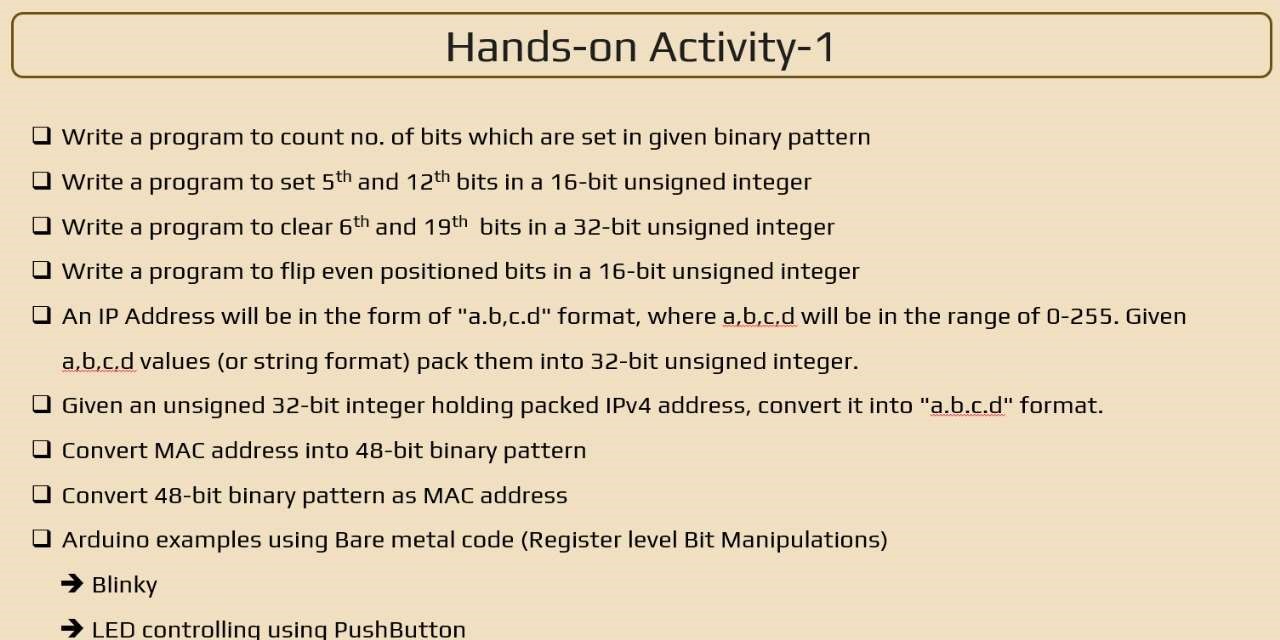


SUMMER INTERNSHIP Embedded C

# - Maruthi M - BU21EECE0100554 - mmuniraj@gitam.in

TASK 12



## Q1)

#include <stdio.h> int countSetBits(int n) { int count = 0; while (n) { count += n & 1; n >>= 1;

}

return count;

}

int main() { int num; prin ("Enter an integer: "); scanf("%d", &num); int setBits = countSetBits(num); prin ("Number of set bits in %d is %d\n", num, setBits);

return 0;

}

## Q2)

#include <stdio.h> int main()

{

unsigned short int value = 0; unsigned short int mask = (1 << 4) | (1 << 11); value |= mask; prin ("The value a er se ng the 5th and 12th bits is: %u\n", value);

return 0;

}

## Q3)

#include <stdio.h> unsigned int clearBits(unsigned int num) { unsigned int mask = ~((1 << 5) | (1 << 18)); return num & mask;

}

int main() { unsigned int num; prin ("Enter a 32-bit unsigned integer: "); scanf("%u", &num); unsigned int result = clearBits(num); prin ("Result a er clearing the 6th and 19th bits: %u\n", result);

return 0;

}

## Q4)

#include <stdio.h> unsigned short flipEvenBits(unsigned short num) { unsigned short mask = 0x5555; return num ^ mask;

