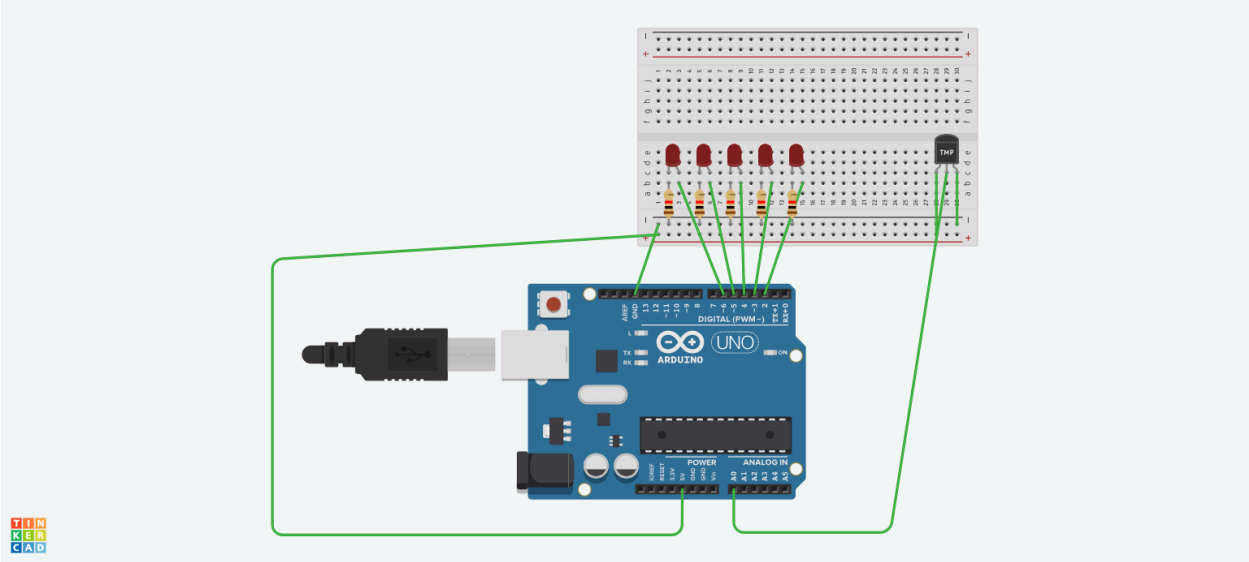
Exercise 1



Link to tinker

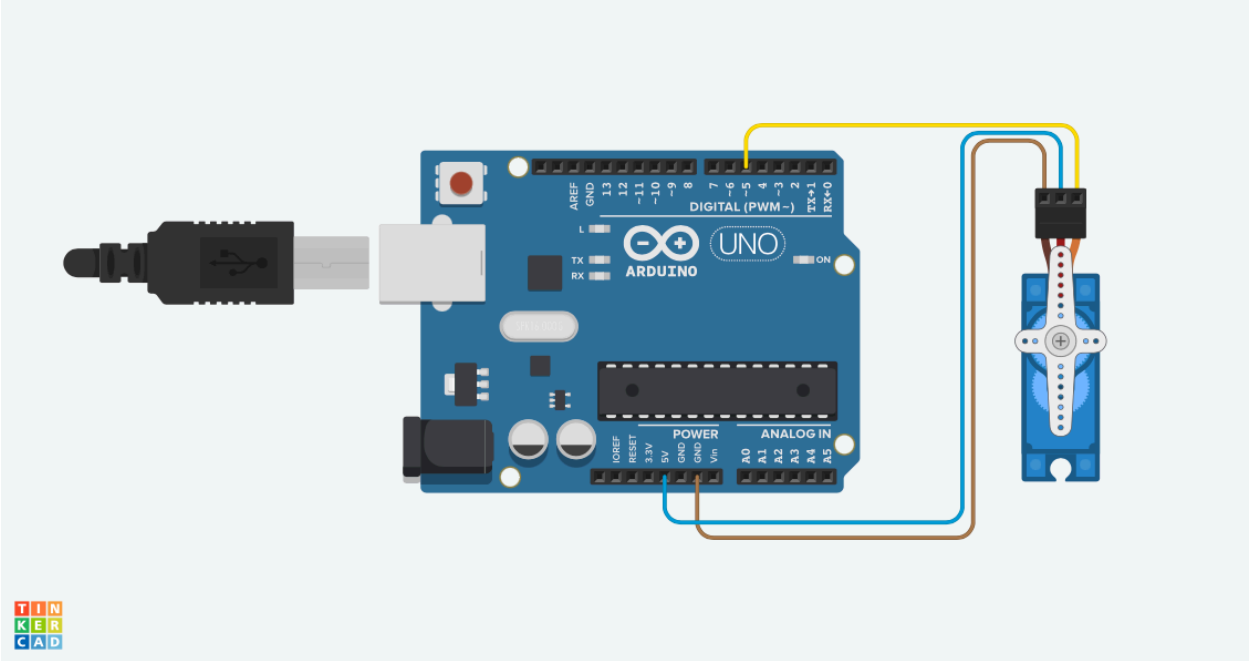
```
// (C) Jitish Rajankumar Padhya, Raghav Tengse, Utkarsh Singh, group:
23 (2024)
// Work package 4
// Exercise 1:
// Submission code: 2351823

// several aspects inspired from WP3-3.1
// Define pins for all LEDs
#define LED1 2
#define LED2 3
#define LED3 4
#define LED4 5
#define LED5 6

// Define temperature sensor analog pin
#define TEMP_SENSOR A0

// Define temperature ranges in degrees celsius
#define TEMP_RANGE0 0
#define TEMP_RANGE1 10
#define TEMP_RANGE2 20
```

```
#define TEMP_RANGE3 30
#define TEMP_RANGE4 40

void setup() { // Runs once everytime it is started or restarted

  // Configures the pins connected to LEDs as output pins
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);

  Serial.begin(9600); // initialise serial communication (for testing)

  // Timer interrupter
  //
https://www.tinkercad.com/things/eS19mAQ9aYu-arduino-timer-interrupt
  cli(); // disable global interrupts (clear interrupt flag)

  // reset values
  TCCR1A = 0; //timer control register A for Timer1
  TCCR1B = 0; //timer control register B for Timer1
  TCNT1 = 0; // counter value, set to zero to begin counting from 0
till interrupt value below

  OCR1A = 15624; // Output compare: when this value is reached (1 hz),
interrupt ((16*10^6) / (1*1024) - 1 (must be <65536))
  TCCR1B |= (1 << WGM12); // configures CTC: resets timer to 0 when
OCR1A value is reached
  TCCR1B |= (1 << CS12) | (1 << CS10); // Prescaler set to 1024
(intervals between interrupts
  TIMSK1 |= (1 << OCIE1A); // interrupt service routine (ISR) occurs
  sei(); //re-enable global interrupts (set interrupt flag)
}

// method is called when counter matches OCR1A value
ISR(TIMER1_COMPA_vect) { // Timer1 ISR is used (1 hz)
