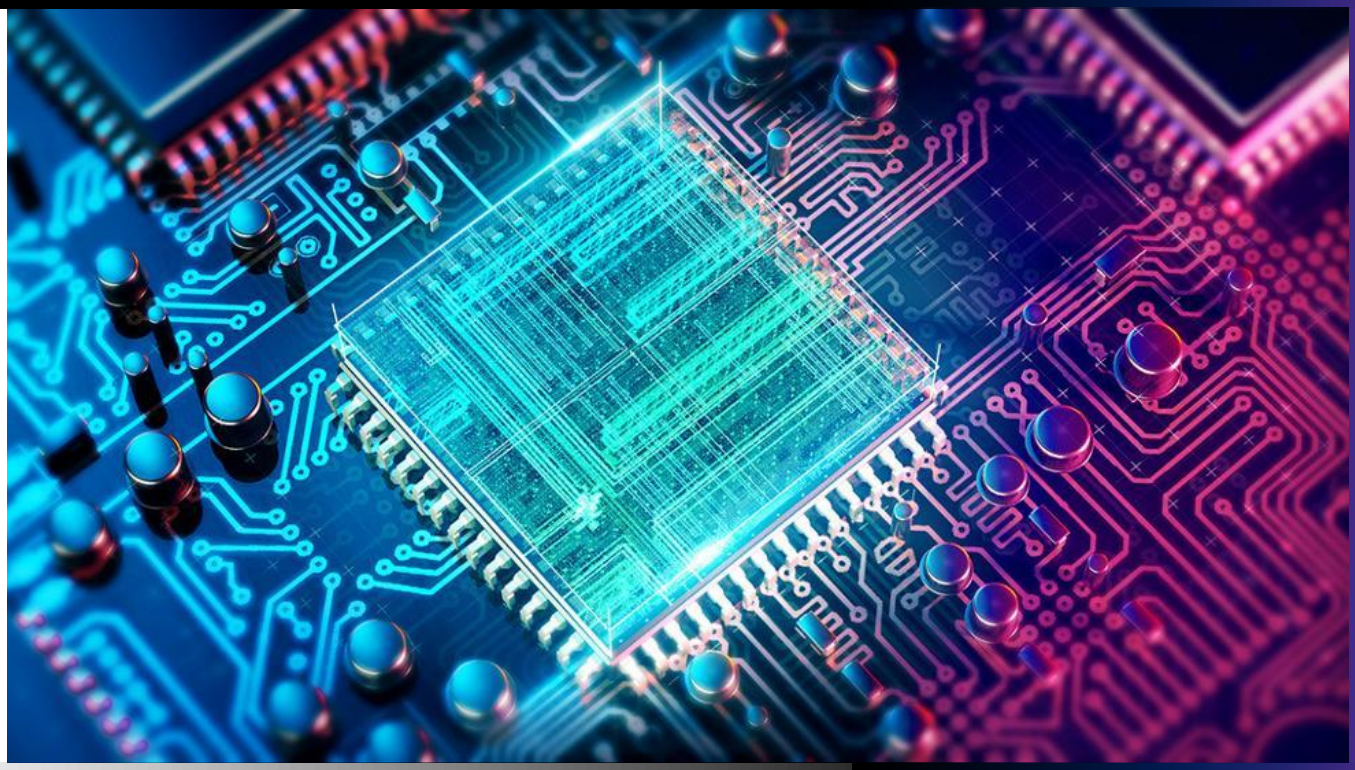
2019

BBM434 TERM PROJECT



SNAKE RUNNERS BOARD GAME

Group: Pepega

Görkem GENÇ 21327991

Cüneyt DUMAN 21327934

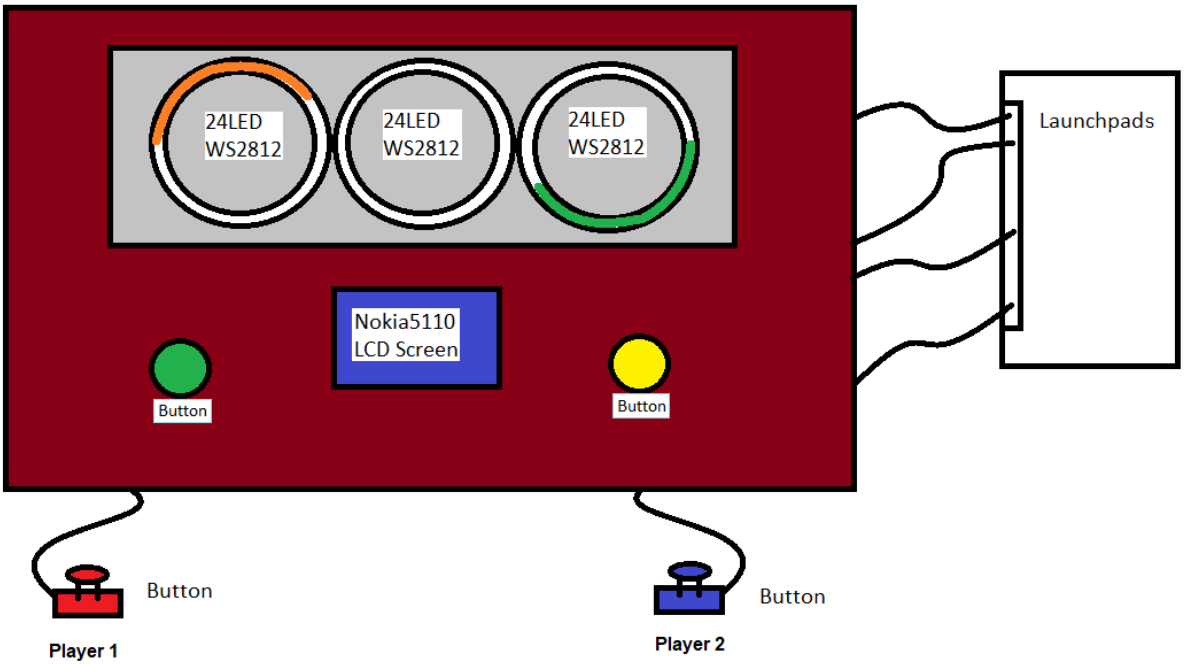
Hacettepe University

22.06.2019

1. Introduction

1.1. Overview

The summer is coming and people enjoy the summer with travelling, going on holiday, seeing different places, visiting friends. Playing games is obviously a part of socializing with people. So that, the snake runner boardgame is an interactive device for people to have fun together. Friend groups can play it, hold a tournament, or just chill and enjoy the game. It is a 1v1 boardgame that two players interact with the device through their joysticks, they can modify their snakes' colors and challenge their friends in a circle path.