}

int main() { unsigned short num; prin ("Enter a 16-bit unsigned integer: "); scanf("%hu", &num); unsigned short result = flipEvenBits(num); prin ("Result a er flipping the even-posi oned bits: %hu\n", result);

return 0;

}

## Q5)

#include <stdio.h> unsigned int packIP(unsigned char a, unsigned char b, unsigned char c, unsigned char d) { return (a << 24) | (b << 16) | (c << 8) | d;

}

int main() { unsigned char a = 192; unsigned char b = 168; unsigned char c = 1; unsigned char d = 100; unsigned int packedIP = packIP(a, b, c, d); prin ("Packed IP address: 0x%X\n", packedIP);

return 0;

}

## Q6)

#include <stdio.h> int main() { unsigned int packed\_ip = 0xC0A80164; unsigned char a = (packed\_ip >> 24) & 0xFF; unsigned char b = (packed\_ip >> 16) & 0xFF; unsigned char c = (packed\_ip >> 8) & 0xFF; unsigned char d = packed\_ip & 0xFF; prin ("The unpacked IP address is: %u.%u.%u.%u\n", a, b, c, d); return 0;

}

## Q7)

#include <stdio.h> #include <stdlib.h> unsigned long long convertMACAddress(const char \*mac) {

unsigned int bytes[6];

if (sscanf(mac, "%x:%x:%x:%x:%x:%x", &bytes[0], &bytes[1], &bytes[2], &bytes[3], &bytes[4],

&bytes[5]) != 6) { fprin (stderr, "Invalid MAC address format.\n"); exit(EXIT\_FAILURE);

}

unsigned long long macBinary = 0;

for (int i = 0; i < 6; ++i) {

macBinary = (macBinary << 8) | (bytes[i] & 0xFF);

}

return macBinary;

}

int main() { char macString[18]; prin ("Enter MAC address in the format XX:XX:XX:XX:XX:XX: "); if (scanf("%17s", macString) != 1) { fprin (stderr, "Failed to read MAC address.\n"); return EXIT\_FAILURE;

}

unsigned long long macBinary = convertMACAddress(macString); prin ("MAC address in 48-bit binary pa ern: %012llx\n", macBinary);

return 0;

}

## Q8)

#include <stdio.h> #include <stdlib.h> void binaryToMac(const char\* binary) {

unsigned int bytes[6] = {0}; for (int i = 0; i < 48; ++i) { bytes[i / 8] = (bytes[i / 8] << 1) | (binary[i] - '0'); }

prin ("MAC Address: %02X:%02X:%02X:%02X:%02X:%02X\n",

bytes[0], bytes[1], bytes[2], bytes[3], bytes[4], bytes[5]);

}

int main() {

const char\* binary\_pa ern = "101010101011101111001100110111011110111111111111"; binaryToMac(binary\_pa ern);

return 0;

}

# Task 14

## 1)bare metal blinky using arduino1

#define F\_CPU 16000000UL

#include <avr/io.h> #include <u l/delay.h> int main(void)

{

// Set pin 7 (PD7) as an output DDRD |= (1 << PD7);

while (1)

{

PORTD |= (1 << PD7);

\_delay\_ms(1000);

PORTD &= ~(1 << PD7);

\_delay\_ms(1000);

}

return 0;

}

## 2)bare metal push bu on1

#define F\_CPU 16000000UL

#include <avr/io.h> #include <u l/delay.h> const uint8\_t bu onPin = PD2; const uint8\_t ledPin = PB5;

uint8\_t bu onState = 0; void setup() {

DDRD &= ~(1 << bu onPin);

PORTD |= (1 << bu onPin);

DDRB |= (1 << ledPin);

}

int main(void) {

setup(); while (1) { bu onState = PIND & (1 << bu onPin);

if (bu onState) {

PORTB |= (1 << ledPin);

} else {

PORTB &= ~(1 << ledPin);

}

\_delay\_ms(10);

}

return 0;

