);
nokia_lcd_write_char((wrong % 10) + '0', 1);
nokia_lcd_set_cursor(30, 40);
nokia_lcd_write_string("UP", 1);
nokia_lcd_render();
break;
if (position == 1)
if
(position
== 3)




```
column_sel = 0x00;
column_val = 0xC0;
if (button == 0x08) { position = 0; }
nokia_lcd_clear();
nokia_lcd_set_cursor(0, 10);
nokia_lcd_write_string("Correct:", 1);
if (correct > 9){
            nokia_lcd_set_cursor(55, 10);
            nokia_lcd_write_char((correct / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 10);
nokia_lcd_write_char((correct % 10) + '0', 1);
nokia_lcd_set_cursor(0, 20);
nokia_lcd_write_string("Wrong:", 1);
if (wrong > 9){
            nokia_lcd_set_cursor(55, 20);
            nokia_lcd_write_char((wrong / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 20);
nokia_lcd_write_char((wrong % 10 + '0', 1);
nokia_lcd_set_cursor(30, 40);
nokia_lcd_write_string("UP", 1);
nokia_lcd_render();
break;
```

```
column_sel = 0x3F;
column_val = 0xFF;
if (button == 0x08) { position = 0; }
nokia_lcd_clear();
nokia_lcd_set_cursor(0, 10);
nokia_lcd_write_string("Correct:", 1);
if (correct > 9){
             nokia_lcd_set_cursor(55, 10);
             nokia_lcd_write_char((correct / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 10);
nokia_lcd_write_char((correct % 10) + '0', 1);
nokia_lcd_set_cursor(0, 20);
nokia_lcd_write_string("Wrong:", 1);
if (wrong > 9){
             nokia_lcd_set_cursor(55, 20);
             nokia_lcd_write_char((wrong / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 20);
nokia_lcd_write_char((wrong % 10) + '0', 1);
nokia_lcd_set_cursor(30, 40);
nokia_lcd_write_string("UP", 1);
nokia_lcd_render();
break;
```




```
column_sel = 0x00;
column_val = 0xFF;
if (button == 0x08) { position = 0; }
nokia_lcd_clear();
nokia_lcd_set_cursor(20, 0);
if (correct > wrong) //win if u get more correct than wrong
{
        nokia_lcd_write_string("YOU WIN!", 1);
        nokia_lcd_set_cursor(0, 10);
        nokia_lcd_write_string("More Correct", 1);
}
else
{
        nokia_lcd_write_string("YOU LOSE!", 1);
        nokia_lcd_set_cursor(0, 10);
        nokia_lcd_write_string("More Wrong", 1);
}
nokia_lcd_set_cursor(0, 30);
nokia_lcd_write_string("Correct:", 1);
if (correct > 9){
        nokia_lcd_set_cursor(55, 30);
        nokia_lcd_write_char((correct / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 30);
nokia_lcd_write_char((correct % 10) + '0', 1);
nokia_lcd_set_cursor(0, 40);
nokia_lcd_write_string("Wrong:", 1);
if (wrong > 9){
        nokia_lcd_set_cursor(55, 40);
        nokia_lcd_write_char((wrong / 10) + '0', 1);
}
nokia_lcd_set_cursor(60, 40);
nokia_lcd_write_char((wrong % 10) + '0', 1);
nokia_lcd_render();
break;
```

# Notes

I couldn't add a transition from every middle state to RGB_init (if user presses reset) so I added one big bolded transition and labeled it ***FROM EVERY STATE!!!***

Similarly, from every middle state to end_display (if total moves = 13).

I've also attached a pdf of my report just in case some text bubble goes out of place.