1.2. Relevance as a Real-time Embedded Device

Since it is a boardgame, the users physically interact with the elements of the device. The joystick buttons are massively used by the users to race and the system gets huge amount of inputs at a time. So the input handling must be reliable to make the game fair. We can consider the snakes as outputs or feedbacks to the users that how close they are to winning. The snakes should be displayed synchronously with the current input state. We have used two different launchpads because there were not usable library of LED rings for Tiva-C launchpad. The LED rings are connected to arduino, and arduino is connected to Tiva-C launchpad as a master-slave relationship. Meanwhile getting the inputs from buttons to tiva-c launchpad, the data must be transferred to arduino to display the game. There is also a connection reliability between these launchpads.

1.3. Report Outline

The report shall discuss the real-time design considerations which are factored during the course of the project development. After which details regarding the various functionalities of the boardgame shall be highlighted. The report wraps up with some photographs of the system, also a video link will be attached to show the overall progress of the game. Lastly, the hardwares used in the system will be explained in detail.

2. Real-time Design Consideration

2.1. Real-time Tasks

There are several features of the system that needs to be handled syncronously in a very limited time.

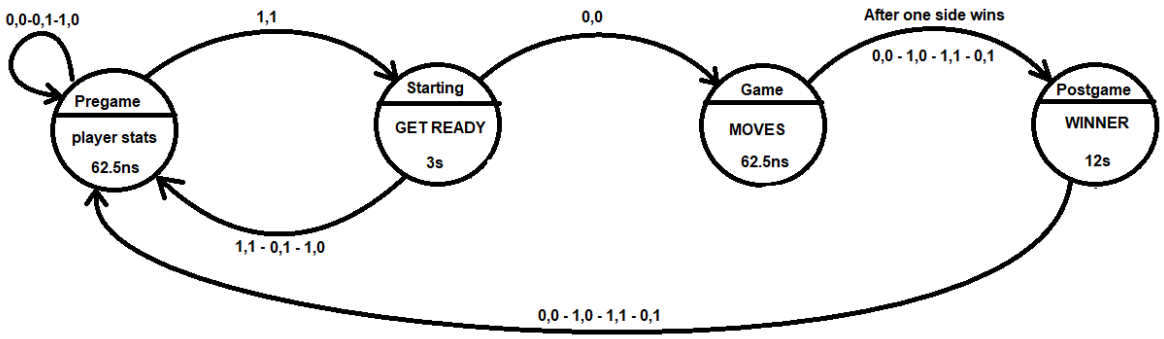
The first feature is changing snake colors. There are two buttons on the board that lets the players changing their own colors and modify their snake colors on their satisfactions. When the player presses the button, the system should give a quick response to change the snake color.

The second feature is joysticks. Players go as fast as they press the button, so that the system gets huge amount of inputs at a time. The system should handle all the button presses and do the necessary steps before a second interrupt by the joystick comes.

The third feature is LEDs, in other words snakes. Snakes' moves must be shown right after button presses and it should be concurrently working because when a player wins, they'll see it through the snakes whether it caught the other snake. So that the output must be shown simultaneously with the system's current state.

2.2. State-machine Design Approach

The game has four different phases in software level: pregame, starting, game, postgame. The phases are skipped in this order and after postgame phase, the leds do some celebration show and the game returns back to pregame phase. Pregame stage waits for users to be ready. If both players are ready, the game skips to starting phase. In this stage the game lets the players be unready if they want for three seconds. If both players remain ready, the game starts. The game continues until one player catch the other player. The game skips to postgame phase and blinks LEDs with the color of the winner player's snake. Finally the game starts again from the pregame phase.



2.3. Scheduling Hardware Interrupts

There are two different interrupts in the system. Systick and joystick buttons. There is also a connection between two launchpads based on interrupt handling. SysTick interrupts is used to set timers in Starting and Postgame phases. The CURRENT value of systick is also used to handle the debounces at joystick buttons. Joystick button interrupts are used to have players get ready/unready, and race each other. In game phase, each button interrupt creates a signal between master and slave, and that signal creates another interrupt at slave. The interrupt at slave moves the snakes on the LED rings.

3. Project Features

3.1. Snake Runners LED Ring

The LED rings are the main feature of the board game because players will race each other through those LEDs. It is the main display unit alongside with LCD screen. It provides players to choose their own color, see how fast they are moving and how close they are to opponent. It is on top of the board game placed under a blurry paper to prevent disturbing eyes and let the snakes move more fluid way.

3.2. LCD Screen

The purpose of LCD screen is to show the game phases to the players. Before the game starts, both players are shown UNREADY. If they hold the joystick button they become ready. They are free to unready by releasing the button in three seconds after both sides are ready. When both sides remain READY state for three seconds, LCD screen shows how many LEDs left between players during the game. After one side wins the game, LCD screen shows who has won for few seconds, and goes back to the initial state.

3.3. Snake Color Change Buttons

They are placed near the LCD screen and work before the game starts. It is a fancy feature for players to change their snakes' colors. Since the LEDs are RGB, we didn't want to waste it by using only one color and added this feature.

3.4. Joystick Buttons

Joystick buttons are the elements of the project that the players interact with the game. The buttons are used to get ready/unready, and spamming it throughout the game will increase the speed of players' snakes to catch the opponent.

4. Project Development

4.1. Software Level

There are two different codes in our project belong to master and slave units. LCD screen, joystick buttons and slave launchpad is connected to master; LED rings and colorchanging buttons are connected to slave.

4.1.1. Master

```
/*********************************************************************
************                  DEFINE                    *************
*********************************************************************/
//16Mhz Speed
#define P1_MOVE (*((volatile unsigned long *)0x40024040))   // PE4
#define P2_MOVE (*((volatile unsigned long *)0x40024080))   // PE5
#define ONE_SEC 16000000    //One Second
#define DEBOUNCE 1600000 //100 ms
#define CELEBRATION_TIME 12 //celebration time
```

PE4 and PE5 pins are connected from master to slave.

```
//Players
struct Player {
    unsigned int isReady, caught, winner;
};
struct Player p1 = {0,26,0};
struct Player p2 = {0,26,0};
```

There are two players. The initial state is: they are both UNREADY(0), they have 26 moves close to opponent, and they both haven't won yet.

```
/*****************************************************************
**************                INITIALIZE              **************
*****************************************************************/
//Initialize SysTick
void SysTick_Init(unsigned long delay) {
    NVIC_ST_CTRL_R = 0;             // disable SysTick during setup
    NVIC_ST_RELOAD_R = delay - 1;   // reload value
    NVIC_ST_CURRENT_R = 0;          // any write to current clears it
    NVIC_SYS_PRI3_R &= 0x00FFFFFF;  // Clear priority bits
    NVIC_SYS_PRI3_R |= 0x40000000;  // priority 2
    NVIC_ST_CTRL_R = 0x07;          // enable SysTick with processor clock
}
```