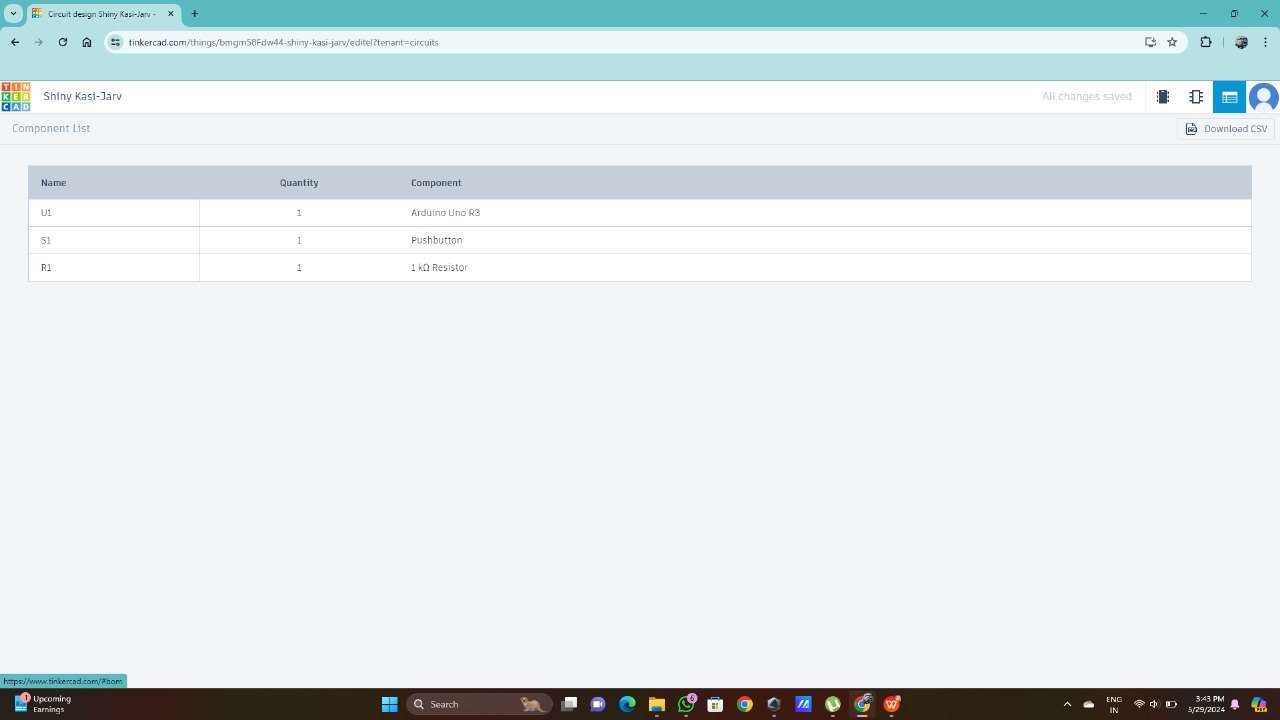
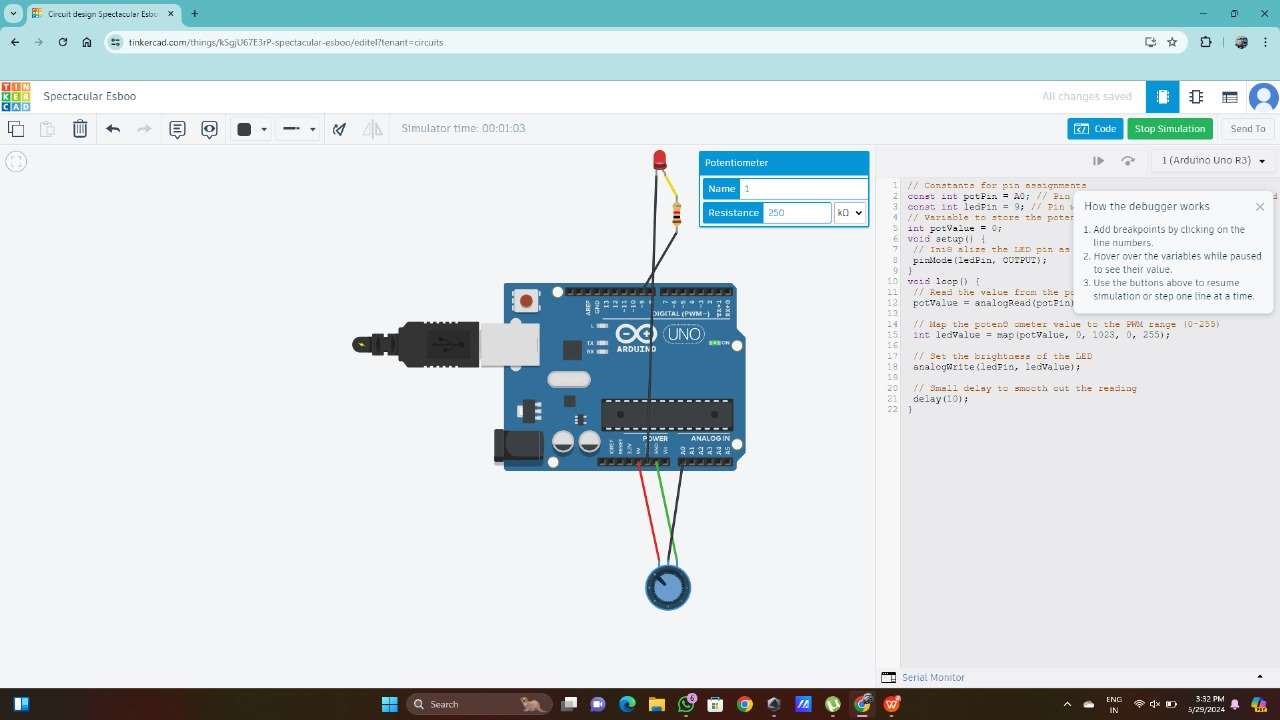
}