  float temp = readTemperature(); // get temperature in degrees
celsius
```

```
  // First turn off all LEDs
  for (int i = LED1; i <= LED5; i++) {
    digitalWrite(i, LOW);
  }

  // If loops to turn on the right set of LEDs depending on the range
of temperature
  if (temp < TEMP_RANGE0){ // temperature less than 0
      // No LEDs
  }
  else if (temp < TEMP_RANGE1) { // temperature less than 10
    digitalWrite(LED5, HIGH);    // 1 LEDs
  }
  else if (temp < TEMP_RANGE2) { // temperature less than 20
    digitalWrite(LED5, HIGH);     // 2 LEDs from the left are on
    digitalWrite(LED4, HIGH);
  }
  else if (temp < TEMP_RANGE3) { // temperature less than 30
    digitalWrite(LED5, HIGH);     // 3 LEDs from the left are on
    digitalWrite(LED4, HIGH);
    digitalWrite(LED3, HIGH);
  }
  else if (temp < TEMP_RANGE4) { // temperature less than 40
    digitalWrite(LED5, HIGH);     // 4 LEDs from the left are on
    digitalWrite(LED4, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED2, HIGH);
  }else{   // // temperature greater than or equal to 40
    digitalWrite(LED5, HIGH);    // 5 LEDs from the left are on
    digitalWrite(LED4, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED1, HIGH);
  }

  // For testing purposes print temp in monitor:
  Serial.print("Temperature: ");
  Serial.print(temp); // print the float temperature
  Serial.println(" C");
}
```

```
float readTemperature() { // Function to read temperature from sensor
  float tempValue = analogRead(TEMP_SENSOR); //analog voltage from the
sensor
  float voltage = tempValue * (5.0 / 1023.0); // Convert sensor value
to voltage between 0-5V
  float temperature = (voltage - 0.5) * 100; // convert volts to
degrees celsius
  return (float)temperature; // return temputrature of type int
}

void loop() {
  // loop occurs in the in the interrupt method when the counter is
matched
}
```