The SysTick timer is initialized to interrupt after every second, and its priority is set to 2.

```
//Initialize Port E
//Pin 0-1 Joystick buttons
//Pin 4-5 Slave interrupts
void PortE_Init(void) {
    //PE0 Player1 PE1 Player2
    SYSCTL_RCGC2_R |= 0x00000010;           // activate clock for Port E
    while((SYSCTL_PRGPIO_R & 0x10) == 0);   // ready?
    GPIO_PORTE_LOCK_R = 0x4C4F434B;         // unlock GPIO Port E
    GPIO_PORTE_CR_R |= 0x03;                // allow changes to PE1-0
    GPIO_PORTE_AMSEL_R &= ~0x03;            // disable analog functions
    GPIO_PORTE_PCTL_R &= ~0x000000FF;       // configure PE1-0 as GPIO
    GPIO_PORTE_DIR_R &= ~0x03;              // PE1-0: Input
    GPIO_PORTE_AFSEL_R &= ~0x03;            // disable alternate functions
    GPIO_PORTE_DEN_R |= 0x03;               // enable digital functions
    //PE4 Player1Snake PE5 Player2Snake XX11XXXX
    GPIO_PORTE_AMSEL_R &= ~0x30;            // disable analog functions
    GPIO_PORTE_PCTL_R &= ~0x00FF0000;       // configure PE4-5 as GPIO
    GPIO_PORTE_DIR_R |= 0x30;               // PE4-5: Output
    GPIO_PORTE_AFSEL_R &= ~0x30;            // disable alternate functions
    GPIO_PORTE_DEN_R |= 0x30;               // enable digital functions
    //Configure interrupts
    GPIO_PORTE_IS_R &= ~0x03;               // PE0-1 is edge-sensitive
    GPIO_PORTE_IBE_R &= ~0x03;              // PE0-1 is not both edges
    GPIO_PORTE_IEV_R |= 0x03;               // PE0-1 rising-edge event
    GPIO_PORTE_ICR_R |= 0x03;               // Clear flag
    GPIO_PORTE_IM_R |= 0x03;                // Arm interrupt on PE0-1
    NVIC_PRI1_R &= 0xFFFFFF00;              // Clear priority bits of Port E
    NVIC_PRI1_R |= 0x20;                    // Port E prority 1
    NVIC_EN0_R |= 0x10;                     // Enable NVIC interrupt
}
```

Joystick buttons are connected to PE0-1 and two outputs from master to slave is connected to PE4-5 pins on masters. Interrupt priority of joystick buttons are set to 1.

```
/*****************************************************************
*************           GAME FUNCTIONS              *************
*****************************************************************/
// Set beginning status for players
void Game_Init(void) {
    state = PREGAME;    //set state PREGAME
    p1.isReady = 0;
    p1.caught = 26;
    p1.winner = 0;
    p2.isReady = 0;
    p2.caught = 26;
    p2.winner = 0;
    lastClick[0] = NVIC_ST_RELOAD_R;
    lastClick[1] = NVIC_ST_RELOAD_R;
}
```

After the end of every game, Game_Init function is called to set the initial state for players and it also clears the last debounce time for each player.

```
/*****************************************************************
*************          INTERRUPT HANDLERS           *************
*****************************************************************/
//Systick Handler for the phases of the game and measuring reaction time
void SysTick_Handler(void) {
    if (state == PREGAME || state == GAME) {
    return;
  }
    else if (state == STARTING) {
        ++count;
        Update_Screen();
        if(flag==count) {
            state=GAME;
        }
    return;
    }
    else if (state == POSTGAME) {
        ++count;
        Update_Screen();
        if(flag==count) {
            Game_Init();
        }
    return;
  }
}
```

If current state is either pregame or game, systick handler is ignored. If the state is starting, SysTick Handler waits for any interrupt by the players for 3 seconds. If there isn't any interrupt during this time, the state is set to GAME. When the state is POSTGAME, SysTick Handler waits for 12 seconds to start the next game. In this duration the winner celebrates and its snake color shines on the LED rings.

```
//Get button press
void GPIOPortE_Handler(void) {
    unsigned int player = (GPIO_PORTE_RIS_R & 0x03) - 1; //P1 is 0, P2 is 1
    GPIO_PORTE_ICR_R |= (GPIO_PORTE_RIS_R & 0x03); //Acknowledge the flag of
    current player pin
    //debounce
    if(lastClick[player] - NVIC_ST_CURRENT_R < DEBOUNCE) { //Debounce
        return;
    }
    lastClick[player] = NVIC_ST_CURRENT_R; //not debounce get the current
    value for next check
```

Acknowledge which player pressed the button and check debounce.

```
if (state == PREGAME) {
    if(player == 0) { //p1
        p1.isReady = p1.isReady ^ 1; //ready or unready
    }
    if(player == 1) { //p2
        p2.isReady = p2.isReady ^ 1; //ready or unready
    }
    if((p1.isReady + p2.isReady)>1) { //both players ready
        state = STARTING;
        count = 0;
        flag = 3;
    }
return;
}
```

PREGAME state, buttons are used to be ready or unready. If both players are ready, switch to STARTING state and wait for 3 seconds.

```
else if (state == STARTING) {
    if(player == 0) { //p1
        p1.isReady = p1.isReady ^ 1; //ready or unready
        state=PREGAME;
    }
    if(player == 1) { //p2
        p2.isReady = p2.isReady ^ 1; //ready or unready
        state=PREGAME;
    }
return;
}
```

STARTING state, if one of the players switch back to UNREADY, switch to PREGAME state because STARTING time has been interrupted.

```
else if (state == GAME) {
    if(player == 0) { //p1
        P1_MOVE = 0x10;
        p1.caught++;
        p2.caught--;
        P1_MOVE = 0x00;
    }
    if(player == 1) { //p2
        P2_MOVE = 0x20;
        p2.caught++;
        p1.caught--;
        P2_MOVE = 0x00;
    }
    if(p1.caught == 0) { // p1 is caught
        state = POSTGAME;
        p2.winner = 1;
        count = 0;
        flag = CELEBRATION_TIME; //celebration
    return;
    }
    else if(p2.caught == 0) { //p2 is caught
        state = POSTGAME;
        p1.winner = 1;
        count = 0;
        flag = CELEBRATION_TIME; //celebration
    return;
    }
return;
}
```