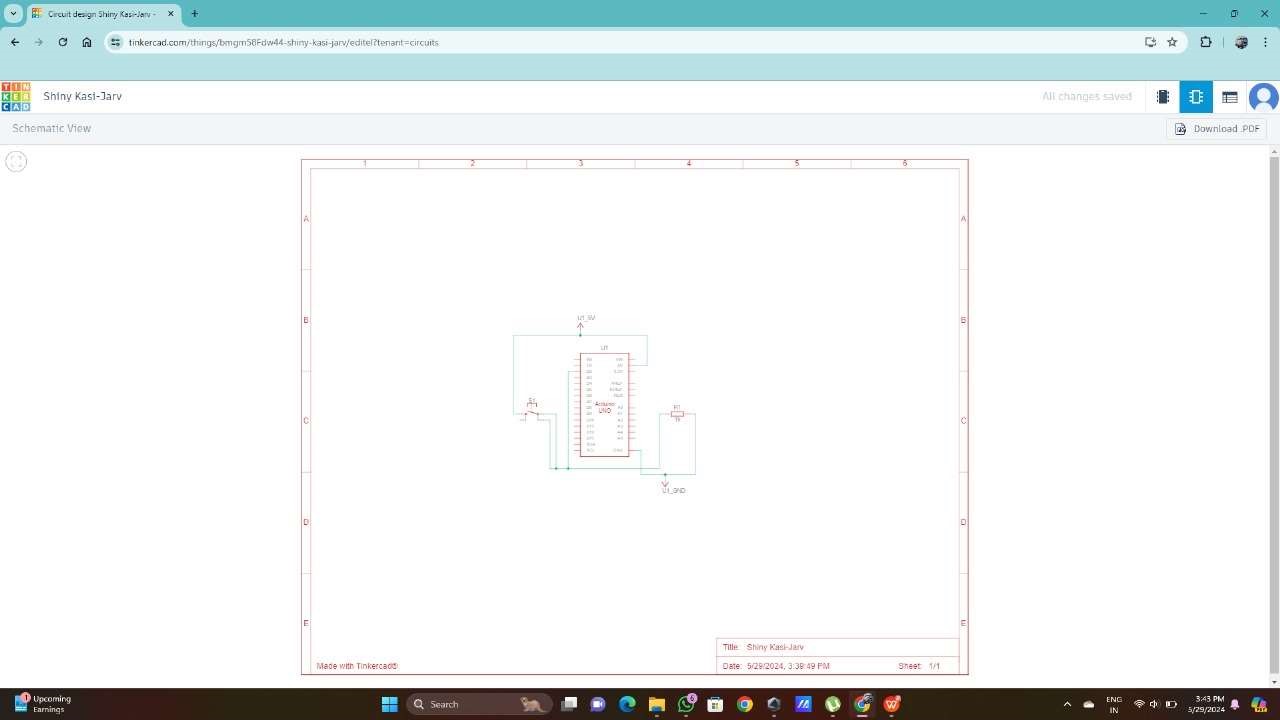
# Task 15

## Analog Read (Poten ometer)

// Constants for pin assignments

const int potPin = A0; // Pin where the poten ometer is connected





const int ledPin = 9; // Pin where the LED is connected

// Variable to store the poten ometer value int potValue = 0;

void setup() {

// Ini alize the LED pin as an output pinMode(ledPin, OUTPUT);

}

void loop() {

// Read the value from the poten ometer potValue = analogRead(potPin);

// Map the poten ometer value to the PWM range (0-255) int ledValue = map(potValue, 0, 1023, 0, 255);