Exercise 2



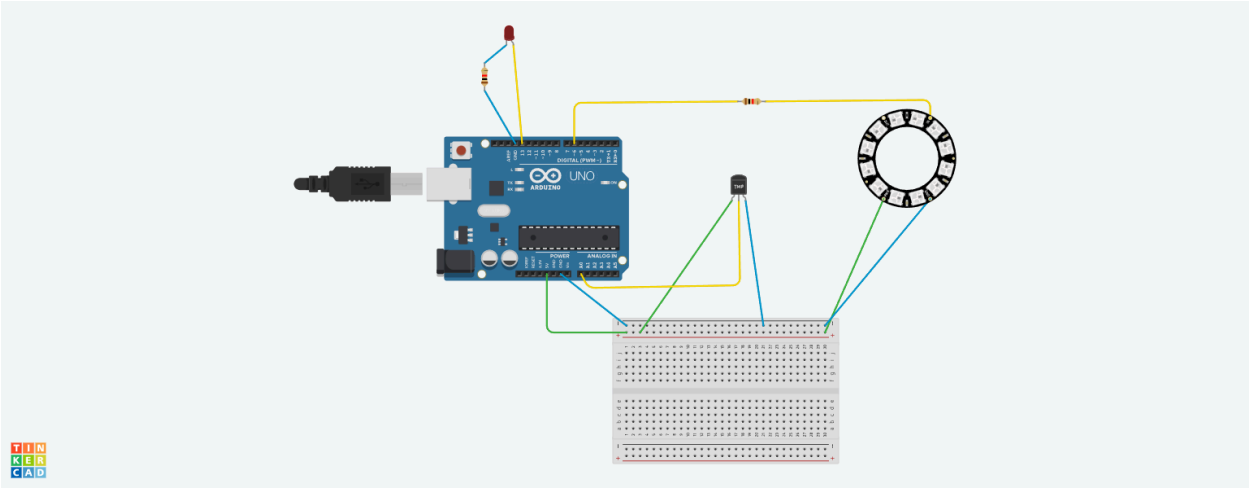
Link to tinker

Code:

```
// (C) Jitish Rajankumar Padhya, Raghav Tengse, Utkarsh Singh, group: 23 (2024)
// Work package 4
// Exercise 2:
// Submission code: 2351823

#include <Servo.h>

Servo myServo;  // servo object to control motor
int ServoOverFlow = 0;  //counts the timer2 overflow for the servo
int seconds = 0;  // time counter

//interrupt function for timer2
ISR(TIMER2_OVF_vect){
//increments the servo overflow by 1,everytime an overflow occurs
  ServoOverFlow++;
}

void setup(){
  myServo.attach(5);  //this attaches the servo to pin 5
```

```
  Serial.begin(9600); //define baud rate

  // TCCR2B is an inbuilt servo variable that tracks
  // the timer inside the servo component
  //each CLOCKSELECT is toggled to reduce the cycles
  TCCR2B |= (1 << CS22) | (1 << CS21) | (1 << CS20);
  // toggle the interrupts specifically for overflow events
  TIMSK2 |= (1 << TOIE2);
  //initializes the interupt
  sei();
}

void loop(){
  //if the overflow is 61(essentially 1 second
  //it resets the servo and counts 1 second
  if (ServoOverFlow >= 61) {
    ServoOverFlow = 0; // overflow reset
    seconds++;//increments to 1 second

    //it moves the rotor by 1 degree every second
    //it has a max limit of 180 and is in sync with
      // the 0-60 second limit of the timer
    myServo.write(map(seconds % 60, 0, 60, 0, 180));

    //prints seconds
    Serial.print("Seconds: ");
    Serial.println(seconds);

    //resets timer if 1 minute has passed
    if(seconds == 10){
      seconds = 0;
    }
  }
}
```