GAME state, player button presses gets him close to opponent. If there are no LEDs left between the players, means that the player caught the opponent. P1_MOVE or P2_MOVE is triggered to send signal to slave to move the snakes. After a snake is caught, CELEBRATION_TIME duration is set to flag and state is changed to POSTGAME.

```
else if (state == POSTGAME) { //ignore presses while celebration is on
return;
}
```

Ignore the button presses during celebration time.

```
void Update_Screen(void) {
    char* result;
    if(state == PREGAME) {
        Nokia5110_Clear(); // Clear LCD screen
        Nokia5110_SetCursor(1, 1);
        Nokia5110_OutString(((char*) "P1"));
        Nokia5110_SetCursor(8, 1);
        Nokia5110_OutString(((char*) "P2"));

        Nokia5110_SetCursor(0, 3);
        if(p1.isReady == 0)
            Nokia5110_OutString(((char*) "UNRDY"));
        else if(p1.isReady == 1)
            Nokia5110_OutString(((char*) "READY"));

        Nokia5110_SetCursor(7, 3);
        if(p2.isReady == 0)
            Nokia5110_OutString(((char*) "UNRDY"));
        else if(p2.isReady == 1)
            Nokia5110_OutString(((char*) "READY"));
    return;
    }
```

Screen is continuously updated. PREGAME state shows the status of both players.

```
else if(state == STARTING) {
    Nokia5110_Clear(); // Clear LCD screen
    Nokia5110_SetCursor(1, 2);
    Nokia5110_OutString(((char*) "GET READY"));
    Nokia5110_SetCursor(5, 4);
return;
}
```

At STARTING state, players must remain ready for three seconds.

```
else if(state == GAME) {
    char* result;
    Nokia5110_Clear(); // Clear LCD screen
    Nokia5110_SetCursor(1, 1);
    Nokia5110_OutString(((char*) "P1"));
    Nokia5110_SetCursor(8, 1);
    Nokia5110_OutString(((char*) "P2"));

    Nokia5110_SetCursor(1, 3);
    snprintf(result, 3, "%d", p1.caught);
    Nokia5110_OutString(result); // Display P1 caught

    Nokia5110_SetCursor(8, 3);
    snprintf(result, 3, "%d", p2.caught);
    Nokia5110_OutString(result); // Display P1 caught
return;
}
```

At GAME state, show how many LED units left between players.

```
else if(state == POSTGAME) {
    Nokia5110_Clear(); // Clear LCD screen
    Nokia5110_SetCursor(1, 3);
    Nokia5110_OutString(((char*) "GAME OVER!"));

    if(p1.winner==1) {
        Nokia5110_SetCursor(3, 1);
        Nokia5110_OutString("P1 WINS"); // Display the winner
    }
    else if(p2.winner==1) {
        Nokia5110_SetCursor(3, 1);
        Nokia5110_OutString("P2 WINS"); // Display the winner
    }
return;
}
```

At POSTGAME, show which player has won. There is also a classic “GAME OVER!” text when the game is over.

```
/*****************************************************************
*************           MAIN FUNCTION              *************
*****************************************************************/
int main(void) {
    Nokia5110_Init();
    PortE_Init();
    SysTick_Init(ONE_SEC);
    EnableInterrupts();
    Game_Init();
    while (1) {
            Update_Screen();
            WaitForInterrupt();
    }
}
```

Finally, main function initializes the elements and updates screen as long as the game is on.

4.1.2. Slave

```
#include <Adafruit_NeoPixel.h>
#define NEOPIN  6  // Pin6 Neopixel out
#define COLORBUTTON_P1 4  // Pin4 P1 color change
#define COLORBUTTON_P2 5  // Pin5 P2 color change
#define NUM_LEDS 72
#define SNAKESPEED 30000 //speed of assistance
```

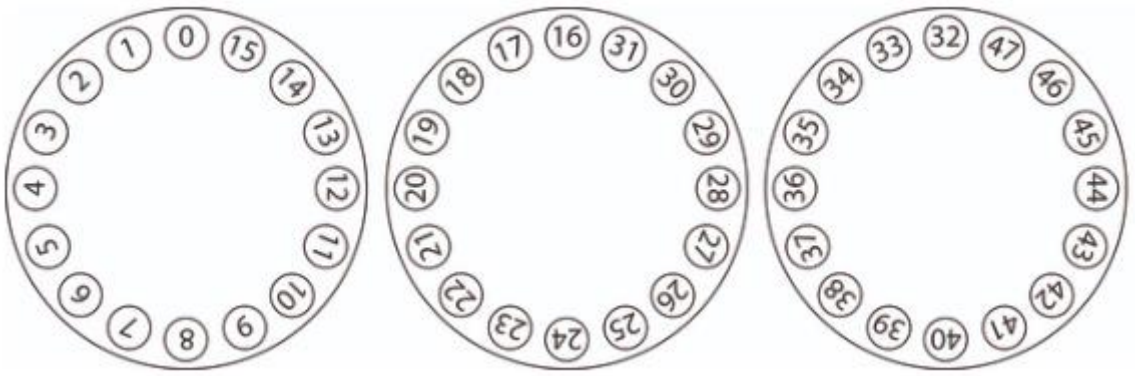
Adafruit_Neopixel.h library is created by the manufacturer for the use of LED rings. We have connected 3x24LED rings so we have total 72 LEDs. Output pin for Neopixel is connected to Pin6, Color buttons are connected to Pin4-5. Snakespeed makes the snakes move after some time. Since it moves the both snakes at the same time, it doesn’t affect the game winner.

```
// Parameter 1 = number of pixels in strip
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type:
//   NEO_GRB  Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB  Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, NEOPIN, NEO_GRB);
//Different colors  - G R B
uint32_t colorP1 = strip.Color(20, 255, 40); // Change RGB color value here
uint32_t colorP2 = strip.Color(255, 10, 50); // Change RGB color value here
int p1, p2;
int val1;
int val2;
int snkassist;
unsigned int p1Moves;
unsigned int p2Moves;
int state; //0 pregame, 1 game, 2 postgame
```

The variables used in the code. There are two players, they have colors, positions, moves.

```
// Array of pixels in order of animation - 72 in total:
int sine[] = {
7, 8, 9, 10, 11, 12, 13, 14, 15, 16,17, 18,
30, 29, 28, 27, 26, 25, 24, 47, 46, 45, 44, 43,
55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
67, 68, 69, 70, 71, 48, 49, 50, 51, 52, 53, 54,
42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31,
19, 20, 21, 22, 23, 0, 1, 2, 3, 4, 5, 6
};
```

Sine wave represents the path in LED rings.