// Set the brightness of the LED analogWrite(ledPin, ledValue);

// Small delay to smooth out the reading delay(10);

}

## Analout Output(fading)

const int ledPin = 9; // Pin where the LED is connected

void setup() {

// Ini alize the LED pin as an output pinMode(ledPin, OUTPUT);

}

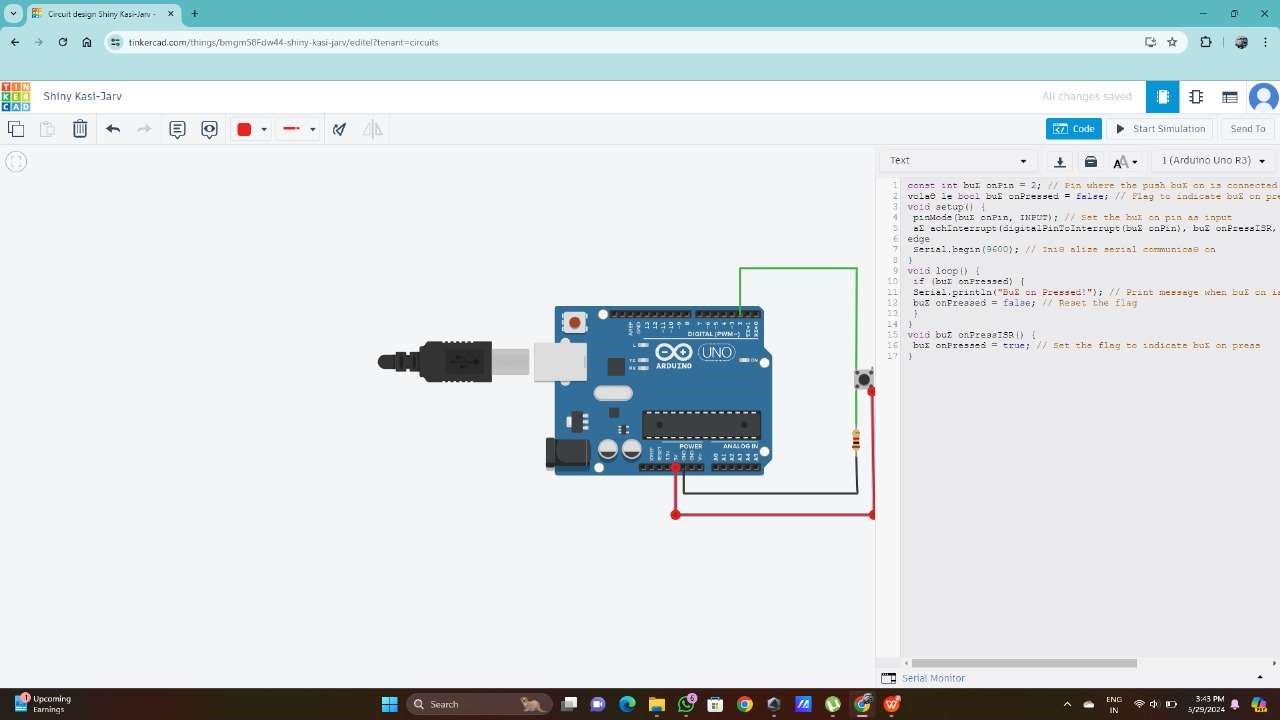
void loop() { // Fade in from 0 to 100^6 for (int brightness = 0; brightness <= 100^6; brightness++) { analogWrite(ledPin, brightness); // Set the brightness delay(10); // Wait for 10 milliseconds