Exercise 3



[Link to tinker]

```
// (C) Jitish Rajankumar Padhya, Raghav Tengse, Utkarsh Singh, group: 23 (2024)
// Work package 4
// Exercise 3:ADDRESSABLE LEDS
// Submission code: 2351823

//include the NeoPixel library
#include <Adafruit_NeoPixel.h>
// Pin where the TMP36 is connected
#define TEMP_SENSOR_PIN A0
// Number of pixels in the NeoPixel ring (change this to 12)
#define NUM_PIXELS 12
// Pin for the additional red LED
#define RED_LED_PIN 13
// Temperature interval for each LED
//Sets the interval of each LED
#define TEMP_INTERVAL 10
// Calculate threshold step size

//creates instance of the NeoPixel
//it specifies the number of pixels, pin of the NeoPixel,colour and data transmission speed
Adafruit_NeoPixel pixels(NUM_PIXELS, 6, NEO_GRB + NEO_KHZ800);
```

```
void setup() {
  Serial.begin(9600);
  //sets the pinmode to INPUT determining that we are going to take
the input from temp sensor
  pinMode(TEMP_SENSOR_PIN, INPUT);
  //determines the pinmode to OUTPUT determining that it will show the
output
  pinMode(RED_LED_PIN, OUTPUT);
  //initializes the NeoPixel ring
  pixels.begin();
}

void loop() {
  //reads the temperature in celsius
  float tempC = readTemperature();
  //checks if the temperature is valid
  if (tempC != -100) {
    displayTemperature(tempC);
    //if not valid print the following statement
  } else {
    Serial.println("Error: Could not read temperature.");
  }
  Serial.print("Temperature: ");
  Serial.print(tempC);
  Serial.println(" C");
  delay(2000); // Delay between temperature readings
}

float readTemperature() {
  // Read analog voltage from temperature sensor
  int sensorValue = analogRead(TEMP_SENSOR_PIN);
  // Convert analog voltage to voltage (0-5V)
  float voltage = sensorValue * (5.0 / 1023.0);
  // Convert voltage to Celsius temperature
  float tempC = (voltage - 0.5) * 100.0;
  //return the temperature
  return tempC;
}
```

```
void displayTemperature(float tempC) {
  // Calculate the number of LEDs to light up based on the temperature
  int numLedsToShow = calculateNumLeds(tempC);
  // Loop through the LEDs and set the color to green for LEDs that
should be lit up
  for (int i = 0; i < numLedsToShow; i++) {
    // Green color for temperature indication
    pixels.setPixelColor(i, pixels.Color(0, 255, 0));
  }
  // Turn off remaining LEDs by setting their color to black (off)
  for (int i = numLedsToShow; i < NUM_PIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));//(0, 0,
0)indicates black color
  }
  // Update the NeoPixel ring to reflect the changes
  pixels.show();
  //if the the number LEDs is greater than or equal to 12 the RED LEDs
will be turned on
  if (numLedsToShow >= NUM_PIXELS) {
    digitalWrite(RED_LED_PIN, HIGH);
  } else {
    //if not greater than or equal to 12 the RED LED will remain off
    digitalWrite(RED_LED_PIN, LOW);
  }
}

//
int calculateNumLeds(float tempC) {
  int numLeds = tempC / TEMP_INTERVAL;
  // Ensure numLeds does not exceed NUM_PIXELS
  //min function returns the min value of the two.
  return min(numLeds, NUM_PIXELS);
}
```