Just like this, 24LED rings connected together can be reached through the position of the LEDs. The first ring LED positions cover 0-23, second ring 24-47, last ring 48-71.

```
//Set variables to initial values
void varInit() {
  state = 0; //pregame
  strip.setBrightness(40); // 40/255 brightness (about 15%)
  int i;
  for(i=0; i < 72; i++) {
    strip.setPixelColor(sine[(i) % 72], 0); // Clear path
    strip.show();
    delay(5);
  }
  //variables initial values
  p1=0; //position of p1
  p2=36; //position of p2
  val1 = 0;
  val2 = 0;
  p1Moves = 26;
  p2Moves = 26;
  snkassist = 0;
}
```

varInit() function is called at the beginning and end of every game. It sets the state to pregame, clears the path, sets the initial positions and other statuses of both players.

```
void setup() {
  strip.begin();
  strip.show();             // Initialize all pixels to 'off'
  attachInterrupt(0, moveP1, RISING); //pin2 P1 move
  attachInterrupt(1,moveP2,RISING); //pin3 P2 move
  pinMode(COLORBUTTON_P1, INPUT);
  pinMode(COLORBUTTON_P2, INPUT);
  varInit();
}
```

In setup, interrupts are set and colorchange buttons are defined.

```
void loop() {
  if(state==0) {
    val1 = digitalRead(COLORBUTTON_P1);
    val2 = digitalRead(COLORBUTTON_P2);
    if(val1 == 1) {
      setP1Color();
    }
    if(val2 == 1) {
      setP2Color();
    }
    int i=0;
    for(i=0; i < 10; i++) {
    strip.setPixelColor(sine[(p1-i+72) % 72], colorP1); // Draw 'head' pixel
    strip.setPixelColor(sine[(p2-i+72) % 72], colorP2); // Draw 'head' pixel
    strip.show();
    delay(10);
    }
  }
  else if(state==1) {
    snkassist++;
    if(snkassist==SNAKESPEED) {
      if(p1Moves<p2Moves) {
        moveP1();
        moveP2();
      }
      else {
        moveP2();
        moveP1();
      }
      snkassist=0;
    }
  }
}
```

In loop, if the state is pregame, allow players to change their colors and display the snakes with the new colors. If state is game, every SNAKESPEED's cycle, move the snakes by one unit to make snakes look more fluid. To prevent the game makes someone winner without any button presses, move the player who is losing first, and then move the other player.

```
void setP1Color() {
  colorP1 = strip.Color(random(0, 255), random(0, 255), random(0, 255));
}
void setP2Color() {
  colorP2 = strip.Color(random(0, 255), random(0, 255), random(0, 255));
}
```

Randomly generate fancy colors when the players want to change their colors.

```
void moveP1() {
  state=1;
  p1Moves++;
  p2Moves--;
  p1 = (p1+1) % 72;
  strip.setPixelColor(sine[p1 % 72], colorP1); // Draw 'head' pixel
  strip.setPixelColor(sine[(p1+62) % 72], 0); //clear tail
  strip.show();
  if(p2Moves == 0) {
    state=2;
    EndGame(1);
  }
}
```

When there is an interrupt by the master or loop(periodicly), moveP1 moves the player 1 snake one unit. Having an interrupt by the master means that the state is switched to the game. Move p1, set the new values between snakes, if p1 caught p2, set state to postgame and end the game with celebration lights.

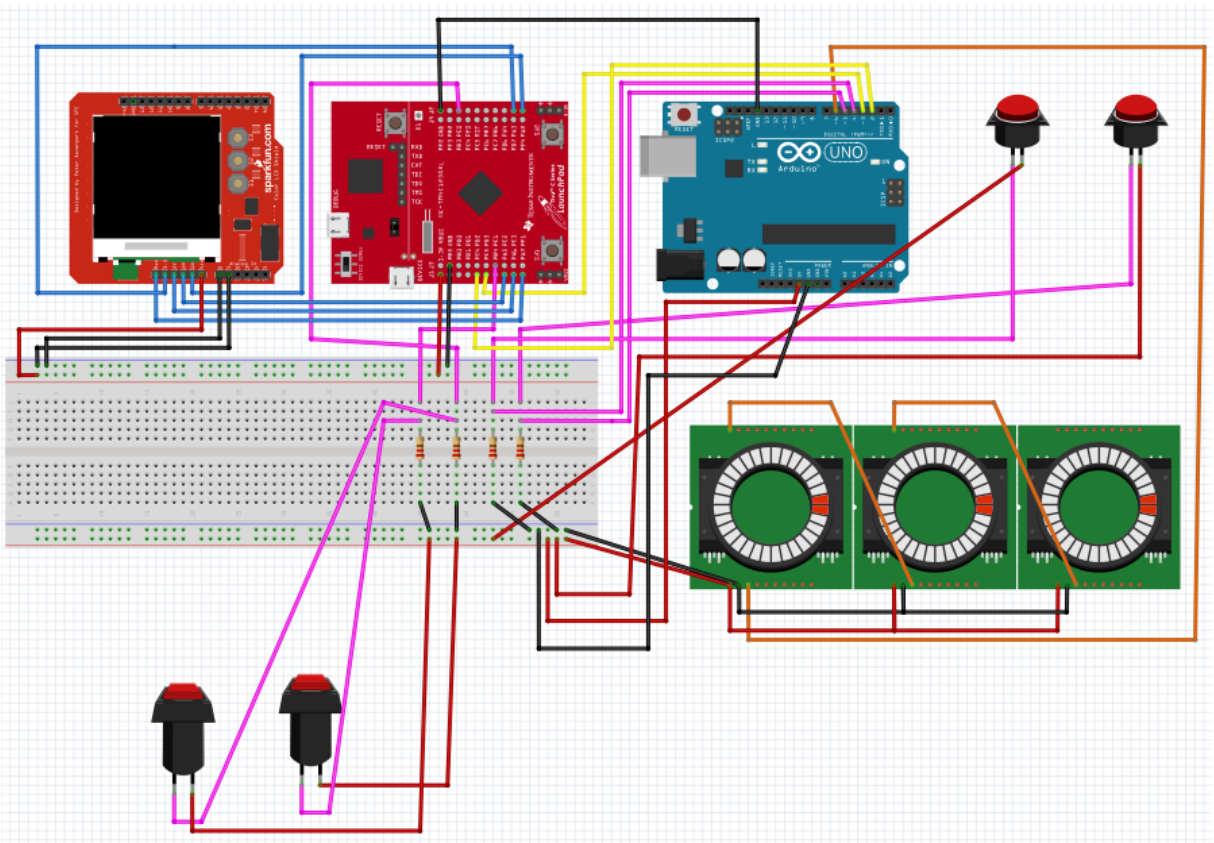
```
void moveP2() {
  state=1;
  p2Moves++;
  p1Moves--;
  p2 = (p2+1) % 72;
  strip.setPixelColor(sine[p2 % 72], colorP2); // Draw 'head' pixel
  strip.setPixelColor(sine[(p2+62) % 72], 0); //clear tail
  strip.show();
  if(p1Moves == 0) {
    state=2;
    EndGame(2);
  }
}
```