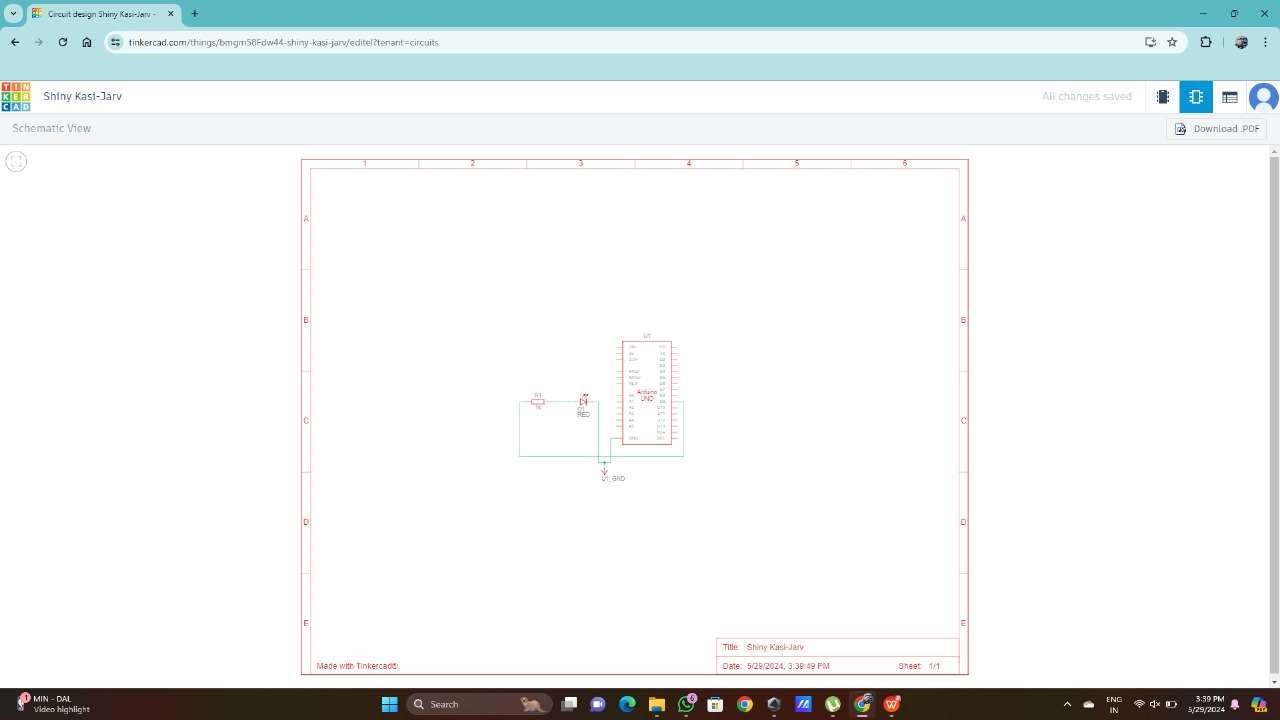
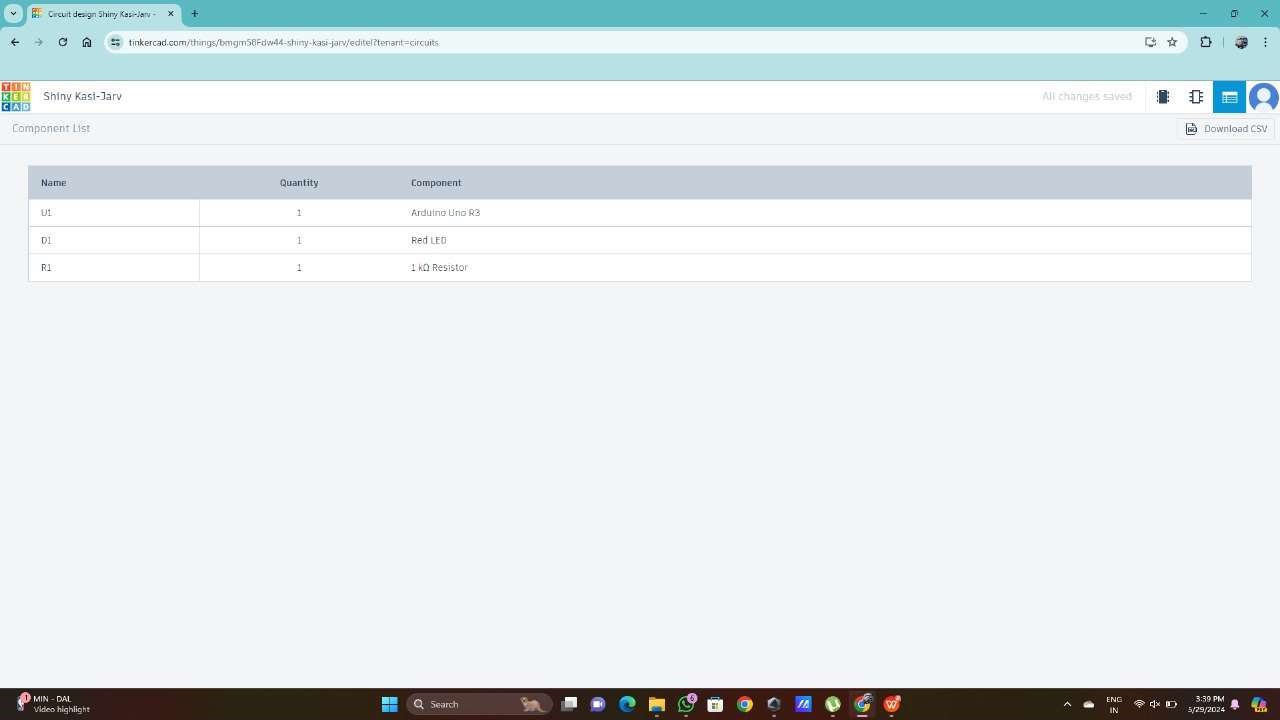
}

// Fade out from 100^6 to 0 for (int brightness = 100^6; brightness >= 0; brightness--) { analogWrite(ledPin, brightness); // Set the brightness delay(10); // Wait for 10 milliseconds

}

}





## Digital Input using Interrupt

const int bu onPin = 2; // Pin where the push bu on is connected vola le bool bu onPressed = false; // Flag to indicate bu on press void setup() { pinMode(bu onPin, INPUT); // Set the bu on pin as input

a achInterrupt(digitalPinToInterrupt(bu onPin), bu onPressISR, RISING); // A ach interrupt on rising edge

Serial.begin(9600); // Ini alize serial communica on

}

void loop() { if (bu onPressed) {

Serial.println("Bu on Pressed!"); // Print message when bu on is pressed

bu onPressed = false; // Reset the flag

}

}

void bu onPressISR() { bu onPressed = true; // Set the flag to indicate bu on press

}