The same case above is valid for player2. The program does the same steps above for player2 when there is an interrupt by the master.

```
void EndGame(int player) {
  state=2;
  int i=0, j=0;
  if(player == 1) { // player1 won
    for(i=0; i < 72; i++) {
    for(i=50; i < 30; i--) {
    for(i=30; i < 50; i++) {
    for(j=0; j<30; j++) {
  }
  else {  // player2 won
    for(i=0; i < 72; i++) {
    for(i=50; i < 30; i--) {
    for(i=30; i < 50; i++) {
    for(j=0; j<30; j++) {
  }
  state=0;
  varInit();
}
```

When one of the players wins the game, the LEDs will shine in different styles with the color of winner snake. Those for loops generate some fancy styles and when they finish, the game state is set back to pregame and varInit is called.

4.2. Hardware Level



Wire Colors: Red--Vcc Black--GND Pink--Buttons Orange--LEDs Blue--LCD Yellow--Master/Slave

Buttons are connected positive logic. Joystick buttons connected to PE0-1, Changecolor buttons connected to Pin4-5.

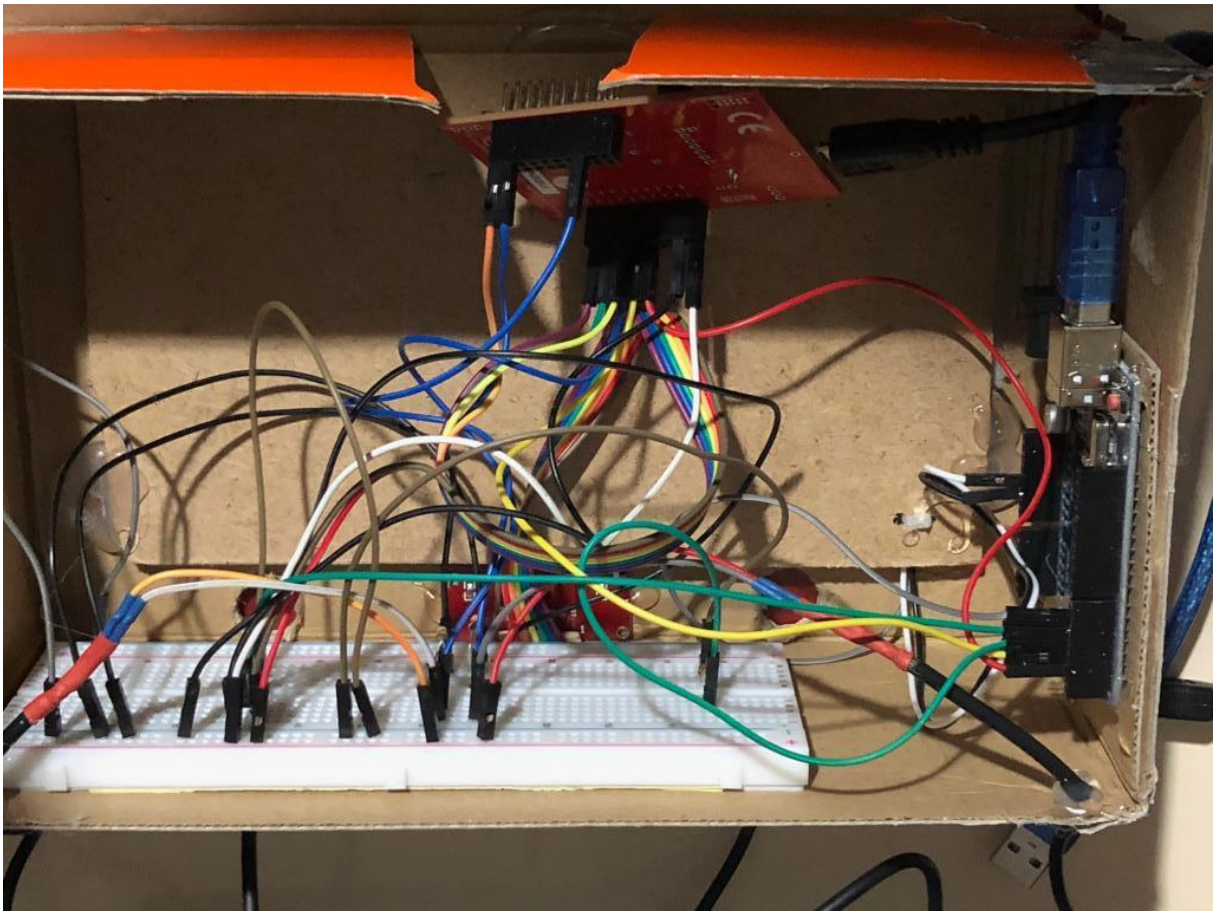
LCD screen connections:

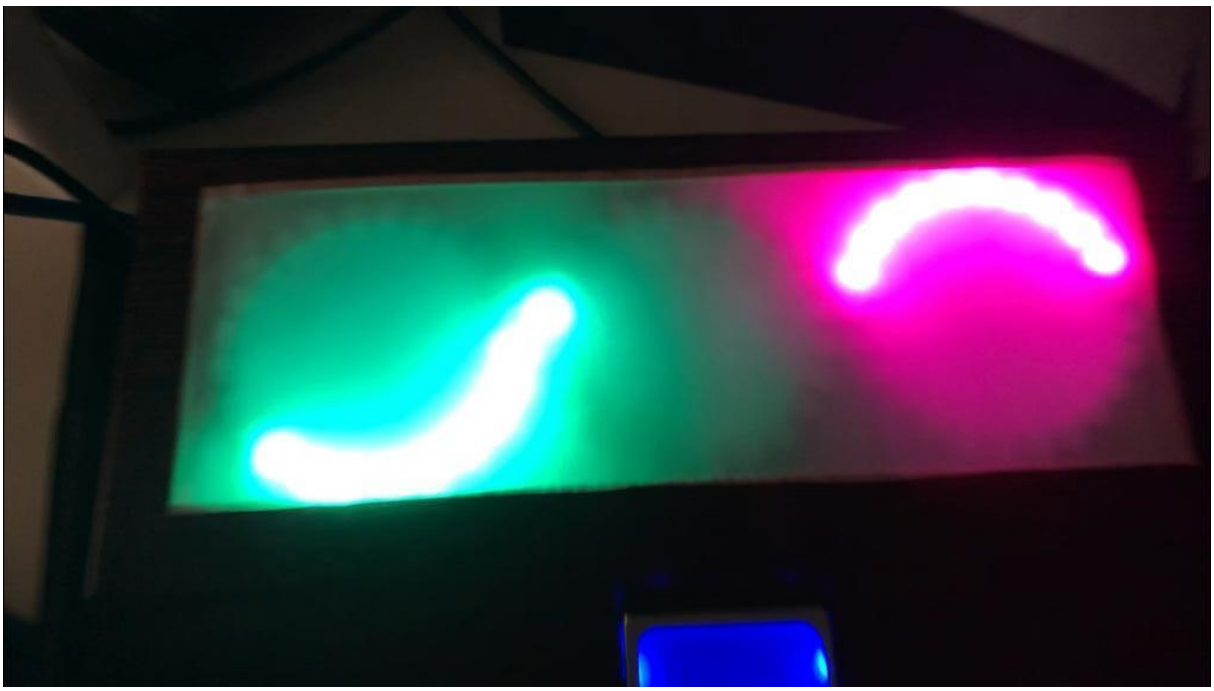
```
// Blue Nokia 5110
// ---------------
// Signal         (Nokia 5110) LaunchPad pin
// Reset          (RST, pin 1) connected to PA7
// SSI0Fss        (CE,  pin 2) connected to PA3
// Data/Command   (DC,  pin 3) connected to PA6
// SSI0Tx         (Din, pin 4) connected to PA5
// SSI0Clk        (Clk, pin 5) connected to PA2
// 3.3V           (Vcc, pin 6) power
// back light     (BL,  pin 7) ground
// Ground         (Gnd, pin 8) ground
```

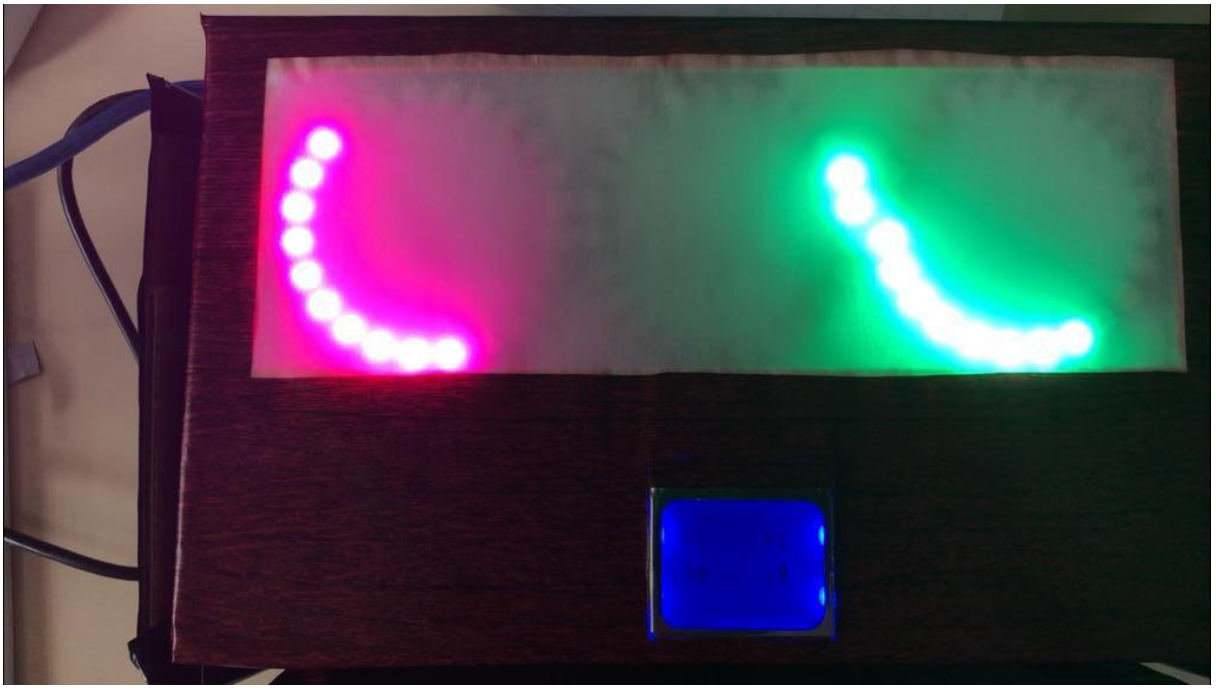
LED-ring0 IN pin connected to Pin6 of Arduino, LED-ring0 OUT pin connected to LED-ring1 IN pin, LED-ring1 OUT pin connected to LED-ring2 IN pin.

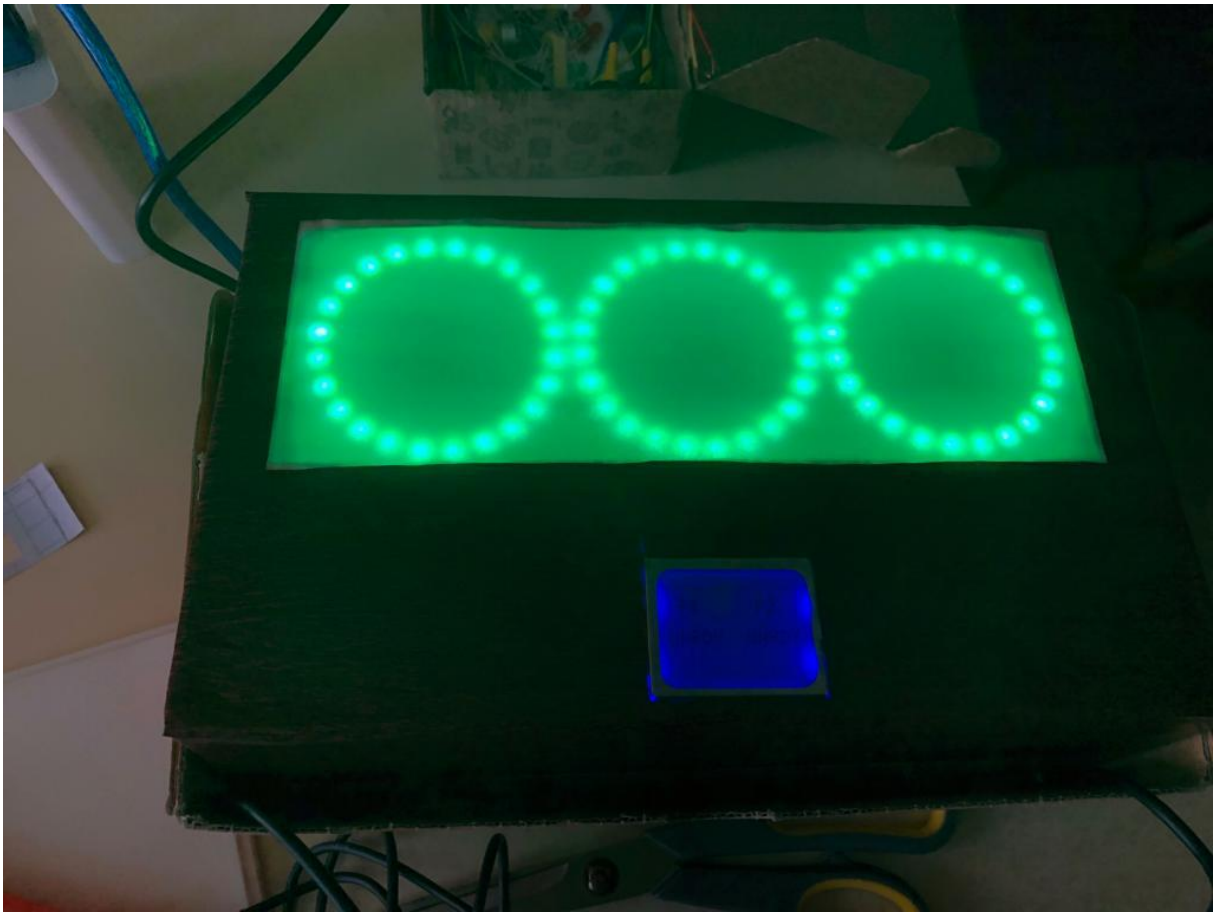
Master PE4-5 connected to Pin2-3 respectively.

5. Photo Documentation





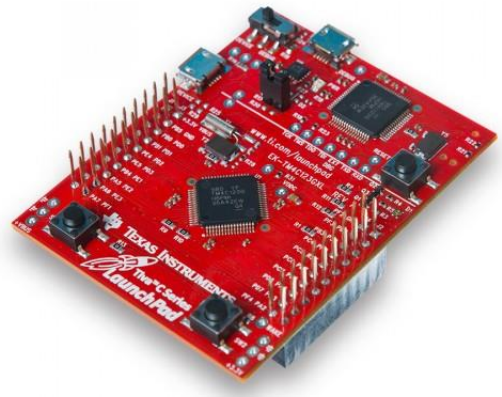




6. Hardware

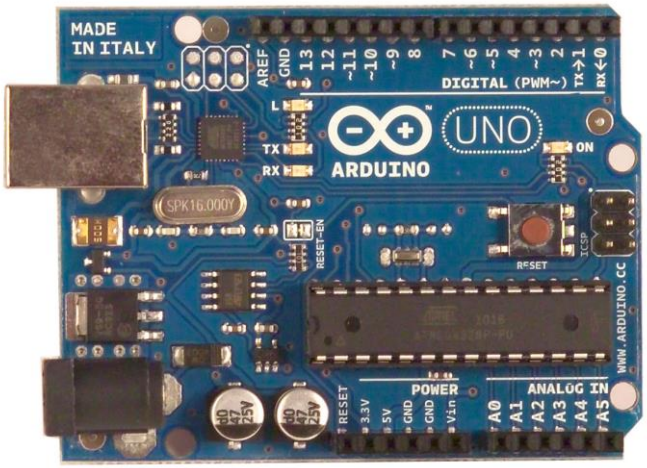
6.1. Tiva-C TM4C123GXL

https://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf?HQS=TI-null-null-alldatasheets-df-pf-SEP-wwe



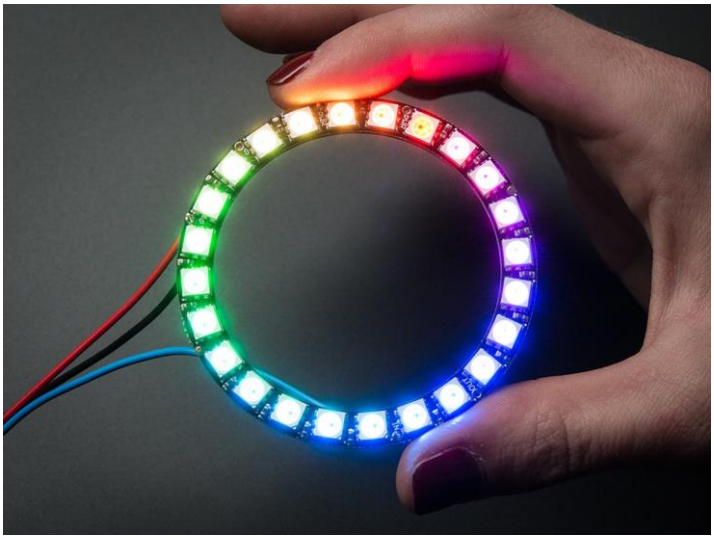
6.2. Arduino UNO

https://www.fecegypt.com/uploads/dataSheet/1522237550_arduino%20uno%20r3.pdf



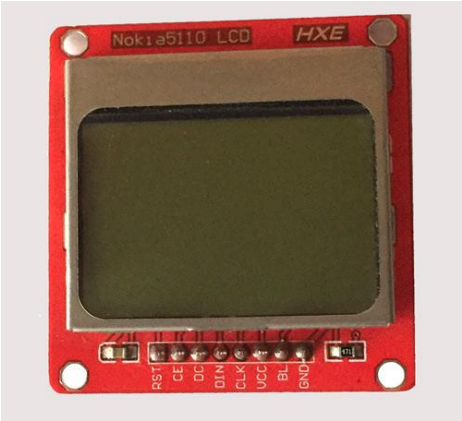
6.3. WS2812 Neopixel LED Circle

http://pdf1.alldatasheet.com/datasheet-pdf/view/1134522/WORLDSEMI/WS2812B-2020.html



6.4. Nokia5110 LCD Screen

https://components101.com/nokia-5110-lcd



6.5. PBS26 16mm Push Button

https://www.motorobit.com/urun/pbs26-16mm-push-buton-siyah